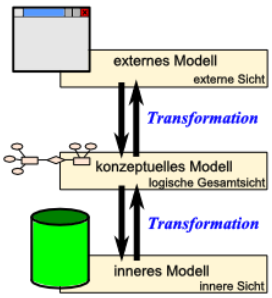
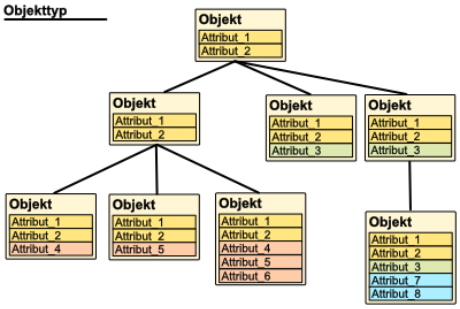
Informatik

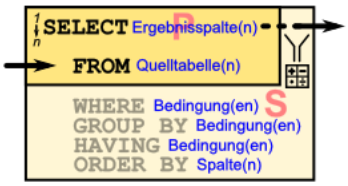
für die Sekundarstufe II

- Datenbanken -

Autor: L. Drews



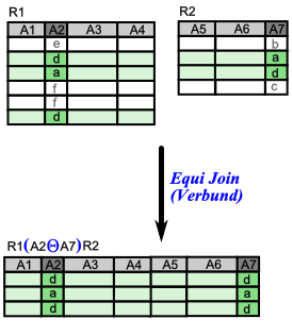




ACCESS

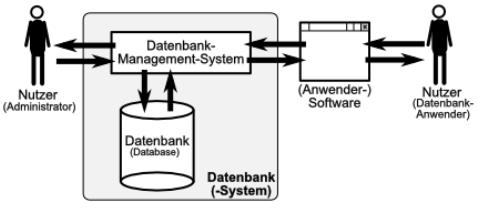
AND | OR | XOR

BASE



Wie jagt ein typischer Programmierer afrikanische Elefanten?

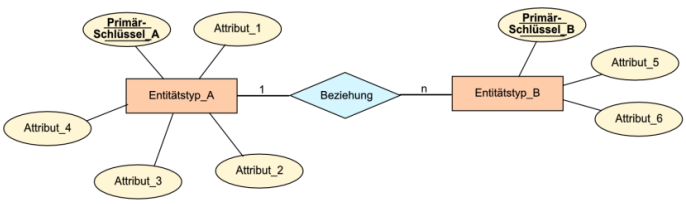
```
{
Gehe nach Afrika
Beginne am Kap der guten Hoffnung
Durchkreuze Afrika von Süden nach Nor-
den bidirektional in Ost-West-Richtung
Für jedes Durchkreuzen tue
   {
   Fange jedes_Tier, das Du siehst
   Vergleiche jedes_Tier mit Elefant
   Halte an bei übereinstimmung
   }
}
```



SQL
NoSQL

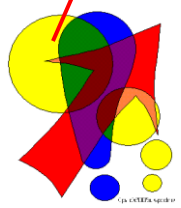
Wie jagt ein SQL-ler afrikanische Elefanten?

```
SELECT elefant
FROM afrika
```



V. 0.9k (2022)

Legende:
mit diesem Symbol werden zusätzliche Hinweise, Tips und weiterführende Ideen gekennzeichnet



Nutzungsbestimmungen / Bemerkungen zur Verwendung durch Dritte:

(1) Dieses Skript (Werk) ist zur freien Nutzung in der angebotenen Form durch den Anbieter (lern-soft-projekt) bereitgestellt. Es kann unter Angabe der Quelle und / oder des Verfassers gedruckt, vervielfältigt oder in elektronischer Form veröffentlicht werden.

(2) Das Weglassen von Abschnitten oder Teilen (z.B. Aufgaben und Lösungen) in Teildrucken ist möglich und sinnvoll (Konzentration auf die eigenen Unterrichtsziele, -inhalte und -methoden). Bei angemessen großen Auszügen gehört das vollständige Inhaltsverzeichnis und die Angabe einer Bezugsquelle für das Originalwerk zum Pflichtteil.

(3) Ein Verkauf in jedweder Form ist ausgeschlossen. Der Aufwand für Kopierleistungen, Datenträger oder den (einfachen) Download usw. ist davon unberührt.

(4) Änderungswünsche werden gerne entgegen genommen. Ergänzungen, Arbeitsblätter, Aufgaben und Lösungen mit eigener Autorenschaft sind möglich und werden bei konzeptioneller Passung eingearbeitet. Die Teile sind entsprechend der Autorenschaft zu kennzeichnen. Jedes Teil behält die Urheberrechte seiner Autorenschaft bei.

(5) Zusammenstellungen, die von diesem Skript - über Zitate hinausgehende - Bestandteile enthalten, müssen verpflichtend wieder gleichwertigen Nutzungsbestimmungen unterliegen.

(6) Diese Nutzungsbestimmungen gehören zu diesem Werk.

(7) Der Autor behält sich das Recht vor, diese Bestimmungen zu ändern.

(8) Andere Urheberrechte bleiben von diesen Bestimmungen unberührt.

Rechte Anderer:

Viele der verwendeten Bilder unterliegen verschiedensten freien Lizenzen. Nach meinen Recherchen sollten alle genutzten Bilder zu einer der nachfolgenden freien Lizenzen gehören. Unabhängig von den Vorgaben der einzelnen Lizenzen sind zu jedem extern entstandenen Objekt die Quelle, und wenn bekannt, der Autor / Rechteinhaber angegeben.

public domain (pd)	Zum Gemeingut erklärte Graphiken oder Fotos (u.a.). Viele der verwendeten Bilder entstammen Webseiten / Quellen US-amerikanischer Einrichtungen, die im Regierungsauftrag mit öffentlichen Mitteln finanziert wurden und darüber rechtlich (USA) zum Gemeingut wurden. Andere kreative Leistungen wurden ohne Einschränkungen von den Urhebern freigegeben.
gnu free document licence (GFDL; gnu fdl)	
creative commens (cc) creative commons	BY: od. neu … Namensnennung … nichtkommerziell … in der gleichen Form … unter gleichen Bedingungen

Die meisten verwendeten Lizenzen schließen eine kommerzielle (Weiter-)Nutzung aus!

Bemerkungen zur Rechtschreibung:

Dieses Skript folgt nicht zwangsläufig der neuen **ODER** alten deutschen Rechtschreibung. Vielmehr wird vom Recht auf künstlerische Freiheit, der Freiheit der Sprache und von der Autokorrektur des Textverarbeitungsprogramms microsoft ® WORD ® Gebrauch gemacht.

Für Hinweise auf echte Fehler ist der Autor immer dankbar.

Inhaltsverzeichnis:

Seite

0. Einleitung

Skript enthält diverse Wiederholungen. Das ist zum Einen der Entstehungs-Geschichte(n) geschuldet und zum Anderen dient es dem flexiblen Einsatz-Konzept. Die Erfahrung sagt auch, das Wiederholungen selten schaden.

Verweise auf andere Skripte dieser Reihe 🕮 **Rechner, Netzwerke und Protokolle**

Eine weitere Möglichkeit zur Erschließung des Thema's "Datenbanken" ist ein Kurs beim Portal OpenHPI (→) des Hasso PLANTNER-Instituts. Der Kurs ist 2013 gelaufen und steht derzeit immer noch als Lese-Version zur Verfügung. D.h. Sie können den Kurs benutzen, die Virdeo's anschauen und die Quize lösen. Leider gibt es keinen Zugriff mehr auf die Hausaufgaben und die abschließende Testung. Aber das wir in Ihrem Kurs vielleicht auch vom Kursleiter organisiert und in vielleicht auch in einer Klausur enden.
Auch die Lern-Plattform AppCamps (→) bietet einen SQL-Kurs an. Dieser beschränkt sich aber eben auch vorrangig auf SQL. Wem es also nur um diese Datenbank-Sprache geht, kann auch dort mal vorbeischauen. Eine Absprache mit dem Kursleiter ist auch hier angebracht.
Unter der Adresse (→) findet man einen weiteren Datenbank / SQL-Kurs. Dieser ist primär in englisch, wird aber auch übersetzt angeboten. Da es sich hierbaei um eine automatische Übersetzung handelt, sollte man sich nicht so sehr auf Ausdruck und Grammatik fixieren. Inhaltlich ist der Kurs anspruchsvoll.
Jeder möge den für am besten geeigneten Weg nutzen, um seine Bildungs-Ziele zu erreichen. Dieses Skript bietet einen breiten, weit gefächerten Zugang zum Thema. Dabei sind die vielen Wiederholungen und Anspielungen oder Verweise als Hilfestellung zu verstehen. Wer es konzentriert und ohne stetige Wiederholungen braucht, der muss springend lesen oder zu einem anderen Material greifen.

Um dem unvorbereiteten Leser ein wenig Orientierung zu geben, was für ihn interessant bzw. notwendig ist, sind die Abschnitte mit Niveau-Kennungen versehen.
Entweder handelt es sich um Bereich von bis oder um Einzel-Festlegungen.
Die Niveau-Stufen sind von mir folgendermaßen gekennzeichnet:
Das Grundlagen-Niveau ist für alle Leser gedacht. Hier werden allgemeine Sachverhalte vorgestellt, die auch nicht immer zwangsläufig zur Informatik gehören.

Im Basis-Niveau betrachten wir die Dinge, die man einem Fach oder Grundkurs zuordnen würde. Ich orientiere mich dabei auch am Kern-Curriculum für Berlin, Brandenburg und Mecklenburg-Vorpommern. Ev. passt es genau so auch in anderen Bundesländern? Dabei können einzelne Themen mehr oder weniger ausgefüllt werden. Vieles liegt im Ermessen des Kurs-Leiters oder der Planung in der Schule.
Für die Leistungskurse oder das Hauptfach sind die Inhalte mit dem Fortgeschrittenen-Niveau. Auch hier gilt, dass ohne weiteres Gebiete aus dem untergeorneten Niveau schon die Grenze des Lehrplans sind, aber es könnten auch Themen des – von mir subjektiv eingestuften - Experten-Niveaus inhaltlich vorgeschrieben sein.

Der Kurs-Leiter sollte immer eine auf die Bedingungen und Anforderungen zugeschnittene Themen-Auswahl vornehmen. Einige deutlich weitergehende Themen sind als Experten-Niveau klassifiziert.
Wer das Skript als Wiederholung / Vorbereitung für das Studium benutzt, muss natürlich explizit auf die Anforderungen des Lehrstuhls achten. Sonst könnte es sein, dass man mit seinem Niveau deutlich unter den Forderungen liegt.
Experten-Themen eigenen sich auch für Projekte, Grob-orientierungen für Schüler-Vorträge usw. usf. Wen sie nicht interessieren, der überspringt sie einfach. Man muss nicht alles wissen, man muss nur wissen, wo es steht und wo man sich ev. wieder belesen kann.

Links:
http://home.f1.htw-berlin.de/scheibl/DB/index.htm

nur für den internen Gebrauch!:

https://books.google.de/books?id=5HXM-lFIXXcC

1. Datenbanken – Erkundungen

Problem-Fragen für Selbstorganisiertes Lernen

Was sind Datenbanken im informatischen Sinn?
Kommen wir überhaupt mit Datenbanken in Berührung?

Was macht man heute mit Datenbanken?
Welchem Zweck dienen Datenbanken? Wozu braucht man unbedingt elektronische Datenbanken?

Sind alle Datenbanken gleich aufgebaut?
Was sind relationale Datenbanken?
Gibt es andere Datenbank-Strukturen?
Welche Datenbank-Struktur ist die Beste?

Wie kann man mit Datenbanken kommunizieren?

Kann man mit Tabellen und Daten rechnen?
Was ist eine Relationen-Algebra?

Was ist / was soll SQL?
Kann man sich selbst Datenbanken anlegen?

Was macht ein Datenbank-Administrator?

Welchen Datenschutzrechtlichen Bedenken / Sachverhalte müssen beim Nutzen von Datenbanken beachtet werden?

Definition(en): Datenbank (allgemein)
Eine Datenbank ist eine Informations-Sammlung.
Datenbanken sind strukturierte Sammlungen von Daten zu bestimmten Verwendungszwecken.

Aufgaben:

1. *Erstellen Sie sich mit einem entsprechenden Programm eine Mindmap zum Thema "Datenbanken"!*
2. *Notieren Sie Fragen und Inhalte, die aus Ihrer persönlichen und aus informatischer Sicht zu klären sind!*
3. *Welche Datenbanken bzw. Datenbank-Systeme kennen Sie? Was verbinden Sie mit diesen Datenbanken bzw. Datenbank-Systemen?*

allgemeine Anforderungen an Datenbanken / Datensammlungen:

- die Daten müssen persistent (dauerhaft) gespeichert werden
- die Daten müssen aktualisierbar sein
- die Daten müssen nach (verschiedenen) Elementen durchsuchbar sein
- die Daten sollten so gespeicher werden, dass möglichst wenige Dopplungen (Redundanzen) auftreten
- die Daten sollten möglichst vielen Nutzer – auch gleichzeitig – zugänglich sein

Aufgaben:

1. *Suchen Sie sich drei Personen aus möglichst verschiedenen Gruppen heraus und ermitteln Sie welche "Datenbanken" sie besitzen / benutzen!*
2. *Suchen Sie sich drei Personen aus möglichst verschiedenen Alters-Gruppen (ersatzweise: aus unterschiedlichen Arbeits-Welten) heraus und analysieren Sie, wie diese Ihre Adress- / Kontakt-Daten sammeln und strukturieren!*

Aufgaben

1. *Wählen Sie sich eine der nachfolgenden Datenbanken aus und sammeln Sie Hintergrund-Informationen zu den angegebenen Schwerpunkten: (weitere Datenbanken in Absprache mit dem Kursleiter möglich!)*
   - *a) allgemeine Informationen zur Datenbank*
   - *b) die Datenbank aus informatischer, ökonomischer und politischer Sicht*
   - *c) Art und Umfang der gespeicherten Daten (auch historische Entwicklung)*
   - *d) Nutzungs-Möglichkeiten der Datenbank (Zugänge, Schnittstellen)*
   - *e) notwendige Angaben (Eingaben) für die Nutzung; was passiert im Hintergrund*
   - *f) Art und Weise der Daten-Ausgabe (Alternativen)*
   - *g) Vorstellung der Webseite, App bzw. des Dienstes (Nutzungs-Praxis)*
   - *h) Absicherung der Daten / Datenschutz / Umgang mit Problemen und Pannen (Skandale)*

Bibliothek	Verkehrszentralregister	wikipedia
dwd.de/klimadaten	check24.de	texxas.de/tv
flickr.com	EAN-Auskunft (gepir.de)	youtube.com
Einwohner-Register (Einwohner-Meldeamt)		Schufa
Fahrplan-Auskunft (z.B. Deutsche Bahn)		Flug-Buchungs-System
onmeda.de/icd-10/icd10-diagnoseschluessel.html		de.statistica.com
AOK-Krankenhausnavigator		Hotel-Buchungs-System
www.cia.gov/library/publications/resources/the-world-factbook/		
Warenwirtschaft-System eines Betriebes / Händlers		Google
online-Reisebüro	Gelbe Seiten	online-Telefonbuch
odlinfo.bfs.de	chefkoch.de	preissuchmaschine.de
Denic	wetter.de	verkehrsinfo.de
facebook.com	twitter.com	pixabay.com

2. *Erstellen Sie eine Mindmap (als informative Gedankenkarte) oder ein Informationsblatt (A4) und zusätzlich eine kurze PowerPoint-Präsentation oder ein WebQuest zur gewählten Datenbank! Das Produkt muss den anderen Kursteilnehmern in digitaler Form zur Verfügung gestellt werden! Bereiten Sie sich auf eine Präsentation vor den anderen Kursteilnehmern vor!*

Beispiele für Datenbank (grobe Typisierung)

Typ / Gruppe	Beispiele / …
- **Lexika**	Papyrus-Sammlung der großen Bibliothek zu Alexandria MEYERS Lexikon in 18 Bänden DUDEN – deutsche Rechtschreibung de.wikipedia.org Encyclopedia Britannica Wörterbuch Brockhaus Enzyklopädie Bertelsmann Lexikothek microsoft Encarta BREHMS Tierleben BEILSTEINS Handbuch der Organischen Chemie Der Große Ploetz PSCHYREMBEL Arzneibuch MERCK's Warenlexikon Liederbücher
- **Personen-Karteien** **Personen-Register**	Daten der Ortsämter Fahndungs-Karteien der Polizei usw. usf.
- **Register für Firmen und Institutionen**	Firmen-Register (Handels-Register) Vereins-Register
- **Warenwirtschaftsysteme**	SAP
- **Bilderdatenbanken**	www.flickr.com www.instagram.com www.pixabay.com
- **Multimediadatenbanken**	www.youtube.com commons.wikimedia.org
- **Personendatenbank**	Schulverwaltungsprogramme (WinSchool, Magellan, …)
- **Soziale Medien** **sociale media**	flickr.com facebook.com youtube.com twitter.com
-	

Orientierungs-Beispiel:

Bücher-online-Händler

Nutzer-Aktionen	Server-Aktionen
1. Besuch der Webseite	1. Cookie auslesen Nutzerdaten lesen personalisierte Seite zeigen
2. Suche nach Buch / Autor / Titel / …	2. Produkt-Index durchsuchen Such-Ergebnisse anzeigen
3. Listen oder Details ansehen	3. Such-Ergebnisse aktualisieren
4. Buch in den Warenkorb legen	4. Nutzer-Warenkorb aktualisieren weitere Produkte vorschlagen Buch in Datenbank reservieren
5. Bestellung aufgeben	5. Verkauf beginnen
5a. Adresse angeben / vergleichen / aktualisieren	5a. Kundendaten aktualisieren Liefer-Adresse festlegen
5b. Kauf bestätigen	5b. Verkauf beenden
5c. Zahlungs-Art festlegen	5c. Transaktion abwickeln Lager-Bestand aktualisieren ev. Nach-Lieferung aufgeben
6. …. warten … warten … warten …	6. Verpackung und Versand initialisieren
	6a. Paket-Info vom Paket-Dienst übernehmen
	6b. Versand-Info verschicken
7. Buch-Paket in Empfang nehmen	7. Liefer-Info vom Paket-Dienst übernehmen (und Vorgang abschließen)

nach: NAUMANN OpenHPI-Kurs SQL (geändert: dre)

Leistungen des Dienstleisters / Datenbank / Website zur Datenbank:

- Anzeige einer Webseite
- Such-Funktionen (Suche nach Buch, Autor, Titel, ISBN, Preis, …)
- sortierte Listen
- detaillierte Informationen / Anzeigen
- Kaufen eines Buches
- Anzeigen von Angeboten, Werbung, Kauf-Vorschlägen

Besonderheiten:

-

Informationen zum technischen System:

- verfügbare Bücher:
- verfügbare …:
- Größe der Datenbank:
- Reaktionszeit:
- …

die größten Datenbanken der Welt (pauschal; ohne klares Ordnungs-Kriterium)

Stand vor 2011! z.T. nur geschätzt

- **US Library of Congress**	**Archiv / Bibliothek** 20 TB Text; 29 Mill. Bücher (+10'000 / d); 5 Mill. digitale Dokumente 130 Mio. Einträge (Bücher, Fotos, Karten, …)
- **CIA**	**Geheimdienst / Informationsdienst** unbekannt zugänglich ist u.a. "the World Factbook" (→ https://www.cia.gov/library/publications/resources/the-world-factbook/) 100 Artikel pro Tag dazu
- **amazon**	**Handelsplattform** 42 TB 60 Mio. Kunden, 250'000 Bücher
- **youtube**	**Videoportal** 45 TB 100 Mill. Video's (+ 65'000 / d)
- **LexisNexis / ChoicePoint**	**Telefon- und (Personen-)Daten-Auskunft)** 250 TB Personen-bezogene Daten Informationen über 250 Mill. Menschen
- **Sprint ()**	**Mobilfunk-Anbieter** unbekannt 55 Mio. Kunden, 365 Mio. Anrufe pro Tag, 2'850 Mrd. Textzeilen (Nachrichten / SMS) Spitzenwert: 70'000 Detail-Informationen über Telefon-Verbindungen pro Sekunde
- **google**	**Suchmaschine** unbekannt 33'000 Mrd. Personen-bezogene Einträge
- **AT&T**	**Telekommunikations-Konzern** 323 TB 1'900 Mrd. Zeilen (Verbindungsdaten)
- **SAP ERP**	**Wirtschaft- und Personal-Software** 67'000 Tabellen mit 700'000 Spalten; 10'000 Sichten (View's); 100 Mio. Zeilen Initialisierungs-Größe 57 GB

- **NERSC**
  National Energy Research Sientific Computing Center
  Forschungs-Rechnernetz und -Datenbank
  2,8 PB
  2'000 Computer-Wissenschaftler und Sachbearbeiter

- **World Data Centre for Climate (WDCC)**
  Internationales Klimadaten-Zentrum
  330 TB + 6 PB (auf Magnet-Bändern)
  220 TB Web-Daten

- **NSA**
  Geheimdienst
  unbekannt
  Zuwachs pro Tag:

Q: u.a.: SQL-Kurs (NAUMANN; OpenHPI; 2013); https://www.comparebusinessproducts.com/fyi/10-largest-databases-in-the-world (2010)

- **Deutsche Klimadatenbank**
  Klima-Datenbank / Klima-Simulation
  10 PB
  Zuwachs pro Tag: 1 Mill. Wetterdaten

- **DeNIC**
  Deutsches Netzwerk Informations-Zentrum
  Verwalter der de-Internet-Domain's
  1997: 105'000 Domain's; 2016: 10 Mill.; 2018: > 16 Mill.

-

-

-

-

1.1. von der Karteikarte zur Datenbank

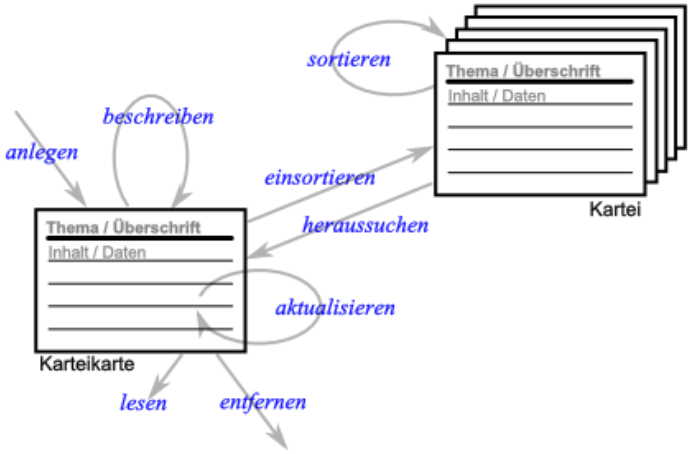
Problem-Fragen für Selbstorganisiertes Lernen

Informationen sammeln, verarbeiten und in geeigneter Form darstellen ist eine der alten Fähigkeiten des Menschen. Wann das genau angefangen hat, ist wohl sicher nicht mehr zu klären. Genau so ungewiss ist die Zukunft der Informations-Verarbeitung.
Vielleicht gehörten die Höhlen-Malereien zu den den ersten Informations-Sammlungen der Menschheit. Sie stellten vielleicht die ersten Datenbanken über jagbare Tiere und geeignete Jagd-Techniken dar.
Mit den Bild- und Schriftsprachen und dem Notieren von Informationen auf Ton, Papyrus oder Papier begann auch die Zeit der Bibliotheken. Diese waren nichts anderes als Datenbanken.
Sinnbildlich können Notiz-Zettel und –Blöcke (z.B. von Dedektiven) oder Skizzen-Blöcke (z.B. von Künstlern) als spezielle Datenbanken gesehen werden. Die großen Archive, Galerien, Museen und Biblioteken gehören sicher zu den größten nicht-elektronischen bzw. nicht-digitalen Datenbanken.

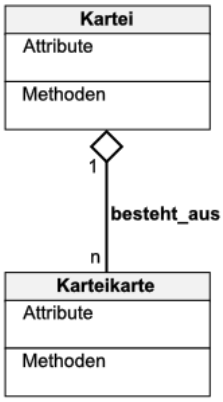
1.1.0. Karteikarten, Karteien und Register

Bei Datenbanken einen Anfang zu bestimmen, scheint fast unmöglich. Zumindestens, wenn man sich Datenbanken als Sammlungen von Daten vorstellt und sie nicht unbedingt mit der elektronischen Daten-Verarbeitung verknüpft.
Gerade Bibliotheken verfügten früher über mehrere Systeme der Informations-Sammlung über ihre Bücher. Mit Hilfe verschiedener Register konnte man Bücher nach Autor, Titel und Themenbereich suchen. Für jedes Buch war in einem Register mindestens eine passende Karteikarte angelegt. Auf einer Karteikarte wurden dabei alle Informationen zu einem Buch notiert. Dazu gehörten auch noch zusätzliche Daten, wie Verlag, Auflage, Seitenzahl usw. usf. Je nach Register waren die Karteikarten aber mit unterschiedlichen Überschriften versehen.
Für ein beliebiges Buch gab es z.B. drei Karteikarten mit dem gleichen Inhalt – aber mit unterschiedlichen Überschriften. Eine Karteikarte für das Autoren-Register mit dem Autor als Überschrift und entsprechend eine Karte mit dem Titel als Überschrift für das Titel-Register.
Die dritte Karteikarte wurde mit dem Thema versehen. Dafür benutzte man definierte Fachbereiche, Themen und Stichworte, damit das Register übersichtlich blieb.
Karteikarten stellen quasi eine Modellierung des Buches für das Register dar. Das Real-Buch wird durch die Karteikarte als informatisches Objekt repräsentiert. Aus informatischer Sicht interessieren uns drei Dinge zu einer Karteikarte bzw. zu einem Buch.
Zuerst ist das der **Name** – die Überschrift – der Karteikarte. Sie ist der Objekt-Name in einem der Register.
Als Nächstes interessieren uns die Informationen zum Buch. Das sind als die bibliographischen Angaben

und der Standort in der Bibliothek.
Diese Informationen, Eigenschaften und Merkmale nennen wir in der Informatik **Attribute**.
Der dritte Bereich sind die möglichen und notwendigen Tätigkeiten, die wir mit den Karteikarten durchführen können. Wir sprechen allgemein von den **Methoden** zu den Objekten.



In einem UML-Diagramm (UML = Unified Modeling Language (vereinheitlichte Modellierungs-Sprache)) sieht die Struktur recht einfach aus. Das Modell beinhaltet die Klassen (Objekt-Typen) **Kartei** und **Karteikarte**. Sowohl die Karteikarte – als auch die Kartei – besitzen eigene Attribute und Methoden.



Eine (**1**) Kartei **besteht aus** beliebig vielen (**n**) Karteikarten. Eine leere Karte besitzt dann eben keine einzige Karteikarte.

Besonders anspruchvolle Register haben an den Rändern der Karteikarten ein Lochungs- und Kerben-System. Mit Hilfe der dort verschlüsselten Daten konnte man große Karten-Stapel recht effektiv nach den (nicht) "eingekerbten" Informationen durchsuchen.

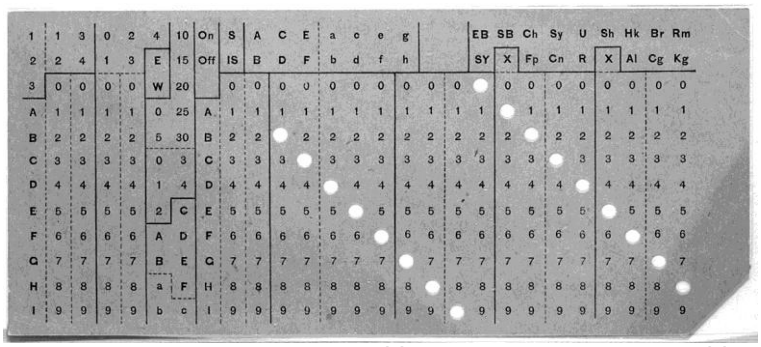
Register sind typische stationäre Datenbanken. In den meisten Fällen gibt es auch nur ein Original. Das Anlegen von Kopieren oder das Führen gleicher Register in verschieden Standorten wurde kaum realisiert, weil der organisatorische Aufwand enorm war. Eher führte man nur Teil-Register in den einzelnen Fach-Bibliotheken.

Die Anfänge der größeren mechanischen (! noch nicht elektrischen oder elektronischen) Daten-Verarbeitung, bei der es auch um große Datenmangen ging, war vielleicht die Lochkarten-Maschine von Herman HOLLERITH (1860 – 1929).

Für die Volkszählung 1890 in den USA entwickelte er eine Datenverarbeitungs-Maschine nach der Idee der programmierbaren Webstühle.

Die Daten der ausgezählten Personen wurden auf den Papp-Lochkarten eingestanzt - praktisch codiert.

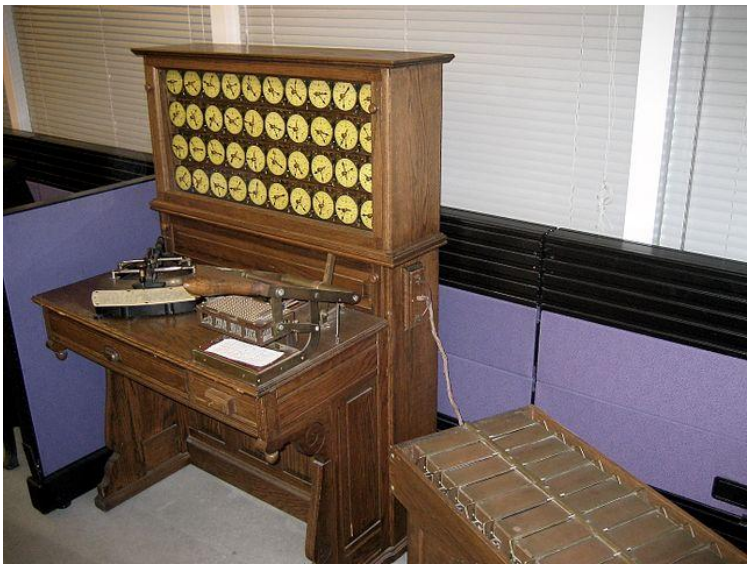
Diese Lochkarten konnten dann sortiert und nach bestimmten Kriterien durchgezählt werden.



Lochkarte zur HOLLERITH-Maschine
Q: de.wikipedia.org (US Library of Congress)

Solche Lochkarten – in universellerer Form - kamen noch bis in die 70er Jahre des 20. Jahrhunderts zur Anwendung. Hier dann aber in elektrischen oder elektronischen Maschinen.

Die HOLLERITH-Maschine verkürzte den Auswertungs-Aufwand für eine Volkszählung bei gleichen Personal-Bedarf von 7 auf 2 Jahre. Zum Einsatz kamen 43 Maschinen, die HOLLERITH der Regierung der USA nur lieh. Aus seiner dafür 1886 gegründeten Firma "Tabulating Machine Company" entstand dann 1924 IBM (**I**nternational **B**usiness **M**achines).



HOLLERITH-Maschine mit Zählwerk (an der Wand),
Lochkarten-Stanzer auf dem Tisch
und einem Lochkarten-Sortierer (rechts vorne)
Q: de.wikipedia.org (Adam Schuster (www.flickr.com))

ZUSE mit seiner Zuse 3 erste elektrische Rechenmaschine; basierend auf Relais; Größe ungefähr wie ein Kleiderschrank
Rechenkapazität ungefähr so groß wie ein heutiger einfacher Schul-Taschenrechner
wenig permanent-Speicher
erste Festplatte; von IBM ("IBM 350 RAMAC"); 3,75 MB auf 50 übereinander gestapelten Metall-Platten, auf die mit einem Schreib-Lese-Kopf Daten geschrieben bzw. gelesen wurden;
nur für Groß-Rechner verfügbar
Winshester-Platten

große Daten-Speicher waren Voraussetzung für die Entwicklung der Datenbank-Technologie im Bereich der elektrischen Daten-Verarbeitung
große Dten-Mengen lassen sich mit klassischen Programmen nicht mehr verarbeiten, dazu bedarf es der speziellen Datenbank-Technologien (diese Ebene wird uns aber wegen ihrer extrem technischen Seite hier im Kurs kaum begegnen).

Festpaltten groß und schwer, wie ein Ziegelstein
30 MB kosteten zu Anfang rund 12 bis 15'000 DM (pro Stück!)

danach Entwicklung nach dem MOOREschen Gesetz; danach verdoppelt sich die Speicher-Kapazität ungefähr alle 2 Jahre
derzeit geht Entwicklung noch schneller voran, als Regel aber gut verwendbar und eigentlich war es immer nur eine Regel (die ungefähr stimmte)

Aufgaben:

1. *Jeder Kursteilnehmer wählt sich jeweils 3 Bücher aus und ermittelt dazu die bibliographischen Angaben!*
2. *Vereinbaren Sie im Kurs ein Schema für Notierung der bibliographischen Angaben auf einer Karteikarte!*
3. *Legen Sie für jedes Buch soviele Karteikarten an, dass drei Register (Autoren-, Titel- und Themen-Register) aufgebaut werden können! Geben Sie den Karteikarten passende Überschriften für die register!*
4. *Legen Sie drei sortierte Register an!*

für die gehobene Anspruchsebene:

5. *Entwickeln Sie ein Lochungs- und Kerbungs-System, mit dem man die Karteikarten danach sortieren / durchsuchen kann, wer sie (von den Kursteilnehmern) angelegt hat!*

Daten – Was ist das eigentlich?

Mit dem Begriff Daten verbinden wir schnell auch die Begriffe Informationen, Nachrichten, ???. Einiges haben wir dazu schon abgeklärt (→).
Aber was sind Daten im informatischen Sinne. Hier sind Daten digitale Repräsentationen von Dingen, Beziehungen oder Prozessen aus der realen Welt.
Allgemein nennen wir die Real-Dinge auch Entitäten. Im informatischen Sinne können aber Beziehungen und Prozesse im Sinne von Daten als Entitäten aufgefasst werden.
Zwischen Entitäten gibt es i.A. Beziehungen, die wir allgemein Relationship's nennen.

Definition(en): Daten
Daten sind digitale Repräsentanten von Objekten, Beziehungen und Vorgängen in einem Informations-verarbeitenden System. .
Daten (Einzahl: Datum) sind interpretierbare / verarbeitbare / kommunizierbare Darstellungen von Informationen in formaler Form.
Daten sind Zeichen und Symbole, die Informationen zum Zwecke einer Verarbeitung darstellen.
Daten sind Gebilde aus Zeichen oder Funktionen, die aufgrund von Vereinbarungen Informationen darstellen, als solche verarbeitet und angezeigt werden (können).

In informatischen Systemen und besonders in Datenbanken interessieren uns die folgenden Kern-Fragen:

- **Welche Daten werden gespeichert?**
  Welche Daten müssen gespeichert werden? — Auswahl notwendiger und sinnvoller Daten
- **Wie werden die Daten gespeichert?** — Datenträger; permanent oder temporär
- **Wie kommt man an die Daten wieder heran?**
  Wie können die Daten genutzt werden? — SQL als Datenbank-Anfrage-Sprache
- **Wie arbeitet man sicher und effizient?** — effektive und geprüfte Algorithmen

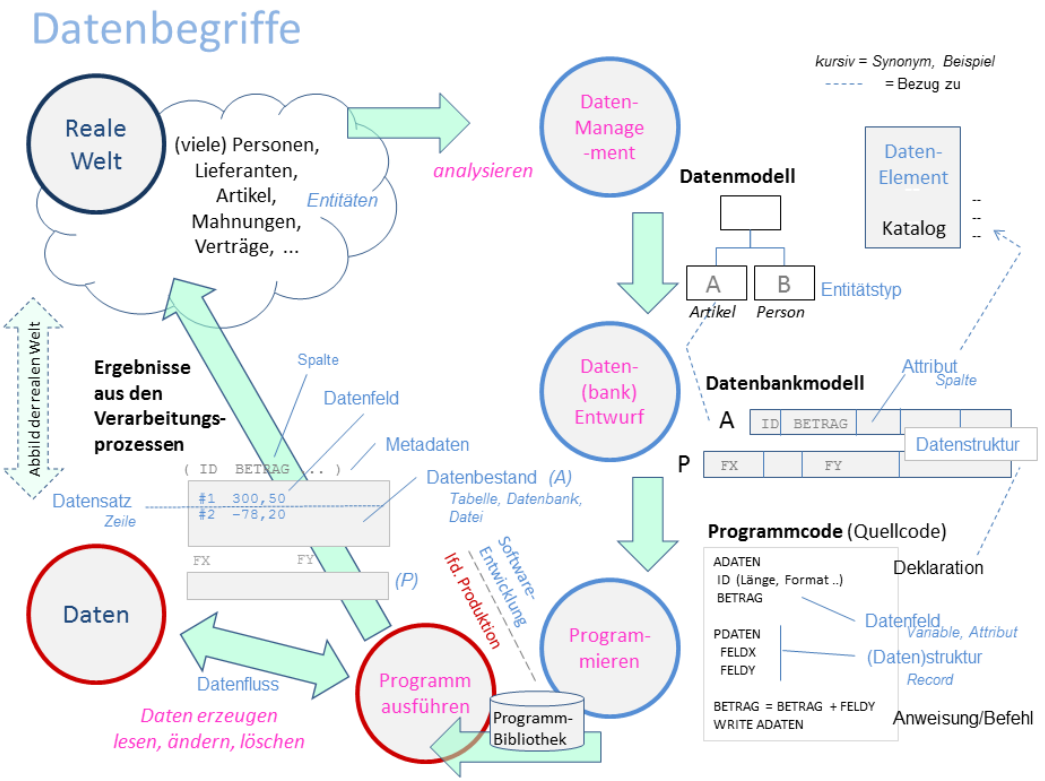
Daten lassen sich kategorisieren:

nach der Höhe der Komplezität:

- **unstrukturierte Daten**
  nicht-strukturierte Daten — beliebige, einfache Texte; Graphiken
- **semi-strukturierte Daten** — XML-Dateien (XML .. Extensible Markup Language)
- **strukturierte Daten** — Datenbanken, Dateien

nach der Art der Beständigkeit:

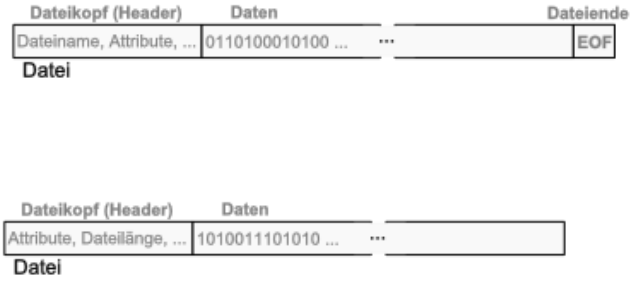
- **permanente Daten (persistent, gespeichert)** — dauerhaft gespeicherte Daten (z.B.: Betriebssystem, Boot-System, Dokumente, Bilder, …)
- **temporäre Daten (Fließ-Daten, Daten-Ströme)** — zur zeitweilig gespeicherte oder existierende Daten (z.B.: Streaming-Daten; Ein- und Ausgabe-Puffer, …)



Entstehung von Daten und dazugehörende Begriffe
Q: de.wikipedia.org (VÖRBY)

1.1.1. Daten in Dateien

Spätestens seit den Disketten-orientierten Betriebssystemen wie SCP und ms-DOS (microsoft ® Disk Operation System) und Großrechnern mit Festplatten werden Daten in Dateien gespeichert. Daten werden dabei als Sequenz (Band-artig) hintereinander auf dem Datenträger gespeichert.
Historisch kann man diese Form der Daten-Verarbeitung ab den 60er Jahren des 20. Jahrhunderts beobachten und zieht sich bis heute durch, wobei der Anteil aber extrem gesunken ist.
Es handelt sich beim Arbeiten mit Dateien um ein naives / sehr einfaches Datenbank-Management-System (DBMS). Auch wenn der Begriff hier noch nicht wirklich getragen wird, wollen wir ihn hier schon mal benutzen, um später die Entwicklung zu einem echten DBMS aufzeigen zu können.

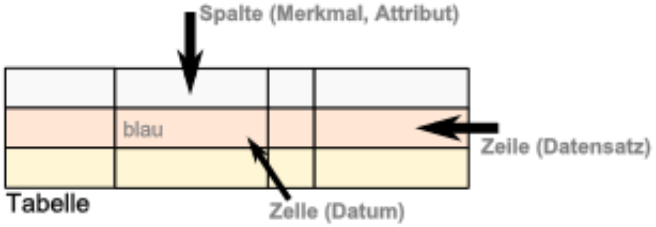


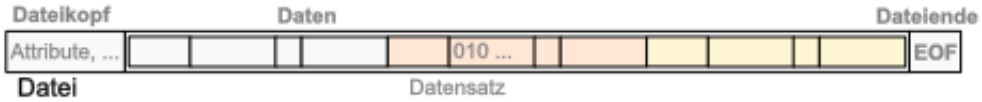
Bei Band-Laufwerken (Streamern) erfolgt das praktisch direkt. Über einen magnetischen Schreib- und Lese-Kopf wurden die Daten auf einen mit Eisen beschichtesten Folienstreifen als unterschiedliche Magnetisierung abgelegt. Das Magnetband wurde an dem Schreib-Lese-Kopf vorbeigeführt und auf Spulen auf- bzw. abgewickelt. Auch heute nutzt man die Technik, um große, wegschleißbare Datensicherungen zu erstellen. Die Geräte werden Streamer genannt.
Bei Disketten und Festplatten benutzt man statt dem Folienband Folien-Scheiben. Bei der Floppy-Diskette war sie relativ beweglich. In Festplatten benutzte man stabile Metall-, Glas- oder Kunststoff-Scheiben. Die Daten werden immer in Kreis-förmigen Spuren (Ringen) auf die magnetische Beschichtung der Scheiben geschrieben bzw. von ihr gelesen.

böse Frage zwischendurch:
Wie sind die Daten auf einer CD bzw. DVD angeordnet?

Das Speicher-Prinzip in Dateien hat sich auch bis heute nicht wesentlich geändert. Das macht auch die Flexibilität und Durchlässigkeit der verschiedenen Betriebssysteme und technischen Weiterentwicklungen (z.B. hinsichtlich der Datenträger (z.B. SSD)) möglich.

Komplexe Daten werden zum Speichern in Sequenzen umgearbeitet, was aber in den meisten Fällen für den Nutzer unbemerkt erfolgt. Dazu gehören z.B. Tabellen, Texte oder Bilder.

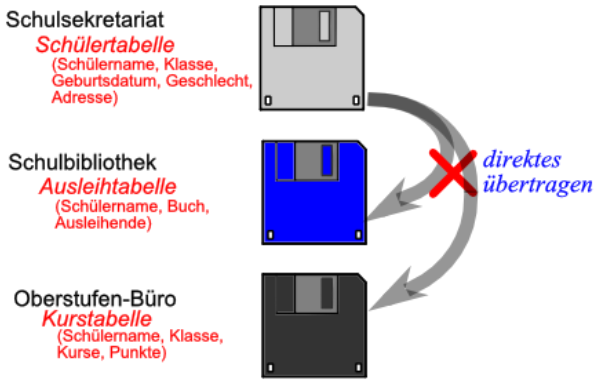




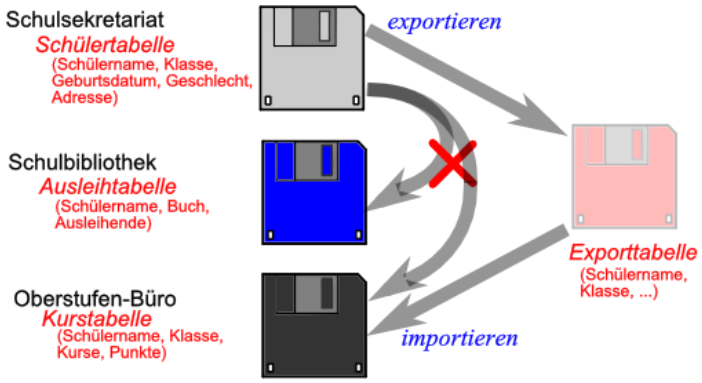
So werden die Zeilen einer Tabelle einfach hintereinander geschrieben. Je nach Datei-Typ gibt es eine feste Zeilen-Länge oder ein definiertes Zeilenende-Zeichen.
Diese Sequenzen werden dann als "Datei" auf einem beliebigen Datenträger abgelegt. Das Prinzip ändert sich auch nicht mit den modernen USB-Sticks oder SSD-Festplatten. Nur sind es hier keine magnetischen Datenträger mehr, sondern mikroelektronische Speicher-Elemente. Sie können zwischen zwei Zuständen wechseln und in diesen dann sehr lange verweilen. Die Zustände repräsentieren die Nullen und Einsen der Digital-Technik.
Nicht-lineare bzw. nicht-sequenzielle Daten, wie sie z.B. in Baum-Strukturen vorkommen, werden ebenfalls in Sequenzen umgeschrieben. Meist werden hier dann Zwischen-Felder mit Sprung-Adressen (sogenannte Pointer) in die Sequenz eingearbeitet. Über diese Pointer kann man dann zur nächsten Verzweigung springen.
Daten auf Disketten waren schon ein großer Fortschritt. Aber mit dem Austausch, dem Ändern und der Weitergabe waren diverse organisatorische Probleme gegeben.
Jeder Anwender bzw. jedes Programm verwaltete und speicherte seine Daten individuell. Dabei war meist keine Lesbarkeit der Daten ohne das Erzeuger-Programm gegeben.
Die Daten des einen Programms sind nicht mit denen eines anderen Programms kompatibel. So kann z.B. Zeitdaten. In dem einem Programm wird das Geburtsdatum vielleicht in der Form 12.03.2001 gespeichert, während es in einem andern (z.B. aus dem amerikanischen Bereich stammende) im Format 2001-03-12 gespeichert wird.

Selbst an einer Arbeitsstelle war nur eine begrenzte universelle und Bereichs-übergreifende Nutzung der Daten möglich.
Für die Programmierer bestand kein Bedarf für iregndwelche anderen Programme kompartible Daten zu erzeugen. Meist ging das auch schon aufgrund der riesigen Anzahl von Datenstrukturen auch garnicht. Die Datenstrukturen wurden auch selten veröffentlicht, denn jeder Nutzer sollte mit seinen Daten natürlich weiter beim Programmierer bleiben, statt zum Konkurrenten zu wechseln, der die Daten ebenfalls lesen kann. Bei Konkurrenten wird das andere Programm eher geschnitten, als unterstützt.

Im Normalfall ist i.A. kein (direkter) Austausch von Daten möglich. Deshalb müssen die Daten (z.B. ev. der Schülername) in jedem Programm einzeln gepflegt werden. Das ist ist sehr Zeit- und Personal-aufwändig.
Eine mögliche Datenübertragung hängt stark von den Programmierern der beteiligten Programme ab. Die Programmierer legen jeweils fest, welche Daten exportiert und welche importiert werden



Die Daten müssen / mussten von Hand von einem Programm / Computer zum anderen transportiert werden. Der Datenaustausch erfolgte z.B. über Disketten.
Viele Importe scheiterten, weil die Datenformate (z.B. Datum nicht stimmten oder andere Zeichen für die Kodierung der Daten benutzt werden. Es reichte schon der Unterschied zwischen "w" und "W" (als Geschlechts-Kodierung), um entweder nur teilweise Daten importieren zu können bzw. schon bei den ersten Daten abzubrechen, weil undefinierte Zeichen auftraten.
Es gab nur wenige allgemeingültige Standards, um Daten von einem Programm zu einem anderen zu übertragen.
In den meisten Programmen wurden und werden z.T. bis heute noch - bestimmte universelle Datei-Formate, wie TXT-, CSV- oder XML-Dateien unterstützt.
Bei den universellen Daten-Formaten waren zwar die Schreibweisen der Daten definiert. Trotzdem heisst das nicht, dass der Programmierer auch alle Daten exportiert. Häufig wurden einfach nur die elementaren Grunddaten verarbeitet.



Und selbst, wenn der Programmierer des einen Programms alle Daten exportiert hat, heisst das noch lange nicht, dass die Daten auch zum neuen Programm passen und von ihm importiert werden können.

Vorteile

- dezentrales Arbeiten (ohne weitere Maßnahmen) möglich
- es werden keine Netzwerke benötigt
- Fehler eines Nutzers haben keine Wirkungen auf andere Instanzen
- offline-Arbeiten unproblematisch möglich (meist garnicht auf online eingestellt)
- meist relativ einfache Korrektur oder Reparatur möglich
- portable Nutzung ohne weiteres möglich
- Daten sind schon an sich mehrfach vorhanden (mindestens 1x je Instanz) → Datensicherheit recht gut
- Datenschutz relativ gut (es werden nur die Daten erfasst, die wirklich gebraucht werden; kein online-Zugriff möglich; Daten nur auf einem Computer vorhanden; Zugriffe auf diesen Computer beschränkt)
- Sekretärin-Like (übersichtlich, klar, verständlich)

Sekretärin-like ist hier nicht abwertend gemeint. Sekretärinnen sind erfahrungsgemäß keine Informatiker. Sie kennen sich mit Schriftsätzen, Akten, Telefonieren usw. usf. super aus, aber wie Daten im Computer gespeichert sind, interessiert sie einen Schnurz (und das ist hier noch höflich ausgedrückt!). Wenn Sekretärinnen also die Datenstrukturen und –Prozesse verstehen, dann sind sie ähnlich (einfach), wie in ihrem Büro organisiert.

Nachteile

- Dateninkonsistenz steigt mit Anzahl der Nutzer und der Nutzungsdauer
- Daten-Synchronisation über mehrere Instanzen aufwändig
- Speicherbedarf auf jeder Instanz
- geringer Datenschutz (aus der Sicht übergeordneter Kontrollen; welche Daten werden vielleicht unberechtigt erfasst)
- Nutzerverwaltung meist sehr gering entwickelt
- Diebstahl möglich (Programm + Daten sind gemeinsam auf dem Computer vorhanden)
- keine oder nur wenig ausgeprägte Standards
- Datenübernahme (Export / Import) in anderere Programm schwierig, unmöglich oder sehr aufwändig
- da großer Speicherplatz-Bedarf vorhanden ist, gilt diese Speicher-Form von Daten als "teuer"
- üblicherweise müssen Dateien nach einer Änderung komplett neu gespeichert werden (das gilt üblicherweise auch bei einfachen Anhängungen)
- aufwändige Suche (praktisch immer von vorne nach hinten durch)
- fehlende Mehrnutzer-Fähigkeit

In den Bereich von recht universellen Daten-Verarbeitungs-Programmen (im Sinne von Datenbanken) gehören z.B. auch die Pseudo-Datenbanken, wie ms-WORKS, apple NUMBERS und ms-EXCEL sowie ähnliche Tabellenkalkulationen mit "Datenbank"-Funktionen.
Die Programme sind leicht zu bedienen und können für eine einzelne Tabelle eine effektive Nutzung ermöglich. Auch als Daten-Erfassungs- und Auf- oder Vorbereitungs-Tool sind sie sehr hilfreich. Häufig stellen die Programme sogar "Datenbank"-Funktionen bereit. Diese haben ein etwas geändertes Arbeits-Prinzip, bleiben aber einfache Tabellen(-Blatt)-Funktionen. Die Daten / Tabellen lassen sich gut sortieren und filtern. Für Diagramme oder andere Veranschaulichungen sind diese Programme meist auch gut geeignet.
Für mehrfach verknüpfte Tabellen mit komplexen Daten sind diese Programme nur rudimentär benutzbar.
(Mit Tabellen-Kalkulationen lassen sich i.A. keine Joins (Verbund-Operationen) durchführen. Eine Skalierung auf eine (deutlich) verkleinerte oder vergößerte Daten-Menge ist selten möglich. Weiterhin fehlen Funktionen zur Nutzung durch mehrere Anwender zur gleichen Zeit. Die Zuverlässigkeit liegt im Rahmen üblicher Office-Dokumente und ein Rechte-Management ist nur elementar vorgesehen.)

Definition(en): Datei
Eine Datei ist eine strukturierte Sequenz von Bit's / Byte's auf / in einem Speichermedium.
Eine Datei (engl.: file) ist ein Bestand / eine Sammlung von Daten, die zusammengehören und auf einem Datenträger gespeichert sind.

In vielen Betriebssystemen ist praktisch alles über Dateien organisiert. Dadurch erreicht man eine hohe Flexibilität und Erweiterbarkeit. Datei-basierte Systeme sind gut weiterentwickelbar und damit Zukunfts-orientiert.

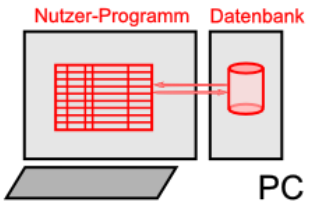
viele kleine Datensätze, viele Anfragen und Änderungen
Arbeitprinzip wird **O**n**l**ine **T**ransaction **P**rocessing (OLTP) genannt

Anwendungsbereiche

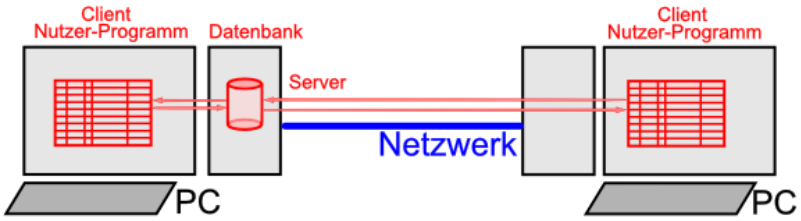
- Flug-Buchungs-Systeme
  - Reservierung (Flug, Sitz, Mahlzeit, …)
  - Personal- und Flugzeug-Verwaltung
  - Bezahl-System
  - Nutzung von vielen Nutzern simultan
- Bank-Systeme
  - Kunden, Konten, Kredite, Überweisungen
  - Geldausgabe am Automaten
- Warenwirtschafts-System (WWS)
  - Buchführung, Lagerverwaltung, Personalwesen, Steuer-Abrechnung
  - Kunden, Lieferanten, Kassen
- …

ev. noch einarbeiten: (Q: SQL-Kurs NAUMANN; OpenHPI, 2013)
Anwender- und Anwendungs-spezifische Daten-Organisation (Geräte-abhängig, redundant, inkonsistent)
Integrated Data Store (IDS) von General Electric
zum Ende der 60er Jahre tauchten dann die ersten Daten-Verwaltungs-System auf (z.B.: SAM, ISAM von IBM)
mit speziellen Dienstprogrammen ließen sich ausgewählte Daten nach bestimmten Kriterien sortieren, filtern usw.
jetzt auch Geräte-unabhängig aber immer noch Anwendungs-spezifisch
von IBM gab es "Information Management System" (IMS), deren erste Anwendung die Stück-Listen-Verwaltung zur "Saturn V"-Rakete war
einige dieser Systeme sind noch heute im Einsatz (auf aktualisierter Hardware und in aktualisierter Software-Version)
hier gilt alter Informatiker-Spruch "Never touch a running system!" ("Fasse nie ein laufendes System an!")
IBM macht heute noch rund 1 Mrd. Dollar Umsatz pro Jahr mit diesen Systemen

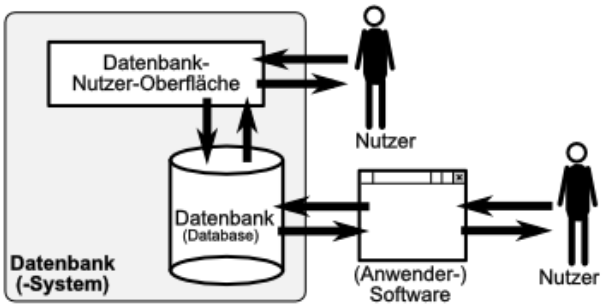
1.1.2. lokale Datenbanken



Mit lokalen Datenbanken meinen wir Systeme, bei denen die Daten-Speicherung und –Nutzung auf dem gleichen Gerät erfolgt. Es handelt meist um kleinere Datenbank-Systeme für den PC. Die Datenbank wird hier als Workstation-Version benutzt. Wenn vorhanden, dann befinden sich Client und Server auf einem Rechner.
Die Vernetzung ist möglich, aber ein gemeinsames Arbeiten mehrerer Nutzer ist eher schwierig.



Bei mehreren Nutzern spricht man dann auch immer von den Client's, die sich beim Server mit Daten versorgen.
Dazu muss z.B. ein Netzwerk eingerichtet sein und der PC, auf dem der Server eingerichtet ist, muss zur Arbeitszeit immer laufen.
Lokale Datenbanken sind die klassische Blütezeit der relationalen Datenbanken gewesen, wie:



- dBase II
- Redabas
- WORKS (Datenbank)
- BASE
- ACCESS (erste Versionen)

Wir sprechen hier auch von Desktop-Datenbank-Systemen.
Zeitlich ist diese Entwicklung den 70er Jahren des letzten Jahrhunderts zuzuordnen. Hier kam es zu einer explosionsartigen Entwicklung relationaler Datenbanken. Ursächlich dafür waren theoretische Vorarbeiten von Edgar (Ted) CODD (1970) zu Konzeption von Datenbanken.
Die Datenbanken stellten echte Relations-Systeme mit mehreren Tabellen dar. Das Arbeiten mit den Tabellen war recht einfach. Sichten, Formulare und Berichte dienten zur Vereinfachung der Arbeit und zur Darstellung der Daten.
Mit den Sichten sorgte man für die sinnvolle Auswahl und Anzeige der Daten. Die Daten-Auswertung war dann auch extern in speziellen Programmen (z.B. "Crystal Reports") möglich. Sie machten sehr komplexe Berichte möglich. Formulare waren für ein Nutzer-bezogenes Daten-Handling gedacht.

Vorteile:

- Daten (Datenbank) noch von Hand händelbar (z.B. kopieren, sichern, …)
- effektives Speichern und Manipulieren der Daten
- viele fertig intergrierte Funktionen, Hilfsmittel, Tools, …
- schnell erlernbar

Nachteile:

- Anfälligkeit gegen Hardware-Ausfall
- nur selten Betriebssystem-übergreifend
- komplexere Aufgaben erfordern hohe Vorkenntnisse
- Schnittstellen sind nicht einheitlich definiert / gestaltet
- Nutzer muss das Datenbank-System verstehen (viele Anwendungen sind nicht mehr Sekretärin-Like)
- Nutzer müssen ausgebildet / fortgebildet werden

Anwendungsbereiche:

- Kunden-Management
- Warenwirtschafts-Systeme (Bestellwesen, Lager-Verwaltung)
- Flug-Buchungssysteme
- Banken
- Versicherungen
- Telekommunikation
- Daten-Analyse
- Resoourcen-Planung (ERP)
- …

ev. noch einarbeiten: (Q: SQL-Kurs NAUMANN; OpenHPI, 2013)
System R von IBM war erster Prototyp (1974) eines RDBMS (relationales Datenbank-Management-System); Code-Größe ungefähr 1,2 MB
spezielle Anfragesprache SEQUEL, die noch nichts mit SQL zu tun hat, obwohl sie so ähnlich klingt auch schon viele Prinzipien vorwegnahm

unabhängig und parallel entwickelte man (1975) an der University California at Berkeley das System "Ingres" mit der Anfragesprache QUEL (query executing language)
das Konzept orientierte sich am CODDschen Modell
tragend war hier Michael STONEBRAKER ()
aus dem Ingres entstanden dann später solche System wie Postgres, Sybase, …
Postgres ist heute open source und wird vielfach weiterentwickelt

1979 erscheint Oracle Version 2; erste kommerzielle Version
ebenfalls basierend auf dem relationalen Datenmodell

Ein gutes Beispiel für eine lokale Datenbank sind immer die eigenen Datenspeicher auf dem Gerät. Ob dies Disketten, Festplatten, optische Datenträger oder moderne elektronische Speicher sind, ist dabei egal.

Definition(en): Dateisystem
Ein Dateisystem ist die Ablage-Organisation von Dateien auf einem Datenträger.
Ein Dateisystem ist Realisierung von Erstellung, Speicherung, Löschung, Ordnung, Umbenennung und Veränderung von Dateien auf einem Datenträger.

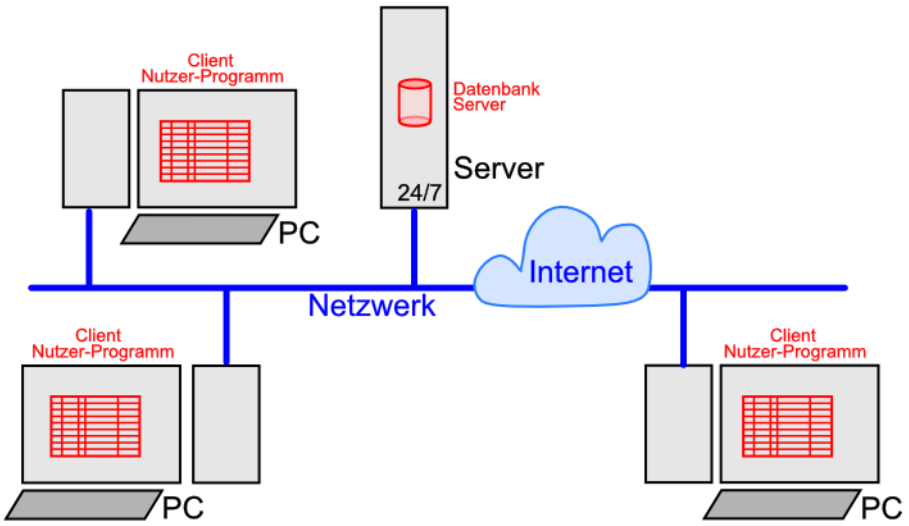
Beispiele für Dateisysteme:

Bezeichnung	**Grob-Beschreibung**	**Verfügbarkeit / Realisierung im Betriebssystem**	**Verwendungs-Empfehlungen / Bemerkungen**
FAT16			
FAT32			
FAT32ext			
NTFS			

Windows ®-Rechner verfügen z.B. mit ihrer **Registry** über eine Baum-artig strukturierte Datenbank zur lokalen Hard- und Software.

1.1.3. globale Datenbank-Systeme

Die globalen Datenbanken sind die heute typischen und allgegenwärtigen echten Datenbank-Systeme. Das Globale muss dabei nicht wirklich ausgeprägt zu sein. Wichtig ist eine klare Trennung von Client und Server, eine allgegenwärtige Vernetzung und praktisch beliebig viele Nutzer.



Globale Datenbanken sind meist Betriebssystem-übergreifend benutzbar. Häufig benutzt man Linux als freies Betriebssystem, da hier nur im hochproffesionellen Bereich Lizenz-Gebühren anfallen. Der Nutzer-Zugriff erfolgt typischerweise Browser-basiert.
Mitlerweile ist das Internet voll von Datenbanken. Da sind z.B.:

- facebook.com
- instagram.com
- twitter.com
- youtube.com
- mediathek.zdf.de
- aol.com
- yahoo.com
- …

zu nennen. Oft bekommt der Nutzer (Konsument) gar nicht mit, dass er eigentlich eine oder mehrere riesige Datenbanken benutzt. Kaum jemanden wird z.B. genau das beim Nutzen von google bewusst. Und trotzdem ist es eine der größten Datenbanken der Welt (2017: 7235 Exabyte), die wir da mehrfach kontaktieren und mit unseren Anfragen bombadieren.
Täglich wächst die Größe von google um über ein Petabyte.

Vorteile:

- schnell, effektiv, hohe Daten-Beständigkeit
- Daten stehen mehreren / vielen Nutzern gleichzeitig zur Verfügung
- Daten können vielfältig verarbeitet, kombiniert, angezeigt und ausgewertet werden
- Nutzerspezifische Sichten auf die Daten möglich (Einhaltung von (Personen-)Datenschutz-Richtlinien)
- hohe Datensicherheit
- Daten werden unabhängig von der speziellen / derzeitigen Nutzung gespeichert
- allgemeingültige, verbreitete Schnittstellen (ODBC, SQL)
- definierte / einheitliche Bedienkonzepte / Oberflächen (Formulare)
- Nutzer (Daten-Konsument) bekommt vom Datenbank-Konzept kaum etwas mit
- zentrale / hoch- profesionelle Bestreuung der Datenbank durch speziellen (Datenbank-)Administrator
- …

Nachteile:

- System-unabhängiges Händling der Daten-Dateien kaum noch möglich
- zur Administration sind sehr gute Vorausbildungen notwendig
- Konzept "Datenbank" muss dem Nutzer (Daten-Produzenten) klar sein
- viele Leistungen / … nicht mehr Sekretärin-Like (System ist häufig überdimmensioniert, wird nicht ausgenutzt)
- Nutzer müssen ausgebildet / fortgebildet werden; vieles aber schon intuitiv möglich
- Planung / Konzeptionierung unbedingt im Vorfeld notwendig
- Personal-Bedarf (es werden hochspezialisierte Administratoren, Service- und Fach-Informatiker benötigt)
- …

Aufgaben:

1. *Ermitteln Sie ev. neuere Angaben zur Speichergröße von google! Ansonsten benutzen Sie die obigen Angaben und stellen Sie die Speichergröße in Byte dar!*
2. *Wieviele Festplatten a 1 Terabyte müssten sich sich täglich kaufen um den gleichen Speicherzuwachs zu realisieren! Was würde Sie das aktuell kosten? Wer bezahlt das eigentlich beim kostenlosen google-Service?*
3. *Zum jährlichen Zuwachs von Rechenleistung und Speicherkapazitäten galt lange das MOOREsche Gesetz. Was besagt es und wie muss es heute eingeordnet werden? Stellen Sie Ihre Recherchen in einer geschlossenen Form dar!*

für die gehobene Anspruchsebene:

4. *Rechnen Sie die Speicherdaten von google in die modernen Speichergrößen (Binärpräfixe) Exbibyte und Pebibyte um!*
5. *Recherchieren Sie, wie die nächstgrößeren Speichergrößen (Dezimal- und Binär-Präfixe) lauten und welche Größe sie repräsentieren!*

Vorteile der elektronischen Datenverarbeitung (bezogen auf Datenbanken und deren Nutzung)

- Verknüpfung und Auswertung von Daten über viele Ebenen und Beziehungen
- Daten können leicht eingeschränkt werden (trotz großer verfügbarer Datenmenge)
- leichte Sortierung (auch über mehrer Ebenen), Gruppierungen und Filterung
- Erzeugung temporärer (Teil-)Datenmengen (für unabhängige Nutzungen, …)
- Schnelligkeit (genau soetwas können Computer gut: stures Abarbeiten festgelegter Algorithmen)
- …

Nachteile / Problemfelder:

- materiell-technischer Aufwand
- Verfügbarkeit von Daten nach / bei Stromausfällen / Zusammenbruch von Daten-Übertragungssystemen (Internet)
- Manipulierbarkeit / Angreifbarkeit
- Fehlfunktion der Technik
- …

Vorteile von Datenbanken

- Verwaltung großer Datenmengen möglich
- Mehrfachspeicherung kann / wird reduziert / unterbunden (Redundanz)
- Datenintegrität (Korrektheit der Daten) kann verbessert / durchgesetzt werden
- Daten können von verschiedenen Nutzern (auch gleichzeit) genutzt werden
- Daten können auf verschiedene Art und Weisen (Sichten) genutzt werden
- Nutzung der Daten kann über Dialoge oder externe Programme erfolgen
- Zugriffsrechte von Nutzern können festgelegt werden (Nutzerverwaltung)
- Datenschutz kann realisiert werden (Zugriffsbeschränkungen)
- Daten können effektiv gespeichert werden
- Daten werden unabhängig von Anwenderprogrammen gespeichert
- Daten können effektiv und zentral gesichert werden
- die Verwaltung, Datenprüfungen, Reparaturen, … können zentral durchgeführt werden
- …

Nachteile von Datenbanken

- Abhängigkeit durch zentrale und im Prinzip einmalige Datenquelle sehr hoch (Ausfallgefahr)
- Manipulation von Daten hat sofortige und meist weitreichende Konsequenz; auch bei (unbewußt) fehlerhaften Dateneingaben
- Gefahr der Unterwanderung des Datenschutzes
- …

typische Arbeiten an / mit Daten (in Datenbanken)

- Datenbank-Konzeption / -Struktur
- Dateneingabe
- Daten-Import / -Export
- Darstellung der Daten (in Tabellen / in Formularen)
- Berechnungen von Kennwerten
- Gruppierung, Sortierung und Filterung von Daten
- Suche von Daten / Daten-Kombinationen
- Erstellen von Situations-Berichten (Jahres-Abschluss, aktuelle Übersichten)
- Erstellen von Statistiken
- …

Aufgaben:

1. ***Ein echter Datenbank-Freak macht immer wieder die folgende Aussage: "Datenbanken sind extrem komplex und so spezielle informatische Lösungen, dass man das EVA-Prinzip auf sie nicht anwenden kann.". Setzen Sie sich mit der Aussage auseinander!***

Wo geht es hin? Was passiert derzeit?

Tendenzen sind zum Einen die Minituarisierung hin zu schlanken, schnellen und z.T. spezialisierten Datenbanken. Eine andere Tendenz geht hin zu den riesigen Datenbanken des WWW (und ähnlicher Internet-Dienste). Die Daten-Mengen sind extrem groß, bis hin zu vielen PetaByte's.
Die Datenbanken können nun auch gut mit Multimedia-Daten umgehen. Davon profitieren Streaming-Plattformen und Suchmaschinen
Datenbanken werden wirtschaftlichen, Geschwindigkeits- und Sicherheits-Gründen verteilt und parallel gespeichert. Durch spezielle Techniken (Blockchain-Technologie) versucht man eine vertrauens-Basis für geschäfte und Kommunikation zu erreichen, da in einer Welt wie unsere kaum noch jeder jeden kennen kann.
Je nach Dringlichkeit werden Daten in verschiedenen Speicher-Arten abgelegt. Daten, die ständig gebraucht werden, speichert man in Cache's nahe an den Prozessoren. Die meisten Daten werden nur realtiv dringend gebraucht. Sie können im Hauptspeicher oder auf Festplatten (oder aktueller SSD's) liegen. Selten gebrauchte Daten lagert man auf teriäre Datenträger, wie Band-Laufwerke (Streamer), DVD's usw. usf aus. Wenn diese gebraucht werden, dann muss man sich mit etwas längeren Zugriffzeiten zufrieden geben, das ist aber günstiger, als alle Daten ständig z.B. im Hauptspeicher zu halten.
Google könnte gar nicht schnell antworten, wenn alle Suchanfragen nur über eine Server-Farm z.B. in Amerika laufen würden.
In der Praxis hat man schnell festgestellt, dass die üblichen CPU's eigentlich mit den ständig anfallenden kleinen Arbeits-Aufträgen unterfordert sind. Besser sind dazu kleine, spezialisierte prozessoren geeignet, wie sie z.B. auf Grafik-Karten ihre Dienste leisten. Was liegt also näher, als solche GPU's für die Verarbeitung von Datenbank-Aktionen zu nutzen.
Eine weitere Tendenz im Datenbank-Bereich geht in Richtung Objekt-Orientierung. Hier kommen sehr komplexe Datenstrukturen zum Einsatz, die sich schwer in relativ statischen

Tabellen-Konstrukten abbilden lassen. Dazu will man Daten (Inhalte) und Struktur voneinander trennen, um so universeller und effektiver zu werden.
Bei der Objekt-orientierten Datenverarbeung verschmelzen die guten alten (drei) Ebenen des CODDschen Modells. Auch die deklarativen Abfragesprachen verschmelzen wieder mit den eigentlich schon lange überholten navigierenden Prinzipien. Zur Nutzung der Daten bedarf es spezieller Programmier- und Anfrage-Sprachen, die zwar existieren, aber derzeit nur eine geringe Anwendungsbreite besitzen.
Mit den neuen Trends Daten in riesigen Mengen zu erfassen und ev. auch zu speichern, kommen neue Datenbank-Typen dazu. Mit BigData-Systemen werden z.B. in den Massen einlaufender Sensor-Daten (Stichwort IoT (Internet of Thinks)) nach auffälligen Mustern gesucht. Auch die Kunden-Karten-Systeme (z.B. Deutschland-Card, Payback, …) nutzen die anfallenden Kaufdaten für weitere Daten-Analysen.

moderne Datenbank-Typen

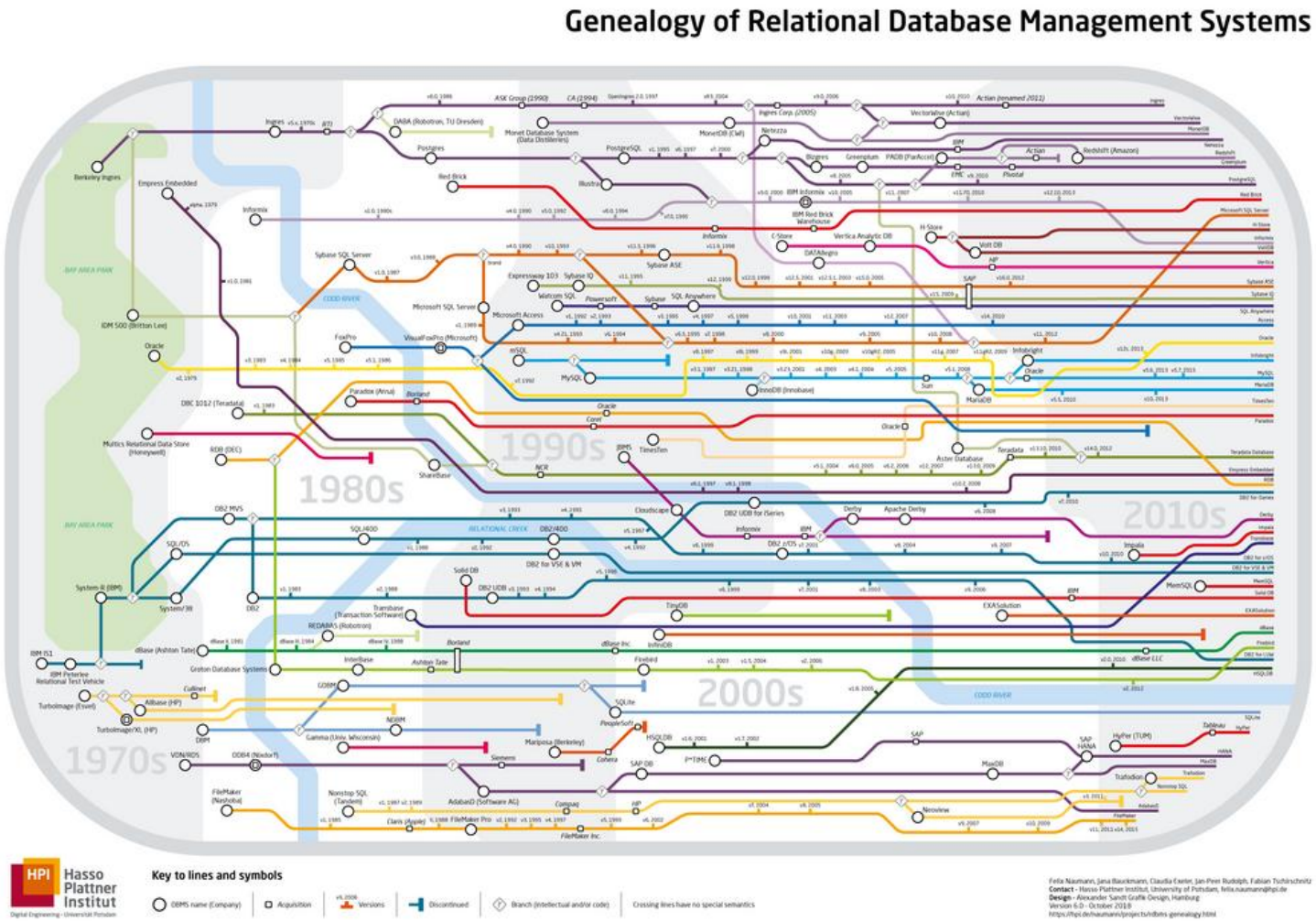
- **Multimedia-Datenbanken**	verarbeiten von Bildern, Musik, Video (große Daten-Mengen mit Meta-Daten)
- **XML-Datenbanken**	Nutzung semi-strukturierter Daten Unterstützung von Daten-Austausch
- **verteilte Datenbanken**	Daten werden mehrfach auf verschiedene Knoten eines Netzwerkes verteilt (Erhöhung der Gerechtigkeit, Absicherung gegen Fälschungen, Kooperation mit potentiell unbekannten Partner (Vertrauen schaffen)
- **förderierte Datenbanken, Multidatenbanken, Mediatoren**	Integration und Zusammenführung von Daten aus verschiedensten Quellen
- **mobile Datenbanken**	Datenbanken auf Kleinstgeräten (Handy's, Smartphone's, …)
- **Suche / Search Suchmaschinen**	schneller Zugriff auf Daten oder Details davon

Q: u.a.: SQL-Kurs (NAUMANN; OpenHPI; 2013)

Zeitraum / Jahr	technischer Stand	Bemerkungen / Hinweise
	•	
	•	
60er Jahre	• Datenbank-Systeme auf der Basis von hierarchischen Modelle • Einbeziehung von Netzwerk-Technologien • informatische(r) Technologien / Tends / Stand ○ Zeiger (Pointer) auf Daten(-Sätze) ○ schwache Trennung von physischer und konzeptioneller Ebene ○ navigierende Abfragesprachen	→ 2.3.1. hierarchisches Datenbank-Modell → 2.3.3. netzwerkartiges Daten(bank)-Modell
70er Jahre **80er Jahre**	• relationale Datenbank-Systeme ○ Daten in Tabellen (Relationen) ○ ausgeprägte Trennung in physische und konzeptionelle Ebene (3-Ebenen-Konzept) ○ deklarative Abfragesprachen	→ 2.2. relationales Daten(bank)-Modell
80er Jahre **90er Jahre**	• viele Entwicklungen zu kleineren und größeren Datenbank-Systemen (Skalierung) • Objekt-orientierte Datenbank-Systeme	→ 2.3.4. Objekt-orientiertes Daten(bank)-Modell
2000er Jahre	• Spezialisierungen auf neue / speziellere Daten-Typen	z.B.: → 2.3.5. Dokument(en)-orientiertes Daten(bank)-Modell
2010er Jahre	• Web-basierte und skalierbare Datenbank-Systeme • NoSQL • neue Hardware (Nutzung von GPU's, Main-Memory, SSD's, auch virtuelle Server, …)	→ 7. Web-Datenbanken
	•	

Q: u.a.: SQL-Kurs (NAUMANN; OpenHPI; 2013)

(kurze) Genealogy der rationalen Datenbank-Systeme



Q: http://www.hpi.uni-potsdam.de/naumann/projekte/rdbms_genealogy.html

2. Datenbank-Entwurf – von der Theorie zum Konzept

Bisher haben wir immer locker von Datenbanken und Datenbank-System gesprochen, ohne uns über die informatischen Begrifflichkeiten einen Kopf zu machen. Wenn wir nun tiefer in Datenbanken i.w.S. (im weiteren Sinne) eintauchen wollen, dann wird eine klare begriffliche Trennung notwendig.

Problem-Fragen für Selbstorganisiertes Lernen

2.0. Grundbegriffe

2.0.1. Daten, Informationen und Nachrichten

Der Begriff Information ist einer der vielgebrauchten in unserer modernen Gesellschaft. Hinterfragt man allerdings eine Definition, dann kommen wir ins Grübeln. Was ist das nun genau? Wozu gehören Informationen überbegrifflich? Durch welche Merkmale sind Informationen von anderen Grund-Elementen zu unterscheiden?
Vielfach wird Information als ein Grund-Element unserer Welt neben Stoff (Materie) und Energie betrachtet. Schwierig wird's aber, wenn man sich vergegenwärtigt, dass Information immer nur an Stoff und / oder Energie gebunden vorkommt. Information allein ist uns nicht bekannt, oder kann von uns nicht wahrgenommen werden.
Es gibt verschiedene Ansätze den Begriff Information zu spezifizieren. Bisher ist aber noch keine Theorie und keine Definition zu Information entwickelt worden, die allgemein anerkannt ist.
Ein erster wissenschaftlicher Ansatz geht auf die Nachrichten-Theorie von SHANNON (1948) zurück. SHANNON (1916 – 2001) entwickelte in seiner Theorie das Modell einer allgemeinen Nachrichten-Übertragung von einer Quelle zu einem Empfänger.
Die Informationen werden zuerst beim Sender kodiert, damit zu einem Übertragungs-Kanal passen.

Beim Empfänger werden die Signale dann wieder dekodiert. Die Kodierung und Dekodierung ist vom Übertragungs-Kanal abhängig. Wird ein "Hallo!" über den akustischen Kanal übertragen, dann müssen unsere Sprechorgane z.B. Luftdruckwellen erzeugen. Das Ohr dekodiert diese Signale dann wieder. Bei einer akustischen Signalisierung wird das Wort vielleicht auf einen Zettel geschrieben oder ein Handzeichen verwendet. In beiden Fällen dekodiert das Auge die Signale beim Empfänger.

Nach SHANNON sind Informationen solche Nachrichten, die Unsicherheiten beim Empfänger beseitigen. Das lässt sich an einem einfachen Beispiel gut erklären. Nehmen wir an alle Stunde werden im Radio die gleichen Nachrichten vorgetragen. Beim ersten Hören sind die Nachrichten für uns sehr informativ. Alles ist neu. Beim zweiten Mal ist kaum noch Neues aus dem Nachrichtentext herauszuhören. Da ist vielleicht noch das eine Detail, was wir beim ersten Mal überhört haben. Insgesamt haben die gleichen gesprochenen Nachrichten jetzt nur noch einen kleinen Informations-Wert. Beim dritten oder vierten Mal enthält die immer noch gleiche Nachricht nichts Neues mehr für uns. Wir wissen schon alles. Der Informations-Gehalt dieser Nachrichten ist für Null.
Somit können wir feststellen:
Informationen müssen also eine Aussage enthalten (, also dürfen nicht leer (nichtssagend) sein. Information müssen Neues enthalten (, also nicht Bekanntes wiederholen).

Ein anderer Erklärungs-Versuch nimmt die Entropie zuhilfe. Die Entropie ist ein Maß für die Ordnung eines Systems. Besser eigentlich für die Unordnung, denn eine steigende Entropie wird mit einer zunehmenden Unordnung in Zusammenhang gebracht. Die normale Tendenz ist eben auch die Zunahme von Entropie. Soll die Entropie (Unordnung) verringert werden, dann muss Energie aufgewendet werden. Von allein passiert das nicht.
Ihr eigens Zimmer zuhause ist ein gutes Beispiel. Unordnung entseht ohne Probleme – praktisch von allein. Um wieder Ordnung zu erzeugen – und die Entropie wieder auf das Zufriedensheits-Level der Eltern zu bringen – braucht sehr viel Energie.
Entropie und Information sind entgegengesetzte Eigenschaften eines Systems. Dies schauen wir uns mal an einem Stoff-Beispiel an.
Betrachten wir die Lösung eines Stoffes in einem Lösungsmittel – z.B. Wasser. Geben wir einen Kristall eines – hier roten Stoffes in das Lösungsmittel, dann beginnt sich der Kristall aufzulösen und bald ist die gesamte Lösung rot gefärbt. Dieser Zustand ist normal und praktisch unendlich lange vorhanden. Die völlig ungeordnete Verteilung des gelösten Stoffes hat praktisch kein Information für uns – es ist der normale Zustand. Dieser Zustand entspricht der maximalen Entropie für dieses System.
Das System enthält zum Zeitpunkt der Zugabe des Kristalls in das Lösungsmittel genau die gleiche Teilchen – nur eben anders verteilt. Dieser Zustand ist ungewöhnlich und selten. In wenigen Augenblicken wird er beendet sein und praktisch nie wieder eintreten (obwohl das statistisch gesehen möglich ist, aber eben extrem unwahrscheinlich). Der System-Zustand mit dem ungelösten Kristall enthält viel Ordnung – also wenig Entropie. Und er ist sehr informativ. Eine geringe Entropie ist also mit einem hohen Informations-Wert verbunden.

Definition(en): Information
Information ist das Maß für die Unsicherheit beim Empfänger.
Information ist das Korrelat von Unkenntnis (Harry PROSS)
Information ist die Teilmenge an Daten (Wissen), die ein Sender einem Empfänger über einen bestimmten Übertragungskanal zur Verfügung stellen kann.

Den Begriff Daten abzuleiten fällt etwas einfacher, weil wir Daten allgemein den Informationen zuordnen können. Bei Daten handelt es sich einfach um die Informationen, die in irgendwelchen Systemen verarbeitet werden. Unter Verarbeiten schließen wir hier auch Eingeben, Speichern und Ausgeben mit ein.

Definition(en): Daten
Daten sind Sammlungen von Zeichen (Symbolen), die für Werte, allgemeine Angaben, Aussagen, … stehen, einen bestimmten Informations-Gehalt haben und dem Zweck der Verarbeitung dienen.
Daten sind interpretierbare Darstellungen von Informationen in formalisierter Art, die zur Kommunikation, Interpretation oder Verarbeitung geeignet sind.
Daten sind Gebilde aus Zeichen oder kontinuierliche Funktionen, die aufgrund bekannter oder unterstellter Abmachungen Informationen darstellen. Sie dienen vorrangig zum Zweck der Verarbeitung und als deren Ergebnis.

Betrachten wir noch kurz die Nachrichten aus informatischer Sicht. Damit meinen wir Daten / Informationen, die in einem System – in einer Kommunikation – übertragen werden, so wie das SHANNON in seiner Theorie ausgeführt hat. Für die Betrachtungs von Datenbanken haben Nachrichten vor allem bei den Transaktionen eine große Bedeutung. Im Allgemeinen werden Nachrichten aber bei den Datenbanken eher stiefmütterlich betrachtet und mehr den Netzwerken (📖 **Rechner, Netzwerke und Protokolle**) oder der allgemeinen Informatik zugeordnet.

Definition(en): Nachrichten
Nachrichten sind die Daten, die bei einer Kommunikation übetragen werden.

2.0.2. Datenbanken und Datenbank-Systeme

Unter Datenbanken im weiteren Sinne (i.w.S.) verstehen wir alle Datensammlungen, egal ob sie in elektronischer oder nicht-elektronischer Form existieren. Während die Datenbank meist die reine Datensammlung meint, versteht man unter dem Datenbank-System Datenbank mit ihren begleitenden Einrichtungen. Das sind bei nicht-elektronischen Datenbanken z.B. die Karteikarten oder die Register-Schränke.

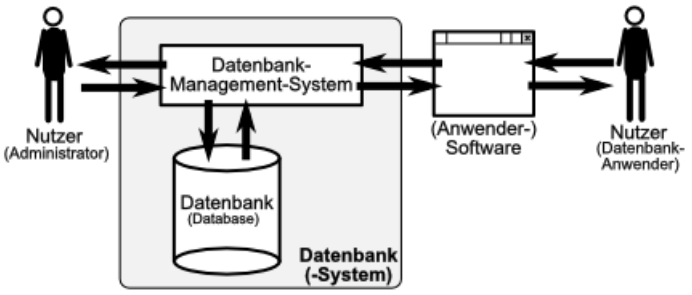
Definition(en): Datenbank(-System) / Datenbank i.w.S.
Ein Datenbank-System ist eine Einrichtung zur Verwaltung von (größeren Mengen an) Informationen.
Eine Datenbank ist ein System zur Verwaltung von (größeren Mengen an) Informationen.
Ein Datenbank-System ist eine Software (Programm od. App), die den Zugriff auf und die Verwaltung von Daten erlaubt.
Eine Datenbank ist eine systematisch strukturierte, dauerhafte Sammlung von Daten (Informationen) einschließlich dazugehörigen Werkzeugen zur Manipulation, Nutzung und Sicherung der gesammelten Daten.
Eine Datenbank (i.w.S.) ist eine Kollektion von Daten, die in einer Daten-Basis (Datenbank i.e.S.) gespeichert sind und von einem Datenbank-Management-System (DBMS) verwaltet werden.
Ein Datenbank-System ist eine Applikation / ein Programm(-System), bei dem die Nutzer (aller Ebenen) über ein Datenbank-Management-System Datenbanken anlegen, Systemkompatible Datenbanken verknüpfen und auf die Daten-Basen zugreifen können.

Computer-basierte Datenbanken sind praktisch immer elektronische Datenbank-Systeme. Wir müssten sie hier dann exakterweise als Datenbanken / Datenbank-Systeme im engeren Sinne (i.e.S.) benennen. Das machen wir nicht und verabreden ab hier, dass die besprochenen Datenbanken / Datenbank-Systeme immer elektronisch sind und aus unserer gehobenen fachinformatischen Sicht betrachtet werden. Wir werden gleich sehen, dass die Festlegung sauberer Begrifflichkeiten nicht ganz leicht ist.
Vergegenwärtigen wir uns noch einmal wozu wir eigentlich Datenbanken brauchen:

Aufgaben eines Datenbank-Systems:

- mit größere Daten-Menge effizient umgehen
- die Daten widerspruchsfrei und dauerhaft speichern
- sichere Speicherung von Daten
- bedarfsgerechte Bereitstellung der Daten für Nutzer und Programme
- Kontrolle und Durchsetzung von Zugriffsrechten und des Datenschutzes

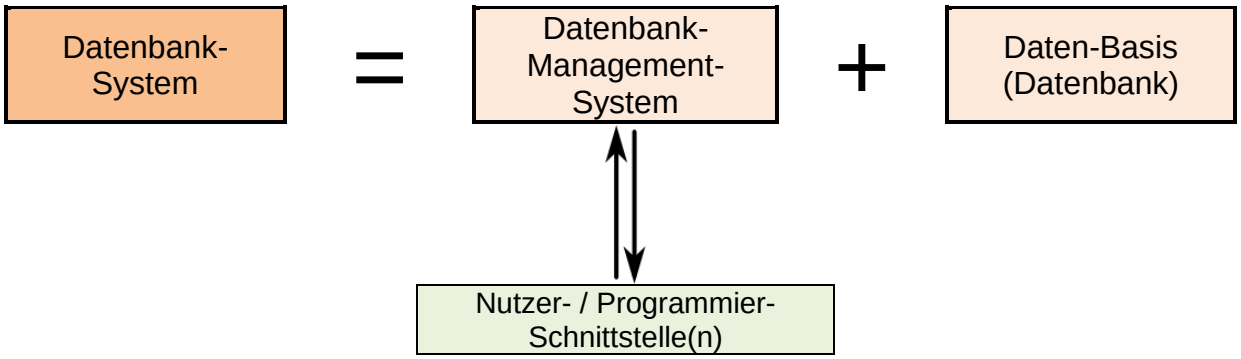
Um diese Aufgaben dauerhaft und universell zu realisieren hat sich die nebenstehende Struktur eines Datenbank-Systems als besonders effektiv ergeben.



Das gesamte Datenbank-System besteht aus der eigentlichen Datenbank – auch Database bzw. Datenbasis genannt – und dem Datenbank-Management-System.
Alle Nutzer greifen auf die Daten-Basis nur über das Datenbank-Management-System (DBMS) zu.
Der Zugriff auf die eigentlichen Daten in irgendwelchen Dateien obliegt dem DBMS. Nur der Programmierer des DBMS weiss genau, wie die Daten in der Daten-Basis abgelegt sind und wie mit ihnen gearbeitet wird. Selbst der Datenbank-Adminsitrator – sonst der Mann für alle Fälle und Probleme – ist hier nicht wirklich involviert. Gute Administratoren wissen natürlich über die Arbeitsweise ihres Systems bescheid, beeinflussen können sie es aber nur über das DBMS.

Datenbank-System (Abk. DBS) (Datenbank i.w.S.) besteht aus:

- Datenbank (Abk. DB) (i.e.S.; Daten-Sammlung, Daten-Basis, Daten-Bestand)
- Datenbank-Management-System (Abk. DBMS) (Daten-Verwaltungs-Software)



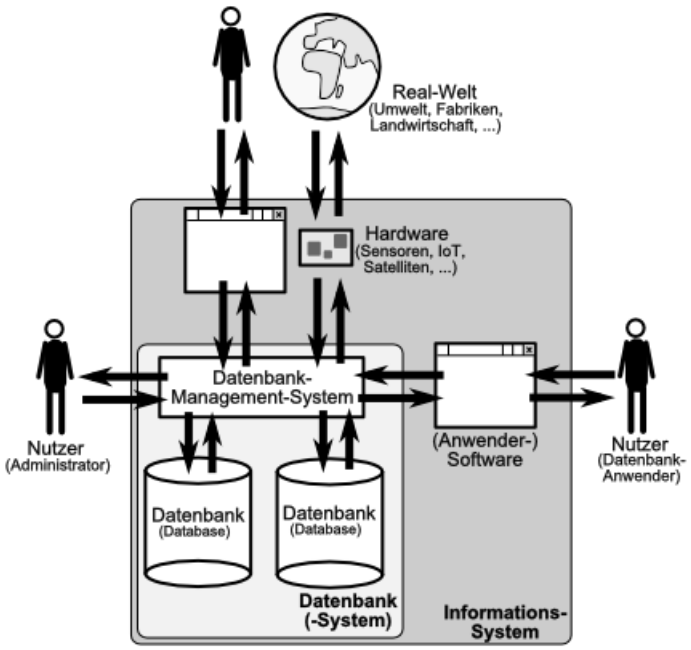
Zugriffe von Programmiersprachen, irgendwelchen Programmen, Apps und Diensten sowie die Datenbank-Sprache SQL (Structured Query Language (Strukturierte Abfrage-Sprache)) erfogen immer über die Schnittstellen des Datenbank-Management-Systems.
Die Verfügbarkeit von SQL-Schnittstellen ist kein Muss – hat sich aber als ein Quasi-Standard herauskristallisiert.

Geht man den nächsten Schritt nach außen, dann kommen neben den Anwenderprogrammen auch noch die automatisierten Systeme dazu. Hierzu zählen wir den gesamten Bereich der IoT (Internet of Thinks) oder die Daten-Erfassung mittels Satelliten. Sie stellen quasi eine direkte Verbindung von Real-Welt zu unserem Datenbank-System her. Die benutzte Schnittstelle ist und bleibt das Datenbank-Management-System.

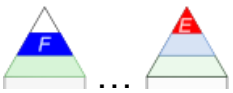
Wir sprechen jetzt von Informations-Systemen.

Besonders im Bereich der Produktion, des Handels und Miltärs sowie der Geodaten haben sich sehr große Informationssysteme entwickelt.

Auch im Bereich der Geheimdienste kann man davon ausgehen, dass hier auf der Ebene von Informationssystemen hantiert wird.



Moderne Datenbank-Systeme sind sehr komplex. Sie bestehen aus vielen Komponenten, die z.T. unabhängig voneinander funktionieren / verwaltet und weiter entwickelt werden. Einige Komponeneten bzw. Aspekte seien hier (umseitig) kurz vorgestellt:



Architektur / Aspekte eines modernen Datenbank-Systems:

- Hardware	
○ **Speicher-Systeme**	Festplatten, Netzlaufwerke, Cloud's, …
○ **Sicherungs-Systeme**	Netzlaufwerke, Streamer, NAS, …
○ **Ausfallsicherungen**	Unterbrechungs-freie Stromversorgungen, Parallel-Systeme
- Software	
○ **Datenbank-Verwaltungs-system**	Bedien-Oberfläche des DBMS SQL (DCL)
○ **Entwicklungsumgebung**	
○ **Datenbanksprache**	SQL
○ **Anwenderprogramme (Nutzerprogramme)**	spezielle Programme / Apps / Dienste für spezielle Anwender Browser-Zugriffe
○ **Hilfsprogramme, Tools**	Sicherungs- / Backup-Tools Schnittstellen zur Umwelt (z.B. IoT)
- Datenbasis	
○ **(reale) Daten**	eigentliche, in der Datenbasis gespeicherte - Daten
○ **organisatorische / Verwaltungs-Daten**	? Wer hat welche Zugriffs-Rechte? ? Wer hat hat wann, was getan?
○ **Struktur- und Hilfs-Daten Datenmodell-abhängige Daten**	z.B. Hash-Daten für die interne Daten-Sicherheit
- Verwaltung	
○ **Gesetze, Verordnungen, …**	Datenschutz-Bestimmungen der Einrichtungen / des Landes / des Staates / der EU
○ **Installation, Einrichtung, Updates**	
○ **Nutzerverwaltung**	? Wer hat welches Passwort und welche Rechte?
○ **Datensicherung**	? Wann und in welcher Form soll die Datenbasis gesichert / repliziert werden?
- Organisation	
○ **Betriebsfestlegungen**	Schließ- und (Hoch-)Sicherheits-Bereiche
○ **Zugangskontrollen**	z.B. RFID-Key's
○	

Da wird schnell klar, das Datenbank-Systeme heute keine Ein-Mann-Projekte mehr sind. In effektive Systeme fließt viel Experten-Wissen.
Das zentrale Teil ist also das Datenbank-Management-System.

Definition(en): Datenbank-Managment-System
Ein Datenbank-Managment-System ist eine Software zur Verwaltung zur Verwaltung einer oder mehrerer Datenbanken.

Dem Datenbank-Management-System müssen deshalb eine Vielzahl von Aufgaben zugeordnet werden:

Aufgaben / Funktionen eines Datenbank-Management-Systems:

- Unterstützung von Daten-Modellen
- einheitliche Verwaltung von Daten eines Problembereiches / Anwendungszweckes
- zentrale / koordinierte Speicherung der Daten
- Realisierung der notwendigen Datensicherheit / Legalität des Zugriffs / Ausfall-Schutz
- Gewährleistung von Transaktionen (Transaktions-Management)
- Schaffung der Daten-Integrität
- Umsetzung der Abfrage-Optimierung
- Unterstützung von (externer) Anwender-Software
- Bereitstellung genormter Schnittstellen (Sprach-Fähigkeit)
- Mehrnutzer-Fähigkeit
- Kontrolle und Umsetzung von Zugriffs-Berechtigungen (Nutzerverwaltung)
- Umsetzung des (Personen-)Datenschutz
- Realisierung von Dokumentations-Pflichten
- Bereitstellung einer (Schnittstelle für eine) Datenbank-Sprache (DDL und DML)
- effektive Anfrage-Abarbeitung (auch für komplexe Probleme)
- Speicher-Verwaltung (des benutzten Arbeits-Speichers (RAM) oder Fest-Speichers (HDD, SSD, Streamer, …)
- hohe Datensicherheit bei Strom-Ausfällen (gute Rekonstruktion von Daten-Beständen)
- Abfangen von System- oder Logik-/Mathematik-Fehler (z.B.: Division durch Null)
- …

Die Zahl der verfügbaren Sysdteme ist recht gross. Vor allem im Linux-Bereich sind sehr viele unterschiedlichste Realisierungen bekannt.
Mehr aus der WINDOWS-Welt stammen die nachfolgenden …

Beispiele:

- dBASE (→)
- LibreOffice / OpenOffice BASE (früher: StarOffice BASE) (→ 3.1.5. Datenbanken mit BASE)
- MySQL
- Informix (IBM)
- Lotus Smart Suite (Office-Suite von Lotos)
- Lotus Notes
- IBM DB2
- Oracle (Database)
- Microsoft ACCESS aus dem microsoft Office (→ 3.1.6. Datenbanken mit ACCESS)

- microsoft / IBM WORKS (Datenbank)
- microsoft SQL-Server
- SAP MaxDB und Sybase
- Teradata
- Corel / Borland Paradox
- SQLite (→ 3.1.7. Datenbanken mit SQLite)
- PostgreSQL
- …

	transaktionale Datenbank	Data Warehouses
Arbeits-Prinzip	Online Transaction Processing (OLTP)	Online Analytical Processing (OLAP)
Zweck	Transaktions-Verwaltung	Daten-Analyse, Entscheidungs-Hilfe
Merkmale	• viele Daten-Veränderungen • viele kurze Anfragen • mehr / viele Schreib-Operationen • kurze Reaktions-Zeiten • Ergebnisse landen wieder in der Datenbank (neue Datenbank-Situation)	• viele (wenig veränderliche) Daten • viele längere / komplexe Anfragen • sehr viele (auch aggregierende) Lese-Operationen • mittlere bis lange Reaktions-Zeiten • Ergebnisse werden extern verwendet (z.B.: Statistiken, Diagramme, Berichte, …) bei fast unveränderlicher Datenbank-Situation
Nutzer	Kunde oder Sachbearbeiter	Entscheidungs-Träger / Management; Analysten

Vielfach werden auch die Tabellenkalkulationen aus irgendwelchen Office-Paketen als Datenbanken angepriesen. Solche Programme, wie Microsoft EXCEL und Libre-/Open-Office Calc stellen zwar diverse Datenbank-Funktionen für ihre Tabellen zur Verfügung, aber praktisch fehlt ihnen das DBMS. Wir zählen sie nicht zu den echten Datenbank-Systemen im modernen Sinn.
Moderne Datenbank-Management-Systeme werden vorrangig für die folgenden Anwendungsgebiete genutzt:

- online-Shop's
- Warenwirtschafts-Systeme (mit Lager-Verwaltung, Bestell- und Abrechnungs-Systemen)
- Kunden-Management (z.B.: Ärzte, Händler, …)
- Lohn-Abrechnung
- Daten-Analyse (Firmen-Beratung) (→ Big Data, …)
- Versichungen
- Telekommunikation
- Ressourcen-Planung und –Verwaltung (ERP); Arbeits-Gruppen-Tools (Groupware)
- Banken (Geld-Karten-Management, Konten-Verwaltung, …)
- touristische Buchungs-Systeme (Hotels, Busreisen, …)
- Ticket-Reservierungs-Systeme (Konzerte, Bahn- oder Flugreisen)
- Wissens-Datenbanken (wikipedia, …)
- …

Die Programme ACCESS, BASE und SQLite sind Datenbank-Systeme, die sowohl lokal als auch global (zumindestens im eigenen Netz) verwendbar sind. Wir gehen im Folgenden vorrangig von lokalen Datenbanken aus, damit andere Nutzer (Kursteilnehmer) nicht unter den Fehlern anderer leiden müssen. In einigen Fällen werden wir aber auch das globale Funktionieren untersuchen.
Relationale – also auf Tabellen basierende – DBMS stellt möglicherweise vier Klassen von Datenbank-Hilfsmittel zur Verfügung

- **Tabellen** Grund-Struktur / -Form der Daten-Organisation (Anordnung der Daten in Zeilen (Datensätzen) und Spalten (Felder od. Attribute))
  (→ 3.1.3.1. Tabellen)
- **Abfragen** Auswahl, Suche, Sortierung, Gliederung, Filterung der Daten (aus Tabellen und / oder anderen Abfragen) und Bereitstellung als temporäre Tabellen
  (→ 3.1.3.2. Abfragen)
- **Berichte** Bereitstellung von Protokollen (Reporte, Darstellungen) zu einer speziellen Datenbank-Situation (aktuell od. z.B. Jahres-Abschluss)
  (→ 3.1.3.3. Berichte)
- **Formulare** Eingabe- und Anzeige-Hilfen für Tabellen und / oder Abfragen
  (→ 3.1.3.4. Formulare)

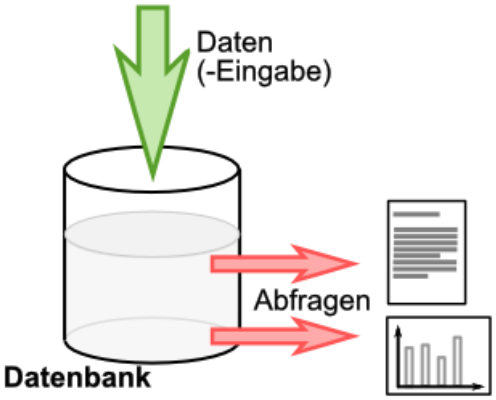
Dabei gehören Tabellen (Relationen) und Abfragen (Sichten, View's) zu den elementaren Betsandteilen.
Berichte sind heute meist schon bei den Anwender-Programmen jenseits der Schnittstelle zum Datenbank-Management-System angesiedelt.
Formulare spielen nur in voll integrierten Datenbank-Systemen – z.B. in einer Office-Suite – eine Rolle. Das liegt daran, dass gerade Formulare an das umgebende Betriebssystem angepasst sein müssen. Solche Datenbanken, wie ACCESS, die praktisch nur unter WINDOWS laufen, können soetwas realisieren.
Weiterhin können / sollten heute dazu gehören

- **Makros** Speicherung und Bereitstellung von Aktions-Folgen zur Arbeits-Effektivierung
  (→)
- **Datenbank-(Programmier-) Sprache** Bereitstellung einer Skript- und / oder Programmier-Sprache zum Erzeugen und Nutzen von Datenbanken, Tabellen, Abfragen und Berichten
  (→ 5. Datenbanken - SQL)

Definition(en): Datenbasis / Datenbank i.e.S.
Die Datenbasis ist die Gesamtheit der gespeicherten Daten eines Datenbank-Systems (Datenbank i.w.S.) einschließlich der zugrundeliegenden Struktur-Beschreibung (Datenbank-Modell).
Eine Datenbank (i.e.S.) ist eine systematisch strukturierte, langfristig verfübare Sammlung von Daten.
Eine Datenbasis / Datenbank i.e.S. ist die Gesamtheit der abgespeicherten Daten und deren Beziehungen untereinander.

Mit dem Daten-Überfluß und modernen Data-Mining-Ansätzen kommen heute neben den reinen Datenbank-Systeme noch Datenstrom-Systeme dazu.

Eine (klassische) **Datenbank** arbeitet in etwa nach dem fogenden Prinzip.



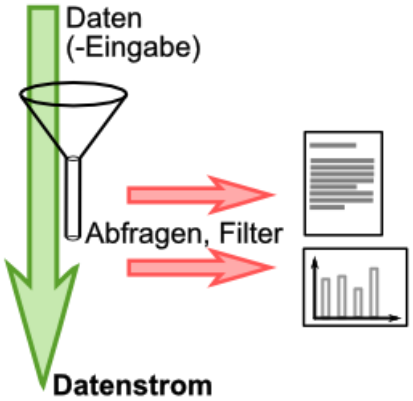
(Neue) Daten werden permanent in der Datenbasis gespeichert. Sie stehen auch nach Beendigung des eingehenden Daten-Strom's weiter zur Verfügung.

Die Daten aus der Datenbasis können zeitlich hintereinander mehrfach und unanbhängig voneinander genutzt werden. Daten, Situationen und Datenfolgen lassen sich im Allgemeinen immer wieder rekonstruieren.

Die Ausgabe-Daten (Abfragen und Berichte) stellen immer Situations-Zustände dar, wobei die verfügbare Datenmenge ständig steigt.

Aus Datenbanken lassen sich nur in sehr geringem Umfang dynamische Anzeigen (laufende Verkäufe, Lagerbestände) erezugen. Man muss dazu immer eine Abfrage starten, die praktisch den gesamten Daten-Bestand durchforstet.

Bei einem **Datenstrom-System** werden (neue) Daten praktisch nicht oder nur temporär gespeichert. Die Menge an Daten ist schier zu groß und selten sind alte Daten noch weiter interessant. Ältere Daten oder Datenfolgen lassen sich in Datenstrom-Systemen deashalb auch nicht wieder rekonstuieren.



Die interessierenden Daten werden nach aktuellen Anforderungen (Abfragen) in Echtzeit herausgefiltert und ausgewertet. Die Ergebnis-Daten sind somit immer aktuell und dynamisch veränderlich.

Verwendung finden Datenstrom-Systeme im Bereich des (online-)Handels, zu Kontrollzwecken oder zurm Abhören der Kommunikation.

Moderne Produktions-Anlagen – wie Chemie-Anlagen – verbinden Datenbank- und Datenstrom-Systeme. Für die Steuerung der Anlage benutzt man mehr das Datenstrom-Prinzip, während Dokumentation und das Beratungs-System (Experten-System) eher Datenbank-orientiert ist.

2.0.3. allgemeine Merkmale / Charakteristika von Datenbanken

Wir haben schon mehrfach aufgezeigt, dass Datenbanken keine "sonebenbei"-Projekte sind. Sie sind sehr komplex und müssen sehr sicher sowie effektiv arbeiten. Daraus ergeben sich diverse …

Anforderungen an ein modernes Datenbank-System

- **Redundanzarmut** — die (ungeordnete) Mehrfachspeicherung von Daten wird unterbunden / reduziert
- **Integritätssicherung** — die Daten werden auf Vollständigkeit und Korrektheit geprüft
- **Datensicherheit (Daten-Konsistenz)** — die Daten werden vor dem Verlust geschützt und regelmäßig und mehrfach gespeichert (/ gesichert)
- **Datenschutz** — der Zugriff auf ausgewählte / notwendige Daten wird nur für berechtigte Nutzer / Programme ermöglicht
- **Mehrbenutzerbetrieb** — mehrere / viele Nutzer / Programme können gleichzeit (parallel) an / mit den Daten arbeiten
- **Datenunabhängigkeit** — Trennung der Strukturen der Daten in der Datenbasis von den Datenbank-Management-Systemen und den Nutzer-Programmen
- **zentrale Adminstration** — die Verwaltung der Nutzerrechte, Datensicherungen, Datenreparaturen, Updates, … erfolgen von einer Stelle aus und werden von wenigen (hoch-)qualifizierten, (hoch-)berechtigen Personen durchgeführt

Diese **Anforderungen** an ein modernes bzw. neues Datenbank-System werden auch **Entwurfs-Prinzipien** oder **Charakteristika** genannt. An ihnen wird die Qualität einer Datenbank oder ihrer Implementierung gemessen. Deshalb müssen sich die Programmierer unbedingt mit Anforderungen auseinandersetzen, um den Entwurf (die Implementierung) Ziel-gerichtet zu realisieren.
Eine andere – mehr praxis-orientierte – Möglichkeit die Leistungen einer Datenbank zu charakterisieren sind:

- **Persitenz** — beschreibt die Dauerhaftigkeit und Verweildauer der Daten in einer Datenbank
  i.A. wird erwartet, dass Daten länger zur Verfügung stehen als es für den eigentliche (Geschäfts-)Prozess notwendig wäre
- **Quantität** — macht Aussagen zum Daten-Bestand in der Datenbank
  ist nur zum Zeitpunkt des Entwurf's fixiert
  während der Lebensdauer der Datenbank ist der Bestand i.A. dynamisch
- **Reaktivität** — beschreibt die Geschwindigkeit mit der eine Datenbank eingehende Daten in den Bestand aufnimmt, auf Anfragen reagiert und ev. autark notwendige Aktivitäten auslöst

- **Integrität** gibt die Vollständigkeit, Korrektheit und Unversehrtheit der Daten wieder

Nachrichten und Informationen aus Literatur, Presse und Internet:

Ostseezeitung

Haribo: Probleme mit den Gummibären

Grafschaft. Der Süßwarenhersteller Haribo hat mit Produktionsproblemen bei Goldbären, Fruchtgummi-Vampiren und anderen Erzeugnissen zu kämpfen. Die Einführung eines neuen Softwaresystems habe zu größeren Lieferschwierigkeiten als erwartet geführt, hieß es am Freitag. Praktisch alle Produkte seien davon betroffen. Doch verbessere sich die Situation inzwischen wieder. Hintergrund: Haribo hatte im Oktober damit begonnen, sein veraltetes Warenwirtschaftssystem auf ein neues Programm umzustellen.

/Q: IN: Ostsee-Zeitung – Sonnabend/ Sonntag 15./16. Dezember 2018, S. 10/

2.0.3.1. Redundanz

Unter Redundanz versteht man das unnötige mehrfache Vorkommen gleicher Daten in einem System. Mit anderen Worten, wenn man aus einem System bestimmte Daten entfernen kann, ohne dass ein Informations-Verlust auftritt, dann sprechen wir bei diesen Daten von redundanten Daten.
Sind (gleiche) Daten mehrfach in einem System vorhanden, dann hat das verschiedene Vor- und Nachteile.

Vorteile von Redundanzen:

- höhere Datensicherheit
- höhere Übertragungssicherheit
- ev. schnelleres Finden
- …

Nachteile von Redundanzen

- höhere Kosten durch mehrfache Eingabe; aufwändigere Verarbeitung (Finden von Doppelungen usw. usf.; Konsistenz-Prüfungen; …); …
- Nutzung / Bedarf von / an Speicherplatz
- ev. mehrfache Datenübertragung (gleicher Daten)
- erhöhter Aufwand beim Ändern / Löschen
- höhere Fehler-Häufigkeit in verarbeitenden Programmen
- …

Definition(en): Redundanz
Redundanz ist ein Maß für das (überflüssige,) mehrfache Vorhandensein von Daten.
Redundanz ist das Maß für die Information pro Zeichen (Symbol), die in einer Datenquelle (, einem Übertragungssignal, …) mehrfach vorhanden ist.

Sind in einem System Informationen unabhängig voneinander mehrfach abgespeichert, dann ist das System bezüglich dieser Daten redundand.
Das ist mit diversen Problemen behaftet:

- Wo müssen diese Daten ev. überall geändert oder gelöscht werden?
- Wurden eigentlich redundante Daten unterschiedlich geändert, welche Daten sind dann die richtigen?
- …

In großen Datenbanken steht natürlich an vorderster Stelle dern unnötig verbrauchte Speicherplatz. Bei den heutigen Preisen für Datenträger ist das zwar nicht mehr bedeutend, aber bei großen Datenmengen läppert sich das schon zusammen. Schließlich müssen und werden die Datenbestände mehrfach gesichert und ev. auch verteilt werden. Heute steht vielfach mehr die Geschwindigkeit von Datenbanken im Fokus. Eine unnötig vergrößerte Datenbank bedeutet eben auch längere Bearbeitungszeiten.

Redundanz

Redundanzen sichern Daten und Kommunikation ab.

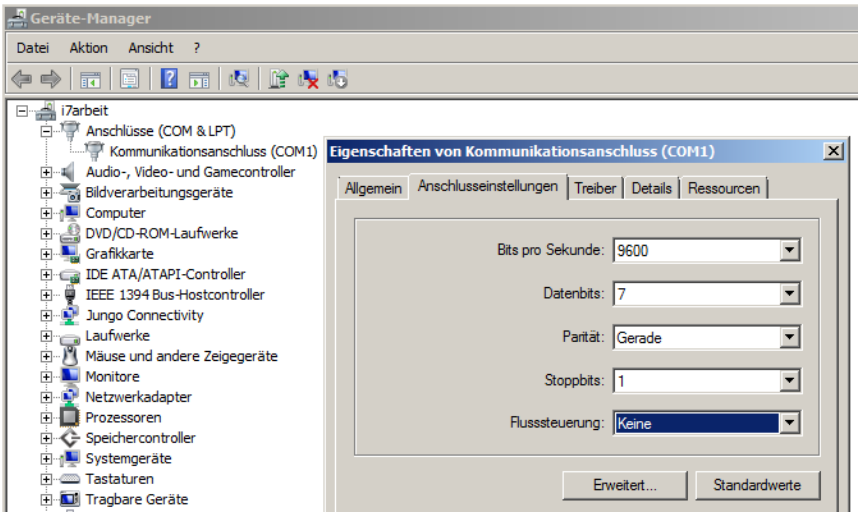
Bei der seriellen Übertragung von Daten z.B. an Consolen-Schnittstellen von Routern usw. benutzt man nur 7 bit für die eigentlichen Zeichen. Das 8. Bit benutzt man zur einfachen Fehler-Erkennung. Dabei gibt es die Möglichkeit der Geraden (eng.: even) und Ungeraden (engl.: odd) Parität.

Daten-Bit-Sequenz	Anzahl Einsen	Paritäts-Bit (even-Parität)	Sende-Sequenz
0000000	0	**0**	00000000
0100110	3	**1**	01001101
0110000	2	**0**	01100000
1001100	3	**1**	10011001
1111111	7	**1**	11111111

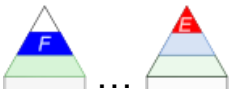
Nehmen wir als Beispiel die Gerade Parität (even). Hier wird das 8. Bit gesetzt, wenn die Anzahl der Einsen ungerade ist. In diesem Fall wird durch das Setzen des Bit's die Parität wieder hergestellt – also die Anzahl aller Einsen ist gerade. Entsprechend wird bei einer geraden Anzahl von Einsen in den sieben Daten-Bits das Paritäts-Bit nicht gesetzt (bleibt also 0).

Beide Kommunikations-Geräte – also Sender und Empfänger – müssen dazu gleichartig eingestellt werden.

Häufig gibt z.B. der Router die Einstellungen vor und die bedienende Konsole (hier z.B. ein PC) wird entsprechend eingestellt.

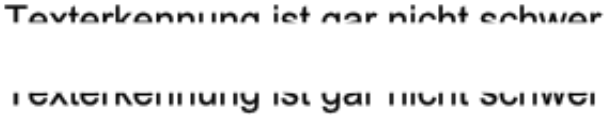


Redundanz 1.Ordnung (informations-theoretische Redundanz)

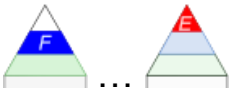


bezieht auf die Zeichen einer Sprache

Bei Texten aus lateinischen Buchstaben ist die obere Hälfte informativer als die untere Hälfte. Es gelingt uns deutlich besser den oberen Text zu entziffern, als den unteren.



Redundanz 2. Ordnung (sprach-wissenschaftliche Redundanz)



bezieht sich auf Wörter

Sicher haben Sie den folgenden Text schon mal irgendwo gesehen und gelesen:

Laut eienr Stidue der Cmabridge Uinverstiaet, ist es eagl in wlehcer Reiehnfogle die Bchustebaen in Woeretrn vokrmomen. Es ist nur withcig, dsas der ertse und lettze Bchusatbe an der ricthgien Stlele snid. Der Rset knan total falcsh sein und man knan es onhe Porbelme leesn. Das ist, wiel das mneschilche Geihrn nciht jeden Bchustbaen liset sodnern das Wrod als gaznes.
Krsas oedr?

originale Quelle dieses Tesxtes nicht mehr identifizierbar

Obwohl er von der Buchstaben-Sequenz her, eher Daten-Müll ist, können wir ihn recht schnell lesen. Mit ein paar Übungen geliengt das dann fast genauso schnell, wie das Lesen eines "ordentlichen" Textes. Für einen Computer wäre der Text wohl kaum lesbar. Schon ein einzelner Buchstaben-Fehler in einem Wort sorgt für Probleme.
Wir sehen hier auch wieder, das Redundanzen immer System-abhängig sind.

Redundanz 3. Ordnung (kommunikations-wissenschaftliche Redundanz)

bezieht sich auf ganze Sätze

In der menschlichen Sprach-Praxis sind das aufgebähte – sich ständig wiederholende - Texte.
Vielfach wird durch ständige Wiederholung von Sätzen, Satz-Fragmenten oder speziellen Begriffen versucht, eine neue Wahrheit zu erzeugen.
In Datenbank-Systemen können wir die Redundanz 3. Ordnung wohl den mehrfach vorkomenden Datensätzen zuordnen. Sie unterscheiden sich nur im System-bezogenen Primär-Schlüssel – also dem Erkennungs-Zeichen des Datensatzes und können folglich gelöscht werden.

In der Datenbank-Praxis werden doppelte / mehrfache Daten aus Tabellen mittels **Normalisierung** (→) entfernt. Dies ist ein wichtiger Arbeitsschritt bei der Konzeption einer Datenbank. Oft müssen im Vorfeld später auftretende Daten und ihre Dopplungen abgeschätzt werden. Das setzt ein hohes Erfahrungs-Potential bei den Entwicklern einer Datenbank voraus.
Das Auffinden von Redundanzen wird als **Deduplikation** bezeichnet. Es beeinhaltet das Auffinden und Reduzieren von Dupletten (doppelten Informationen).
Unter Umständen ist aber eine höhere Redundanz in Kauf zu nehmen, wenn dadurch die Verarbeitungs-Geschwindigkeit deutlich verbessert wird. Das viele Nachschauen in untergeordneten Tabellen dauert u.U. zu viel Zeit. In solchen Fällen wird mit gezielten Denormalisierungen dafür gesorgt, dass Daten –Verarbeits-optimal zur Verfügung stehen.

Aufgaben:

1. ***Ist das Paritäts-Bit wirklich eine redundante Information? Begründen Sie Ihre Meinung!***
2. ***Überlegen Sie sich für die folgenden 7-Bit-Sequenzen das Paritäts-Bit bei eingestellter odd-Parität!***

a)

0	0	0	0	0	0	0	

b)

1	0	0	1	1	1	0	

c)

1	1	1	1	1	1	1	

d)

0	1	1	1	0	0	0	

3. ***Überlegen Sie sich für die fehlenden Bit's für 7-Bit-Sequenzen mit eingestellter even-Parität!***

a)

0	0		0	0	0	0	1

b)

1	0	0		1	1	0	1

d)

1	1	1	1	1	1		0

d)

0	1	1		0	0	0	1

3. ***Überlegen Sie sich, welche Art von Fehlern bei der einfachen Paritäts-Prüfung nicht ermittelt werden können!***
4. ***Recherchieren Sie, welche Kontroll-Mechanismen zur Erkennung weiterer Übertragungs-Fehler möglich sind! Stellen Sie eine ausführlich vor!***

für die gehobene Anspruchsebene:

5. ***Auf Disketten wird das CRC-Verfahren zur Absicherung der gespeicherten Daten benutzt. Stellen Sie dieses Verfahren in einem eigenen geschlossenen Text kurz vor! Gehen Sie dabei auch auf die erkennbaren Daten-Fehler ein!***
6. ***Wie groß ist der Speicherverlust durch redundante Informationen bei einer 3,5" 1,44 MB-Diskette Bit-genau?***

2.0.2.2. Daten-Integrität

lat.: integritas (Unversehrtheit, Reinheit, Unbescholtenheit)
meint Korrektheit der Daten und korrekte Funktion des Systems
Schutz vor unberechtigter Löschung und Veränderung bei der Übertragung

Makellodigkeit, Unbestechlichkeit, Unverletzlichkeit

Die Integrität einer (relationalen) Datenbank beinhaltet:

- die Entity-Integrität (Gegenstands-Integrität → Gegenstands-Integrität)
- die referentielle Integrität (→ 2.2.0. Grund-Begriffe, -Elemente und –Verfahren in relationalen Datenbank-Modellen; sowie: → Schlüssel-Integrität)
- sowie die Erfüllung aller anderen Nebenbedingungen und Beschränkungen

fängt schon bei der Wahl passender Daten-Typen zur einem Attribut an

neben Verfügbarkeit und Vertraulichkeit eines der drei Zielrichtungen der Informations-Sicherheit

Überprüfen z.B. mit einer mitgelieferten Prüfsumme
sind die selbst erzeugte Prüfsumme und die mitgelieferte gleich, dann sind die Daten integer.

Definition(en): Integrität
Unter der Integrität von Daten versteht man deren Korrektheit (Fehlerfreiheit und Widerspruchsfreiheit).
Daten-Integrität beschreibt die Korrektheit der Daten.

Schlüssel-Integrität

Primär-Schlüssel einer Relation müssen immer eindeutig und einmalig in einer Relation sein
doppelte Schlüssel sind unzulässig und verletzen die Integrität der Relation

Gegenstands-Integrität

jeder Datensatz / jedes Tupel muss mit einem Schlüsselwert identifizierbar sein; NULL-Werte sind wie nicht vergebene Schlüssel unzulässig und verletzen die Integrität der Relation

Integritäts-Ebenen

- **Wertebereichs-Integrität**
  Integrität auf Datenfeld-Ebene
  alle Attributwerte liegen im gültigen Wertebereich des Attributes (Datenfeldes)
  Attributwert liegt innerhalb der Domäne

- **Datensatz-Integrität**
  Integrität auf Datensatz-Ebene
  hier setzt Normalisierung (→) an

- **referenzielle Integrität**
  Integrität auf der Beziehungs-Ebene
  die Beziehungen zwischen Tabellen besteht auch noch dem Manipulieren der Tabellen
  es existieren keine Verweise auf einen schon gelöschten Datensatz in einer anderen Tabelle
  mögliche Anomalien:
    - Einfüge-Anomalie
    - Änderungs-Anomalie
    - Lösch-Anomalie

Voraussetzungen für eine Beziehung mit referentieller Integrität:

- die zu verbindenden Tabellen (Master- und Detail-Tabelle) befinden sich in der gleichen Datenbank
- beide Tabellen verfügen ein Attribut mit gleichem Datentyp, über dass die Beziehung hergestellt werden kann (beinhaltet den gleichen Attribut-Wert)
- das (Referenz-)Attribut der einen Tabelle (Master-Tabelle) ist ein Primär-Schlüssel (dieser wird in der Detail-Tabelle (unabhängig vom dortigen Primär-Schlüssel) eingetragen)

Herstellen der referentiellen Integrität:

- in einem graphischen DBMS wird die Beziehung von der Detail-Tabelle zur Master-Tabelle gezogen
- bei den Optionen der Beziehung wird mindestens "referentielle Integrität" ausgewählt

weitere Optionen:

- Aktualisierungs-Weitergabe (verhindert die Manipulation des Primärschlüssel-Wertes in der Master-Tabelle; Änderungen werden an die Detail-Tabelle weitergegeben)
- Lösch-Weitergabe (verhindert das Löschen eines Datensatzes aus der Master-Tabelle, wenn in der Detail-Tabelle noch ein Verweis auf diesen vorhanden ist; beim Durchsetzen (Bestätigen) des Lösch-Vorganges werden neben dem Datensatz aus der Master-Tabelle auch alle verweisenden Datensätze in der Detail-Tabelle gelöscht)

Integritäts-Zustände

- **korrekter Inhalt** — passende Abbildung der Realwelt(-Objekte)
- **unmodifizierter Zustand**
- **Erkennung von Modifikationen**
- **temporale Korrektheit**

relationale Integritäts-Regeln

- **physische Integrität** steht für die Vollständigkeit der Zugriffs-Pfade und der physikalischen Speicherstrukturen (nicht vom Datenbank-Programmierer beeinflussbar; Betriebssystem-Ebene)
  Funktionsfähigkeit der Hardware
  verantwortlich: Hardware-Ausstatter, Einkäufer

- **Ablauf-Integrität** Korrektheit der eingesetzten Algorithmen und ablaufenden Programme
  keine Daten-Inkonsistenzen im Mehrbenutzerbetrieb
  verantwortlich sind Datenbank-Designer und Anwendungs-Programmierer

- **Zugriffs-Integrität** korrekte Vergabe von Zugriffsrechten
  korrekte Umsetzung des Rechte-Systems
  verantwortlich: Datenbank-Administrator; Programmierer

- **semantische Integrität** Übereinstimmung der Daten mit der realen Welt durch Überprüfung während der Eingabe
  (z.B. durch enge Begrenzung der Domänen (Werteberei-che))
  verantwortlich: Anwendungs-Programmierer, Datenbank-Adminsistrator, Hersteller des DBS

referentielle Integrität (Beziehungs-Integrität; RI; R.I.)
meint die Aufrechterhaltung von Datenbeziehungen nach / bei Löschaktionen
ein Verweis von einer Tabelle (→ Fremdschlüssel) muss immer auf einen existierenden Quelleintrag (→ Primärschlüssel) verweisen
fehlt der Verweis, dann ist die Integrität der Datenbank verletzt

Problem bei Löschen von abhängigen Daten in mehreren Tabelle ist vorhanden
welche Daten in der einen Tabelle dürfen gelöscht werden, ohne dass in anderen Tabellen Unbeständigkeiten / Falschdaten / Fehler / … auftreten (Lösch-Integrität)
Problem kann zur bestimmten Lösch-Reihenfolge (aus den verschiedenen Tabellen) geklärt werden

besteht aus zwei Teilen:
- ein neuer Datensatz mit einem integrierten Verweis kann nur dann in die DB eingabut werden, wenn der Eintrag auf den der Verweis deutet auch / schon existiert oder ein eindeutiger Alternativ-Schlüssel vorhanden ist (einzutragender Fremdschlüssel muss als Primärschlüssel in anderer (verbundener) Tabelle vorhanden sein)
- die Änderung eines Verweises oder die Löschung eines Datensatzes ist nur dann zulässig, wenn keine Abhängigkeiten (in / zu anderen Tabellen) bestehen (bevor der Primärschlüssel gelöscht oder geändert werden kann, müssen alle seine Nutzungen in anderen (verbundenen) Tabellen geändert / gelöscht werden)

Integritäts-Bedingungen
beschreiben die Vorausssetzungen, die vom System (Daten-Strukturen und Prozess-Abläufe) erfüllt sein müssen, damit das System ordnungsgemäß arbeitet / arbeiten kann
dazu gehören z.B. die eindeutigen Festlegungen und Zuweisungen von Primär- und Fremd-Schlüsseln

Aufgaben:

1. ***Überlegen Sie sich für ein Adressbuch neben Name und Vorname noch fünf andere Attribute! Legen Sie für jedes Attribut Regeln (mindestens eine!) fest, um die Intigrität des (eingegebenen) Datum's zu prüfen!***

2.0.2.3. Daten-Konsistenz

Bei der Konsistenz geht es um die exakte Abbildung der (Real-)Objekte in der Datenbank. Es geht also um die inhaltliche Korrektheit der Daten.

Betrachten wir einige Merkmale von Personen. Aus dem nebenstehenden Formular-Dialog ist die oberste Option eindeutig. Was besagt nun die untere Option über das Geschlecht. Wir gehen hier der Einfachheit halber nur von weiblich und männlich als Geschlecht aus.
Auf den ersten Blick würde man vielleicht sagen "nicht-weiblich" – also männlich.
Aber was ist, wenn das Geschlecht aus irgendeinem Grund noch gar nicht erfasst wurde. Eine nicht-gesetzte Option kann eben auch immer auch Unklarheit der Daten bedeuten.
Aber auch die "Volljährigkeit" könnte uns vor Probleme stellen. Ist die Option z.B. nicht ausgewählt, dann kann das beim letzten Aufruf des Personen-Datenbestandes noich gestimmt haben, aber heute auch noch? Jeder Bearbeiter müsste zur Gründlichkeit und Prüfung all solcher Daten verpflichtet werden. Das klappt niemals, weil eben immer mal Fehler und Unaufmerksamkeiten passieren. Besser ist also eine Tag-genaue Berechnung. Das kostet aber wieder Rechenzeit.
Als Kompromiß könnte man die Kontrolle nur dann durchführen, wenn die Option nicht gesetzt wurde. Eine nachlaufende Prüfung der Daten – vor der Übernahme des Datensatzes in die Datenbank – ist also fast immer angebracht. Der Bearbeiter des Datensatzes kann dann auf Hinweise oder Fehler-Meldungen reagieren. Er hat ja gerade diesen Datensatz geändert.
Ist ein fehlerhafter Datensatz erst einmal im System, ist es schwer nachträglich für Korrektheit zu sorgen

Grad und Art des Zusammenhalts, strenger gedanklicher Zusammenhang

Überprüfung z.B. im direkten Vergleich von zwei Dateien, Datensätzen oder Attribut-Werten usw. usf.
sind beide gleich dann besteht Daten-Konsistenz
Daten-Konsistenz beschreibt

Definition(en): Konsistenz
Unter der Konsistenz von Daten versteht man deren Aktualität, Einheitlichkeit sowie deren exakte logische / funktionelle Abhängig zu anderen Daten.
Daten-Konsistenz beschreibt die Widerspruchsfreiheit der Daten.
Daten-Konsistenz ist ein Ausdruck für die (vollständige,) korrekte und widerspruchsfreie Abbildung der Miniwelt / Realwelt in den Daten der Datenbank.

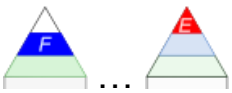
Sind Daten in einem System nicht kosistent, dann sprechen wir von Inkonsistenz des Systems bzw. des Datenbestandes. Diese ist bestmöglich zu vermeiden, da nachträgliche Korrekturen extrem aufwändig und damit teuer sind.

Daten-Konsistenz ist eines der großen Probleme in Datenbanken. Durch kleine menschliche Fehler oder Unkenntnisse kommen schnell leicht abweichende Daten in die Datenbank. So könnte ein Ort erfasst werden. Das soll mal beispielhaft "" sein.
Solange der Ort immer so geschrieben wird, ist alles ok. Was passiert aber, wenn jemand – ganz aus Versehen ein Leerzeichen anhängt. Schon ist ein neuer Ort kreiert, weil für den Computer ein Unterschied besteht.
Oder jemand schreib das Örtchen mit Leerzeichen um den Bindestrich, oder nur auf der einen oder anderen Seite des Bindestriches. In Windeseile existieren in der Datenbank unzählige Schreibweisen für ein und denselben Ort. Für das Datenbank-System sind es unterschiedliche Orte. Solche Fehlschreibungen später zu finden und richtig zu stellen ist auch sehr schwer, wenn nicht gar unmöglich. Häufig sind Daten nämlich über solche Begriffe mit anderen Tabellen verbunden. U.U. müssen erst diese Daten korrigiert werden, bevor man sich an die vorgeordnete Tabelle machen kann.

Konsistenz in relationalen Datenbanken

- **Bereichs-Integrität** die Attribut-Werte müssen innerhalb der definierten Grenzen liegen
- **Entitäts-Integrität** die Primärschlüssel müssen eindeutig sein (ein leeres Feld ist nicht zulässig)
- **referentielle Integrität** die Fremdschlüssel-Felder enthalten entweder einen gültigen Wert (Verweis auf einen Datensatz in einer anderen Tabelle) oder sind leer (NULL)
- **logische Konsistenz** beeinhaltet Regeln zur Abbildung von (logischen) / realistischen Zusammenhängen / Folgen / Beziehungen
    - Eltern sind vor ihren Kindern geboren
    - Personen müssen volljährig sein, um zu heiraten
    - Bearbeitung A muss vor Bearbeitung B erfolgen
    - Genehmigung 2 kann erst nach Genehmigung 1 erfolgen
    - …

Eine Datenbank gilt dann als konsistent, wenn alle Konsitenz-Bereiche erfüllt sind. Ist ein Bereich nicht konsistent, dann ist auch die Datenbank inkonsistent.



Sind Datenbanken z.B. über die Welt verteilt, dann müssen die Zugriffe und Aktionen mit den anderen Teilen abgestimmt werden.
Verteilte Systeme (Daten-Cloud's, Cloud-Computing) werden später noch etwas ausführlicher betrachtet (→).

Konsistenz-Bereiche in verteilten Systemen

- **Client-bezogene Konsistenz (Client-centric Consistency)**	
○ **monotone Lese-Konsistenz (Monotonic Read Consistency)**	aufeinander folgende Lese-Vorgänge müssen sich immer auf die gleiche oder eine neuere Version beziehen
○ **monotone Schreib-Konsistenz (Monotonic Write Consistency)**	aufeinander folgende Schreib-Vorgänge müssen immer in der gleichen Reihenfolge durchgeführt werden
○ **Konsistenz (Read Your Writes Consistency)**	nach dem Schreiben einer Version beschränkt sich das Lesen auf die gleiche oder eine neuere Version
○ **Konsistenz (Write Follows Reads Consistency)**	gelesene und dann geänderte Versionen werden nur auf Replika übertragen, wenn diese mindestens die gleiche Version beeinhalten
- **Daten-bezogene Konsistenz (Data-centric Consistency)**	
○ **kausale / logische Konsistenz (Causal Consistency)**	Operations-Folgen auf dem einen Replikat müssen / werden auf den anderen Replikaten in der gleichen Reihenfolge abgearbeitet
○ **sequentielle Konsistenz (Sequential Consistency)**	betrachtet die exakte zeitliche Folge von Prozessen
○ **(Linearizability)**	beinhaltet sowohl die sequenzielle – als auch die zeitliche – Abfolge

2.0.2.4. Authentizität

Bei den riesigen Daten-Mengen kann heute niemand mehr nachträglich die Quelle oder die Korrektheit von Daten prüfen. Dabei ist es egal, ob die Daten von einem Menschen oder aus dem technischen System stammen.
Gerade in Zeiten von "Fake News" ist es umso dringender die Daten zu ihren Quellen zurückverfolgen zu können bzw. nur solche Daten zuzulassen deren Quelle geprüft wurde.

Der Begriff leitet sich vom altgriech. "authenticus" ab, was für "verbürgt" und "zuverlässig" steht. Ein Ding / Prozess / eine Person ist authentisch, wenn seine "Echtheit" bestätigt bzw. seine Charakterisierung als "Original" möglich ist.

Mitlerweise wird die Authentizität zu den übergeordneten Zielen in der IT-Branche gezählt. Alle anderen Ziele werden sinnlos, wenn nicht die Verlässlichkeit geprüft ist.
In jedem Fall wollen wir also eine eindeutige Zuordnung von Daten zu ihren Quellen. Dabei soll ein Vortäuschen einer Quelle nicht möglich sein.

Definition(en): Authentizität
Unter der Authentizität von Daten versteht man deren sichere (Zweifels-freie) Zuordnung zu einer Quelle oder einem Benutzer.

Durch eine Prüfung der Quelle / des Nutzers vor dem Datenbank-Zugriff wird seine Authentizität bestätigt. Der Vorgang der Prüfung wird Authentifikation bzw. Authentifizierung genannt. Für dieses Prüfverfahren müssen Daten sicher übertragen und ihre Identität bestätigt (verifiziert) werden.
Eine Authentifizierung ist z.B. dadurch möglich, dass eine Person (oder eine technische Quelle) ein bestimmte **Information** besitzt. Das kann ein Passwort oder eine Ziel-Adresse sein. Weiterhin sind der **Besitz von Schlüssel**n (echte Hardware od. moderne kryptographische) oder die **persönliche Anwesenheit** bzw. Personen-Identität (z.B. biometrische Merkmale) als Grundlage für eine Authentifizierung geeignet.

Authentifizierung-Grundlage	Information	Besitz	Gegenwart / Körper-Merkmale
Beispiele	- Passwort - PIN - Sicherheitsfrage - Zero-Knowledge-Beweis (Kenntnis-freier Beweis)	- Chip-Karte - Zertifikat - TAN-Liste - USB-Stick (als Passwort-Tresor) - SIM-Karte - RFID-Karte	- Finger-Abdruck - Gesichts-Erkennung - Tipp-Verhalten - Stimm-Erkennung - Iris-Erkennung - Handschrift-Erkennung - Handflächen- und Handlinien-Erkennung - Hardware-Nummer od.ä. - Netzwerk-Adresse (MAC)
Vorteile	- einfach zu benutzen - keine technischen Mittel notwendig - leicht transportierbar -	- schwerer zu duplizieren - kann als Objekt geschützt werden - ersetzbar bzw. austauschbar -	- ist immer dabei - nicht übertragbar oder duplizierbar -
Nachteile	- kann vergessen werden - kann verteilt oder verraten werden - lässt sich ev. erraten oder ermitteln (Brute-Force-Angriff) - Nutzung schwer zu kontrollieren -	- Verlust, Diebstahl, Weitergabe möglich - Verlust des Objektes bedeutet auch Verlust der Authentizität - Erstellung notwendig und z.T. aufwändig -	- ist öffentliche Information - Hardware und eigentliche Prüfung aufwändig - auch Fälschliche Apzeptanz möglich - auch Zurückweisung trotz Authentizität möglich - ev. Probleme beim Datenschutz -

Häufig werden die Authentifizierungs-Methoden kombiniert. Sehr effektiv ist die Zwei-Stufen- bzw. Zwei-Faktor-Authentifizierung (2FA). Dabei werden eben zwei voneinander unabhängige Verfahren zur Prüfung benutzt. Üblicherweis kombiniert man unterschiedliche Authenitfizierungs-Grundlagen (Information, Besitz, Körpermerkmal).

Beispiel Bankkarte: Zum Geld-Abheben an einem Bank-Automaten oder auch zum Bezahlen benötigt man die Karte selbst und den zugehörigen PIN. Bei bestimmten Bezahl-Systemen reicht auch die Karte und eine Unterschrift.
Beim online-Banking benutzt man die Zugangsdaten (z.B. Kontonummer und Passwort) für das Einlggen in das System. Für Überweisungen müssen entweder mit einem TAN-Generator (setzt Karten-Besitz voraus) eine Überweisungs-Freigabe (TAN-Nummer) generiert. Bei anderen Versionen des TAN-Verfahrens wird eine bestimmte TAN aus einer Liste angefordert oder eine TAN per SMS an ein Mobil-Telefon gesendet.

Je nach der benötigten Schritt-Anzahl unterscheiden wir:

- Ein-Weg-Authentifizierung (Client meldet sich beim Server an)
- Zwei-Wege-Authentifizierung (Client und Server verifizieren sich gegenseitig)
- Drei-Wege-Authentifizierung ()

Aufgaben:

1. *Überlegen Sie sich, welche Authentifizierungen im Laufe eines Tages bei Ihnen möglich sind! Welche Authentifizierungs-Grundlage(n) wird / werden jeweils benutzt und um welche Anzahl Stufen / Wege werden genutzt?*
2. *Welche Authentifizierung wird beim Aufrufen einer https-Webseite angewendet? Charakterisieren Sie das Verfahren in seinem Verlauf!*
3.

für die gehobene Anspruchsebene:

4. *Was versteht man unter dem Byzantinischen Fehler? Was hat der Fehler mit IT-Systemen und der Authentizität zu tun? Recherchieren Sie und bereiten Sie einen kleinen Vortrag vor!*
5.

2.0.2.5. Vertraulichkeit

Die Vertraulichkeit gehört zu den drei CIA-Schutz-Zielen der Informations-Verarbeitung. CIA steht dabei nicht für die Central Intelligence Agency oder die Möglichkeit für deren Mitlesen von Daten, sondern für die Anforderungen **Confidentiality** (Vertraulichkeit), **Integrity** (Integrität) und **Availability** (Verfügbarkeit). Weitere Schutzziele sind Privatsphäre, Nicht-Abstreitbarkeit und Verlässlichkeit.

Daten in seiner Datenbank sollen vor unberechtigten Zugriffen geschützt werden. Niemand möchte sensible Daten frei im Internet verfügbar sehen.
Auch allgemein ist es gewünscht, dass die Daten beim Übertragen durch das Internet von irgendjemand mitgelesen wird.
Zu den vertrauensbildenden Maßnahmen gehören also solche, die nur authentifizierte Zugriffe zulassen und solche, die Daten in verschlüsselter Form vom Server zum Client und zurück übertragen.

Zum Anderen geht es auch um die Durchsetzung der Datenschutz-Bestimmungen, um die schutzwürdigen Personendaten ausreichend abzusichern und vor fremden Augen zu verbergen.

Definition(en): Vertraulichkeit
Unter der Vertraulichkeit von Daten versteht man deren Schutz vor unberechtigter Benutzung bzw. Mitlesen durch Dritte.

mögliche Maßnahmen(-Bereichen):

- Zugangs-.Kontrolle
- Zugriffs-Kontrolle (restriktiver Datenzugriff)
- Verschlüsselung

Gefährdungsbereiche des IT-Grundschutz:

- **organisatorische Mängel**
  - **mangelhafte Auswertung von Protokolldaten**
  - **mangelhafte Test- und Freigabe-Verfahren**
  - **mangelhafte Aktivierung von Datenbank-Sicherheits-Mechanismen**
  - **mangelhafte Konzeption des DBMS**
  - **mangelhafte Konzeption des Datenbank-Zugriffs**
  - **unzureichende Speicher-Medien für Notfälle**
  - **mangelhafte Organisation bei Migrationen und Versions-Wechseln**
- **menschliche Fehlhandlungen**
  - **mangelhafte Adminstration von Zugriffs- und Zugangs-Rechten**
  - **mangelhafte Administration des DBMS**
  - **Gefährdung durch Reinigungs- und Fremd-Personal**
  - **unbeabsichtigte Daten-Manipulation**
  - **fehlerhafte Synchronisation von Datenbanken**
- **technisches Versagen**
  - **Ausfall der Datenbank**
  - **Unterlaufen von Zugroffskontrollen (über ODBC)**
  - **Daten-Verlust einer Datenbank**
  - **Verlust der Daten-Integrität und -Konsistenz**
- **vorsätzliche Handlungen**
  - **unberechtige IT-Nutzung**
  - **Missbrauch der Fernwartung**
  - **systematisches Ausprobieren von Passwörtern (Brute-Force-Angriff)**
  - **Manipulation von Daten oder der Software in Datenbank-Systemen**
  - **Ver- od. Behinderung von Datenbank-Diensten**
  - **SQL-Injektion**

Q: nach Bundesamt für die Sicherheit in Informationssystemen (bsi.de)

2.1. Datenbank-Modellierung

Der Anfänger, der glaubt besonders schnell und effektiv zu arbeiten, beginnt gleich mit dem Eingeben der Daten in den Computer.
Ein häufig beobachtetes Phänomen in der Welt der Computer-Nutzer, die erst im Laufe ihres Berufslebens mit dem Computer vertraut geworden sind, ist, dass sie ein Programm favorisieren, dass scheinbar ihre Computer-Berufs-Welt am Besten repräsentiert. Dieses Programm wird dann für alles benutzt. Ob andere Programme für ihre Probleme besser geeignet sind, wird gar nicht hinterfragt.
Ich habe Nutzer beobachtet, die sich super in EXCEL auskannten. Sie waren der Naturwissenschaft und Mathematik beruflich immer sehr nahe. Selbst Geschäftsbriefe wurden von diesen Personen mit EXCEL erledigt. Dass sie WORD direkt daneben auf ihrem Computer hatten, wurde einfach ignoriert. Selbst nach Hinweisen auf die doch sehr gute Textverarbeitung wurde das Programm wegen der "fehlenden" Raster
Der Einstieg in ein neues Programm oder ein neues Thema ist immer mit Mehraufwand verbunden. Diesen Aufwand darf man nicht scheuen, er zahlt sich spätestens beim zweiten oder dritten Projekt schon wieder aus.
In der Praxis hat die folgende Sequenz von Prktiken als sehr effektiv erwiesen. Abweichungen bringen – auch schon bei kleineren Projekten – Schwierigkeiten und Verzögerungen mit sich. Ganz häufig sind die Ergebnisse kaum (breit und effektiv) nutzbar. Außer Speesen ist dann nichts gewesen.

Praktiken

- **Modellierung**
  - Planung des Projektes
  - Erfassung der Bedarfe
  - Prüfen der Ressourcen
  - Festlegen von Schnittstellen
- **Implementierung**
  - Umsetzen des / der Modell(e) in ein Computer-System
  - Prüfen der Funktions-Fähigkeit mit Spiel-Daten
  - Testen der Schnittstellen
  - Erfassen der Real-Daten
- **Optimierung**
  - Beobachtung / Überwachung des Systems
  - Verbesserung der Daten-Strukturen (wenn das noch geht)
  - Optimierung von Prozess-Abläufen
  - Beschleunigung der Funktionen
  - Erweiterung des Systems mit neuen Funktionen

Wir gehen nun im Folgenden nach der klassischen Methodik vor und beginnen mit der Modellierung (→ 2.x.1. Datenbank-Modellierung). Hier werden wir auch verschiedene Modellierungs-Werkzeuge besprechen.
Die Implementierung der Datenbank wird sich dann anschließen (→ 3. Datenbanken – Implementierung). Hier wird man sich als Leser oder im Kurs entscheiden müssen, welches Datenbank-Management-System benutzt werden soll. Ich empfehle sich auch aml andere Implementierungs-Möglichkeiten anzuschauen. Interessant sind dabei vielmehr die Gemeinsamkeiten (→ SQL: 5. SQL – die Datenbank-Sprache) als die Unterschiede.
Mit der Optimierung werden wir uns im Rahmen dieses Kurses nicht beschäftigen können. Zum Einen sind unsere Projekte zu klein, zu theoretisch und haben auch kaum realen Dauer-Betrieb. Die Optimierung von Datenbank-Systemen ist echte Profi-Arbeit.

2.1.1. Datenbank-Modelle

Bei der Datenbank-Modellierung können wir den Computer beruhigt ausschalten und uns ganz auf unser Problem konzentrieren. Lediglich als Werkzeug zum Schreiben von Texten oder dem Erstellen von (digitalen) Skizzen werden wir den Computer brauchen.
Die Notwendigkeit zur Modellierung einer Datenbank im Vorfeld der Daten-Eingabe und -Nutzung ergibt sich schon deshalb, damit wir ein(e) …:

- sinnvolle Einschränkung der realen Welt auf eine Modellwelt (→ Miniwelt) zu erreichen, z.B. um die Datenflut einzugrenzen
- Effektivierung der Daten-Eingabe, -Verwaltung und –Nutzung erzielen
- Anpassung an die Bedingungen für die konkrete / hauptsächliche / vorrangige Daten-Bereitstellung und –Nutzung möglich machen
- Verwaltbarkeit der Datenbank bzw. der Daten durchsetzen zu können
- gesetzliche und betriebliche Vorgaben des Datenschutzes und der Datensicherheit einzuhalten
- optimales Daten-Modell auszuwählenden
- schnellere (ev. externe) Programm-Entwicklung zu ermöglichen

Die verwendeten Modelle für die Beschreibung und Konzeption von Datenbanken ist im wesentlichen von der Struktur der Daten bestimmt. Die möglichen Daten-Strukturen stellen wir nachfolgend vor. Für die Praxis dieses Kurses ist das relationionale Modell dominierend.

Arten von Datenbank-Modellen / Datenbank-Arten

- (flaches) **Datei-Modell** — alte, einfache Daten-Sammlungen und Datenbanken (→ 1.1.1. Daten in Dateien)
- **relationales Modell** — Tabellen-orientierte Datenbank-Systeme (→ 2.x. relationales Daten(bank)-Modell)
- **hierachisches Modell** — Datenbank-Systeme mit hierrarchischen oder Baum-artigen Daten-Strukturen (→ 2.x.1. hierarchisches Datenbank-Modell) (→ 2.x.2. baumartiges Daten(bank)-Modell)
- **Netzwerk-Modell** — verteilte oder Netz-artige Datenbank-Systeme (→ 2.x.3. netzwerkartiges Daten(bank)-Modell)
- **Objekt-orientiertes Modell** — Datenbank-System mit Objekt-orientierten Daten-Strukturen und Datenbank-Management-Systemen (→ 2.x.4. objektorientiertes Daten(bank)-Modell)
- **Dokument-orientiertes Modell** — Datenbank-Systeme mit Journal- oder Dokumenten-orientierter Daten-Struktur (→ 2.x.5. dokumentenorientiertes Daten(bank)-Modell)

ev. Mischformen, wie

- **objektrelational** — Datenbank-Systeme mit Tabellen-artig angelegten und gespeicherten Daten-Strukturen (→ 2.x.4. objektorientiertes Daten(bank)-Modell)

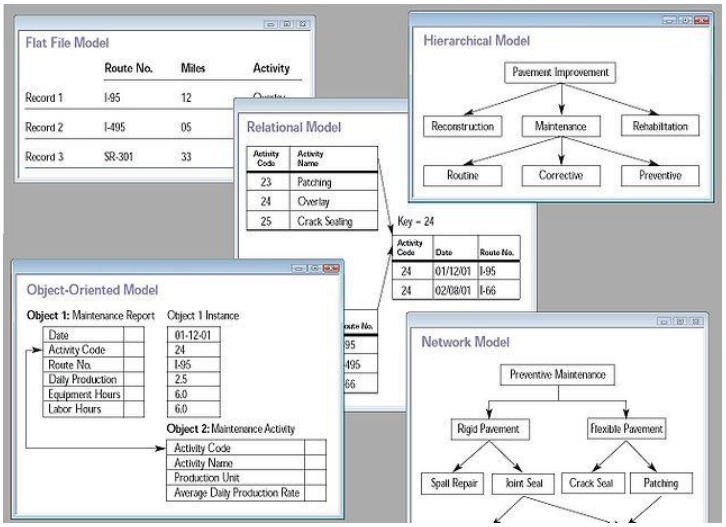
Die anderen Modelle sind aber für das Verständnis von informatischen Strukturen und den Konzepten moderner Datenbanken ebenfalls interessant und besprechenswert.

Definition(en): Datenbank-Modell
Ein Datenbank-Modell ist eine Sammlung von Strukturen und deren Beziehungen untereinander zur Charakterisierung eines Datenbank-Systems.
Ein Datenbank-Modell ist die Beschreibung der Daten und ihrer Strukturen in einem Datenbank-System.
Ein Datenbank-Modell ist die Darstellung / Veranschaulichung der logischen Struktur einer Datenbank.
Ein Datenbank-Modell ist die Zusammenführung von Konzepten zur Beschreibung der Struktur einer Datenabnk.

Daten lassen sich in vielen Strukturen darstellen. Diese Strukturen sind Modell-bestimmend.

Die Struktur einer Datenbank schließt die Daten-Typen, Bedingungen und deren Beziehungen untereinander mit ein.
Weiterhin gehören die generischen Dienste der modellierten Datenbank mit in die Beschreibung.

Die Semantik eines Datenbank-Modells ähnelt stark der Semantik einer Programmiersprache. Sie enthält z.B. Elemente (Deklarationen) der Datenbank-Sprache.



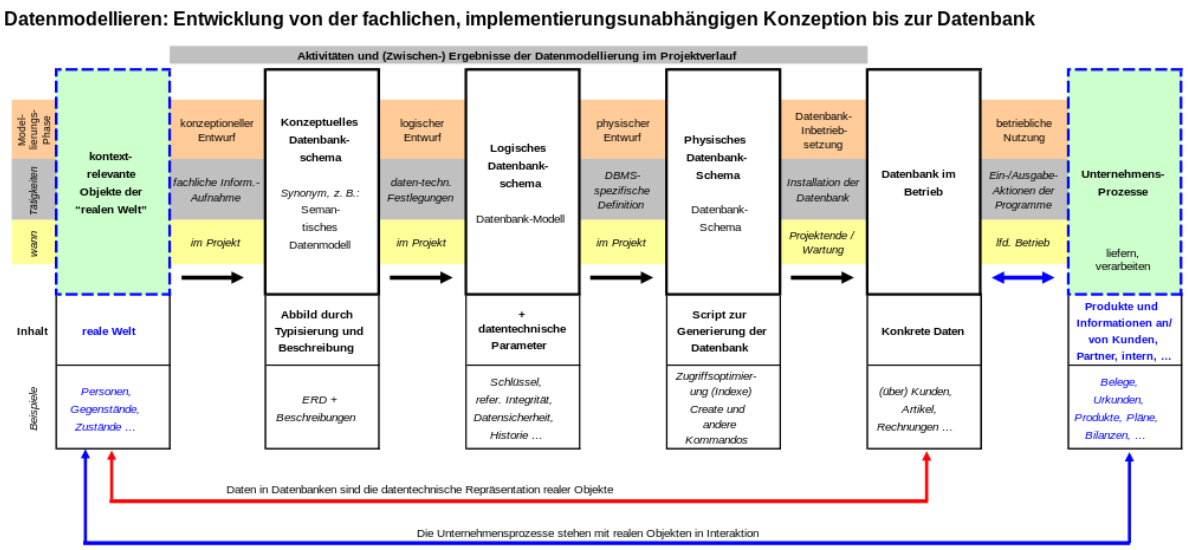
Beispiele für Datenbank-Modelle
Q: de.wikipedia.org (Marcel Douwe Dekker)

Die Aufgabe der Daten-Modellierung ist die eindeutige Definition der zu verarbeitenden Daten-Objekte. Besonderes Auganemerk wird dabei auf die Attribute gelegt, die in den folgenden Speicher- und Verarbeitungs-Prozessen eine Rolle spielen sollen.
Das Ergebnis der Daten-Modellierung ist ein Daten-Modell. Dieses Modell wird dann mehrstufig in eine funktions-fähige Datenbank umgesetzt.
Daten-Modelle besitzen eine besondere Bedeutung im Entwicklungs-Prozess einer Datenbank. Da die Daten – wenn sie denn einmal festlegt sind – ein sehr stabiles, schwer zu veränderndes Element darstellen. Die begleitenden Verbindungen und vor allem die programmierten Funktionen (Methoden) sind veränderlich und können i.A. unproblematisch angepasst und aktualisiert werden. Selbst völlig neue Verwendungen der Daten sind möglich. In der Brache wird das durch den Grundsatz "Data is stable – functions are not." (Daten sind stabil – Funktionen sind es nicht.) ausgdrückt.
Dieses Prinzip wird schon durch die von CODD definierten Eigenschaften(-Ebenen) eines Datenbank-Modell wiedergegegen.

definierende Eigenschaften eines Datenbank-Modells (nach Edgar F. CODD)

1. generische Daten-Struktur	beschreibt die Struktur der Datenbank; generisch, weil die Relationen und Attribute frei wählbar sind
2. generische Operatoren	Menge von Operatoren bezüglich der benutzten Daten-Struktur; dienen zum Einfügen, Ändern, Abfragen und Verknüpfen der Daten
3. Integritäts-Bedingungen	beschreiben die Vorschriften und Regeln, die für eine sichere, Daten-konsitente und Bestimmungs-bedingte Nutzung notwendig sind

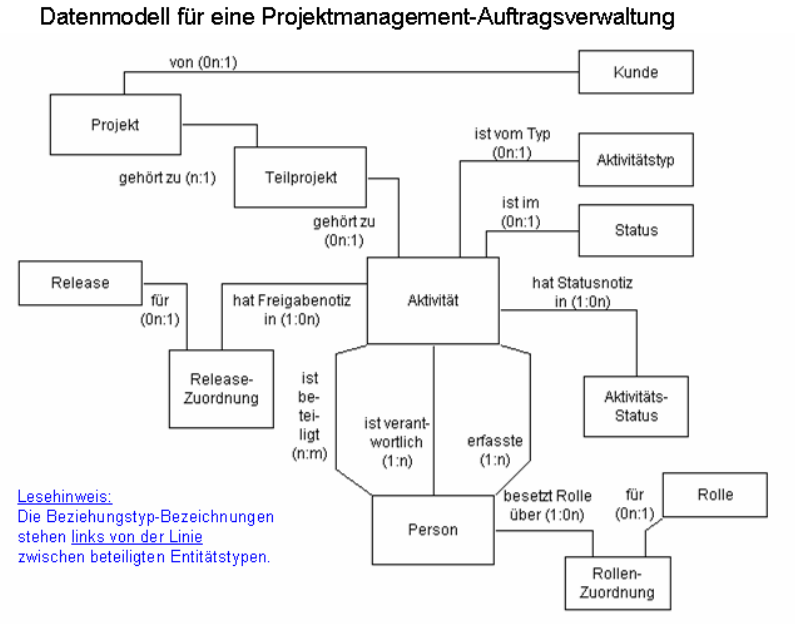
Als Fluß-Diagramm kann man sich die Erstellung einer Datenbank aus einem Entwurf in etwa so vorstellen:



Daten-Modellierung - vom fachlichen Entwurf zur Datenbank
Q: de.wikipedia.org (VÖRBY), gemeinfrei

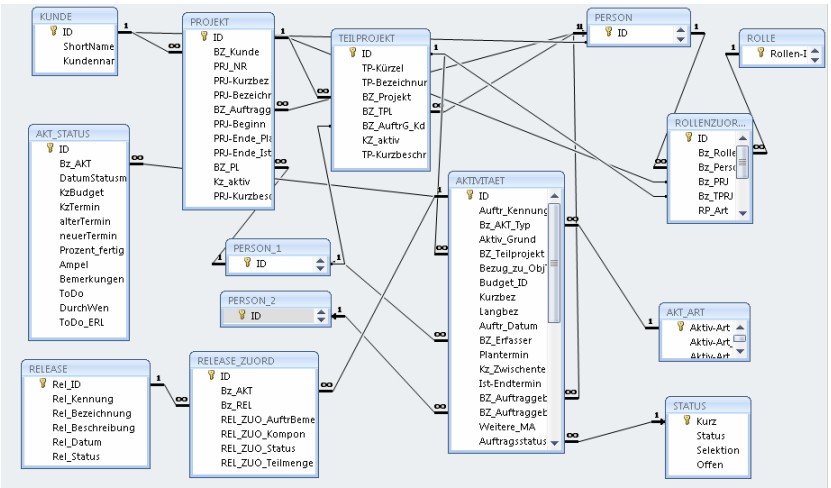
Definition(en): Daten-Modellierung

Daten-Modellierung ist das Verfahren zur formalen Abbildung der Objekte einer Mini-Welt mittels ihrer Attribute und Beziehungen.



semantisches Modell (einer Auftrags-Verwaltung)
Q: de.wikipedia.org (VÖRBY), gemeinfrei

Als Zwischenstufe kommt hier das Datenbank-Modell zum Tragen.



Datenbank-Schema zum obigen Modell (Auftrags-Verwaltung)
Q: de.wikipedia.org (VÖRBY), gemeinfrei

2.1.1.1. Datenmodell

Definition(en): Daten-Modell
Das Daten-Modell ist die Defintion der zu bearbeitenden und verarbeitenden Daten eines Anwendungs-Bereiches.
Ein Daten-Modell ist eine Sammlung aus Sprachen zur Definition (Datendefinitionssprachen), Zustandsänderungen (Anfragesprachen), Manipulationen (Manipulationssprachen) und Integritätsbedingungen von Daten.

2.1.2. übergreifende / übergeordnete Modelle

3-Ebenen-Modell

Drei-Schema-Architektur,

folgte aus dem 4-Ebenen-Modell DIAM von Michael SENKO (1973)
DIAM steht für Data Intependent Accessing Model

die Begriffe Ebene, Schicht und Sicht werden praktisch äquivalent benutzt
von Ebenen und Schichten spricht man eher, wenn es um das Objekt geht
soll mehr die Arbeit von uns Menschen mit der Ebene / Schicht betont werden, dann wird der begriff Sicht bevorzugt

Data Intependent Accessing Model (DIAM) von SENKO (1973)

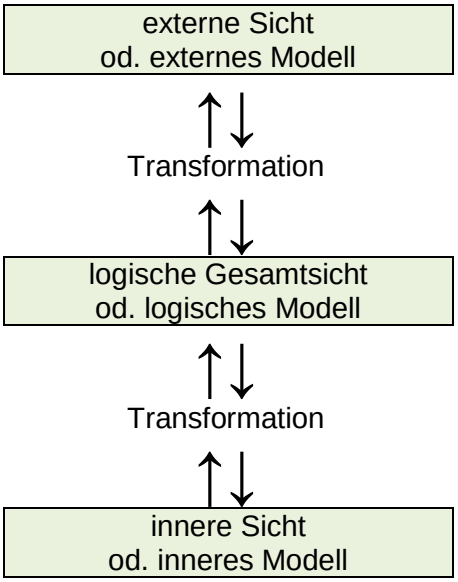
- **entity set model** — logische und anwenderneutrale Datenstrukturen (Mengen von Objekten, die vom DBMS verwaltet werden)
- **string modell** — logische Zugriffspfade (z.B. Indexe, Hash-Tabellen, …)
- **encoding model** — Speicherungsstrukturen; physischer Zugang zu den Daten und Datenstrukturen (Abbildung der Daten in speziellen Strukturen (Baum, Tabelle, …))
- **physical device model** — Speicherordnungsstrukturen; Zuordnung der Daten ins das Dateisystem und den Datenträgern

wegen der Verwechslungsgefahr mit dem 3-Ebenen-Modell (zur Beschreibung menschlichen Verhaltens in Mensch-Maschine-Systemen, 1980) von Jens RASMUSSEN häufig ausführlich als ANSI-SPARC-Architektur bzw. ANSI-SPARC-Modell bezeichnet

Die ANSI (American National Standards Institute) ist die amerikanische Standardisierungs-Institution. Das SPARC (Standards Planing and Requirements Committee) ist eine Abteilung innerhalb der ANSI.

Beschreibung und Definition von drei Ebenen für Datenbank-Schemata aus dem Jahr 1975

ANSI-SPARC-Modell



Nutzer / Anwender
Sicht des Nutzers auf die Daten (Formulare, Masken-Layouts, Listen, Sichten, Schnittstellen)

logische Unabhängigkeit

Datenbank-Entwickler (-Ersteller)
Beschreibung der Zusammenhänge zwischen den Daten

konzeptionelle Unabhängigkeit

Programmierer
Realisierung der Datenstruktur auf der DV-Anlage (z.B. als Baum oder Tabellen)

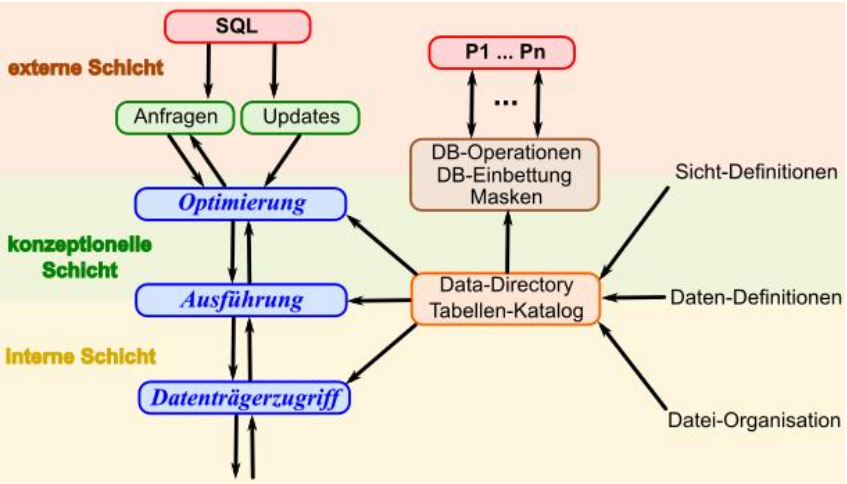
je nach Ebene gibt es unterschiedliche Verantwortliche:
- Anwendungs-Adminstrator
- Unternehmens-Administrator
- Datenbank-Adminstrator

die innere Sicht praktisch in der technischen Informatik angesiedelt
Aussagen dazu, wie werden die Zeichen schnell und sicher gespeichert, gelesen, geschrieben, …
diese Ebene kann beliebig realisiert und ausgetauscht werden
vielleicht braucht man irgendwann andere Arten von Datenträgern, auf die andersartig zugegriffen werden muss, dann wird dies durch die interne (innere) Ebene umgesetzt
auch die Verteilung einer Datenbank auf mehrere Datenträger wird hier organisiert; dies ist für die Daten-Zusammnehänge und die Nutzer völlig uninteressant
hier wird vielleicht auch die Vermeidung von Redundanz, die wir eigentlich anstreben durch die technischen Vorgänge von Backup's usw. wieder aufgehoben
aus Datenschutz-Gründen möchten wir hier schon Redundanzen haben, die inner halb der Datenbank unerwünscht sind

die konzeptionelle Ebene ist von solchen Vorgängen unbeeindruckt, sie beschreibt die Zusammenhänge zwischen den Daten

der Entwickler legt hier das Datenbank-Design fest, er bestimmt wie die Daten gespeichert werden sollen 8Daten-Typ) miteinander und wie die Daten miteinander verbunden sind (Relationship)
Definition von Tabellen, Optimierung der Tabellen (→ Normalisierung)
Inhalt und Zweck der Datenbank bestimmen das Arbeiten in diesem Bereich

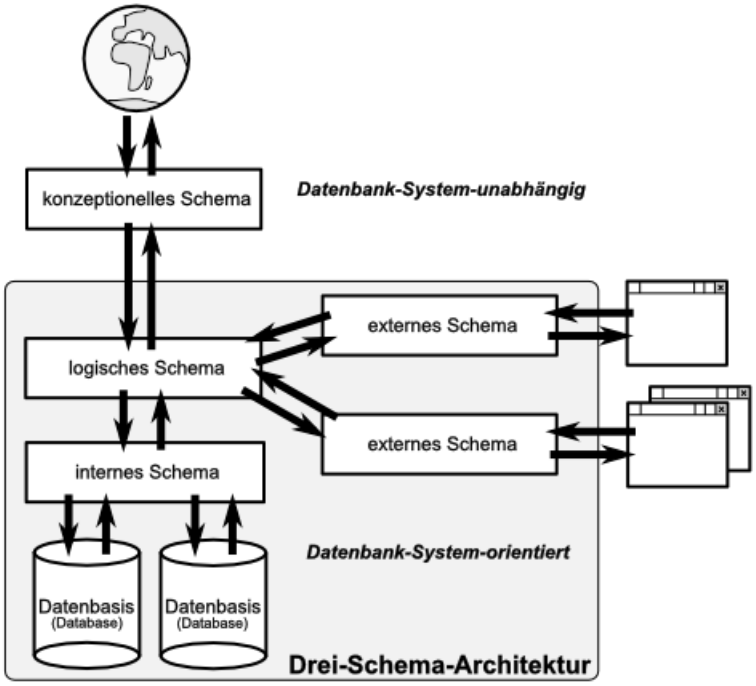
die externe Schicht ist für die Kommunikation mit dem Nutzer verantwortlich
das sind zum Einen SQL-Anweisungen von Anfragen an die Datenbank oder Übertragungen von Daten an diese (→ Daten-Updates)
zum Anderen können hier Kommunikationen zwischen Anwendungs-Programmen und der Datenbank realisiert werden. Neben SQL gibt es auch noch andere Operationen / Anweisungen / Befehle die in bestimmten Programmiersprachen die Kommunikation mit der Datenbank ermöglichen.



die konzeptionelle Unabhängigkeit ermöglicht die ungebundene Konzeption und Entwicklung einer Datenbank
wie das Datenbank-System das konkret umsetzt, wie genau die Daten in Bits und Bytes umgesetzt werden und wie sie auf dem Datenträger organisiert sind, das ist die eigenständige Aufgabe der internen Schicht
irgendwann wird vielleicht mal von 32 bit auf 64 oder auch 128 Daten-Einheiten umgestellt
dies ist erst einmal für die konzeptionelle Schicht egal
natürlich ergeben sich neue Möglichkeiten, die stören aber die bestehenden Datenbanken i.A. nicht

die logische Unabhängigkeit beschreibt die jeweilige Eigenständigkeit der externen und der konzeptionellen Schicht.
sollen neu(artig)e Daten hinzugefügt werden, dann kann das unabhängig in der konzeptionellen Schicht angelegt werden
die meisten Anwendungen / Nutzungen in der externe Schicht wird das wahrscheinlich nicht betreffen
nur neu(artig)e Abwendungen / Dialoge / Abfragen / …, die sich eben mit den neu(artig)en Daten beschäftigen müssen dann (unabhängig) angepasst
werden die neu(artig)en Daten in irgendwelchen Abfragen nicht gebraucht, dann werden sie praktisch ignoriert, obwohl sie in den betroffenen Tabellen sehr wohl vorhanden sind

Das Drei-Schema-Modell ist die Erweiterung des klassischen Schichten-Modell's. Durch die neue Drei-Schema-Architektur sind zusätzliche Betrachtungen und Sichtweisen auf Datenbank-Systeme möglich.
Mit den externen Schemen wird die unterschiedliche Sicht der Anwendungen und Nutzer auf die Daten beschrieben.
Das interne Schema dient der Beschreibung der Inhalte in der Datenbasis sowie die weiterhin benötigten Services. Mit Hilfe dieses Schemas werden die Daten aus technischer Systemsicht betrachtet.
Jede Datenbank hat somit immer ein logisches und ein internes Schema.



Die Anzahl der externen Schemen wird durch die Nutzungs-Möglichkeiten der Daten bestimmt.

externe Ebene	**Benutzer** Anwendungen, View's, Datenschutz, Datensicherheit	Sicht des Nutzers auf die Daten (Formulare, Masken-Layouts, Listen, Sichten, Schnitstellen)
konzeptuelle Ebene	**DB-Adminstrator** logischer Entwurf Daten-Modell	Definition des globalen Gesamtmodells der Miniwelt
interne Ebene	**DB-Entwickler** physischer Speicher Zugriff, Indizierung, Optimierung	Wo und wie sind die Daten physisch abgelegt und wie wird der Zugriff auf sie effektiviert?

Phasen der Datenbank-Entwicklung

Phasen		Ergebnisse
externe Phase	**Informationsbeschaffung** Ermittlung der Informations-Struktur und des Informations-Bedarf's	Festlegung der **Miniwelt** Spezifikations- und Anforderungs-Analyse
↑↓		
konzeptionelle Phase	**semantisches Modell** formale Beschreibung des betrachteten Sachverhalt's	**Entity Relationship Modell** Fachkonzept der Datenbank
↑↓		
logische Phase	**logisches Daten-Modell** Übertragung des semantischen Modell's in ein Daten-Modell (z.B. das relationale)	Transformation des konzeptionellen Modell's in ein **Datenbank-Schema** Optimierung der Tabellen-Daten-Struktur → **(optimiertes) Relationen-Schema**, z.B. durch Normalisierung
↑↓		
physische Phase	**Datenbank-Schema**	Implementierung → **leere Datenbank (Datenbank-System)** Übersetzung des logischen Modell's in ein Datenbank-Schema mittels einer Definitions-Sprache

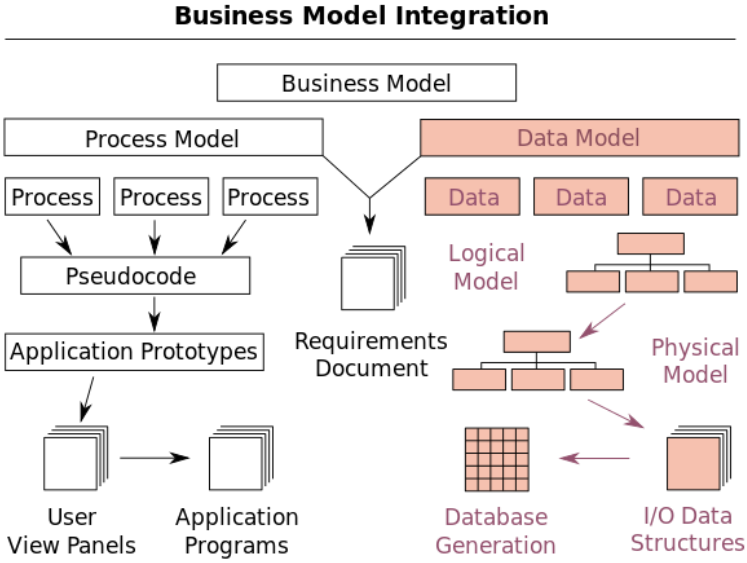
begrifflich nicht sauber getrennt
viele Begriff-Welten parallel
Orientierung aber gut an den Adjektiven des Modell-Ebenen möglich, die sind nämlich immer gleich. Es gibt also eine Konzeption, eine Logik und eine Physis.

Drei-Schemen-Modell

drei Modell-Ebenen

Drei-Ebenen-Modell

Daten-Modell-Typen meint ebenfalls die Ebenen



Zusammenwirken von Daten-Modell-Typen
Q: de.wikipedia.org (Paul R. Smith)

Arten und Ebenen von Daten-Modellen

- **konzeptionelles Datenbank-Schema** (Daten-Modell, Objekt-Modell, semantisches Daten-Modell)
  Implementations-unabhängiges Modell, z.B. als Entity-Relationship-Modell od. UML-Diagramm; modelliert Objekte der Realität mit deren Eigenschaften und Beziehungen
- **logisches Datenbank-Schema** (logisches Daten-Modell)
  Transformation / Abbildung / Umsetzung des konzeptionellen DB-Schema auf die Vorgaben des verwendeten DBMS (Vorbereitung der konkreten Implementation)
- **physisches Datenbank-Schema** (Daten-Schema, Datenbank-Schema)
  beschreibt die Vorgaben / Umgebungsbedingungen / Festlegungen für den technischen Betrieb (z.B. Indizierung, Replikation, Verteilung, …) (für Nutzer unsichtbar)

Das konzeptionelle Modell (Schema) hat praktisch noch keinen Bezug oder eine Kenntnis vom Ziel-System. Ganz im Gegenteil, auf dieser Ebene soll noch völlig frei geplant und modelliert werden. Man kann sich also völlig frei Gedanken über die Daten, Strukturen und Leistungen der neuen Datenbank machen. Ensprechend offen sich die Modellierungs-Werkzeuge, zu denen z.B. das Entity-Relationship-Diagram (ERD; →) gehört. Natürlich könnte man sich auch anders behelfen, aber so ist eine Kommunikation mit anderen Beteiligten auf der Grundlage einer standardisierten Modell-Sprache möglich.
Erst im logischen Modell muss eine Umsetzung auf das Ziel-DBMS erfolgen. Ev. muss man sich aufgrund des Daten-Modells für ein anderes Ziel-System entscheiden oder eben vielleicht mit Einschränkungen leben.
Das physische Schema hat für uns einfache Datenbank-Nutzer keine Bedeutung. Selbst wenn wir uns die Dateien ansehen, werden wir aus diesen nicht viel Nutzen ziehen können. Die Programmierer des DBMS haben irgendwelche Verfahren und informatische Strukturen

gewählt, um aus ihrem System die maximale Leistung herauszuziehen. Die physische Ebene ist für den Datenbank-Nutzer praktisch unsichtbar. Die Betreuung obliegt nur dem technischen Personal.

Definition(en): konzeptionelles Daten-Schema
Das konzeptionelle Daten-Schema ist eine System-unabhängige Daten-Beschreibung. Sie ist unabhängig vom (später eingesetzten) Datenbank- und Computer-System.

Wir werden das sogenannte Entity-Relationship-Modell als konzeptionelle Daten-Beschreibung verwenden. Die in diesem Modell etaplierten Entity-Relationship-Diagramme (ERD; →) sind ein Quasi-Standard.

Definition(en): logisches Daten-Schema
Das logische Daten-Schema ist die Beschreibung der Daten in einer speziellen Daten-Beschreibungs-Sprache (DDL (Data Definition Language)). Die Daten-Beschreibungs-Sprache ist vom verwendeten Datenbank-Management-System abhängig.
Das logische Daten-Schema ist die DBMS-abhängige, formale Darstellung / Modellierung / der Daten-Strukturen.

Da wir in diesem Kurs fast ausschließlich mit relationalen System arbeiten, ist für uns die Datenbank-Sprache SQL (→) das Mittel der Wahl.

Definition(en): physisches Daten-Schema
Das physische Daten-Schema ist

Mit der physischen Ebene kommen der normale Datenbank-Nutzer und wir als Datenbank-Ersteller praktisch nicht in Kontakt.
Lediglich wenn wir mal Datenbank-Dateien austauschen (replizieren), dann ist das eine Arbeit auf der physischen Ebene. In professionellen System ist das nicht angebracht, da dort die Datenbanken ständig auf ihre Daten-Dateien zugreifen, bzw. diverse Daten (zuersteinmal nur) im Arbeits-Speicher vorliegen.
Man kann von außen nicht erkennen, was das Datenbank-System auf der physischen Ebene macht. Selbst wenn es keine Daten speichert oder Abfragen erledigt, kann es sein, dass Backup gefahren werden oder Daten-Bestände im Hintergrund geprüft oder indiziert werden.
Also Hände weg von Dateien aus laufenden Datenbank-Systemen!

Aufgaben:

1. *Analysieren Sie die nachfolgenden Veränderungen / Situationen und ordnen Sie diese einer Ebene oder einer Transformation zu! Erläutern Sie die Zuordnung!*
   - *a) Übertragung einer Datenbank auf ein neues Datenbank-System*
   - *b) Übertragung der Datenbank auf einen neuen Rechner mit dem gleichen Betriebs- und Datenbank-System*
   - *c) Einführung eines neuen Anwendungs-Programm's für die bestehenden Daten*
   - *d) Erweitern der Datenbank um neue Detail-Daten*
   - *e) Zusammenführen von zwei Tabellen zu einer*
   - *f) Übertragen der Datenbank auf einen neuen Rechner mit einem neuen Betriebs- und Datenbank-System*
   - *g) Anpassen des Datums-Formates auf 4stellige Jahres-Angaben*
   - *h) Wechseln der Festplatten des Rechners (nach Ausfall)*
   - *i) Aufteilen von Spalten einer Tabelle auf zwei neue Tabellen*
   - *j) Aufteilen desDatenbestandes auf zwei unabhängige Datenbanken*
   - *k) Ersetzen einer Spiegel-Festplatte nach einem Crash*
2. *Erläutern Sie, was man in der Informatik unter Modellbildung / Modellieren versteht!*
3.

2.1.3. das Entity-Relationship-Modell (ERM)

kurz ERM
deutsch: **Gegenstands-Beziehungs-Modell**
heute de-facto-Standard

Modell wurde ursprünglich von P.P. CHEN (1976) veröffentlicht und in der Folgezeit nur geringfügig / in Details verändert / weiterentwickelt

dient der Umsetzung der umgangssprachlich formulierten Zielstellungen (externe Sicht im ANSI-SPARC-Modell (→ 3-Ebenen-Modell)) der Auftraggeber / Nutzer einer Datenbank in ein informatisch nutzbares Modell (logische Sicht / konzeptionelle Ebene)

Ein ERM ist ein semantisches Daten-Modell. Es orientiert sich im Wesentlichen an den Inhalten und deren Verbindungen zueinander. Mit anderen Worten, ein ERM besteht aus zwei Strukturierungs-Konzepten. Das sind zum Einen die Entity-Typen und zum Anderen die Realtionship-Typen.

Als graphische Darstellung der Objekte (Gegenstände) und deren Merkmale / Eigenschaften und deren Beziehungen zueinander und der Beschreibung der Bedeutungen (Semantik) und der Struktur wird vorrangig das Entity-Relationship-Digramm (→ 2.1.4. Entity-Relationship-Diagramme (ERD)) benutzt.

Begriffe des ERM

- **Entität (entity) (Gegenstand, Objekt)**	individuell, identifizierbares Objekt der Realität z.B.: Herr Müller; ein bestimmtes Buch; das Projekt 17a; … entspricht dem informatischen Objekt-Begriff
- **Beziehung (relationship) (Verbindung)**	Verknüpfung / Zusammenhang zwischen 2 (od. mehr) Entitäten z.B.: Frau Müller besitzt ein bestimmtes Buch; Herr Fritz leitet das Projekt 17a; …
- **Eigenschaft (attribute) (Attribut, Merkmal)**	Merkmale / Charakteristika der Entität z.B.: Herr Müller ist im Projekt seit 01.05.2010 (Eintrittsdatum), hat die Gehaltsgruppe 5; fährt den Dienstwagen 8; …
- **Entitäts-Typ (Klasse)**	Typisierung gleichartiger Entitäten z.B.: (alle) Angestellten; Bücher; Projekte; … entspricht dem informatischen Klassen-Begriff
- **Beziehungs-Typ ()**	Typisierung gleichartiger Beziehungen z.B.: Angestelter leitet Projekt; Buch stammt vom Verlag
- **Attribut(s-Typ) (Domäne)**	Typisierung gleichartiger Eigenschaften z.B.: Name für Angestellte; Bezeichnung für Projekt Texte, Zahlen, Wahrheitswerte, Bilder, …

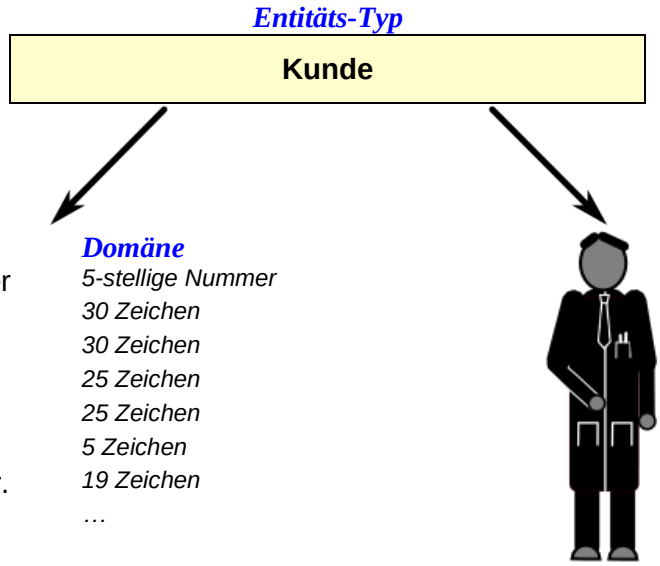
Üblich ist bei semantischen Modellierungen ein Top-Down-Ansatz. Dabei werden die Informationen / Elemente aus der Mini-Welt zuerst einmal stark abstrahiert, um dann im Folgenden immer konkreter zu werden.
Reine Bottom-Up-Modellierungen sind weniger effektiv, da sie ev. schlechter zusammenzuführen sind, um die gewünschte höchste Abstraktions-Ebene zu erreichen.
Erfahrene Modellierer kombinieren Top-Down- und Bottom-Up-Vorgehensweisen. Dabei nutzen sie vor allem ihren breiten Erfahrungs-Schatz.

CHEN verstand unter einer Entität etwas, was klar und eindeutig bestimmt / identifizierbar ist.
Zuerst fallen uns dazu sicher die typischen Gegenstände in unserem Leben, wie Auto's, Haustiere, Häuser, Bücher, Haushalts-Geräte usw. ein. Im informatischen Gebrauch werden aber auch Personen, abstrakte Begriffe bzw. Konzepte oder Ereignisse als Entität benutzt.
In der Programmierung würden wir für Entitäten den Begriff Objekt benutzen. Die Begriffs-Anwendungen verschwimmen in der informatischen Praxis aber häufig.
Wichtig ist vor allem, dass sich zwei Entitäten durch mindestens ein Merkmal – hier Attribut genannt – unterscheiden lassen müssen. Im Normalfall haben die Entitäten eine Vielzahl von Merkmalen. Ob eine Erfassung in einem Modell – und später in der Datenbank – wirklich sinnvoll ist, wird in der Entitäts-Analyse entschieden.

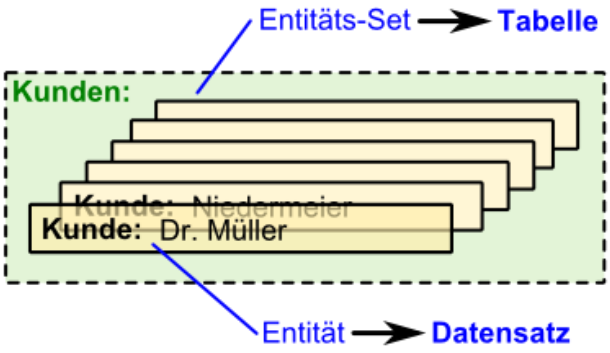
Definition(en): Entität / Entity
Eine Entität ist ein individuelles und / oder identifizierbares Exemplar eines Dinges, einer Person oder eines Begriffes aus / der realen Welt.
Eine Entität ist ein bestimmtes, wohl unterscheidbares Objekt.
Ein Entity ist ein Informations-Objekt (Individuum, Real-Objekt (Ding), akbstraktes Konzept, Ereignis, …), welches in einer Datenbank verwaltet werden soll.

In der Entitäts-Analyse erkundet man die möglichen und bekannten Merkmale / Eigenschaften einer Entität. Der Anwender / Nutzer bzw. das Ziel der Datenbank bestimmt dann darüber, welche Attribute wirklich verarbeitet werden sollen. Ev. muss der Informatiker / Datenbank-Entwickler hier auch noch seine Interessen einbringen, um z.B. zusätzliche Attribute mit einbringen, die der eindeutigen Unterscheidung z.B. von Massenprodukten dienen.

Wir befinden uns in der logischen Ebene des ANSI-SPARC-Modell's. Die Modellierungs-Arbeiten begrenzen die Realität praktisch horizontal auf den Teil, den wir letztendlich informatisch umsetzen wollen / müssen.



Mit anderen Kunden bildet Herr Dr. Müller zusammen ein Entitäts-Set – also die Menge der Entitäten (Entitäts-Menge) zu einem bestimmten Zeitpunkt.



Definition(en): Attribute
Attribute sind Merkmale, Charakteristika und Eigenschaften von Entitäten.
Ein Attribut ist eine Eigenschaft eines / des Entitätstyps. Jedem Entitätstyp ***E*** wird eine nicht-leere endliche Attributs-Menge ***A*** zugeordnet.

Attribut-Werte oder auch Properties

Definition(en): Domäne / Attributs-Ausprägung
Eine Domäne ist der Werte-Bereich, die Menge möglicher Ausprägungen eines Attributs bzw. eines Attribut-Types.
Die Domäne (domain, Domain) ist der Wertebereich eines Attributs. Jedes Attribut $a \in A$ besitzt einen zulässigen Wertebereich dom(***a***).

Definition(en): Entitäts-Typ
Ein Entitäts-Typ ist eine Gruppe von Entitäten mit gemeinsamen, charakterisierenden Merkmalen / Strukturen.
Ein Entitytyp (Entitätstyp) ist eine Zusammenfassung von Entitäten mit gleichen charakteristischen Merkmalen.
Ein Entitäts-Typ ist die Gesamtheit aller Informations-Objekte (Entitäten), die eine gleiche Datenstruktur besitzen.
Jeder Entitätstyp ***E*** enthält Entitäten ***e***.

Nach der Entitäts-Analyse folgt die Beziehungs-Analyse (Relationship-Analyse). Hier untersuchen wir die vorhandenen und nutzbringenden Verknüpfungen zwischen Entitäten bzw. ihren Entitäts-Typen.

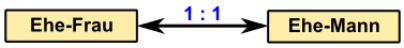
Definition(en): Beziehungen
Beziehungen sind Zusammenhänge zwischen Entitäten.
Eine Beziehung ist die Verknüpfung von mindestens zwei Entitäten.
Beziehungen sind sachliche Verbindungen der Datensätze der einer Tabelle mit denen einer anderen.

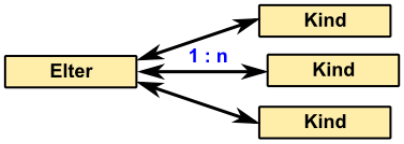
Die Anzahl der beteiligten Entitätstypen an einer Beziehung bestimmt ihren **Grad**.

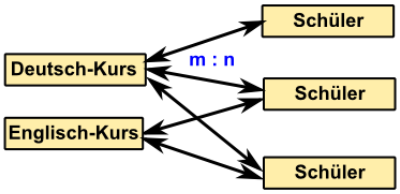
Definition(en): Kardinalität / Beziehungs-Typ
Die Kardinalität beschreibt das Verhältnis zwischen zwei Entitäten.
Die Kardinalität ist ein Maß zur Beschreibung der Verhätnisse zwischen Entitäten.

Definition(en): Relation
Eine Relation ist eine Menge von Tupeln / Datensätzen, deren Elemente / Attributwerte aus einem bestimmten Grundbereich / Domäne stammen.
.

Statt von Kardinalität spricht man auch von der Komplexität einer Beziehung.







Definition(en): Entity-Relationship-Modell (ERM)
Ein Entity-Relationship-Modell ist eine Repräsentation einer Datenstruktur einer Datenbank. Es stellt die Entitäts-Typen und deren Beziehungen untereinander dar.
Das Entity-Relationship-Modell ist ein System-relevanter Ausschnitt der Realität oder einer Abstraktion.

(informatische) Begriffs-Welten

allgemein	**Datenbanken (allgemein)**	**relationale Datenbank**	**Relationen-Modell**	**Entity-Relationship-Modell**	**UML**	**(Objekt-orientierte) Programmierung**	
	Datenbank						
		Tabelle	Relation	Entitäts-Menge Entitäts-Set		Klasse Objekt-Typ	
Merkmals-Bezeichnungen / Eigenschaft(en-Name)		Kopfzeile	Relationen-Typ Relationen-Format	Entitäts-Typ		Klasse Objekt-Typ	
		Spalten-Überschriften	Fremdschlüssel	funktionelle Beziehung (Relationship)	Assoziation		
		Spalte	Attribut				
Gegenstand, Person, Begriff, Ereignis	Entität			Entität		Instanz, Objekt	
	Datensatz	Zeile	Tupel	Entität	Instanz, Objekt		
Eigenschaft / Merkmal	Attribut			Attribut	Attribut	Attribut	
		Zelle	Attributwert				
Zusammenfassung von Gegenständen, Personen, Begriffen. Ereignissen	Entitäts-Typ					Klasse	
Verbindungen zwischen den gegenständen, Personen, Begriffen, Ereignissen	Relationship					Beziehung	
Werte-Bereich / Ausprägung einer Eigenschaft / eines merkmal's		Domäne Werte-Bereich Attribut-Wert					
typische Modellierung	Entity-Relationship-Modell					UML-Modell	

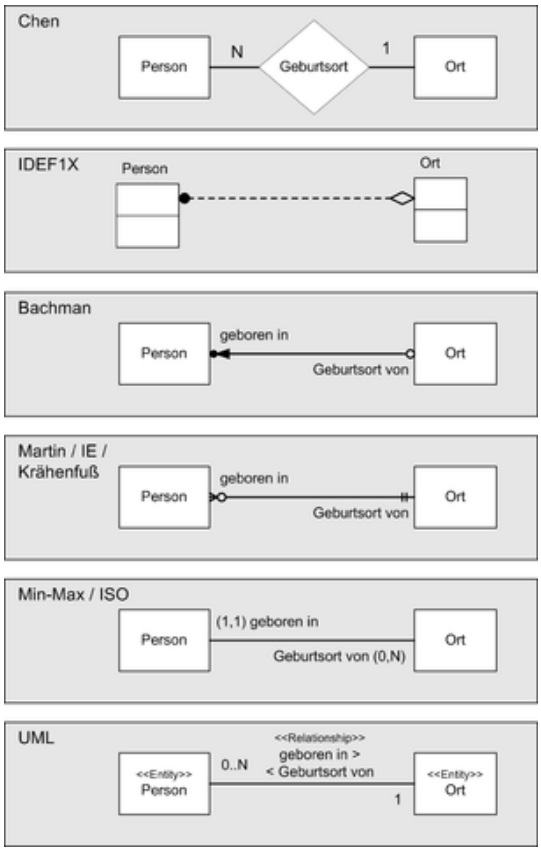
Aufgaben:

1. *Übernehmen Sie die folgende Tabelle und ergänzen Sie die fehlenden Inhalte!*
2.
3.

Entitäts-Typ 1	Entitäts-Typ 2	Beziehungs-Typ	Beziehungs-Richtung	Kardinalität
Hand	Finger	gehört_zu	←	1 : 4 (ev. auch 1 : 5)
Tutor	Schüler	betreut		
Tutor	Schüler	hat		
Frau	Mann	liebt		
Bruder	Schwester	hat		
Vater	Sohn	hat		
Rad	Auto	hat		
Tochter	Vater	hat		
Zertifikat	Zertifizierungsstelle	vergibt		
Ort	Ort	kürzeste_Entfernung		
Programm	Funktion	verfügt_über		
Personalausweis	Person	besitzt		

2.1.4. Entity-Relationship-Diagramme (ERD)

Entity-Relationship-Diagramme – kurz ERD – sind eine Form der Darstellung vom Daten-Modellen. Sie sind besonders für relationale Modelle geeignet.
Die Symboliken sind in den letzten Jahrzehnten mehrfach standardisiert und verbessert worden.
Es sind verschiedene Notationen üblich, die im Wesentlichen die gleichen Aussagen machen, aber die Daten und ihre Beziehungen anders darstellen.
In moderneren Notationen sind auch umgangssprachliche Umschreibungen enthalten, um vor allem Beziehungen sachgerechter lesen zu können. Üblicherweise wird dabei Englisch als Arbeitssprache genutzt. Für lokale und schulische Projekten ist Deutsch die bevorzugte Sprache.
In der Schule ist es üblich, die Notierungs-Form nach CHEN zu benutzen. Deren Umsetzung ist sowohl auf dem Papier als auch mit Grafik-Programmen gut möglich.
Will man Datenbanken Objekt-orientiert programmieren, dann wird neben dem ERD auch UML (Unified Modeling Language (vereinheitlichte Modellierungs-Sprache)) benutzt.



Varianten zur Notation in ER-Modellen
Q: de.wikipedia.org (Frank Roeing)

Objekte und Symbole der Entity-Relationship-Diagramme

klassische Variante nach Peter CHEN (CHEN Pin-Shan, 1976)

Entitätstypen

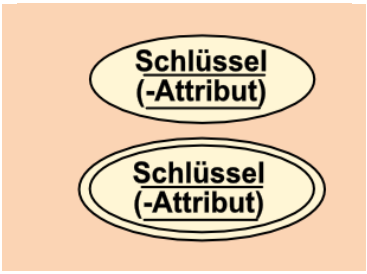
Symbol ist ein Rechteck mit innen liegendem Namen des Entitätstyps
Im Skript-eigenem Farbschema wird ein helloranger Hintergrund verwendet.

Attribute

Attribute werden in Ovale notiert. Der Name des Attributs steht im Oval. In unserem Farbschema wird ein helles gelb als Hintergrund genutzt.

Primärschlüssel

Bei Attributen mit Primäschlüssel-Charakter werden wird der Attributsname unterstrichen.
In unübersichtlichen / komplexen Entity-Relationship-Diagrammen werden die Primärschlüssel zur besseren Erkennung fett geschrieben oder die Ovale doppelt gezeichnet. Diese Varianten entsprechen aber nicht dem Standard. Sie sind hier aufgezeigt, weil viele externe ERD eben solche Symbole (eingemischt) verwenden.



Beziehungen

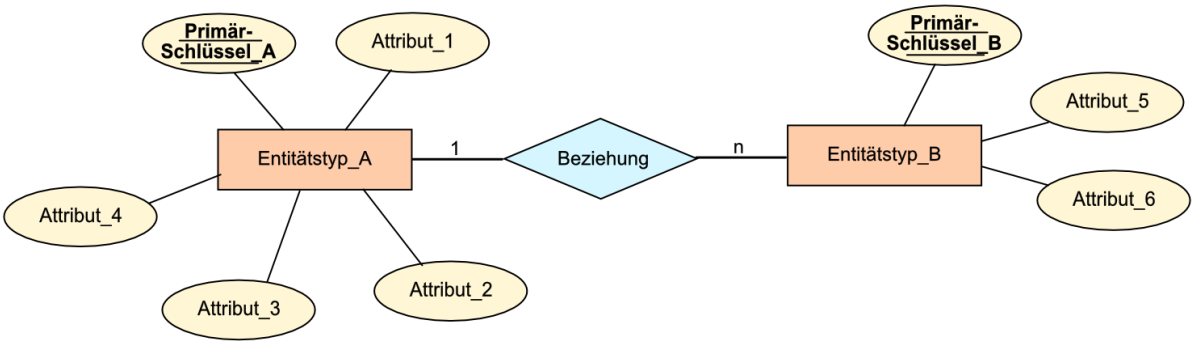
Beziehungen zwischen zwei Entitätstypen werden als Rauten (Rhomben) dargestellt. Der Name liegt ebenfalls innen. Als Hintergrund wählen wir – hier im Skript – eine hellblaue Farbe.

Beziehung, Assoziationen, Kardinalität

Beziehung besteht aus zwei Assoziationen in beide Richtungen mit zugehörenden Kardinalitäten. Die Assoziationen enden an Entitätstypen. An die Assoziationen werden die Kardinalitäten rangeschrieben.

Eine farbliche Gestaltung der Diagramm-Elemente ist eigentlich nicht üblich. Die Strukturen werden aus meiner Sicht aber gerade durch eine farbliche Unterscheidung noch besser sichtbar, so dass wir hier vorrangig auf farbige ERD's setzen.

Beispiel-Diagramm (allgemein):



Betrachten wir ein ERD – quasi in seinen Entstehungs-Phasen – für ein einfaches Beispiel.

Wir wollen eine kleine Datenbank zu Fahrrädern und ihren Besitzern anlegen.

Als ersten Entitäts-Typ haben wir das Fahrrad. Als Symbol ist das Rechteck vereinbart.

Die Bezeichnung für den Entitäts-Typ – also "Fahrrad" kommt in die Mitte.

Der zweite Entitäts-Rtyp ist der Besitzer. Auch für ihn wählen wir ein Rechteck und beschriften dieses entsprechend. Zwischen den beiden Entitäts-Typen muss nun eine Beziehung bestehen. Diese haben wir hier unverfänglich mit "hat" bezeichnet. Das kann man hier dann von beiden Seiten lesen:

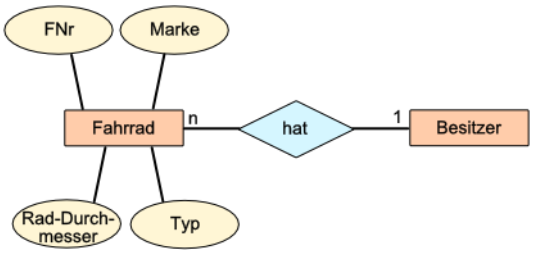
Fahrrad hat Besitzer oder Besitzer hat Fahrrad.

Das Symbol für eine Beziehung ist der Rhombus.

Im nächsten Schritt ermitteln wir die Kardinatät zwischen den beteiligten Entitäts-Typen. Üblicherweise hat ein Fahrrad nur einen Besitzer. Ein Besitzer kann aber mehrere Fahrräder haben. Somit haben wir eine klassische N:1-Beziehung vorliegen.

Zu den Entitäten legen wir nun die notwendigen oder zur Verfügung gestellten Daten fest. Dabei handelt es sich um Attribute. Sie werden im Diagramm als Ovale eingezeichnet.

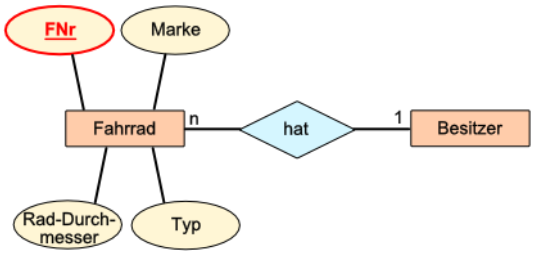
Für unsere einfache Datenbank sollen das nur die Fahrrad-Nummer ("FNr"), die "Marke", der Fahrrad-"Typ" und der "Rad-Durchmesser" (in Zoll sein).



Die Fahrrad-Nummer bietet sich auch als Schlüssel für die Entitäten an.

Das Schlüssel-Attribut kennzeichnen wir durch eine Unterstreichung.

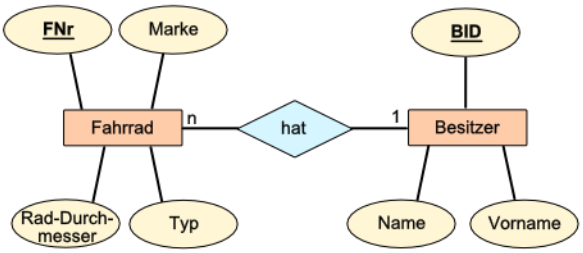
Die rote Kennzeichnung soll nur die Auswahl gegenüber der einfachen Attribute hervorheben.



Genau, wie beim Fahrrad – verfahren wir jetzt auch beim zweiten Entitäts-Typ.

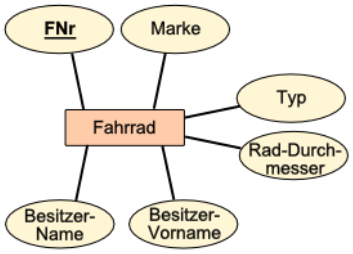
Er bekommt seine Atrribute und ein geeignetes Schlüssel-Attribut.

Das fertige ERD würde dann so aussehen:



Aufgaben:

1. *Erstellen Sie ein ERD für Lern-Patenschaften in einer Schule! (Es werden nur die notwendigsten Daten erfasst!)*
2. *Für das obige Fahrrad-Besitzer-Beispiel wurde auch das nebenstehende ERD angeboten. Diskutieren Sie das zugrundeliegende Daten-Modell!*



Beziehungs-Typen zwischen zwei Entitäten (→ Kardinalität)

- **1 : 1** eine Entität vom Typ1 steht in direkter Beziehung zu genau einer Entität vom Typ2
  z.B. eine konkrete Frau heiratet einen konkreten Mann
  ev. können die Geschlechter hat wegelassen werden, dann können Typ1 und Typ2 gleich sein, z.B. vom Typ Person, trotzdem sind nur einfache direkte Beziehungen zulässig

- **1 : N** eine Entität vom Typ1 ist mit mehreren Entitäten vom Typ2 verbunden
  1 : n z.B. kann eine (konkrete) Person mehrere (konkrete) Auto's besitzen
  auch z.B.: eine Klasse besteht aus mehreren (- ev. auch wechselnden -) Schülern oder eine Abteilung hat mehrere Angestellte

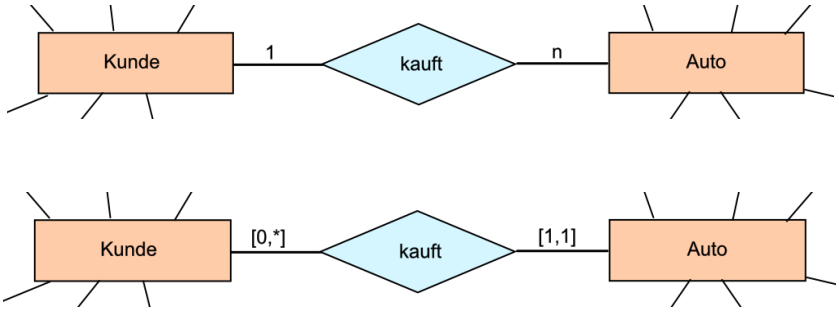
- **N : 1** praktisch auch 1 : N, nur im Modell und von der Bezeichnung umgekehrt
  n : 1 viele Entitäten (z.B. besondere Bauteile) sind einer anderen Entität (z.B. einem Produkt) zugeordnet

- **N : M** zwischen den Entitäten der beiden Typen bestehen beliebige – meist mehrfache / vielfache – Beziehungs-Verhältnisse
  n : m z.B. kommt die Schraube M4x20 in vielen Produkten einer Firma vor

Beziehungen können zwingend (eng.: mandatory) sein, d.h. es müssen immer Paare existieren und kein Element des einen (niedrigzahligeren) Entitäts-Types darf ohne ein zugehöriges Objekt des anderen Entitäts-Types sein.
Nicht zwingende Beziehungen sind freigestellt (optional). In beiden Entitäts-Typen können unverbundene Entitäten vorkommen.
Man bezeichnet diese Unterscheidung als Mitgliedsklassen.

Restriktionen / Beschränkungen / Bedingungen

Im Bereich der Kardinalitäten kennen wir auch noch die sogenannten Restriktionen. Dabei werden minimale und maximale Werte spezifiziert.
Die Restriktionen werden in eckigen Klammern anstatt der reinen Kardinalitäten an die Assoziationen notiert.



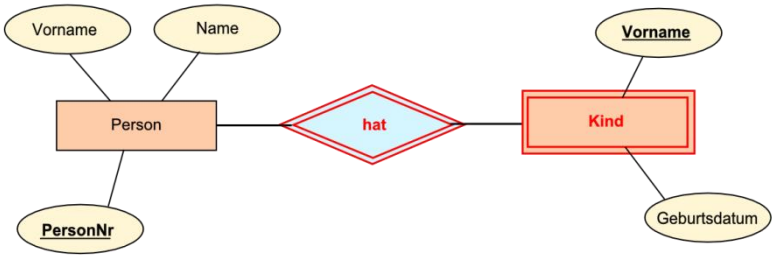
Dabei können die folgenden Einträge auftauchen:

- **[0,*]** keine Einschränkung vorgesehen; ein Kunde kauft (z.B. in diesem Jahr) kein Auto oder beliebig viele
- **[1,*]** der Kunde muss mindestens 1 Auto kaufen / gekauft haben
- **[1,1]** hier ist die Einschränkung so, dass 1 Kunde auch genau nur 1 Auto kauft / gekauft hat

zu den Integritäts-Bedingung zählt, dass von jeder Beziehung mindestens eine Kante (Assoziation) aus- / ab-geht
Entitäten müssen nicht zwingend eine Kante (Assoziation) zu einer Beziehung besitzen

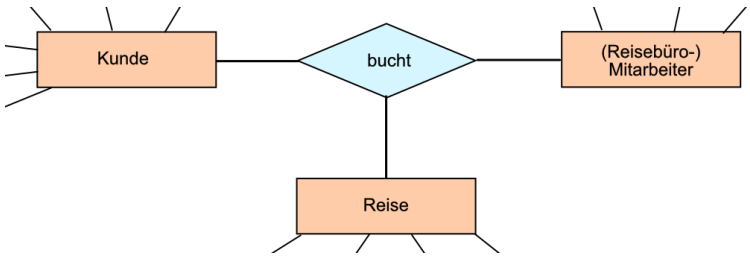
Im ERD werden sowohl die Beziehung und der "schwache" Entitätstyp doppelt umrandet gezeichnet.

schwache Entitäten benötigen immer eine eindeutige Identifikation über eine Beziehung



schwache Entität und schwache Beziehung

mehrdeutige Beziehungen



Schlüssel dienen der eindeutigen Identifizierung von Entitäten

Definition(en): Entitätschlüssel / Schlüsselfeld / Schlüsselattribut
Ein Entitätschlüssel ist ein Attribut eines Entitätstypes, bei dem alle Entitäten einen einmaligen – von anderen Faktoren unabhängigen – Wert besitzen.
Ein Entitäts-Schlüssel (engl.: candidate key) ist ein möglicher Kandidat für einen Primärschlüssel.
Ein Entitäts-Schlüssel ist ein identifizierendes Attribut einer Entität.

Alle Entitäts-Schlüssel sind Schlüssel-Kandidaten für die Auswahl eines Primär-Schlüssel's. Unter den vorhandenen Schlüssel-Kandidaten wählt man solche aus, die möglichst kurz (klein) und wenig strukturiert sind. Reine Zahlen (auch als solche gespeichert) sind Zeichenketten immer vorzuziehen.

Aufgaben:

1. *Von Auto's sollen alle verfügbaren Daten gespeichert werden. Schlagen Sie mindestens 10 Attribute vor! Erläutern Sie, welche Attribute sich als Entitätsschlüssel eignen!*
2. *Überlegen Sie sich, welche Daten von Ihren Kursmitgliedern gespeichert werden könnten! Bestimmen Sie die Schlüsselattribute! Wovon ist die Wahl der Schlüsselattribute abhängigg? Erläutern Sie!*
3.

Definition(en): Primärschlüssel
Ein Primärschlüssel (engl.: primary key) ist ein eindeutiges Identifikationsmerkmal (oder eine eindeutige Merkmalskombination) der Entitäten.
Der Primärschlüssel ist ein Attribut des Entitätstyp, dessen Wertebereich (Domäne) eindeutige Werte für die Entitäten enthält.
Ein Primärschlüssel ist ein ausgewählter / festgelegter Entitätsschlüssel. Dabei sind natürlichen Zahlen (Nummern) vor Texten und vor zusammengesetzten Attributs-Kombinationen der Vorrang zu geben.
Ein Primärschlüssel ist das eindeutige / einmalige, primär-genutzte Zugriffselement auf einen Datensatz (einer Tabelle).

Häufig werden Primär-Schlüssel künstlich gewählt, um die Domäne der natürlichen Zahlen zu nutzen, was wiederum vor allem eine schnelle Bearbeitung und Prüfung zulässt; auch der Forderung nach möglichst kleinen Werten kann so entsprochen werden
In sortierten Tabellen lassen sich bestimmte Datensätze mit natürlichen Nummern schneller finden, ohne dass man alle Schlüssel (von oben nach unten) durchprobieren muss.
durch die zusätzliche / Attribut-unabhängige / automatische Festlegung von künstlichen Primärschlüsseln wird auch die Unabhängigkeit von den Basisdaten erreicht und das Datenbank-System besitzt eine schnelle unabhängige Prüfmöglichkeit für die Daten-Konsistenz
auch wenn sich Schlüssel zukünftig ändern könnten, dann wird zu künstlichen Schlüsseln gewechselt
Ein künstlicher Schlüssel ist eine Zahlenfolge, bei der das DBMS dafür sorgt, dass keine Werte doppelt vorkommen. Bei einer manuellen Eingabe prüft das System immer sofort, ob der Schlüssel schon vergeben ist. Der Computer kann aber auch die Vergabe von Schlüssel selbstständig übernehmen. In großen Daten-Beständen ist das meist sinnvoll.

Aus der Besprechung der Normalisierung (→ 2.2.1. Normalisierung) wissen wir schon, dass in normalisierten Tabellen Verweise auf die Datensätze in anderen Tabellen enthalten sind. Diese Schlüsselnummern sind sogenannte Fremdschlüssel. Man spricht auch von Sekundärschlüsseln (secondary keys).
In der Objekt-Tabelle (Master-Tabelle) können sie auch mehrfach vorkommen, weil ja bestimmte Daten auch mehrfach vorkamen. (Deshalb hatten wir ja überhaupt eine Normaliserung vorgenommen.)
In der Detail-Tabelle (Referenz-Tabelle) sind die Fremdschlüssel (der Objekt-Tabelle) immer eindeutige Primärschlüssel (z.B.: ID's). Hier dürfen sie nur einmalig vorkommen.

Definition(en): Fremdschlüssel
Ein Fremdschlüssel ist ein Attribut einer Tabelle, der in einer anderen Tabelle ein Primärschlüssel ist.
Ein Fremdschlüssel ist der Primärschlüssel einer anderen Tabelle, der in dieser Tabelle als Attribut eingesetzt ist.
Fremdschlüssel sind Primärschlüssel anderer Tabellen, die als Verweise (Zeiger, Pointer) auf eben diesen Datensatz in der fremden Tabelle dienen.

andere Attribute, die nicht identifizierend eingesetzt werden können, sind beschreibende Attribute

Aufgaben:

1. *Definieren Sie den Begriff beschreibende Attribute!*
2. *Prüfen Sie die nachfolgende Tabelle auf Schlüssel-Kandidaten und legen Sie einen geeigneten Primär-Schlüssel fest!*

Service-Nummer	Kennzeichen	Hersteller	Typ	Baujahr	Besitzer	Kaufpreis
2017-0395-AT	S-FT 295	Skoda	Rapid	12/2016	L. Meier	15000
2018-1094-BZ	HRO-GH 73	Opel	Corsa	05/2007	C. Zander	2000
2017-0502-BZ	B-ZZ 2075	Mercedes	C-Klasse	10/2012	G. Muster	43000
2017-1196-BZ	M-SX 5867	Fiat	Panda	11/2006	P. Panzer	1200
2018-0609-AT	HH-HH 666	Audi	A6	11/2007	K. Meier	27000
2018-0306-AT	DBR-BM 2	Renault	Clio	03/2004	H. Otto	4050
2018-0606-CK	M-SX 4976	Rover	Mini	03/2014	O. Fredov	17000
2018-0606-AT	BBG-U 959	Seat	Alhambra	03/2010	I. Meier-Bauer	24500
2018-0113-AT	ROS-T 888	Porsche	Cayenne	07/2017	Tang L.	28000
2018-1227-CK	DD-OO-7	Audi	A8	05/2015	D. Zander	41000

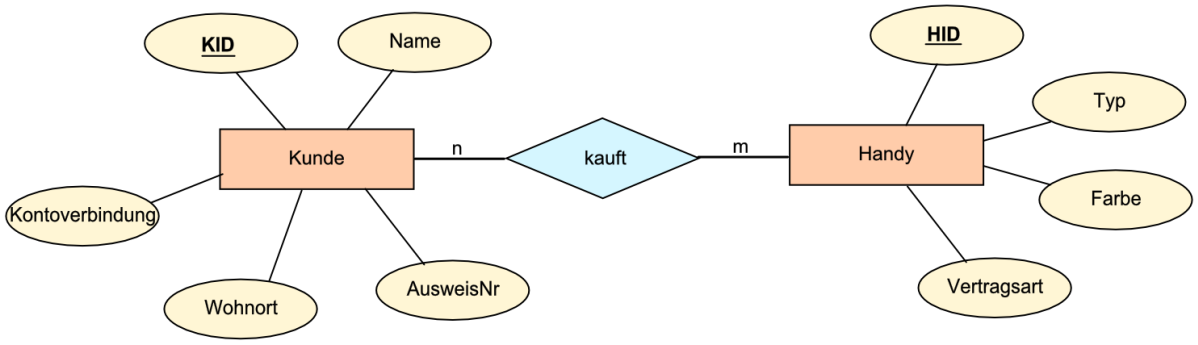
3. *Wenn Sie bei der Primärschlüssel-Auswahl nur die vorhandene Tabelle bedacht haben, dann prüfen Sie Ihre Wahl noch einmal unter dem Gesichtspunkt, dass die Datenbank für Gesamt-Deutschland genutzt werden soll!*

für die gehobene Anspruchsebene:

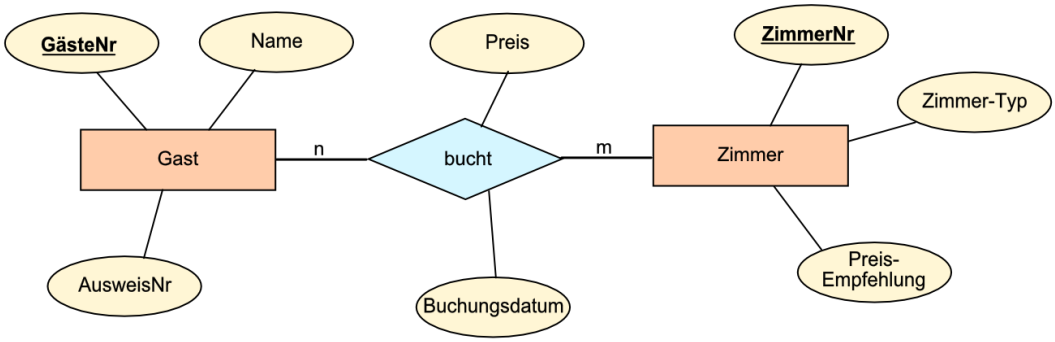
4. *Normalisieren Sie die Tabelle (Verwendung als Gesamt-Deutschland-Datenbank)!*

Übungs-Aufgaben:

1. Gegeben ist das nachfolgende Entity-Relationship-Diagramm (ERD). Interpretieren Sie das ERD! Gehen Sie auf alle Elemente ein und erläutern Sie auch kurz die Arten von Grundelementen eines ERD!



2. Skizzieren Sie ein ERD (nur) für die Mitschüler Ihres Kurses mit den Schul-wichtigen Daten!
3. Diskutieren Sie das abgebildete Entity-Relationship-Diagramm!



4. Erfassen Sie folgende Gegebenheit!

   Bei einem Computerhändler ("Computer NullEins") sollen die Computerverkäufe fertig konfektionierter Computer in einer kleinen Datenbank verwaltet werden. Die Computer unterscheiden sich im Prozessor (i3, i5 oder i7), im Speicher-Umfang (4, 8 od. 16 GB), in der Festplatten-Größe (1, 2, 3 od. 4 TB), in der Farbe (grau od. schwarz) und der eingebauten Graphikkarte (G500, G1000, G2000 od. G4000). Alle Kombinationen sind möglich. Die Kunden stammen alle aus dem gleichen Gewerbegebiet "Mitte". Das Gewerbegebiet hat vier Rauten-förmig angeordnete Straßen (Nord-, Ost-, Süd- und West-Str.). Dort sind die folgenden Firmen ansässig: Büro-Meier GmbH, Schulz & Söhne OHG, Auto-24/7 GmbH und eine Niederlassung von Handy-Repair-ABC. Alle Firmen erhalten fortlaufend (wie aufgezählt) eine Kundennummer. Je nach Gewerbe kaufen sie fortlaufend unterschiedliche Computer in unterschiedlicher Stückzahl. Die Garantie eines gekauften Rechners beträgt 2 Jahre und beginnt am Kauftag.

   a) Skizzieren Sie ein ERD nur für die Computer, die im Angebot sind! Legen Sie ev. weitere notwendige Merkmale fest!
   b) Diskutieren Sie Ihr ERD mit anderen Kursteilnehmern! Optimieren Sie ERD zu einem gemeinsam akzeptieren Modell!
   c) Zeichnen Sie nun das ERD für die gesamte Gegebenheit! Wenn es notwendig ist, ergänzen Sie weitere Merkmale!

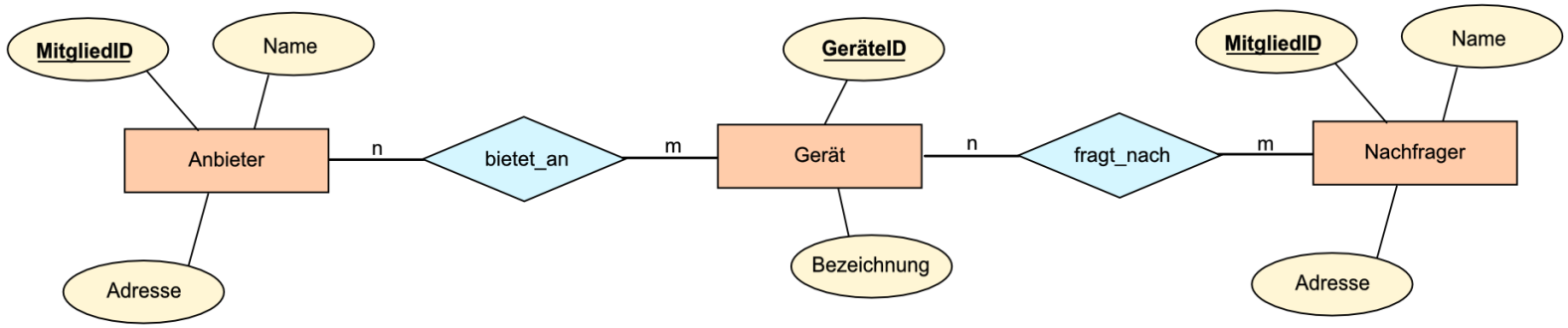
für die gehobene Anspruchsebene:

5. Wählen Sie eine andere Gegebenheit (mit mindestens zwei Entitäts-Typen) und beschreiben Sie diese in einem kleinen Text! Erstellen Sie dann ein ERD!

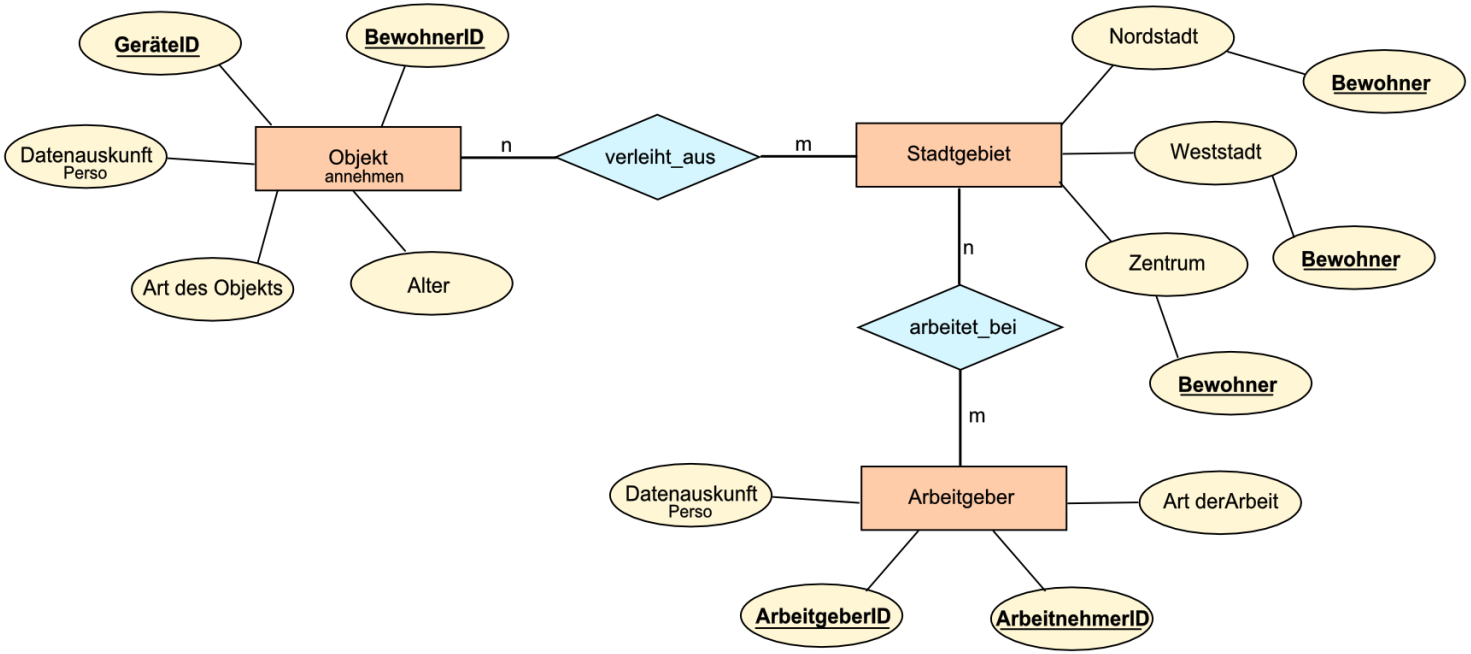
Komplexe Aufgabe zu den Entity-Relationsship-Modellen

1. Stellen Sie das Entity-Relationship-Modell (ERM) mit seinen Elementen vor!
2. Geben Sie für die Grundelemente des ERM die Symbolik innerhalb von Entity-Relationsship-Diagrammen (nach CHEN) an!
3. In den Stadtgebieten "Weststadt", "Nordstadt" und "Zentrum" haben sich unabhängige Gruppen von Bewohnern zusammengefunden, die nach dem Prinzip "Let's share / Sharing Economy" Arbeits-Leistungen (z.B.: Bügeln, PC reparieren, …) und Geräte (z.B.: Bohrmaschine, Schnellkochtopf, …) in gegenseitiger Ausleihe nutzen wollen. Die Gruppe aus dem Gebiet "Weststadt", sowie deren Geräte und Leistungen sind schon so umfangreich, dass man eine im Internet verfügbare Datenbank aufbauen möchte. Zuerst einmal will man sich nur auf den Geräte-Verleih konzentrieren.
   (Die Datenbank ist zuerst auch nur zum Testen gedacht! Es müssen nur die derzeit notwendigen Elemente bedacht werden!)
   Erstellen Sie ein Entity-Relationship-Diagramm (ERD) für die Problem-Situation!
   Erläutern Sie kurz die Auswahl der Elemente Ihrer Miniwelt und des ERD!
4. Auf den nächsten Seiten sind verschiedene Vorschläge für ERD's zu Aufgabe 3 abgedruckt. Setzen Sie sich mit diesen auseinander!

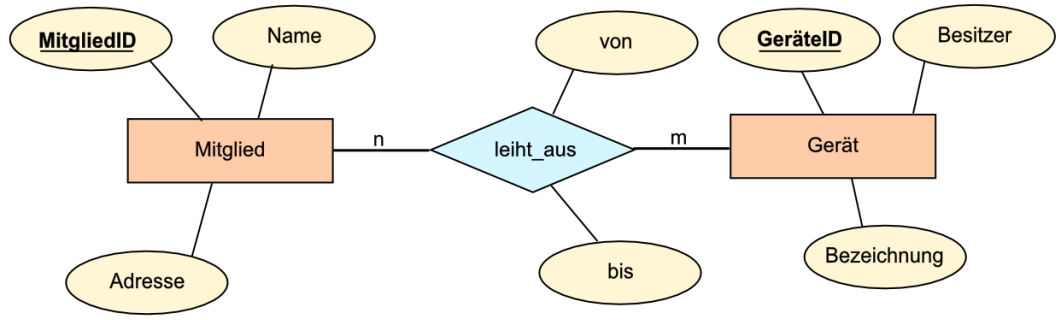
Vorschlag1:



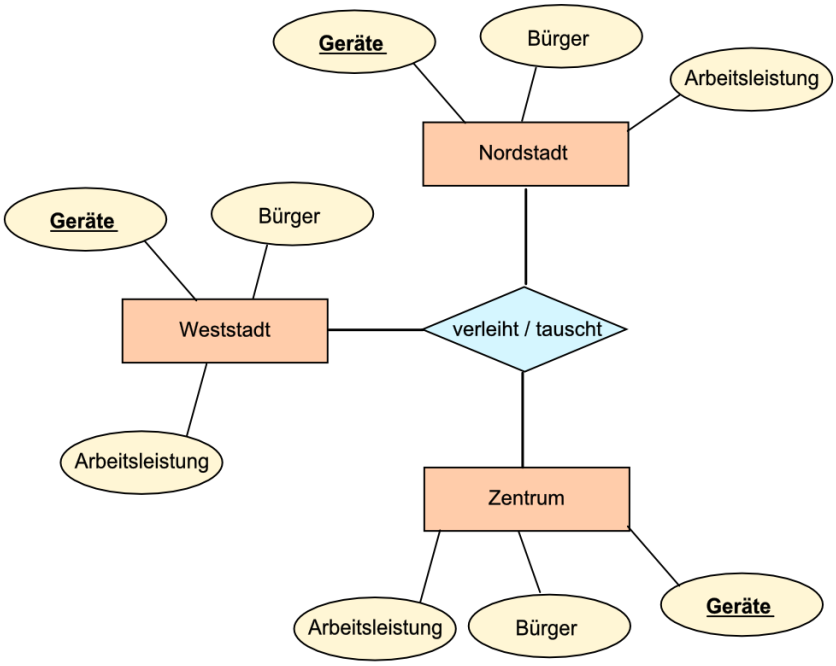
Vorschlag2:



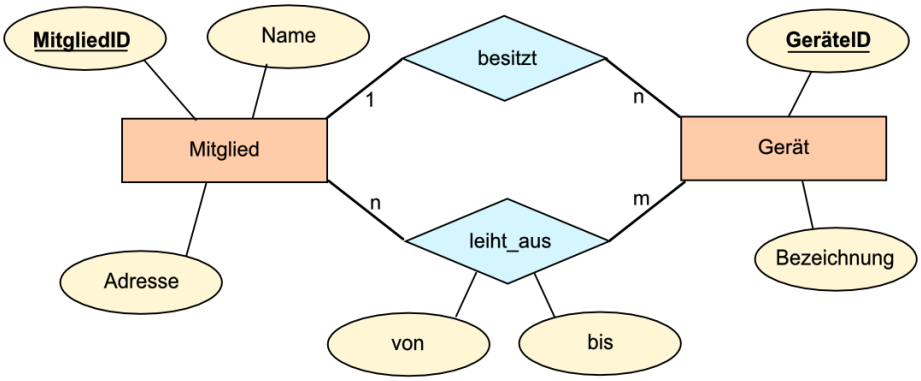
Vorschlag3:



Vorschlag4:



Vorschlag5:



2.1.5. Erweiterungen des Entity-Relationship-Modells

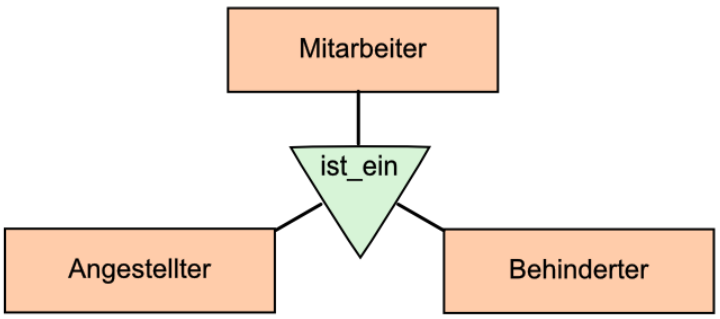
Erweiterungen bei Attribut

- **optionale Attribute** — sind Attribute, die nicht bei allen Entitäten eines Types einen Wert annehmen müssen
  (als graphisches Zeichen in ERD wird ein Kringel in der Verbindungslinie verwendet)
- **strukturierte Attribute** — sind Attribute, die sich aus mehreren (Einzel- / Unter-) Attributen zusammensetzen
- **mengenwertige Attribute** — sind Attribute, die mehrere Werte – in Form einer Menge – beinhalten
- **virtuelle Attribute** — sind Attribute, die sich aus (aktuellen) Berechnungen ergeben und nicht (dauerhaft) gespeichert werden

Erweiterungen zu Spezialisierung und Generalisierung

Generalisierung meint die Ermittlung ähnlicher Entitätstypen und der Zuordnung zu einem Obertyp. Die ähnlichen Entitätstypen heißen dann Untertypen.

Spezialisierung ist die die Umkehrung der Generalsierung, d.h. einem (allgemeinen) Entitätstyp werden (spezialisierte) Entitätstypen (Untertypen) zugeordnet. Der allgemeine Entitätstyp wird Obertyp genannt.
In ERD wird eine Beziehung zwischen den Typen dann als (gleichseitiges) Dreieck gekennzeichnet.



2.1.6. Transformation eines ERM (ERD) in eine relationale Datenbank

Überführung in ein relationales Modell (Relationen-Modell) in 7 Schritten

1. starke Entitäts-Typen	erstellen einer Relation R mit den Attributen a_x und einem Primärschlüssel ***k*** $R = \{a_1, a_2, a_3, \ldots, a_n\} \cup \{\boldsymbol{k}\}$
2. schwache Entitäts-Typen	erstellen einer Relation R mit den Attributen a_x und einem Fremdschlüssel ***k'*** sowie einem Primärschlüssel {***k***} ({***k***} repräsentiert den starken Entitäts-Typ u. **{*k'*}** den schwachen) $R = \{a_1, a_2, a_3, \ldots, a_n\} \cup \{\boldsymbol{k}\} \cup \{\boldsymbol{k'}\}$
3. 1:1-Beziehungen	eine der beiden Relationen (T od. S) wird um den Fremdschlüssel der anderen ergänzt $S = \{a_1, a_2, a_3, \ldots, a_n\} \cup \{\boldsymbol{k}\} \cup \{\boldsymbol{k_T}\}$ oder $T = \{a_1, a_2, a_3, \ldots, a_n\} \cup \{\boldsymbol{k}\} \cup \{\boldsymbol{k_S}\}$
4. 1:N-Beziehungen	N-kardinale Relation T wird um den Fremdschlüssel der 1-kardinalen Relation S ergänzt $T = \{a_1, a_2, a_3, \ldots, a_n\} \cup \{\boldsymbol{k}\} \cup \{\boldsymbol{k_S}\}$
5. N:M-Beziehungen	erstellen einer (neuen) Relation R mit den Attributen a_x und einem Primärschlüssel ***k*** und den Fremdschlüsseln der beteiligten Relationen T und S $R = \{a_1, a_2, a_3, \ldots, a_n\} \cup \{\boldsymbol{k}\} \cup \{\boldsymbol{k_T}\} \cup \{\boldsymbol{k_S}\}$
6. mehrwertige Attribute	erstellen einer Relation R mit dem mehrwertigen Attribut a_x aus (der Relation) S und dem Fremdschlüssel von S $R = \{a_x\} \cup \{\boldsymbol{k_S}\}$
7. n-äre Beziehungs-Typen	erstellen einer Relation R, wenn n größer als 2 ist, die alle Fremdschlüssel und die Attribute beinhaltet für alle Kardinalitäten >1 ist der Primärschlüssel die menge aller Fremdschlüssel $R = \{a_1, a_2, a_3, \ldots, a_n) \cup \{\boldsymbol{k}_1, \boldsymbol{k}_2, \boldsymbol{k}_3, \ldots, \boldsymbol{k}_m)$ für alle anderen Fälle ist der Primärschlüssel die Menge von n-1 Fremdschlüssel (, wobei die mit der Kardinaltät >1 immer im Primärschlüssel enthalten sein müssen

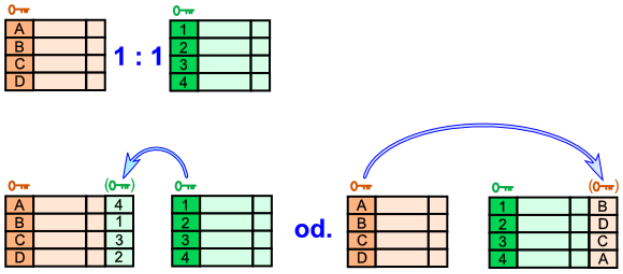
Legende:

R, S, T	…	Relationen
k	…	Schlüssel (Index R bei Fremdschlüssel zu Relation R; Zahlen-Index für gezählte Fremdschlüssel)
$a_1, a_2, \ldots, a_x$	…	Attribute

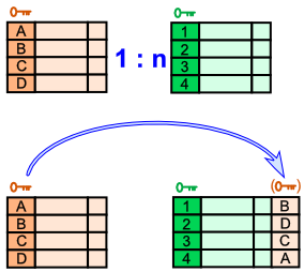
abgeleitet: vereinfachte Transformations-Regeln zur Überführung eines ERM in eine relationale Datenbank

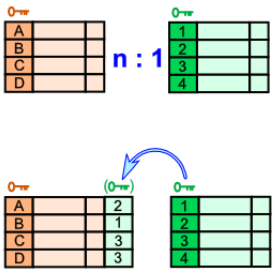
1. Regel Jede Entität wird als Tabellen-Name und die zugehörigen Attribute als Tabellen-Schema (Relationsschema, Spalten-Überschriften) übernommen!
Prüfen auf Schlüssel-Kandidaten und festlegen eines Primärschlüssels (ev. ein künstlichen).

2. Regel Gilt zwischen zwei Entitäten der Beziehungs-Typ / die Kardinalität 1 : 1, dann wird an einer der Tabellen eine Spalte / ein Attribut (mit dem Namen der Beziehung) ergänzt und mit den zugehörigen Primär-Schlüsseln der Datensätze der anderen Entität / Tabelle (vollständig) ausgefüllt.

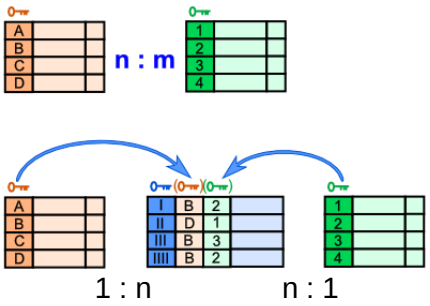


3. Regel Gilt zwischen zwei Entitäten der Beziehungs-Typ / die Kardinalität 1 : n, dann wird an der Tabellen mit der Assoziation n eine Spalte / ein Attribut (mit dem Namen der Beziehung) ergänzt und mit den zugehörigen Primär-Schlüsseln der Datensätze der Entität / Tabelle mit der Assoziation 1 (vollständig) ausgefüllt (heißen dann Fremd-Schlüssel).





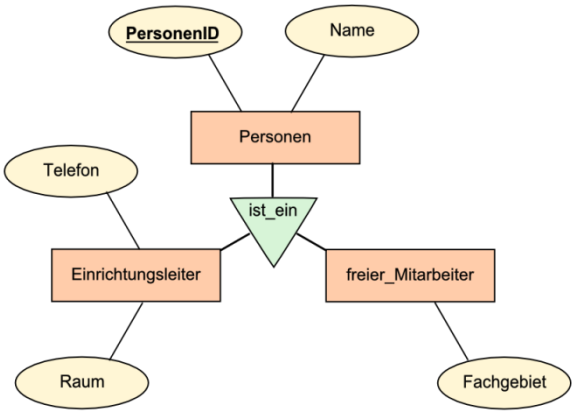
4. Regel Gilt zwischen zwei Entitäten der Beziehungs-Typ / die Kardinalität n : m, dann wird eine neue Tabelle erstellt, die neben einem eigenen Primär-Schlüssel und ev. zusätzlichen Attributen im Realtions-Schema auch zwei Spalten (Felder / Attribute) enthält, welche die Namen der Tabellen tragen und mit den Primärschlüssel-Werten diese Tabellen (vollständig) ausgefüllt werden (Die Schlüssel heißen hier dann Fremd-Schlüssel).



weitere (mögliche) Umsetzungen

IS-A-Beziehungen
(IST-EIN-Beziehung)
ist-ein-Beziehung

ERD (gekürzt):



Möglichkeit 1: (3 Tabellen mit 1 : n Beziehungen)
Personen*(PersonenID, Name)*
Einrichtsleiter*(PersonenID, Raum, Telefon)*
freier_Mitarbeiter*(PersonenID, Fachgebiet)*

Möglichkeit 2: (3 eigenständige Tabellen)
Einrichtsleiter*(PersonenID, Name, Raum, Telefon)*
freier_Mitarbeiter*(PersonenID, Name, Fachgebiet)*
(restl.) ***Personen****(PersonenID, Name)*

Möglichkeit 3: (1 Tabelle mit NULL-Werten)
Mitarbeiter*(PersonenID, Name, Raum, Telefon, Fachgebiet)*

2.1.6.1. Weiterentwicklungen des Entity-Relationship-Modells

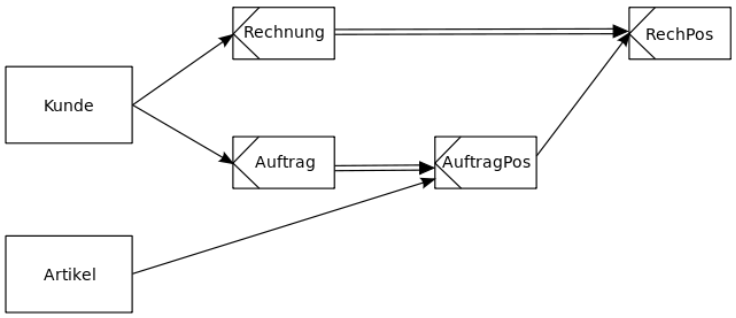
Structured ERM (SERM)

1988 von Elmar SINZ

Ziele

Strukturierung großen Daten-Schemata durch quasi-hierarchische Anordnung
Visualisierung von Existenz-Abhängigkeiten durch Beziehungs-Semantik(en)
Vermeidung von Inkosistenzen durch Nicht-Zulassung von Zirkel-Bezügen
Vermeidung unnötiger Relations-Typen durch Schlüssel-Vererbung

geänderte graphische Darstellung, angelehnt an Graphen-Theorie
aus dem ERD (→ 2.1.4. Entity-Relationship-Diagramme (ERD)) wird dann ein SERD



SER-Diagramm (Beispiel)
Q: de.wikipedia.org (unbekannt)

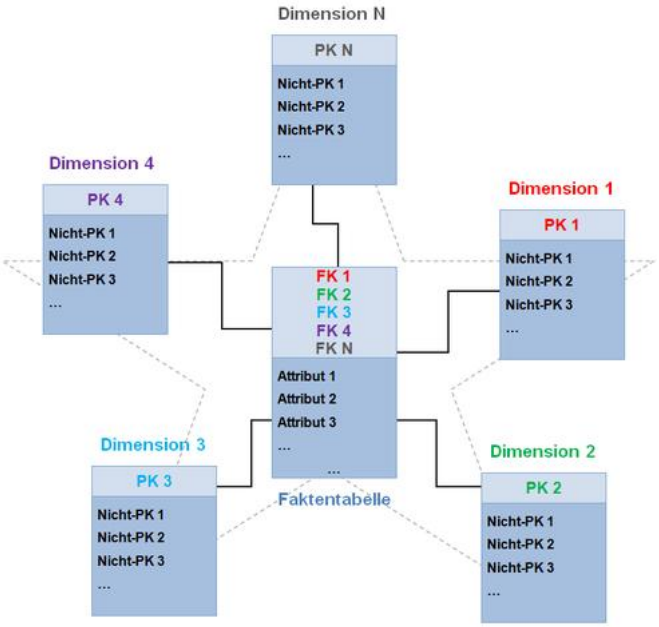
EER-Modell

E3R-Modell

SAP-SERM

Stern-Schema

Darstellung der Entitäten als Dimensions-Tabellen
die Verbindungen werden durch einfache Linien dargestellt. An den Start- und Ziel-Punkten sind besondere Symbole , die Aussagen zur Abhängigkeit verdeutlichen.



Stern-Schema (allg.)
(Die Fakten-Tabelle hat einen zusammengesetzten Primärschlüssel aus den Primärschlüsseln der Dimensions-Tabellen)
Q: de.wikipedia.org (Dhbw dbi); cc-by-sa 3.0

vor allem in den Bereichen "Big data", "online analytical pocessing" (OLAP) und Data-Warehouse genutzt
hier ist es das derzeit übliche logische Datenbank-Schema

Tabellen liegen hier i.A. denormalisert vor
Ziel ist eine hohe Performance
Nachteil ist ein erhöhter Speicherbedarf

Fakten-Tabelle besteht vorrangig aus Fremdschlüsseln zu Dimensions-Tabellen
eigene Attribute sind in der Fakten-Tabelle auch zugelassen
zu den Dimensions-Tabellen besteht eine 1 : n –Beziehung (Dim-Tab : Fak-Tab)
die Dimensions-Tabelle beinhalten beschreibende Daten und können somit auch eigene Attribute besitzen

nächste Erweiterung / Verbesserung ist das Schneeflocken-Schema

Vorteil der Trennung von Fakten (Geschäfts-Daten, Messwerte, Kennzahlen, …) und Dimensionen liegt in der Möglichkeit, die Daten-Bestände für jede Dimension einzeln, aber auch im Verbund mit anderen zu analysieren
gesucht werden z.B. bestimmte Zusammenhänge (Verkaufsangebote, Fehlermuster, Betrügerein, …)

in SQL sind die Abfragen charakteristisch aufgebaut und deshalb auch als **"Star-Join"** bezeichnet:

SELECT Fakt-Attribut | Dimensions-Attribut
FROM Fakten-Tabelle | Dimensions-Tabelle
WHERE Bedingung
GROUP BY Fakt-Attribut | Dimensions-Attribut
GROUP BY Fakt-Attribut | Dimensions-Attribut

2.1.7. Überführung von Tabellen in eine relationale Datenbank durch Normalisierung

Alternative Möglichkeit – neben der Transformation (→) – um zu einem logischen DB-Modell zu kommen
Bedingung Daten liegen vollständig in Tabellen vor

Optimierung der Tabellen-Struktur
- Vermeidung von Redundanzen
- Vermeidung von Inkonsistenzen
- Vermeidung von Integritäts-Verletzungen (z.B.: Lösch-Anomalien, Änderungs-Anomalien)

Problem ERM / ERD: die Attribute können komplexer Natur sein → also z.B. die Adresse könnte unstrukturiert noch so verstanden werden : *12345 Mustern; Musterstr. 23*
praktisch hätten wir dann ein Problem später z.B. nur nach Orten oder Straßen zu suchen
Normalsierungen vermeiden solche mehrwertigen Attribute
Auflösung mehrwertiger Attribute erster Schritt zur Optimierung von Tabellen → Normalisierung 1. Ordnung
führt zu Tabellen in der sogenannten 1. Normalform

Normalisierung aber auch Mittel zur Optimierung von logischen DB-Modellen, die aus der Transformation (→) stammen

Normalisierung orientiert sich an der funktionellen / logischen Abhängigkeit der Daten

2.2. relationales Daten(bank)-Modell

relatio (lt: zurückfragen); Beziehungen
Daten werden in Tabellen-Form gespeichert, verwaltet

basiert auf Arbeiten von Edgar Frank "Ted" CODD ()

Kartei		**Datenbank**	**allgemein**
Aktenschrank		**Datenbank**	Gesamt-Sammlung
Karteikasten		**Tabelle**	Teil / Bereich der Sammlung
Karteikarte		**Datensatz**	abgelegtes / gespeichertes / eingelagertes Objekt
Überschrift		**Primär-Schlüssel**	Name / Kennung des Objektes
Eintrag		**Attribut**	Merkmale des Objektes
Reiter od. Lochung		**(Primär-)Schlüssel**	hervorgehobene Such- und Kennungs-Merkmale

2.2.1. Sind Tabellen und Relationen ein und dasselbe?

Die Begriffe Tabellen und Relationen werden vielfach gleichbedeutend verwendet. In der Praxis ist dies auch oft ohne Probleme möglich. Informatiker nehmen die begriffliche Trennung nicht so genau.
Relationen sind erst einmal reine Zuordnungen von Daten zueinander. Einem Datum (hier Einzahl von Daten gemeint) wird ein anderes Datum oder eine Gruppe von Daten zugeordnet. Alle zusammengehörenden Paare oder Tupel bilden eine Relation.
So könnten wir vielleicht die Mitschüler so charakterisieren:

Tom(1,75; 69; grau; blond; 44; Michaelis)
Anne(1,68; 56; braun; braun; 39; Briesing)

Es wird hier das folgende Schema für die Tupel (Relations-Elemente) verwendet:

Vorname(Größe; Gewicht; Augenfarbe; Haarfarbe; Nachname)

Die Relation könnten wir als "Mitschüler" bezeichnen.
Der Vorname einer Person aus unserer Beispiel-Personengruppe ist mit bestimmten anderen Informationen verbunden. Aus dem Objekt (Entität) lassen sich bestimmte Attribute ableiten. Alle Ableitungen oder Verbindungen bilden eine Relation.
Die obigen Mitschüler-Daten lassen sich aber auch in einer Tabelle darstellen:

Vorname	Größe	Gewicht	Augenfarbe	Haarfarbe	Schuhgröße	Nachname
Tom	1,75	69	grau	blond	44	Michaelis
Anne	1,68	56	braun	braun	39	Briesing

Eine Tabelle ist also ersteinmal eine Form der Darstellung einer Relation, genau so, wie die obige Objekt-Attribut-Liste.
Die obige Tabelle stellt aber so nicht exakt die Relation Objekt-Attribut-Listen dar. Während in der Listen-Darstellung eine Zuordnungs-Richtung unterstellt wird, ist diese in einer Tabelle der obigen Form nicht vorhanden. Die Spalten könnten willkürlich getauscht werden. Für die Listen ist dies nicht so ohne weiteres möglich oder sinnvoll:

44(1,75; 69; grau; blond; Tom; Michaelis)
39(1,68; 56; braun; braun; Anne; Briesing)

(Bei ausschließlicher Betrachtung der gegebenen Entitäten wäre das aber möglich. Hier kommen alle Attribut-Werte nur einmal vor.)
Eine exakte Übersetzung der Listen ist eine Tabelle mit einer Vorrang- oder Bezugs-Spalte, die wir gemeinhin als Schlüssel-Attribut oder den Primär-Schlüssel (Primär-Spalte) verstehen:

Vorname	Größe	Gewicht	Augenfarbe	Haarfarbe	Schuhgröße	Nachname
Tom	1,75	69	grau	blond	44	Michaelis
Anne	1,68	56	braun	braun	39	Briesing

Man kann kann den Unterschied zwischen einer Relation und einer Tabelle auch am Beispiel einer mathematischen Funktion veranschaulichen. Nehmen wir als Beispiel mal die Quadrat-Funktion im Werte-Bereich von -3 bis 3:

$f(x) = x^2;\ \mathbb{R}: -3 \leq x \leq 3$ od. $y = x^2$

und betrachten nur einige Stütz-Punkte. Für x = -3 ergibt sich die Relation:

-3 → 9 -2 → 4 … 3 → 9

Diese können wir auch schön in einer Tabelle darstellen (s.a. rechts).
Beim Betrachten der Funktions-Werte (y-Werte) wird schnell klar die Funktion produziert nur in der Form x → y eindeutige Werte.
Versucht man y → x, dann können sich mehrere x-Werte ergeben. Diese Relation ist also nicht eindeutig.
Eine eindeutige Tabelle mit einer Charakterisierung der Relation muss also so aussehen (siehe rechts). Dies entspricht im informatischen Verständnis der Festlegung von x als den Primär-Schlüssel der Funktion bzw. der Tabelle.

x	y
-3	9
-2	4
-1	1
0	0
1	1
2	4
3	9

x	y
-3	9
-2	4
-1	1
0	0
1	1
2	4
3	9

Aufgaben:

1. *Gibt es eigentlich auch Funktionen, bei denen ein Funktions-Wert mehr als zwei x-Werten zugeordnet werden kann? Wenn JA, dann zeigen Sie das auf! Wenn NEIN, dann begründen Sie warum dies nicht geht!*
2. *Ein ewig zweifelnder Schüler behauptet, dass es aber auch Relationen gibt, bei denen sich die Seiten (bzw. die Ableitungs-Richtung) tauschen lassen. Setzen Sie sich mit dieser Behauptung auseinander!*
3. *Suchen Sie sich mindestens zwei weitere Relations-Gruppen (keine mathematischen Funktionen) heraus und charakterisieren Sie die Art der Relation!*
3.

Definition(en): Relation (allg.)
Eine Relation ist eine Menge von Objekten, die durch Paare / Tupel einer eindeutigen und vollständigen Attributs-Kombination gebildet wird.

2.2.0. Grund-Begriffe, -Elemente und –Verfahren in relationalen Daten-bank-Modellen

Modellierung z.B. über Entity-Relationship-Modell (ERM)

Schlüssel-Kandidaten
Candidate Keys

die Attribute, die als Schlüssel-Kandidat eingesetzt werden sollen müssen eindeutig sein, d.h. jedes Objekt / jede Entität muss unverwechselbar über seinen Attribut-Wert indentifizierbar sein
kommen völlig gleichartige Objekte mehrfach vor und gibt es keinen inneren Schlüssel-Kandiadten, dann muss ein weiteres Attribut – z.B. eine einfache Aufzählung – als "zusätzliches" Attribut hinzugefügt werden
Schlüssel-Kandidaten müssen auch irreduzierbar (nicht ableitbar) sein. D.h., dass nach dem Entfernen von einem oder mehreren Attributen die Eindeutigkeit verloren geht. Mit anderen Worten, wenn man bei (aus mehreren Attributen) zusammengesetzten Schlüssel-Kandidaten ein Attribut entfernt, dann darf sich diese Kombination nicht als Schlüssel-Kandidat eignen, weil die Objekte / … nicht mehr eindeutig identifizierbar sind.
Nur alle gewählten Attribute eines zusammegesetzten Schlüssel-Kandidaten zusammen ermöglichen die eindeutige Kennzeichnung.

Definition(en): Schlüssel-Kandidat
Ein Schlüssel-Kandidat ist ein Attribut oder eine Attribut-Kombination, welche(s) sich prinzipiell zur Festlegung als Primär-Schlüssel eignet.
Schlüssel-Kandidaten sind die Attribute oder Attribut-Kombinationen, die sich zur eindeutigen und redundanzfreien Kennzeichnung von Objekten / Datensätzen / Entitäten eignen.

Primär-Schlüssel

Primary Key

bei der Festlegung eines Primär-Schlüssel's sind einfache Schlüssel-Kandidaten den zusammengesetzten vorzuziehen

in Computer-Systemen sind nummerische Attribute besser als Primär-Schlüssel geeignet, da ihre Verarbeitung optimierter abläuft, als z.B. das Durchsuchen / Identifizieren von Zeichenketten

im Zweifelsfall ist es effektiver, statt eines zusammengesetzten Schlüssel's einen zusätzlichen nummerischen Schlüssel (z.B. fortlaufende Aufzählung) zu benutzen

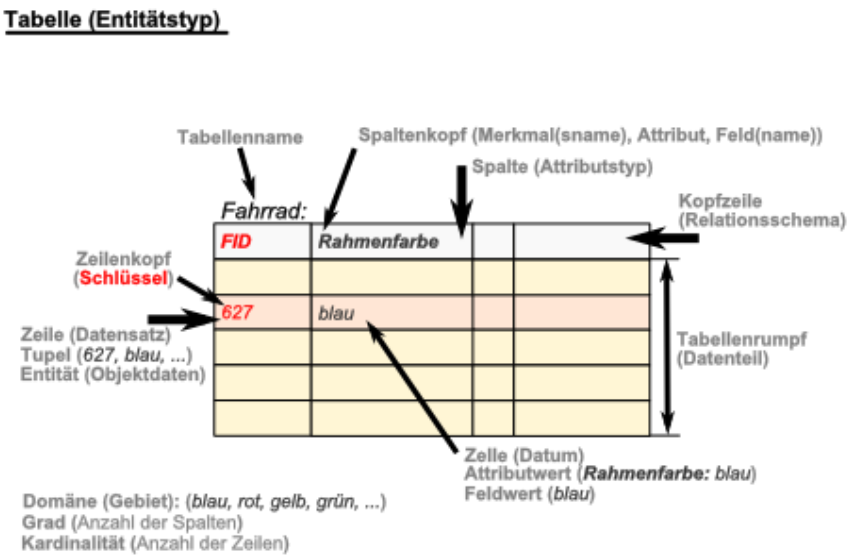
Definition(en): PrimärSchlüssel
Der Primär-Schlüssel ist das Attribut einer Tabelle / Relation, anhand derer jedes einzelne Objekt / jeder Datensatz / jede Entität in der Tabelle wiedergefunden / erkannt werden kann.

es wird empfohlen solche Attribute als Primär-Schlüssel zu verwenden, die für die Anwender keinerlei Bedeutung haben (und sich deshalb ändern könnten)
so könnte z.B. die Artikel-Nummer in einem Shop ein Schlüssel-Kandidat und auch eine Primär-Schlüssel in einer Artikel-Tabelle sein, bei einer Neuorganisation der Artikelnummern kann es aber zu gewaltigen Änderungs-Kaskaden kommen
das spricht ganz stark für eine zusätzliche ID-Spalte

Fremd-Schlüssel

Objektreferenzen in Beziehungen

Definition(en): Fremd-Schlüssel
Ein Fremd-Schlüssel ist ein Verweis auf einen anderen Eintrag gewöhnlich in einer anderen verknüpften / referenzierten Tabelle / Relation. (Der Verweis ist der Primär-Schlüssel in der verknüpften / referenzierten Tabelle.)



Nachteile:

Daten sind segmentiert; Segemente können wieder segmentiert sein
durch (häufige / mehrfache) Indizierung sind Perfomance-Probleme bei größeren Daten-Mengen zu erwarten
Manipulation der Daten durch indirkete und indizierende Strukturen nur über Programmier-sprachen und spezielle Algorithmen möglich
haben zumeist eine flache Struktur (Objekte werden "flach geklopft"); Aufbau ist aus dem Design nicht mehr direkt ersichtlich
keine Rekursion möglich (→ Stücklisten-Problem)
JOINs auf zusammgehörige Reationen veringern System-Leistung

grundlegende Tabellen-Typen

- **Master-Tabelle** — Tabelle ist Sammlung der Objekt-Eigenschaften eines Objekttypes / einer Objektklasse; als Schlüssel sollten Zeichenfolgen (bei kleinen Tabellen) und sonst Ganzzahlen verwendet werden
- **Referenz-Tabelle** — relativ dauerhafte Tabelle mit wenigen Spalten (meist nur 2 (Schlüssel und Wert)); möglichst Zeichenfolgen als Primär-schlüssel
- **Querverweis-Tabelle** — beinhaltet die Beziehungen zwischen Master-Tabellen; als Schlüssel werden zumeist Kombinationen aus mehre-ren Spalten genutzt
- **Transaktions-Tabelle** — speichert die Interaktionen und deren Ergebnisse zwi-schen Master-Tabellen; als Schlüssel werden häufig au-tomatisch generierte Ganzzahlen verwendet

referenzielle Integrität

Integrität wird hier als Korrektheit der Daten im Zusammenhang mit anderen Daten verstan-den
bei der referenziellen Integrität wird gefordert, dass jede Referenz auch mit einem existie-renden Eintrag / Objekt / Datum verbunden ist, d.h. es darf keinen Verweis auf ein nicht-existrierendes Objekt erstellt oder erzeugt werden.
Ist dies passiert, dann ist die referenzielle Integrität verloren gegangen
Reparaturen solcher Daten-Verluste / Daten-Fehler sind extrem aufwendig und nur selten maschinell lösbar

Definition(en): referentielle Integrität
Unter referentieller Integrität versteht man die Konsistenz, der in einer Tabelle angebenen Fremd-Schlüssel-Wert, d.h. zu jedem angegebenen Fremd-Schlüssel existiert in der refe-renzierten / verknüpften Tabelle ein gültiger Primär-Schlüssel-Wert.

Um dem Problem mit (versehendlich) gelöschten referentierten Daten / Schlüsseln aus dem Weg zu gehen, kann man ein geändertes Lösch-Konzept in seiner Datenbank integrieren.
Dazu werden die Tabellen mit einem zusätzlichen Attribut "gelöscht" ("IsDeleted", "Deleted") versehen. Standard-mäßig setzt man das Attribut auf ***false***. Im Lösch-Fall wird nur dieses Attribut auf ***true*** gesetzt.
Man kann aber auch das negierte Konzept einsetzen. Dabei nutzt man z.B. das neue Attribut "aktiv". Dieses wird dann normal auf ***true*** gesetzt und im Falle, dass der Datensatz nicht mehr benutz werden soll auf ***false***. Dieses Konzept ermöglicht verständlichere SQL-Ausdrücke.

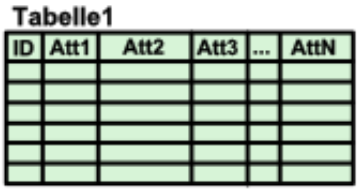
Vorteile:

- Daten werden praktisch nie gelöscht
- ermöglicht sogenanntes "weiches Löschen"
- leichtes Reaktivieren von "gelöschten" (als "gelöscht" markierte) Daten

Nachteile:

- aufwendigere SQL-Anweisung (weil "gelöscht"-Merkmal mit beachtet werden muss)
- Daten-Bestand wächst ständig (bei sich häufig ändernden Daten besonders schnell)
- Datenschutz-Auflagen werden ev. (langfristig) nicht exakt umgesetzt
- für "hartes Löschen" müssen extra Anweisungen und Arbeits-Zyklen geplant werden

Master-Tabellen sind die klassischen Daten-Tabellen in relationalen Datenbanken. Je Objekt / Ding / Sachverhalt, der in der Tabelle gespeichert werden soll existiert eine Zeile. Die Spalten enthalten die Attribute der Objekte, wobei jedes Objekt eine charakterisierende ID (Identifizierung) besitzt.



Unter Umständen werden bei Attributen nicht die Rohdaten eingetragen, sondern nur eine Referenz auf eine andere Tabelle. In dieser sind dann ev. noch weitere Detail zur Referenz enthalten.

In der Tabelle1 ist unter Attr3 nur eine ID aus Tabelle2 eingetragen. Werden die Informationen dazu gebraucht, dann wird in dieser Tabelle nachgeschaut und deren Attribute ausgelesen.

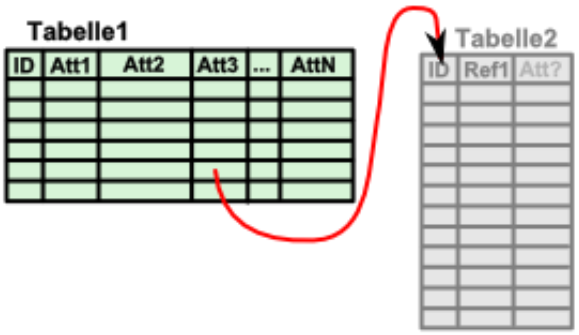
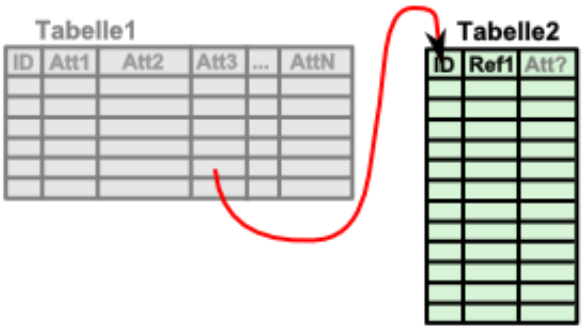


Tabelle mit einer Referenz auf eine andere

Solche Tabellen sind häufig die Ergebnisse von Normaliserungen. Darunter verstehen wir grob betrachtet Verbesserungen der Redundanzen.

Die Tabelle2 ist somit eine Referenz-Tabelle.

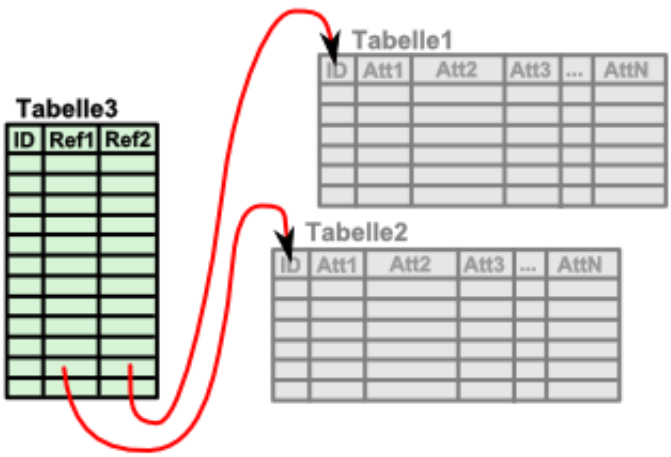


Referenz-Tabelle enthält Detail-Informationen

Sie enthält solche Objekte, die in Master-Tabellen häufiger vorkommen (können) bzw. einen spezifischen Sachverhalt abdecken. Datenbanken werden durch solche Referenz-Tabellen übersichtlicher und leichter zu warten. Bei Veränderungen im Datenbestand dieser Tabelle muss man sich primär nur um diese (separate) Tabelle kümmern.

In Querverweis-Tabellen sind vorrangig die Verbindungen (Relationship's) dargestellt. Wir sprechen auch von Verknüpfungen.

So werden in Tabelle3 die Beziehungen zwischen den Objekten in Tabelle1 und Tabelle2 (mit ihren Objekten) gespeichert.



Eine Querverweis-Tabelle enthält außer einer eigenen ID-Spalte immer mindestens noch zwei Spalten mit Verweisen / Referenzen auf andere Tabellen. Daneben können noch weitere Attribute enthalten sein, die spezielle Informationen zur Verknüpfung enthält.

erweiterte Tabellen-Typen

- **begrenzte Transaktion** beschreibt Einschränkungen / Zulässigkeiten für Transaktionen

- **vergänglicher Primär-Schlüssel** wird für veränderliche Schlüssel verwendet; Veränderungen der Schlüssel werden gespeichert (History-Tabellen)

andere Unterscheidung / Benennung von Tabellen einschließlich ihrer Funktion

Tabellen	**Merkmale** **SQL-Anweisung**	**Funktion**
Basis-Relationen	enthalten die eigentlichen Daten, sind real und dauerhaft in der Datenbank gespeichert CREATE TABLE	Datenspeicherung
Sichten **View's**	virtuelle bzw. nicht reale Tabellen, die praktisch Ausschnitte aus Basis-Tabellen darstellen CREATE VIEW	Anzeige der notwendigen bzw. der Daten, zu denen der Nutzer Zugriffsrechte besitzt
Abfragen **Abfrage-Ergebnisse** **Query-Results**	temporäre Tabellen für Monitore, Drucker usw. SELECT	Filtern, Gruppieren und Sortieren von Daten
temporäre Relationen	temporäre Abfrage-Ergebnisse, bleiben aber erhalten bis bestimmte Transaktionen oder Datenbank-Operationen (Aktionen) beendet sind bzw. die Daten durch Aktionen zerstört werden CREATE TEMPORARY TABLE	dient zum Speichern / Sichern von Zwischen-Ergebnissen

Die Integrität der Daten (→ 2.0.2.3. Daten-Integrität) ist eines der entscheidenden Charakteristika eines Datenbank. In relationalen Datenbank ergeben sich einige spezielle Anforderungen an die Integrität.

relationale Integritäts-Regeln

- **physische Integrität** — steht für die Vollständigkeit der Zugriffs-Pfade und der physikalischen Speicherstrukturen (nicht vom Datenbank-Programmierer beeinflussbar; Betriebssystem-Ebene)
  Funktionsfähigkeit der Hardware
  verantwortlich: Hardware-Ausstatter, Einkäufer
- **Ablauf-Integrität** — Korrektheit der eingesetzten Algorithmen und ablaufenden Programme
  keine Daten-Inkonsistenzen im Mehrbenutzerbetrieb
  verantwortlich sind Datenbank-Designer und Anwendungs-Programmierer
- **Zugriffs-Integrität** — korrekte Vergabe von Zugriffsrechten
  korrekte Umsetzung des Rechte-Systems
  verantwortlich: Datenbank-Administrator; Programmierer
- **semantische Integrität** — Übereinstimmung der Daten mit der realen Welt durch Überprüfung während der Eingabe
  (z.B. durch enge Begrenzung der Domänen (Wertebereiche))
  verantwortlich: Anwendungs-Programmierer, Datenbank-Adminsistrator, Hersteller des DBS

Entitäts-Regel – 1. Integritäts-Regel
Jeder Primärschlüssel identifiziert die Tupel eindeutig.
Jedes Tupel hat einen eindeutigen Primärschlüssel.
Kein Tupel darf als Primärschlüssel ***nichts*** enthalten (leere Primärschlüssel sind unzulässig).

Referenz-Integritäts-Regel – 2. Integritäts-Regel
Fremdschlüssel verbinden / verknüpfen Tabellen um zusammengehörende Objekte in verschiedenen Tabellen in Beziehung zu setzen.
Eine relationale Datenbank darf keine Fremdschlüssel beinhalten, die auf einen nicht existierenden Primärschlüssel verweisen.

Lösungs-Ansätze mit SQL-Ausschnitt (s.a. → 5.1.6.1. Tabellen mit Fremdschlüsseln erstellen)

1. gewünschtes Löschen bzw. Ändern nicht durchführen
   ON DELETE NO ACTION
   ON UPDATE NO ACTION
2. gewünschtes Ändern oder Löschen rekursiv auf verwiesene Tupel anwenden
   ON DELETE CASCADE
   ON UPDATE CASCADE
3. NULL-Setzen aller darauf verweisender Fremdschlüssel
   ON DELETE SET NULL
   ON UPDATE SET NULL

Gegenüberstellung und Einordnung von Grundbegriffen verschiedener informatischer Modelle

allgemein	**Relationen-Modell**	**Entity-Relationship-Modell (ERM)**	**Relationale Datenbank**	**Unified Modeling Language (UML)**
Tabelle	Relation	Entitätsmenge Entitäts-Set	Tabelle	Objektmenge, Instanzmenge, Klasse
Tabellenname	Relations-Name	Entitäts-Typ		
Spalte	Attribut			
Spaltenkopf **Spaltenüberschrift**	Attribut-Name	Attribut	Spaltenüberschrift	Attribut
Zeilenkopf **(Kopfspalte)** **(Zeilenüberschrift)**	Schlüssel	funktionale Beziehung (Relationship)	Primärschlüssel	Assoziation
Zeile	Tupel	Entität	Datensatz Zeile	Objekt, Instanz
Kopfzeile	Relationstyp Relationsformat	Entitäts-Typen	Relations-Schema	Klasse, Objekttyp
Zelle	Attribut-Wert Wert	Attribut-Wert	Feld Zelle	Attributwert
Menge zulässiger Einträge	Wertebereich (Domäne)	Wertebereich (Domäne)	Wertebereich (Domäne)	Wertebereich (Domäne)
Spaltenanzahl	Grad			
Zeilenanzahl	Kardinalität			
Tabellen-Körper / **Daten-Zeilen**	Relations-Instanz			

!!! noch prüfen!!!

Eine saubere Trennung zwischen den Modellen und Ebenen wird in der Praxis kaum vorgenommen.

Relationen-Schema

In einem Relationen-Schema (auch Realtions-Schema) stellen wir eine Datenbank oder Teile aus ihr in einer Text-basierten Form dar. Dabei wird einem Namen (Name aus ERD oder Tabellen-Name) eine Menge an Attributen zugeordnet. Dieses entspricht später den Spalten in den benannten Tabellen.
Das Grund-Schema sieht so aus:

RelationenSchema = (Attribut **{**, Attribut**}**)

Ein Relationen-Schema ist also praktisch ein Tupel aus den Eigenschaften eines Objektes. Es stellt weiterhin einen Datensatz der entsprechenden Tabelle (Relation) dar.
Die Relation (Tabelle) hat dann die folgende Notierung

Relation(RelationenSchema) oder Relation(Attribut **{**, Attribut**}**)

Wir brauchen also mindestens ein Attribut (eine Spalte) in der Tabelle. Dies ist minimal der Primär-Schlüssel. Um die Primär-Schlüssel deutlich von anderen Attributen abzugrenzen, werden diese fett geschrieben oder unterstrichen. Ich verwende hier beides kombiniert.

RelationenSchema = (**ID**, Attribut1, Attribut2, **…** , AttributN)

Manche Tabellen enthalten Fremdschlüssel. Auch sie werden besonders gekennzeichnet, da sie die Beziehungen zu anderen Tabellen beschreiben. Im Allgemeinen benutzt man einen aufrechten oder leicht schrägen Pfeil vor dem Attribut-Namen zur Typisierung.

Tabelle = (**ID**, Attribut1, Attribut2, ↗FremdID, … , AttributN)

Das folgende Relationen-Schema kann als Beispiel gelten:

Schueler= (**SID**, Name, Vorname, GebDatum, GebOrt, Geschlecht)

Klasse = (**KID**, Bezeichnung, KlassenLeiter, KlassenRaum)

IstInKlasse = (**IIKID**, ↗KID, ↗SID)

Nach KEMPER und EICK werden Realtionenschemata mit Daten-Typen notiert, Sie verwenden folgende Struktur:

Realtionenschema : { [Attribut1: Typ, Attribut2: Typ], **…**, AttributN: Typ }

Relationship's mit Fremdschlüsseln werden genau so notiert, d.h. auch wieder mit Attribut-Name und Typ sowie der Schlüssel-Kennzeichnung. Ein Verweis-Pfeil oder soetwas ähnliches gibt es hier nicht.
Realtionen werden dann in der Form:

Relation : { Attribut1. Attribut 2, **…**, AttributN }

notiert.

Für praktische Anwendungen könnte man sicher eine Kombination der oben besprochenen Realtionenschema-Strukturen nutzen. Die Verweis-Pfeile bieten schließlich wichtige Implementierungs-Hinweise.
Eine weitere – mehr Implementierungs-orientierte – Formulierungs-Form verwendet wieder nur die Attributnamen und erweitert die Definition durch ein Zusatz für die Fremdschlüssel. Ein Realtionenschema wird dann so strukturiert:

Relation1 (ID, Name, …)

Relation2 (EigenID, ID, Attribut, …)
FOREIGN KEY (ID) REFERENCES Relation1

Die EigenID ist nicht unbedingt notwendig. Wird sie nicht verwendet, dann muss ID (aus relation1) als Fremdschlüssel die Funktion des Primärschlüssel's (für Relation2) übernehmen.
In dieser Art der Definition sind auch Neu- bzw. Umbenennungen der Fremd-Schlüssel möglich:

Relation1 (ID, Name, …)

Relation2 (EigenID, **Rel1ID**, Attribut, …)
FOREIGN KEY (**Rel1ID**) REFERENCES Relation1(**ID**)

Viele Relationen-Schemata lassen sich gut 1 : 1 in ein PROLOG-System übertragen. Dabei bilden die Relationen-Schemata die Notierungs-Hilfe für die Atome einer PROLOG-Anwendung oder –Datenbank. Die Attribut-Werte finden sich in der Argument-Liste wieder. Der Name der Relation wird durch den Funktor abgebildet.
Praktisch notieren wir die Tupel (Datensätze) als Atome.
(Problematisch ist die fehlende Schlüssel-Unterstützung von PROLOG. Es kann nicht automatisch sichergestellt werden, dass bestimmte Einträge immer eindeutig bezüglich der Tabelle (Primärschlüssel-Eigenschaft) sind. Auto-Inkremente oder ähnliches lassen sich nicht realisieren. Das wesentliche Hindernis ist dabei, dass der Nutzer die Daten-Basis (Sammlung der Atome) jederzeit händisch ändern oder ergänzen kann. In PROLOG lassen sich Beziehungen aber über Regel abbilden.)

Fakten / Atome

```
dsSchueler(1276, 'Meier','Anna','04.07.2001','Berlin','w').
dsSchueler(1196, 'Bauer','Erny','27.06.2001','Halle','m').
…
dsSchueler(1502, 'Zander','Jo','15.10.2000','Rostock','w').

dsKlasse(72,'11/4','Fr. Labs','2.015').
dsKlasse(73,'11/5','Hr. Grün','2.004').
…
dsKlasse(77,'12/1','Fr. Herder','1.005').


dsIstInKlasse(2764,73,1276).
dsIstInKlasse(2765,73,1502).
…
dsIstInKlasse(2794,77,1196).
```

Regeln

```
…
istInKlasse(Name,Vorname,Klasse):- dsSchueler(S,Name,Vorname,_,_,_)),
                                   dsKlasse(K,Klasse,_,_),
                                   dsIstInKlasse(_,K,S).
…
```

Relationen-Schemata sind auch gut mathematisch betrachtbar. In vielen Schreibweisen einer Relationen-Algebra (→ 4.0. Relationen-Algebra / Relationen-Kalküle) nutzt man diese Form der Notierung für die Beschreibung der Tabellen.

2.2.1. Überführung der Entity-Relationship-Modell's in ein relationales

also Transformation aus der externen Ebene (/ Phase) auf die konzeptionelle Ebene (/ Phase

Abbildungen:

- Entitäts-Typ → Relation
- Attribut → Attribut
- Beziehungs-Typ → Fremd-Sschlüssel oder einer zusätzlichen Relation

Überführungen:

- starke Entitäts-Typen: Anlage einer Relation mit einem geeigneten Primär-Schlüssel
- schwache Entitäts-Typen: Anlage einer Relation mit den Attributen und einem internen Primär-Schlüssel sowie einem Fremd-Schlüssel der Relation eines starken Entitäts-Typ's
- 1:1-Beziehungen: Erweiterung einer der beteiligten Relationen um den Fremd-Schlüssel der anderen Relation
- 1:n-Beziehungen: die Relation mit der n-Kardinalität wird um den Fremd-Schlüssel der 1-Kardinalität ergänzt
- n:m-Beziehungen: Anlage einer neuen Relation mit eigenen Attributen (dieser Beziehung) und den Fremd-Schüsseln beider beteiligter Relationen
- mehrwertige Attribute: Anlage einer Relation mit den mehrwertigen Attributen und einem Fremd-Schlüssel auf die ursprüngliche Relation
- n-äre Beziehungs-Typen: Anlage einer Relation mit den Fremd-Schlüsseln aller / zu allen beteiligten Relationen sowie eigenen Attributen

Exkurs: CODD's Regel zur Spezifikation einer relationalen Datenbank

2.2.2. Normalisierung

Problem komplexe Beziehungen zwischen Tabellen
Problem mit der Redundanz und strukturierten Attributen

Überführung in einfache Beziehungen und übersichtliche Strukturen
Tabellen mit weniger Einträgen insgesamt
stabile und flexible Daten-Strukturen

Eine Relation ist dann normalisiert, wenn:

- sie Redundanz-frei ist
- keine Probleme bei der Daten-Pflege verursacht
- sie beschreibt die Modellwelt (als Ausschnitt aus der Realität) angemessen und exakt

Eleminierung von (unnötigen / vermeidbaren) Redundanzen

Alle gleichartigen aber mehrfach auftretenden Bezeichnungen usw. führen im Nutzer-Betrieb zu Problemen. Da werden gleiche Objekt ausversehen unterschiedlich geschrieben. Schon einzelne Leerzeichen können zu unterschiedlichen Objekten führen. Nehmen wir z.B. die Orts-Bezeichnung: Musterhausen – Nord.
Die normale Zeichenkette – in der Informatik häufig String genannt – besteht hier im Beispiel aus 19 Zeichen.

Zeichen-Position:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
Zeichen:	**M**	**u**	**s**	**t**	**e**	**r**	**h**	**a**	**u**	**s**	**e**	**n**		**-**		**N**	**o**	**r**	**d**		

Gleich beim ersten oder zweiten Eingeben stellt sich für den Datenbereitsteller die brennende Frage, wie wird das nun richtig geschrieben – mit oder ohne Leerzeichen am Bindestrich?

Zeichen-Position:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
Zeichen:	**M**	**u**	**s**	**t**	**e**	**r**	**h**	**a**	**u**	**s**	**e**	**n**	**-**	**N**	**o**	**r**	**d**				

Und schon existieren zwei Bezeichnungen für den gleichen Ort in der Datenbank. Und das Chaos hat erst angefangen …

Zeichen-Position:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
Zeichen:	**M**	**u**	**s**	**t**	**e**	**r**	**h**	**a**	**u**	**s**	**e**	**n**		**-**	**N**	**o**	**r**	**d**			
Zeichen:	**M**	**u**	**s**	**t**	**e**	**r**	**h**	**a**	**u**	**s**	**e**	**n**	**-**		**N**	**o**	**r**	**d**			

Zeichen-Position:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
Zeichen:	**M**	**u**	**s**	**t**	**e**	**r**	**h**	**a**	**u**	**s**	**e**	**n**		**-**		**N**	**o**	**r**	**d**		

Technisch brauchen wir noch mindestens eine Speicher-Einheit, um die Begrenzung der Zeichenkette kenntlich zu machen. Dazu gibt es unterschiedliche Verfahren. Zum Ersten kann man das Ende einer Zeichenkette durch ein spezielles Zeichen festlegen. Im obigen Beispiel würde dieses dann in der Position 20 gespeichert. Ab Position 20 würde dann vielleicht eine neue Zeichenkette folgen – dann natürlich mit seiner eigenen Nummerierung. Die zweite Möglichkeit ist eine Längen-Angabe in einer Speicherzelle. Vielfach wir hierfür die 0. Zeichen-Position genutzt. In underem Beispiel würde dann die 19 darin gespeichert sein.

Für unsere Datenbank-Betrachtungen spielen diese technischen Details keine Rolle. Das gehört zur physischen Ebene und damit in den Bereich des Betriebssystem. Werden Datenbanken zwischen Betriebssystem mit verschiedener Zeichenketten-Codierung ausgetauscht, dann muss das Datenbank-Management-System entsprechend flexibel sein.

Definition(en): Normalisierung
Unter Normalisierung versteht man die Umorgansierung / (Neu-)Aufteilung der Daten einer Tabelle in der Form, dass sie keine vermeidbaren Redundanzen mehr enthält.
Normalisierung ist die Neu-Strukurierung der Daten einer Tabelle in mehrere Teil-Tabellen, um Dopplungen von Daten-Gruppen zu vermeiden. Durch Normaliserung soll die Datenbank optimiert werden und weniger Problemen bei Update's von Daten unterliegen.

häufig wird der Begriff der Normalsierung auf ganzen Datenbanken bezogen, was sachlich nicht ganz exakt ist
die Ausweitung der Normalisierung lässt sich als Maß für die Qualität einer Datenbank benutzen
wird deshalb auch als Entwurfs-Prinzip genutzt

(klassische) Algorithmen für die Normalisierung

- **Synthese-Algorithmus** → 3NF (3. Normalform)
- **Zerlegungs-Algorithmus** → BCNF (BOYCE-CODD-Normalform)
-

manchmal verzichtet man bewußt auf die Normalsierung bzw. macht diese wieder rückgängig (Denormalisierung), um

- **Verarbeits-Geschwindigkeit zu erhöhen** — Mehrfach-Verweise erfordern erneute Suchvergänge in immer neuen Tabellen
- **Anfragen zu vereinfachen** — Mehrfach-Verweise sind gedanklich, aber auch technisch kompliziert und aufwendig zu programmieren
  mehrfache JOIN's verlangsamen das System, da i.A. immer die ganzen betroffenen Tabellen bearbeitet werden müssen
- **Fehler-Anfälligkeiten zu verringern** — Mehrfach-Verweise lassen Fehler oder Besonderheiten zu, die eigentlich nicht gewollt sind; Fehlerfindung sehr aufwendig
- **spezielle Prozesse abzubilden** — z.B. Geschäfts-Prozesse

Von einer **funktionale**n **Abhängigkeit** spricht man, wenn ein oder mehrere Attribute ein anderes Attribut bestimmt.
Die Attribute "**Name**" und "**Vorname**" sind funktional abhängig vom Attribut "**ID**".

ID	Name	Vorname
2	Müller	Heinz
8	Bauer	Claudia
15	Müller	Frank
26	Meiser	Claudia
32	Bauer	Monika

ID → Name *__ID__ bestimmt funktional __Name__*
ID → Vorname
ID → {Name, Vorname}

Der **Identifikations-Schlüssel** ist ein Attribut dessen Werte jeweils immer nur einmalig innerhalb des Attributes / der Spalte vorkommen.
Die Spalten "**Name**" und "**Vorname**" eignen sich nicht als Schlüssel, da hier Doppelungen auftreten könnten. Dies meist jetzt unterschiedliche Personen mit eben zufällig dem gleichen Namen.

ID	Name	Vorname
2	Müller	Heinz
8	Bauer	Claudia
15	Müller	Frank
26	Heinz	Claudia
32	Bauer	Monika

Nur das Attribut "**ID**" eignet sich – wegen der jeweils einmaligen Werte – als Identifikations-Schlüssel. Würde man allerdings z.B. "**Name**" und "**Vorname**" kombiniert betrachten, dann könnte auch die Kombination als Identifikations-Schlüssel dienen.

Von einer **volle**n **funktionale**n **Abhängigkeit** spricht man, wenn ein Attribut durch die Werte von zwei oder mehreren Attributen abhängig ist und die Kombination von Attribut-Werten immer den gleichen Wert im funktional abhängigen Attribut ergibt.
Hier ist die Note voll funktional abhängig von der Kombination aus Schüler, Lehrer und Fach.
Der Schüler kann nicht bei der gleichen Kombination mit Fach und Lehrer eine andere Note bekommen.

ID	SID	SName	SVorname	LehrerID	FachID	Note
1	382	Behrens	Dorothea	KOL	Ma	2
2	735	Decker	Nils	EZN	Bio	2
3	193	Zenkert	Lisa-Marie	KOL	Ma	3
4	443	Neuber	Sven	MEI	Chem	2
5	864	Müller	Albert	ZAN	Ma	3
6	742	Koch	Ina	EZN	Bio	4

{SchülerID, LehrerID, FachID} **==>** Note *__SchülerID, LehrerID__ und __FachID__ bestimmen voll-funktional __Note__*

Zwischen den Attributen "SID", "SName" und "SVorname" besteht eine **transitive Abhängigkeit**. Alle drei Attribute sind kein Schlüssel-Attribut dieser Tabelle (das ist "ID").

ID	SID	SName	SVorname	LehrerID	FachID	Note
1	382	Behrens	Dorothea	KOL	Ma	2
2	735	Decker	Nils	EZN	Bio	2
3	193	Zenkert	Lisa-Marie	KOL	Ma	3
4	443	Neuber	Sven	MEI	Chem	2
5	864	Müller	Albert	ZAN	Ma	3
6	742	Koch	Ina	EZN	Bio	4

Die Werte der Attribute "SName" und "SVorname" sind funktional abhängig von "SID". Es hätte also gereicht nur "SID" in der Tabelle aufzuführen.

Definition(en): funktionale Abhängigkeit
Bei einer funktionalen Abhängigkeit bestimmen einzelne Attribute eindeutig den Wert anderer Attribute.

Bei einer funktionalen Abhängigkeit tauchen also bei Gruppen von Attributen immer die gleichen Gruppen von Werten auf. Mit anderen Worten statt der Gruppe von Attributen / Werten kann man auch mit einem einzelnen Stellvertreter-Wert arbeiten, hinter dem die Gruppe von Attributen / Werten versteckt ist.
Funktionale Abhängigkeit tritt meist dann auf, wenn in einem Objekt / einer Entität in einer Tabelle / Relation mehrere Daten zu einer anderen Entität gespeichert werden. Kommt diese Entität mehrfach in der Tabelle / Relation vor, dann ist eine Auslagerung deren Daten in einer neuen (referenzierten) Tabelle / Relation zu prüfen. Das Verfahren der Umstrukturierung solcher Tabellen / Relationen wird Normalisierung genannt.
Die Ergebnisse eine Normalisierung bezeichen wir als eine normalisierte Tabelle. Die Tabelle befindet sich dann in einer Normal-Form. Es werden mehrere Stufen der Normaliserung unterschieden.

2.2.2.1. Nullte Normalform

Um eine Relation in der Nullten Normalform zur weiteren Bearbeitung zur Verfügung zu haben müssen die Elemente der abzubildenden Realität (→ Miniwelt) in einer Tabelle abgebildet werden.
Das sein z.B. die nebenstehenden Personen aus einem Zettel-Personen-Register.
Zur Identifizierung benutzen wir die Zettel-Nummer. Vielleicht waren die mal dazu da, die Entstehungs-Reihenfolge der Zettel zu verdeutlich.
Jeder Zettel wird zuerst einmal als neue Zeile in die Tabelle übernommen.

ID	Name	Vorname
2	Müller	Heinz
8	Bauer	Claudia
15	Müller vom Stein	Frank-Emanuel Phillip
26	Meiser	Claudia
32	Bauer	Monika
41	Müller vom Stein	Frank-Emanuel Phillip
46	Bauer	Claudia

Nun werden doppelte Einträge entfernt und / oder durch zusätzliche Merkmale unterscheidbar gemacht.
Bei "Frank-Emanuel Phillip Müller vom Stein" sind wir uns vielleicht sich, dass dieser Name nicht wirklich zweimal vorkommt. Also könnten wir die doppelte Zeile (41) wahrscheinlich unbedenklich löschen.
Beim "Claudia Bauer" ist das nicht so eindeutig. Die Wahrscheinlichkeit, dass es eine zweite Person mit diesem Namen gibt ist ziemlich hoch. Hier wäre ein Löschen (von Zeile 46) eher schädlich.

Sicherheitshalber ergänzt man die Tabelle ersteinmal um weitere Daten.
Nehmen wir an, uns ständen die Geburtsdaten zur Verfügung. Also nehmen wir sie in unsere Roh-Tabelle mit auf.

ID	Name	Vorname	GebDatum
2	Müller	Heinz	17.01.1978
8	Bauer	Claudia	08.10.1984
15	Müller	Frank-Emanuel Phillip	23.03.1995
26	Meiser	Claudia	31.12.1997
32	Bauer	Monika	08.10.1984
41	Müller vom Stein	Frank-Emanuel Phillip	23.03.1995
46	Bauer	Claudia	04.08.1993

Nun können wir die gleichen Personen auf Zettel 15 und 41 bestätigen und den doppelten Datensatz unbedenklich entfernen.
Für "Claudia Bauer" ergeben sich wirklich zwei Personen, so dass wir hier nicht löschen.

ID	Name	Vorname	GebDatum
2	Müller	Heinz	17.01.1978
8	Bauer	Claudia	08.10.1984
15	Müller	Frank-Emanuel Phillip	23.03.1995
26	Meiser	Claudia	31.12.1997
32	Bauer	Monika	08.10.1984
41	Müller vom Stein	Frank-Emanuel Phillip	23.03.1995
46	Bauer	Claudia	04.08.1993

Übrig bleibt eine Tabelle, die wir in der Nullten Normalform sehen.
Dazu muss vielleicht noch gesagt werden, dass es eine echte Nullte Normalform nicht wirklich gibt.
Sie ist mehr eine vorgedachte Arbeits-Tabelle für die weiteren Normalisierungen.

ID	Name	Vorname	GebDatum
2	Müller	Heinz	17.01.1978
8	Bauer	Claudia	08.10.1984
15	Müller	Frank-Emanuel Phillip	23.03.1995
26	Meiser	Claudia	31.12.1997
32	Bauer	Monika	08.10.1984
46	Bauer	Claudia	04.08.1993

Weiterhin entfernt man u.U. noch aus anderen Tabellen-Teilen (Felder) berechnete (Ergebnis-)Felder. Damit soll vermieden werden, dass man es vergisst, diese Felder bei Daten-Änderungen neu zu berechnen.

ID	Kunde	Artikel	Anzahl	EinzPreis	GesPreis
1	Müller	Arbeitshose blau 54	2	45,00 €	90,00 €
2	Zander	Kittel weiß M	1	23,00 €	23,00 €
3	Friedrich	Kochhemd weiß XL	4	17,50 €	70,00 €
4	Stein	Kochhose grau XXL	2	18,95 €	37,90 €
5	Stein	Kochhemd weiss XXL	3	17,50 €	52,50 €
6	Bauer	Arbeitsanzug grau 50	1	36,75 €	36,75 €

Man könnte dann später solche Inkonsitenzen nicht mehr aufklären und beheben.
Da die Spalte "GesPreis" immer wieder aus "Anzahl" und "EinzPreis" berechnet werden kann, verzichtet man auf ein Speichern dieser redundanten Daten.
Übrig bleibt eine "saubere" Datentabelle (in der 0.NF).

ID	Kunde	Artikel	Anzahl	EinzPreis
1	Müller	Arbeitshose blau 54	2	45,00 €
2	Zander	Kittel weiß M	1	23,00 €
3	Friedrich	Kochhemd weiß XL	4	17,50 €
4	Stein	Kochhose grau XXL	2	18,95 €
5	Stein	Kochhemd weiss XXL	3	17,50 €
6	Bauer	Arbeitsanzug grau 50	1	36,75 €

2.2.2.2. Erste Normalform

Nehmen wir ein einfaches Beispiel einer "Anfänger"-Tabelle. Sowohl die Namen der Personen und deren Adressen sind strukturiert und hier auch noch leicht chaotisch gespeichert. Eine Suche, Sortierung oder Auswahl ist nur schwer oder gar nicht möglich.
Man nennt eine solche Tabelle unnormalisierte Form (UNF) bzw. sagt, sie habe die Non-First-Normal-Form (NF^2).

KID	Name	Wohnort	GebDatum	GebJahr
1	Mustermann, Dieter	12345 Musterhausen, Musterstr. 23	10.03.1980	1980
2	Monika Mustermann	Musterstr. 23; 12345 Musterhausen	09.12.1982	1982
3	Schmidt, Anne	98765, Mustern, Lange Allee 74	28.07.1976	1976
4	Anselm Hinterseher	A-245 Bedorf, Berg-Ring 26	17.04.2001	2001
5	Prof. Kurt Frank	98765 Mustern, Hauptstr. 8	31.05.1984	1984
6	Anne Schmidt	Mustern, 98765, Lange Allee 74	28.07.1976	1976
7	Musterfrau, Maria	88888 St. Glückstadt, Haus Nr. 36	09.09.1991	1991

Zur Klarstellung, diese Tabelle befindet sich in der 0. NF. Sie enthält keine – zumindestens nicht offensichtlich – doppelten Datensätze.
Für ein effektives Arbeiten mit den Daten werden alle strukturierten Attribute atomisiert, also so zerlegt, dass einzelne – für sich gültige – Attribute entstehen.
Das Ergebnis könnte so aussehen:

KID	Name	Vorname	PLZ	Ort	Adresse	GebDatum	GebJahr
1	Mustermann	Dieter	12345	Musterhausen	Musterstr. 23	10.03.1980	1980
2	Mustermann	Monika	12345	Musterhausen	Musterstr. 23	09.12.1982	1982
3	Schmidt	Anne	98765	Mustern	Lange Allee 74	28.07.1976	1976
4	Hinterseher	Anselm	A-245	Bedorf	Berg-Ring 26	17.04.2001	2001
5	Prof. Frank	Kurt	98765	Mustern	Hauptstr. 8	31.05.1984	1984
6	Schmidt	Anne	98765	Mustern	Lange Allee 74	28.07.1976	1976
7	Musterfrau	Maria	88888	St. Glückstadt	Haus Nr. 36	09.09.1991	1991

Alle zusammengesetzten, strukturierten oder mengenwertige Attribute müssen zerlegt werden.
Wahrscheinlich ist bei so wilden Daten (oberste Tabelle) eine händische Bearbeitung notwendig. Deshalb ist es wichtig, sich immer schon im Vorfeld, Gedanken üner Daten-Strukturen zu machen. Einmal verwilderte Daten lassen sich kaum noch 100%ig sicher umwandeln.

1. Normalform (1NF):
Eine Tabelle (Relation) befindet sich in der ersten Normalform, wenn alle ihre Attribute atomar (nicht strukturiert) sind.

Weiterhin sind Wiederholungen oder Wiederholungsgruppen zu eleminieren, d.h. wird z.B. neben dem Geburtsdatum noch das Geburtsjahr extra in der Tabelle verzeichnet, dann ist das Geburtsjahr ein redundanter Sachverhalt. Wahrscheinlich kann man sich dann einfach vom Geburtsjahr trennen oder beim Geburtsdatum nur Tag und Monat speichern.

KID	Name	Vorname	PLZ	Ort	Adresse	GebDatum
1	Mustermann	Dieter	12345	Musterhausen	Musterstr. 23	10.03.1980
2	Mustermann	Monika	12345	Musterhausen	Musterstr. 23	09.12.1982
3	Schmidt	Anne	98765	Mustern	Lange Allee 74	28.07.1976
4	Hinterseher	Anselm	A-245	Bedorf	Berg-Ring 26	17.04.2001
5	Prof. Frank	Kurt	98765	Mustern	Hauptstr. 8	31.05.1984
6	Schmidt	Anne	98765	Mustern	Lange Allee 74	28.07.1976
7	Musterfrau	Maria	88888	St. Glückstadt	Haus Nr. 36	09.09.1991

Man sollte es aber auch nicht übertreiben, die Aufhebung der Strukturisierung beim Straßennamen (in eigentlichen Namen und die Nummer) ist in kleinen Datenbanken eher unnötig. In großen Datenbanken, wo z.B. viele Kontakte in der gleichen Straße (eines Ortes) wohnen, bringt eine weitere Atomisierung dann Speicherplatz-Vorteile.
Durch das Atomisieren und ev. Sortieren usw. fallen dann auch irgendwann doppelte Zeilen, wie z.B. die 3 und die 6 auf. Bereinigt man solche Fehler nicht schon frühzeitig, dann sind später Einträge zu Anne Schmidt nicht mehr eindeutig zuordbar. Die Nacharbeit ist dann sehr / extrem aufwendig.

KID	Name	Vorname	PLZ	Ort	Adresse	GebDatum
1	Mustermann	Dieter	12345	Musterhausen	Musterstr. 23	10.03.1980
2	Mustermann	Monika	12345	Musterhausen	Musterstr. 23	09.12.1982
3	Schmidt	Anne	98765	Mustern	Lange Allee 74	28.07.1976
4	Hinterseher	Anselm	A-245	Bedorf	Berg-Ring 26	17.04.2001
5	Prof. Frank	Kurt	98765	Mustern	Hauptstr. 8	31.05.1984
6	Schmidt	Anne	98765	Mustern	Lange Allee 74	28.07.1976
7	Musterfrau	Maria	88888	St. Glückstadt	Haus Nr. 36	09.09.1991

Erst hier haben wir nun eine Arbeits-Tabelle, die nicht mehr den Anforderungen einer Nullten Normalform entspricht und entsprechend nachbearbeitet werden muss.

KID	Name	Vorname	PLZ	Ort	Adresse	GebDatum
1	Mustermann	Dieter	12345	Musterhausen	Musterstr. 23	10.03.1980
2	Mustermann	Monika	12345	Musterhausen	Musterstr. 23	09.12.1982
3	Schmidt	Anne	98765	Mustern	Lange Allee 74	28.07.1976
4	Hinterseher	Anselm	A-245	Bedorf	Berg-Ring 26	17.04.2001
5	Prof. Frank	Kurt	98765	Mustern	Hauptstr. 8	31.05.1984
7	Musterfrau	Maria	88888	St. Glückstadt	Haus Nr. 36	09.09.1991

Am Ende der 1. Normalisierung ist jedes in der Relation enthaltene Attribut elementar. Außerdem sollte sich die Tabelle auch in der 0. NF befinden. Das ist aber meist durch den Datenbestand selbst schon so vorgegeben.
Bei Tabellen in der 1. NF bestehen immer noch Anfälligkeiten gegenüber DELETE- und INSERT-Anomalien.
Deshalb reicht es in der Praxis nicht aus, mit Tabellen zu arbeiten, die nur in der 1NF vorliegen. Zur weiteren Effektivierung werden weitere Normalisierungen angestrebt.

alternative Formulierungen / Definitionen

1. Normalform (1NF):
Eine Tabelle (Relation) befindet sich in der 1NF, wenn alle ihre Attribute atomisch sind / einen atomaren Werte-Bereich besitzen und sie frei von Wiederholungsgruppen sind.

1. Normalform (1NF):
Ein Relationsschema befindet sich in der 1. Normalforrm, wenn alle ihre Attribute einfach und eindeutig sind.

1. Normalform (1NF):
Jedes Attribut (/ Feld) einer Relation (/ Tabelle) muss einen atomaren Werte-Bereich besitzen und die Relation muss frei von Wiederholungen sein.

Aufgaben:

1. Prüfen Sie ob die nachfolgenden Tabellen in der 1. Normalform sind! Begründen Sie Ihre Entscheidung!

a) CD- / Alben-Datenbank

Nr.	Interpret	Titel	Info's	Quelle
CD28	Sally Oldfield	Milestones	2 CD; 20,00 DM	Castle Communications; 1989
CD93	Cohen, Leonard	Live in London	19,99 €; 2 CD	Sony Music; 2012
CD16	Enya	the celts	CD; 17,99 Euro	1980 Warner Music UK Ltd.
CD85	Apocalyptica	Worlds Collide	CD; 9,99 Euro	Sony Music; 2007
CD87	Apocalyptica	7th Symphony	9,99 Euro; CD	2010; Sony Music
CD33	Yello	essential	CD; 17,99 DM	1992 phonogram
CD36	Within Temptation	the Unforgiving	CD; 15,99 Euro	2011; Sony Music
CD46	Depeche Mode	violator	CD; 16,99 €	2006; Mute Records
CD77	Pink Floyd	The many Face of	3 CD; 22,99 Euro	2013 Music Brokers
CD30	Oldfield, Mike	Tubular Bells II	CD; 19,99 DM	1992 Warner Music UK Ltd.
CD64	City	Am Fenster 2	CD; 13,99 €	2002 BMG Berlin Music
CD58	Within Temptation	Mother Earth	17,99 €; CD	Gun Records; 2003
CD66	Rammstein	Sehnsucht	16,99 DM	1997 Motor Music
CD97	Adele	19	14,99 Euro; CD	XL Recordings Ltd.; 2008
CD84	Northern Lite	Super Black	14,99 €; CD	1stDecade Records; 2008
CD80	Rammstein	Völkerball	22,99 €; CD + DVD	2006 Universal Music
CD27	Rainbirds	Rainbirds	CD; 19,99 DM	phonogram; 1987
CD124	Northern Lite	Reach the Sun	CD; 12,99 Euro	2005; 1stDecade Records
CD95	KISS	Sonic Boom	2 CD + DVD; 19,99 €	2009; KISS catalog Ltd.
CD91	Apocalyptica	Shadowmaker	CD; 16,99 Euro	2015; Odyssey music network
CD52	Yello	essential	CD; 12,99 €	1992 phonogram
CD69	Within Temptation	Black Symphony	2 CD; 21,99 €	Sony Music; 2008
CD92	Mike Oldfield	Tubular Bells II	CD; 9,99 DM	1992 Warner Music UK Ltd.
CD35	Paul Simon	Graceland	CD; 14,99 DM	1986; Warner Bros. Comp.
CD103	Fleetwood Mac	Bare Tres	CD; 9,99 €	1972 ´Warner Bros.
CD72	Jean Michel Jarre	Electronica 2	CD; 14,99 €	2016; Sony Music
CD98	Kalkbrenner, Paul	self	CD; 14,99 Euro	2004 BPITCH CONTROL083
CD29	Jarre, Jean Michel	Oxygene	CD + DVD; 22,99 €	EMI Music France; 2007
CD114	Beyond The Black	lost in forever	CD; 15,99 €	2016 WE LOVE MUSIC
CD118	Deep Purple	infinite	CD + DVD; 16,99 €	2017 EDEL GERMANY

2. Erstellen Sie aus der nachfolgenden Tabelle eine, die in der 1NF vorliegt! Erläutern Sie Ihr Vorgehen!

Speicher-Gerät	Verfügbarkeit	Herkunft	Wert
USB-Stick; USB 3.0; schwarz; 8 GB	13 Stk.; 9 €	Deutschland; Huawei	117
silberne Festplatte; 2 TB; USB 2.0	89 Euro; 3 Stk.	Taiwan; GoldTech	267
Festplatte; schwarz; 1 TB; USB 2.0	79 Euro; 4 Stk.	Japan, EPSON	316
USB-Stick; silber; USB 2.0; 16 GB	10 Pkg. a 10 Stk.; 8 €	hp; USA	800
USB 3.0 externe HDD 500 GB	49 €; 2 Pkg. a 2 Stk.	Seagate; USA	196
Laser-Drucker; farbig; 128 MB; USB 3.0	1 Gerät; 369 Euro	Kyocera; Japan	369
Festplatte; schwarz; 1,5 TB; USB 2.0	119 Euro; 6 Stk.	Japan, EPSON	714
USB-Stick; grün; USB 2.0; 16 GB	8 Pkg. a 10 Stk.; 8 €	hp; USA	640
USB 3.0 Stick; rot; 32 GB	26 Stk.; 18 Euro	Taiwan; GoldTech	468
Festplatte; 2 TB; USB 2.0; blau	99 Euro; 9 Stk.	hama; Deutschland	891
USB 3.0 externe HDD 500 GB	49 €; 2 Pkg. a 3 Stk.	Seagate; USA	294

3. Erstellen Sie aus 10 Objekten Ihres Lebensbereiches / Umfeld's – die aus einer (Objekt-)Klasse stammen – eine Daten-Tabelle mit mindestens 5 sachlich korrekten Attributen in der 1. Normalform!

2.2.2.3. Zweite Normalform

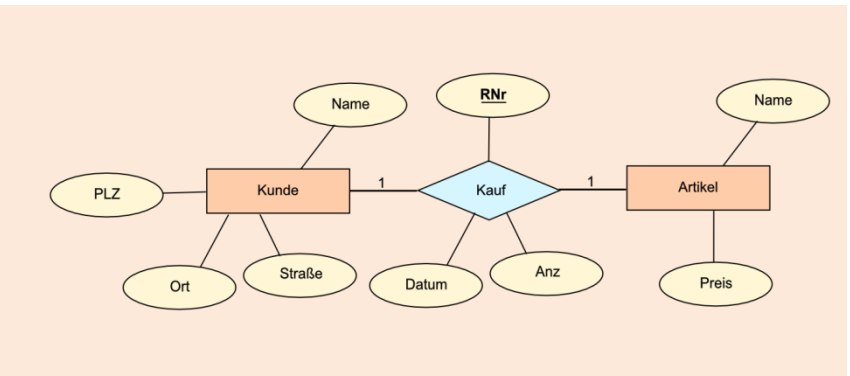
Gruppierung von Tabellen-Daten nach Sachverhalten / Themen / Inhalten zu neuen (monothematischen) Tabellen
Z.B. könnten in einer Datenbank die Lieferanten in einer extra Tabelle bzw. in einem komplexeren Tabellen-Konstrukt (aus mehreren Tabellen) ausgelagert werden. Zuerst enthält die Tabelle vielleicht nur die elementarsten Daten, die schon immer vorhanden waren, wie Adresse und Telefon-Nummer. Nun will man aber irgandwann seine Datenbank um weitere Informationen erweitern, z.B. um eine spezielle Ansprech-Person. Mit solchen (Personen-)Netzwerken lassen sich vertrauensvolle, stabilere und unkomplizierte Geschaäfts-Beziehungen aufbauen, die i.A. zum gegenseitigen Vorteil sind. Die Daten müssen dann in unsere Datenbank mit eingebaut werden. Gibt es schon Tabelle zu den Lieferanten, dann ist eine passende Ergänzung / Erweiterung leicht möglich.

Beispiel:
klassische "EXCEL-Datenbank" einer Firma mit einer Tabelle "Rechnungen"

Trennung von Kunden-Daten und Verkaufs-Daten

RNr	Datum	Kunde	PLZ	Ort	Straße	Artikel	Anz.	Preis
223	13.10.17	Frank Müller	12345	Mustern	Musterstr. 12	Box 21	5	49,95 €
224	16.10.17	Perl AG	23456	Bdorf	Hauptstr. 9	Box 46	20	69,95 €
225	16.10.17	Garnev GmbH	34567	Hellzig	Dorf-Allee 45	Box 38	14	55,95 €
226	16.10.17	Clara Schmidt	12345	Mustern	Bergweg 12	Box 21	1	49,95 €
227	17.10.17	Perl AG	23456	Bdorf	Hauptstr. 9	Box 33	10	59,95 €
228	18.10.17	Perl AG	23456	Bdorf	Hauptstr. 9	Box 46	20	69,95 €
229	18.10.17	BOX GmbH	98765	Testern	Marktplatz 14	Box 38	9	55,95 €

Das aus der Roh-Tabelle herauskristallisierte ER-Roh-Diagramm mit den Zuordnungen der Tabellen-Spalten zu den Objekten und Beziehungen könnte dann so aussehen.



Diese Struktur setzen wir auch erst einmal um, und denken auch gleich an passende Schlüssel.
Praktisch heißt das z.B. für die Kunden-Daten, dass wir "Name", "PLZ", "Ort" und "Straße" in eine neue Tabelle übernehmen.
Jeder Kunde bekommt eine kurze Kunden-Nummer (KID), die eine eindeutige Zuordnung ermöglicht.
Sicher hätte man auch den Namen des Kunden als Schlüssel benutzen können.

KID	Name	PLZ	Ort	Straße
1	Frank Müller	12345	Mustern	Musterstr. 12
2	Perl AG	23456	Bdorf	Hauptstr. 9
3	Garnev GmbH	34567	Hellzig	Dorf-Allee 45
4	Clara Schmidt	12345	Mustern	Bergweg 12
5	BOX GmbH	98765	Testern	Marktplatz 14

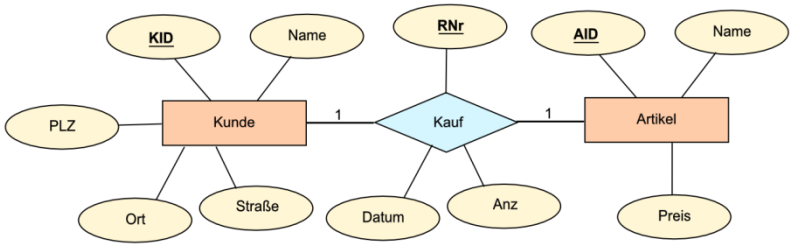
In unserem Daten-Bestand gibt es keine Doppelungen, was aber bei Namen, wie "Frank Müller" aber schnell man passieren kann.
Die Daten zu den Kunden werden nun in einer separten Tabelle geführt.

Es folgt das gleiche Vorgehen auch für die Artikel. Somit erhalten wir eine einfache, nicht-dedundante – leicht zu pflegende – Artikel-Tabelle. Als Schlüssel hätten wir auch die Bezeichnung nutzen können. In Computer-Systemen ist es effektiver reine Zaheln oder kurze Text zu nutzen. Damit kann ein Computer deutlich schneller arbeiten.

AID	Bezeichnung	Preis
B21	Box 21	49,95 €
B33	Box 33	59,95 €
B38	Box 38	55,95 €
B46	Box 46	69,95 €

Damit haben wir zu den Objekten monothematische Tabellen, die sich für sich in der 1. Normalform befinden.

Durch Ergänzen der Schlüssel in das ER-Diagramm erhalten wir ein's, dass für die Haltung der Daten in der 2. Normalform geeignet ist.



Die Beziehung zwischen den Tabellen "Kunden" und Artikel stellt die Tabelle "Kauf" dar.

Sie nethält neben den spezifischen Attributen "Datum" und "Anz" die Rechnungsnummer als Schlüssel.

RNr	Datum	KID	AID	Anz
223	13.10.17	1	B21	5
224	16.10.17	2	B46	20
225	16.10.17	3	B38	14
226	16.10.17	4	B21	1
227	17.10.17	2	B33	10
228	18.10.17	2	B46	20
229	18.10.17	5	B38	9

Für die zugehörigen Kunden und Artikel werden nur noch Verweise auf die entsprechenden Tabellen eingerichtet. Somit ist auch dieser Teil der ursprünglichen Tabelle optimal dargestellt.

Für jedes sachlich separate Daten-Element gibt es jetzt nur noch einen Stelle im neuen Tabellen-Konstrukt.

2. Normalform (2NF):
Eine Tabelle (Relation) befindet sich in der zweiten Normalform, wenn sie in der 1. Normalform ist und alle ihre Nicht-Schlüssel-Attribute funktional abhängig von der gesamten Schlüssel-Kombination, aber nicht von ihren Teilen.

mit der 2NF werden monothematische Relationen erzeugt, d.h. eine Relation stellt (nur) einen Teil des Gesamtsachverhaltes dar

jedes Attribut ist entweder von einem Schlüssel abhängig oder selbst ein Schlüssel
Vermeidung von transitiven Abhängigkeiten (Abhängigkeit eines Attributes von einem anderen Attribut, welches wiederum von einem anderen (dritten) Attribut abhängig ist)

es besteht noch Anfälligkeit gegenüber DELETE-Anomalien

2NF-Datenbank:
Eine Datenbank befindet sich in der zweiten Normalform, wenn alle ihre Tabellen (Relationen) in der 2. Normalform sind.

Algorithmus zur Überführung einer Relation in die 2NF

0. BEDINGUNG: Relation liegt in 1NF vor
1. Festlegen des Primärschlüssels der (gegebenen) Relation
2. WENN der Primärschlüssel nur aus einem Attribut besteht, DANN weiter bei 8
3. WENN aus Teilschlüssel-Attributen mehrere Attribute folgen, DANN weiter bei 4, SONST weiter bei 8
4. Erstellen einer neuen Relation, die das Teilschlüssel-Attribut und alle von diesem abhängigen Nicht-Schlüssel-Attribute enthält
5. Festlegen des Teilschlüssel-Attributs als Primärschlüssel (der neuen Relation)
6. Löschen der ausgelagerten Nichtschlüssel-Attribute aus der vorgegebenen Relation
7. WIEDERHOLE SOLANGE ab 3 BIS alle Nichtschlüssel-Attribute funktional abhängig sind
8. 2NF erreicht (STOPP)

Algorithmus zur Überführung einer Datenbank in die 2NF

0. BEDINGUNG: Datenbank liegt in 1NF vor
1. WIEDERHOLE FÜR alle Relationen:
   1a. Überführe Relation in 2NF

alternative Formulierungen zur Charakterisierung der 2. Normalform

2. Normalform (2NF):
Eine Relation (/ Tabelle) ist in der 2NF, wenn sie in der 1NF vorliegt und kein Nicht-Schlüssel-Attribut funktional abhängig von einer echten Teilmenge eines Schlüssel-Kandidaten ist.

2. Normalform (2NF):
Ein Relations-Schema ist in der 2. Normalform, wenn sie in der 1NF vorliegt und jedes nicht zum Identifikations-Schlüssel gehörende Attribut voll funktional von diesem abhängig ist.

2. Normalform (2NF):
Eine Relation (/ Tabelle) befindet sich in der 2. Normalform, wenn die Relation ausschließlich atomare Daten enthält und mindestens ein einstelliger Kandidat für die Verwendung als Primärschlüssel existiert.

2.2.2.4. Dritte Normalform

3. Normalform (3NF):
Eine Tabelle (Relation) befindet sich in der dritten Normalform, wenn sie in der zweiten Normalform ist und kein Nicht-Schlüssel-Attribut von einem Schlüssel-Attribut abhängig ist.

Was meint die Regel?
Beim genauen Betrachten der Daten in der Tabelle finden wir diverse Redundanzen:

KID	Name	Vorname	PLZ	Ort	Adresse	Geburtsdatum
1	Mustermann	Dieter	12345	Musterhausen	Musterstr. 23	10.03.1980
2	Mustermann	Monika	12345	Musterhausen	Musterstr. 23	09.12.1982
3	Schmidt	Anne	98765	Mustern	Lange Allee 74	28.07.1976
4	Hinterseher	Anselm	A-2345	Bedorf	Bergsteiger-Ring 26	17.04.1991
5	Prof. Frank	Kurt	98765	Mustern	Hauptstr. 8	31.05.1984
6	Schmidt	Anne	98765	Mustern	Lange Allee 74	28.07.1976
7	Musterfrau	Maria	88888	St. Glückstadt	Haus Nr. 36	09.09.1991

Einige davon – z.B. die Postleitzahl verknüpft mit dem Ort tauchen sehr häufig auf.

nach dem Herstellen der 3NF sind die Relationen zuverlässig monothematisch

Bei aller Reduktion darf man aber den gesunden Menschen-Verstand nicht ausschalten. Selbst, wenn eine aktuelle Tabelle noch eine weitere / tiefere Normalisierung zulassen würde, sollte man immer die Möglichkeiten / Varianten zukünftiger Einträge bedenken. So können sich auch mal Straßen-Name ändern oder deren Zuordnungen zu Postleitzahlen oder Ort(steil)en. Eine starke Strukturierung mit vielen Referenzen bedeutet auch immer wieder erneutes Nachschlagen in einer anderen Tabelle. Das kostet Zeit und Rechen-Leistung. Ob das den ev. eingesparten Speicherplatz wett macht, muss ev. auch mal geprüft werden. Erfahrene Datenbank-Planer haben da ihre Erfahrungen, was sich für den einen Anwendungsfall anbietet, oder was man lieber unter bestimmten Bedingungen läßt.

Algorithmus zur Überführung in die 3NF

0. BEDINGUNG: gegebene Relation liegt in 2NF vor
1. WENN aus einem (untersuchtem) Nichtschlüssel-Attribut ein oder mehrere andere Nichtschlüssel-Attribute folgen, DANN weiter bei 2, SONST weiter bei 6
2. Erstellen einer neuen Relation aus dem (untersuchtem) Nichtschlüssel-Attribut und den anderen – von diesem – abhängigen Nichtschlüssel-Attribute
3. Festlegen des (untersuchten) Nichtschlüssel-Attributes als Primärschlüssel (der neuen Relation)
4. Löschen der ausgelagerten, abhängigen Nichtschlüssel-Attribute aus der gegebenen Relation (untersuchtes Nichtschlüssel-Attribut verbleibt als Fremdschlüssel in der Relation)
5. WIEDERHOLE SOLANGE ab 1 BIS keine Abhängigkeiten in der gegebenen Relation bestehen
6. 3NF erreicht (STOPP)

alternative Formulierungen zur Charakterisierung der 3. Normalform

3. Normalform (3NF):
Ein Relationsschema ist in der 3NF, wenn es sich in der 2NF befindet und kein Nicht-Schlüssel-Attribut von einem Schlüssel-Kandidaten transitiv abhängt.

3. Normalform (3NF):
Ein Relationsschema ist in der 3. Normalform, wenn es sich in der 2NF befindet und kein Attribut, das nicht zum Identifikations-Schlüssel gehört, von diesem transitiv abhängig ist.

3. Normalform (3NF):
Ein Relationsschema ist in der 3NF, wenn es sich in der 2NF befindet und kein Nicht-Schlüssel-Attribut Determinate ist.

3. Normalform (3NF):
Ein Relationsschema ist in der 3NF, wenn es sich in der 2NF befindet und kein Nicht-Schlüssel-Attribut von einem anderen Nicht-Schlüssel-Attribut abhängig ist.

Aufgaben:

1. Gegeben ist die folgende Tabelle, die sich in der 1. Normalform befindet. Normalisieren Sie diese Relation bis zur 3NF!

KursID	SchID	Tag	LehID	Fach	Raum	Note	LBuch	LehEMail
D2	759	Die	MEI	Deu	R 3.20	1	Duden DeuAbi	Hr.Meier@dbs.loc
M3	759	Don	ZAN	Ma	R 2.05	3	PAETEC Ma3	Hr. Zander@dbs.loc
B1	382	Mon	SCH	Bio	R 1.04	3	Schroedel SII	Fr.Schulz@dbs.loc
C2	382	Die	MEI	Chem	R 2.11	2	Winkler C3	Hr.Meier@dbs.loc
B1	665	Mon	SCH	Bio	R 1.04	2	Schroedel SII	Fr.Schulz@dbs.loc
D1	277	Mit	MEI	Deu	R 1.04	4	Duden DeuAbi	Hr.Meier@dbs.loc

x. Prüfen Sie Ihr Normalisierungs-Können unter der folgenden URL!
http://edb.gm.fh-koeln.de/nf/start.jsp?action=wl

für die gehobene Anspruchsebene:

y. Normalisieren Sie die folgende Relation bis zur 3NF! Fehlende Daten dürfen Sie logisch passend ersetzen.

TID	Tier	Ort						
34	Elephant "Dumbo"; männlich							
56	Tiger "Indi"; männlich							
22	Löwe "Lois"							
23	"Martha"; Löwin							
50	weiblich; Tiger "Ursa"							
72	Elephant "Elphi"; weiblich							
94	Nashorn "Bernd"; männlich							

Beispiel:

Buch-Ausleihe

Schüler *Lehrer* *Buch*

SNr	SName	SVorname	LNr	LName	BNr	Fach	Preis
527	Müller	Anne	85	Zander	7917	Bio	30
527	Müller	Anne	85	Zander	7356	Deu	20

→ 1NF

→ 2NF

Buch-Ausleihe

BAID	SNr	SName	SVorname	LNr	LName	BNr	Fach	Preis
1	527	Müller	Anne	85	Zander	7917	Bio	30
2	527	Müller	Anne	85	Zander	7356	Deu	20

→ 3NF

Buch-Ausleihe

BAID	SNr	LNr	BNr	Fach	Preis
1	527	85	7917	Bio	30
2	527	85	7356	Deu	20

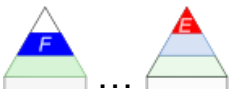
Schüler

SNr	SName	SVorname
527	Müller	Anne

Lehrer

LNr	LNr	LName
85	527	Zander

2.2.2.5. weitere Normalformen



2.2.2.5.1. BOYCE-CODD-Normalform (BCNF)

Superschlüssel (Oberschlüssel)
Menge von Attributen einer Relation, welche die Tupel dieser Relation eindeutig indentifizieren
ein trivialer Superschlüssel ist z.B. die Menge aller Attribute einer Relation gemeinsam → es darf deshalb in einer Realtion keine zwei gleichen Tupel (/ Datensätze) geben

BOYCE-CODD-Normalform (BCNF):
Ein Relationsschema ist in der BCNF, wenn es sich in der 2NF befindet und jede Determinante ein Superschlüssel ist.

durch die BCNF wird verhindert, dass die Bestandteile zweier aus mehreren Attributen zusammengesetzten Schlüsselkandidaten voneinander abhängig sind

Überführung eines Relationsschemas in die BCNF ist immer Verlust-frei aber nicht immer Abhängigkeits-erhaltend

ursprünglich war die BCNF als Verschärfung der 3NF gedacht
eine BCNF kann als Erweiterung einer 3NF verstanden werden

mit Zerlegungs-Algorithmus ist eine Relation in die BCNF überführbar

2.2.2.5.2. die 4. Normalform

4. Normalform (4NF):
Ein Relationsschema ist in der 4NF, wenn es sich in der BCNF befindet und nur noch nicht-triviale mehrwertige Abhängigkeiten enthält.

nicht-triviale Abhängigkeiten (MWA, mwA)

in einer Relation darf es nicht mehrere 1:n- oder n:m-Beziehungen zu einem Schlüssel-wert geben, die thematisch (/ inhaltlich) nichts miteinander zu tun haben

Im nebenstehenden Beispiel haben die Hör-Gewohnheiten nichts mit den gewünschten Auto's zu tun.

durch die 4NF wird erreicht, dass n-äre Beziehungen (- also mehr als 2 Tabellen stehen in Beziehung -) korrekt modelliert sind

Vorlieben

PersID	Lieblingsmusik	Traumauto
45	Hip-Hop	Golf
45	House	Porsche
71	Hip-Hop	Mercedes
88	Schlager	Ferrari
88	Volksmusik	Ferrari
88	Country	Ferrari
94	House	Maserati

hört

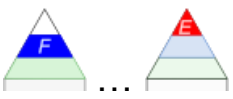
PersID	Lieblingsmusik
45	Hip-Hop
45	House
71	Hip-Hop
88	Schlager
88	Volksmusik
88	Country
94	House

mag

PersID	Traumauto
45	Golf
45	Porsche
71	Mercedes
88	Ferrari
94	Maserati

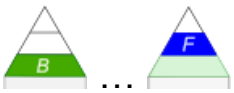
Die Vermeidung der Mehrfach-Einträge bei den Attributen (hier bei "hört") wird dann in der Normalisierung 5. Ordnung angestrebt. ImBeispiel ist auch bei "mag" eine Mehrfach-Nennung zu erwarten, so dass die Tabelle ebenfalls weiter normalisiert werden sollte.

2.2.2.5.3. die 5. Normalform



5. Normalform (5NF):
Ein Relationsschema ist in der 5NF, wenn es sich in der 4NF befindet und keine mehrdeutigen Abhängigkeiten enthält, welche voneinander abhängen.

2.2.2.6. Algorithmen der Normalisierung



(klassische) Algorithmen für die Normalisierung

- **Synthese-Algorithmus** → 3NF (3. Normalform)
- **Zerlegungs-Algorithmus** → BCNF (BOYCE-CODD-Normalform)
-

Algorithmus zum Herstellen der 2. Normalform

→ 2.x.1.3. Zweite Normalform

Algorithmus zum Herstellen der 3. Normalform

→ 2.x.1.4. Dritte Normalform

2.2.2.7. Zusammenfassung, …

The key, the whole key, and nothing but the key. So help me CODD!
(Der Schlüssel, der ganze Schlüssel und nichts als der Schlüssel. So wahr mir CODD helfe!)
angelehnt an den amerikanischen Gerichts-Eid

alle (impliziert:atomaren) Werte beziehen sich auf den Schlüssel der Relation → 1NF
bei zusammengesetzten Schlüsseln beziehen sie (?) sich jeweils auf den gesamten Schlüssel → 2NF
die Werte hängen nur vom Schlüssel (und nicht von den Nicht-Schlüssel-Attributen) ab → 3NF

Test-Regeln:
1. WENN eine Relation in der 1NF vorliegt UND der Primärschlüssel nur aus einem Attribut besteht, DANN liegt die 2NF vor.
2. WENN eine Relation in der 2NF vorliegt UND sie außer dem Primärschlüssel höchstens noch ein weiteres Atrribut besitzt, DANN liegt sie in der 3NF vor.
3. WENN die 1. UND 2. Test-Regel erfüllt ist, DANN ist die 3NF die letzte.

WENN Attribute von einem Teil des Schlüssels eindeutig identifiziert werden, DANN liegt keine 2NF vor.
WENN in einer Relation in der 2NF aus einem Nichtschlüssel-Attribut ein anderes Nichtschlüssel-Attribut folgt, DANN liegt keine 3NF vor.

Definition(en): relationale Datenbank

Eine relationale Datenbank ist eine Datensammlung, die aus normalisierten Relationen (Tabellen) sowie Beziehungen zwischen diesen besteht.
Im Sinne eines relationalen Datenbank-Systems (DBS) miuss auch ein geeignetes Verwaltungssystem enthalten sein.

Reisekosten

Rechnungs-Nummer	Reise-Datum	Kunden-Name	Kunden-Vorname	PLZ	Ort	Straße	Kostenart	Anzahl	Einzel-Vergütung

Überführung in die 2. Normalform

- Auslagern von feststehenden Rechnungs- und Kosten-Parametern

Reise

Rechnungs-Nummer	Reise-Datum	Kunden-Name	Kunden-Vorname	PLZ	Ort	Straße

Position

Rechnungs-Nummer	Kostenart	Anzahl

Kostenarten

Kostenart	Einzel-Vergütung

Überführung in die 3. Normalform

- Auslagern Personal-Daten und Ortsangaben

Reise

Rechnungs-Nummer	Reise-Datum	Personal-Nummer

Personal

Personal-Nummer	Name	Vorname	PLZ	Straße

Position

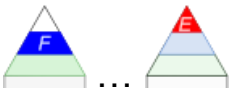
Rechnungs-Nummer	Kostenart	Anzahl

Kostenarten

Kostenart	Einzel-Vergütung

Orte

PLZ	Ort



Besitz (Nicht-4NF)

Name	Haustier	Auto
Musterfrau	Katze	Audi
Mustermann	Hund	Audi
Mustermann	Katze	BMW
Schneider	Hund	Porsche

Überführung in die 4. Normalform

- Aufteilung in unabhängige Tabellen

Tierbesitz

Name	Haustier
Musterfrau	Katze
Mustermann	Hund
Mustermann	Katze
Schneider	Hund

Autobesitz

Name	Auto
Musterfrau	Audi
Mustermann	Audi
Mustermann	BMW
Schneider	Porsche

Kursbelegung (Nicht-5NF)

Fach	Klasse	Schüler
Biologie	11	Meier
Biologie	11	Schmidt
Biologie	12	Bauer
Biologie	12	Müller
Chemie	11	Friedrich
Chemie	11	Müller
Chemie	11	Zander
Physik	11	Friedrich
Physik	11	Zander
Physik	12	Bauer
Physik	12	Müller

Überführung in die 5. Normalform

- Aufteilung in 3 triviale Tabellen

Kurse

Fach	Klasse
Biologie	11
Biologie	12
Chemie	11
Physik	11
Physik	12

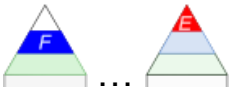
Fachbelegung

Fach	Schüler
Biologie	Bauer
Biologie	Meier
Biologie	Müller
Biologie	Schmidt
Chemie	Friedrich
Chemie	Müller
Chemie	Zander
Physik	Bauer
Physik	Friedrich
Physik	Müller
Physik	Zander

Klassenzugehörigkeit

Klasse	Schüler
11	Meier
11	Schmidt
12	Bauer
12	Müller
11	Friedrich
11	Müller
11	Zander
11	Friedrich
11	Zander
12	Bauer
12	Müller

2.2.2.8. mathematisch orientierte Darstellung des relationalen Datenbank-Modell



Realtionsschema:
Schüler(SID, Name, Vorname, Geschlecht, Geburtsdatum, Geburtsort, …)

allg: R1(

Relation r ist endliche Menge $\boldsymbol{r} \subseteq \text{Tup}(\boldsymbol{X})$
Menge aller Relationen über X: $\text{Rel}(\boldsymbol{X})$
Instanz von X: $\boldsymbol{r} \in \text{Rel}(\boldsymbol{X})$
Relationsbezeichner: ***R***
Relationsschema: R(X) doer als Attributmenge: $\boldsymbol{R(\{A_1, A_2, \ldots, A_k\})}$
Stelligkeit der Relationsbezeichner: ***k***
$\boldsymbol{R(A_1: dom(A_1), A_2: dom(A_2), \ldots A_k: dom(A_k))}$

Relationsschema: ***R(X)***

Schlüssel K ist $\boldsymbol{K \subseteq X}$

Relationsschema $\mathbf{R}$: $\mathbf{R} = \{R_1(X_1), R_2(X_2), \ldots, R_m(X_m)\}$ bzw. $\mathbf{R} = (R_1, R_2, \ldots, R_m)$

Instanz $\mathbf{I}$: $\mathbf{I}(\boldsymbol{R_i}) = \boldsymbol{r_i}, 1 \leq \boldsymbol{i} \leq \boldsymbol{m}$

Aufgaben:

1. ***In einer Datenbank sollen die Personen eines Landes erfasst werden. Zuerst sollen nur der Nachname und die Vornamen erfasst und verarbeitet werden. Es stehen drei Vorschläge im Raum (dunkle Spalten sollen die Primärschlüssel darstellen):***

Vorschlag: 1

PID	Nachname	Vorname(n)
1	A…	A…
…		
5162	Bauer	Monika
5163	Bauer	Frank Dieter
…		
…		
…		
…	Z…	Z…

Vorschlag: 2

Nachname	Vorname(n)
A…	A…
…	…
Bauer	Monika
Bauer	Frank Dieter
…	…
…	…
…	…
Z…	Z…

Vorschlag: 3

NID	Nachname
1	A…
…	
29	Bauer
…	
…	
…	
…	
…	Z…

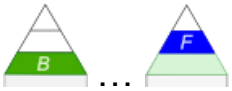
VID	Vorname(n)
1	A…
…	…
73	Frank Dieter
	…
364	Monika
	…
	…
…	Z…

PID	NID	VID
1		
…		
5162	29	364
5163	29	73
…		
…		
…		
…		

a) Beurteilen Sie die verschiedenen Vorschläge hinsichtlich benötigtem Speicherplatz und der zu erwartenden Geschwindigkeit von Such-Anfragen nach einer Person!

b) Erstellen Sie einen weiteren Vorschlag! Begründen Sie diesen und bewerten Sie ebenfalls Speicherbedarf und Such-Geschwindigkeit!

2.3. andere Daten(bank)-Modelle



häufig auch als **NoSQL-Datenbanken** bezeichnet
NoSQL wird im Datenbanker Slang auch *no'sig'wel* gesprochen
moderne, über das übliche hinausgehende, Funktions-Umfänge für relationale und andersartige Datenbanken
gemeint ist nicht "Kein-SQL" sonder not only SQL
NoSQL-Datenbanken bieten mehr als nur SQL (wobei SQL nicht die primäre Schnittstelle ist)SQL-Schnittstelle wird meist am Schluß auch mit angeboten, damit die Kompatibilität zur SQL-Datenbanken erhalten bleibt (oft auch nur Teilmengen von SQL implementiert)
Betonung liegt auf neuen Konzepten

Motivation für die Schaffung andersartiger Datenbanken
(bezüglich relationaler Datenbanken)

- schlecht breit skalierbar (von sehr klein (mini) bis sehr groß (fat) (es gibt realiserunge für eher kleine oder eben für eher große oder sehr große Datenbanken)
- schlecht verteilbar (bedarf / braucht (ev. auch nur sehr kurze) Stillstands-Zeit)
- lineare Beziehung von Daten-Menge und notwendige Speicher-Hardware
-

Beispiele für NoSQL-Datenbank-Modelle

- Graph-Datenbank-Modell
- Mehrwert-Modell
- Dokumenten-orientiertes Modell
- Datei-Systeme (in Betriebssystemen)
-

konkrete Anwendungen (Beispiel für NoSQL-Datenbank-Systeme)

- cassandra
- riak
- CouchDB
- Apache HBASE
- redis
- mongoDB
- memSQL
- VoltDB
- HyPer

Vorteile

- breit skalierbar
- gut (weit) verteilbar
- einzelne Datenbank-Eigenschaften werden optimal realisert
-

Nachteile

- stellen einzelne Datenbank-Eigenschaften / -Merkmale / -Anforderungen in den Vordergrund unter Inkaufnahme von Schwächen bei anderen Eigenschaften
-

NewSQL-Datenbanken

wollen SQL vollständig anbieten und dazu bestimmte Datenbank-Eigenschaften gegenüber relationalen SQL-Datenbanken anbieten (z.B. Garantie von Konsistenzen)

Vorteile

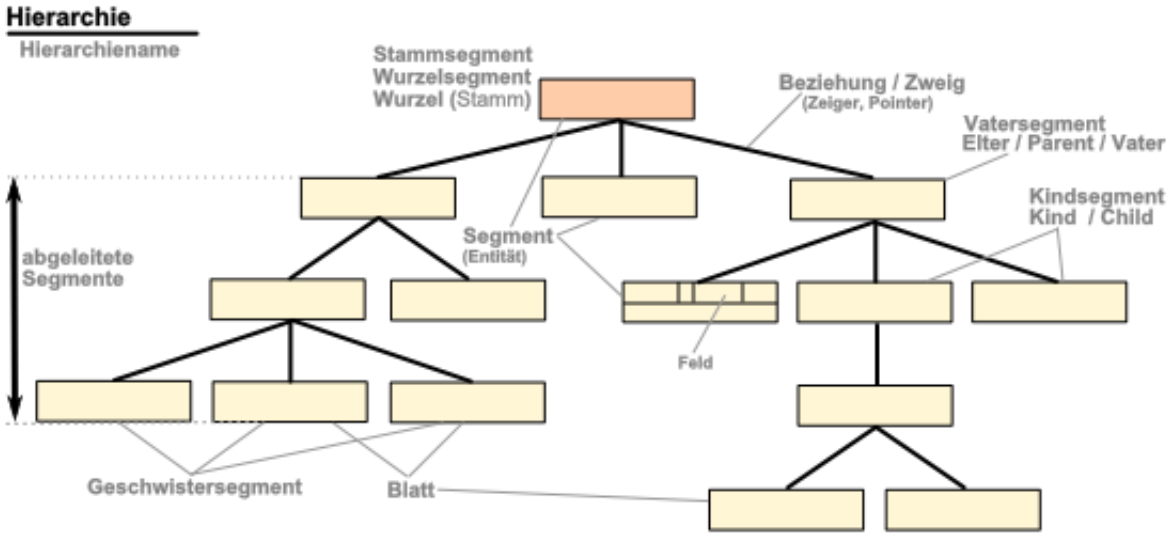
- lassen sich mit gut bekannter Sprache (SQL) benutzen
- praktisch vollständiger SQL-Sprach-Umfang
- einzelne Datenbank-Eigenschaften werden optimal realisert
-

Nachteile

-

2.3.1. hierarchisches Datenbank-Modell

ältestes Datenbank-Modell
Basis sind Über-, Unter- und Neben-Ordnungen
Datenstruktur ist Baum-artig oder eigentlich eher Wurzel-artig



Eltern-Kind-Beziehungen
Parent-Child-Relationship (PCR)

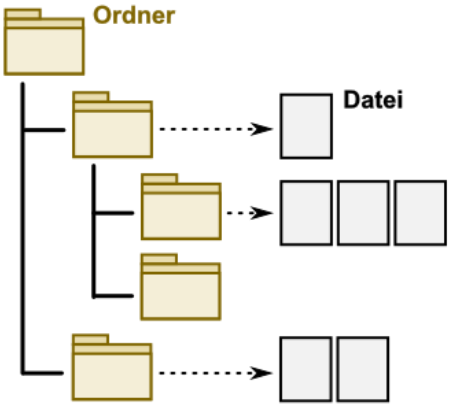
jeder Datensatz hat einen Vorgänger, mit Ausnahme des Wurzel-Datensatzes
der Wurzel-Datensatz ist somit niemals Kind / Child
es gibt keinen, einen oder mehrere Nachfolger
ein Datensatz der nicht als Elter auftritt wird Blatt genannt
verwaiste (abgetrennte Kindsegmente sind nicht zulässig, hier muss das Anwender-Programm die Konsistenz der Datenbank prüfen, was ungünstig ist (Programmierfehler)

Vorteile:

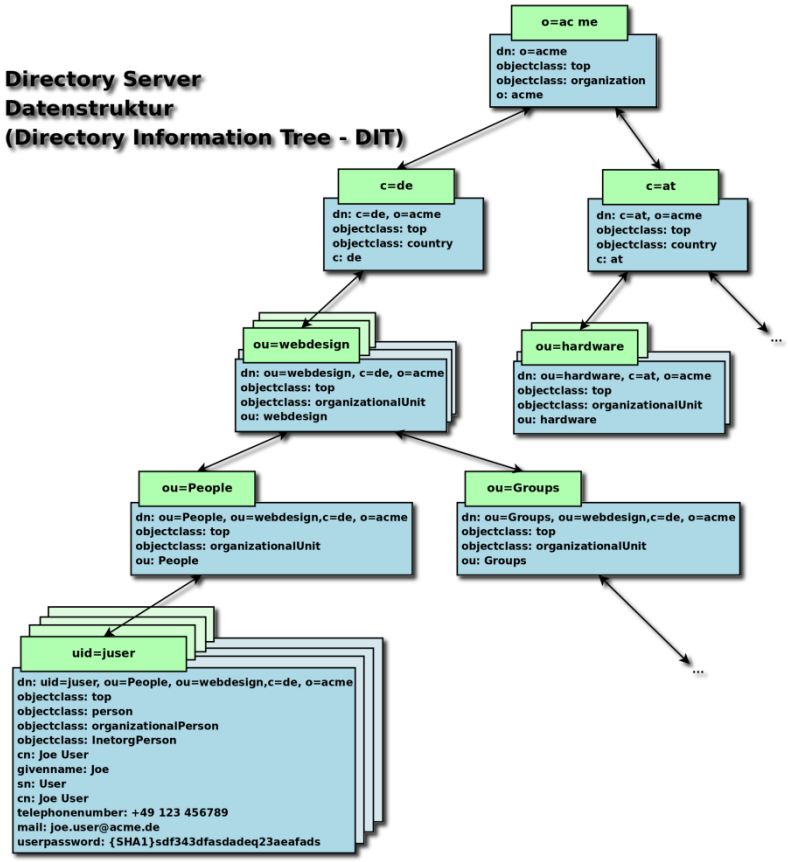
Nachteile:

keine Beziehungen zwischen verschiedenen Bäumen
keine Beziehungen über die PCR hinaus (also keine Großeltern-Enkel-Beziehungen; nur indirekt vorhanden)
es lassen sich nur 1 :1- und 1 : n-Beziehungen darstellen
n : m-Beziehungen sind immer mit Redunzen verbunden

Beispiel Datei-System in microsoft DOS bzw. WINDOWS-Betriebssystemen



typischerweise sind z.B. die Dateisysteme der meisten Betriebssysteme hierarchische Datenbanken Rechte-Strukturen (Active Directory auf Windows-Rechnern / -Systemen / -Servern; LDAP (Verzeichnisdienste))



Beispiel für Verzeichnis-Dienste / -Strukturen
Q: de.wikipedia.org (Pgergen, Denny-pug)

heute mit XML-Dateien wieder aktuell

XML (**Ex**tended Markup Language

praktisch sehr universelle Erweiterung von HTML
zuerst sollten vor allem Texte formatiert und anotiert werden

dann kam später die textuelle Speicherung von Daten dazu
enthalten Inhalts-bezogene Tag's

als reine Speicher-Struktur wefgen des großen Overhead's nicht interessant, aber als universelles Austausch-Format für Daten (z.B. aus Datenbanken)

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!-- Dateiname: SchulDaten.XML -->

<schule>
   <Name> Zweistein-Schule Odorf </Name>
   <Lehrer>
      <Name> Arendt </Name>
      <Name> Bauer </Name>
…
      <Name> Zander </Name>
   </Lehrer>
   <Klasse>
      <KlLehrer> Bauer </KlLehrer>
      <StellvKlLehrer />
      <Schueler>
         <Name> Baier </Name>
         <weiblich> 1 </weiblich>
      </Schueler>
      <Schueler>
         <Name> Friedermann </Name>
         <weiblich> 0 </weiblich>
      </Schueler>
      <Schueler>
…
      </Schueler>
   </Klasse>
</schule>
```

Prinzip-Beispiel einer XML-Datei

Die Elemente können weiter strukturiert werden, jenachdem, wieweit man Informationen braucht.
beleibig viele Ebenen möglich
i.A. wird empfohlen die Schachtelung nur bis Ebene 10 bis 15 vorzunehmen, ansonsten verlieren sich die Vorteile der Strukturierung und Lesbarkeit / verständlichkeit für menschen (zumiondestens bei Importen usw. muss dieser die Datenstruktur ja auch erkennen / verstehen / zuordnen

Will man z.B. bei den Lehrern auch das Geschlecht speichern, dann muss hier eine Umstrukturierung des Tag's Lehrer erfolgen.

```
<?xml version="1.0" encoding="ISO-8859-1" ?><!--
Dateiname: SchulDaten.XML --><schule>
<Name> Zweistein-Schule Odorf </Name><Lehrer> <Na-
me> Arendt </Name><Name> Bauer </Name> … <Name>
Zander </Name></Lehrer><Klasse> <KlLehrer> Bauer
</KlLehrer> <StellvKlLehrer /><Schueler><Name>
Baier </Name><weiblich> 1
</weiblich></Schueler><Schueler><Name> Friedermann
</Name><weiblich> 0 </weiblich>
</Schueler><Schueler> … </Schueler></Klasse>
</schule>
```

Prinzip-Beispiel einer XML-Datei
ohne Betrachter-Strukturierung

Vorteile:

selbstbeschreibendes Daten-Format (Attribute und Werte erkennbar)
Maschinen- und von Menschen lesbar
Betriebssystem- und Anwendungs-übergreifend nutzbar (→ Austausch-Format)
Daten haben eine Ordnung

Speicherung gemischter Daten (bis hin zu Links auf andere Dateien usw.) möglich

Vorteile:

großer Overhead durch die ständige Wiederholung der Tag's sowie durch schließende Tag's

Aufgaben:

1. *Überlegen Sie sich, wie eine XML-Struktur aussehen könnte, bei der die Lehrer auch ein Geschlecht bekommen! Welche Tag's würden Sie benutzen? Bereiten Sie Ihr Modell zur kurzen Vorstellung als Diskussions-Grundlage vor!*
2. *Stellen Sie Ihre XML-Struktur / -Tag's vor und diskutieren Sie die verschiedenen vorgestellten Modelle!*
3. *Erstellen Sie eine hierarchische Struktur zu einem Objekt z.B. ein Mensch, ein PC, ein Auto usw. usf. mit seinen Bestandteilen, Unterbestandteilen usw. usf.! (Erwartet werden mindestens drei Hierrarchie-Ebenen!)*

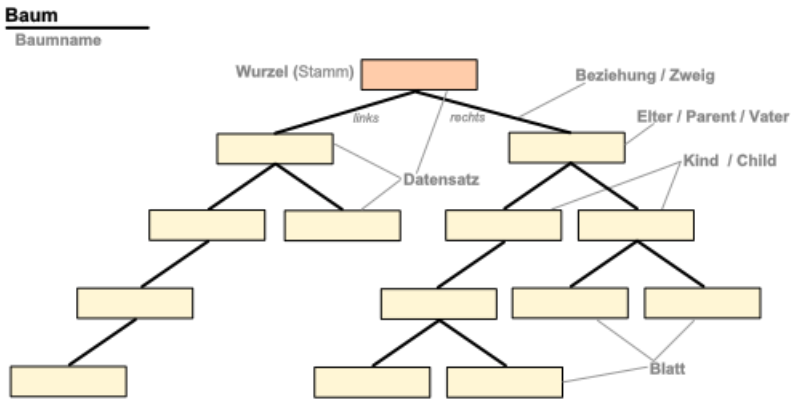
2.3.2. Baum-artiges Daten(bank)-Modell

Eltern-Element hat max. 2 Kinder

binärer Baum

Spezial-Fall der hierarchischen Struktur

im Allg. gleichartige Datensätze



per Definition immer nur max. 2 Nachfolge-Elemente (Kinder), ob dies Verzeigungen oder Blätter oder eine Mischung aus beiden ist, ist irrelevant

In der Programmierung besteht dann ein Baum-Element immer aus drei Teilen. Das ist (-meist in der Mitte dargestellt -) der Inhalt. Dieser kann für sich weiter strukturiert sein. Hier hinein würde z.B. ein Schlüssel gehören.
Daneben gibt es zwei Verweis-Felder (meist links und rechts an das Inhalts-Feld gezeichnet), die Verweise (Zeiger, Pointer) auf andere Baum-Elemente beinhalten. Gibt es kein Nachfolge-Element, dann wird ein definiertes Ende-Zeichen – z.B. NULL oder NIL – in die Verweis-Felder eingetragen. Diese Ende-Zeichen werden dann in Algorithmen zum Durchforsten oder bearbeiten ds Baumes abgefragt und entsprechend dem Inhalt (ein Verweis oder eben keiner) weiter verfahren.

Vorteile:

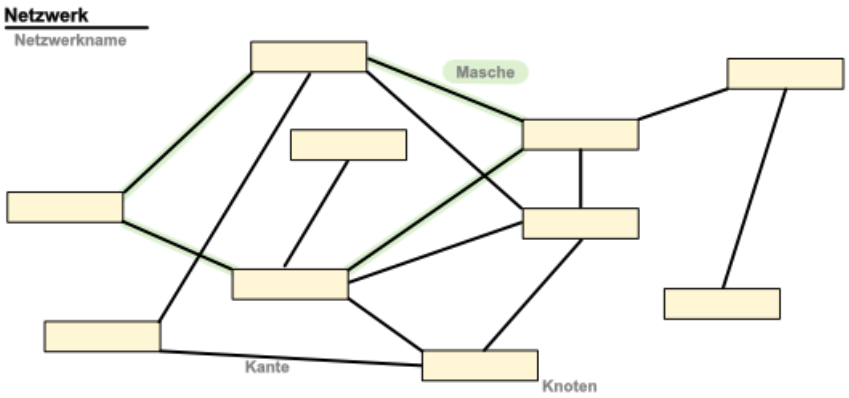
sehr effektive Algorithmen verfügbar

Nachteile:

2.3.3. netzwerkartiges Daten(bank)-Modell

hohe Ausfall-Sicherheit
andere Datenbank-Modelle haben das Problem: "Was passiert, wenn der Server ausfällt?"

modernes Stichwort: Clouds



Vorteile:

hohe Ausfall-Sicherheit
hohe Datensicherheit, da Daten mehrfach vorhanden sind
verschiedene Zugriff-Möglichkeiten
auch bei vielen parallelen Zugriffen stabil

Nachteile:

hohe Redundanz der Daten, erheblicher Aufwand für Synchronisierung notwendig
größerer Hardware-Bedarf
hohe Belastung des Netzwerkes
höheres Angriff-Potential, da einfach mehr Knoten usw.
unkontrollierte Verbreitung von Daten (unberechtige Replikationen, Fake News, Gerüchte, …)
Lösch-Möglichkeit (u.U. stark eingeschränkt bzw. nicht mehr möglich)

12 Regeln / Charakteristika von J. F. DATE zu verteilten Datenbanken

- **lokale Eigenständigkeit jedes Rechners**	Rechner arbeiten autonom; → garantiert hohe Ausfall-Sicherheit; → erhöhter Kommunikations-Aufwand zwischen Knoten
- **keine zentrale Verwaltungs-Instanz**	Rechner verwalten sich gegenseitig; → erhöhter Kommunikations-Aufwand zwischen Knoten
- **ständige Verfügbarkeit**	Gesamtsystem kann kaum / sollte nicht abgeschaltet werden
- **lokale Unabhängigkeit**	der Zugriff auf die (gewünschten) Daten ist unabhängig von der Anfrage-Position (lokal, entfernt)
- **Unabhängigkeit gegen Fragmentierung**	Verteilung der Daten (Relationen) auf mehrere Rechner / Server (sharding) hat keine Auswirkungen auf die verfügbarkeit der (gewünschten) Daten
- **Unabhängigkeit hinsichtlich Daten-Replikation**	die Replikation von Daten hat keine Auswirkungen auf die bereitstellung der (gewünschten) Daten
- **optimierte verteilte Zugriffe**	Aktionen auf der Datenbank werden durch interne Prozesse gleichmäßig auf die Knoten verteilt
- **verteilte Transaktions-Verwaltung**	vollständige Unterstützung des Transaktions-Modells; Unterstützung von Recovery (Wiederherstellung eines Datenbank-Zustandes) und Concurrency (konkurrierende Transaktionen)
- **Unabhängigkeit von der Hardware**	alle Arten von Rechner und Skalierungs-Größen werden unterstützt
- **Unabhängigkeit vom Betriebssystem**	Resultate, … sind unabhängig vom Betriebssystem des benutzten Knoten-Rechners
- **Unabhängigkeit vom Netzwerk**	Unterstützung der üblichen Netzwerk-Protokolle (z.B. TCP/IP)
- **Unabhängigkeit vom Daten-Verwaltungs-Systemen**	Unterstützung der verschiedenen / vieler Zugriffs-Sprachen (z.B. JSON, PDO, …) und SQL sowie NoSQL

CAP-Theorem (nach BREWER) besagt, dass in verteilten Datenbanken nicht gleichzeitig Konsistenz (**C**onsistence), Verfügbarkeit (**A**vailability) und Ausfall-Toleranz (Tolerance of Network **P**artitions) erfüllt sein können.
(s.a. bei **Blockchain-Technologie** im Skript 🕮 **Rechner, Netzwerke und Protokolle**)

Verfügbarkeit ist wichtiger als Konsistenz; Konsistenz ergibt sich zumeist nach Transaktion fließend / automatisch / von allein

ACID steht kontrahär dagegen; bezieht sich vorrangig auf die Transaktionen im System
ist ein etwas anders orientiertes Eigenschaften-System

beinhaltet atomity (Abgeschlossenheit), consistency (Konsistenz), isolation (Abgrenzung) und durability (Dauerhaftigkeit)
im Deutschen auch mit AKID abgekürzt (Atomarität, Konsistenz, Isolation und Dauerhaftigkeit)

in verteilten NoSQL-Datenbanken wird aber auch das BASE-Prinzip verfolgt
BASE steht für **B**asically **A**vailable, **S**oft state, **E**ventual consistency

2.3.4. Objekt-orientiertes Daten(bank)-Modell

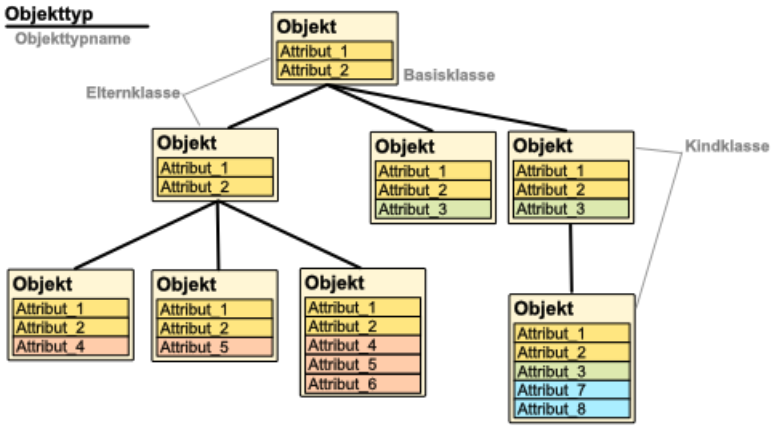
geringe Verbreitung, damit auch wenig Unterstützung durch Software-Firmen / Anwender-Programme
Algorithmen von höhergeordneten Objekten können an niedriger geordnete Objekte vererbt werden
müssen dann durch Algorithmen-Teile für die Spezialitäten der untergeordneten Objekte erweitert oder geändert (überschrieben) werden

wegen besonderer Struktur wird auch den Datenbank-Management-Systemen (DBMS) auch ein O für Objekt vorgesetzt (ODBMS)

Daten sind gekapselt, d.h. nur innerhalb der definierenden Objektklasse manipulierbar

reele Strukturen des Objektes / Systems lassen sich sehr gut abbilden / definieren / …

Beispiele:
Oracle DB, PostgreSQL



Aufgaben:

1. ***Überlegen Sie sich z.B. für das Basisobjekt Person (mit wenigen Personendaten) eine Objektstruktur mit speziellen Personen-Gruppen (Personen(-Rollen)) und ihren speziellen Attributen (Aufgaben, Merkmalen, Eigenschaften, …)!***

Nachteile:

geringe Unterstützung und wenige etaplierte Schnittstellen
durch hohe Komplexität der Objekte schwerer handhabbar
in größeren Datenbeständen häufig Performance-Probleme
zur Manipulation der Daten Objekt-orientierte Programmiersprache notwendig

Definition(en): Objekt-orientierte Datenbank
Eine Datenbank heißt Objekt-rientiert, wenn ihr grundlegendes Konzept auf Ojkete, Klassen, Vererbung und Kapselung basiert.

Begriffe des Objekt-Konzeptes

- **Objekt**	einzelne – z.T. sehr komplexe – Dinge, Begriffe, Entitäten z.B. ein bestimmter Mensch
- **Attribut (eines Objektes)** **(Eigenschaft, Merkmal)**	ein einzelnes, individuelles Merkmal des Objektes z.B. Farbe des Pullovers
- **Klasse**	Gruppe gleichartiger / vergleichbarer / klassifizierter Objekte z.B. alle Kinder; Auto's; Geld-Konten; …
- **Klassen-Attribut**	Merkmale, die für die gesamte Klasse gelten oder die gesamte Klasse beschreiben z.B. Anzahl der Kinder
- **Methode (einer Klasse)** **(Funktion, Leistung)**	Routine / Algorithmus / Aktivität, die für das, oder mit dem Objekt durchgeführt werden können
- **Kapselung**	die Eigenschaften, Methoden sind nur innerhalb des Objektes verfügbar oder mit mit Methoden ermittel- bzw. manipulierbar
- **Vererbung**	Eigenschaften und Methoiden werden an spezialisierte Objekte weitergereicht

Heute sind Objekt-orientierte Datenbank-Modelle dann vielfach objekt-relational umgesetzt. Sie verbinden die Objekt-Struktur mit Performance relationaler Datenbanken
Objekte werden in Tabellen verwaltet, die Tabellen sind flexibel aufgebaut und lassen strukturierte Inhalte zu
die Datensätze sind sehr variabel, darauf müssen die Algorithmen eingestellt werden

2.3.5. Dokument(en)-orientiertes Daten(bank)-Modell

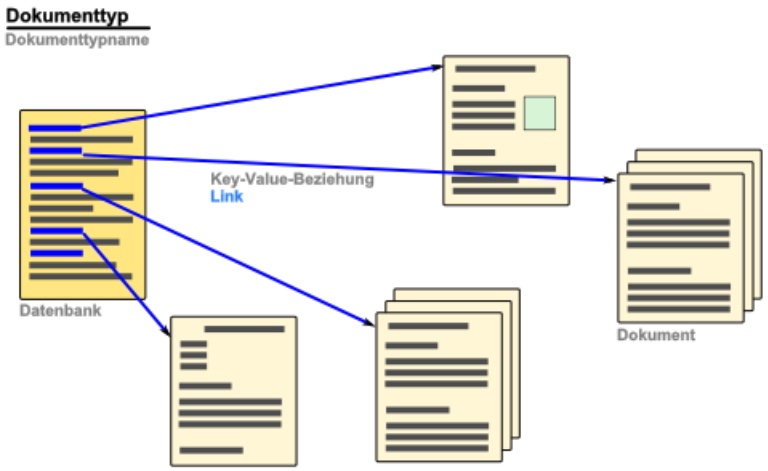
es gibt keine Relationen
in der Datenbank wird jedes Dokument über seine eindeutige ID identifiziert
alle Informationen werden über Key-Value-Paare strukturiert / gespeichert

basiert auf Lotus Notes

z.B. amazon-System Dynamo oder Riak

weitere Beispiele MongoDB, CouchDB

häufig in Web-Applikationen
Daten praktisch universell gestaltbar und formfrei



Daten werden über XML- oder JSON-Dateien ausgetauscht

Java**S**cript **O**bject **N**otation

leicht mit JavaScript verarbeitbar

etwas Speicher-effektiver als XML
zudem auch leichter lesbarer

statt 2 Tag's wird nur ein **Schlüssel** verwendet, diesem ist ein **Wert** zugeordnet
Schlüssel und Wert sind **Doppelpunkt**-getrennt

weitere Strukturierung (z.B. Hierrarchien) lassen sich durch geschweifte **{ }** und eckige **[ ]** Klammern erzeugt werden

```
{
  "Key" : 33A9CC-862F45-400BD53AF32
  "Name" : "Müller",
  "Vorname" : "Maria",
  "maennlich" : false,
  "Hobbys" : [ "Lesen", "Reiten", "Chillen" ],
  "Groesse" : 168,
  "EinheitGroesse" : "cm",
  "Elter_1" : {
    "Name" : "Müller",
    "Vorname" : "Monika",
    …
  }
  "Elter_2" : {
    "Name" : "Müller",
    "Vorname" : "Frank",
    …
  }
  …
}
```

Dokumenten-Struktur(en) im JSON-Format

auch hier sind strukturierende Leerzeichen / Einrückungen optional und können in Produktiv-Umgebungen auch weggelassen werden

Vorteile:

selbstbeschreibendes Daten-Format (Attribute und Werte erkennbar)
Maschinen- und von Menschen lesbar
Betriebssystem- und Anwendungs-übergreifend nutzbar (→ Austausch-Format)
Daten haben eine Ordnung
Speicherung gemischter Daten (bis hin zu Links auf andere Dateien usw.) möglich

Nachteile:

ungeeignet für komplexe Datenstrukturen mit hoher Beziehungstiefe
immer sehr individuelle, anpassungsfähige Algorithmen notwendig
Algorithmen passen oft nur zu bestimmter Gruppe von Entitäten (Dokumenten)

2.3.6. Graph-Datenbanken

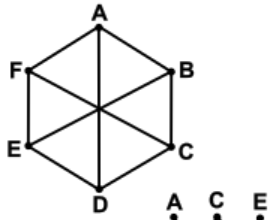
Zwischen zwei Objekten besteht u.U. eine Beziehung. In diesem Fall können wir diese Konstellation als **Graph** darstellen. Die Objekte sind die **Knoten** (auch: **Vertices**) und die Beziehung ist die **Kante** zwischen den Knoten.
Als Objekte kommen diverse Elemente in Frage. Häufig sind es:

- Personen
- Gegenstände
- Orte
- Datums- oder Zeit-Angaben
- Begriffe
- Kategorien
- Organisationen
- …

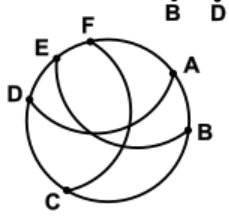
Beziehungen können auch wieder Objekte sein. Die Straßen einer Stadt sind ein schönes Beispiel. Sie verbinden die Knoten(-Punkte), die im Volksmund Kreuzungen heißen.
Die Beziehungen (auch: **Edges**) zwischen Objekten können gerichtet oder ungerichtet sein. Meist ist der gesamte Graph homogen gerichtet oder ungerichtet. Aber auch Misch-Formen sind möglich. Man denke dabei nur an die Umsetzung eines Straßen-System's als Graph. Hier kommen Einbahn-Straßen gemischt mit gewöhnlichen Straßen vor.
Ein gerichteter Graph wird auch **Digraph** genannt. Die Kanten werden mit Pfeilen versehen und dann Bogen genannt.
Existieren mehrere Kanten zwischen zwei Knoten, dann spricht man von einem Multigraphen. In Multigraphen sind auch Kanten erlaubt, die wieder zum (Ausgangs-)Knoten zurückführen.

Den Kanten (Beziehungen) können Attribute zugeordnet werden. Oft sind sie durch die Länge der Kante repräsentiert. Als Attribute eignen sich z.B.:
Größe
Länge / Abstand
Kosten
Durchfluss-Mengen / -Kapazitäten
Übergangs-Raten
…

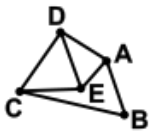
Sind die Kanten-Länge kein Repräsentant für ein Attribut, dann kann ein Graph auf sehr unterschiedliche Art und Weise dargestellt werden. Man nennt den Graph dann **isomorph**.



Graphen stellen für Computer ein recht komplexes Problem dar.
Oft sind wir Menschen bei graphischen Interpretationen und Erkennen den Computern überlegen.
Welche Schwierigkeiten aber auch wir bei der Graphen-Verarbeitung haben können, zeigen die nebenstehenden Bilder. Auf den ersten Blick scheinen die Graphen sehr verschieden zu sein. Bei einer genaueren Betrachtung sind die Graphen aber äquivalent.



Kreuzen sich die Kanten nur in den Knoten, dann nennt man den Graph auch **planar**. Die Räume einer gewöhnlichen Wohnung könnten durch einen planaren Graphen dargestellt werden. Die Zimmer liegen dann in einer Ebene. Überkreuzen sich Kanten außerhalb der Knoten, dann muss eine zusätzliche Ebene (Etage) vorgesehen werden und Übergänge zwischen den Ebenen (z.B. Treppen) eingeplant werden.



Die Anzahl der Kanten, die sich in einem Knoten treffen steht für den **Grad des Knoten**'s.
Der Mittelwert aller Knoten-Grade nennt sich Branching-Faktor.

Für alle Züge einer Schach-Partie kann man z.B. einen Graphen aufstellen. Hier liegt der typische Branching-Faktor zwischen 31 und 35. Somit hat ein Spieler immer nur durchschnittlich 33 verschiedene Zug-Möglichkeiten.
Beim Spiel Go liegt der Branching-Faktor bei 250. U.a. deshalb hat es bis vor wenige Jahre gedauert, dass ein Computer einen erfahrenen Go-Spieler schlagen kann.

Eine endliche Folge von Knoten und Kanten, die wieder in einem Knoten enden, werden **Pfad** genannt.
Bei einem **geschlossenen Pfad** ist der Start- und End-Knoten identisch.

Ein geschlossener Pfad mit mindestens 3 Knoten wird Zyklus genannt.
Einfache Pfade, die nicht geschlossen sind, nennt man Bäume.

Ein **HAMILTON-Kreis** ist ein geschlossener Weg (Pfad) durch einen Graphen, bei dem die Knoten genau einmal durchlaufen werden müssen. Die Nutzung aller Kanten ist nicht notwendig.
HAMILTION-Kreise sind z.B. bei der Planung von Liefer-Routen bedeutsam.
Das HAMILTON-Problem ist NP-vollständig.
Leichter lösbar ist das Erstellen eines EULER-Kreises. Hier geht es darum einen geschlossenen Weg durch den Graphen zu finden, bei dem alle Kanten exakt einmal durchlaufen werden. Die Knoten können beliebig oft angesteuert werden.

Beim Traversieren sucht man einen Pfad, bei dem jeder Knoten und jede Kante genau einmal enthalten ist.

Im Projekt-Management werden Abläufe häufig als ein gerichteter Graph dargestellt. Die Kanten(-Längen) repräsentieren dann z.B. die Zeit-Dauer, die von einem Projekt-Zustand (Knoten1) zu einem anderen (Knoten2) benötigt wird. Der kürzeste Pfad vom Start-Knoten (Projekt-Start) bis zum End-Knoten (Projekt-Abschluss) stellt dann die minimale / kürzeste Projekt-Dauer dar. Dieser kürzeste Pfad muss aber alle notwendigen Zwischen-Zustände (Zwischen-Knoten) beinhalten. Dieser Pfad enthält eine Kette der benötigten Knoten und wird auch kritischer Pfad genannt.
Mit der CPM (Critical Path Method) versucht man den kritischen Pfad in einem (Projekt-)Graphen zu finden.

Graphen und Graph-Datenbanken werden für die Planung, Optimierung und Steuerung von verschiedensten Problemen und Strukturen benutzt. Dazu gehören:

- Straßen-Netze, U-Bahn-Netze
- Leiterbahnen
- Projekt-Planungen
- Stammbäume
- Liefer- und Abhohl-Aufgaben
- Ablauf-Pläne
- Landkarten

- Computer-Netze
- Organigramme
- usw. usf.

Mit Hilfe spezieller informatischer Modelle und Modellierungs-Methoden werden Graphen in Datenbank umgesetzt.

Beim **Resource Description Framework (RDF)** handelt es sich
semantic web
Daten werden hier weiteren mit Kontext-Informationen versehen
Graph-Daten werden in Form von Triple gespeichert. Ein Triple besteht aus dem Start-Knoten als Objekt sowie ein die Kante beschreibendes Prädikat und als Drittes dem End-Knoten – wieder als Objekt.

Die Daten können dabei auch schon in textueller Form vorliegen. So könnten Twitter-Beziehungen z.B. so aussehen:

Nils	--folgt	Ronja
Nils	--folgt	Tobias
Tobias	--folgt	Tom
Ronja	--folgt	Helena

Alle Objekte, Knoten, Kanten und Prädikate werden durch sogeannte URI's (Unified Resource Identifier) bestimmt.
URL's (Unified Resource Locator) von Webseiten sind Spezial-Fälle solcher URI's. Sie beschreiben eben die Quelle / Lage einer Web-Seite / Internet-Ressource.

Vorteile:
- Eindeutigkeit

Labeled Property Graph Model (LPG)

beim LPG haben die Knoten und Kanten eine innere Struktur
diese wird vorrangig in Key-Value-Store's gespeichert
für einen Knoten oder eine Kante können unterschiedliche viele Key-Value-Paare (Schlüssel-Wert-Paare) vorliegen

Vorteile:
- i.A. kompakter
- einfachere Beschreibung der Objekt-Subjekt-Beziehungen

Es gibt sowohl für RDF also auch für LPG geeignete Datenbank-Systeme

Native Graph-Datenbanken arbeiten direkt auf den Daten. Bei den nicht-nativen Graph-Datenbanken werden die Daten serialisiert und in typischen relationalen od.a. Datenbanken verarbeitet.

bei den Abfrage-Sprachen gibt es keinen Standard, der mit SQL vergleichbar wäre
bekannte Abfrage-Sprachen sind Gremlin, SPARQL und GraphQL

Fluss-Überquerungs-Problem als Graph-Modell

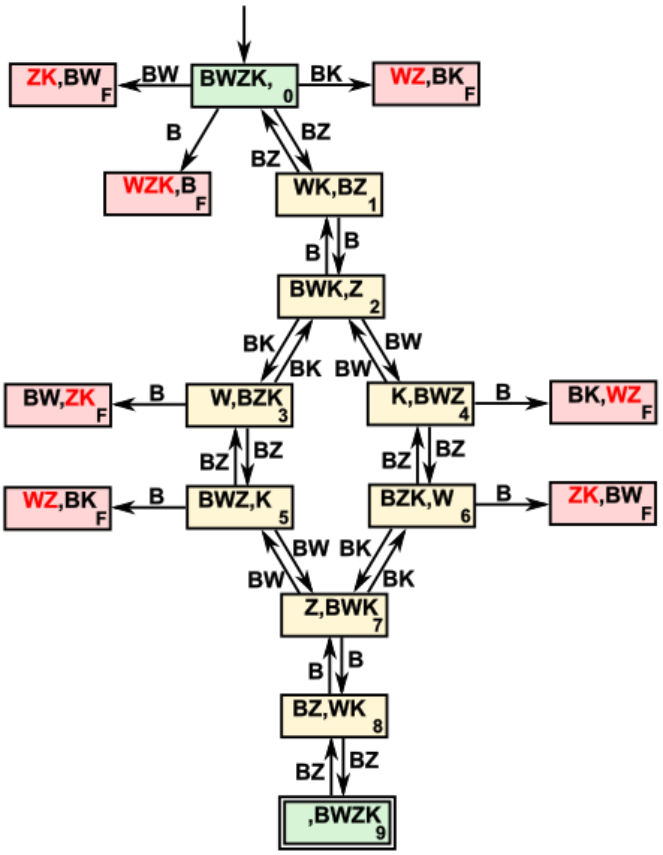
Wahrscheinlich kennen Sie das Rätsel schon, aber es ist immer wieder interessant, wie man es lösen kann:
Ein Bauer will mit einem Wolf, einer Ziege und einem Kohlkopf in die Stadt. Dazu muss er über einen Fluss. Für die Überquerung steht ihm nur ein sehr einfaches Boot zur Verfügung. Es trägt nur den Bauern und eines seiner mitgeführten Objekte. Das Problem besteht nun darin die Überfahrt(en) so zu planen, dass bei der Abwesenheit des Bauern's auf einer Seite die natürlichen Fress-Beziehungen nicht ihren Lauf nehmen können und z.B. die Ziege den Kohl frisst, während der Bauer den Wolf über den Fluss bringt.

Aufgaben:

1. *Lösen Sie das Problem auf dem Papier! Notieren Sie sich den von Ihnen erkundeten Lösungs-Pfad in der folgenden Form:*
   z.B. Ausgangs-Situation: alle (Bauer, Wolf, Ziege, Kohl) auf der einen Seite:
   BWZK// (die zwei Striche stehen für den Fluss)
   bei einem Boot-Wechsel von Bauer und Ziege schreiben wir an die Kante: BZ und dann den neuen Knoten (Situation, Zustand)
   BZ
   also: BWZK// ⟶ WK//BZ
2. *Vergleichen Sie Ihren Pfad mit den Pfaden anderer Kurzteilnehmer! Wieviel Lösungen gibt es / haben Sie im Kurs gefunden? Welcher Pfad ist der kürzeste? Wieviele kürzeste Pfade gibt es?*

Im nebenstehenden Graphen sind die Start- und Ziel-Zustände grün markiert. Die rötlichen Zustände sind Fehler-Zustände. Das "Problem" ist nochmals hervorgehoben.

Gesucht werden die möglichen Pfade zur Lösung des Problem's und dann natürlich der kürzeste Pfad (- eine Traverse).



Aufgaben:

1. *Bei der Betrachtung des obigen Graphen ist mir ein Fehler aufgefallen. Während man bei allen benachbarten Knoten-Paaren immer Hin- und Zurück-gelangen kann, fehlt dies bei den Fehler-Knoten (/-Zuständen). Muss hier berichtigt werden? Wenn JA, in welcher Form, wenn NEIN, warum nicht? Erläutern Sie Ihre Meinung!*

Quelle (zu diesem Abschnitt):

BRENDEL, Jens-Christoph: Beziehungsfragen – Wie Graph-Datenbanken funktionieren und was sie leisten.-IN: Linux Magazin 07.2020.-S. 20 ff.

Das hört sich alles sehr modern an, aber die Theorie über Graphen (auch wenn sie zuerst nicht so genannt wurden) ist rund 300 Jahre alt. Sie geht auf den Mathematiker Leonhard EULER (1707 – 1783) zurück. Er benutzte die Zusammenhänge, um das Königsberger Brücken-Problem zu lösen.

Aufgaben:

1. ***Recherchieren Sie, warum es sich beim Königsberger Brücken-Problem handelt! Finden Sie eine Lösung?***
2. *2.*
3. *3.*

Die einfachste mathematische Darstellung von Graphen sind Adjazenz-Listen. In diesen werden für jeden Knoten alle Nachbarn bzw. Nachfolger (in gerichteten Graphen) notiert. Praktisch ist der Graph dann eine Liste aus Listen von Nachbarschaften.
In mathematischen Berechnungen oder im Computer werden Graphen häufig durch Matrizen dargestellt. Die klassische Form sind sogenannte Adjazenz-Matrizen (Nachbarschafts-Matrizen). Für Matrizen gibt es viele definierte Operationen, die auch in vielen Computer-Systemen durch besonders schnelle, maschinen-nahe Programme / Funktionen umgesetzt sind.
Für einen Graphen mit 10 Knoten ergibt sich eine 10x10-Matrix. Das ergibt 100 mögliche Beziehungen und entsprechend viele Werte inder Matrix. Soll nur ein ungewichteter also einfacher Graph dargestellt werden, dann erhält das Matrix-Element den Wert 1. Bei gewichteten Graphen wird der Kanten-Wert eingesetzt. Hat die Kante keine Richtung gibt man den gleichen Wert beim (an der Haupt-Diagonalen) gespiegelten Matrizen-Element ein.
Der große Nachteil von Adjazenz-Matrizen ist ihre quadratische Skalierung. Die Größe der Matrix ist immer das Quadrat der Kanten-Anzahl. Praktisch heißt das, dass der Aufwand mit der Größe quadratisch wächst. Das ist besonders ärgerlich, wenn es nur relativ wenige Kanten (besetzte Elemente) im Graphen gibt. In diesem Fall sind ja die meisten Elemente der Matrix Null-Elemente.

Bei den seltener verwendeten Inzidenz-Matrizen handelt es sich um Knoten-Kanten-Matrizen.

alter Text:
Speicherung von Graphen in Datenbanken

z.B. für sozial Media notwendig
besonders gut für die Mustersuche geeignet

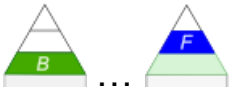
üblicherweise gibt es Tabellen für Knoten und es gibt welche für Kanten

Graph ist System aus Knoten und Kanten (Verbindungen zwischen Knoten)
Kanten können gerichtet oder ungerichtet sein

Beispiele:

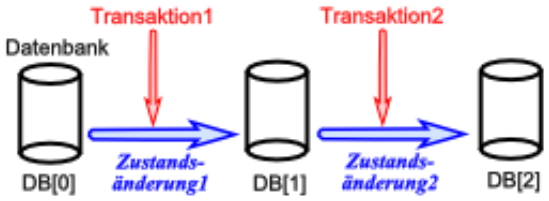
- neo4j
- OpenLink Virtuoso

2.4. Transaktionen und das Transaktions-Modell



Transaktionen sind Prozesse (Aufgaben, …) in einer Datenbank durch ein Anwendungs-Programm

transformieren einen gegebenen Datenbank-Zustand in einen neuen



Transaktionen werden protokolliert
i.A. ist ein Zurückversetzen der Datenbank (Rollback) in einen vorherigen Zustand durch Anwendung der umgekehrten / reziproken Transaktion möglich
die Transaktionen müssen in umgekehrter Reihenfolge ausgewertet werden
Probleme bei Lösch-Aktionen oder Umstellen von Tabellen / Datenbanken möglich

Integritätsbedingungen sind unabhängig von den Daten und Anwendungsprogrammen verwaltete Regeln und Voraussetzungen, die für die Daten und die Datenbank insgesamt in jedem Zustand eingehalten werden müssen

zeitlich verzahnte Transaktionen dürfen keine inkorrekten Ergebnisse oder Verstöße gegen die Integritätsbedingungen erbringen
bei einem Buchungs-Versuch eines Sitzplatzes in einem Flugzeug muss eine offene Abfrage durch den einen Client diesen Platz solange blockieren, bis entweder gebucht oder nicht gebucht wird. erst danach darf der Sitzplatz für andere Clients wieder sichtbar werden

Transaktionen müssen vollzogen (abgeschlossen) sein oder nicht

ACID-Eigenschaften von Transaktionen

- **Atomicity**
  Atomität
  Zustandsänderungen der Datenbank sind atomar und sind entweder vollständig oder nicht vollzogen

- **Consitency**
  Konsistenz
  jede Transaktion ergibt ausgehend von einem konsistenten Zustand wieder einen konsistenten (Bedingungs-gerechten) Zustandsübergang in der Dantenbank
  Datenbank ist nach der Transaktion also wieder konsistent

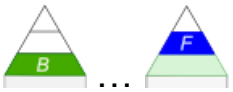
- **Isolation**
  simulierter / virtualisierter Einzelnutzer-Betrieb
  die System-Zustände die von einer Transaktion erzeugt werden, können von anderen nicht gesehen oder verändert werden

- **Durability**
  Dauerhaftigkeit
  nach Vollzug der Transaktion (oder eben keiner) ist der neue Zustand persistent (dauerhaft)
  Zustand kann nur wieder durch eine Transaktion geändert werden

Definition(en): Transaktion
Eine Transaktion ist eine Sequenz von Operationen, die in einem System als eine logische Einheit konzipiert und auch so behandelt werden.

write-ahead-log-Regel:
Die Protokoll-Informationen für eine Transaktion muss physikalisch in die Protokoll-Datei geschrieben werden, bevor die Transaktion physikalisch die Veränderungen vornimmt.

2.5. Daten-Verteilung und -Sicherheit in Datenbanksystemen



Funktions-Prinzipien

- **Strukturierung**
- **Replikation**
- **Partitionierung**

Gründe für die Verteilung von Daten

- **große Datenmengen**
- **Ausfallsicherheit** — Schaffung von Redundanzen
- **schneller (regionaler) Zugriff**
  Last-Balanzierung

Typen / Architekturen der Verteilung

- **Shared Memory** — gemeinsam genutzter Speicher von mehreren Verarbeitungs-Einheiten (z.B. CPU's einschließlich eigener Datenträger)
- **Shared Disk** — Verarbeitungs-Einheiten (mit eigener CPU und Hauptspeicher) benutzen den gleichen Datenträger (z.B. Disk-Array's)
- **Shared Nothing** — es gibt keine gemeinsame Nutzung von Hardware
  Daten werden zwischen eigenständigen System z.B. über das Internet verteilt

insgesamt ist Shared Nothing die aktuell am weitesten verbreitete Architektur → z.B. Cloud's

einfache Datenbank-Systeme sind Single-Server-Systeme
praktisch ohne Verteilung, Replikation und Fragmentierung

Verteilungs-Muster von Daten in Datenbank-Systemen

- **Fragmentierung**
  Sharding — verteilen unterschiedlicher Daten oder von Daten-Teilen auf mehrere Server; verbessert Performance und sorgt für gleichmäßige / angepasste Lasten-Verteilung

- **Replikation** — verteilen der gleichen Daten auf mehrere Server; verbessert die Daten- und Ausfall-Sicherheit

Verfahren der Daten-Verteilung (Partitionierung)

- **Round Robin Verfahren** — Verteilung neuer Datensätze (Tupel) oder Daten-Blöcke immer reihum auf die einzelnen Server
  Vorteile:
  - gute Last-Verteilung

  Nachteile:
  - kein Server hat den gesamten Datenbestand
  - Ausfallsicherheit nur begrenzt gegeben

- **Partitionieren** — Aufteilung der Daten nach bestimmten Kriterien
  Vorteile:
  - gute Kombination von Zugriffzeit und Speicherort

  Nachteile:
  - kein Server hat den gesamten Datenbestand
  - Ausfallsicherheit nur begrenzt gegeben
  - ev. schlechte Lasten auf den Servern

- **Hash-basierte Verfahren** — aus den (einzelnen) Daten(sätzen) wird ein Hash berechnet und dieser Hash-Wert wird dann zur Partitionierung / Verteilung der Daten genutzt
  Vorteile:
  - gute Last-Verteilung

  Nachteile:
  - hoher Rechenaufwand
  - zusätzliche Daten (Hash-Wert)

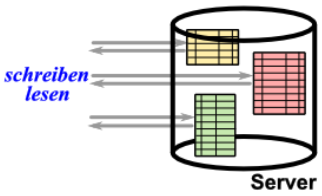
Kombinationen

- **Master/Slave-Replikation** neben dem Haupt-System (Master) gibt es untergeordnete Systeme (Slave's)
  untergeord. Systeme enthalten i.A. nicht alle Daten; Master verfügt über den gesammelten / zusammengesetzten Daten-Bestand

- **Peer-to-Peer-Replikation** alle Systeme sind gleichberechtigt und brauchen / enthalten den gesamten Daten-Bestand

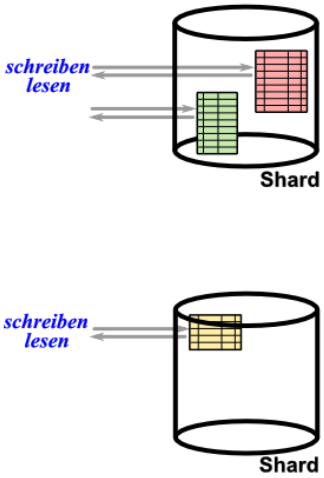
Die Datenbank besteht nur aus einem Server. Das kann ein lokaler oder auch ein Web-Server sein.

lokale Datenbanken auf einem einzelnen Rechner oder auch in einem internen Netz (Intranet)



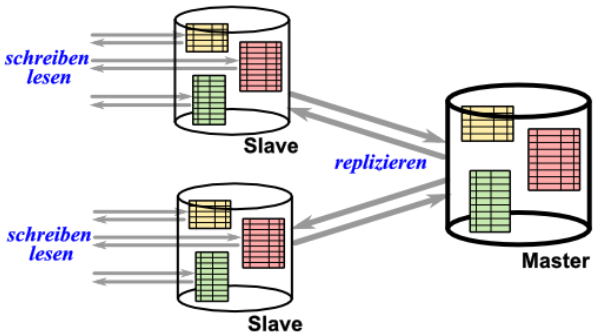
Single-Server-Verteilung

Hier besteht die Datenbank aus mehreren – voneinander unabhängigen Datenbank-Teilen, die quasi als eigenständige Datenbanken fungieren. Auf jeder der Teil-Datenbanken (Fragmente) wird separat zugegriffen. Shard ist die englische Bezeichnung für ein Fragment
Man spricht auch vom Sharding (Fragmentieren).
Zwischen den Fragmenten erfolgt praktisch keine Replikation.
Die normale Nutzung beschränkt sich auf ein Fragment.
Eine gemeinsame Nutzung der Fragmente in einer Super-Abfrage ist aber möglich.



Shard-Verteilung /
fragmentierte Verteilung

Slave-Datenbanken enthalten entweder nur Teile der einzelnen Tabellen oder stellen temporäre (Arbeits-)Kopien der Datenbank dar. Die Nutzer können sie quasi lokal bei sich nutzen, was meist deutlich schneller abläuft als ein entfernter Zugriff auf den Master.
Zu bestimmten Zeiten (meist nachts) werden die Daten repliziert, das heisst die Datenbestände werden abgeglichen / synchronisiert.

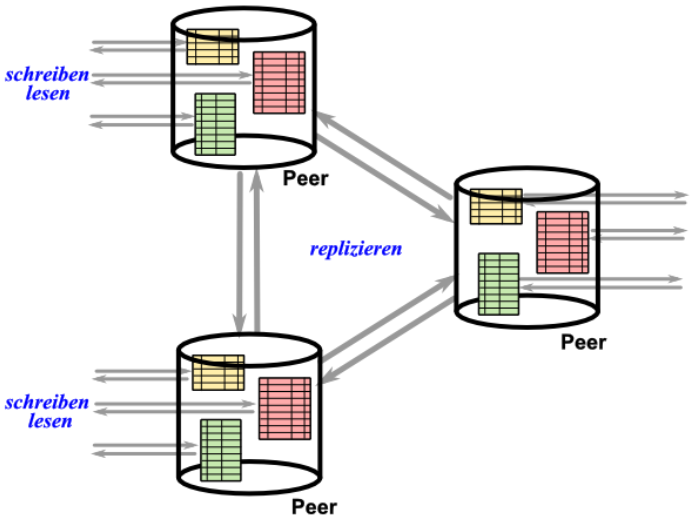


Master/Slave-Verteilung

Der Slave schreibt die am Tage veränderten Daten auf die Master-Datenbank. Andere Slave's tun das dann auch. Dannach wird die Master-Datenbank wieder zu allen Slave-Systemen geschrieben, so dass diese dann über den aktuelen Gesamt-Daten-Bestand verfügen.

Daten-Banken im Peer-to-Peer-Verbund sind gleichberechtigt. Jede hat den gleichen Daten-Bestand.
Änderungen in einer Datenbank werden dann sofort auf die anderen Peer's übertragen. Es erfolgt also eine ständige Replikation.

z.B. in Blockchain-Systemen realisiert



Peer-to-Peer-Verteilung

Links:
http://www.datenbanken-verstehen.de/
http://www.oszhandel.de/gymnasium/faecher/informatik/datenbanken/index.htm

3. Datenbanken – Implementierung

Problem-Fragen für Selbstorganisiertes Lernen

Wie macht man ausgehend von einer Problem-Situation / einer allgemein formulierten Aufgabe eine passende Datenbank?

Welche Entwicklungs-Schritte muss man zur sicheren Konzeption einer Datenbank unternehmen?
Was muss man bei der Erstellung von Datenbanken unbedingt beachten?
Gibt es dabei immer nur eine Lösung?

Mit welchem Programm / Datenbanksystem / Datenbank-Management-System arbeitet man am Besten?

Voraussetzung ist konzeptionelles Modell

Welche Daten müssen erfasst werden? In welchen Beziehungen stehen sie zueinander?

Analyse, Wer braucht Welche Daten in Welcher Darstellung und Was soll mit den Daten gemacht werden
Welche zusätzlichen Daten sind notwendig (z.B. um Datenintegrität oder Datenschutz zu realisieren)?

Extension sind die Daten in den Tabellen (zum Datenbank-Entwurf)

verschiedene Modell-Typen bekannt, die jeweils bestimmte Aspekte besser oder weniger gut beachten

Objekte mit ihren Eigenschaften
Beziehungen (Verbindungen) zwischen Objekten oder Beziehungen zu anderen Beziehungen

Menge der betrachteten / relevanten Objekte und Beziehungen → Miniwelt

3.1. Problem-Lösen – von der Realität zum Modell, vom Problem zur Lösung

Nachdem wir schon die verschiedenen Daten-Modelle besprochen haben und uns auf die relationellen Datenbanken konzentrieren wollen, kommen wir nun zur praktischen Umsetzung in eine Datenbank-Management-System (DBMS).
Als Modellierungs-Werkzeug haben wir die Entity-Relationship-Diagramme (ERD) gewählt.
Es folgt die Umsetzung von Sachverhalten aus der Realität oder ausgedachter Situationen in Datenbank-Anwendungen.

Vielleicht ist die Wahl des Begriffs Problem zu Anfang etwas überzogen. Bei Problemen liegt die Schwierigkeit im Vergleich zu einer sofort lösbaren Aufgabe darin, dass kein direkter Lösungs-Algorithmus vorliegt, sondern ein grobes Anleitungs-Schema umgesetzt werden muss. Sachverhalte orientieren sich im Wesentlichen am bisher Gelernten, den vorhandenen Fähigkeiten und Fertigkeiten.
Bei echten Problemen sind i.A. neue Wege und Lösungen gesucht. Hier kommen zur Lösung / besser Bearbeitung vor allem Heuristiken zur Anwendung.
Problem-Lösen ist aber gängiger Begriff. Die überhöhte Begriffs-Verwendung Bauch-pinselt eher unser Ego, wenn wir dann die Lösung gefunden haben.
Später und Praxis-orientiert werden wir es aber dann auch wirklich mit Problemen zu tun haben. Vor allem, wenn neuartige Umsetzungen von Anwendungs-Szenarien gefragt sind.

aus praktischer Sicht:

1. Daten-Aquise (Datenbeschaffung)	Sammlung aller Daten / Daten-Arten, die in der Datenbank gespeichert, … werden sollen
2. Daten-Analyse	Untersuchung der Daten auf Redundanzen (Mehrfachspeicherung), Strukturen und Muster
3. Daten-Modellierung	Erstellen eines abstrakten Schemas der Daten

Bei vorliegenden Aufgaben / Aufgabenstellungen ist es immer notwendig, eine gründliche Analyse vorzunehmen. Vielfach sind die Aufgaben-Steller (Auftraggeber) keine Informatiker oder Datenbank-Profi's – sie geben die Aufgabe so vor, wie ihnen der Schnabel gewachsen ist. Es ist also die erste Aufgabe, das eigentlliche Problem oder die Aufgabe an sich zu erkennen und sauber zu umreißen. In der Praxis bietet es sich an z.B. die Aufgabe zu paraphrasieren, also den Sachverhalt an den Aufgabensteller zurück zu erläutern (Wiedergeben des Gesagten mit eigenen Worten.). Das kann man z.B. so anfangen: "Wenn ich Sie recht verstanden habe, …. " oder "Gehe ich recht in der Annahme, dass …?" oder "Meinten Sie, ?".
Wenn der Auftraggeber dann zustimmt, hat man zumindestens eine gemeinsame Verständnis-Ebene gefunden. Ansonsten nähert man sich durch die erneuten Erläuterungen oder Richtigstellungen schrittweise einer gemeinsamen Betrachtung an.
Wichtig ist weiterhin, sich auf die Aufgabe zu konzentrieren. Man sollte mit eigenen Ideen, Interpretationen und Erweiterungen sehr sparsam umgehen. Erst die Aufgabe, wie sie vorgegen ist, lösen und dann, wenn dafür Zeit und Raum ist, die eigenen kreativen Ideen beitragen.

Für zusätzlichen "Schnick-Schnack" bezahlt ein Auftraggeber nur, wenn seine Grundanforderungen (des sogenannten "Pflichtenheftes") erfüllt sind und er die Zusätze als wirklich Gewinn-bringend ansieht.

Schritte der praktischen Datenbank-Erstellung

1. **Datenbank planen**
2. **Datenbank-Datei erstellen**
3. **Tabellen erstellen**
4. **Formulare, Abfragen und Berichte hinzufügen**

typische Datenbank-Anwendungen

- **Lager-Verwaltung**
- **Schul-Datenbank**
- **Bibliotheks-Verwaltung**
- **Warenwirtschaftssyteme**
- **Vereins-Verwaltung**
- **…**

3.1.1. konzeptionelle Phase – Grenzen ziehen

Problemwelt (Miniwelt, Modellwelt, …) als Ausschnitt der Realität, für die eine informatische Lösung gesucht wird
häufig separat als externe Phase (auch Spezifikations- und Anforderungs-Analyse) verstanden

- Beschreibung der Daten
- Festlegung / Erkundung des Informations-Bedarf's der Nutzer
- Strukurieren des Informatins-Bedarf's
-

→ informelle Beschreibung des Fach-Problems

Modell als eingeschränktes Abbild der Realität für bestimmte Einsatzzwecke
Grund-Eigenschaften eines Modells

- Abbildung
- Verkürzung / Vereinfachung
- Pragmatismus / Handhabarkeit

eigentliche konzeptionelle Phase
Ziel ist formale Beschreibung des ausgewählten Sachverhalts
gesucht ist ein semantisches Modell des Sachverhaltes
eine Möglichkeit der Darstellung ist das ER-Modell

→ Fachkonzept der Datenbank

Manchmal ist es einfacher zu beschreiben, was alles nicht zum Problem gehört. Sollte dabei etwas auftauchen, wass eigentlich in der Aufgabenstellung gefordert ist, muss passend korrigiert und abgegrenzt werden.
Erfahrungsgemäß laufen hier die meisten Modellierungen aus dem Ruder. Den Modellierern fällt es meist sehr schwer, sich an den harten Fakten und den Verfahren (Algorithmen) zu orientieren.

Klassifizierung der Real-Objekte in Klassen (Entitäts-Typen), Objekten (Entitäten), Beziehungen

zusammensammeln der abzubildende Elemente der Miniwelt und Einordnung in Datenbank-Kategorien

abzubildendes Teil / Element / … aus der Miniwelt	**allg. Kategorie**	**Datenbank-Kategorie**	**Bemerkungen / Hinweise**

umsortieren und mit etwas Erfahrung auch gleich systematisch zusammengestellt

allg. Kategorie	**Datenbank-Kategorie**	**abzubildendes Teil / Element / … aus der Miniwelt**	**Bemerkungen / Hinweise**

3.1.2. logische Phase – die Realität abbilden – ein Modell erstellen

Prozess der Modell-Bildung

- **Abgrenzung** Ausschluss / Ausgrenzung irrelevanter Objekte und Beziehungen
- **Reduktion** Eingrenzung der Objekt-Merkmale / Weglassen nichtrelevanter Eigenschaften und Beziehungen
- **Dekomposition** Zerlegung der übriggebliebenen Problemwelt (Modellwelt) in Segemente / einzelne Objekte und Beziehungen
- **Aggregation** Zusammensetzen / Vereinigen der Segemente / Objekte und Beziehungen
- **Abstraktion** Bildung von Begriffen und Klassen
- **Testung** Prüfen, ob mit dem Modell die Realität angemessen abgebildet wird

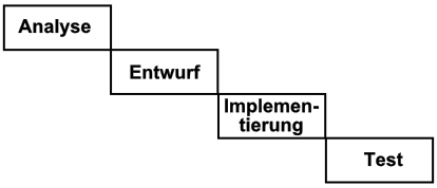
Beim Modellieren ist es wichtig, sich auf das Wesentlich zu konzentrieren. Für den Computer und für die Datenbank-Arbeit ist es z.B. bei einem Auto gleichwichtig, mit welcher Farbe, mit welcher Leistung (PS) und mit welchem Preis es daherkommt. Für uns Menschen sieht das sich anders aus. Während ich mich vielleicht sehr stark an der Anzahl der Pferdestärken orientiere, interessiert sich meine Frau vielleicht nur für die Farbe (um mal die Klischee's zu bedienen. Mein Sohn wird dagegen nur auf den Preis achten, weil er noch kein eigenes Geld verdient.

Erstellung des logischen Modell's

Entscheidung für ein Daten-Modell (z.B. Tabelle → realtionale Datenbank)
Transformation des ER-Modell in das Daten-Modell / Datenbank-Schema
Optimierung des realtionalen Schema's (/ der Daten-Strukturen) z.B. durch Normalisierungen

Vorgehens-Modelle für die Entwicklung einer Datenbank

- **Wasserfall-Modell**

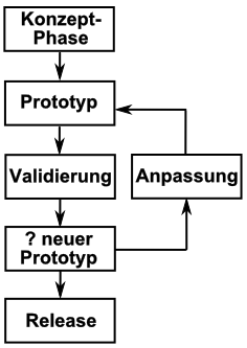


die Phasen Analyse, Entwurf, Implementierung und Test werden als sequenzielle Abfolge betrachtet
Phasen werden streng hintereinander abgearbeitet
es ist kein Rücksprung in die vorhergehende Phase vorgesehen (nur im erweiterten Wasserfall-Modell)
dieses Vorgehen ist dann möglich, wen die Anforderungen, Leistungen und Abläufe klar definiert wurden

Vorteile:
- saubere Phasen(-Grenzen) / Abrechnungs- / Kontroll-Punkte

Nachteile:
- Ergebnisse einer Phase erst sehr spät sichtbar
- wenig (Phasen-übergreifende) parallele Arbeiten möglich
- Entwicklung kann nicht interativ erfolgen
- wenige Richtlinien für die Projekt-Dokumentation

- **Rapid Prototyping**



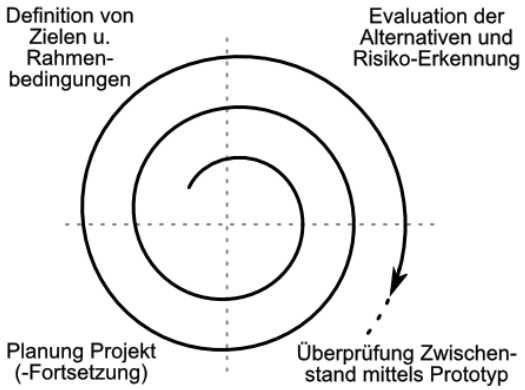
agiles, schnelles Erstellen von (teilweise) funktions-fähigen Prototypen, die beim Auftraggeber immer schon mal (an-)getestet werden können
Prototypen werden immer weiter verbessert, Leistungs-fähiger gemacht, bis sie die Anforderungen erfüllen

Vorteile:
- schnelles Bereitstellen eines Prototypen, der dem Ziel und der der Endversion schon in Grenzen entspricht
- setzt i.A. großes und Ziel-gerichtetes Entwicklungs-Potential frei

Nachteile:
- hohe Fehler-Anfälligkeit bei Erst-Projekten oder bei Anfängern
- Kosten können schnell Butget sprengen (da aus der Anwendungs-Praxis schnell neue Ideen / Ziele / Wünsche) entstehen

- **Spiral-Modell**

Entwicklung läuft in einem interativen Zyklus
nach jedem Durchlauf ist ein Prototyp der IST-Zustands vorhanden (reduziert Entwicklungs-Risiken)



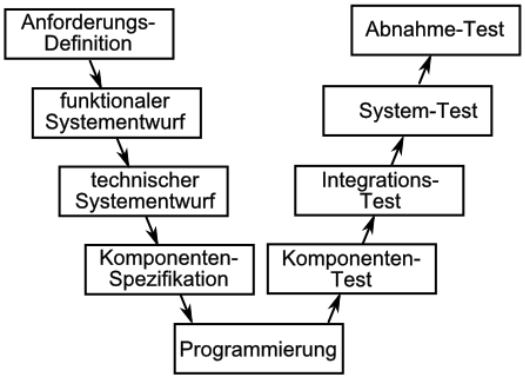
Vorteile:
- interatives Arbeiten
- häufige Risiko-Analyse
- Erstellung / Vorhandensein von Prototypen

Nachteile:
- keine Festlegung von Verantwortlichkeiten
- keine parallelen Abläufe vorgesehen

Vorgehens-Modelle für die Entwicklung einer Datenbank (Fortsetzung)

- **V-Modell**
  (V-Modell XT)

Entwicklung wird besonders aus technischer und funktionaler Sicht betrachtet
ähnelt dem Wasserfall-Modell, sieht aber darüber hinaus auch Rückkopplungen auf vorherige Phase vor
verstärkte Berücksichtigung von Qualitätssicherungs-Maßnahmen
beinhaltet auch Festlegung von Verantwortlichen



Vorteile:
- für alle Unternehmes-Arten / Projekte geeignet
- gut für kleine und mittlere Projekte geeignet
- interatives Arbeiten
- Verringerung von Risiken / Begrenzung der Projekt-Kosten
- gute Qualitätssicherung
- verbesserte Kommunikation zwischen Projekt-Beteiligten
- Erstellung von (teilweise nutzbaren) Prototypen im Projekt-Verlauf

Nachteile:
- Vergabe von Dienstleistungen ist nicht definiert
- keine Unterscheidung von Projekt-Auftraggeber und –nehmer bei der Einführung und Pflege des Projekts
- Organisation und Ablaufsteuerung nicht bestimmt

- **Scrum**

leichtes Framework für komplexe Projekte
stark auf Team-Arbeit und Eigenverantwortlichkeit orientiert
Team besteht minimal aus drei Rollen: Product Owner, Entwicklungs-Team und Scrum-Master
viel Kommunikation, Kreativität und Transparenz; hohe Produktivität
ständige Festlegungen und Weiterentwicklungen / auch Richtungs-Änderungen möglich

Vorteile:
- saubere Festlegung von "Definition of Done" (DoD, Fertigstellungs-Kriterium (unabhängig vom weiteren Verbesserungs-Potential))

Nachteile:
- sehr viel Eigenverantwortung, Transparenz und Team-Fähigkeit notwendig
- viele Team-Sitzungen (aufwendige Planung und Koordinierung)

-

Aufgaben:

1. *In einem AccessPoint mit 5 Kanälen (Antennen) soll in einer Datenbank die aktuelle Verbindungs-Situation gespeichert werden. Jedem Kanal sind bestimmte veränderliche und eigenständige Eigenschaften (Frequenz, Orientierung, Verschlüsselung) zugeordnet. Die sich verbindenden Endgeräte (Client's) sind durch eine MAC-Adresse und einem Gerätenamen gekennzeichnet. Neue Client's sollen später durch andere Funktionen des AccessPoint's hinzugefügt werden. Alte Client's bleiben in der Datenbank.*
   Erstellen Sie ein Entity-Relationship-Diagramm für die Verbindungs-Datenbank!
2. *Die Fahrrad-Ausleihe "Ostsee-Bike" hat 25 Fahrräder für Kunden bereitstehen. Einige der Fahrräder sind eBike's. Außer der Rahmennummer haben die Fahrräder eine fortlaufende interne Nummer. Die Fahrräder sind in verschiedenen Rad-Größen und Rahmenfarben verfügbar. Es gibt Fahrräder extra für Frauen und Männer sowie Unisex-Versionen. Aus den umliegenden Hotels leihen sich die Gäste die Fahrräder immer für ein, zwei Tage aus. Einige Gäste legen Wert auf die gleiche Rahmenfarbe. Die Abrechnung erfolgt auf der Basis der Ausleihdauer in Tagen und der gewählten Fahrrad-Art (normal oder eBike). Die Hotelgäste bezahlen über ihre Hotel-Rechnung (Aufbuchung auf ihre Zimmer-Rechnung). Die Hotel's überweisen die Beträge direkt auf ein Konto. Erstellen Sie auf der Basis eines Entity-Relationship-Diagramms eine Datenbank-Struktur, um den beschriebenen Sachverhalt in einer Datenbank zu modellieren!*
3. *Eine Schüler-Firma will Nachhilfe vermitteln! Überlegen Sie sich, welche Daten für eine sinnvolle Datenverarbeitung in einer Datenbank erfasst werden müssten. Erstellen Sie ein passendes ER-Diagramm!*

3.1.3. physische Phase – Umsetzung / Implementierung in ein DBS

Jetzt kommen wir zur Qual der Wahl: Mit welchem Datenbank-System wollen wir unser(e) Projekt(e) umsetzen?
In der Praxis sind die Wahl-Möglichkeiten vor allem von den technischen und finanziellen Aspekten geprägt.
In der Schule werden uns nur einige wenige Systeme zu Verfügung stehen. Da in der Schule die finanziellen Ressourcen eher mager gestrickt sind, orientieren wir hier vorrangig auf kostenfrei zu habene Software. Und diese kann sich in vielen Punkten gut gegen Kostenpflichtige Systeme behaupten.
Die Empfehlung geht in Richtung SQLite-Studio, welches weiter hinten vorgestellt wird (→ 3.1.7.1.3. Arbeiten mit dem SQLiteStudio).
Für die Anwender klassischer Office-Suiten sind es Libre-Office bzw. OpenOffice, die in die engere Wahl genommen werden sollten. In ihnen findet sich z.B. das Programm BASE, was viele Aspekte von microsoft ® ACCESS ® mitbringt. BASE besprechen wir im Kapitel → 3.1.5. Datenbanken mit BASE.
Alle empfohlenen Programme / Suiten sind kostenfrei und mit freien Lizenzen im Internet zu haben. Sie sind auch auf dem IoStick installiert.
Da ACCESS quasi ein Standard im Office-Bereich darstellt, werden wir auch die Umsetzung einer Datenbank vorstellen (→ 3.1.6. Datenbanken mit ACCESS).
Alle Vorstellungen verwenden das gleiche Daten-Modell. Es werden aber Programm-spezifische Unterschiede bei der Reihenfolge und den vorgestellten Programm-Leistungen gemacht.
Windows ® ist unser bevorzugtes Betriebssystem. Ab und zu werden wir auch auf die Möglichkeiten von Datenbank-Nutzungen auf dem Raspberry Pi eingehen. Gerade SQLite lässt sich hier hervorragend nutzen (→ 3.1.7.0.1. SQLite auf einem Raspberry Pi). Der kleine Rechner ist und bleibt die Geheimwaffe der Schul-Informatik.

physische Phase ist durch Entwicklung einer leeren Datenbank geprägt
aus dem logischen Modell wird durch die Nutzung einer Daten-Definitionssprache (DDL; z.B. SQL)

Aufgaben:

1. ***Erstellen Sie eine Skizze, in der Sie die Nutzer-Oberfläche (Eingabe-Bereiche und Bedien-Elemente) einer Adressbuch-Applikation (z.B. für ein Smartphone oder ein Tablet) mit Name, Vorname sowie mindestens fünf weiteren Attributen planen! (Für ev. Unter-Fenster etc. geben Sie weitere Skizzen an!)***

Aufgaben:

1. Stellen Sie die Attribute (Farbe, Türenanzahl, Motorstärke, Modellname, Preis) für die Autos eines Autohändlers als Entity-Relationship-Diagramm dar!
2. Setzen Sie das Diagramm von 1. in einem Zeichen-Programm (z.B. LibreOffice- bzw. OpenOffice-Draw) um! Die Datei soll "ERD_Autohändler_V1" heißen!
3. Bestimmen Sie die Kardinalitäten der nachfolgenden Beziehungen! Erläutern Sie Ihre Entscheidung!

a)	Klasse	-----	hat	-----	Schüler
b)	Elternpaar	-----	hat	-----	Kind
c)	Frau	-----	heiratet	-----	Mann
d)	Frau	-----	kauft	-----	DesignerSchuh
e)	Schüler	-----	erhält	-----	Zeugnis
f)	Künstler	-----	malt / hat gemalt	-----	Bild
g)	Student	-----	besucht	-----	Kurs
h)	Schüler	-----	ist_befreundet_mit	-----	Schüler
i)	Leser	-----	leiht_aus	-----	Buch
j)	Schüler	-----	hat	-----	Zeugnis
k)	Kind	-----	hat	-----	Elter

4. Eine Video-DVD-Ausleihe möchte ihre Videos und DVDs sowie die Kunden und Ausleihen in einer Datenbank erfassen und verwalten. Wandeln Sie das ERD von Aufgabe 3 der letzten Aufgaben-Gruppe (Zimmer-Vermietung) passend ab und erstellen Sie eine passende Skizze am Computer (z.B. mit LibreOffice Draw)!
5. Ein Fitness-Studio möchte seine Mitglieder, Kurse und Kursleiter in einer Datenbank verwalten. Erstellen Sie gemeinsam an der Tafel / am Whiteboard ein ERD für eine praktikable Datenbank!
6. Die Rechtsanwalts-Gemeinschaft "Dr. Schröder, Schröder und Schmittchen" will ihre Termine verwalten. Als notwendige Angaben haben sich weiterhin herausgestellt:
   - die Mandanten mit Name (können als Kläger oder Beklagte auftauchen), ev. Rechtsschutzversicherung, Anschrift, eMail-Kontakt, Telefonnummer
   - Fall, zuständiges Gericht, Aktenzeichen
   - Termin (Datum, Uhrzeit), Ort, Zweck, Dauer

   Erstellen Sie eine ERD am Computer! Wenn Sie weitere Daten als notwendig erachten, dann können Sie diese ergänzen.
7. Für eine kleine lokale Verleihstation von Elektro-Autos soll eine kleine Verwaltungs-Datenbank erstellt werden. Analysieren Sie die praktische Situation und erstellen Sie ein ERD als Skizze auf einem Blatt Papier!
   Diskutieren Sie die verschiedenen Vorschläge innerhalb des Kurses!
   Erstellen Sie ein gemeinschaftlich akzeptiertes Diagramm am Computer!
8. Die nachmittäglichen Arbeitsgemeinschaften, Sportgruppen usw. einer Schule sollen in einer Datenbank (für den Koordinator / verantwortlichen Lehrer / die Schulleitung) verwaltet werden. Erstellen Sie ein ERD zu dieser Situation am Computer!

9. Übernehmen Sie folgende Situations-Beschreibung in ein Dokument (microsoft Word oder LibreOffice Writer)! Als Überschrift nehmen Sie bitte: "Machbarkeits-Analyse Wohnungs-Genossenschaft". Setzen Sie sich als Bearbeiter darunter.

Eine Wohnungs-Genossenschaft hat mehrere Häuser mit jeweils unterschiedlich vielen Wohnungen und unterschiedlicher Größe. Zwei Hausmeister betreuen die Wohnungen häuserweise.

Von den Mietern sollen in der Wohnungs-Verwaltung nur die Nachnamen und Vornamen der Haupt-Mieter (Ansprechpartner) erfasst werden!

Da die Menge an Häusern und Wohnungen das Mass für einen kleinen Karteikasten mitlerweile überschreitet und die Kommunikation über Computer-Dokumente erfolgen soll, überlegt die Wohnungs-Genossenschaft eine Datenbank anzuschaffen. Weiterhin soll auf einer Tafel im Verwaltungs-Gebäude eine Übersicht zu sehen sein, welche Wohungen frei sind, wo und wie sie liegen, wie groß sie sind und was sie kostet.

Zuerst soll nur die Umsetzbarkeit und der inhaltliche Aufwand geprüft werden.

a) Überlegen Sie sich, welche Daten für eine Umsetzung des Sachverhaltes "Wohnungs-Verwaltung" erfasst werden müssten! Klassifizieren Sie die Merkmale / Daten nach Wichtigkeit in drei Gruppen (1 … "unbedingt notwendig"; 2 … "wichtig"; 3 … "verzichtbar"!

b) Erstellen Sie mit Hilfe eines Zeichen-Programms (z.B. LibreOffice Draw) ein geeignetes ERD! Bauen Sie das ERD in das Dokument ein! Das Dokument darf nicht größer als 1 Seite (A4) werden.

c) Erläutern Sie in kurzen Texten, warum Sie die Merkmale den bestimmten Gruppen zugeordnet haben!

d) Drucken Sie das Dokument 1x aus und tauschen Sie es innerhalb des Kurses im Rotations-Verfahren (immer über zwei Teilnehmer weiterreichen) aus!

e) Analysieren Sie den vorliegenden fremden Vorschlag und notieren Sie in einem Dokument Fragen, Widersprüche usw.! Machen Sie mit Bleistift Fragezeichen an die Stellen im Ausdruck der "Machbarkeits-Analyse"! Geben Sie dann die Analyse mit Ihrem ausgedruckten Fragen usw. zurück!

f) Setzen Sie sich mit den aufgeworfenen Fragen usw. auseinander und verbessern Sie u.U. Ihre Analyse!

3.1.4. Aufbau klassischer relationaler Datenbanken

manchmal als Objekt-Typen bezeichnete unterschiedliche Teile / Nutzungs-Prinzipien einer Datenbank

Datenbanken-Bestandteile

- **Tabellen** — Datenbasis in Tabellen (Spalten-Zeilen-Anordnung)
  echte Daten(-Tabellen)
- **Abfragen** — temporäre Tabellen durch Umordnung, Sortierung, Gruppierung und Filterung der echten Daten (aus einer oder mehrerer "Tabellen"
  praktisch sind nur die Vorschriften gespeichert, die dann bei Nutzung aufgerufen und ausgeführt werden
- **Berichte** — Zusammenfassung der Daten für eine bestimmte Datenbank-Situation
  praktisch sind nur die Auswerte-Vorschriften und –Kriterien gespeichert; sie werden bei Bedarf aufgerufen und ausgeführt und produzieren ein abspeicherbares bzw. ausdruckbares Dokument
- **Formulare** — dienen der Benutzer-freundlichen Eingabe und Darstellung der Daten (es können Daten aus mehreren Tabellen oder Abfragen benutzt werden; eingegebene Daten werden automatisch richtig den betreffenden Tabellen zugeordnet)
- **Makros / Programme** — dienen der Automatisierung von Nutzer-Aktionen, realisieren Kontrollen und spezielle Daten-Umsetzungen (z.B. Importe od. Exporte) usw. usf.
  es besteht die Möglichkeit der Erstellung einer scheinbar (vom Datenbank-System) unabhängigen Applikation

die ersten vier werden als klassische Bestandteile für Datenbanken a'la ACCESS verstanden
die meisten enthalten auch Programmier-Sprachen für die Erzeugung von Makros bzw. Applikationen
die ersten drei sind praktisch in allen Datenbanken und Datenbank-Systemen vorhanden

3.1.4.1. Tabellen

reine Datensammlungen mit fester Struktur
deshalb sehr schnell

besonders wichtig ist es, auf die Definition des Primär-Schlüssels einer jeden Tabelle zu achten
fehlt dieser in der erarbeiteten Tabelle, dann erstellt das System meist selbstständigen einen Primär-Schlüssel als neue Spalte
ev. sind Tabellen-Struktur-Korrekturen notwendig oder die gesamte Planung (ERD, Zusammenhänge, …) muss angepasst werden
Tabellerische Daten sind aber auch erst einmal als erste "Datenbank" ok. Natürlich muss dann schnell eine Optimierung erfolgen (→ Normalisierung (→ 2.2.1. Normalisierung)), damit wir dann auch wirklich einen Datenbank mit all ihren Leistungen und Vorteilen zur verfügung haben.
Will man lediglich die Daten in einer Tabelle halten und bearbeiten, dann reicht vielleicht auch EXCEL® (von microsoft ®) bzw. CALC aus Open- oder Libre-Office. Diese bieten erste "Datenbank"-Funktionen an. Für komplexe Datenstrukturen sind Tabellenkalkulationen aber nicht geeignet.

Um den verfügbaren Speicherplatz optimal zu nutzen müssen die Daten in geeigneten Formaten gespeichert werden. Auch die weitere Verwendung spielt bei der Auswahl von Datenformaten eine Rolle. Zahlen, die als Text – also Zeichenfolgen (vielleicht sogar noch mit Leerzeichen usw. usf.) – gespeichert sind, eignen sich kaum zum Verrechnen.
Ständiges Neuformatieren, Umrechnen usw. beinhaltet immer ein großes Fehler-Potential. Schließlich müssen oft auch wieder Ergebnisse in die "seltsamen" Formate zurück übertragen werden.
In der nächsten Tabelle sind diverse Daten-Formate zusammengestellt. Ev. muss man aber die eigenen System-Bedingungen beachten. Nicht jedes System versteht ein Format so wie vielleicht die Mehrheit oder die Standards. Gerade proprietäre System sind da sehr erfinderisch, um Kunden mit ihren unübertragbaren Formaten Kunden an sich zu binden.

Definition(en): NULL-Wert
Ein NULL-Wert ist ein spezieller Eintrag für ein Attribut. Es charakterisiert einen leeren Eintrag. Ein NULL-Wert entspricht **nicht** dem Zahlenwert einer Null.

Daten-Typen

Bezeichnung	Werte-Bereich (Domäne)	Name in SQL	Erläuterung	Speicher-Bedarf	Name in BASE	Name in ACCESS		
ganze Zahl	0 .. 255			1 Byte		Byte		
	-32'768 .. 32'767	SMALLINT		2 Byte		Integer		
	-2'147'483'648 .. 2'147'483'647	INTEGER		4 Byte		Longinteger		
Autowert Zähler				4 – 16 Byte		Auto		
	1, 2, 3 ...			16 Byte		Replication ID		
reele Zahl		DECIMAL (s,n)	Fließkomma-Zahl mit mindestens **s** Stellen (insgesamt) und davon **n** Nachkommastellen					
		NUMERIC (s,n)	Fließkomma-Zahl mit genau **s** Stellen (insgesamt) und davon **n** Nachkommastellen					
		FLOAT (n)						
	$-3{,}403 * 10^{38}$ .. $3{,}03 * 10^{38}$		Genauigkeit: 10^{-7}	4 Byte		Single		
	$-1{,}798 * 10^{308}$ .. $1{,}798 * 10^{308}$		Genauigkeit: 10^{-15}	8 Byte		Double		
Währung					Währung			
logischer Wert			True oder False Wahr oder Falsch Ja oder Nein	1 bit (1 Byte)				
Zeichenkette		CHAR (n)						
					Text	Text		
variable Zeichenkette		VARCHAR (n)						
			variabler Text	→ 64'000 Byte		Memo		
Datum		DATE			Datum	Datum		
Uhrzeit					Zeit	Zeit		

eingebettete Objekte			z.B. Bilder, Graphiken, Videos, Musik	→ 1 GB		OLE-Objekt		

3.1.4.2. Abfragen

In den wenigsten Datenbanken dürfen alle Nutzer auch auf alle Daten zugreifen. Vielfach macht es auch keinen Sinn alle Daten einer Tabelle mit einem Mal darzustellen. Z.B. darf nicht jeder Nutzer einer Firmen-Datenbank die Löhne einsehen oder sich in einem online-shop die Kreditkarten-Daten anzeigen lassen.
Um die Auswahl der Spalten und Zeilen für den Nutzer bzw. seinem Bedarf entsprechend passgerecht zu machen, werden **Abfrage**n benutzt. Andere Namen für Abfragen sind **View**'s oder **Sicht**en.
Praktisch unterscheiden wir die folgenden Arten von Abfragen:

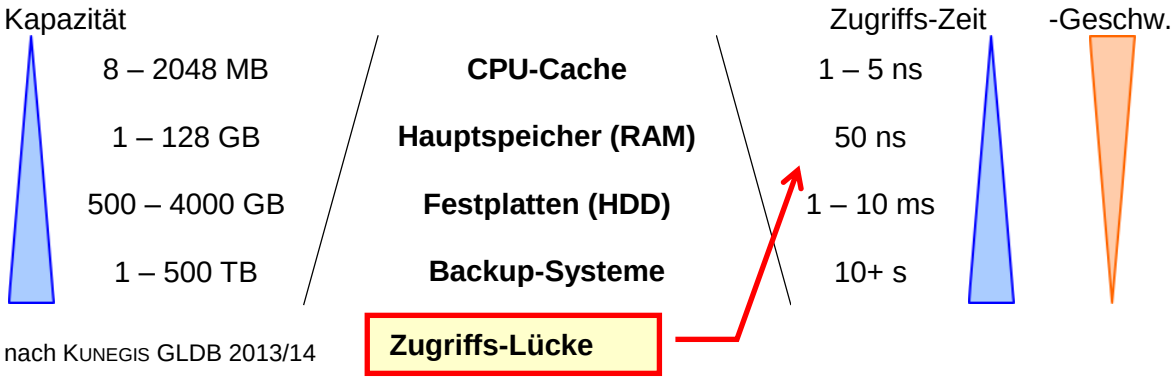
- **Auswahl-Abfragen** — ermöglicht die eingeschränkte Anzeige bzw. Tabellierung von Inhalten aus einer oder mehrerer Tabellen bzw. Abfragen
  praktisch wird eine temporäre Auswahl von Spalten (Felder, Attributen) und Zeilen (Datensätzen) aus den Tabellen u. / od. Abfragen erzeugt
- **Auswahl-Abfragen mit Auswertungen (bzw. Berichts-Charakter)** — neben den Beschränkungen (Auswahl) hinsichtlich der Zeilen (Datensätze) und Spalten (Felder, Attribute) können bestimmte Feldwerte gezählt, summiert sowie Minimum und Maximum bestimmt werden
  innerhalb von Zeilen lassen sich auch Felder berechnen (z.B. "volljährig" aus Geburtsdatum und heutigem Datum)
- **Parameter-Abfragen** — hierbei ist die nachfolgende Abfrage von einer oder mehrerer zu tätigen Eingaben (- den Parametern -) abhängig; sie stellen die Möglichkeit dar, flexible Abfragen zu erstellen / nutzen

Von der Art und Weise der Erstellung unterscheiden sich diese Abfrage-Arten recht wenig, so dass wir hier nicht so einen Wert auf die Unterschiede legen.
Manchmal wird auch noch nach den folgenden Abfrage-Arten unterschieden:

- **Erstellungs-Abfragen**
- **Anfüge-Abfragen**
- **Lösch-Abfragen**
- **Aktualisierungs-Abfragen**

3.1.4.3. Indicies

Tabellen lassen sich praktisch nur nach einem Attribut ordnen / sortieren. Natürlich sind untergeordnete (sekundäre) Ordnungen / Sortierungen möglich.
Häufig brauch man abe auch schnelle Zugriffe auf andere Attribute. Liegen diese ungeordnet vor, ist der Zugriff meist sehr aufwändig.
Mit einem Index erzeugt man sich eine geordnet Hilfs-Tabelle, die neben dem geordneten Wert nur noch den Primärschlüssel aus der Daten-Tabelle enthält. Wenn es notwendig und sinnvoll ist, kann man sich für jedes Attribut – und praktisch auch für Attribut-Kombinationen – einzelne Indicies anlegen.
Die eigentlich Daten-Tabelle wird von Aktionen auf den Indicies nicht im geringsten beeinflusst. Die Indicies sind quasi Schnell-Zugriffs-Listen zu Daten-Tabelle. Da die Indicies nur sehr kleine Tabellen sind, können sie häufiger aktualisert und optimiert werden.
Ein weiterer Grund für die Nutzung von Index-Tabellen ist die sogenannte Zugriffs-Lücke bei Datenspeichern. Sie erstreckt sich über rund fünf Zehner-Potenzen.



Festplatten sind für ein ständiges Daten-Hin-und-her-Schaufeln zu langsam. Schließlich muss ja die superschnelle CPU mit Daten versorgt werden. Hauptspeicher ist zwar deutlich schneller, aber eben nicht als dauerhafter Speiche geeignet. RAM ist auch relativ Kostenintensiv.
Der Kompromiß ist eben eine leichte (kleine) Datenstruktur im Speicher und schnellere ZUgriffe auf die Festplatten-Daten über Indicies.
Durch SSD-Festplatten (Solid Disk) wird die Lücke nur geringfügig ausgeglichen. Praktisch kommte es zur Verbesserung um eine Zehner-Potnz. SSD's sind aber auch noch relativ teuer und weisen noch relativ kleine Speicher-Kapazitäten auf. In den kommenden Jahren wird sich dies aber sehr schnell ändern.

3.1.4.4. Berichte

enthalten Zusammenfassungen, Gruppierungen und Berechnungen
sonst praktisch den Sichten und Tabellen sehr ähnlich
in Berichten können die elementaren Daten auch fehlen (für die Erstellung des Berichts werden sie natürlich benutzt, aber im Bericht selbst nicht angezeigt)

häufig in besondere Druck-Formen gepresset

es gibt hier spezielle Software, die ausgehend von beliebigen Datenbanken sich nur um die Erstellung geeigneter / spezieller Berichte kümmert (z.B. CrystalReports)
der Datenbank-Zugriff erfolgt über spezielle Schnittstellen (→ 6.2. universelle Datenbank-Schnittstellen)

3.1.4.5. Formulare

nicht in allen Datenbank-System enthalten
hier ist dann spezielle Anwender-Software notwendig, die über die klassischen Schnittstellen (→ 6.2. universelle Datenbank-Schnittstellen) auf die Datenbank zugreift

wenn Formulare im DBMS machbar sind, dann sind diese meist sehr System-spezifisch und kaum auf neuere / andere Datenbanken (selbst mit gleicher Struktur) übertragbar

bilden wesentliche Bausteine für programmierte Datenbanken (→ 6. Datenbanken - Programmierung)

3.1.4.5. Makros / Programmierung

Makro's sind aufgezeichnete oder programmierte Arbeits-Schritte, die über Tastatur-Kürzel, spezielle Schaltflächen oder andere Makro's bzw. Programme aufgerufen werden können
gehören nicht zu klassischen DBMS

eignen sich besonders dazu – bestimmte, sich häufig wiederholende Tastatur-Befehle oder Maus-Aktivitäten od.ä. – als Gesamtheit automatisch ausführen zu lassen

benutzen zumeist eine DBMS-eigene Programmiersprache
oft von anderen Programmiersprachen abgeleitet oder Teil-Klassen großer Programmiersprachen

typische Phasen bei der Makro-Nutzung durch Anwender:

0. Phase: einfache Makro-Nutzung

vorhandene Makro's werden – z.T. ohne Kenntnis darüber, dass es sich um solche handelt – durch Vorgaben / Programm-Beschreibungen usw. usf. genutzt

1. Phase: reine Makro-Erstellung und -Nutzung

Aufzeichnen der Nutzer-Aktivitäten unter einem (Makro-)Namen und Zuordnung einer Aufruf-Möglichkeit (Name, Tasten-Kürzel, Schaltfläche)

2. Phase: Ansicht des Quell-Codes und kleine Änderungen

aufgezeichnete Makro's im Quellcode ansehen und z.B. unnötige Befehle herauslöschen oder bestimmte Befehle umgruppieren oder z.B. Durchlaufzahlen bei Schleifen verändern

3. Phase: Programmierung von Makro's

Erstellen von speziellen Makro's / Funktionen, die spezielle Aufgaben übernehmen
damit sind meist solche gemeint, die nicht über die Programm-Oberfläche realisierbar sind oder nur mit einem extrem hohen Aufwand

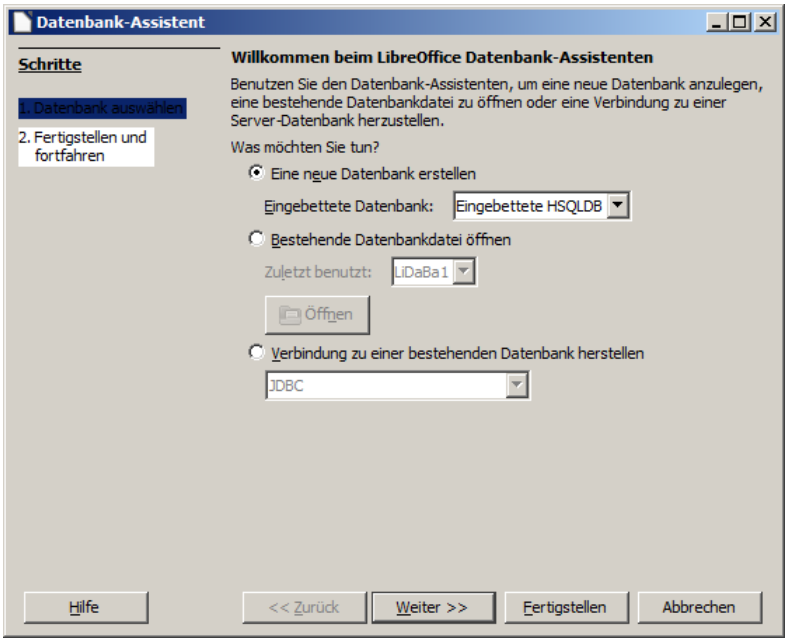
Es folgt dann meist die echte Programmierung von Datenbanken bzw. Programmen, in deren Mittelpunkt echte Datenbanken stehen.

3.1.5. Datenbanken mit BASE

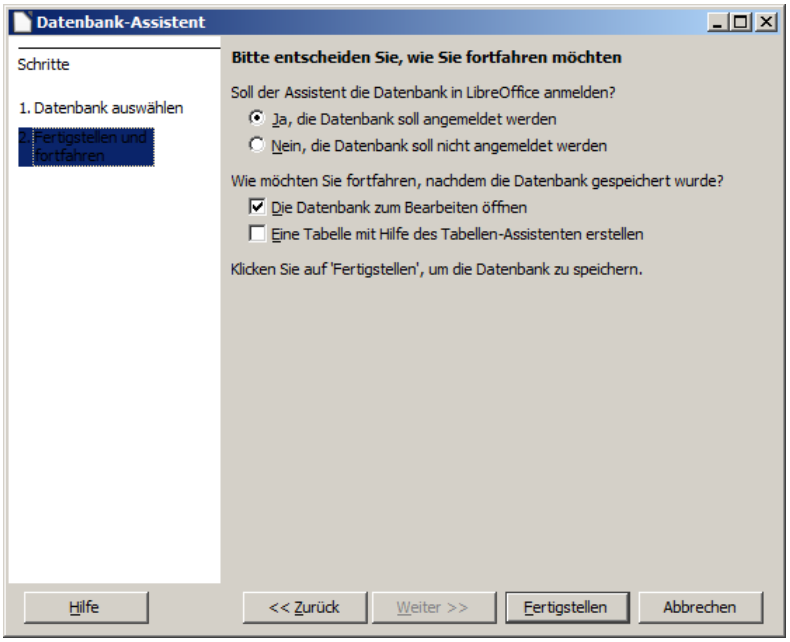
Die Nutzung von Datenbank-Komponenten aus Office-Produkten wird im neuen Rahmenplan "Informatik" (Sek.II; für MV) abgelehnt.
Für Einsteiger-Projekte sind sie aber sehr gut geeignet!

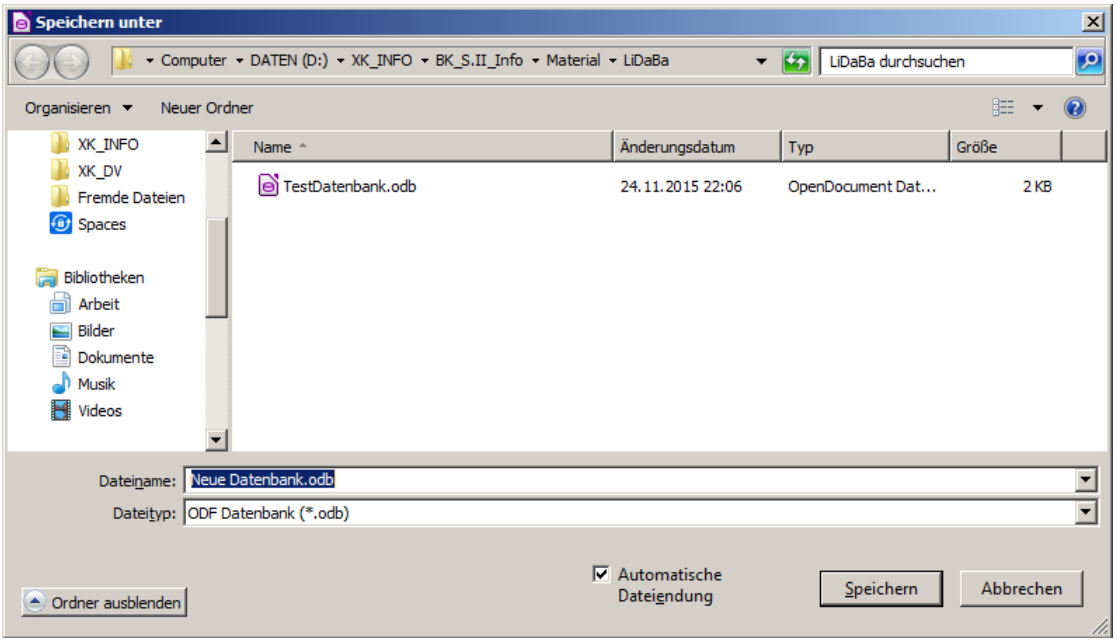
Gemeint sind hier die Datenbank-Systeme aus den Schwester-Programmen Libre- und Open-Office. Beide enthalten als Datenbank-Bestandteil das Datenbank-Management-System (DBMS) BASE.
Viele Arbeiten / Assistenten / … arbeiten sind sehr ähnlich zu microsoft ACCESS (bis Version 2003; → 3.1.6. Datenbanken mit ACCESS) – dem Standard-Programm für Desktop-Datenbanken (kann als Vorlage für BASE angesehen werden).
Üblicherweise ist die Arbeitsgrundlage für die Implementierung einer Datenbank ein Entity-Relationship-Diagramm (ERD).
Da BASE über ein Assistenten-System verfügt, wollen wir anhand diesem die Erstellung einer Datenbank aufzeigen. Später folgen dann auch die etwas professionelleren Erstellungs-Möglichkeiten (→ 3.1.5.1.2. Erstellen einer Tabelle in der Entwurfs-Ansicht)
Start mit Assistenten zum Erstellen oder Auswählen (Öffnen) einer Datenbank. Das ist praktisch eine Container-Datei, in der dann alle Bestandteile gespeichert werden.

Erstellen einer neuen Datenbank, das Öffnen einer vorhandenen Datenbank oder die Kontakt-Aufnahme zu einer externen Datenbank.
Im letzteren Fall wird meist eine Schatten-Datenbank angelegt, die quasi dem Original entspricht, aber BASE-typisch dargestellt wird.



"Anmeldung" bedeutet hier, dass andere Programme – hier besonders WRITER und CALC – relativ einfach auf die Daten der neuen Datenbank zugreifen können. Das ist z.B. für die Erstellung von Serienbriefen sinnvoll.
Es geht aber auch ohne diese vorherige Anmeldung.





Zu beachten ist hier, dass ein Überspeichern nicht möglich ist, es muss also immer ein neuer Dateiname vergeben werden.

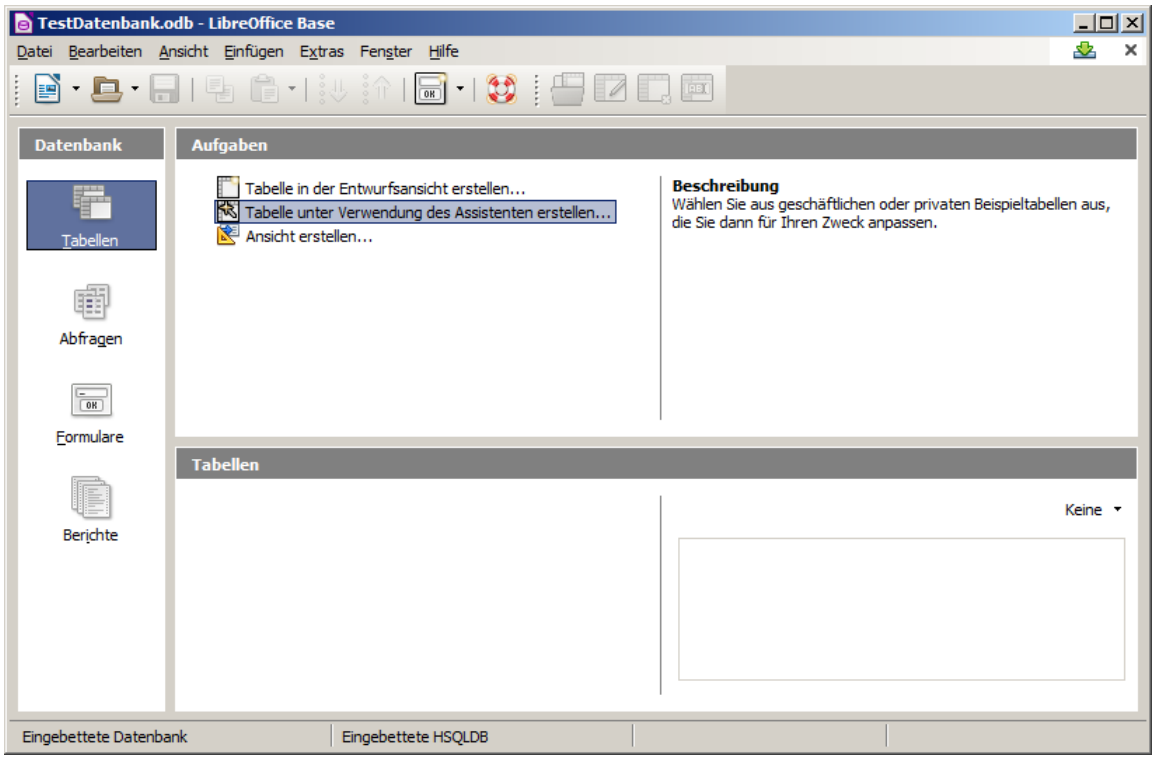
3.1.5.1. Tabellen

Tabellen lassen sich in BASE auf unterschiedliche Art und Weisen erstellen. Da wäre zum Ersten die Möglichkeit einen Assistenten zu nutzen. Das werden wir auch zuerst einmal so tun (→ 3.1.5.1.1. Erstellen einer Tabelle mit Hilfe des Assistenten). Die Assistenten eignen sich vor allem für Anfänger und bedacht arbeitende Nutzer, da man innerhalb des Assistenten-Dialoges ständig "vor" oder "zurück" gehen kann. Mögliche Fehler lassen sich bis zum

"Fertigstellen" immer noch korrigieren. Ein weiterer Einsatzfall für Assistenten sind die klassischen Dtenbank-Anwendungen, wie z.B. eine Kontakt-Datenbank. Da spart man sich eine Menge Tipp-Arbeit, da die Elemente meist nur ausgewählt werden müssen.
Zum Zweiten kann man Tabellen in der sogenannten Entwurfs-Ansicht erstellen (→ 3.1.5.1.2. Erstellen einer Tabelle in der Entwurfs-Ansicht). Das ist praktisch ein Tabellen-Struktur-Editor. Hier legen wir die Tabellen und ihre Struktur an. Das Befüllen mit Daten erfolgt dann in der "Tabellen-Ansicht" (→ 3.1.5.1.3. Eingeben von Daten in eine Tabelle).

3.1.5.1.1. Erstellen einer Tabelle mit Hilfe des Assistenten

Bei einer geöffneten Datenbank sehen wir im Bereich "Tabellen" die möglich "Aufgaben", die wir erledigen können.



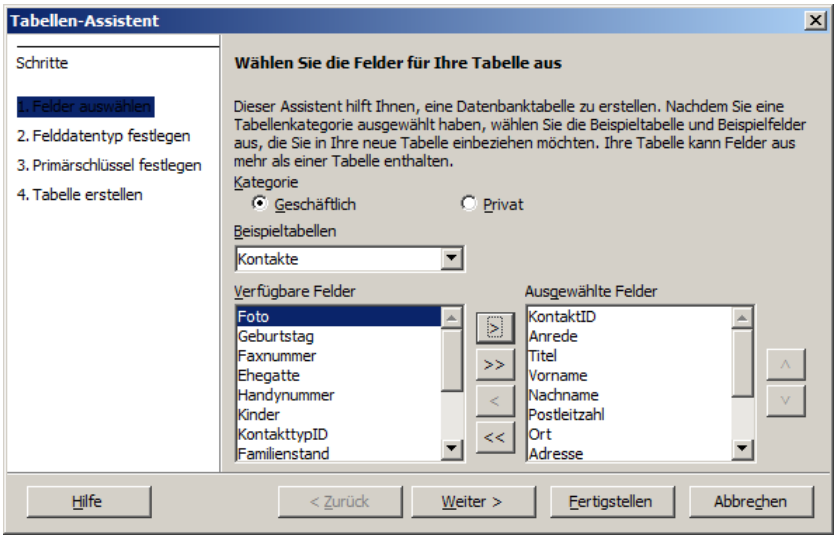
Für uns ist nun die Aufgabe "Tabelle unter Verwendung des Assistenten erstellen …" interessant.

Im ersten Schritt wählen eine möglichst geeignete Beispieltabelle aus. Da unten die dann verfügbaren Felder (Attribute) gleich angezeigt werden, fällt uns die Wahl meist leicht.
Es gibt zwei große gedachte Bereiche, einmal die geschäftlichen und dann die privaten Datenbanken.
Fehlen Felder, dann kann man sie später immer noch ergänzen.

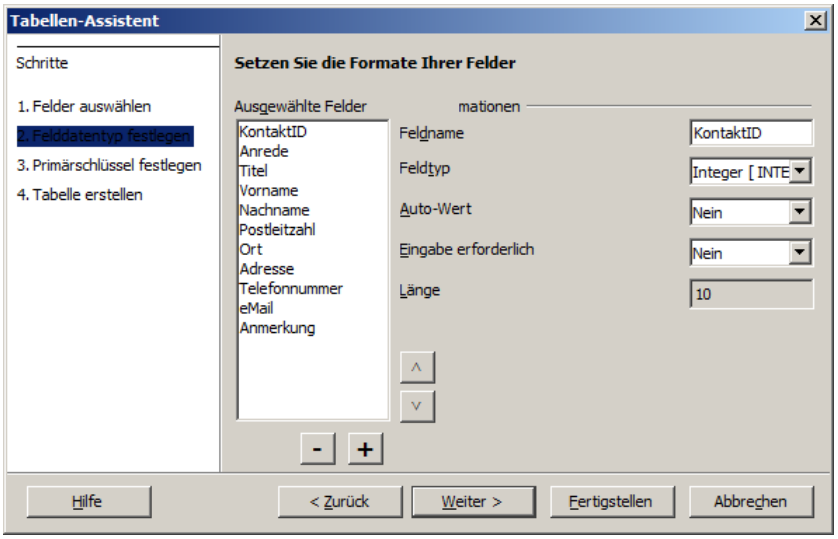
Es folgt das Zusammenstellen der gebrauchten Spalten (Attribute).

Wichtig ist hier die Fortsetzung mit "Weiter >"!

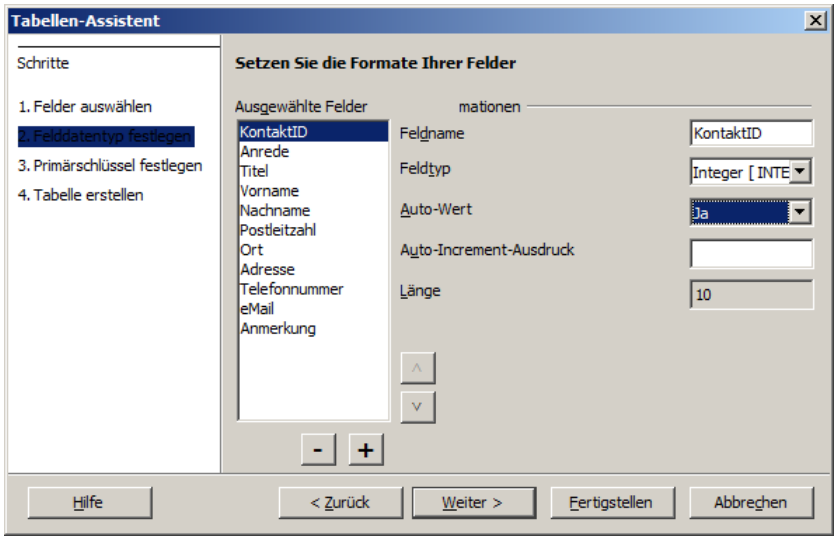
"Fertigstellen" nur dann, wenn Details, wie z.B der Primär-Schlüssel für die Datenbank-Nutzung uninteressant sind.



Der zweite Assitenten-Schritt ermöglicht uns jetzt etwas Fein-Tuning hinsichtlich der einzelnen Felder.
Für die meisten Felder sind Änderungen nicht notwendig.
Profi's können hier ihre Datenbank zumindestens hinsichtlich der Daten-Größe optimieren.



Für den zukünftigen Primär-Schlüssel unserer Tabelle nehmen wir aber noch einige Änderungen vor.
So setzen wir "Auto-Wert" auf "Ja". Dadurch wird die Vergabe der KontaktID als Schlüsselwert automatisiert. Wollen wir eigene ID's eingeben, dann belassen wir die Einstellung auf "Nein".

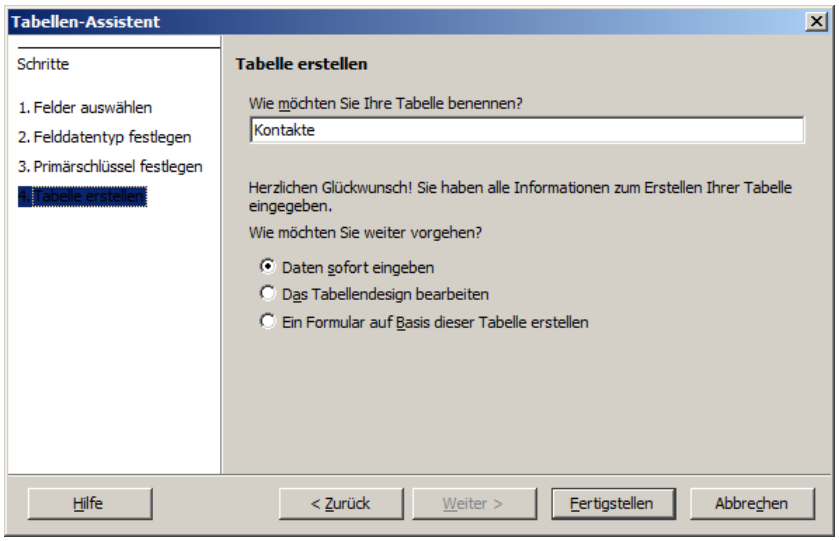


Der dritte Assistenten-Schritt ist ganz für die Festlegung des Primär-Schlüssels der Tabelle gedacht. Wir haben hier die Möglichkeit eine neue Spalte mit einem neuen Primärschlüssel hinzuzufügen. Die Auto-wert-Funktion haben wir ja schon besprochen.
Die zweite Möglichkeit ist die Nutzung einer vorhandenen Feldes als Primär-Schlüssel.

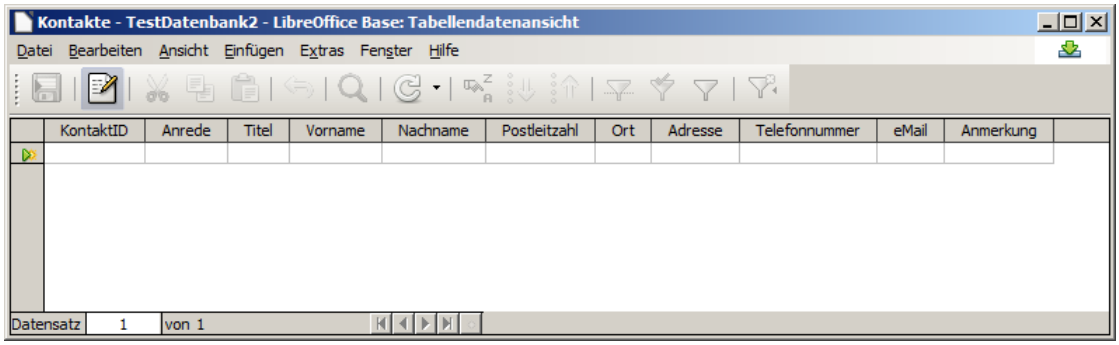
Und zu guter Letzt kann ein Primär-Schlüssel auch aus mehreren vorhanden Feldern zusammengesetzt werden.
Da wir schon eine ID-Spalte geplant haben, nutzen wir natürlich die zweite Option und wählen in dem Auswahl-Feld den richtigen Feld-Namen aus.
Eine Erstellung des Autowertes geben wir ebenfalls an.
Da wir dass schon weiter vorne festgelegt haben, kann die Eintragung hier eigentlich übergangen werden, aber sicher ist sicher.

Es folgt das Abspeichern der Tabellen-Definition innerhalb der Datenbank-Datei (eingekapselt).

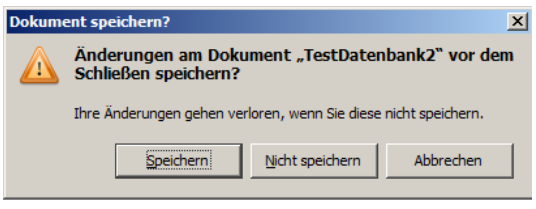
Er will, kann jetzt sofort zum Eingeben der Daten wechseln (s.a. → 3.1.5.1.3. Eingeben von Daten in eine Tabelle).
Nacharbeiten für besondere Konstrukte sind über die zweite Vorgehens-Option möglich. Dazu findet man noch ein paar Hinweise im Kapitel → 3.1.5.1.3. Ändern des Relationsschema einer Tabelle – Korrekturen am Entwurf.

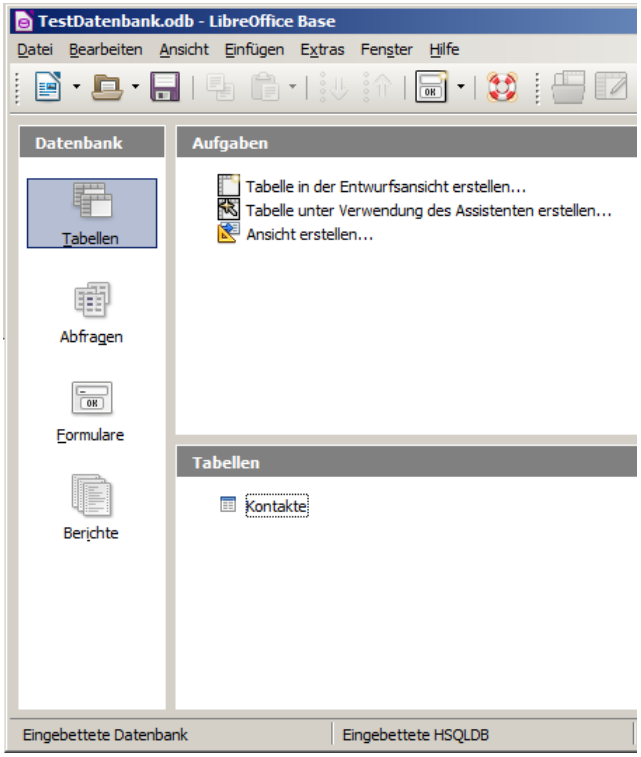


Wählt man "Daten sofort eingeben", dann sieht man die konstruierte Tabelle in ihrer ganzen Pracht – nur eben ohne Daten.
Liegen die Daten der Tabelle in Papier-Form vor, dann bleibt i.A. nur der händische Weg der Daten-Eingabe.



Die gleiche Ansicht erhält man, wenn im Datenbank-Menü auf die betreffende Tabelle klickt.
Es gibt verschiedene Möglichkeiten Daten aus strukturierten Dateien zu importieren. Dazu eignen sich CSV-, TXT-, XML- und JSON-Dateien.
Obwohl wir vielleicht noch gar keine Daten eingegeben haben, werden wir am Schluss zum Abspeichern aufgefordert. Dieses Speichern bezieht sich auch gar nicht auf Daten – wir werden noch sehen darum müssen wir uns überhaupt nicht mehr kümmern – sondern auf die vorgenommenen Definitionen sowie die Format-Einstellungen und -Veränderungen in unserer Datenbank.



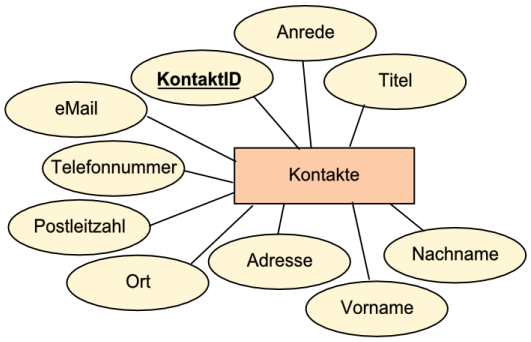


Aufgaben:

1. ***Vollziehen Sie den Weg zur Erstellung einer Tabelle mit dem Assistenten nach!***
2. ***Erstellen Sie mit Hilfe der Assistenten in einer neuen Datenbank ("CD-Sammlung") eine Tabelle für Ihre eigene CD-, Album-/MP3-Sammlung! (Benutzen Sie mindestens 7 Attribute!)***

3.1.5.1.1.1. Rück-Ableitung eines Entity-Relationship-Diagramms

Wenn man nun so eine Tabelle mit dem Assistenen erstellt hat oder auch vielleicht von wo anders her bezogen hat (z.B. aus einer externen Datenbank), dann benötigt man vielleicht für eigene Weiterentwicklungen wieder das ERD. Die Ableitung ist denkbar einfach. Der Tabellenname ist der Entitäts-Typ. Die Spalten-Überschriften entsprechen dem Relationsschema und stellen somit die Attribute dar. Das Schlüssel-Attribut ist ebenfals schnell aus dem Primär-Schlüssel abgeleitet.

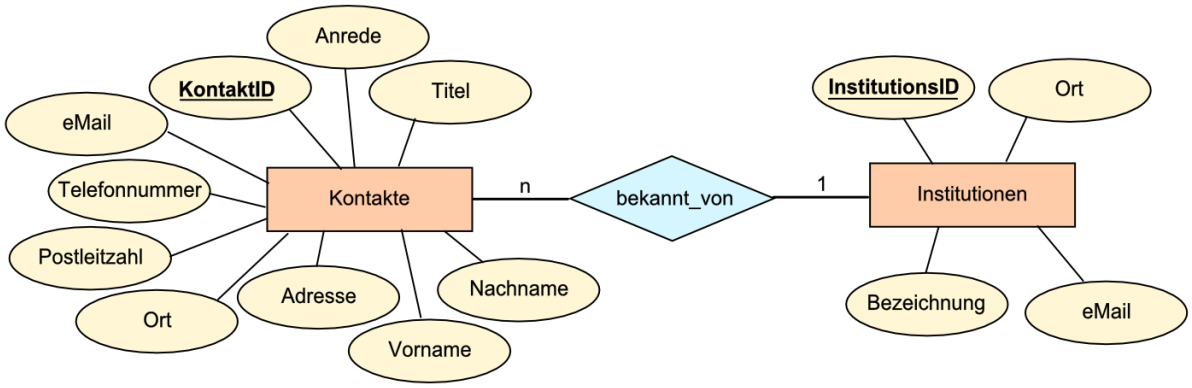


3.1.5.1.2. Erstellen einer Tabelle in der Entwurfs-Ansicht

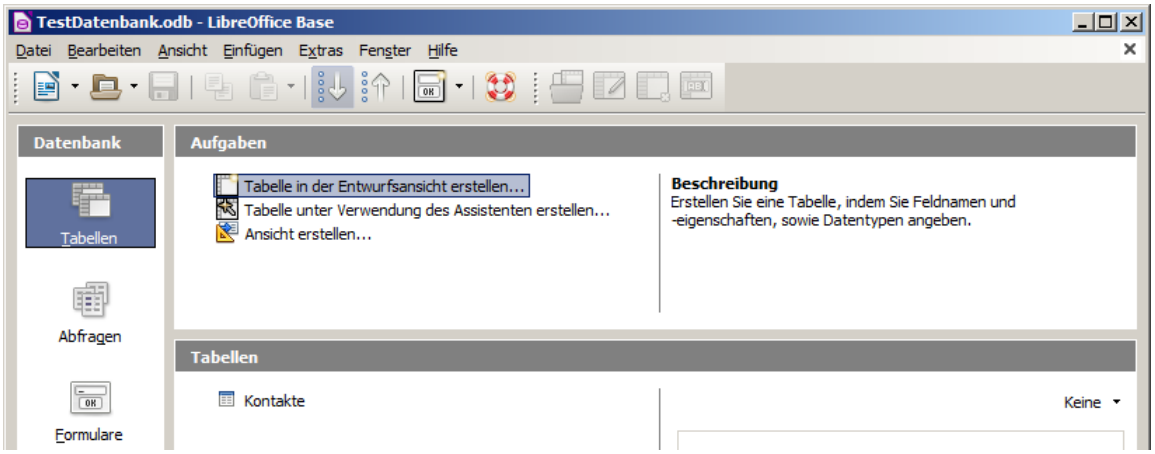
Der Assistent bietet aber nur begrenzte Möglichkeiten. Wir tun hier mal so, als könnten wir die nächste Entwicklungsstufe unserer Beispiel-Datenbank nicht mit dem Tabellen-Assistenten realisieren.
Gewünscht soll eine Erweiterung der Kontakte-Datenbank sein, in der die Institutionen (Schule, Verein, …) für die Kontakte gespeichert sind.

böse Frage zwischendurch:
Ist die bisherige Konstruktion unserer "Datenbank" wirklich schon eine Datenbank oder ein Datenbank-System?

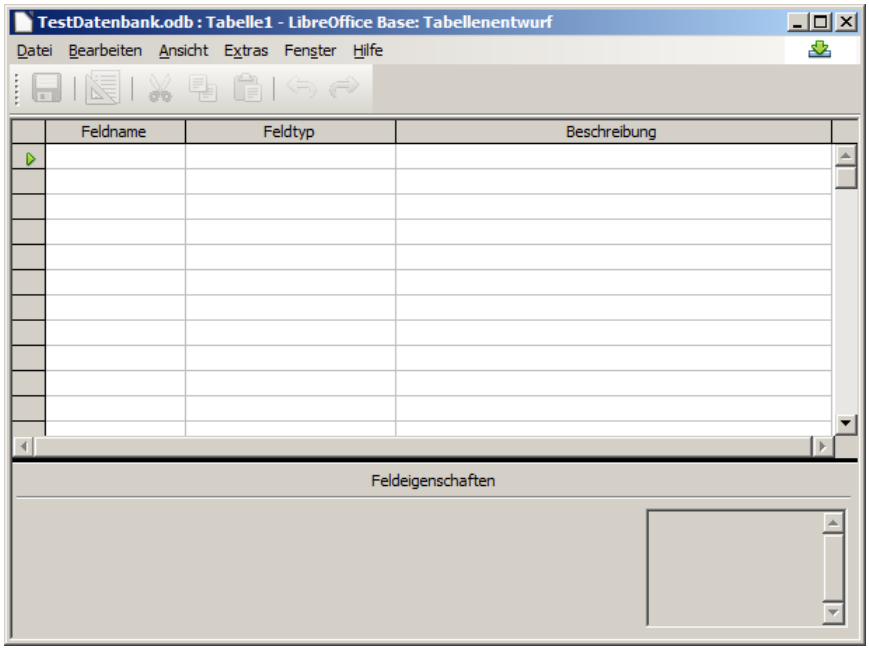
Das ERD für diese Miniwelt könnte dann so aussehen:



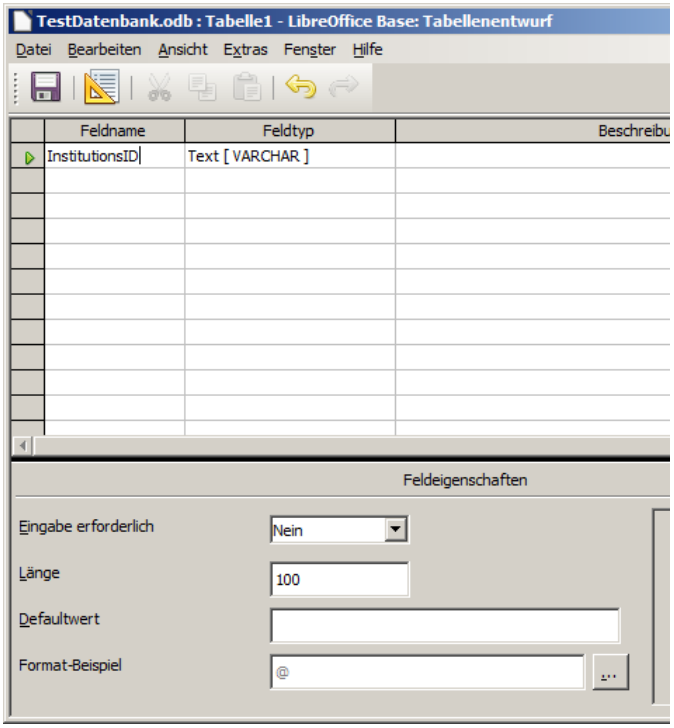
Zumindestens die Tabelle "Institutionen" können wir mit unseren Kenntnissen schon erstellen. Als Herausforderung erstellen wir die Tabelle eben nicht mit dem Assistenten, sondern "manuell". Der Aufgaben-Punkt dafür lautet "Tabelle in der Entwurfsansicht erstellen …".



Offensichtlich benötigen wir für jedes Feld (= Attribut) ersteinmal genau drei Angaben, den Namen, einen Typ und eine Beschreibung.
Die Beschreibung ist optional und dient nur der Dokumentation und Kommentierung (ev. kryptischer Feldnamen / Attribute).

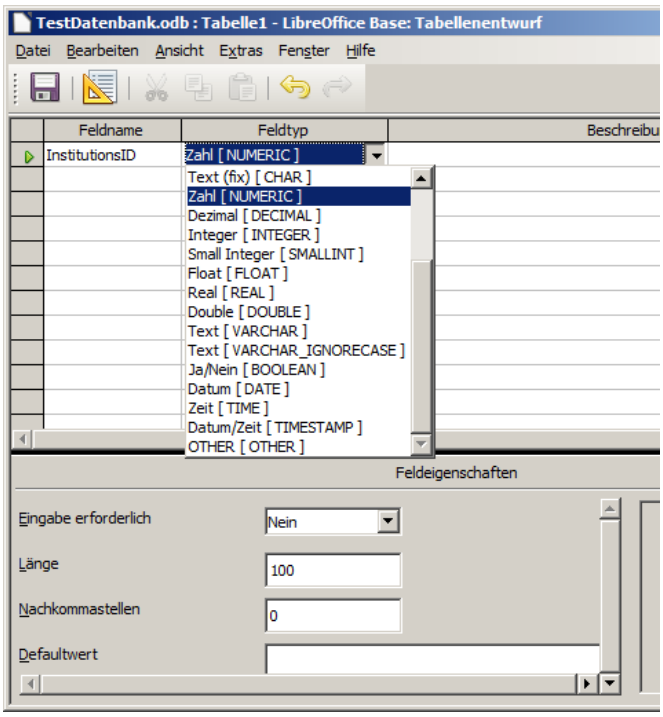


Für jede zukünftige Spalte unserer neuen Tabelle müsen wir hier jetzt eine Zeile anlegen.
Zuerst geben wir einen Feldnamen (Attributs-Namen) an. Achten Sie auf exakte Schreibung. Buchstaben-Dreher sind hier immer Stolpersteine, wenn dann später Daten aus der "falsch" geschriebenen Spalte geholt werden sollen.
Als Datentyp schlägt BASE immer Text vor. Für einfache Zwecke reicht das.
Zumindestens bei der Spalte des zukünftigen Primär-Schlüssels nehmen wir aber eine Korrektur vor.

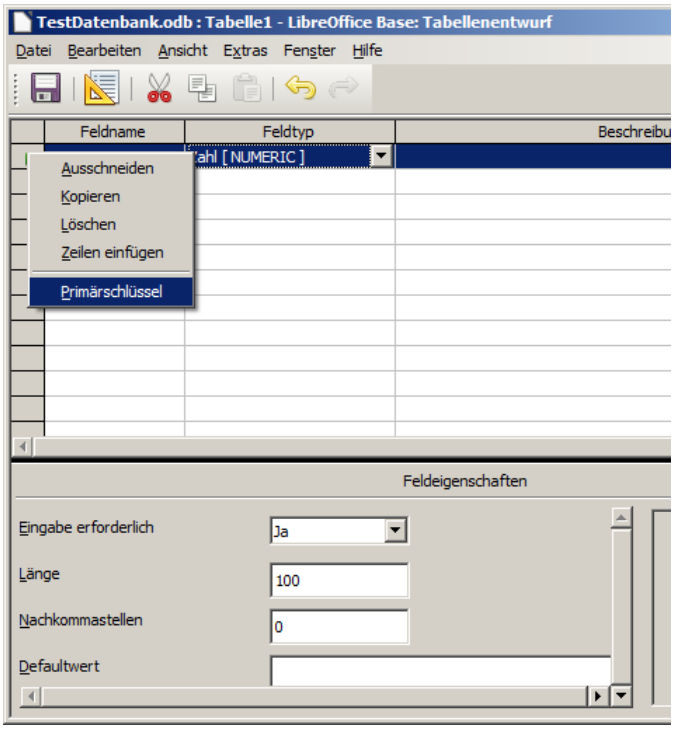


Für die zukünftige Primärschlüssel-Spalte ändern wir den Feldtyp (Datentyp) auf Zahl. Unten erscheinen dann die speziellen Möglichkeiten für Optionen.

Eine passgenaue Auswahl der Daten-Typen zu unseren Daten ist später immer dann von Vorteil, wenn diese Daten weiter verarbeitet werdn soll. In vielen DBMS gibt es für diverse Spezial-Funktionen, um z.B. Zeit-Angaben zu verrechnen.
Kaum jemand, der es schon mal gemacht hat, wird freiwillig die Differenz zwischen zwei Zeitwerten über eine eigenen Funktion berechnen wollen.



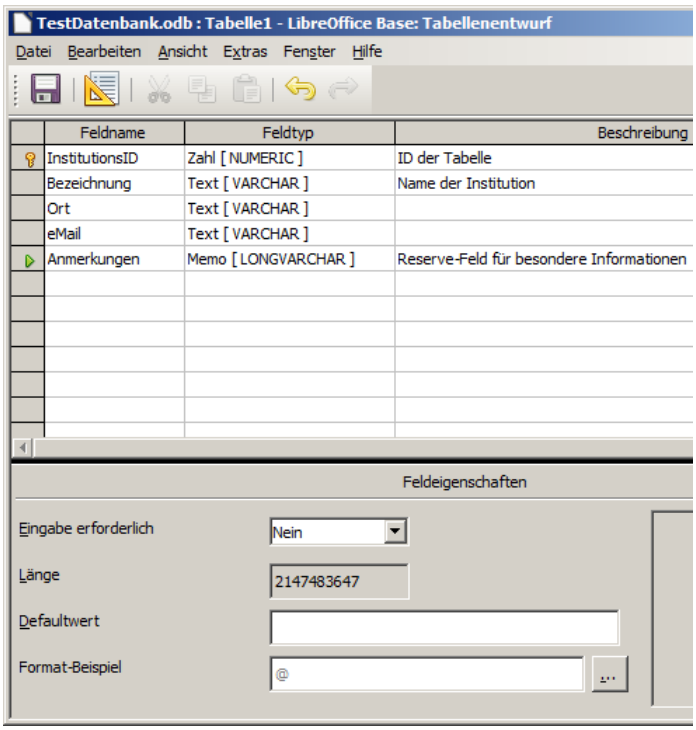
Auch wenn es an dieser Stelle noch nicht unbedingt notwendig ist, legen Sie gleich den Primärschlüssel fest.
Dazu klicken wir rechts auf die Feld-Zeile und wählen den passenden Kontextmenü-Punkt aus.



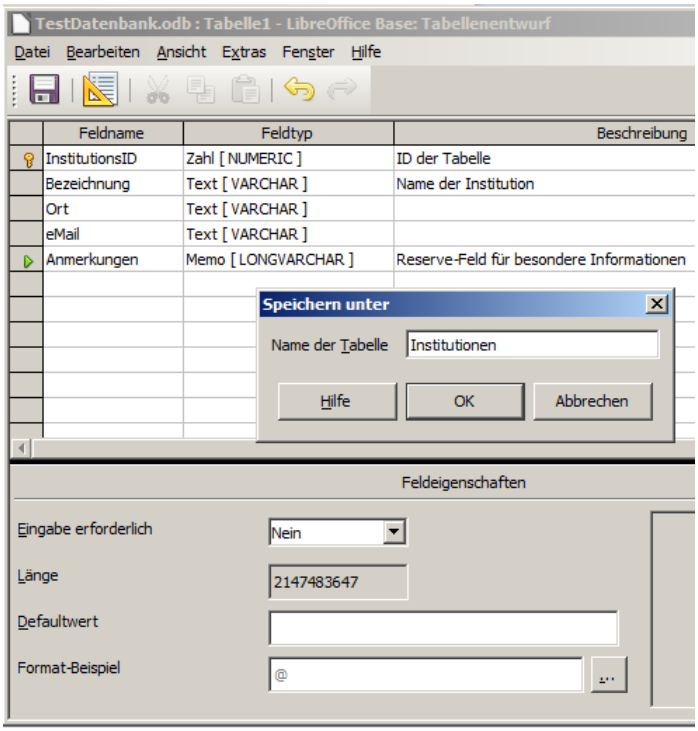
Aus Erfahrung weiss ich, wie oft man das sonst am Ende der Tabellen-Definition vergisst. Da legt BASE eine eigene Schlüssel-Spalte an und man muss wieder nacharbeiten, oder seine Daten-Modelle verändern.

Der kleine gelbe Schlüssel vor der Attribut-Definition (Feld-Definition) zeigt eine erfolgreiche Zuordnung an.

Ist die gesamte Definition vorgenommen worden, dannkönnte man eigentlich die Entwurfs-Ansicht verlassen.
In der Praxis kommt es fast immer vor, das irgendwann mal zusätzliche Daten zu einem Objekt auftauchen. Vielleicht ist es nur der Hinweis, dass dieser Datensatz noch korrigiert / überprüft werden muss oder ähnliches. Für solche Fälle empfehle ich, immer ein zusätzliches Feld "Anmerkungen" in die Tabellen-Definition aufzunehmen. Wählt man für diese Feld den Datentyp "Memo", dann wird fast nur soviel Speicherplatz verbraucht, wie Daten im Feld notiert sind (s.a. → Daten-Typen unter → 3.1.4.1. Tabellen).
Leere Memo-Felder nehmen fast keinen Speicherplatz weg.
Kommen bei vielen Objekten einer Tabelle ähnliche Daten zusammen, dann sollte man natürlich über eine zusätzlich zu definierende Spalte nachdenken.

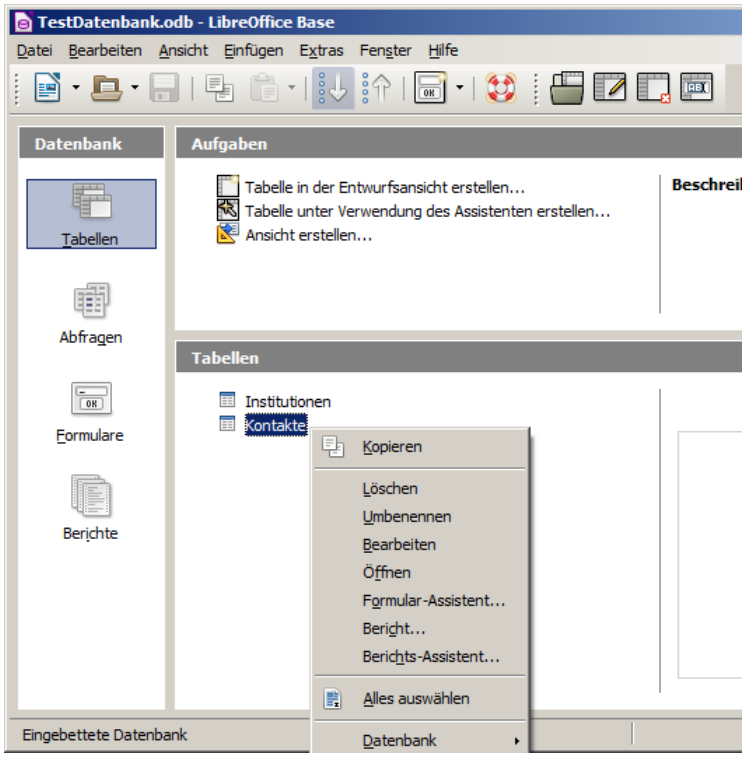


Tabellen ohne geänderten Namen müssen bei Verlassen der Entwurfs-Ansicht noch mit einem Namen versehen werden.

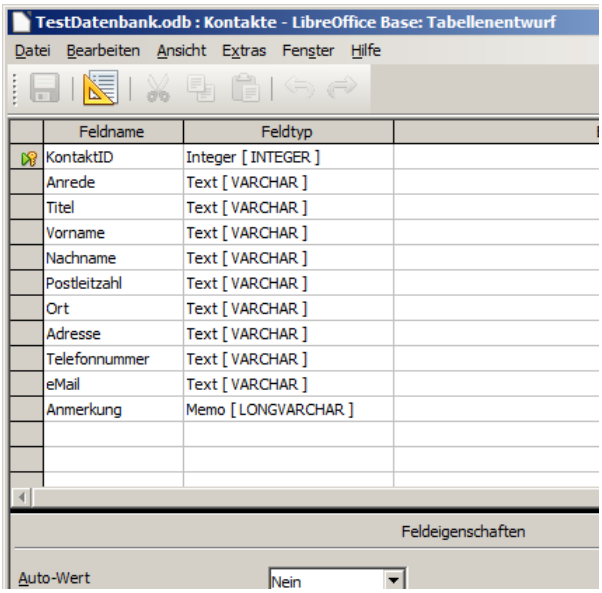


3.1.5.1.3. Ändern des Relationsschema einer Tabelle – Korrekturen am Entwurf

Wollen wir eine Tabellen-Defintion nachträglich ändern, dann geh das über das Kontext-Menü zur entsprechenden Tabelle. Der Menü-Punkt "Bearbeiten" ist vielleicht etwas verwirrend. Was bearbeiten wir, die Tabellen-Inhalte oder die Tabellen-Definition?
Es ist also die Tabellen-Definition – also kommen wir zur Entwurfs-Ansicht.
Änderungen an vorhandenen Felder müssen – zumindestens, wenn schon Daten eingetragen worden – sehr sehr vorsichtig vorgenommen werden. Da droht Daten-Verlust. Lieber neue Felder anlegen, die Daten gezielt und kontrolliert ins neue Feld übertragen und dann erst das akte Feld löschen.

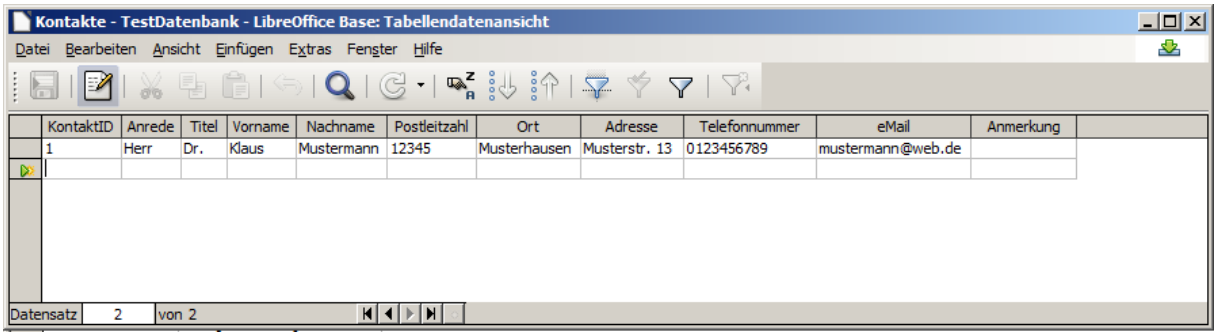


Das Hinzufügen neuer Felder ist i.A. unproblematisch.
So ergänzen wir die "Kontakte"-Tabelle um eine "Anmerkung"s-Spalte, so wie wir das bei der Institutions-Tabelle besprochen haben. Als Typ wählen wir das übliche Memo-Format.
Jede Veränderung an der Tabellen-Definition erfordert ein Speichern beim Verlassen der Entwurfs-Ansicht.
Machen wir das nicht, sind die Änderungen auch nicht permanent.



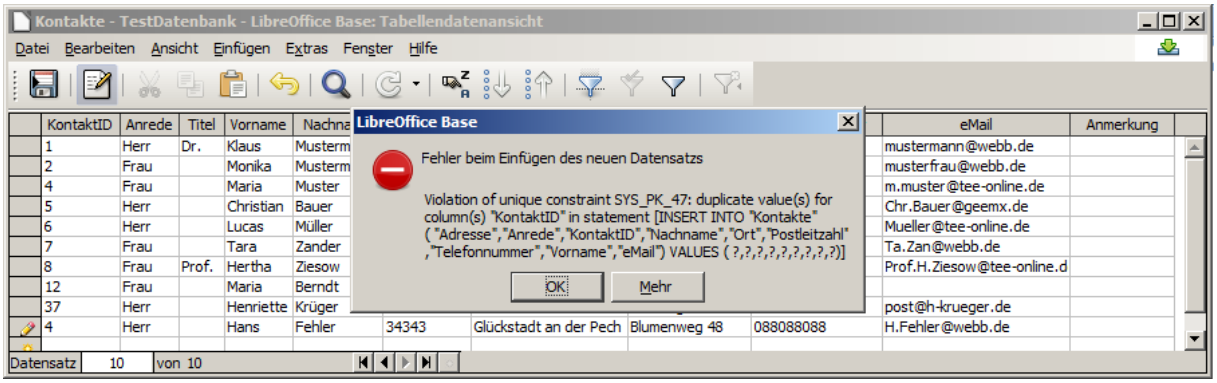
3.1.5.1.4. Eingeben von Daten in eine Tabelle

Klickt man in die zweite Zeile oder wechselt man z.B. mit [Tab] aus dem ersten Datensatz (1. Zeile) dorthin, dann ergeben sich zwei unscheinbare Effekte:

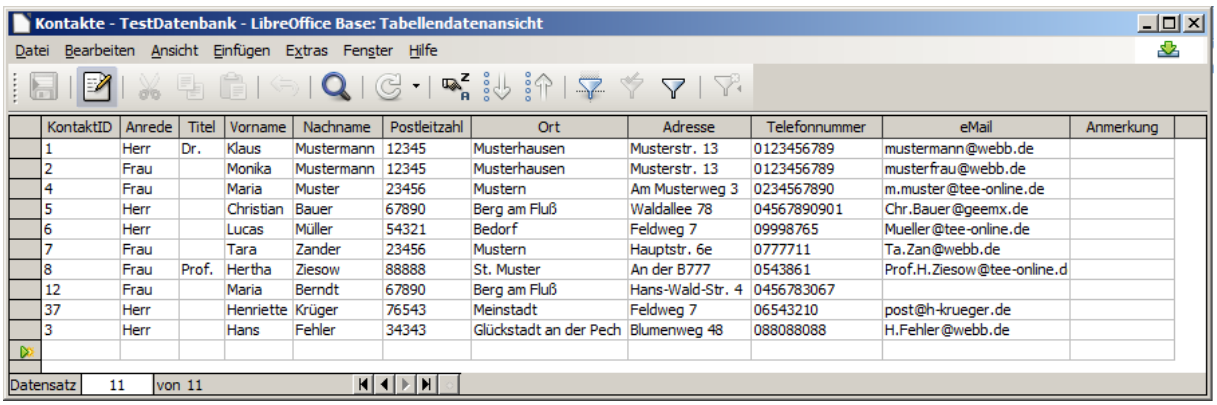


Der Stift – für den Eingabe-Modus – verschwindet und das aktivierte "Speichern"-Symbol (Diskette in der Symbolleiste) wird blaß. Beides bedeutet, dass die Daten - der gesamte Datensatz – wurde abgespeichert. Darum muss man sich in Datenbank-Programmen nicht kümmern.

Trotzdem kann es passieren – z.B. in ms ACCESS – dass man trotzdem zum Speichern aufgefordert wird. Dieses Speichern bezieht sich auf ev. Layout-Änderungen. Speichert man nicht, dann sind die Daten genau wie eingegeben gespeichert, nur das Layout wird nicht aktualisiert.
Zum Daten-Eingeben reicht sonst auch einfach der Doppel-Klick auf die gewünschte Tabelle in der Datenbank-Übersicht.
Zu einer kritischen Situation kommt es, wenn man versucht beim Primär-Schlüssel einen doppelten Wert einzugeben. Die Fehler-Meldung ist "super" informativ:



Hier den doppelten Schlüsselwert als Fehler-Ursache zu identifizieren ist wohl nur Datenbank-Neerds zuzutrauen.
Fertig eingegeben – und korrigiert - könnte die Kontakt-Tabelle dann so aussehen.



KontaktID	Anrede	Titel	Vorname	Nachname	Postleitzahl	Ort	Adresse	Telefonnummer	eMail	Anmerkung
1	Herr	Dr.	Klaus	Mustermann	12345	Musterhausen	Musterstr. 13	0123456789	mustermann@webb.de	
2	Frau		Monika	Mustermann	12345	Musterhausen	Musterstr. 13	0123456789	musterfrau@webb.de	
4	Frau		Maria	Muster	23456	Mustern	Am Musterweg 3	0234567890	m.muster@tee-online.de	
5	Herr		Christian	Bauer	67890	Berg am Fluß	Waldallee 78	04567890901	Chr.Bauer@geemx.de	
6	Herr		Lucas	Müller	54321	Bedorf	Feldweg 7	09998765	Mueller@tee-online.de	
7	Frau		Tara	Zander	23456	Mustern	Hauptstr. 6e	0777711	Ta.Zan@webb.de	
8	Frau	Prof.	Hertha	Ziesow	88888	St. Muster	An der B777	0543861	Prof.H.Ziesow@tee-online.d	
12	Frau		Maria	Berndt	67890	Berg am Fluß	Hans-Wald-Str. 4	0456783067		
37	Herr		Henriette	Krüger	76543	Meinstadt	Feldweg 7	06543210	post@h-krueger.de	
3	Herr		Hans	Fehler	34343	Glückstadt an der Pech	Blumenweg 48	088088088	H.Fehler@webb.de	

Aufgaben:

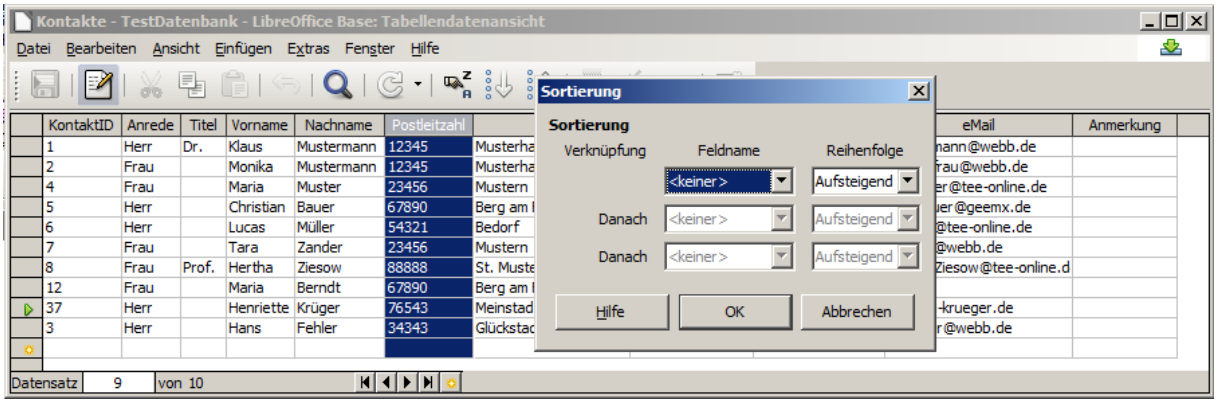
1. *Geben Sie die Daten aus der obigen Tabelle in die Tabelle "Kontakte! ein!*
2. *Erstellen Sie die Tabellen-Struktur der Tabelle "Institutionen" wie oben angegeben!*
3. *Füllen Sie die Tabelle "Institutionen" mit den folgenden Daten!*

InstitutionsID	Bezeichnung	Ort	eMail	Anmerkungen
1	Goethe-Gymnasium	Mustern	GoeGymn@webb.de	
2	Tanzverein "Polka"	Musterhausen	post@tv-polka.de	
3	Hilfe e.V.	Meinstadt	info@hilfe.info	
4	Traditionsverein	Cedorf	tradi@verein.de	
5	Let's share	Meinstadt	letsshare@verein.de	
6	Grundschule	Mustern	gs-mustern@webb.de	

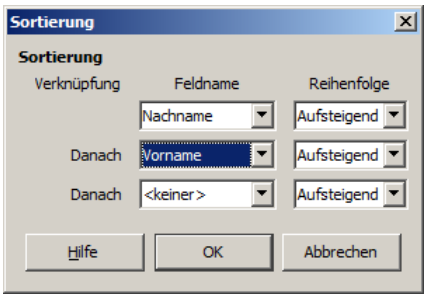
3.1.5.1.5. Filter und Sortierungen in Tabellen

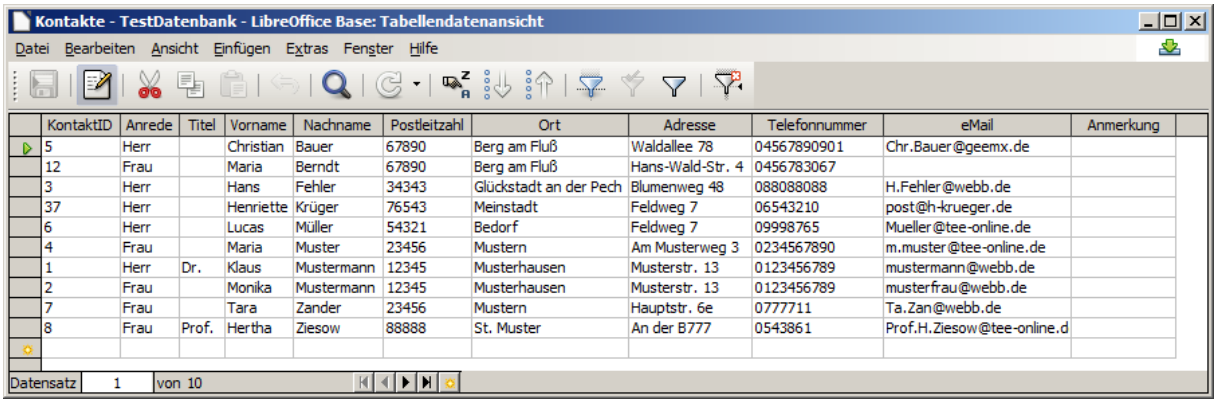
Bei großen Tabellen verliert man u.U. schnell mal die Übersicht, od das Eine oder Andere schon eingegeben wurde. Hier wäre es schön, wenn man sich nur Teile der Tabelle heraus-picken könnte oder die Werte in einer bestimmten Reihenfolge angezeigt werden.
Das Auswählen bestimmter Teile einer Tabelle nennt man Filtern, ein in die Reihenfolge bringen wird als Sortieren bezeichnet.
Markiert man eine Spalte, dann kann diese schnell mittels der Pfeil-Schaltfächen aus der Symbolleiste realisiert werden.

Für mehrstufige ode komplexere Sortierungen gibt es eine andere Möglichkeit. Egal, ob man eine Spalte markiert oder nicht, mit dem Sortieren-Symbol (A-Z-Symbol) gelangt man immer zu einem Assistenten.

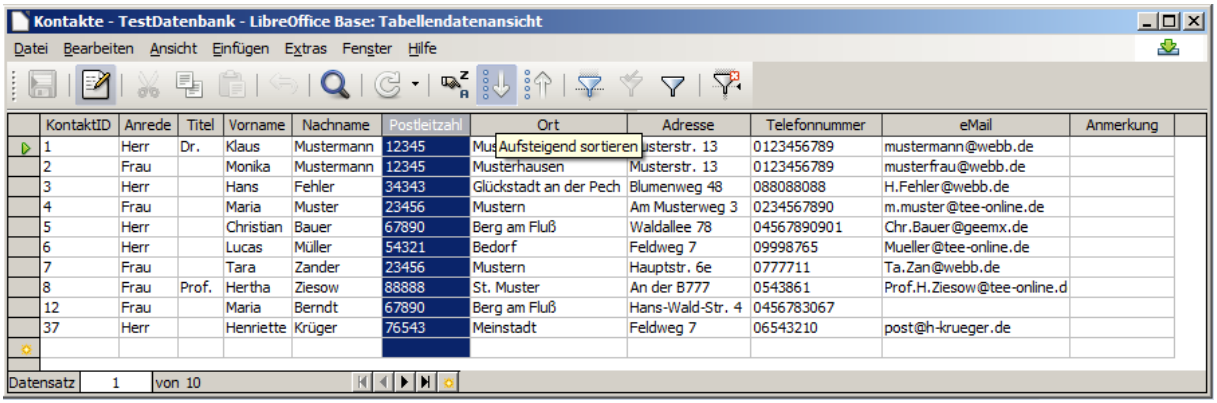


In diesem Sortierung's-Dialog lassen sich drei einzelne Sortierungen einrichten, die dann von oben nach unten abgearbeitet werden.
Im Besipiel (Abb. rechts) ist die klassische Sortierung zuerst nach dem Nachnamen und dann nach den Vor-namen eingerichtet.
Das Ergebnis ist wenig überraschend. An der Daten-Tabelle an sich änder die Sortierung garnichts. Es ist eine reine Darstellungsfrage und eigentlich schon einen Art Abfrage.





Will man nur nach einer Spalte sortieren, dann reicht das Markieren der Spalte und dann die Auswahl der Sortier-Richtung in der Symbolleiste.



KontaktID	Anrede	Titel	Vorname	Nachname	Postleitzahl	Ort	Adresse	Telefonnummer	eMail	Anmerkung
1	Herr	Dr.	Klaus	Mustermann	12345	Mus	usterstr. 13	0123456789	mustermann@webb.de	
2	Frau		Monika	Mustermann	12345	Musterhausen	Musterstr. 13	0123456789	musterfrau@webb.de	
3	Herr		Hans	Fehler	34343	Glückstadt an der Pech	Blumenweg 48	088088088	H.Fehler@webb.de	
4	Frau		Maria	Muster	23456	Mustern	Am Musterweg 3	0234567890	m.muster@tee-online.de	
5	Herr		Christian	Bauer	67890	Berg am Fluß	Waldallee 78	04567890901	Chr.Bauer@geemx.de	
6	Herr		Lucas	Müller	54321	Bedorf	Feldweg 7	09998765	Mueller@tee-online.de	
7	Frau		Tara	Zander	23456	Mustern	Hauptstr. 6e	0777711	Ta.Zan@webb.de	
8	Frau	Prof.	Hertha	Ziesow	88888	St. Muster	An der B777	0543861	Prof.H.Ziesow@tee-online.d	
12	Frau		Maria	Berndt	67890	Berg am Fluß	Hans-Wald-Str. 4	0456783067		
37	Herr		Henriette	Krüger	76543	Meinstadt	Feldweg 7	06543210	post@h-krueger.de	

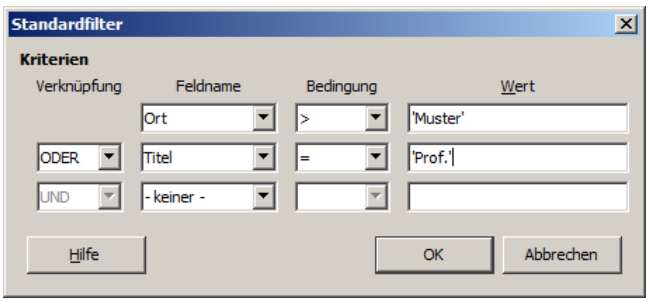
Will man dann wieder den Grundzustand bzw. den letzten Filter- bzw. Sortier-Zustand, dann können die aktuellen Sortierungen und / oder Filterungen mit dem rot-geïxten Trichter-Symbol zurückgesetzt werden.

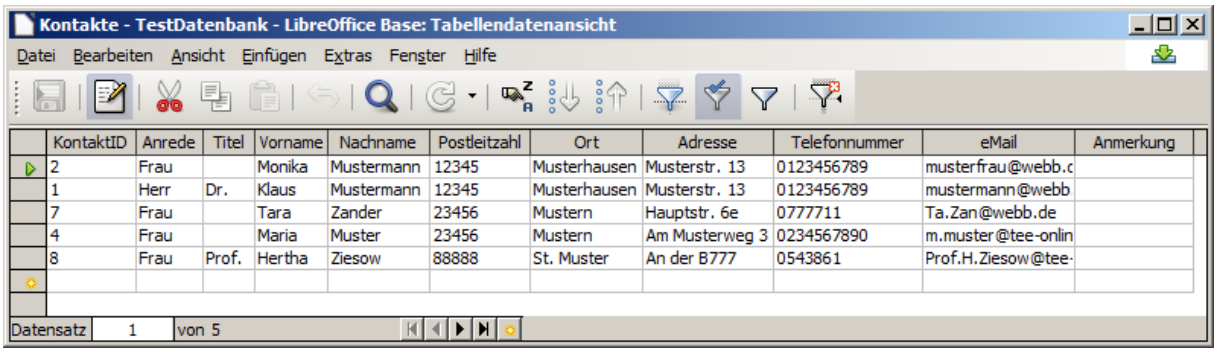
Die erste Schaltfläche ("Autofilter") steht für eine Filterung nach dem markierten Begriff / Eintrag in der Tabelle. Ist eine Filterung aktiviert worden, dann wird auch das zweite Symbol ("Filter anwenden") nutzbar. Mit ihm kann zwischen gefilteter Ansicht und der vollständigen Tabelle gewechselt werden.

Hinter dem einfachen Trichter-Symbol (Standardfilter) versteckt sich ein kleiner Assisten, der uns ein wenig an die mehrstufige Sortierung von oben erinnert.

Hier können wir nun Bedingungen für die Auswahl in freier Kombination eingegeben werden.

Als Ergebnis erhalten wir eine aufgekürzte Tabelle mit den betreffenden / gewünschten Datensätzen.

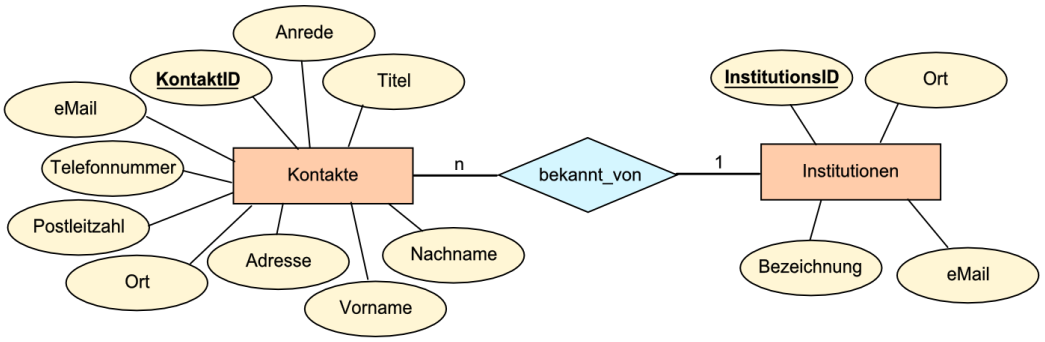




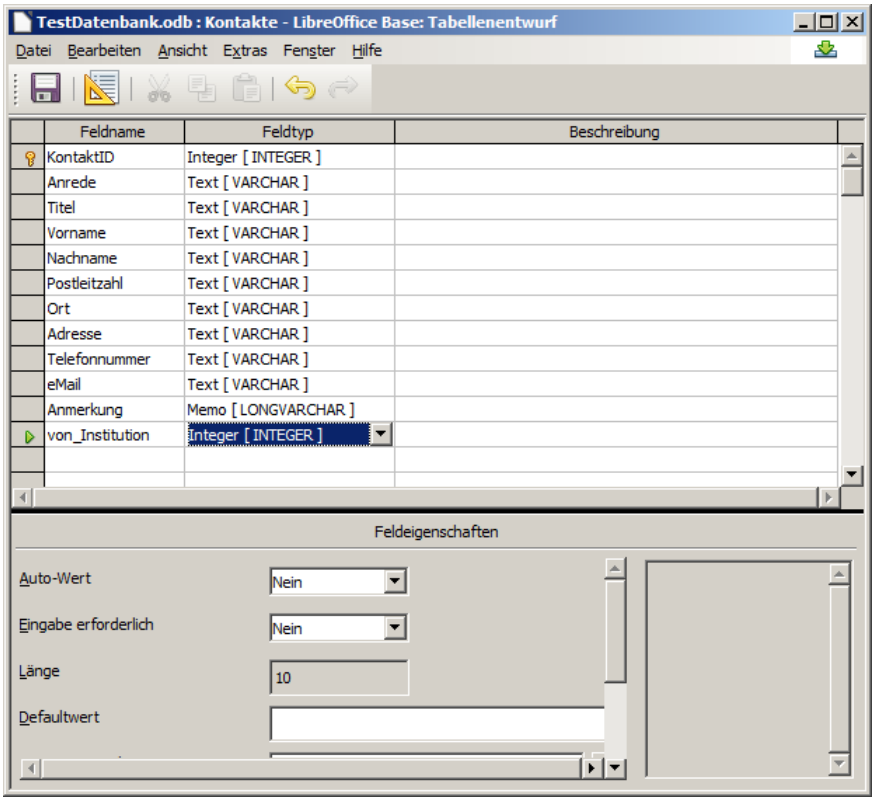
KontaktID	Anrede	Titel	Vorname	Nachname	Postleitzahl	Ort	Adresse	Telefonnummer	eMail	Anmerkung
2	Frau		Monika	Mustermann	12345	Musterhausen	Musterstr. 13	0123456789	musterfrau@webb.c	
1	Herr	Dr.	Klaus	Mustermann	12345	Musterhausen	Musterstr. 13	0123456789	mustermann@webb	
7	Frau		Tara	Zander	23456	Mustern	Hauptstr. 6e	0777711	Ta.Zan@webb.de	
4	Frau		Maria	Muster	23456	Mustern	Am Musterweg 3	0234567890	m.muster@tee-onlin	
8	Frau	Prof.	Hertha	Ziesow	88888	St. Muster	An der B777	0543861	Prof.H.Ziesow@tee-	

3.1.5.1.6. Beziehungen zwischen Tabellen umsetzen

Die Tabellen "Kontakte" und "Institutionen" sind die Basis für die Beziehung "bekannt_von". In diese Tabelle werden nun keine echten Daten (Klar-Bezeichnungen) eingetragen, sondern nur Verweise auf die beiden aggregierten Tabellen. Und was eignet sich dabei besser als die Primär-Schlüssel der jeweils betroffenen Datensätze.
Solange wir noch keine "Formulare" zur Verfügung haben (→ 3.1.3.4. Formulare), ist die Dateneingabe noch sehr mühselig.



Nach den Transformations-Regeln (→ 3.0.4. Transformation eines ERM (ERD) in eine relationale Datenbank) müssen wir an die Tabelle Kontakte ein Spalte "bkannt_von" oder so ähnlich anhängen und diese dann passend mit den Primär-Schlüsseln der anderen Tabelle füllen.



Dazu wechseln wir in die Tabellen-Entwurfsansicht ("Bearbeiten" aus dem Kontext-Menü). Hier fügen wir dann an passender Stelle oder hinten an ein neues Feld ein. Als Datentyp muss einer gewählt werden, der dem Primärschlüssel der anderen Tabelle entspricht.
Das Auffüllen mit den zugehörigen Daten ist dann mehr eine Finger-Übung.
Später kann man dafür dann auch spezielle Formulare nutzen, welche die Institutionen im Klarnamen angeben und dann die Umsetzung in den passenden Primär-Schlüssel selbstständig übernehmen.

3.1.4.1.6.1. Erstellen einer einfachen Beziehung zwischen zwei Tabellen

Was wir jetzt noch brauchen ist die Verbindungsliste zwischen den beiden Entitäts-Typen:
Als Erstes müssen wir ermitteln um welchen Verknüpfungs-Typ es sich handelt.
In der Tabelle rechts sind die Beziehungen in Menschen-verständlicher Form zusammengetragen.
Zur Wahl stehen 1 : 1, 1 : n bzw. n : 1 oder eben n : m.
Da die Datensätze bei Kontakte immer nur einfach vorkommen (werden) und die Institutionen hier mehfach genannt sind, handelt es sich um eine "1 : n"-Beziehung.

Daten aus Tabelle …

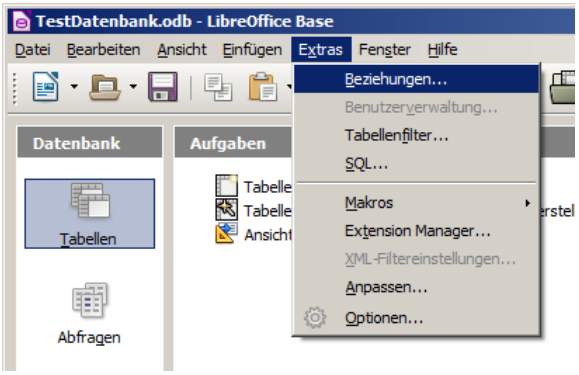
Kontakte		Institution
Name	**Vorname**	**bekannt_von**
Mustermann	Klaus	Goethe-Gymnasium
Mustermann	Monika	Tanzverein
Muster	Maria	Traditionsverein
Bauer	Christian	Let's share
Müller	Lucas	Goethe-Gymnasium
Zander	Tara	Grundschule
Ziesow	Hertha	Traditionsverein
Berndt	Maria	Hilfe e.V.
Krüger	Henriette	Goethe-Gymnasium
Fehler	Hans	Goethe-Gymnasium

Natürlich könnte man auch für eine Person mehrere Institutionen als Kontakt konstruieren. Dann wäre es eine "n : m"-Beziehung. Das geben unsere Daten aber nicht her. Trotzdem besprechen wir diese Möglichkeit weiter hinten.
Wie nun die Abbildung der Beziehungs-Typen in realtionalen Datenbanken erfolgt, haben wi schon vorgestellt (→ 2.1.6. Transformation eines ERM (ERD) in eine relationale Datenbank).

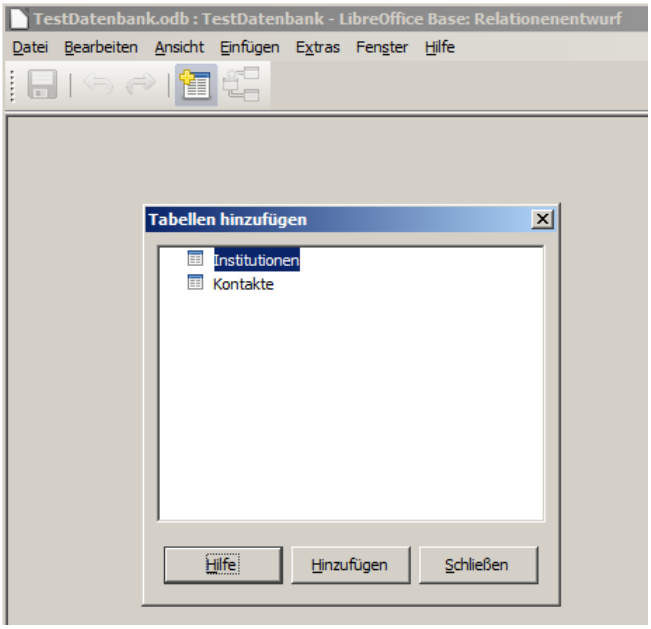
Aufgaben:

1. *Stellen Sie in einer Papiertabelle die Beziehung zwischen den Tabellen Kontakte und Institutionen über Fremdschlüssel her!*
2. *Ergänzen Sie eventuell fehlende Daten für einen Datenbank-Einsatz!*

Mit der obigen Aktion haben wir nun Daten-technisch ein Verbindung hergestellt, wir wissen um die Zusammenhänge – nicht aber das System.
Zwar könnte man jetzt schon spezielle Abfragen erzeugen (→ 3.1.5.2. Abfragen), die auf unserem Hintergrundwissen zu den Tabellen-Beziehungen basieren, aber wir wollen ja mehr dem DBS die Arbeit aufnacken. Außerdem würden fremde Nutzer kaum alle Beziehungen wissen können. Sie müssten sich dazu mit der Struktur der Datenbank beschäftigen. Diese ist ihnen aber nicht unbedingt zugänglich.
Das Erstellen der Beziehungen erfolgt graphisch und quasi Assistenten-gestützt.
Wir wählen dazu aus dem "Extras"-Menü den Punkt "Beziehungen …) aus.

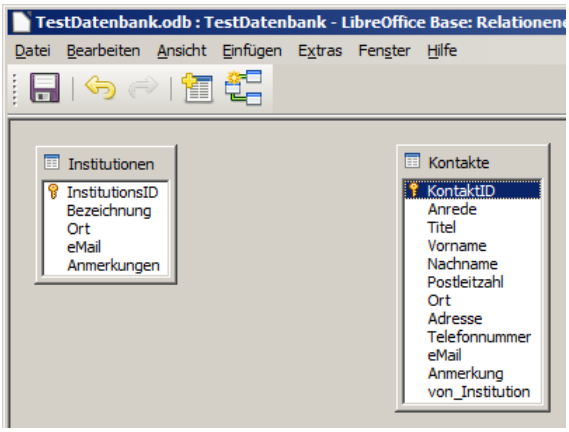


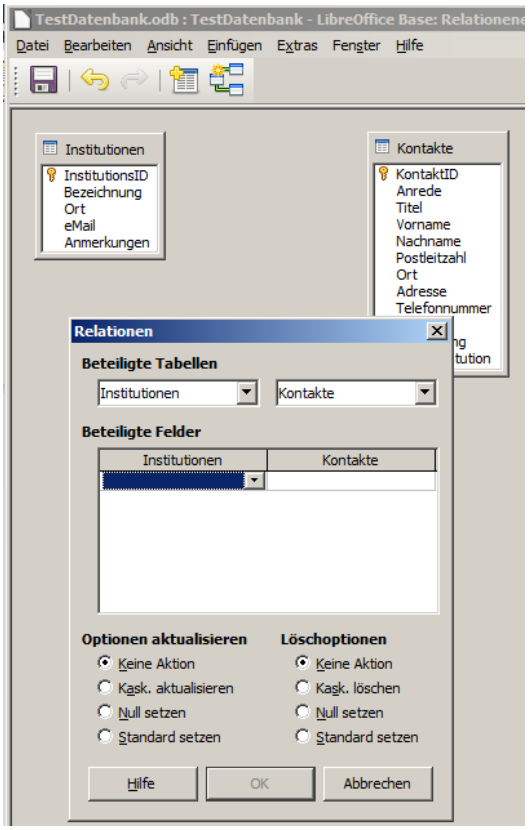
Es erscheint ein Arbeitsfläche (Relationsentwurf), auf der die Tabellen und ihre Beziehungen untereinander graphisch dargestellt werden können.
Hier sind dann später auch Änderungen möglich. Diese sollten aber frühzeitig erfolgen, da sonst Inkonsistenzen mit Abfragen usw. auftreten können, die auf "veraltete" bzw. "fehlerhafte" Beziehungen beruhen.
Zuerst müssen wir die benötigten Tabellen zur graphischen Arbeitsfläche "Hinzufügen".
Dabei sollte man zuerst einmal so sparsam wie möglich hinzufügen. Die Übersichtlichkeit kann sonst stark leiden.
Natürlich müssen aber alle (für die aktuelle Aufgabe) benötigten Tabellen hinzugefügt werden.



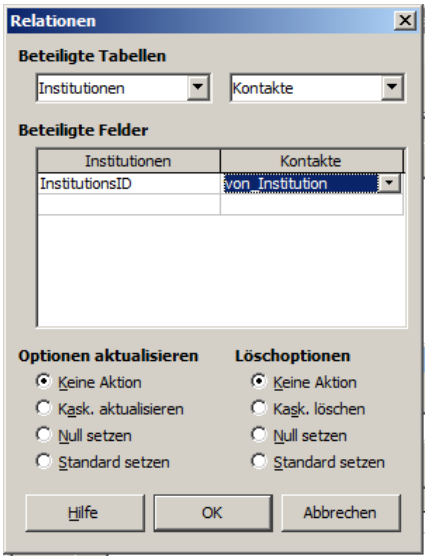
Sollten auf dem Relationsentwurf Tabellen fehlen, dann lassen sich diese mit der Schaltfläche "Tabelle hinzufügen".

Die Tabellen-Struktur-Tabellen (Tabellen-Bildchen) können frei auf der graphischen Oberfläche angeordnet werden. Schon vorhandene Beziehungen werden beim Umsortieren mit verschoben.
Über die Schaltfläche "Neue Relation" kann nun eine neue Beziehung eingerichtet werden.

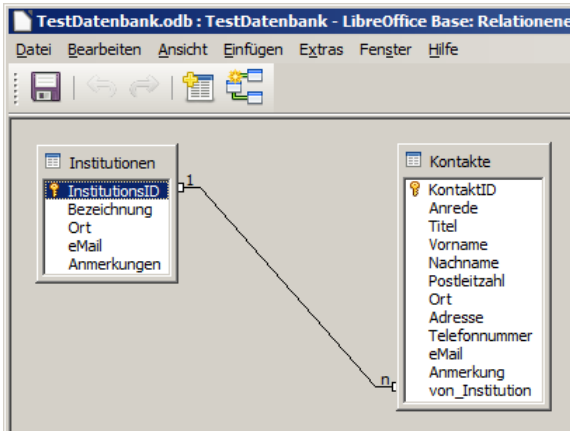




Zuerst wählt man die beteiligten Tabellen und dann die Felder aus. Die Datenliefernde Tabelle steht dabei links.
Nun werden die passenden Felder ausgewählt. Wie üblich erscheint der Auswähler erst, wenn man in die Zelle klickt.
Die verfügbaren Optionen lassen wir erst einmal unberührt. Sie können später geändert werden.



Als Ergebnis erhalten wir eine Übersicht, die schon gut die von uns erdachte Beziehungen aus dem ERD abbildet.
Eine nicht-gebrauchte Beziehung (Relation) lässt sich durch Anklicken und drücken der "Entf"-Taste löschen. Bitte hier mit Bedacht vorgehen, denn die Relationen sind auch dann weg, wenn man versucht den Relationsentwurf ohne Abspeichern zu Schließen!



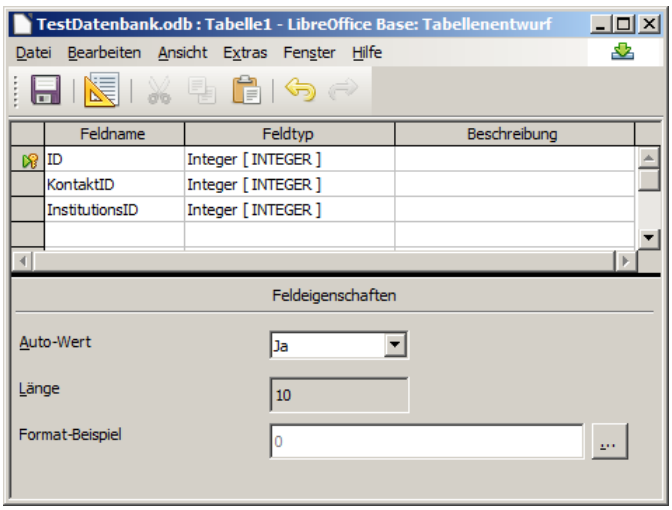
3.1.4.1.6.2. Erstellen einer "n : m"-Beziehung

Aufgaben:

1. ***Überlegen Sie sich, wie die Beziehung realisiert werden müsste, wenn es sich um eine "n : m"-Beziehung handeln würde! Bereiten Sie eine passende Realisierung vor! Diskutieren Sie Ihren Vorschlag mit anderen Kurs-Mitgliedern!***

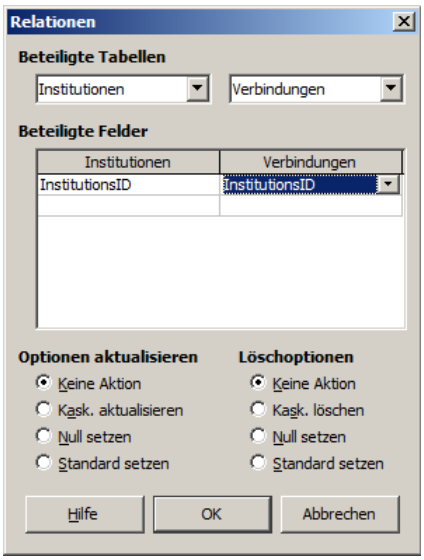
Für "n : m"-Beziehungen brauchen wir ja eine Zwischentabelle (Beziehungs-Tabelle) (s.a. → 3.0.4. Transformation eines ERM (ERD) in eine relationale Datenbank). Die "n : m"-Beziehung wird dann in zwei Beziehungen zerlegt, welche die Kardinalitäten zwischen den Daten-Tabellen und der Beziehungs-Tabelle darstellen.

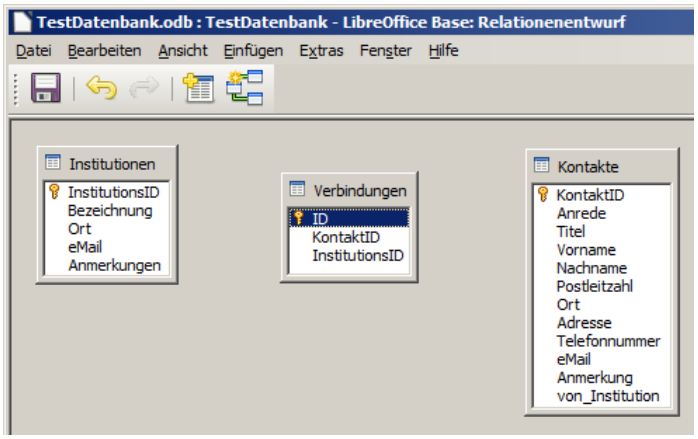
Nehmen wir an, wir wollten die "n : 1"-Beziehung zwischen "Kontakte" und "Institutionen" in eine "n : m"-Beziehung umwandeln, um alle Kontakt-Möglichkeiten abzubilden, dann könnte man etwa so vorgehen. Zuerst wird die Beziehungstabelle erstellt und mit diese dann mit Daten gefüllt.



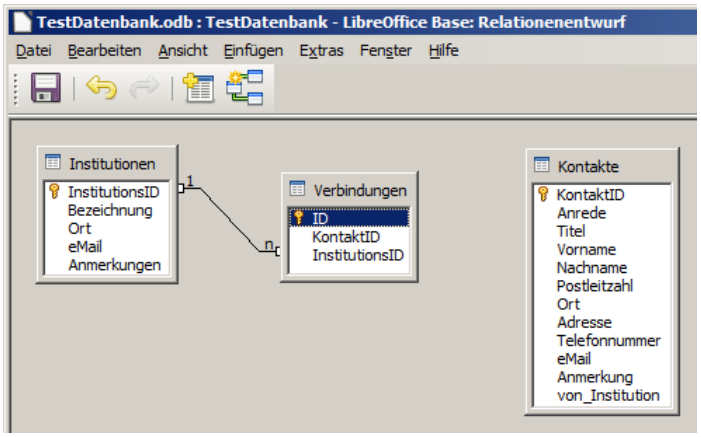
In dieser Umsetzung bleiben wir mal bei den ursprünglichen Beziehungen zwischen den Kontakten und den Institutionen, aus denen sie bekannt sind. Praktisch könnten wir jetzt beliebig viele weitere Kontakt-Beziehungen hinzugeben. Tara Zander könnte uns dann z.B. aus allen Institutionen bekannt sein.

Das Einfügen einer Tabelle in den Relationsentwurf sollte kein Problem mehr darstellen.
Im nächsten Schritt werden jetzt die Beziehungen zwischen den Daten-Tabellen und der Beziehungs-Tabelle hergestellt.
Auch das sollte für uns kein Hinderniss mehr sein.

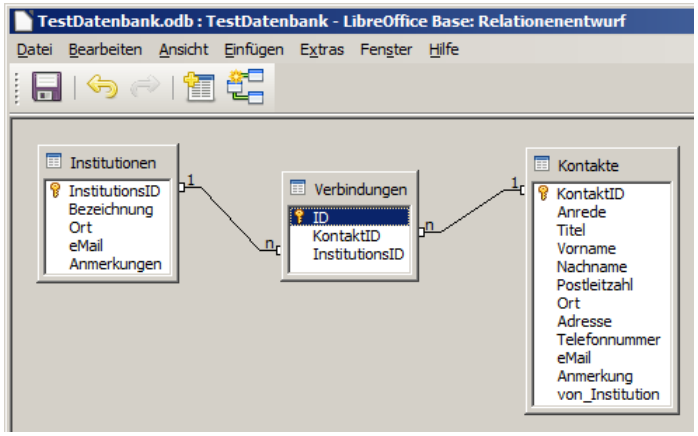




Das Ergebnis ist ebenfalls wenig überraschend (s. Abb. unten).



Die zweite Beziehung (Relation) wird genau so – wie vorne besprochen – erstellt.
Fertig ist unsere "n : m"-Beziehung. Aus der ursprünglichen "n : m"-Beziehung sind nun zwei "1 : n"-Beziehungen geworden.
Später könnte man z.B. die Tabelle "Verbindungen" um persönliche Verbindungsdaten, wie die individuelle eMail-Adresse für die konkrete Kontaktperson in dieser Institution.



Nun können neben den Erst-Kontakt-Beziehungen (gräulich unterlegt) auch noch diverse weitere Kontake auftauchen, so wie es in der Realität wohl wahrscheinlich ist:
Die Spalte "von_Institution" wird mit der n : m-Beziehung natürlich überflüssig und könnte / sollte entfernt werden, es sei den man will den Erst-Kontakt noch weiterhin als Information behalten.

Name	Vorname	bekannt_von
Mustermann	Klaus	Goethe-Gymnasium
Mustermann	Monika	Tanzverein
Muster	Maria	Traditionsverein
Bauer	Christian	Let's share
Müller	Lucas	Goethe-Gymnasium
Zander	Tara	Grundschule
Ziesow	Hertha	Traditionsverein
Berndt	Maria	Hilfe e.V.
Krüger	Henriette	Goethe-Gymnasium
Fehler	Hans	Goethe-Gymnasium
Berndt	Maria	Grundschule
Mustermann	Monika	Traditionsverein
Zander	Tara	Goethe-Gymnasium
Mustermann	Monika	Let's share
Mustermann	Klaus	Traditionsverein
Mustermann	Monika	Goethe-Gymnasium
Krüger	Henriette	Grundschule
Fehler	Hans	Hilfe e.V.

Aufgaben:

1. ***Setzen Sie nun den – im Gruppen-Gespräch – favorisierten Vorschlag (n : m-Beziehung) in BASE um! Nutzen Sie die Beziehungs-Informationen aus der obigen Tabelle!***

3.1.4.1.8. Erstellen von Tabellen mit SQL-Statement's

Die Erstellung von Tabellen und das Daten-Befüllen über SQL-Anweisungen ist möglich. Bei den von BASE gebotenen graphischen und Assistenz-basierten Möglichkeiten ist diese Variante eher zu aufwändig.
Anders sieht es aus, wenn man einen SQL-Text schon vorliegen hat oder diesen z.B. für einen universellen Einsatz auf unterschiedlichsten DBS schon vorbereitet hat. In diesem Fall ist eine SQL-basierte Definition und Daten-Eingabe eine echte Alternative.
Einige Datenbanken bieten einen SQL-Export ihrer Daten-Bestände an. Dies sind oft geeignete SQL-Texte für eine vollständige Einspielung in ein anderes DBMS.

3.1.5.2. Abfragen

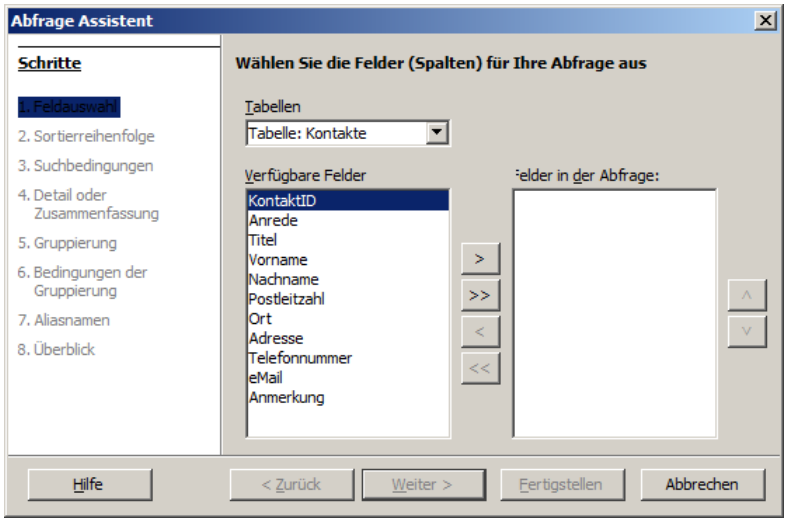
Die Daten-Menge großer Datenbanken übersteigen die Darstellungs-Möglichkeiten auf eine Bildschirm und die Verarbeitungs-Kapazität eines Menschen. Irgendwie müssen die Daten für die Anzeige reduziert werden. Auch gesetzliche Regelungen oder Verschwiegenheits-Vereinbarungen lassen nicht für jeden Nutzer einen Zugang zu allen Daten einer Datenbank zu. Über Selektionen (→ Selektion / Restriktion) und Projektionen (→ Projektion) werden die Daten auf das notwendige Maß zusammengestrichen und für die weitere Nutzung bereitgestellt. Die Auswahl der Daten und Bereitstellung in neuen (temporären) Tabellen nennt man Abfragen (Sichten / View's). Eine weitere Nutzung einer Abfrage kann dann in weiteren Abfragen, Berichten (→ 3.1.5.3. Berichte) oder Formularen (→ 3.1.5.4. Formulare) erfolgen.

3.1.5.2.1. Erstellen einer Abfrage mit dem Assistenten

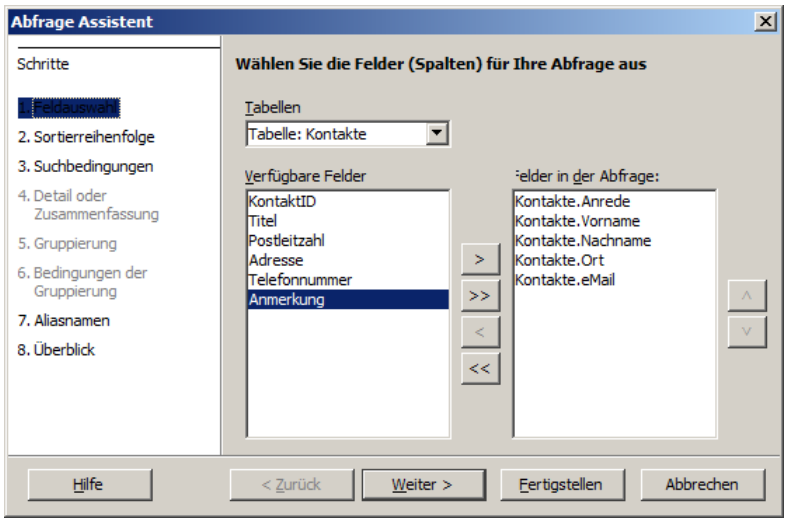
Für Anfänger ist die Assistenten-Methode wohl die einfachte Variante. Schrittweise klickt man sich die Elemente und kriterien für die Abfrage zusammen. Man kann jederzeit wieder zurück bis hin zum Anfang, um notwendige Korrekturen vorzunehmen oder vernachlässigte Kriterien mit einzubauen.

Auswahl der Felder

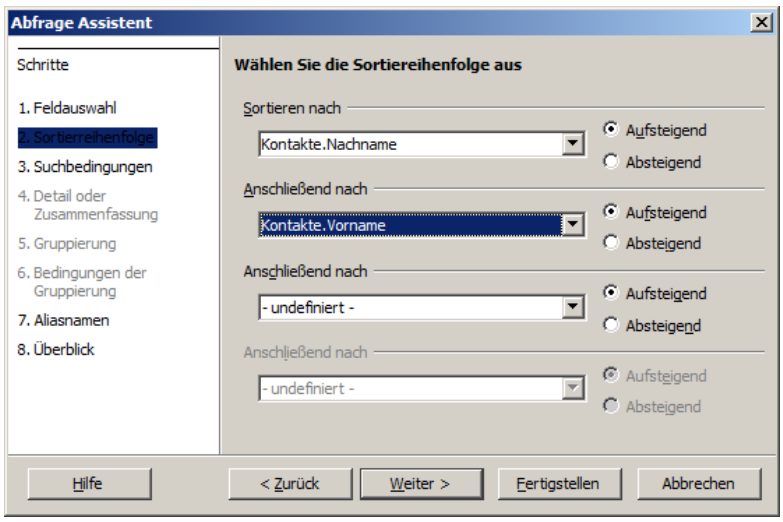
Durch die Wahl einzelner Felder aus der Gesamt-Tabelle beschränken wir also die Anzahl der bereitgestellten Spalten.
Hierbei handelt sich also praktisch um eine Projektion (→Projektion).



Bestätigung der Auswahl

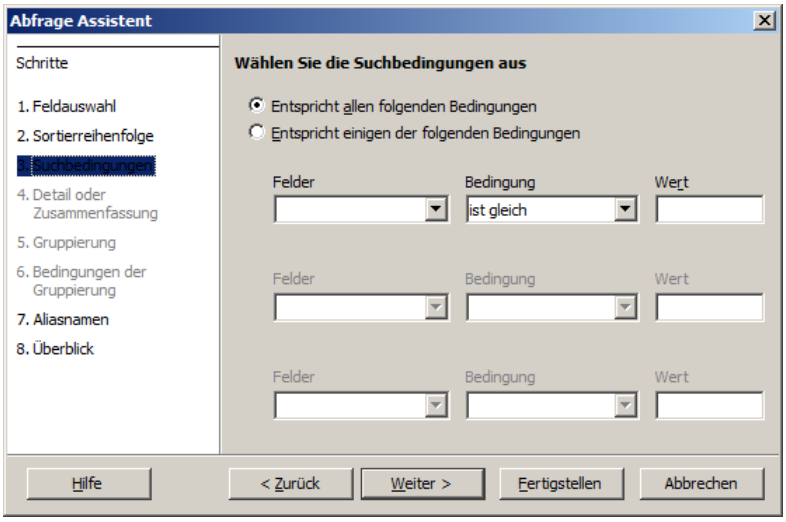


Sortierung



weitere Suchbedingungen

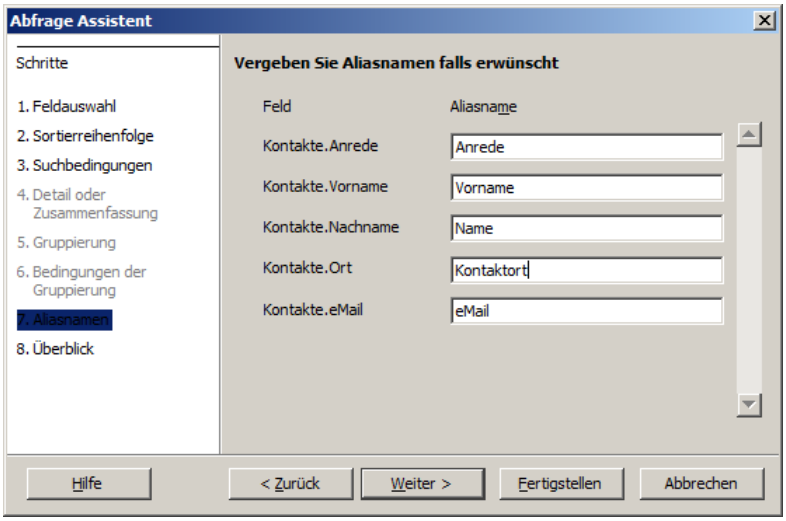
Durch weitere Auswahl-Kriterien beschränken wir die Anzahl der Ergebnis-Zeilen. Es werden nur noch ausgewählte Datensätze angezeigt.
Hier handelt es sich somit um eine Selektion (→Selektion / Restriktion).



Wenn wir vorne schon eine Auswahl der Felder vorgenommen haben, dann liegt insgesamt eine Kombination aus Selektion und Projektion vor. Resultierende Tabellen sind dann immer deutlich kleiner als die ursprünglichen.

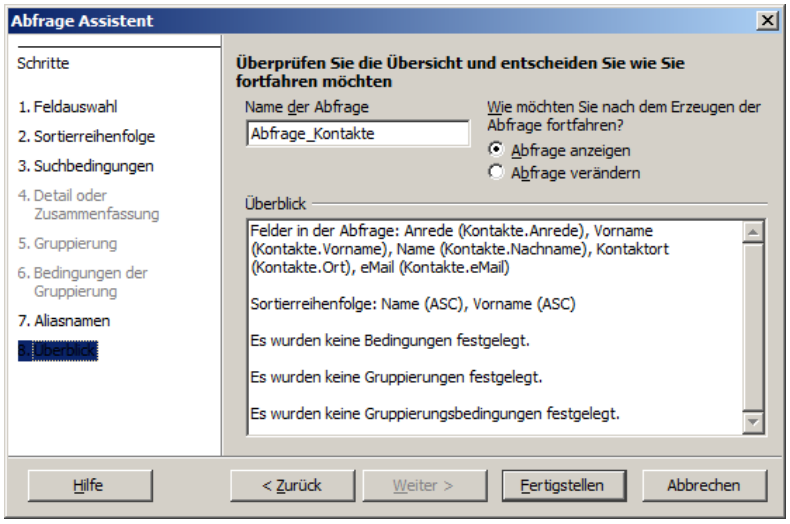
Aliasnamen – also spezielle Namen für die Abfragen

Verschleierung der Datenbank-Interna oder Verbesserung der Nutzerfreundlichkeit



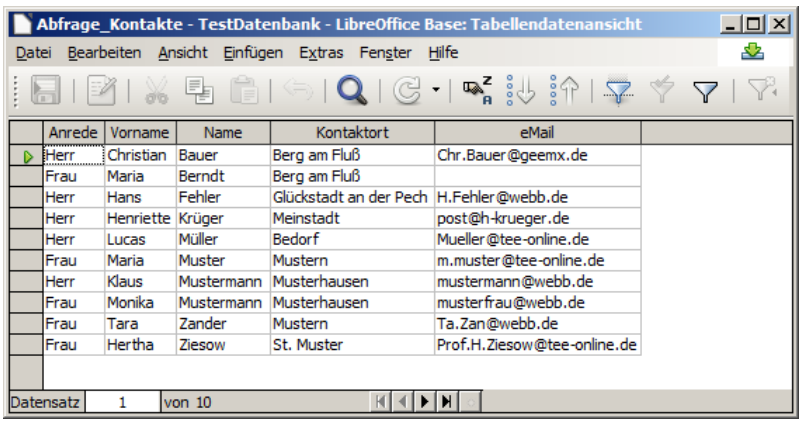
Überblick

Zusammenfassung des Assistenten

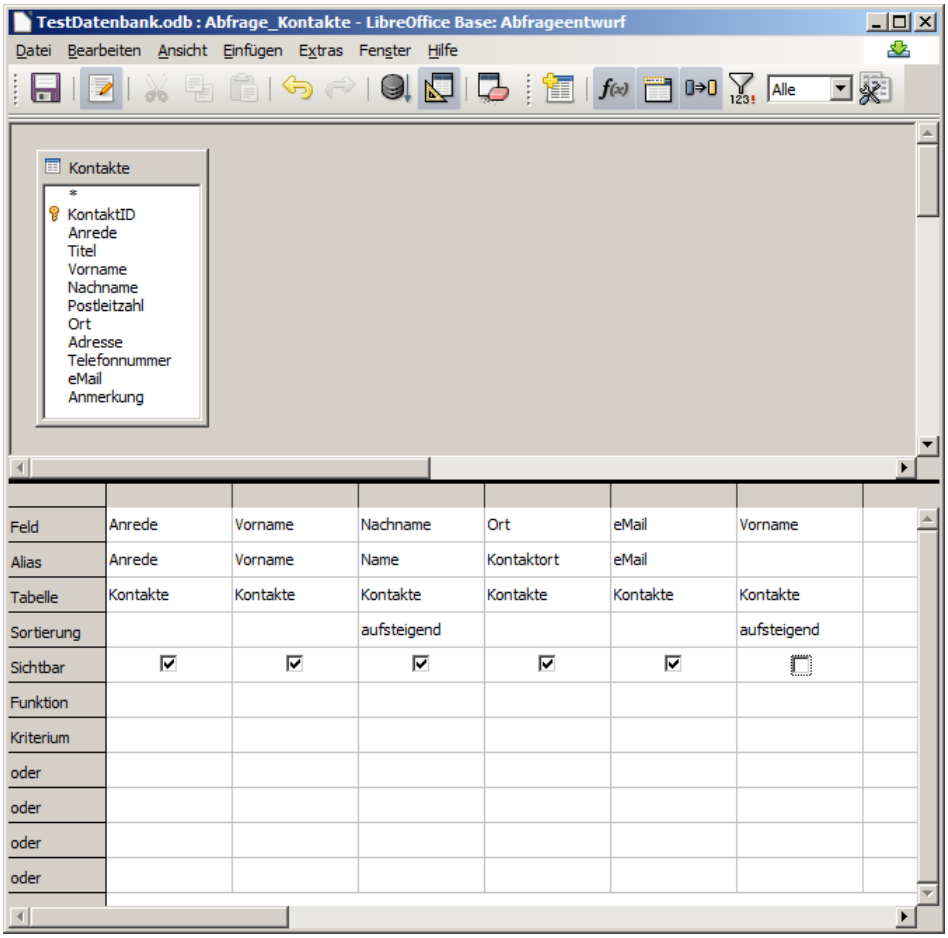


Ergebnis ist eine Tabelle

ist temporär, wird also nur erzeugt, wenn der Bedarf besteht, es werden dann die jeweils aktuellen Daten benutzt



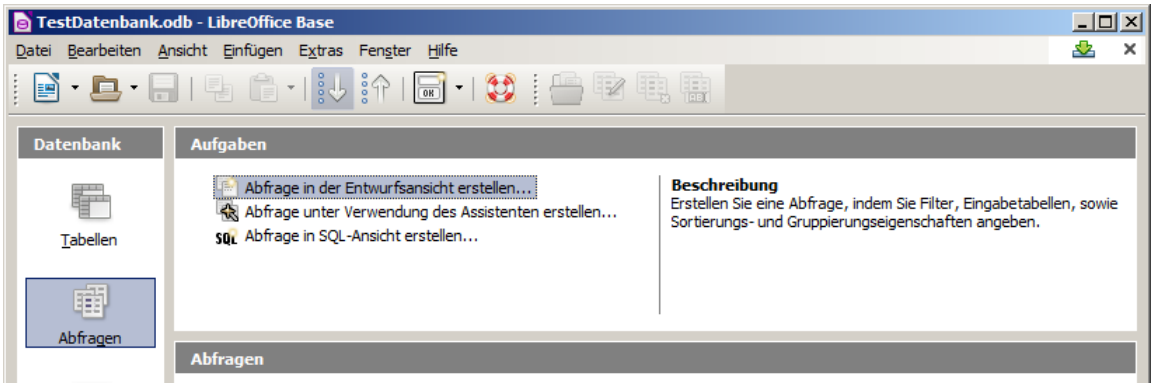
Wer etwas über die Umsetzung der Assistenten-Schritte in eine Abfrage lernen möchte, kann sich die fertige Abfrage auch in der Entwurfs-Ansicht (für Abfragen; → 3.1.5.2.2. Erstellen einer Abfrage in der Entwurfs-Ansicht) anzeigen lassen.



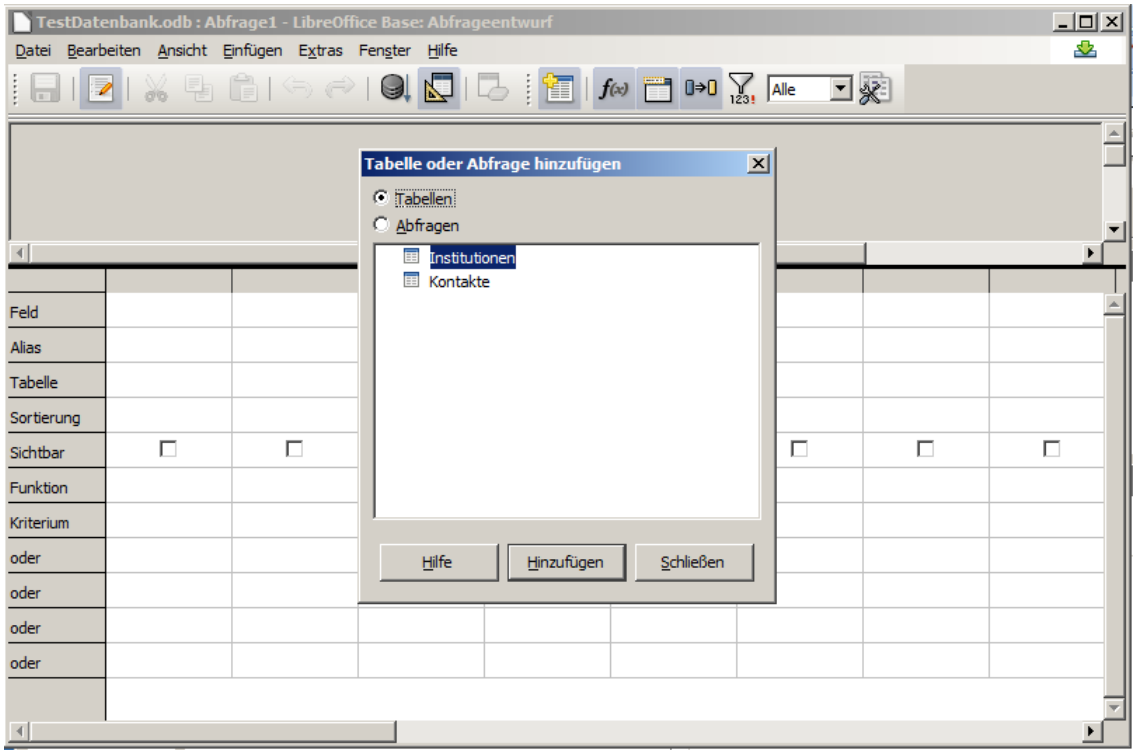
In Vorbereitung der Arbeit mit SQL ist es immer gut, sich Funktions-bekannte Abfragen usw. als fertiges SQL-Statement anzuschauen. Man erkennt dann relativ schnell das dahintersteckende Prinzip. Die SQL-Statements können dann auch als Vorlagen für eigene SQL-Anweisungen genutzt werden.

3.1.5.2.2. Erstellen einer Abfrage in der Entwurfs-Ansicht

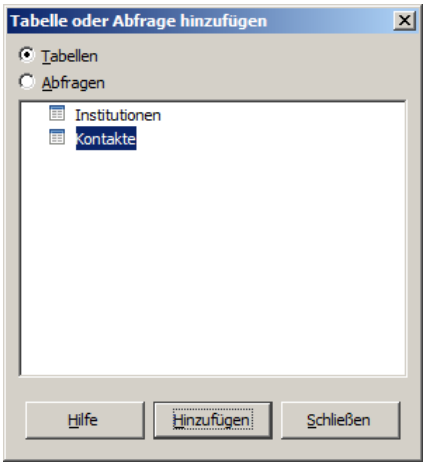
Diese Variante der Abfragen-Erstellung ist wohl die beliebteste. Es ist sehr einfach und schnell möglich sich die Abfragen zusammenzuklicken und die Ergebnisse sind meist sofort zufriedenstellend.
Für viele Zwecke ist auch eine Vorbereitung über den Assistenten sinnvoll. Später kann die Abfrage kopiert werden und dann in einer Kopie mit den Feineinstellungen experimentiert werden.
Der Aufruf erfogt im oberen Teil der Teil der Abfrage-Übersicht.



Es erscheint die Entwurfs-Ansicht mit einem "Hinzufügen"-Dialog.



Für die weitere Nutzung von Daten müssen zuerst einmal die Tabellen oder andere – schon vorhandene – Abfragen hinzugefügt werden.
Wenn man genug Tabellen und / oder Abfragen hinzugefügt hat, verläßt man den "Hinzufügen"-Dialog mit "Schließen".



Zum Schluß wieder der Hinweis, sich das zugehörige SQL-Statement anzusehen.

Aufgaben:

1. *Öffnen Sie sich die Entwurfs-Ansicht und fügen Sie sich die Tabelle "Kontakte" hinzu!*
2. *Realisieren Sie eine "Abfrage2 Kontake", indem Sie die Abfrage aus der Assitenten-Nutzung* (→ 3.1.3.1.1. Erstellen einer Tabelle mit Hilfe des Assistenten) *nachbauen!*
3. *Versuchen Sie die Abfrage2 so zu verändern, dass die nicht angezeigte Spalte "Vorname" (ganz rechts) nicht mehr benötigt wird, aber die Sortierung trotzdem erst nach der des Nachnamens (!) getätigt wird! (Hinweis!: Überlegen Sie sich, wie ein Computer(-Programm) die Entwurfs-Ansicht lesen / interpretieren wird!)*
4. *Erstellen Sie eine Abfrage, die aus den Institutionen solche heraussucht, die alphabetisch nach H folgen! Handelt es sich bei dieser Abfrage um eine Selektion, Projektion, um beides und / oder einen Verbund (wenn dieser schon besprochen wurde)? Begründen Sie Ihre Meinung!*
5. *Gesucht ist eine Übersicht aller Kontakt-Personen mit Name und Vorname, die in Orten wohnen, die nicht mit M beginnen! Handelt es sich bei dieser Abfrage um eine Selektion, Projektion, um beides und / oder einen Verbund (wenn dieser schon besprochen wurde)? Begründen Sie Ihre Meinung!*
6. *Erstellen Sie ein Liste der Institutionen mit den Personen, die aus diesen bekannt sind! Handelt es sich bei dieser Abfrage um eine Selektion, Projektion, um beides und / oder einen Verbund (wenn dieser schon besprochen wurde)? Begründen Sie Ihre Meinung!*

Bedingungs-Operatoren in BASE

Bedingungs-Operatoren in Abfragen

Operator / Symbol(e)	Benennung	Bemerkungen / Bedeutung
=	(ist) gleich	in Abfrage-Feldern wird der "="-Operator nicht angezeigt; → wird kein Operator angegeben, dann wird automatisch "=" angenommen
<>	(ist) ungleich	alle anderen Werte werden akzeptiert
<	(ist) kleiner (als)	alle Werte, die kleiner als der angegebene sind, werden akzeptiert
<=	(ist) kleiner oder gleich	erfüllt, wenn der Wert kleiner als der angegebene oder gleich groß, wie dieser, ist
>	(ist) größer (als)	alle Werte, die größer als der angegebene sind, werden akzeptiert
>=	(ist) größer oder gleich	erfüllt, wenn der Wert größer als der angegebene oder gleich groß, wie dieser, ist

logische Operatoren in Abfragen

Operator / Symbol(e)	Benennung	Bemerkungen / Bedeutung
AND	logisches UND	beide Bedingungen (links und rechts) müssen den gleichen Wahrheits-Wert besitzen (, dann ist das Ergebnis auch wieder wahr)
OR	logisches ODER	es reicht, wenn einer der beiden Bedingungen (links und rechts) wahr ist (, dann ist auch das Ergebnis wahr)
NOT	logisches NICHT Negation	der Wahrheits-Wert der folgenden Aussage wird negiert / umgedreht (aus wahr wird falsch und umgekehrt)

Platzhalter / Joker-Symbole

Zeichen / Symbol	Bedeutung	Bemerkungen / Hinweise
***** **%**	entspricht einer beliebigen (auch leeren) Zeichenkette bzw. eines beliebigen Wertes	M* bzw. M% bedeutet somit, dass alle Texte / Zeichenketten gefunden werden / gemeint sind, die mit M beginnen (es müssen auch keine weiteren Zeichen folgen – nur M reicht)
? _	steht für (genau) ein Zeichen in einer Zeichenkette	M???? bzw. M_ _ _ _ bedeutet, dass alle Texte / Zeichenketten gefunden werden / gemeint sind, die mit M anfangen und dann folgend genau (exakt) noch 4 beliebige Zeichen enthalten (Leerzeichen werden nicht als Zeichen verstanden)

Filter-Bedingungen für Abfragen

Option	**Auswirkung / Bedeutung**	**SQL-Schlüsselwort**	**Bemerkungen Beispiel(e)**
ohne	*keine*		
IST LEER	ist erfüllt, wenn das Feld einen NULL-Wert besitzt	IS NULL IS LEER	bei Options-Feldern (JA / NEIN) wird der unbestimmte Fall genutzt (also weder Ja noch Nein)
IST NICHT LEER	Negation von "IST LEER"; ist erfüllt, wenn das Feld einen (von NULL abweichenden) Wert besitzt	IS NOT NULL IS NOT LEER	
LIKE **WIE**	Übereinstimmung; ist erfüllt, wenn das Feld den entsprechenden Wert besitzt	LIKE	WIE 'Muster*' oder WIE 'Muster????' (liefert: z.B. "Mustermann" und "Musterfrau" zurück) bei Zahlen besser **=** benutzen
ZWISCHEN *min* **UND** *max*	im Intervall; ist erfüllt, wenn der Feldinhalt zwischen den Angaben ***min*** und ***max*** liegt	BETWEEN *min* AND *max*	
IN (*Liste***)**	Entsprechung; Enthaltung ist erfüllt, wenn das Feld mit einem Wert aus der (Semikolon-getrennten) übereinstimmt		IN (3; 6; 12; 24) IN ('Moskau'; 'Berlin'; 'New York')
NICHT …	Negation; negiert den folgenden Ausdruck	NOT …	
= WAHR	Validierung; ist erfüllt, wenn der Feldinhalt ***wahr*** enthält	= TRUE	
= FALSCH	Falsifizierung ist erfüllt, wenn der Feldinhalt ***falsch*** enthält	= FALSE	
EXISTS(SELECT-Anweisung**)**	falls mind. ein Datensatz existiert, dann gibt die Funktion TRUE zurück, sonst FALSE		

3.1.5.2.3. Berechnungen in einer Abfrage

In Abfragen fehlen oft einige Zusatz-Daten, die uns das Leben unwahrscheinlich einfacher machen würden. Wieviele Datensätze sind denn nun in der Abfrage enthalten, welcher Durchschnitt ergibt sich z.B. für die Umsatz-Zahlen usw. usf.?
Dafür brauchen wir kleine Rechnungen / Funktionen, die quasi parallel oder im Hintergrund die abgefragten Daten auswerten.
Wenn wir Berechnungen zu Abfragen hinzufügen, dann handelt es sich sachlich eigentlich schon um einen Bericht (→ 3.1.5.3. Berichte). Heute wird die Trennung nicht mehr so hart gemacht.

statt des Feldnamens (Attributes) werden Formeln mit den Feldnamen (z.B. "Nettopreis") ohne führendes Gleichheitszeichen eingegeben (z.B. zur Berechnung des Bruttopreises: "Nettopreis" * 1,19)
es muss dann ein Alias vergeben werden, da sonst die berechnete Spalte keine Überschrift hätte, sie ist ja immer neu

Rechen-Operatoren in Abfragen

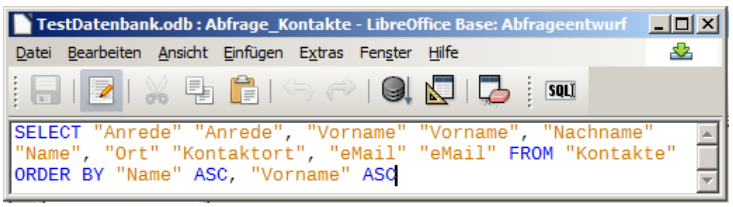
Operator / Symbol(e)	Benennung	Bemerkungen / Bedeutung
+	Addition	fortlaufende Addition z.B. über Konstrukt: Gesamt = Gesamt **+** Feldwert möglich
-	Subtraktion	fortlaufende Subtraktion z.B. über Konstrukt: Bestand = Bestand **-** Feldwert möglich
*****	Multiplikation	fortlaufende Multiplikation z.B. über Konstrukt: Produkt = Produkt ***** Feldwert möglich
/	Division	fortlaufende Divisiion z.B. über Konstrukt: Anteil = Anteil **/** Feldwert möglich

Funktionen in Abfragen

Option / Funktion	Auswirkung / Bedeutung	SQL-Schlüssel-wort	Bemerkungen
ohne	*keine*		
Durchschnitt	berechnet das (arithmetrische) Mittel (Mittelwert) des Feldes (Attributes)	AVG	
Anzahl	zählt die Datensätze, die in der Abfrage auftauchen	COUNT	COUNT(*) .. es werden alle Datensätze (in der Abfrage) gezählt COUNT(*Spalte*) .. es werden die Datensätze gezählt, bei denen die ***Spalte*** einen Nicht-NULL-Wert enthält
Maximum	ermittelt den größten Wert des Feldes (Attributes)	MAX	
Minimum	ermittelt den kleinsten Wert des Feldes (Attributes)	MIN	
Summe	berechnet die Summe der Feld-Werte (Attribut-Werte)	SUM	
Gruppiert	gruppiert die Datensätze nach den Werten im ausgewählten Feld (Attribut)	GROUP BY	

3.1.5.2.4. Erstellen einer Abfrage mittels SQL

Diese Methode zum direkten Erstellen einer Abfrage ist mehr was für Profis oder Programmierer. Für einfache Abfragen ist es aber ein gleichwertiges Mittel. Komplizierte Abfragen lassen sich besser über den Assistenten (→ 3.1.3.2.1. Erstellen einer Abfrage mit dem Assistenten) oder die Entwurfs-Ansicht (→ 3.1.3.2.2. Erstellen einer Abfrage in der Entwurfs-Ansicht) zusammenstellen und austesten.
Als Beispiel sei hier der Code für die mit dem Assisten erstellte Abfrage aufgezeigt.
Man kann den Text fast verstehen.
Mehr dazu im SQL-Teil dieses Skriptes (→ 5. Datenbanken - SQL; → 5.1.x. CREATE VIEW).
Zum langsamen Herantasten an SQL kann man sich zuerst einmal den SQL-Code von fertigen – funktionierenden – Abfragen ansehen.



Später ist man vielleicht schneller, wenn man nur noch den Code eingibt. Das setzt aber auch leider immer voraus, dass man die ganzen Namen von Tabellen, anderen Abfragen und den Attributen im Kopf oder auf einer Übersicht hat. Diese werden immer – exakt geschrieben – für einen funktionierenden SQL-Code gebraucht.
Letztendlich lesen sich SQL-Codes fast wie eine umgangssprachliche – wenn auch englische – Arbeits-Aufforderung. Dieses war ja auch ein erklärtes Ziel bei der Entwicklung von SQL.

Aufgaben:

1. *Verändern Sie im obigen SQL-Statement das zweite Auftreten eines Attributes in die englische Bezeichnung (z.B. "Vorname" durch "Firstname")! Lassen Sie das Statement dann ausführen und schauen Sie sich das Ergebnis genauer an! Welche Funktion erfüllt die "Attribut-Wiederholung"?*
2. *Überlegen Sie sich, ob man das erste Auftreten eines Attribut-Namens ebenfalls ändern kann? Tragen Sie Ihren Standpunkt vor den anderen Kursteilnehmern vor! Diskutieren Sie den Vorschlag / die Vorschläge!*
3.

komplexe Aufgaben (zu Abfragen):

1. *Erstellen Sie eine Abfrage, die zur Bezeichnung der Institution noch die verbundenen Kontakte mit Nachnamen und Vornamen zusammenstellt!*
   Geben Sie ob es sich um eine Selektion, Projektion oder einen Verbund bzw. um Kombinationen davon handelt! Begründen Sie Ihre Aussage!
   Notieren Sie sich die zugehörige SQL-Anweisung mit Platz zwischen den Zeilen! Kommentieren Sie die verschiedenen Teile der SQL-Anweisung!
2. *Stellen Sie eine Abfrage zusammen, die Titel, Vorname, Nachname und eMail-Adresse der Kontakte (aufsteigend) sortiert nach den Institutions-Bezeichnungen (mit anzeigen!) darstellt!*
   Notieren Sie sich die zugehörige SQL-Anweisung! Durch welchen SQL-Teil wird die Sortierung realisiert! Unterstreichen Sie diesen!
3. *Erstellen Sie eine neue Abfrage, wie bei 2., bei der eine 2. Sortierung (aufsteigend) nach dem Nachnamen erfolgt!*
   Notieren Sie sich die zugehörige SQL-Anweisung!
4. *Erstellen Sie eine weitere Abfrage, wie bei 3., in der nur Einträge angezeigt werden, bei denen der Textteil "Muster" im Nachnamen des Kontaktes auftaucht! (Testen Sie auch mal den Unterschied, wenn "muster" gesucht wird!)*
   Notieren Sie sich die zugehörige SQL-Anweisung!
5. *Kopieren Sie sich die Abfrage von Aufgabe 4 und verändern Sie diese so, dass alle Einträge herausgesucht werden, bei denen der Nachname auf "er" endet!*
6. *Erstellen Sie nur mit SQL eine Abfrage, die den Nachnamen, Vornamen und deren Telefonnummer sowie die eMail der Institution anzeigt!*

für die gehobene Anspruchsebene:

7. *Erweitern Sie die SQL-Anweisung von 6. um eine Sortierung nach den Nachnamen!*
8. *Ermitteln Sie, welche Kontakte eine Verbindung zu einer Institution haben, bei denen in den eMail-Adressen der Textabschnitt "web" vorkommt!*
9. *Stellen Sie eine vollständige SQL-Anweisung zusammen, die alle Institutionen mit all ihren Daten mit allen zugehörigen Kontakten (ebenfalls mit allen Daten) in einer Tabelle wiedergibt!*

Aufgaben für alle:

10. a) *Geben Sie die folgende SQL-Anweisung ein! Überlegen Sie sich dabei, was diese machen wird! Notieren Sie Ihre Vermutung!*

```
INSERT INTO "Institutionen" VALUES(7,'auto e.V.',NULL,'post@auto.ev','');
```

    b) *Führen Sie die SQL-Anweisung aus! Prüfen Sie Ihre Vermutung anhand des neuen Datenbestandes (ev. gegen die Tabellen auf Papier)! Sollte Ihre Vermutung nicht richtig gewesen sein, dann berichtigen Sie die Vermutung!*
11. *Geben Sie die folgende SQL-Anweisung ein! Überlegen Sie sich dabei, was diese machen wird!*

```
UPDATE "Kontakte" SET "Adresse"='Blumenweg 14' WHERE "Nachname"='Fehler';
```

12. *Überlegen Sie sich eine SQL-Anweisung, die Ihre Orts-Tabelle um einen Ort (45678 Neuberg) ergänzt! Sie können diese auch ausprobieren!*

3.1.5.3. Berichte

Unter Berichten verstehen wir Ergebnis-Auskopplungen meist als Ausdruck oder eben als PDF-Dokument). Dabei soll der Zustand der Datenbank – bzw. seiner Daten – zu einem bestimmten Zeitpunkt dokumentiert werden. Ein klassisches Beispiel ist eine Klassen-Liste zum Anfang des Schuljahres oder eine Umsatz-Zusammenstellung zum Quartals- oder Jahres-Ende.
Die in einem Bericht erzeugten Daten sollen im Allgemeinen dann später durch weitere Daten nicht verändert werden. Niemanden nützt ein Quartals-Bericht z.B. vom II. Quartal, in dem auch Daten vom III. mit eingeflossen sind. Auch nachträgliche Veränderungen (außer echte Fehler-Korrekturen) sollen dadurch ausgeschlossen werden.

3.1.5.4. Formulare

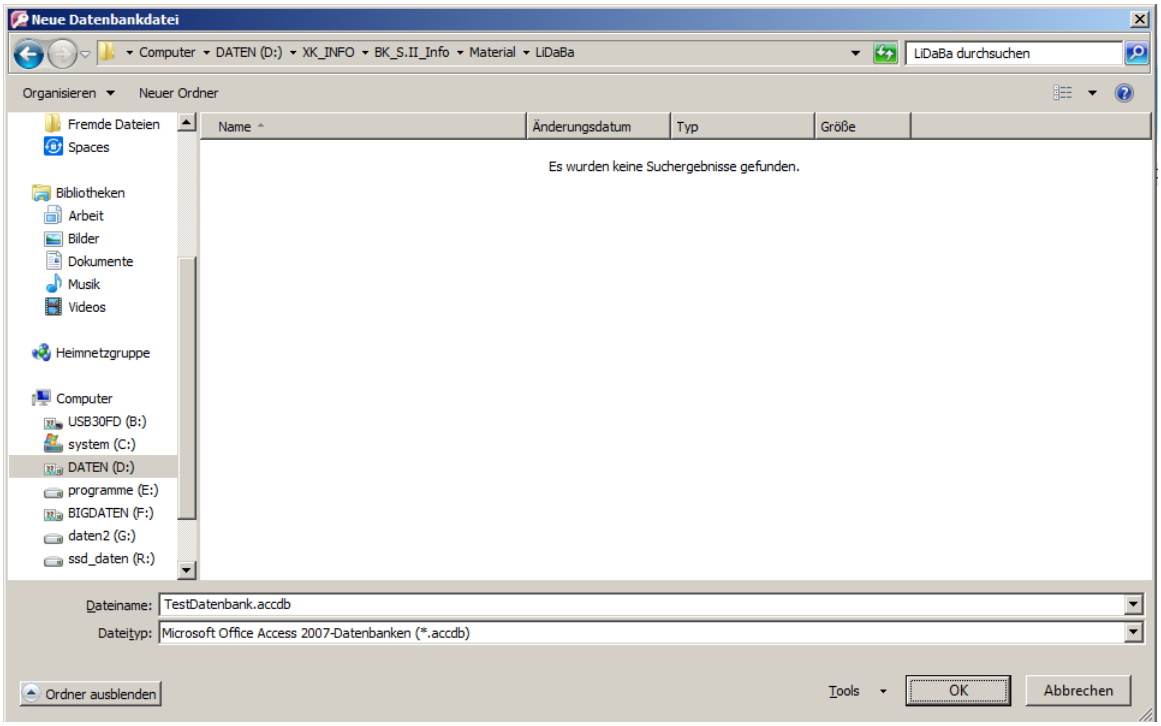
Tabellen sind übersichtlich und stellen die Daten sehr kompakt dar. Für den normalen Nutzer sind sie aber eher schwierig zu benutzen. Schnell ist man in Zeile oder Spalte verrutscht und schon stehen die falschen Daten irgendwo.
Besonders schwierig ist die Arbeit mit Fremdschlüsseln. Der Nutzer müsste sie entweder alle kennen oder ständig irgendwo nachschlagen. Das wäre sehr Zeit-aufwändig und wohl auch Fehler-anfällig.
Nutzer-freundlicher sind die üblichen Formulare / Dialoge, die wir zur Genüge aus Windows kennen. Formulare sind die modernen Ein- und Ausgabe-Hilfen für die normalen Datenbank-Benutzer. Sie müssen auch nicht unbedingt verstehen, wie alles in der Datenbank abläuft. Solche Dinge, wie Primär- und Sekundär-Schlüssel können wir mit Formularen gut vor dem unbedarften Nutzer verbergen.
Fremdschlüssel werden z.B. in Form von Auswahl-Feldern bereitgestellt, die nicht den Fremdschlüssel selbst, sondern ein beschreibendes Attribut anzeigt. Damit können die Nutzer dann auch etwas anfangen.

3.1.6. Datenbanken mit ACCESS

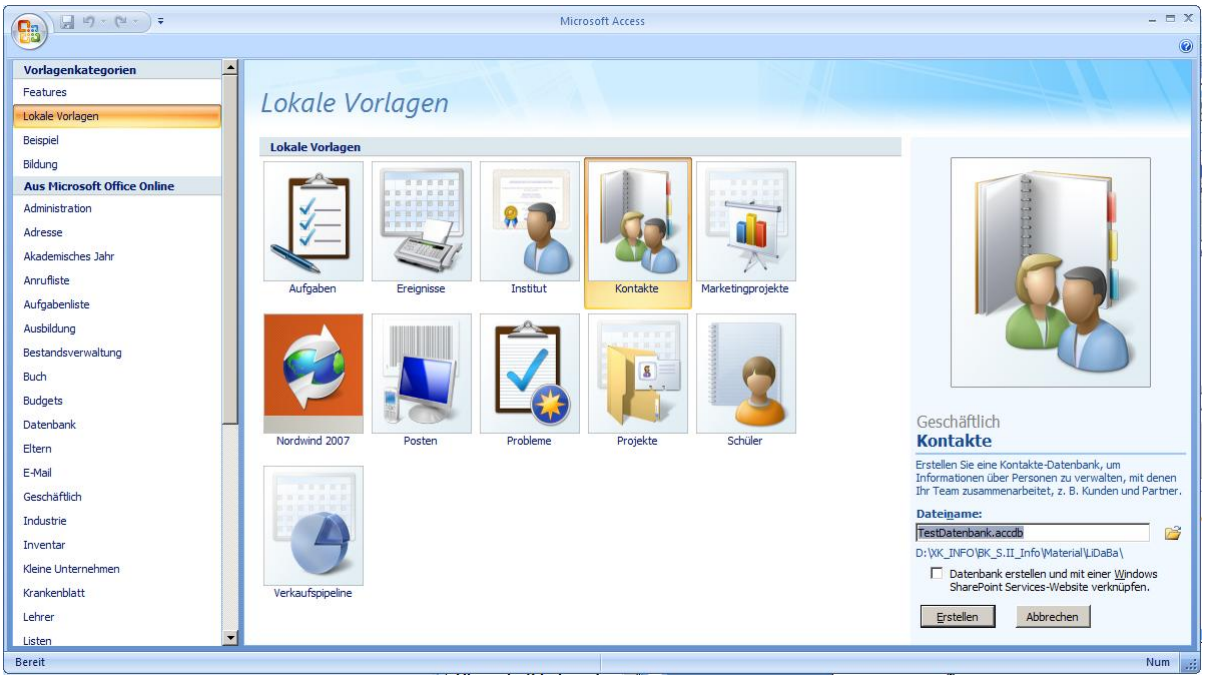
Die Nutzung von Datenbank-Komponenten aus Office-Produkten wird im neuen Rahmenplan "Informatik" (Sek.II; für MV) abgelehnt.
Für Einsteiger-Projekte sind sie aber sehr gut geeignet!

Die neueren Versionen (ab 2007) von ACCESS sind sicher für unbedarfte Nutzer ohne Datenbank-Theorie-Hintergrund ein nutzbares Tool. Viele Tools und Assistenten führen Anfänger bzw. Datenbank-Beginner gut durch das Programm. Allerdings muss man schon wissen, was man will. Ohne theoretische Vorkenntnisse kommt auch in ACCESS nicht weit.
Wer aber strukturiert Datzenbank-Entwürfe umsetzen will, der braucht erst einmal viel Erfahrung und Pionier-Geist. So manche wichtige Funktion ist in die hintersten Menüband-Gefilde verstoßen worden. Auch die Benennung der Menü-Punkte ist sehr uneinheitlich und wenig hifreich, wenn man etwas sucht.
Nach dem Start von ACCESS erscheint ein graphisch aufgepeppter "Erste Schritte"-Dialog.

ACCESS bietet von sich aus nun die ersten Vorlagen für Datenbanken an bzw. ganz rechts die letzten benutzten DB-Dateien.
Um jetzt nicht gleich die Nachhause-Telefonieren-Funktion von ACCESS zu aktivieren und irgendwelche Vorlagen von Microsoft-Online zu beziehen, nutzen wir erst einmal den Bereich "Lokale Vorlagen".
Hier wählen wir "Kontakte" als gewünschte Vorlage und speichern in einem beliebigen Ordner mit einem passenden Namen ab. Die aktuelle Datei-Endung für ACCESS-Datenbanken lautet *.ACCDB.



Nun bleibt noch das eigentliche "Erstellen", d.h. von der Vorlage wird eine konkrete Kopie (Instanz) erstellt.



Ältere Versionen von ACCESS entsprechen dem Handling, wie wir es bei der BASE-Datenbank vorgestellt haben. Leider versucht ACCESS das Fahrrad neu zu erfinden und verkompliziert das Vorgehen gleich von Anfang an.

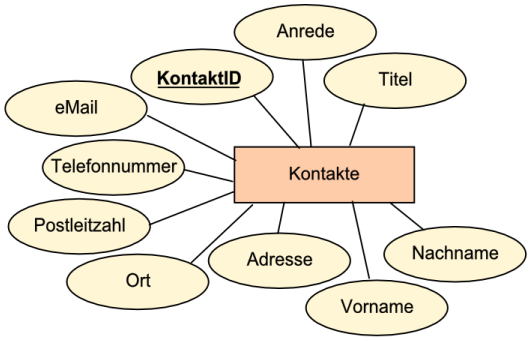
Tips und Hinweise für Arbeits-Erleichterungen in neuen ACCESS-Versionen

- wenn man zuerst die Detail-Tabellen anlegt, dann kann man schon im Entwurf der übergeordneten Tabellen Nachfrage-Spalten erstellen, die sich aus den Detail-Tabellen speisen, aber nur den (Fremd-)Schlüssel speichern
- Abfragen lassen auch Daten-Eingaben zu; damit lassen sich schnell Daten eingeben, vor allem, wenn nur bestimmte Felder benutzt werden sollen / ausreichen
-

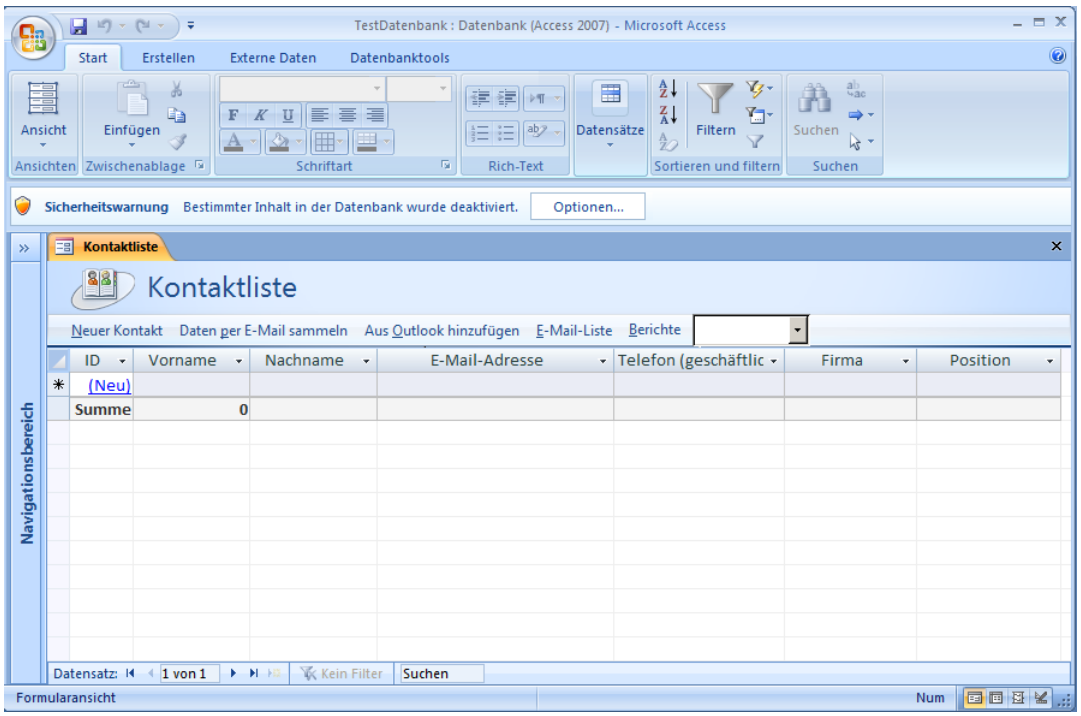
3.1.6.1. Tabellen

ACCESS leitet uns nach dem ersten Öffnen einer Datenbank gleich in die Daten-Eingabe. Das wollen wir auch gleich tun, um uns an die Arbeitsweise zu gewöhnen.

Ziel soll auch hier eine ähnliche Tabelle, wie in der BASE-Lösung (→ 3.1.5.1. Tabellen). Das nebenstehende ERD einer ersten Teil-Lösung ist die Orientierung für uns.

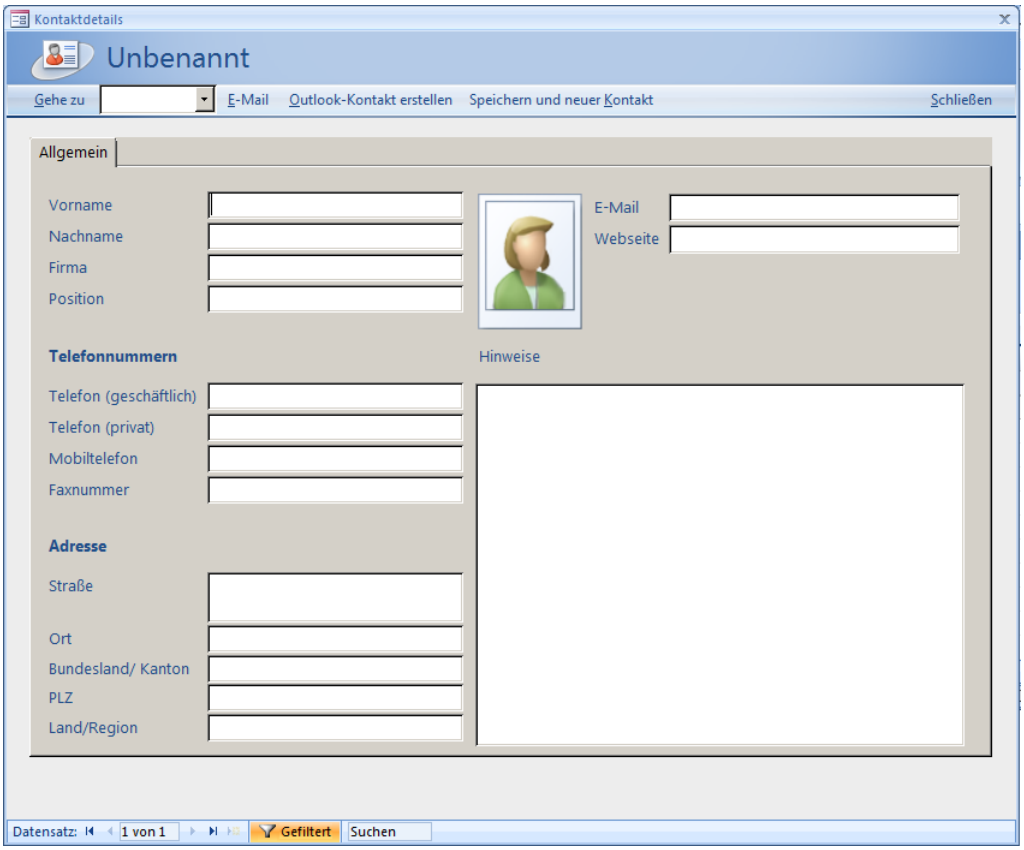


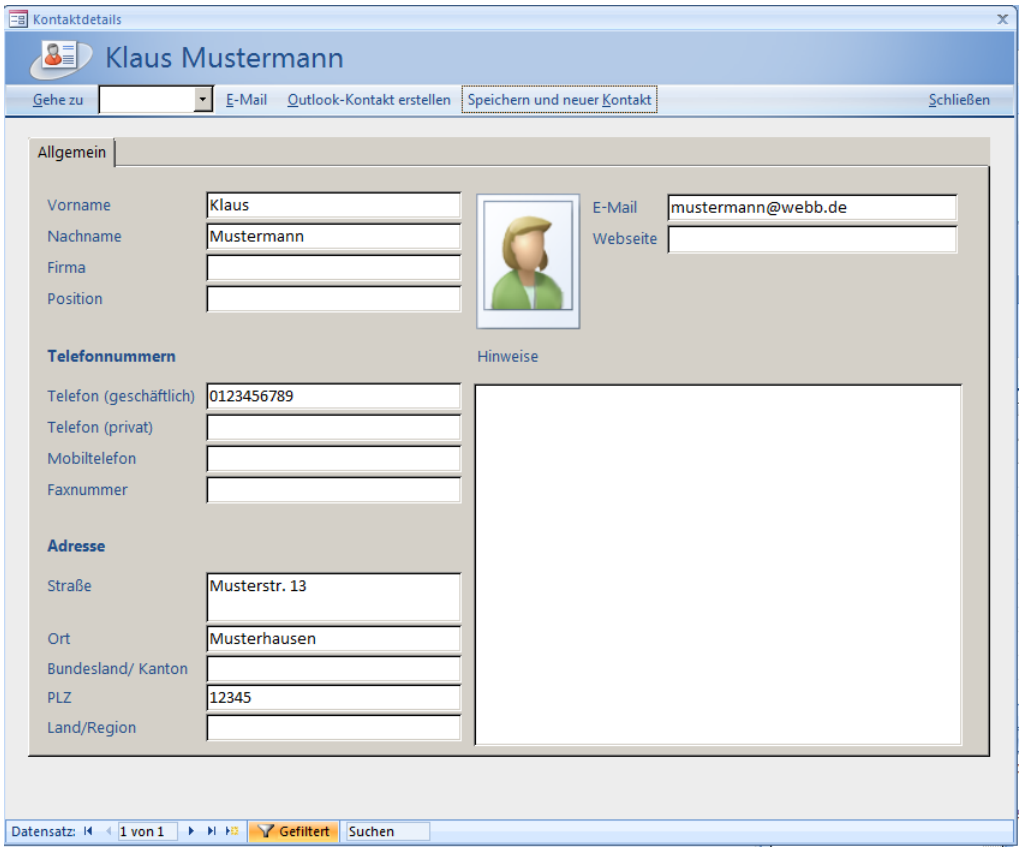
3.1.6.1.1. Eingeben von Daten in eine Tabelle



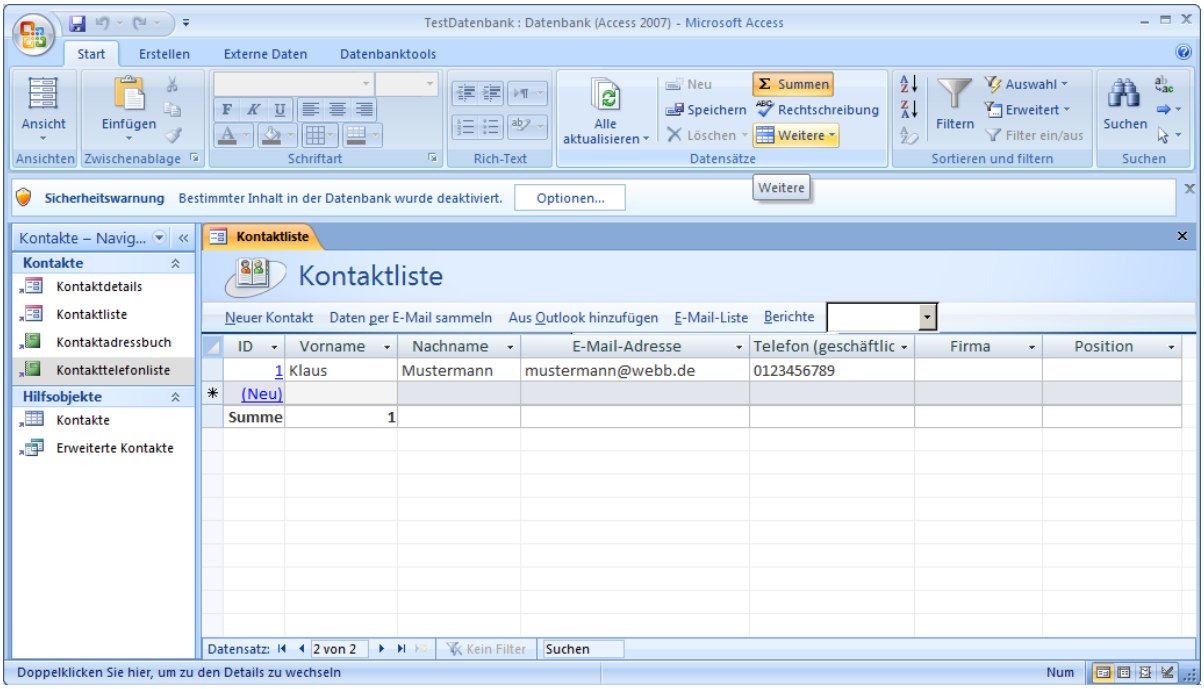
Zu existierenden Tabellen (oder auch Abfragen) lassen sich in ACCESS automatisch einfache Formulare erstellen (→ 3.1.6.4. Formulare). Mit diesen ist dann eine Nutzer-freundliche Dateneingabe – aber auch –Anzeige – möglich.

Die fesrtigen Beispiel-Datenbanken beinhalten auch schon Bedienungs-freundliche Formulare. Alle notwendigen Einstellungen und Funktionen sind schon integriert.

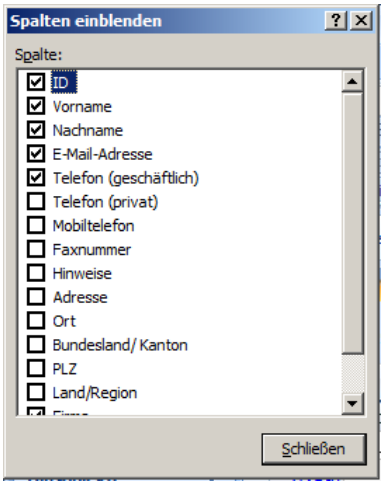


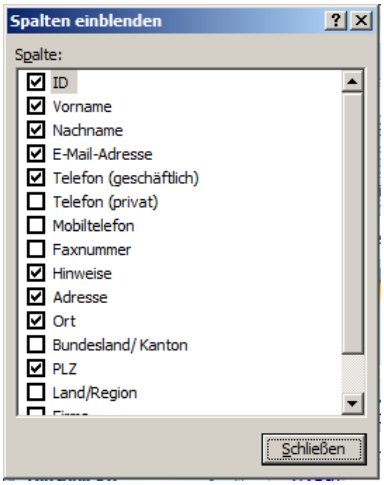


Jetzt können mit dem Formular alle Daten eingegeben werden, für die ein passendes Feld vorgesehen ist. Nach jeder Eingabe eines Kontaktes geht man auf "Speichern und neuer Kontakt" oder "Schließen".
Wenn wir uns dann die Tabelle ansehen, sind die vielen eingebenen Daten trotzdem nicht sichtbar geworden. Das diese aberimmer noch da sind, können wir bei der Nutzung des Formulares über die Datensatz-Navigation unten links einsehen.
Warum die Datenfelder nur eingeschränkt angezeigt werden, erschliesst sich wohl nur Microsoft. Wir können uns aber alle Felder einblenden – und natürlich auch wieder ausblenden – wenn im Menüband "Start" im Menübandbereich "Datensätze" "Weitere" ausgewählt wird.

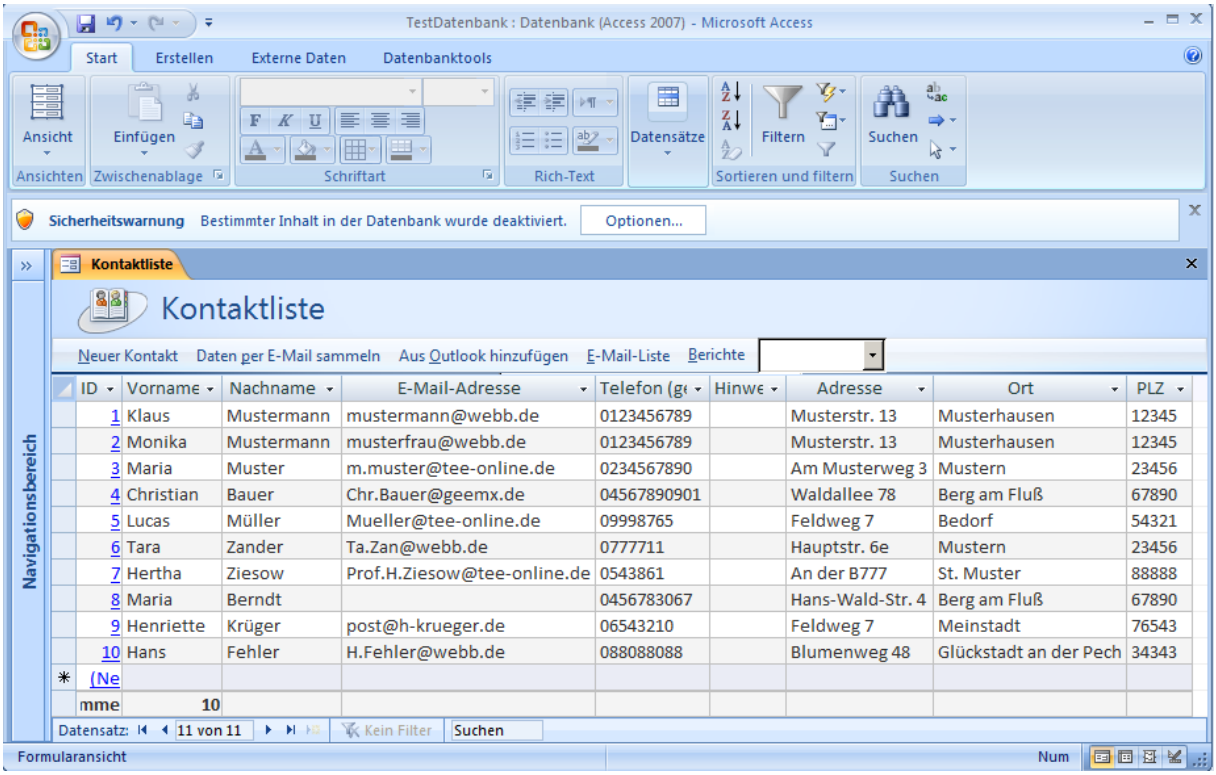


Über das Menü können nun Spalten (Felder, Attribute) eingeblendet werden. Zuerst werden nur die Spalten angezeigt, die derzeit in der Tabellen-Ansicht aktiv sind.
Durch hinzu- oder raus-klicken kann die Auswahl jetzt für unsere Bedürfnisse angepasst werden.
Zwar sind immer noch nicht alle Informationen untergebracht, aber dazu kommen wir später.





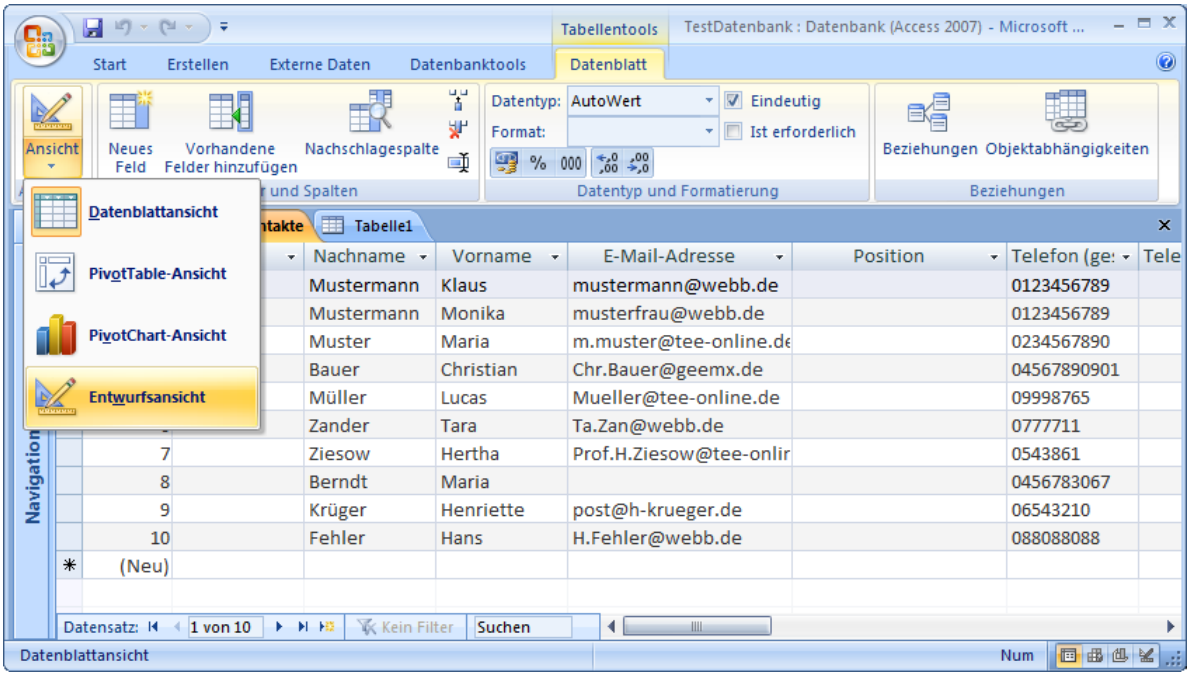
Nach dem "Schließen" ist die Tabellen-Ansicht um die gewünschten Spalten erweitert und auch die eingegebenen Daten werden angezeigt.
Ob man dann die Daten lieber über die Tabelle oder die Formular-Ansicht erledigt, ist Geschmackssache.
Erfahrungsgemäß geht die Arbeit in der Tabelle etwas zügiger von der Hand, dafür ist das Formular übersichtlicher.



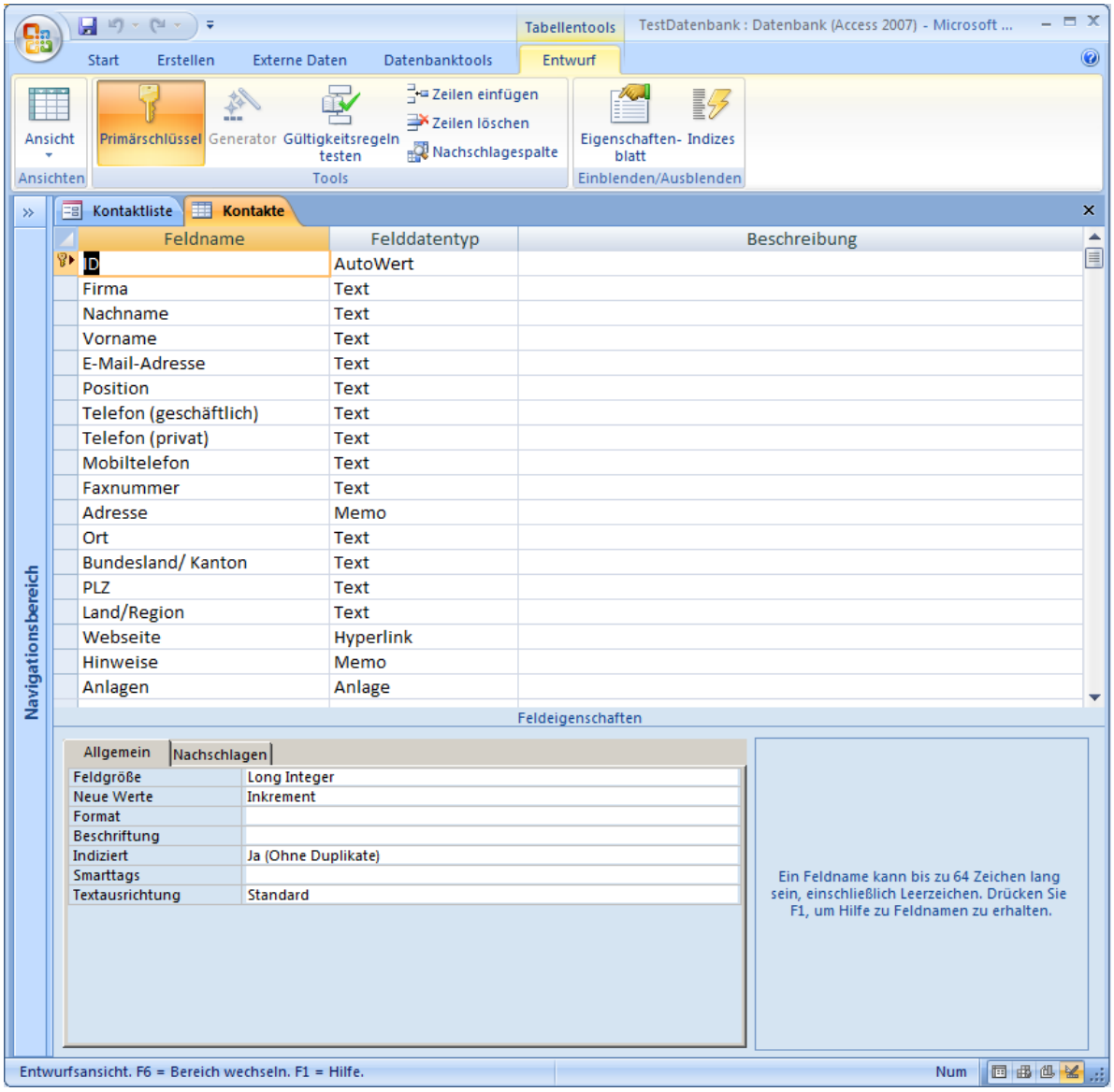
Letztendlich bleiben nun die fehlenden Felder "Anrede" und "Titel".
Wer sich mit der Datenbank BASE beschäftigt hat, weiss, dass es eine Entwurfs- oder Struktur-Editier-Ansicht gibt. Genau diese werden wir jetzt nutzen, um die fehlenden Felder zu ergänzen und die weniger geeigneten Spalten-Namen korrigieren.

3.1.6.1.2. Ändern des Relationsschema einer Tabelle – Korrekturen am Entwurf

Nicht dass man diese Ansicht irgendwo gleich direkt und offensichtlich findet, sie ist im Menüband "Tabellentools" im Band "Datenblatt" unter "Ansicht" versteckt.



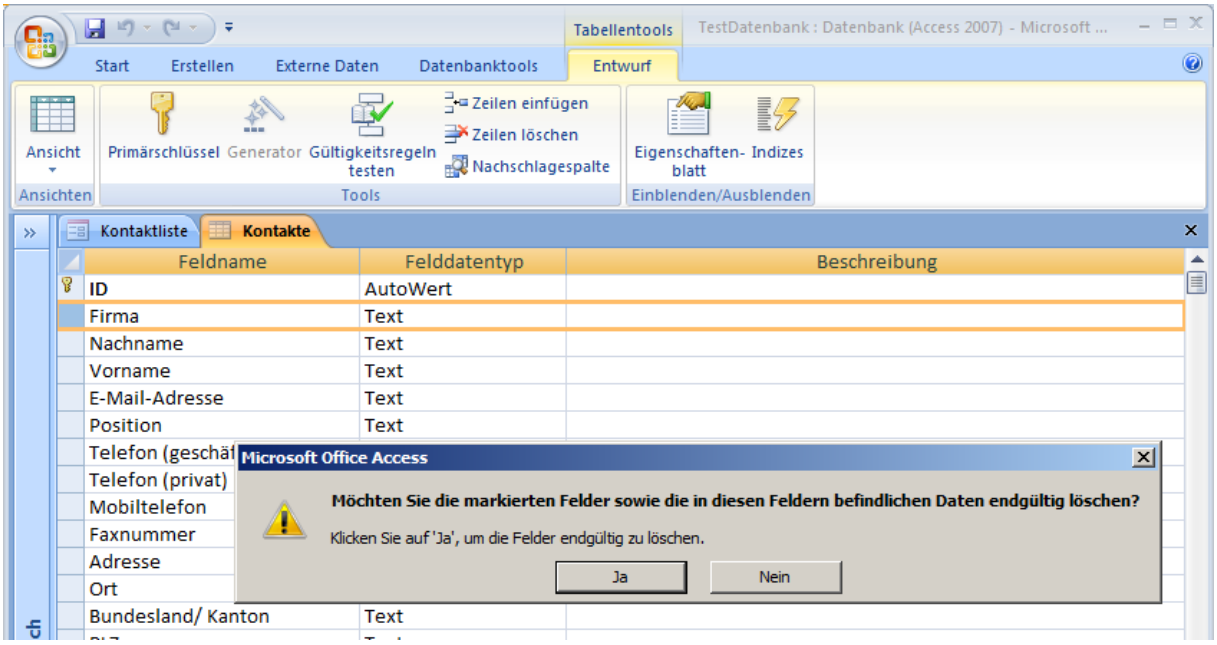
Die Tabellenstruktur ist klassisch aufgestellt. Neben dem frei wählbaren Feldnamen (Attribut, Spalten-Überschrift, Spaltenname) wird für jedes Feld noch der Datentyp gefordert. Die Beschreibung ist optional und kann bei kleinen, übersichtlichen und mit sprechenden Feldnamen versehenen Datenfeldern beruhigt weggelassen werden.



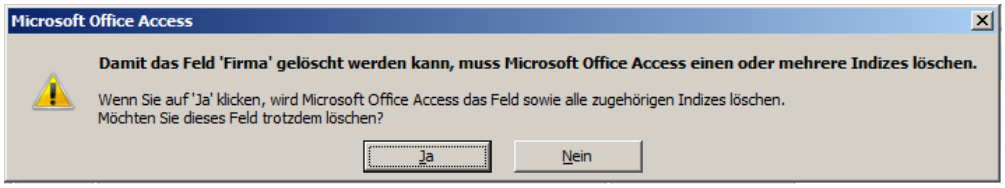
über die rechte Maustaste lässt sich die unerwünschte Zeile löschen
die nachfolgende Bestätigung sollte man sich immer gut überlegen und nicht routiniert auf "Ja" klicken
die Daten sind dann nämlich wirklich gelöscht

<u>Vorgehen bei Umgestalten / Neuerstellen von Spalten (aus alten)</u>

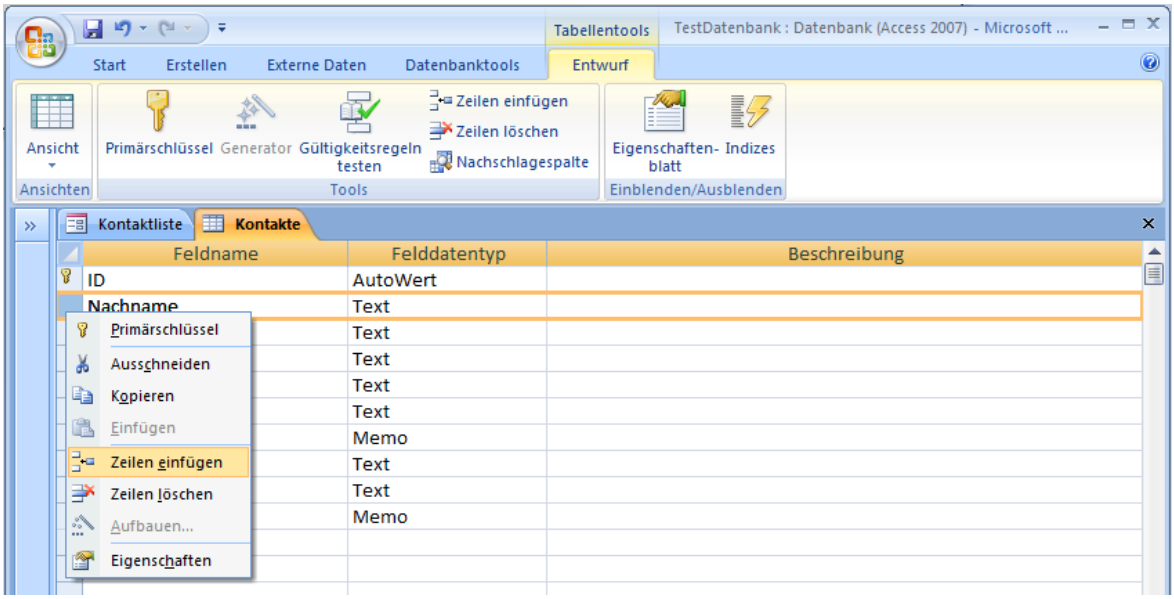
- Erstellen der neuen Spalte(n)
- Übernahme der Daten aus der alten Spalte in die neue(n) Spalte(n)
- Löschen der alten Spalte

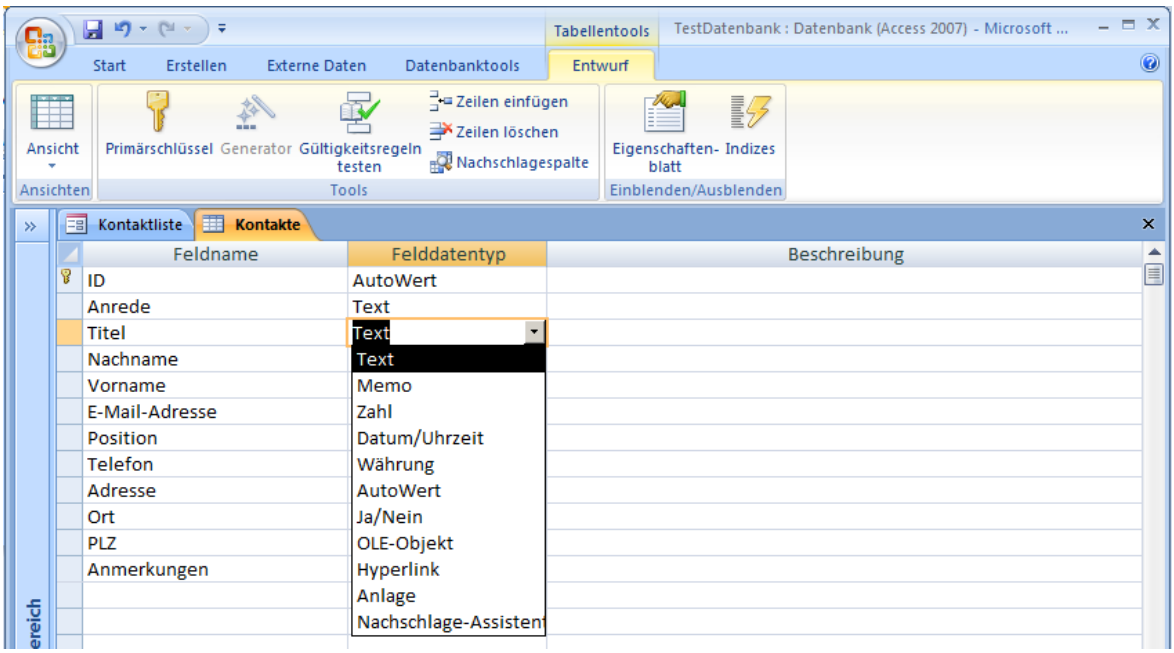


U.U. kommt noch eine 2. Bestätigungs-Anfrage nach einem zu löschenden Index, das kann man dann beruhigt bejahen.

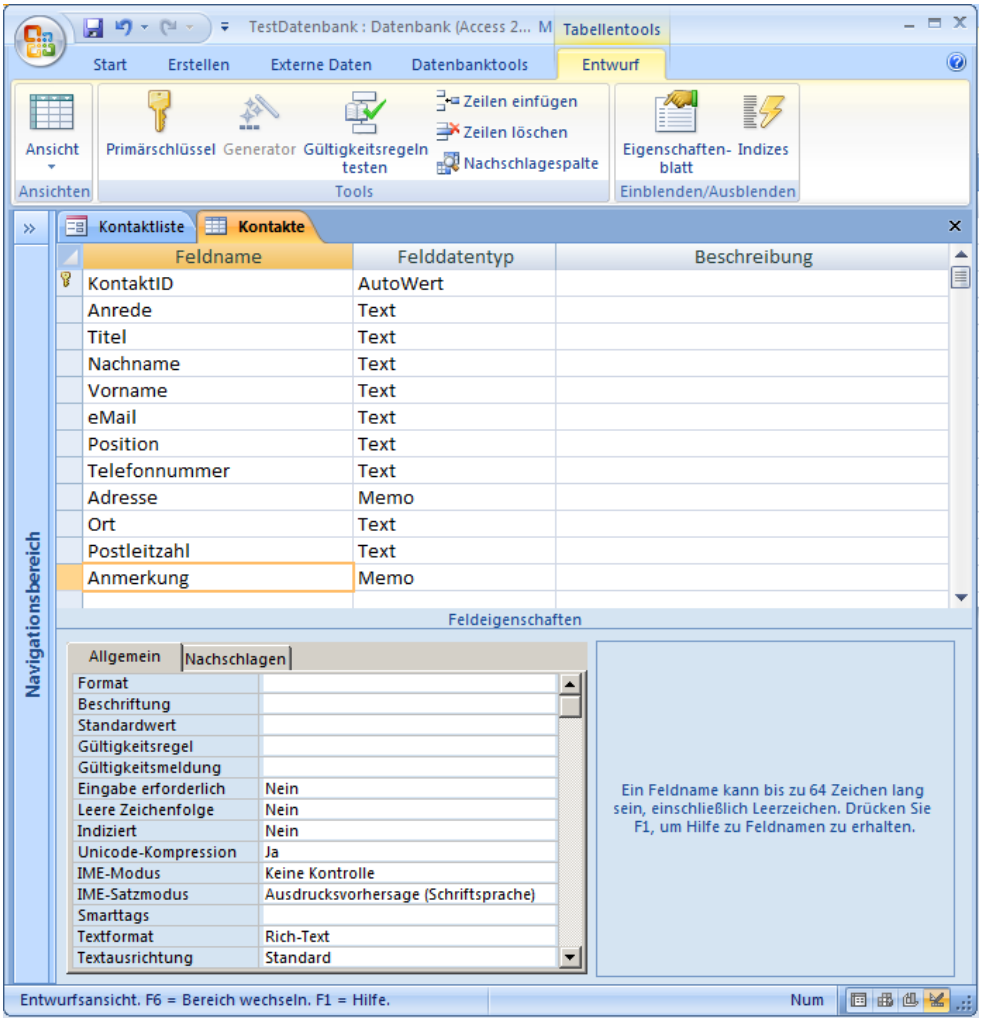


Die Indexes sind nur Dabenbank-interne Hilfs-Tabellen, um Zugriffe auf größere Datenbestände / Tabellen zu beschleunigen. Indexies werden entweder automatisch aktualisert oder neuaufgebaut. Bei selbst erstellten Indexes sollte man ev.die Aktualisierung oder den Neuaufbau händisch anstoßen.



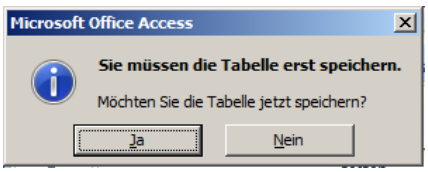


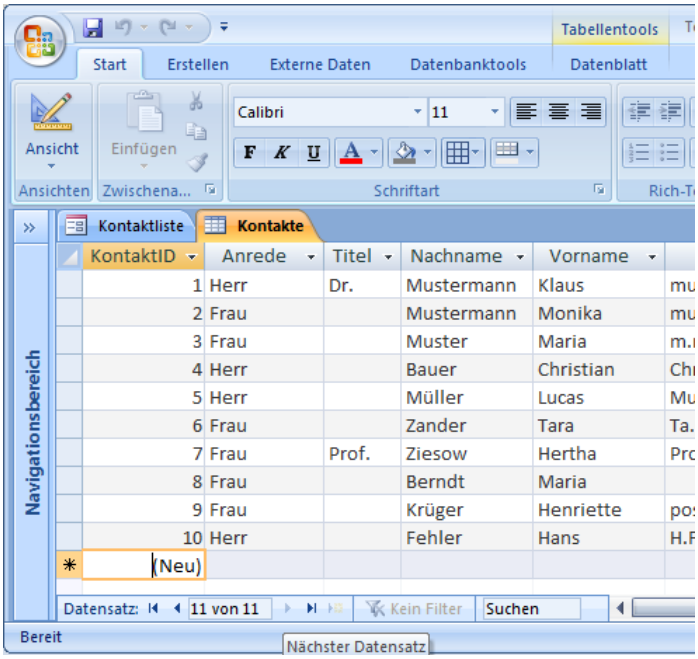
Feldnamen lassen sich direkt ändern
insgesamt sollte die Struktur dann wie folgt aussehen
(wir benötigen die gleiche Struktur in dieser ACCESS-Datenbank wie die in der BASE-Datenbank. Nur so können wir die späteren Zugriffe über SQL bzw. Programmier-Schnittstellen sinnvoll behandeln oder gegenüberstellen
ansonsten macht es praktisch nichts aus, ob ein Feld "Telefon" oder "Telefonnummer oder gar "Klingelling" heisst, für interne Zwecke sollte man nur verständliche Namen nutzen



Die im unteren Bereich sinchtbaren Feldeigenschaften (bei "Allgemein") sind für Spezialisten interessant. Für unsere Anwendungen sollte man dort nicht so viel herumschrauben. Ev. kann man die ""Feldgröße" bei Text-Felder anpassen. Das spart u.U. Speicherplatz – ist aber bei dem Platz-Angebot moderner Datenträger eher nicht notwendig. Da ärgert man sich dann eher, wenn man den Nachnamen auf 30 Zeichen beschrägt hat und dann einer daherkommt, der einen Namen mit 31 Zeichen hat.

Der geänderte Entwurf muss nun noch gespeichert werden, um ihn zu aktivieren.
Als Letztes müssen noch die fehlenden Daten ergänzt werden.



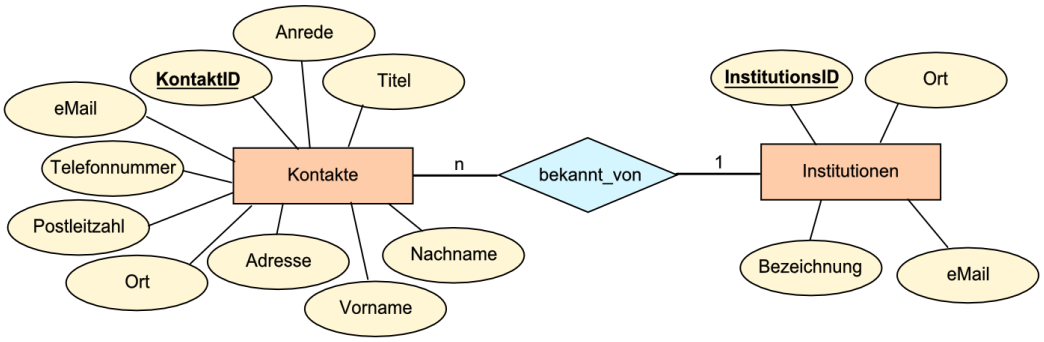


Aufgabe:

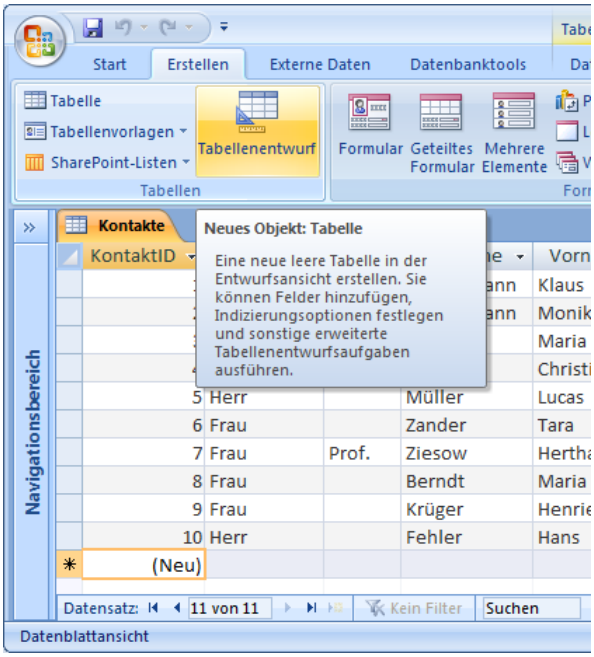
1. ***Geben Sie die Daten aus der obigen Tabellen bzw. Tabellen-Ausschnitten in die Tabelle "Kontakte! ein!***
   (Sie können auch die vollständige Tabelle aus dem Abschnitt 3.1.5.1.4. verwenden!

3.1.6.1.3. Erstellen einer neuen Tabelle über den Tabellenentwurf

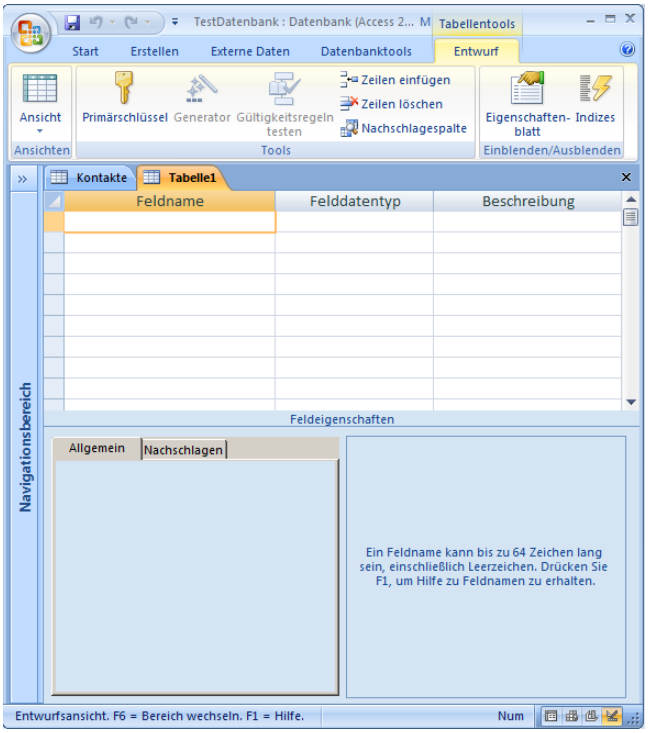
Die Datenbank soll nun um die zweite Tabelle "Institutionen" erweitert werden.

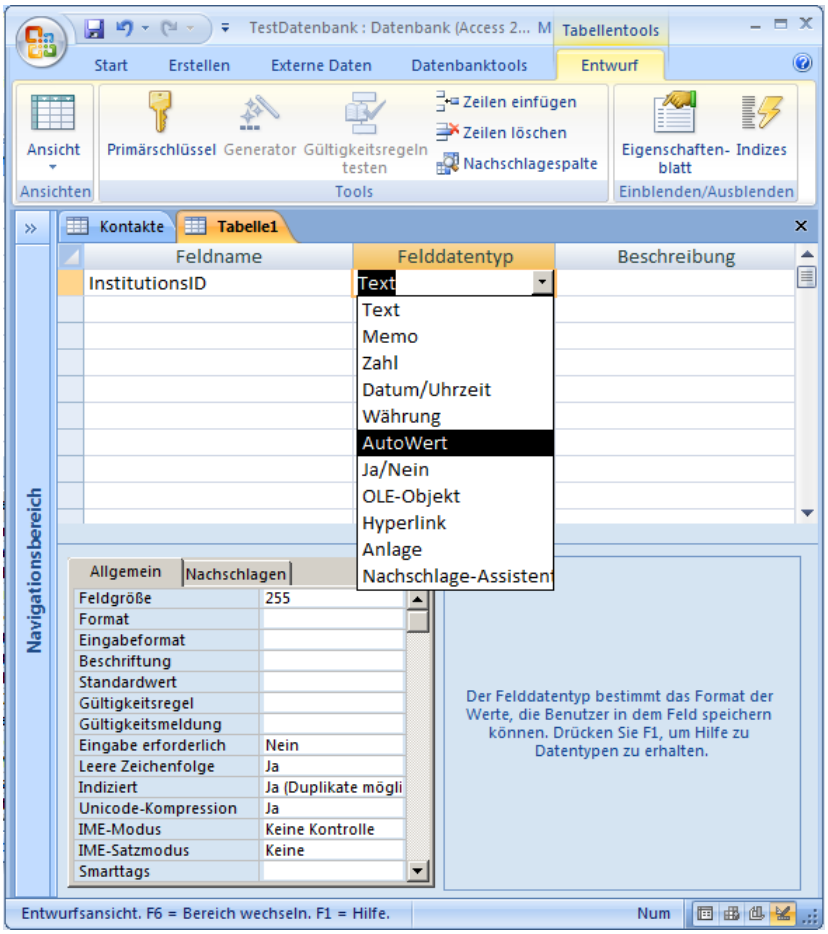


Zum effektiveren Arbeiten gehen wir gleich über den "Tabellenentwurf" aus dem Menüband "Erstellen" "Tabellen" an die Aufgabe heran.

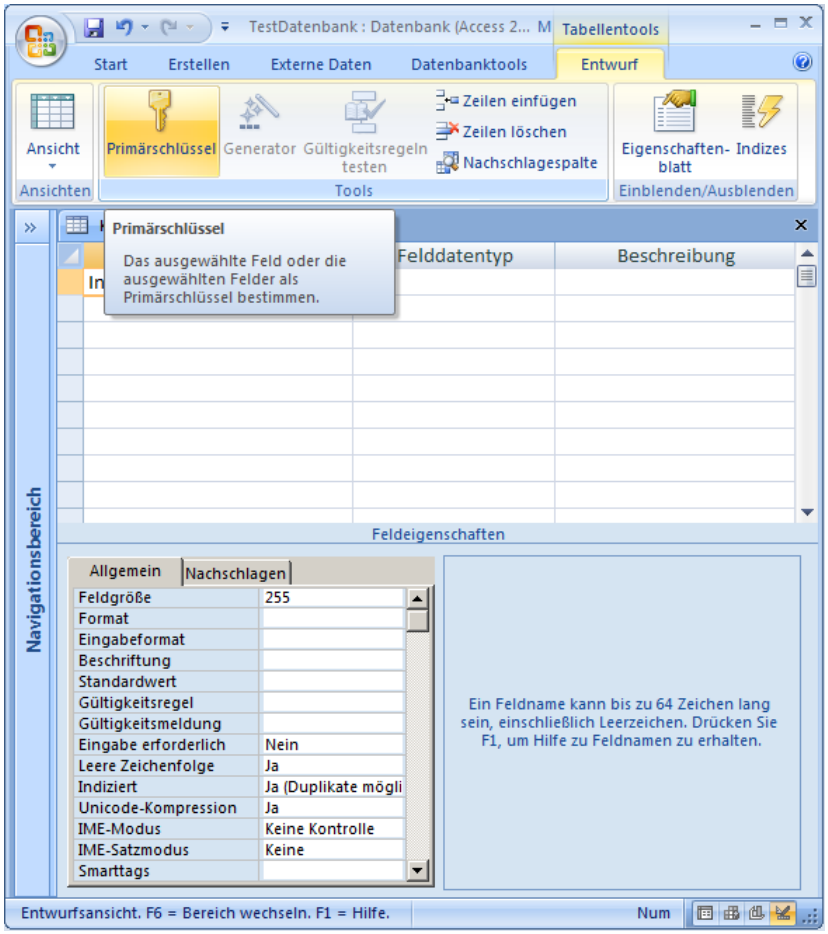


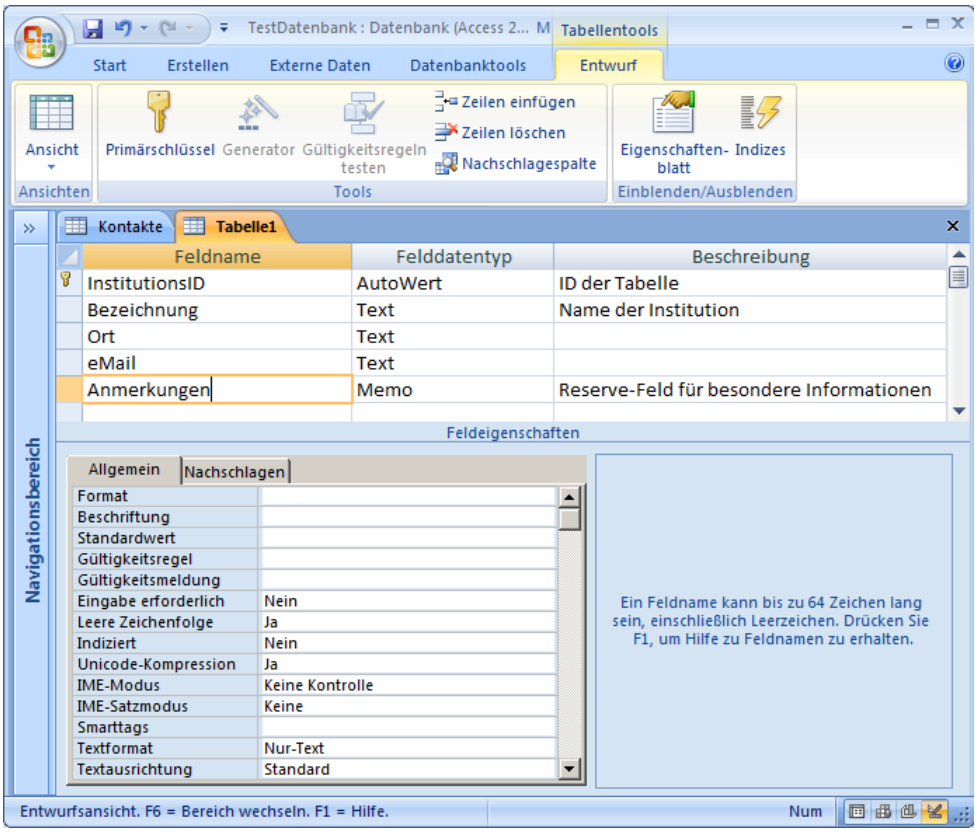
Die neue – zu erstellende – Tabelle Institutionen soll analog zur Besprechung in BASE erstellt werden.
Sachlich ist auch kein Unterschied zu erkennen. Es werden die gleichen Informationen zum Definieren einer Tabelle gebraucht.
Die Feldnamen leiten wir aus dem ERD ab und die Felddatentypen sind hier auch einfach definiert.
Für die ID benutzen wir "Autowert", da dies ja auch die Schlüssel-Spalte werden soll.
Alle anderen Spalten sind Texte. Wer klever vorausdenkt, fügt auch wieder eine Spalte "Anmerkungen" mit dem Typ "Memo" hinzu.



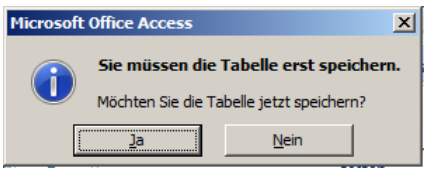


Den Primär-Schlüssel sollte man gleich zuweisen, dann vergißt man dies später nicht. ACCESS legt sonst einen eigenen Schlüssel fest und das passt vielleicht nachher nicht immer in unsere Datenbank-Struktur mit den geplanten und vielleicht auch schon nummerisch festgelegten Verweisen.

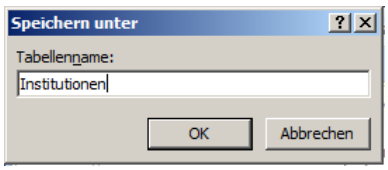




Beim Versuch die Entwurfsansicht zu verlassen oder bei einem ordentlichen Betätigen des "Speichern"-Symbols müssen wir uns entscheiden, ob der Entwurf so in die Datenbank aufgenommen werden soll Da meint hier wieder nur die Struktur der Tabelle. Die Daten sind ja noch gar nicht erfasst und werden ja auch dann – ganz Datenbank-typisch – immer gleich automatisch gespeichert.



Noch schnell den Namen eingeben und die Entwurfsarbeit ist erledigt.



Aufgaben:

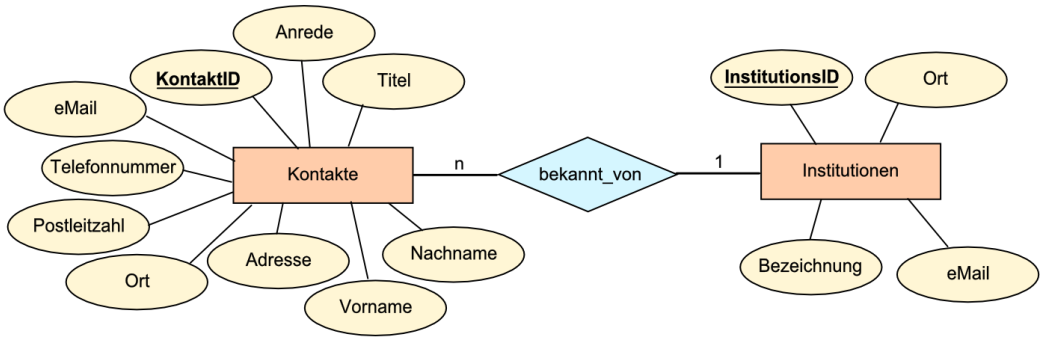
1. *Erstellen Sie die Tabellen-Struktur der Tabelle "Institutionen" wie oben angegeben!*
2. *Füllen Sie die Tabelle "Institutionen" mit den folgenden Daten!*

InstitutionsID	Bezeichnung	Ort	eMail	Anmerkungen
1	Goethe-Gymnasium	Mustern	GoeGymn@webb.de	
2	Tanzverein "Polka"	Musterhausen	post@tv-polka.de	
3	Hilfe e.V.	Meinstadt	info@hilfe.info	
4	Traditionsverein	Cedorf	tradi@verein.de	
5	Let's share	Meinstadt	letsshare@verein.de	
6	Grundschule	Mustern	gs-mustern@webb.de	

Übungs-Aufgaben zu ACCESS:

1. *Vollziehen Sie den Weg zur Erstellung einer Tabelle mit dem Assistenten nach!*
2. *Erstellen Sie mit Hilfe der Assistenten in einer neuen Datenbank ("CD-Sammlung") eine Tabelle für Ihre eigene CD-, Album-/MP3-Sammlung! (Benutzen Sie mindestens 7 Attribute!)*

3.1.6.1.4. Beziehungen zwischen Tabellen umsetzen



Die Tabellen "Kontakte" und "Institutionen" sind die Basis für die Beziehung "bekannt_von". In diese Tabelle werden nun keine echten Daten (Klar-Bezeichnungen) eingetragen, sondern nur Verweise auf die beiden aggregierten Tabellen. Und was eignet sich dabei besser als die Primär-Schlüssel der jeweils betroffenen Datensätze.
Solange wir noch keine "Formulare" zur Verfügung haben (→ 3.1.6.4. Formulare), ist die Dateneingabe noch sehr mühselig.

3.1.6.2. Abfragen

3.1.6.2.1. Erstellen einer Abfrage mit dem Assistenten

Für Anfänger ist die Assistenten-Methode wohl die einfachte Variante. Schrittweise klickt man sich die Elemente und kriterien für die Abfrage zusammen. Man kann jederzeit wieder zurück bis hin zum Anfang, um notwendige Korrekturen vorzunehmen oder vernachlässigte Kriterien mit einzubauen.

Auswahl der Felder

Bestätigung der Auswahl

Sortierung

weitere Suchbedingungen

Aliasnamen – also spezielle Namen für die Abfragen

Verschleierung der Datenbank-Interna oder Verbesserung der Nutzerfreundlichkeit

Überblick

Zusammenfassung des Assistenten

Ergebnis ist eine Tabelle

ist temporär, wird also nur erzeugt, wenn der Bedarf besteht, es werden dann die aktuellen Daten benutzt

Wer etwas über die Umsetzung der Assistenten-Schritte in eine Abfrage lernen möchte, kann sich die fertige Abfrage auch in der Entwurfs-Ansicht anzeigen lassen.

3.1.6.2.2. Erstellen einer Abfrage in der Entwurfs-Ansicht

Diese Variante der Abfragen-Erstellung ist wohl die beliebteste. Es ist sehr einfach und schnell möglich sich die Abfragen zusammenzuklicken und die Ergebnisse sind meist sofort zufriedenstellend.
Für viele Zwecke ist auch eine Vorbereitung über den Assistenten sinnvoll. Später kann die Abfrage kopiert werden und dann in einer Kopie mit den Feineinstellungen experimentiert werden.
Der Aufruf erfogt im oberen Teil der Teil der Abfrage-Übersicht.

Es erscheint die Entwurfs-Ansicht mit einem "Hinzufügen"-Dialog.

Für die weitere Nutzung von Daten müssen zuerst einmal die Tabellen oder andere – schon vorhandene – Abfragen hinzugefügt werden.
Wenn man genug Tabellen und / oder Abfragen hinzugefügt hat, verläßt man den "Hinzufügen"-Dialog mit "Schließen".

Aufgaben:

1. ***Öffnen Sie sich die Entwurfs-Ansicht und fügen Sie sich die Tabelle "Kontakte" hinzu!***
2. ***Realisieren Sie eine "Abfrage2 Kontake", indem Sie die Abfrage aus der Assitenten-Nutzung*** (→ 3.1.3.1.1. Erstellen einer Tabelle mit Hilfe des Assistenten) ***nachbauen!***
3. ***Versuchen Sie die Abfrage2 so zu verändern, dass die nicht angezeigte Spalte "Vorname" (ganz rechts) nicht mehr benötigt wird, aber die Sortierung trotzdem erst nach der des Nachnamens (!) getätigt wird! (Hinweis!: Überlegen Sie sich, wie ein Computer(-Programm) die Entwurfs-Ansicht lesen / interpretieren wird!)***

Prüfen, ob alle so auch für ACCESS gültig sind!?

Bedingungs-Operatoren in Abfragen

Operator / Symbol(e)	Benennung	Bemerkungen / Bedeutung
=	(ist) gleich	in Abfrage-Feldern wird der "="-Operator nicht angezeigt; → wird kein Operator angegeben, dann wird automatisch "=" angenommen
<>	(ist) ungleich	alle anderen Werte werden akzeptiert
<	(ist) kleiner (als)	alle Werte, die kleiner als der angegebene sind, werden akzeptiert
<=	(ist) kleiner oder gleich	erfüllt, wenn der Wert kleiner als der angegebene oder gleich groß, wie dieser, ist
>	(ist) größer (als)	alle Werte, die größer als der angegebene sind, werden akzeptiert
>=	(ist) größer oder gleich	erfüllt, wenn der Wert größer als der angegebene oder gleich groß, wie dieser, ist

Platzhalter / Joker-Symbole

Zeichen / Symbol	Bedeutung	
***** **%**	entspricht einer beliebigen (auch leeren) Zeichenkette bzw. eines beliebigen Wertes	
? _	steht für (genau) ein Zeichen in einer Zeichenkette	

Filter-Bedingungen für Abfragen

Option	Auswirkung / Bedeutung	SQL-Schlüsselwort	Bemerkungen Beispiel(e)
ohne	*keine*		
IST LEER	ist erfüllt, wenn das Feld einen NULL-Wert besitzt	IS NULL IS LEER	bei Options-Feldern (JA / NEIN) wird der unbestimmte Fall genutzt (also weder Ja noch Nein)
IST NICHT LEER	Negation von "IST LEER"; ist erfüllt, wenn das Feld einen (von NULL abweichenden) Wert besitzt	IS NOT NULL IS NOT LEER	
LIKE **WIE**	Übereinstimmung; ist erfüllt, wenn das Feld den entsprechenden Wert besitzt	LIKE	
ZWISCHEN *min* **UND** *max*	im Intervall; ist erfüllt, wenn der Feldinhalt zwischen den Angaben ***min*** und ***max*** liegt	BETWEEN *min* AND *max*	
IN (*Liste***)**	Entsprechung; Enthaltung ist erfüllt, wenn das Feld		IN (3; 6; 12; 24)

	mit einem Wert aus der (Semikolon-getrennten) übereinstimmt		IN ('Moskau'; 'Berlin'; 'New York')
NICHT …	Negation; negiert den folgenden Ausdruck	NOT …	
= WAHR	Validierung; ist erfüllt, wenn der Feld-inhalt ***wahr*** enthält	= TRUE	
= FALSCH	Falsifizierung ist erfüllt, wenn der Feld-inhalt ***falsch*** enthält	= FALSE	

3.1.6.2.3. Berechnungen in einer Abfrage

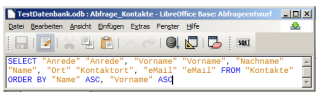
statt des Feldnamens (Attributes) werden Formeln mit den Feldnamen (z.B. "Nettopreis") ohne führendes Gleichheitszeichen eingegeben (z.B. zur berechnung des Bruttopreises: "Nettopreis" * 1,19)
es muss dan ein Alias vergeben werden, da sonst die berechnete Spalte keine Überschrift hätte

Funktionen in Abfragen

Option / Funktion	**Auswirkung / Bedeutung**	**SQL-Schlüssel-wort**	**Bemerkungen**
ohne	*keine*		
Durchschnitt	berechnet das (arith-metrische) Mittel (Mittelwert) des Feldes (Attributes)	AVG	
Anzahl	zählt die Datensätze, die in der Abfrage auftauchen	COUNT	COUNT(*) .. es werden alle Da-tensätze (in der Abfrage) gezählt COUNT(*Spalte*) .. es werden die Datensätze gezählt, bei denen die ***Spalte*** einen Nicht-NULL-Wert enthält
Maximum	ermittelt den größten Wert des Feldes (Attributes)	MAX	
Minimum	ermittelt den kleinsten Wert des Feldes (Attributes)	MIN	
Summe	berechnet die Summe der Feld-Werte (Attribut-Werte)	SUM	
Gruppiert	gruppiert die Datensätze nach den Werten im aus-gewählten Feld (Attribut)	GROUP BY	

3.1.6.2.4. Erstellen einer Abfrage mittels SQL

Diese Methode zum Erstellen einer Abfrage ist mehr was für Profis oder Programmierer. Für einfache Abfragen ist es aber ein gleichwertiges Mittel. Komplizierte Abfragen lassen sich besser über den Assistenten (→ 3.1.3.2.1. Erstellen einer Abfrage mit dem Assistenten) oder die Entwurfs-Ansicht (→ 3.1.3.2.2. Erstellen einer Abfrage in der Entwurfs-Ansicht) zusammenstellen und austesten.
Als Beispiel sei hier der Code für die mit dem Assisten erstellte Abfrage aufgezeigt.
Man kann den Text fast verstehen.
Mehr dazu im SQL-Teil dieses Skriptes (→ 5. Datenbanken - SQL; → 5.1.x. CREATE VIEW).



Zum langsamen Herantasten an SQL kann man sich zuerst einmal den SQL-Code von fertigen – funktionierenden – Abfragen ansehen.
Später ist man villeicht schneller, wenn man nur noch den Code eingibt. Das setzt aber auch leider immer voraus, dass man die ganzen Namen von Tabellen, anderen Abfragen und den Attributen im Kopf oder auf einer Übersicht hat. Diese werden immer – exakt geschrieben – für einen funktionierenden SQL-Code gebraucht.
Letztendlich lesen sich SQL-Codes fast wie eine umgangssprachliche – wenn auch englische – Arbeits-Aufforderung. Dieses war ja auch ein erklärtes Ziel bei der Entwicklung von SQL.

3.1.6.3. Berichte

In modernen Datenbanken verschmelzen Abfragen und Berichte immer stärker. Ursprünglich waren vor allem Berechnungen und Zusammenfassungen Elemente von Berichten.
Der ursprüngliche Gedanke eines berichtes ist es ja auch den Zustand einer Datenbank, oder einzelner Elemente davon, dauerhaft zu dokumentieren. Ein gutes Beispiel ist der Quartals-Bericht oder eine Jahres-Abschluß. Diese Berichte konnten dann ausgedruckt und abgeheftet werden. Schon einige Tage nach den Stichtagen waren die Datenbanken dann in anderen Zuständen. Heute lassen sich fast alle Aktionen auch Zeit-bezogen auswerten. Dadurch kann der Jahres-Bericht auch erst ein halbes Jahr später sicher und immer wieder gleich abgefragt werden.
Berichte eignen sich auch heute noch besonders gut, wenn man mehr oder wenige umfassende Auszüge / Übersichten seiner Daten benötigt.
Da in unserer einfachen Datenbank keine Zeit-basierte Erfassung und Protokollierung der Daten eingebaut ist, können wir uns die aktuelle Situation einer Tabelle oder eines Ausschnittes aus der Datenbank gut als Bericht zusammenstellen lassen.

Aufgaben:

1.
2. *Erstellen Sie einen Bericht der Kontakte-Datenbank, der eine Liste der Kontakte, alphabetisch geordnet und gruppiert nach den Einrichtungen zeigt! Am Ende jedes Einrichtungs-Blockes und am Ende des Berichtes sollen auch die Anzahlen der Kontakte auftauchen!*
3. *Erweitern Sie den Bericht von Aufgabe 2 noch um die Anzahl der Einrichtungen!*

3.1.6.4. Formulare

ACCESS zeichnet sich durch eine Menge praktischer Hilfmittel, Funktionen und Assistenten zur Erstellung und Benutzung von Formularen aus. Mit ein wenig Geschick und Planung gelingt einem schnell ein Konstrukt aus Formularen, die einem unbedarften Nutzer eine eigene Applikation vorgaukeln. Nicht in allen Fällen muss ein Nutzer wissen, das er jetzt mit seiner Heiligkeit – einer Datenbank – arbeitet. Die meisten Nutzer verbinden damit sowieso andere Vorstellungen, als ein (praktischer) Informatiker.

Aufgaben:

1.
2.
3.

Hilfe (Tutorial) zu microsoft ACCESS
https://www.access-tutorial.de/

3.1.7. Datenbanken mit SQLite

sachlich gesehen ist SQLite eine Bibliothek aus Programmen, die ein relationales Datenbank-System ermöglichen

Logo von SQLite
Q: de.wikipedia.org
(D. Richard Hipp)

besonders für eingebettete Datenbank-Systeme (embedded database (management system)) gedacht, also integriert in andere Programme oder Systeme
tritt praktisch nach außen nicht in Erscheinung, macht im Hintergrund die Arbeit

kann man sich wie eine Runtime-Version vorstellen, der Nutzer / Programmierer bekommt hoch-professionelle Programm-Routinen, die er in seine Projekte einbauen bzw. in diesen dann nutzen kann

von Richard HIPP (1961 -) in der Programmiersprache C (sehr Maschinen-nah) geschrieben, deshalb klein und schnell

Schnittstellen für Java und C++
z.B. in Python ab Version 2.5 schon enthalten
wird von vielen anderen System (Adobe AIR) oder Programmen (Apple Safari; mozilla Firefox; google Chrome; Skype) genutzt

mit PHP nutzbar

```php
$db = new SQLite3('test.db');
$result = $db->query('SELECT * FROM tbl2');

while($user = $result->fetchArray(SQLITE3_ASSOC)){
  var_dump($user);
}
```

ähnlich erfolgt Zugriff in anderen Programmier- bzw. Skript-Sprachen

Vorteile von SQLite:

einfach, schnell, klein
sehr kleine, kompakte Implementierungen, deshalb auch auf älterer, wenig Leistungs-fähiger Hardware nutzbar (es gibt auch Umsetzung, die auf einem Raspberry Pi funktioniert!)
erzeugt eine Datei pro Datenbank, kleine Dateien
für Windows, Linux, Unix, Android, OS X, Symbian OS X, iOS, Windows Phone, … verfügbar
Datei kann Betriebssystem-übergreifend genutzt werden
funktioniert / gibt es auch (in Verbindung mit bestimmten Fontends) als portable App (z.B. integriert im IoStick von Tino HEMPEL → www.tinohempel.de)
erfüllt Sprachstandard SQL92
schnell
von Konsole oder aus anderen Programmen heraus nutzbar
gemeinfreie Lizenz (Public domain)
Konfigurations-frei
Server-los (funktioniert ohne Server)
in sich geschlossenes / eigenständiges System (self contained), (netzunabhängig)
Transaktions-basiert
sehr verbreitet → viele Hilfen / Foren / …
es existiert eine ODBC-Schnittstelle

arbeitet auch mit Open-/Libre-Office zusammen
in-memory-Nutzung möglich (Datenbank nur im RAM; keine persistente Speicherung / Haltung der Daten)

Nachteile:

keine eigene Bedienoberfläche (nur Konsolen-Bedienung / -Benutzung möglich bzw. Zugriff über andere Programme)
Tabellenstrukturen lassen sich nachträglich nur begrenzt ändern, in neueren Versionen lassen sich aber schon Spalten hinzufügen und Tabellen und Spalten umbenennen
bestimmte Nutzer- / Zugriffs-Rechte sind nicht umgesetzt – also sind nicht nutzbar (diese Funktionen müssen durch das übergeordnete Programm realisert werden!)
fehlende Client-Server-Architektur (Schreib-Zugriffe verschiedener Prozesse / Threads / Programme / Nutzer können nur nacheinander abgearbeitet werden)
fehlende Typ-Sicherung (fehlerhafte Eingaben werden immer in Texte umgewandelt)
nicht in deutsch verfügbar
nicht für große Datenbestände

3.1.7.0. Voraussetzungen, Einschränkungen, Download, Installation

für schulische Zwecke (Arbeiten im Unterricht, Hausaufgaben, Projekte, …) bietet sich die fertige(n) Installation(en) auf dem IoStick von Tino HEMPEL (→ www.tinohempel.de) an

auf der Normal-Version des Io-Sticks gibt es die folgenden SQLite-Programme – schon fertig installiert und eingerichtet.
Man muss nur noch das passende Programm auswählen, und kann sofort anfangen zu arbeiten.

SQLite-Programme auf dem IoStick

- **SQLiteStudio** — Fontend mit dem größten Leistungs-Umfang
  sehr empfehlenswert
  → 3.1.7.1.3. Arbeiten mit dem SQLiteStudio
- **SQLite Database Browser**
  → 3.1.7.1.1. Arbeiten mit dem SQLite Database Browser
- **SQLiteman**
  →
- **SQLite Manager**
  → 3.1.7.1.2. Arbeiten mit dem SQLite Manager

Sollten keine speziellen Argumente für ein anderes Programm sprechen, würde ich ab hier die Weiterarbeit mit dem SQLiteStudio empfehlen (→3.1.7.1.3. Arbeiten mit dem SQLiteStudio). In dessen Bespechung gehen wir auch auf weitere Konzepte der Datenbanken ein. Diese können teilweise auch von anderen Programmen realisiert werden.

Die Abitur-Version bietet nur die Programme:

SQLite-Programme auf dem IoAbiStick
(in Mecklenburg-Vorpommern zum Abitur zugelassen)

- **SQLiteStudio** — Fontend mit dem größten Leistungs-Umfang sehr empfehlenswert
  → 3.1.7.1.3. Arbeiten mit dem SQLiteStudio
- **SQLite Database Browser**
  → 3.1.7.1.1. Arbeiten mit dem SQLite Database Browser
- **SQLite Manager** — (nur auf älteren Versionen)
  → 3.1.7.1.2. Arbeiten mit dem SQLite Manager

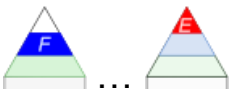
Der Vollständigkeit halber sei hier noch erwähnt, dass der Normal-Stick auch noch weitere Datenbank- und SQL-Prgramme bietet.

(weitere) Datenbank- und SQL-Programme auf dem IoStick

- **HeidiSQL** — graphisches Fontend für mySQL
- **Webserver mit mySQL** — fertig konfigurierter Webserver mit mySQL als Datenbank-System
- **LibreOffice BASE** — Datenbank-System / -Programm aus der Office-Suite von LibreOffice.org bzw. OpenOffice.org

Links:
http://www.sqlite.org
http://www.tinohempel.de
http://www.libreoffice.org
http://www.openoffice.org ; http://www.openoffice.org/de/

3.1.7.0.1. SQLite auf einem Raspberry Pi



Datenbanken sind nur was für große Rechner oder Server – könnte man denken: Aber da täuscht man sich gewaltig. Es geht auch im Mini-Maßstab. Gerade sogenannte eingebettete Datenbanken eignen sich auch für kleine Systeme und Anwendungen.

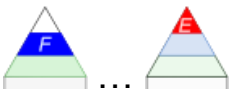
Der Mini-Rechner RaspberryPi ist ein gutes Beispiel für ein Arbeiten im Super-Kleinen. Wer sich intensiver mit dem Gerät beschäftigt, merkt schnell, dass der RaspberryPi ein Tausend-sassa ist.
Der RaspberryPi 2 hat mit 1 GB RAM und einem Vierkern-Prozessor schon einiges unter der Haube.
Der aktuelleste Pi (Version 3) ist noch Leistungs-stärker.
Mit ziemlich großer Sicherheit funktioniert aber auch der Raspberry Pi zero – ein echter Zwerg unter den Einplatinen-Rechnern.

Eine Datenbank auf SQLite-Basis ist ebenso kein Problem, wie ein laufendes Libre- oder Open-Office mit BASE.

Natürlich merkt man die sparsam ausgestattete Hardware, aber nicht immer geht es um Schnelligkeit und riesige Datenmengen.

Und gerade zum Ausprobieren ist Übersichtlichkeit und ein beobachtbares Tempo ein gutes Hilfsmittel.

3.1.7.0.2. Daten-Typen in SQLite



Basis-Datentypen

- **NULL** NULL-Wert (Attribut hat NULL-Wert)
- **INTEGER** Vorzeichen-behaftete Ganzzahl
  6 verschiedene Größen / Längen möglich
  für Primärschlüssel wird Standard-Typ empfohlen
- **REAL** Fließkomma-Zahl / Gleitkomma-Zahl
  8 Byte IEEE Fließkomma-Zahl
- **TEXT** Text im Text-Typ der Datenbank
  (UTF-8, UTF-16BE, UTF-16LE)
- **BLOB** variabler binärer Typ
  es wird das gespeichert, was eingegeben wurde

Hinweis:
Es gibt in SQLite keinen Datentyp BOOLEAN. Als Äquivalent wir empfohlen die Interger-Werte 0 für false und 1 für true zu benutzen.

spezielle Datentypen

- **Datum** kann in verschiedenen Versionen gespeichert sein
  - als Text
    → ISO8601-String: YYYY-MM-DD HH:MM:SS.SSS
  - als Real-Zahl
    → Greenwich-Zeit im Julianischen Kalender nach dem 24.November 4714 v.Chr.

  als Integer-Zahl
    → UNIX-Zeit: Anzahl der Sekunden seit dem 01. Januar 1970 00:00 UTC
- **Numeric** Datenbank-System wandelt Eingaben in ein passendes Zahlenformat um (auch Texte)
-

3.1.7.1. Nutzung von SQLite

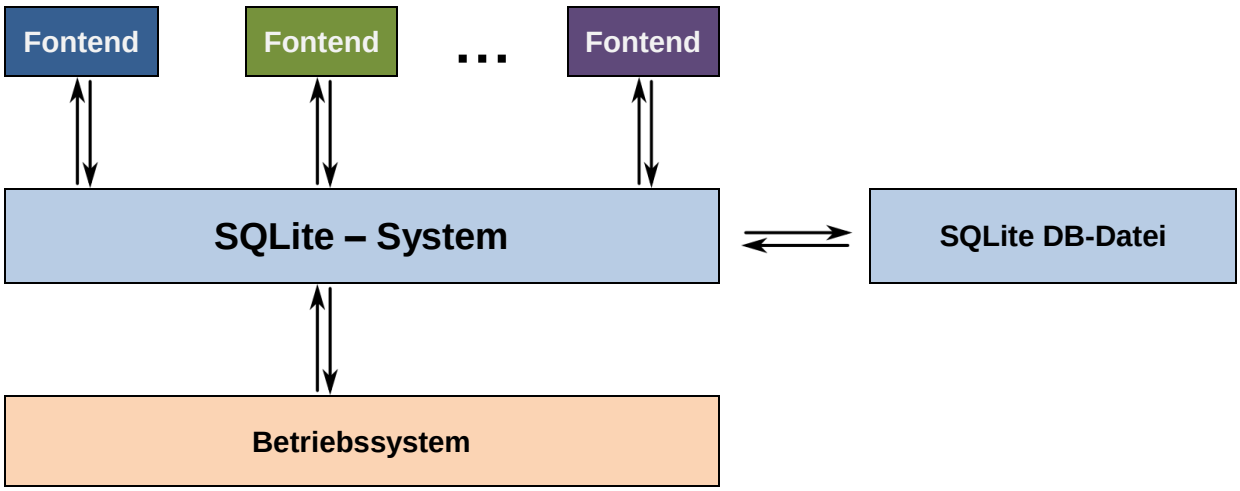
einfache Fontends zum direkten Arbeiten
stammen alle von Dritt-Anbietern

Fontend's / Benutzeroberflächen für SQLite

- **SQLite Browser (DB Browser for SQLite; SQLite Database Browser)**
  - graphische Oberfläche
  - für Win (inklusive portApps), MacOS, Linux
  - Source code verfügbar → praktisch auf weitere Systeme übertragbar
  -
- **SQLite Manager**
  - Konsolen-orientierte Oberfläche
  - auch in deutsch verfügbar
  -
- **SQLite Expert**
  - graphische Oberfläche
  - sehr umfangreich und Leistungs-stark
  - diverse Importe
  - SQL-Scripts- und PDF-Export
  - für Windows; freie Basis-Version, erweiterte professionelle Version erhältlich
  -
- **SQLite Adminstrator**
  - sehr Leistungs-fähiges Tool
  - enthält guten SQL-(Text-)Editor (automatische Code-Koplementierung; Syntax-Highlighting)
  -
- **Navicat for SQLite**
  - intuitiv, Praxis-orientiert
  - für Win, MacOS und Linux verfügbar; auch für andere Datenbanken verfügbar
  - kostenpflichtig (14tägiger Test möglich)
  -
- **SQLite Studio**
  - sehr Leistungs-fähig
  - gut für die schulische Umgebung geeignet
  - für Win (inklusive portApps)
- **SQLite ???**
  - ???
  -
- **SQLite ???**
  - ???
  -

SQLite Expert wohl am umfangreichsten, aber direkte Installation notwendig, was in den meisten Fällen kein Problem sein sollte

in schulischen / Ausbildungs-Situationen sind die anderen Oberflächen vorteilhaft / unproblematischer
Meine Empfehlung geht dabei in Richtung SQLite Studio.
Man kann sich aber jederzeit umorientieren, da alle Fontends auf SQLite als Basis-System aufsetzen. Das Einzige, was sich ändert, ist die vielleicht etwas andere Bedienung des Fontends.

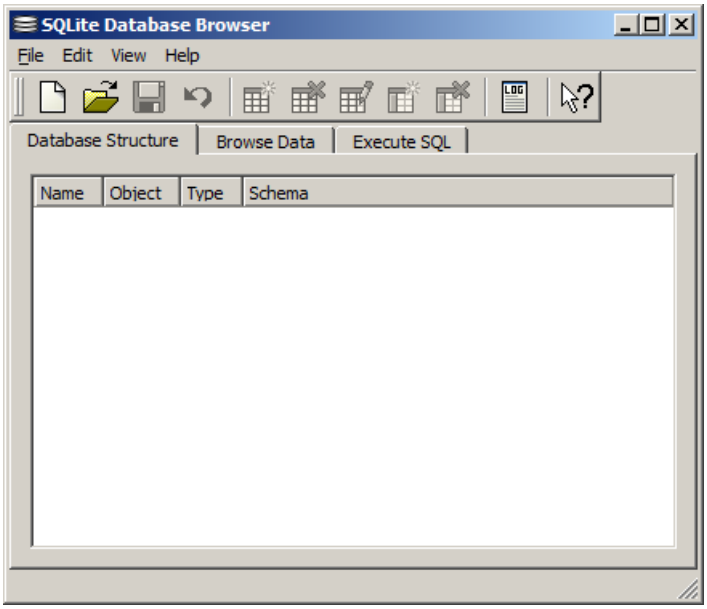


3.1.7.1.1. Arbeiten mit dem SQLite Database Browser

Der SQLite Database Browser ist ein recht einfach gestricktes Tool. Besonders geeignet ist der Browser für die Erstellung und das bestrachten von Datenbank-Strukturen.
Eine praktische Datenbank-Arbeit ist z.T. schwierig bis hin zu ganz unmöglich. Dafür ist der Browser einfach nicht gedacht.
Assistenten oder andere Hilfs-Tools sucht man im SQLite Database Browser deshalb auch vergeblich.

3.1.7.1.1.x. Erstellen einer Beispiel-Datenbank (Kontakte-Institutionen)

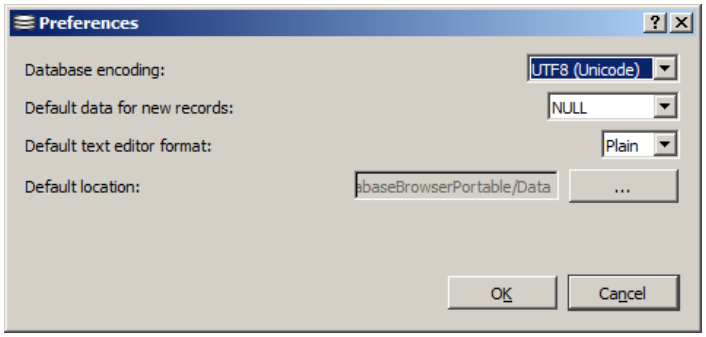
Oberfläche für SQLite



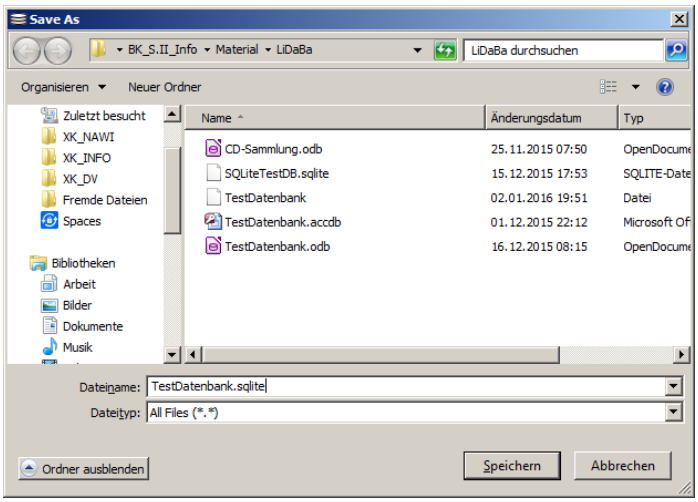
Der Reiter "Browse Data" zeigt den Daten-Inhalt einer auszuwählenden Tabelle (oben links). Neben einem einfachen Datensatz-Navigator gibt es sehr einfache Möglichkeiten zum Eingeben und Löschen von Datensätzen.

Bei "Executable SQL" können direkte SQL-Anweisungen eingegeben werden.

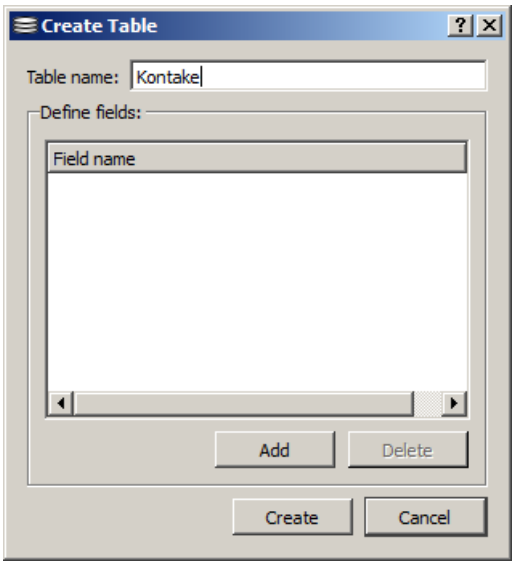
selbst die Benutzer-Oberfläche bietet nur sehr wenige Einstellungs-Möglichkeiten



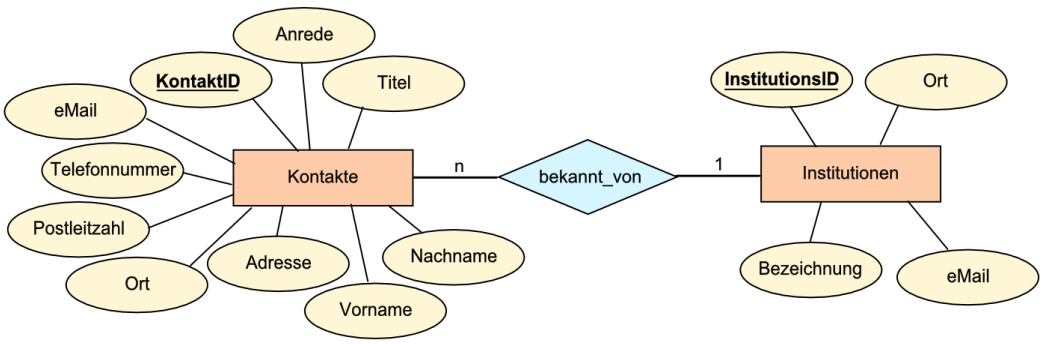
Erzeugen der neuen Datenbank(-Datei)

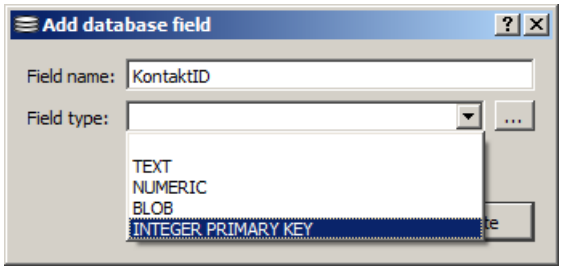


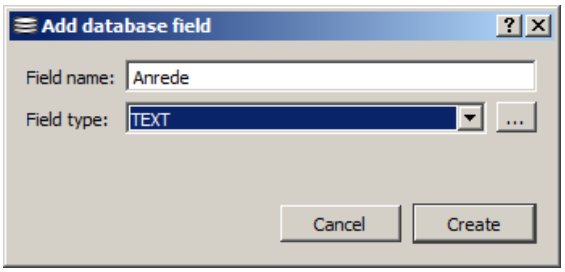
Erstellen einer Tabelle



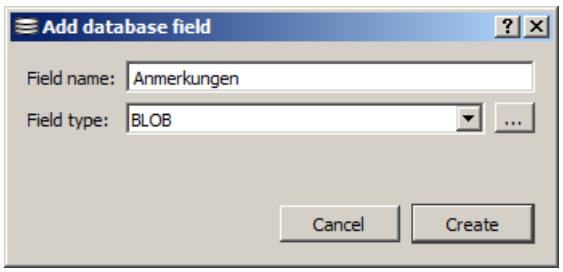
Als zu realisierende Tabelle wollen wir uns zuerst an die "Kontakte" machen. Grundlage ist das vorne besprochene und bei BASE und ACCESS ebenfalls umgesetzte folgende ERD:



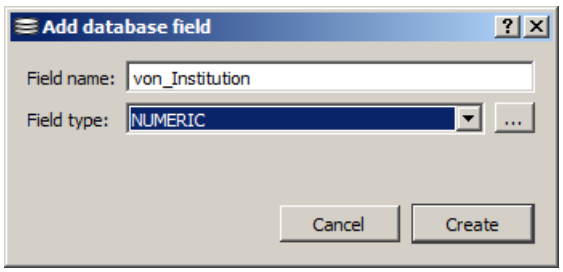




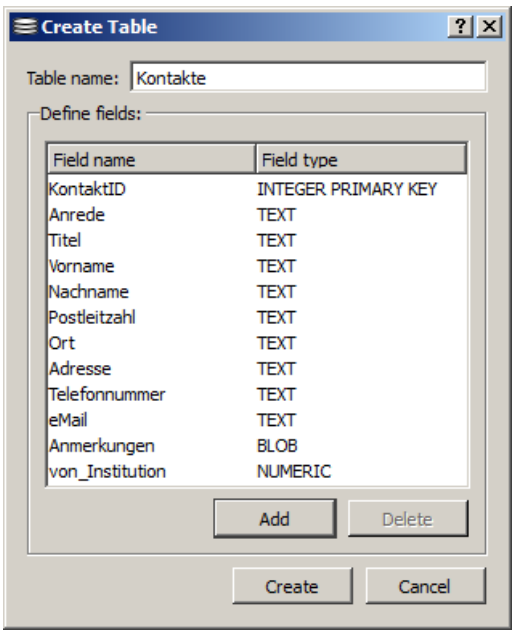
Das optionale "Anmerkung"s-Feld für "alle" Fälle kann hier natürlich auch weg gelasen werden. Das ERD gibt es jedenfalls nicht her.



Wer die Kapitel zu ACCESS und BASE schon durchgearbeitet hat, weiss um die notwendige Spalte "von_Institution" für die Realisierung der Beziehung "" aus dem ERD.
Für die anderen Nutzer wird die notwendige Änderung / Ergänzung später ausführlich beschrieben (→ Tabellen-Struktur bearbeiten).



Nachdem alle Felder / Attribute defniert sind, kann die Tabelle kreiert ("Create") werden.



Am Schluß kann man sich das SQL-Äquivalent in der Spalte "Schema" ansehen. Wer es nicht mit der Maus und dem Zusammenklicken hat, könnte den SQL-Text auch gleich unter "Execute SQL" eingeben (→ Alles schneller mit SQL).

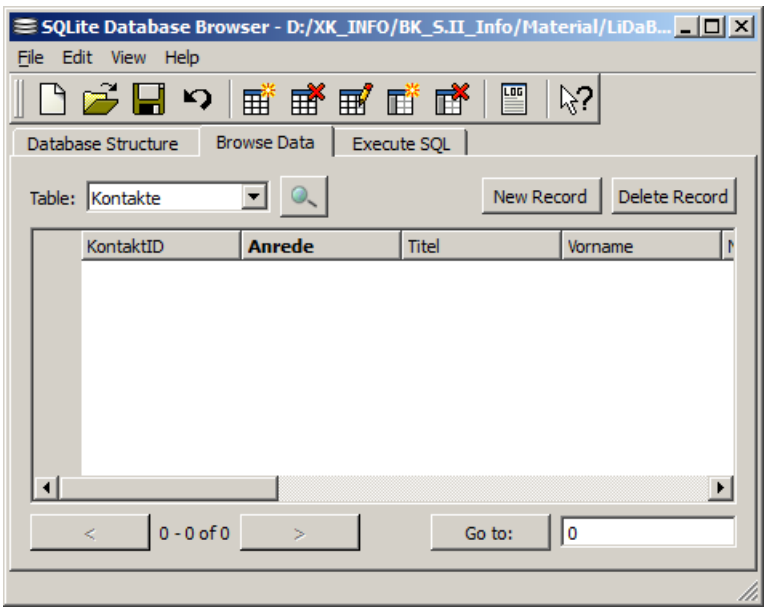
Im Database Browser können wir uns auch den SQL-Text unserer zusammengeklickten Tabellen-Struktur ansehen.

```
CREATE TABLE Kontakte (KontaktID , Anrede , Titel , Vorname , Nachname ,
Postleitzahl , Ort , Adresse , Telefonnummer , eMail , Anmerkung ,
von_Institution )
```

Die typisierte Tabellen-Erstellung und das Setzen des Primär-Schlüssels wird weiter hinten mit einer vollständigen SQL-Anweisung realisiert (→ Alles schneller mit SQL Erstellen der Tabellen).

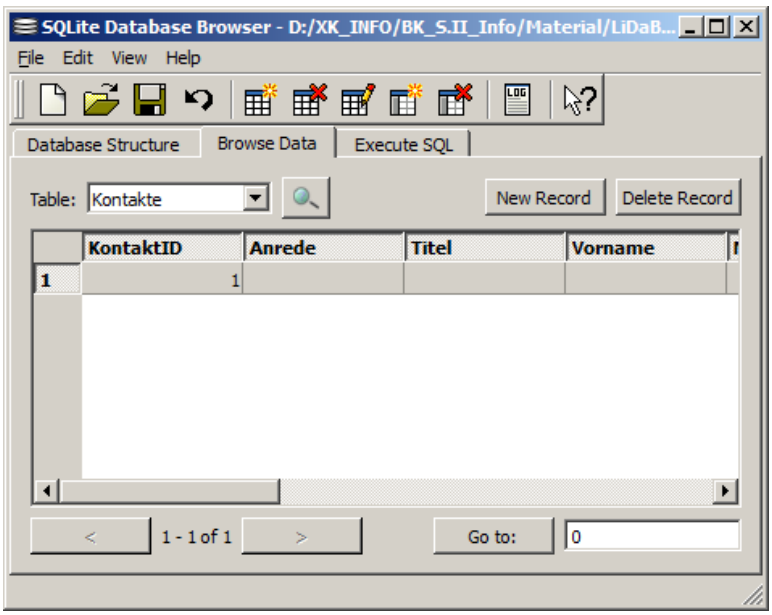
Daten eingeben

Der Reiter "Browse Data" zeigt den Daten-Inhalt einer auszuwählenden Tabelle (oben links). Neben einem einfachen Datensatz-Navigator (unten) gibt es sehr einfache Möglichkeiten zum Eingeben und Löschen von Datensätzen.



Mit "New Record" wird lediglich eine neue Zeile – also ein Datensatz – angelegt. Als einziger Wert ist der primäre Schlüssel vorgegeben.

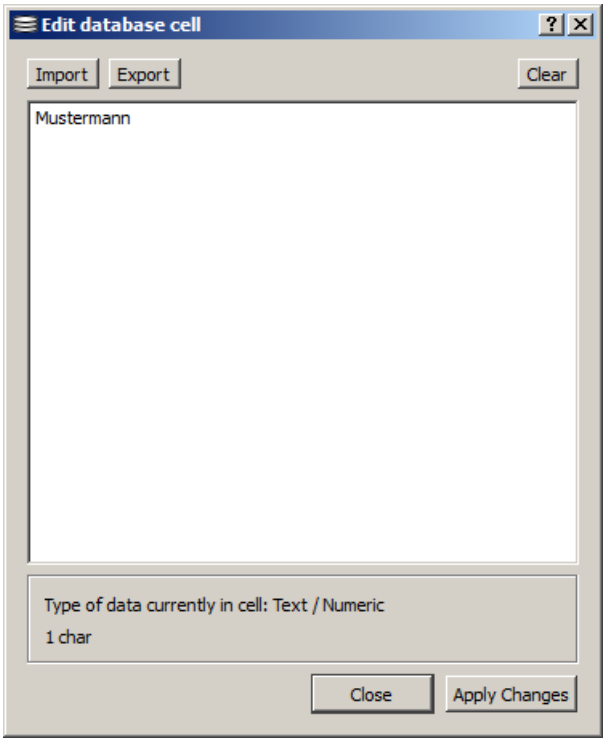
Die Eingabe einzelner Werte für die einzelnen Spalten (Attribute) ist eine mühselige Angelegenheit. Da lohnt es sich entweder mit einem Daten-Import oder mit einer Eingabe über SQL-Anweisungen nachzudenken.



in zentralen Feld kann man die Eingaben schreiben

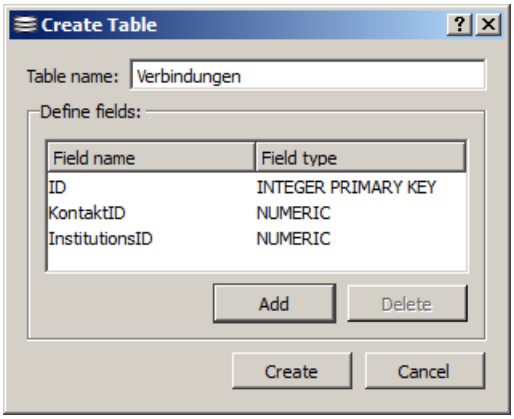
Die Eingabe wird nur übernommen, wenn man "Apply Changes" anklickt. "Close" entspricht einem Abbruch.

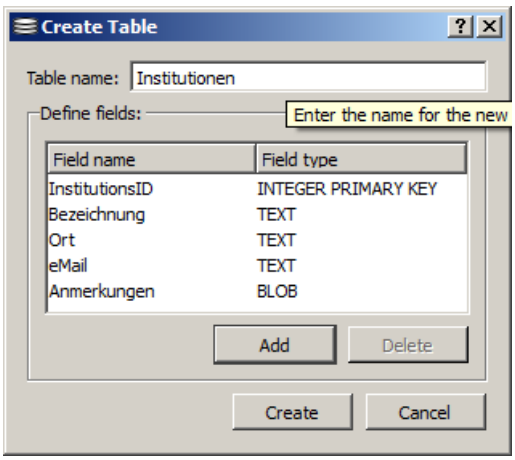
Wem das Eingeben auf diese Weise zu nervig wird, der sollte sich schon mal mit der Vorgehensweise mit SQL informieren (→ Alles schneller mit SQL) und vielleicht dahin wechseln.



Wichtig ist hier, dass man sich regelmäßig um das Speichern kümmert. Anders als bei den klassischen Datenbanken werden die Daten erst mit einem expliziten "Speichern" in die Datenbank-Datei geschrieben!

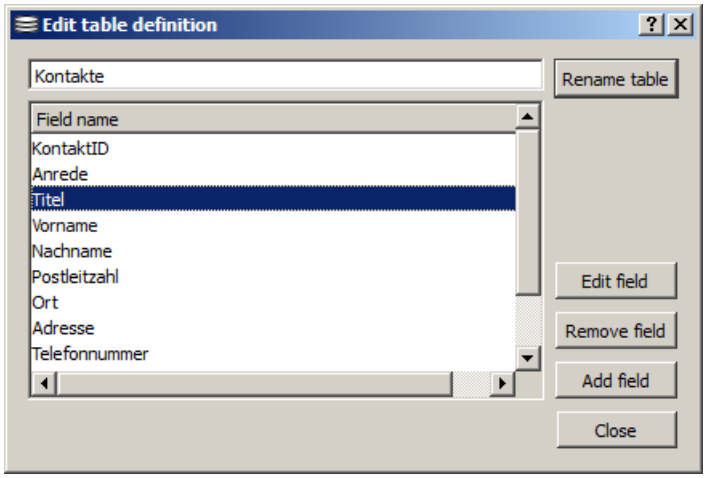
Genau so legen wir nun die beiden weiteren Tabellen "Institutionen" und "Verbindungen" an und füllen sie mit den notwendigen Daten.





Tabellen-Struktur bearbeiten

Der eine oder andere Fehler lässt sich bei " Modify Table" korrieren. Das Ändern funktioniert beim Namen der Tabelle, genauso wie bei den Feldern.
Aber Vorsicht, werden z.B. Spalten / Felder gelöscht, dann verschwinden auch die enthaltenen Daten. In so einem Fall muss man sich dann das "Speichern" genau überlegen, denn bis dahin ist alles nur im Speicher passiert.



Alles schneller mit SQL

Erstellen der Tabellen

```
CREATE TABLE Kontakte (KontaktID INTEGER PRIMARY KEY, Anrede TEXT, Titel
TEXT, Vorname TEXT, Nachname TEXT, Postleitzahl TEXT, Ort TEXT, Adresse
TEXT, Telefonnummer TEXT, eMail TEXT, Anmerkungen BLOB, von_Institution
NUMERIC);
```

```
CREATE TABLE Institutionen (InstitutionsID INTEGER PRIMARY KEY,
Bezeichnung TEXT, Ort TEXT, eMail TEXT, Anmerkung BLOB);
```

```
CREATE TABLE Verbindungen (ID INTEGER PRIMARY KEY, KontaktID NUMERIC,
InstitutionsID NUMERIC);
```

Eingeben von Daten

```
INSERT INTO Kontakte VALUES(1, 'Herr', 'Dr.', 'Klaus', 'Mustermann',
'12345', 'Musterhausen', 'Musterstr. 13', '0123456789',
'mustermann@webb.de', '', 1);
```

Zur Eingabe-Hilfe kann man sich ja die unveränderlichen SQL-Teile in die Zwischenablage kopieren und dann bei jedem Datensatz wieder einfügen. Dann müssen nur noch die eigentlichen Daten eingearbeitet werden.

```
INSERT INTO Kontakte VALUES(, '', '', '', '', '', '', '', '', '', );
```

Es macht zwar den Eindruck, man müsse unwahrscheinlich viel schreiben, aber beim genaueren Hinsehen fällt auf, dass man ja auch in der Einzeleingabe alles tippen musste. Dazu kommen dann noch die vielen Maus-Aktionen.
Hier noch die fehlenden SQL-Anweisungen für die restlichen Datensätze für die Tabelle "Kontakte".

```
INSERT INTO Kontakte VALUES(2,'Frau','','Monika','Mustermann',
'12345','Musterhausen','Musterstr. 13','0123456789',
'musterfrau@webb.de','',2);
INSERT INTO Kontakte VALUES(4,'Frau','','Maria','Muster',
'23456','Mustern','Am Musterweg 3','0234567890',
'm.muster@tee-online.de','',4);
INSERT INTO Kontakte VALUES(5,'Herr','','Christian','Bauer',
'67890','Berg am Fluß','Waldallee 78','04567890901',
'Chr.Bauer@geemx.de','',5);
INSERT INTO Kontakte VALUES(6,'Herr','','Lucas','Müller',
'54321','Bedorf','Feldweg 7','09998765',
'Mueller@tee-online.de','',1);
INSERT INTO Kontakte VALUES(7,'Frau','','Tara','Zander',
'23456','Mustern','Hauptstr. 6e','0777711',
'Ta.Zan@webb.de','',6);
INSERT INTO Kontakte VALUES(8,'Frau','Prof.','Hertha','Ziesow',
'88888','St. Muster','An der B777','0543861',
'Prof.H.Ziesow@tee-online.de','',4);
INSERT INTO Kontakte VALUES(12,'Frau','','Maria','Berndt',
'67890','Berg am Fluß','Hans-Wald-Str. 4','0456783067',
'','',3);
INSERT INTO Kontakte VALUES(37,'Herr','','Henriette','Krüger',
```

```
'76543','Meinstadt','Feldweg 7','06543210',
'post@h-krueger.de','',1);
INSERT INTO Kontakte VALUES(3,'Herr','','Hans','Fehler',
'34343','Glückstadt an der Pech','Blumenweg 48','088088088',
'H.Fehler@webb.de','',1);
```

Da wird schnell klar, warum viele Datenbank-Freaks gleich in SQL oder mit passenden Text-Editoren arbeiten. Einige der Editoren bieten sogar Syntax-Highlighting (Syntax-Hervorhebung).
Hier die SQL-Anweisungen für die restlichen zwei Tabellen-Inhalte mit so einem Syntax-Highlighting. Man erkennt deutlich die verschiedenen Syntax-Elemente und damit die Struktur der Anweisungen.

```
INSERT INTO Institutionen VALUES(1,'Goethe-Gymnasium','Mustern',
'GoeGymn@webb.de','');
INSERT INTO Institutionen VALUES(2,'Tanzverein "Polka"','Musterhausen',
'post@tv-polka.de','');
INSERT INTO Institutionen VALUES(3,'Hilfe e.V.','Meinstadt',
'info@hilfe.info','');
INSERT INTO Institutionen VALUES(4,'Traditionsverein','Cedorf',
'tradi@verein.de','');
INSERT INTO Institutionen VALUES(5,'Let''s share','Meinstadt',
'letsshare@verein.de','');
INSERT INTO Institutionen VALUES(6,'Grundschule','Mustern',
'gs-mustern@webb.de','');
```

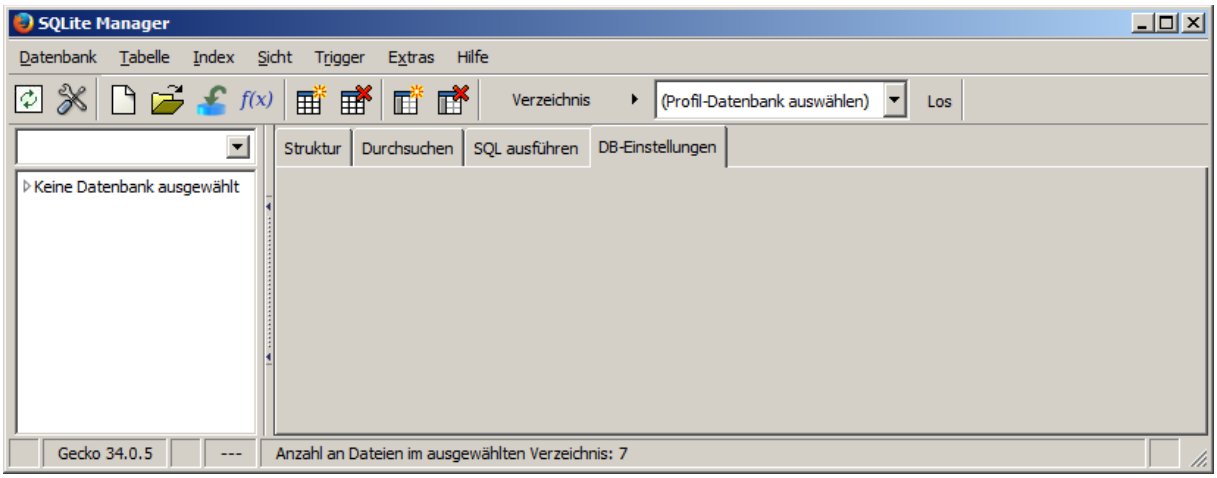
```
INSERT INTO Verbindungen VALUES(0,1,1);
INSERT INTO Verbindungen VALUES(1,2,2);
INSERT INTO Verbindungen VALUES(2,3,1);
INSERT INTO Verbindungen VALUES(3,4,4);
INSERT INTO Verbindungen VALUES(4,5,5);
INSERT INTO Verbindungen VALUES(5,6,1);
INSERT INTO Verbindungen VALUES(6,7,6);
INSERT INTO Verbindungen VALUES(7,8,4);
INSERT INTO Verbindungen VALUES(8,12,3);
INSERT INTO Verbindungen VALUES(9,37,1);
```

Fehler oder Probleme lassen sich so schnell finden und korrigieren.
So, oder so ähnlich sieht das dann in einem anderen Editor aus. Die abgespeicherten Text-Dateien (mit dem SQL-Anweisungen) können dann später importiert werden.
Der umgekehrte Weg – also ein Export des SQL-Anweisungen – ist möglich. Dazu müssen die SQL-Anweisungen nicht wirklich eingegebene worden sein. Der SQLite Database Browser exportiert eine ganze Datenbank auch als SQL-Quelltext.

Links:
Q: http://sqlitebrowser.sourceforge.net

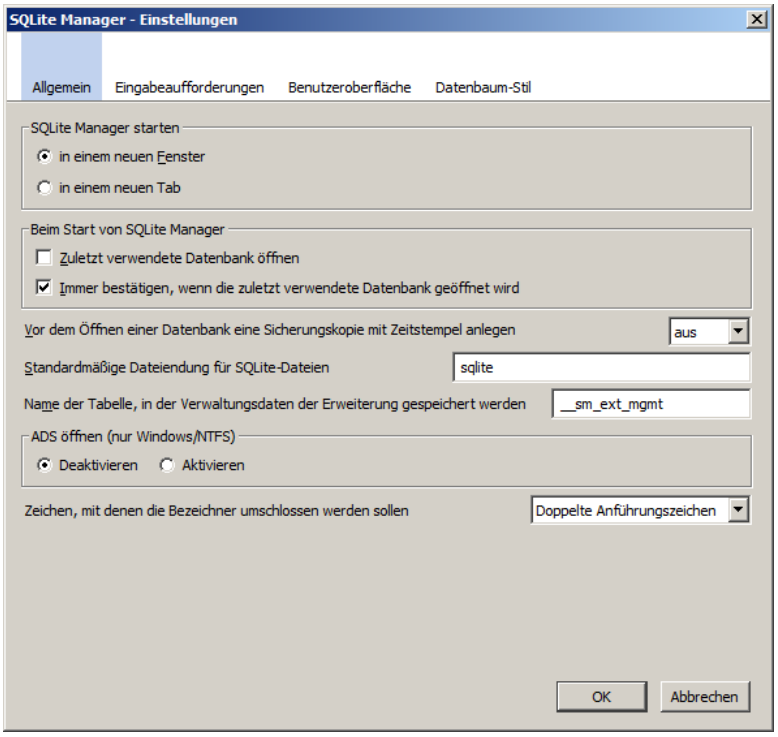
3.1.7.1.2. Arbeiten mit dem SQLite Manager

Die Oberfläche vom SQLite Manager erinnert in vielen Punkten an den SQLite Database Browser (→ 3.1.7.1.1. Arbeiten mit dem SQLite Database Browser). So unterschiedlich sind die beiden Programme auch gar nicht.



Der SQLite-Manager übernimmt nun auch wieder das automatische Speichern nach Eingaben. So erwartet man es ja auch bei Datenbank-Systemen.

Wer will kann sich bei jedem Start eine Sicherungskopie seiner SQLite-Datei erstellen lassen. Dieses ist eine der Optionen, die im Bereich "Allgemein" eingestellt werden können.

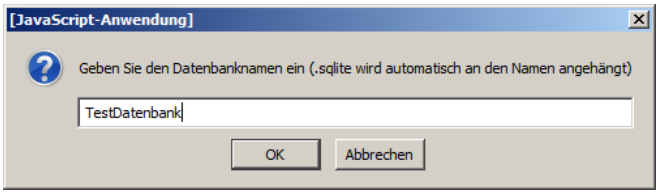


Auch sonst bietet der SQL-Manager einige Einstellungen mehr, als der einfacher gestrickte SQLite-Database-Browser.

Aber viele Einstellungen und viele Bedien-Knöpfe erhöhen das Risiko einer Fehlbedienung. Die Wahl der Bedien-Oberfläche für SQLite sollte also an den Aufgaben erfolgen.
Die erzeugten Datenbank-Dateien können gegenseitig verwendet / geöffnet werden.
Die hohe Kompatibilität der SQL-Daten-Dateien (komplette Datenbanken) ergibt sich aus dem aufgesetzten Anwendungen auf das elementare SQLite-System (s.a. → Fontend's / Benutzeroberflächen für SQLite).

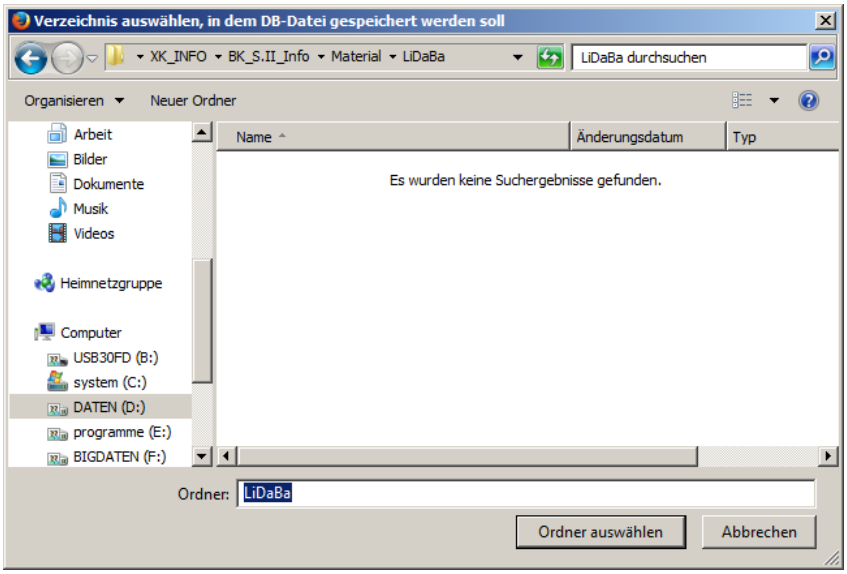
3.1.7.1.2.x. Erstellen einer Beispiel-Datenbank (Kontakte-Institutionen)

Namen für Datenbank vergeben

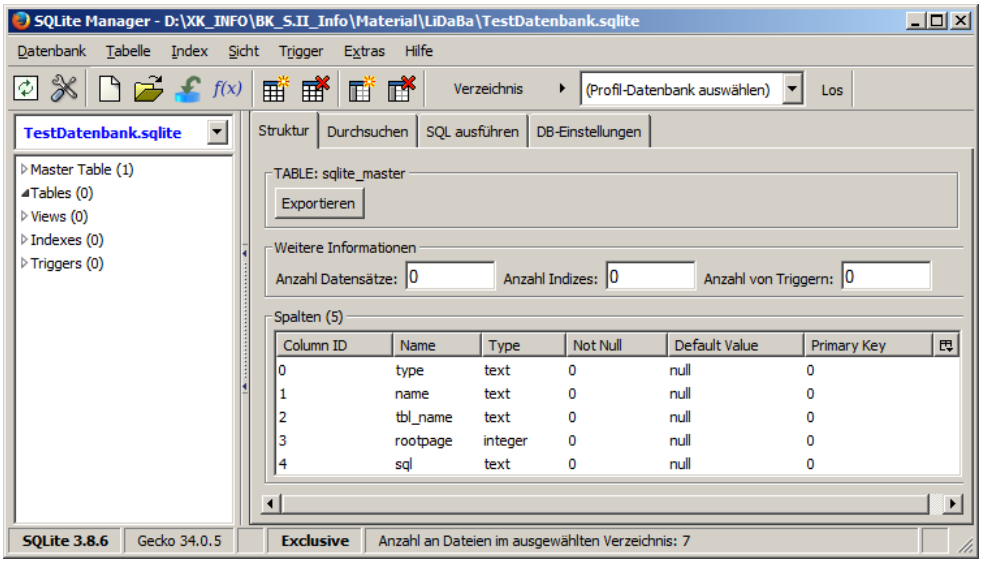


Ordner zum Speichern auswählen

der Name der Datei im gewählten Verzeichnis wurde ja schon vorher bestimmt



leere SQLite-Datenbank

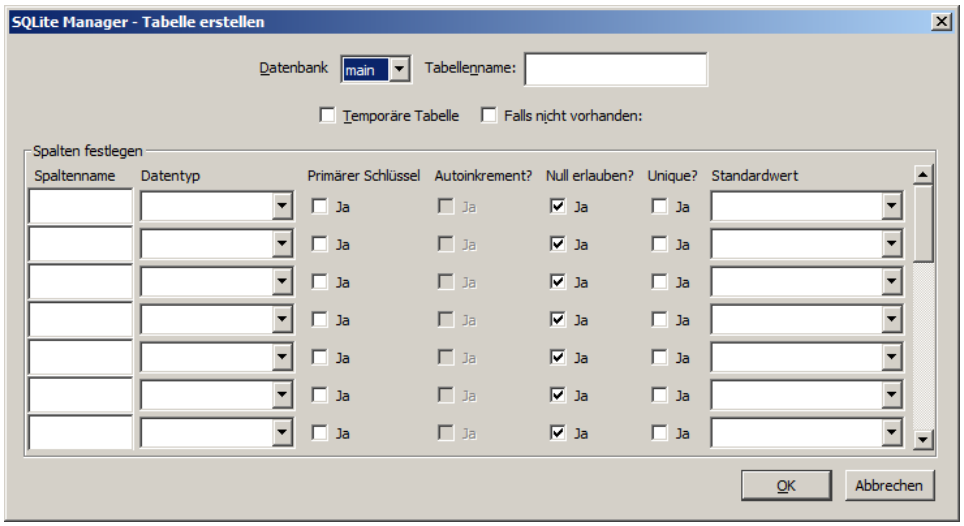


Tabellen

neue Tabelle erstellen

leeres Tabellen-Erstellungs-Raster

sehr übersichtliche und verständliche Möglichkeit die Struktur einer Tabelle zu entwickeln bzw. einzugeben

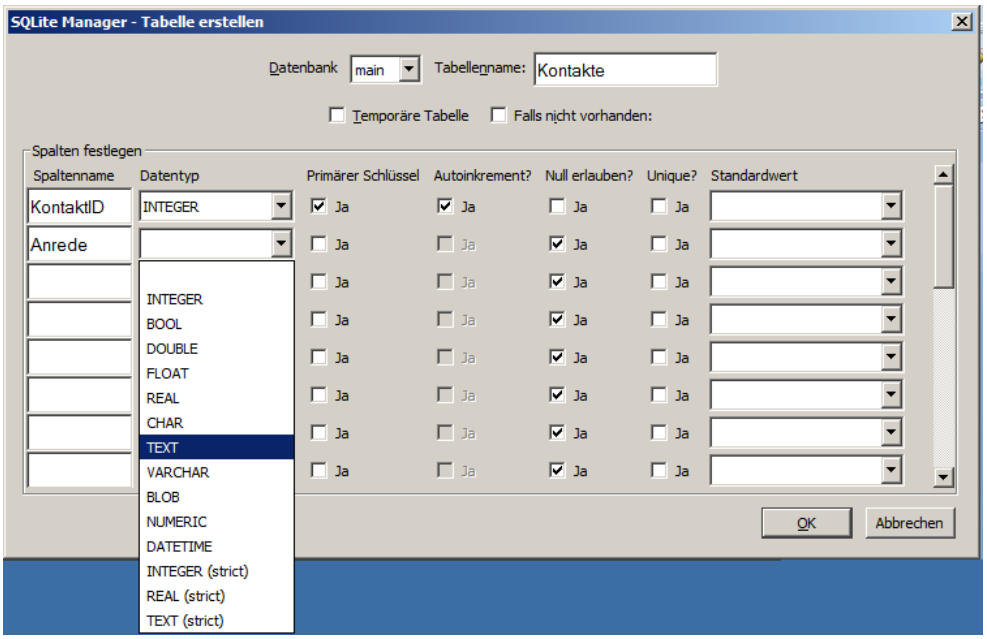


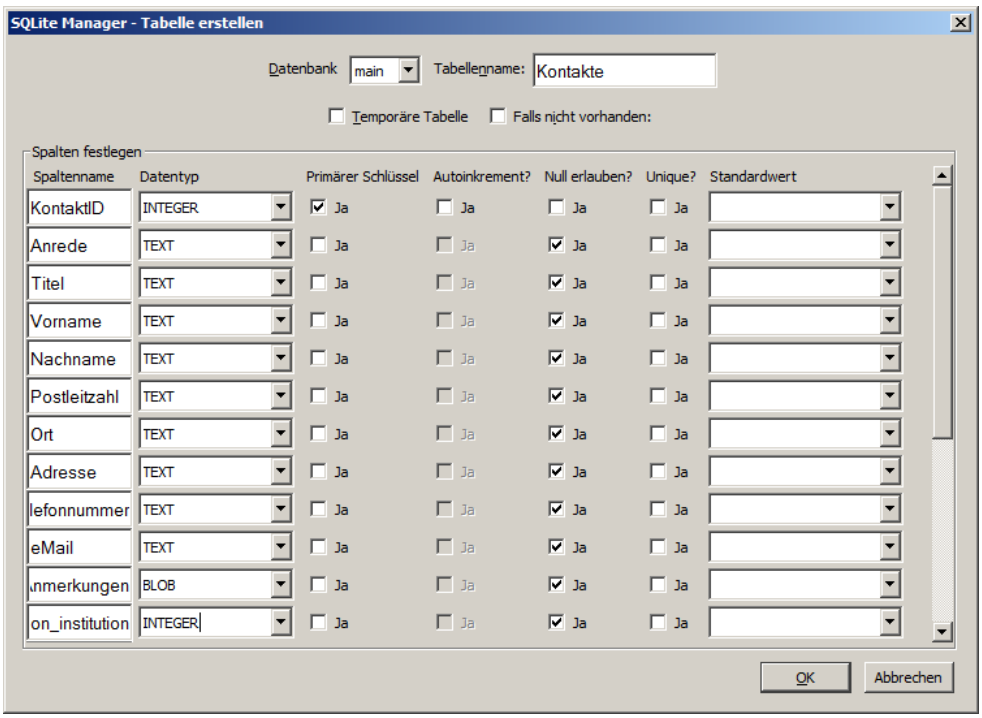
Tabellen-Struktur eingeben

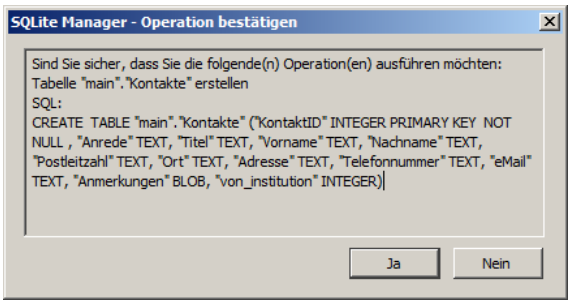
Primärer Schlüssel normalerweise mit Autoinkrement günstig

in unserem Fall wegen vorgegebener Schlüssel ungeeignet

Auswahl der Datentypen für die Felder / Attribute umfangreicher als beim SQLite-Database-Browser

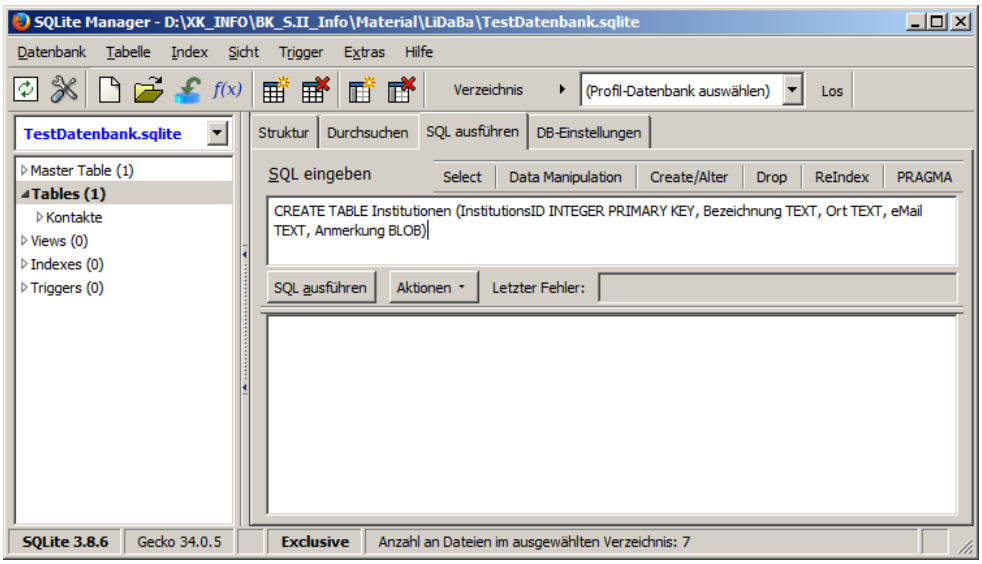






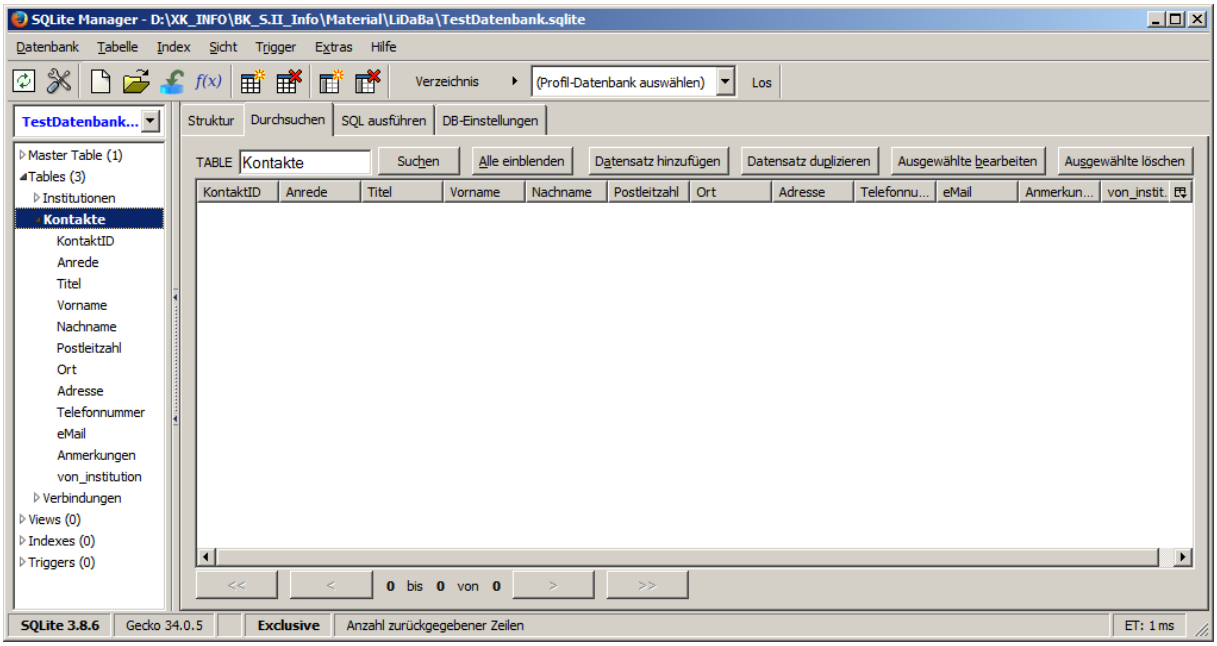
Institutionen über SQL-Quelltext-Eingabe

bei "letzter Fehler" sollte dann: "not an error" stehen

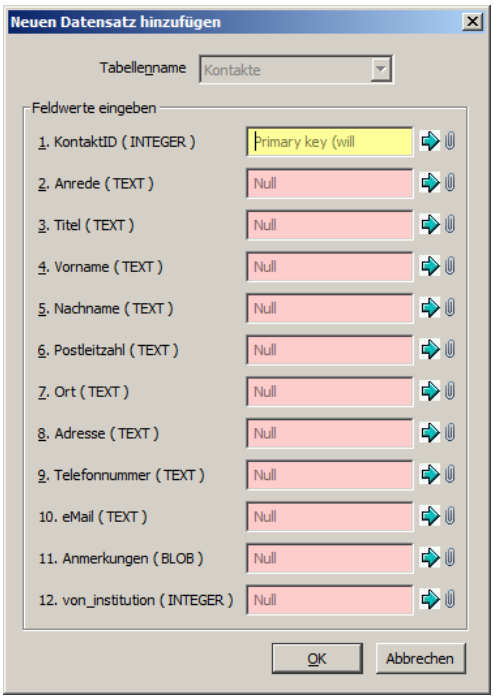


Daten eingeben

Reiter "Durchsuchen"



einen neuen Datensatz / eine neue Zeile mittels "Datensatz hinzufügen"



es folgt ein einfaches Formular mit farblichem Feedback

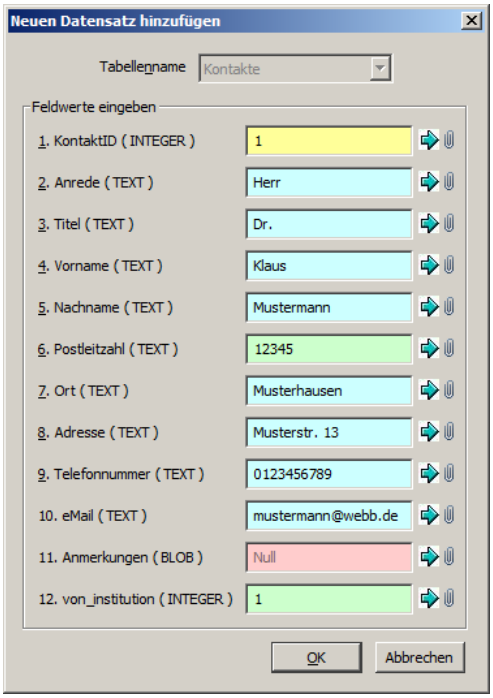
es werden die Felder mit Texten hellblau hinterlegt, die mit nummerischen Werten grünlich

Zellen mit NULL-Werten sind rötlich
auch wenn das so ein bisschen dramatisch aussieht, es ist in den meisten Fällen kein Problem, wenn die Felder frei bleiben

Was natürlich belegt werden muss (- zumindestens in unserem Fall -) ist der primäre Schlüssel bei "KontaktID", als Hinweis dient die gelbliche Feldfarbe

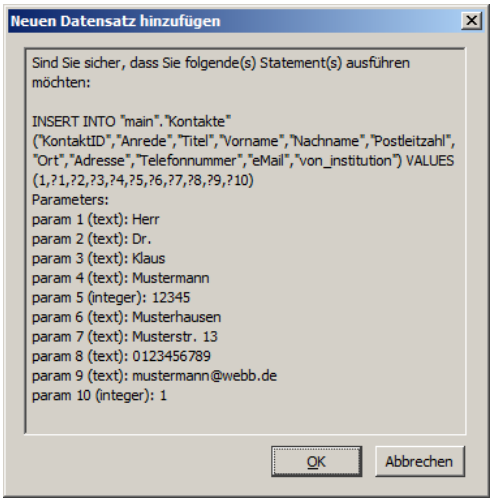
der fertig eingebene Datensatz kann hier schon gut kontrolliert werden

Trotzdem erfolgt nach dem "OK" eine nochmalige Rückfrage.

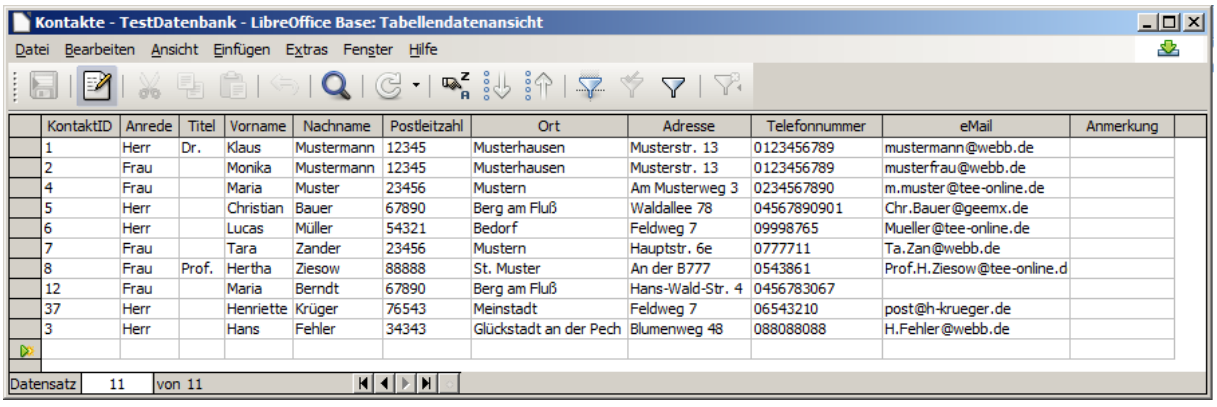


Übersetzung der im Formular gemachten Eingaben in eine SQL-Anweisung. Diese kann zwar nicht direkt geändert werden, aber bei einem "Abbrechen" gelangt man zum Formular zurück. Mit einem "OK" der resultierenden SQL-Anweisung wird die Datensatz-Eingabe bestätigt.

nach Bestätigung scheinbar trotzdem Rückkehr zur (alten) Datensatz-Eingabe
jetzt können aber die Daten des neuen (!) Datensatzes unter Verwendung des vorherrigen zusammengestellt werden
Achtung! Daten wirklich gründlich überprüfen, es schleichen sich durch die Vorbelegung schnell Übernahme-Fehler ein!

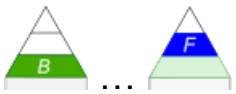


Hier noch mal die komplette Daten-Tabelle aus einer BASE-Datenbank:

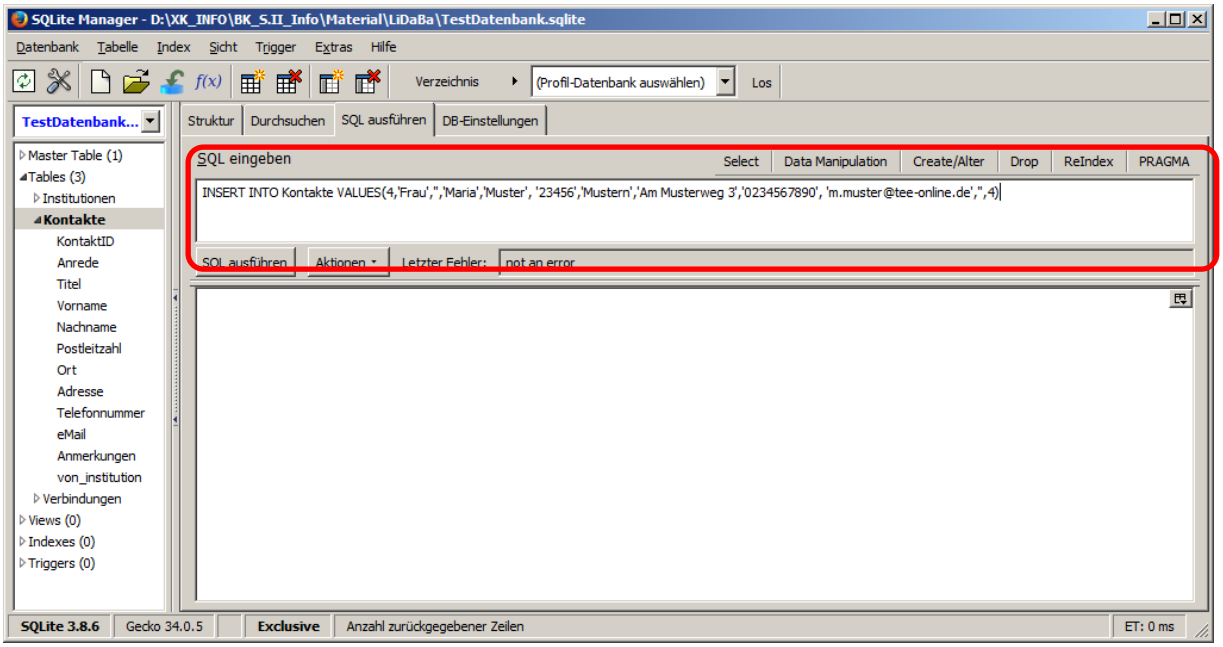


KontaktID	Anrede	Titel	Vorname	Nachname	Postleitzahl	Ort	Adresse	Telefonnummer	eMail	Anmerkung
1	Herr	Dr.	Klaus	Mustermann	12345	Musterhausen	Musterstr. 13	0123456789	mustermann@webb.de	
2	Frau		Monika	Mustermann	12345	Musterhausen	Musterstr. 13	0123456789	musterfrau@webb.de	
4	Frau		Maria	Muster	23456	Mustern	Am Musterweg 3	0234567890	m.muster@tee-online.de	
5	Herr		Christian	Bauer	67890	Berg am Fluß	Waldallee 78	04567890901	Chr.Bauer@geemx.de	
6	Herr		Lucas	Müller	54321	Bedorf	Feldweg 7	09998765	Mueller@tee-online.de	
7	Frau		Tara	Zander	23456	Mustern	Hauptstr. 6e	0777711	Ta.Zan@webb.de	
8	Frau	Prof.	Hertha	Ziesow	88888	St. Muster	An der B777	0543861	Prof.H.Ziesow@tee-online.d	
12	Frau		Maria	Berndt	67890	Berg am Fluß	Hans-Wald-Str. 4	0456783067		
37	Herr		Henriette	Krüger	76543	Meinstadt	Feldweg 7	06543210	post@h-krueger.de	
3	Herr		Hans	Fehler	34343	Glückstadt an der Pech	Blumenweg 48	088088088	H.Fehler@webb.de	

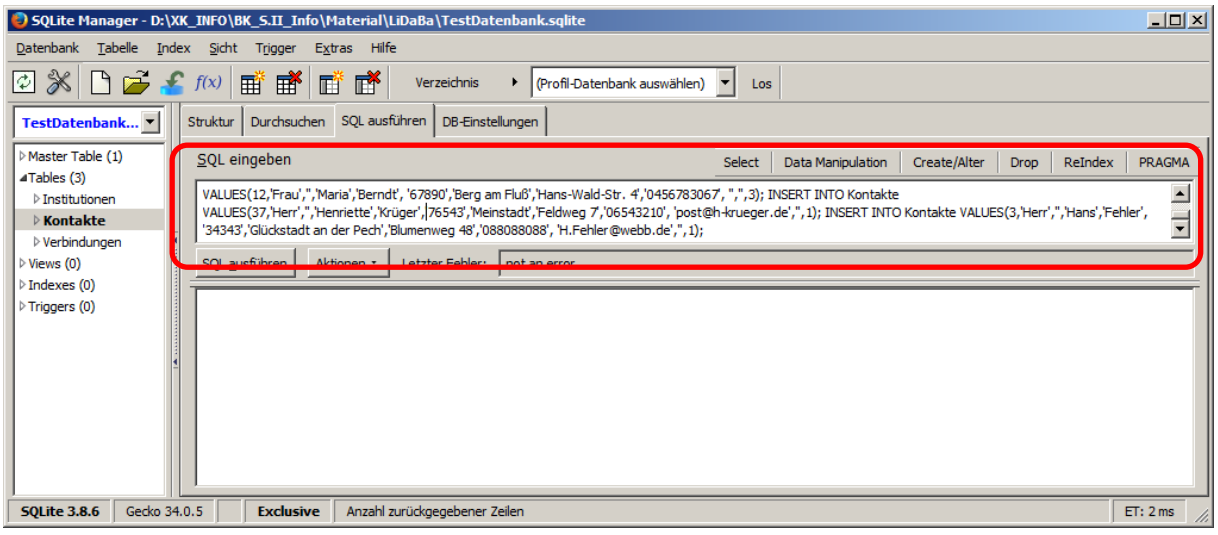
Arbeiten mit SQL



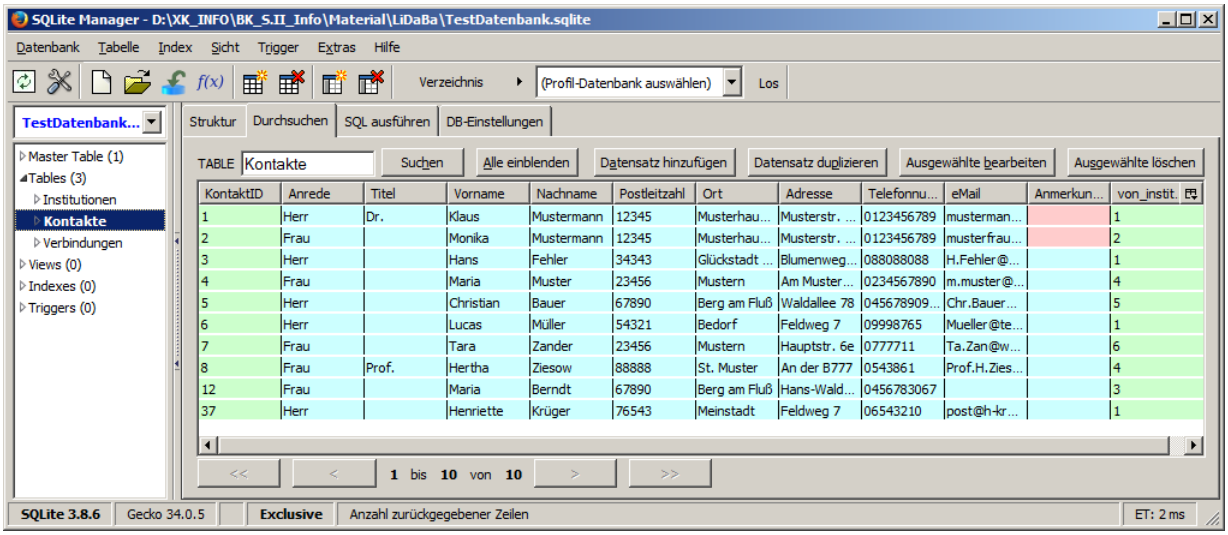
Obwohl im SQLite-Manager die Formular-Eingabe schon recht komfortabel ist, geht es mit einer direkten Eingabe unter "SQL ausführen" meist noch schneller



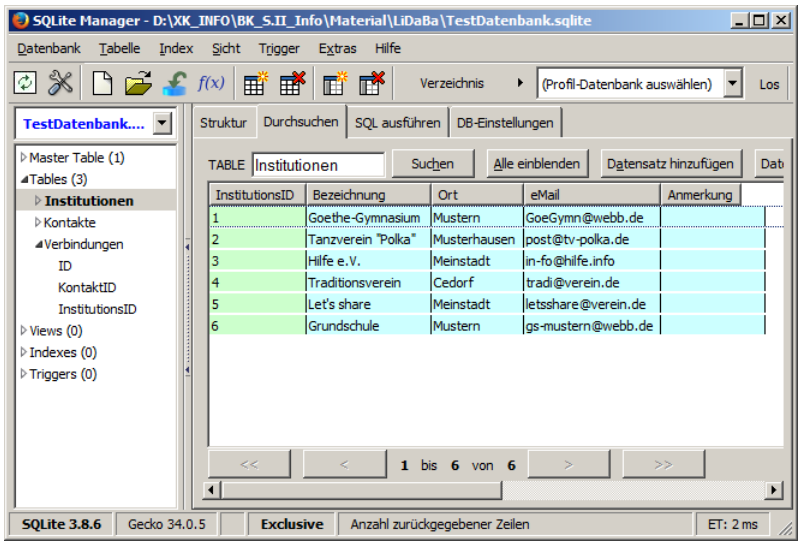
Das funktioniert auch für mehrere SQL-Anweisungen hintereinander. Die eingegebenen Anweisungen müssen dann Semikolon-getrennt notiert werden.

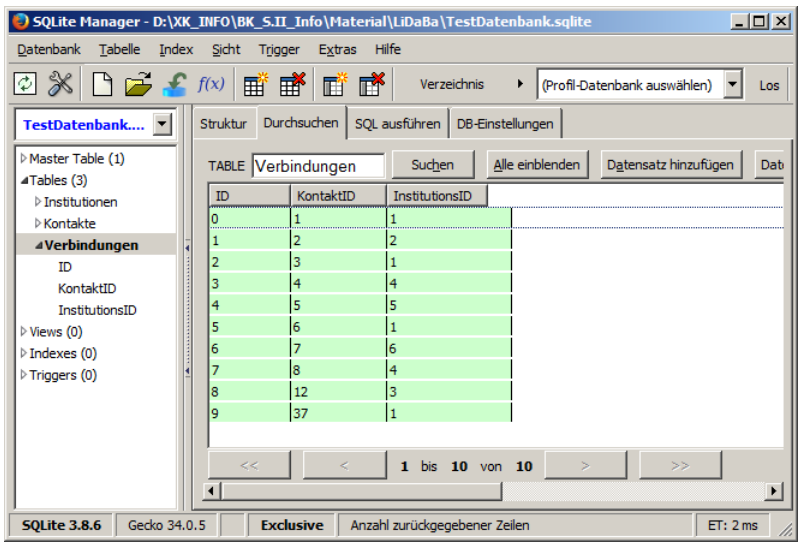


Am Schluss können wir uns die Tabelle mit den Daten nochmals ansehen. Der Reiter heißt etwas ungewönlich "Durchsuchen". Er bietet neben der reinen Tabelle auch einen kleinen Datensatz-Navigator unter der Tabelle.



Eingabe der Daten der anderen beiden Tabellen entweder über das Formular oder die SQL-Eingabe





Aufgaben:

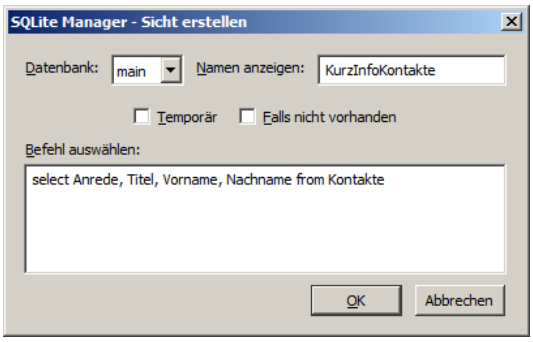
1. Vervollständigen Sie die Daten der "Verbindungen"-Tabelle!

3.1.7.1.2.x. Arbeiten mit der Beispiel-Datenbank (Kontakte-Institutionen)

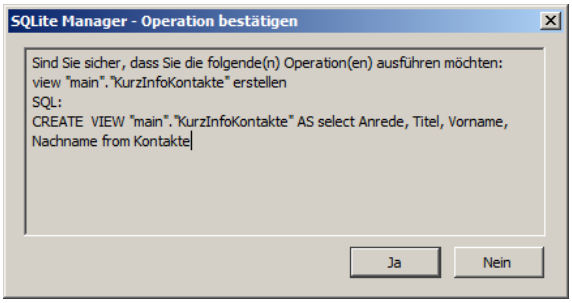
Abfragen

Neben den klasischen Tabellen und Indizies bietet der SQLite-Manager auch die Möglichkeit Abfragen zu erstellen. Sie heißen hier eingedeutscht "Sichten". In anderen Programmen oder Beschreibungen wird auch von "View"s gesprochen, was ja der wörtlichen Übersetzung entspricht.

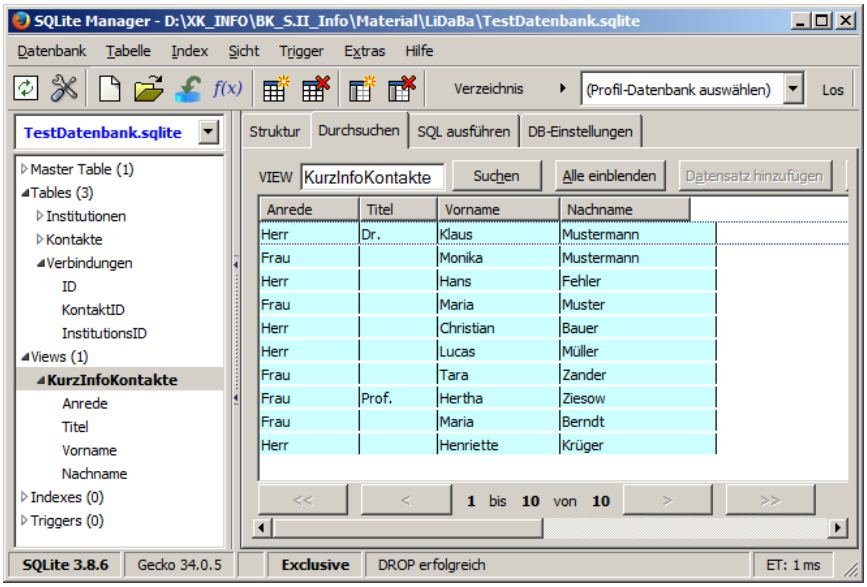
Eingeben als SQL-Anweisung



Bestätigung der übersetzten / korrigierten / angepassten SQL-Anweisung



Ergebnis taucht im Datenbank-Strukturbaum (links) unter "Views" auf und kann – wie üblich bei "Durchsuchen" angezeigt werden



Links:
Q: http://sqlite-manager.googlecode.com

3.1.7.1.2.x. Erstellen und Nutzen von Indizies

für SQLite Studio beschrieben → 3.1.7.1.3.4. Erstellen von Indizes

3.1.7.1.3. Arbeiten mit dem SQLiteStudio

Q: sqlitestudio.pl

SQLiteStudio ist ein Open-Source-Projekt (→ http://sqlitestudio.pl). Fertige ausführbare Dateien / Programme gibt es für die gängigen Betriebssysteme:

- Windows (9x → 7)
- Linux
- MacOS X
- UNIX

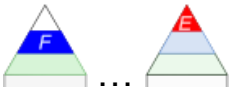
Das Studio ist eine graphische Benutzer-Oberfläche für das Datenbank-System SQLite.
Besonders für Nutzer von restiktiven Betriebssystem-Umgebungen ist interessant, dass SQLiteStudio eine portableApp ist. Das bedeutet, man muss das Programm nicht installieren. Es kann in der eingeschränkten Nutzer-Umgebung ausgeführt werden. Deshalb läuft es eben auch so gut vom IoStick oder auch als portable App im gleichnamigen Menü-System.
Wie bei Open-Source-Projekten üblich ist auch der gesamte Quell-Code zugänglich. Man kann das Programm also auch für ein anderes Betriebssystem compilieren, wenn ein passender Compiler verfügbar ist.
Wem das Programm nicht genug leistet und der Programmierung mächtig ist, kann sich gerne an der Weiterentwicklung des Programm's beteiligen. Auch kleine Geldspenden helfen den Entwicklern weiter.
Das SQLite Studio erinnert schon sehr stark an das Daenbank-Händling in den verschiedenen Betriebssystemen. Wer schon mal eine ODBC-Schnittstelle einrichten musste, weiss, was ich hier meine.
Im folgenden Kapitel beziehen wir uns auf die Version 3.1.1. (vom IoStick (2017)).

3.1.7.1.3.0. SQLiteStudio "installieren"

Wer den IoStick (Quelle: → https://tinohempel.de/info/info/IoStick/index.html) benutzt, hat schon eine vorbereite Version dabei und kann sofort mit dem Arbeiten beginnen (→ 3.1.6.1.3.1. SQLiteStudio starten).

Ansonsten lädt man sich die aktuelle Version (→ https://sqlitestudio.pl/index.rvt?act=download) – bzw. wenn neue Versionen ganz anders aussehen sollte und man weiter bei diesem Script bleiben will, dann die Version 3.1.1. – aus dem ZIP-Archiv (→ https://sqlitestudio.pl/files/sqlitestudio3/complete/zip/) herunter.
Die ZIP-Datei wird in einem beliebigen Ordner entpackt. In dem neu entstandenen Ordner "SQLiteStudio" findet man die Datei SQLiteStudio.exe. Wer die Endung EXE nicht sieht, kann die ausführbare Datei aber am Symbol (siehe oben) erkennen. Es emphielt sich eine Verknüpfung auf dem Desktop anzulegen, damit man sich nicht jedesmal bis zur EXE durchklicken muss.
Eine schöne und komfortable Möglichkeit ist die Einbindung von SQLiteStudio in ein vorhandenes PortableApps-System (Download: →https://portableapps.com/). Die gedownloadete ZIP-Datei von SQLiteStudio wird dann in den PortableApps–Ordner entpackt. Weiter sind keine Arbeiten notwendig. Beim nächsten Start des PortableApps-System steht die App im Menü-System unter "Sonstige" bereit.

SQLite Studio auf einem Raspberry Pi



Das SQLiteStudio kann auch auf einem Raspberry Pi verfügbar gemacht werden. Im Raspian – einer gängigen Linux-Distrubution für den Rasp Pi – ist das Studio aber nicht enthalten. Es kann aber nachträglich installiert werden. Dazu benutzt man die nachfolgenden – teilweise auch kommentierten – Befehle in einer Konsole. Jede Zeile ist dabei ein Konsolen-Befehl, der i.A. auch immer irgendwelche Kontrollanzeigen generiert. Einige Befehle können auch langerfristige Aktionen / Downloads auslösen. Man sollte also etwas mehr Zeit einplanen.
Die erste – jetzt folgende – Konsolen-Kommando-Sequenz kann entfallen, wenn SQLite schon fuktioniert. Es schadet aber nicht die Kommando's nochmals einzugeben.

`$ sudo apt-get update`	
`$ sudo apt-get dist-upgrade`	
`$ sudo apt-get install build-essential tcl`	kann entfallen
`$ sudo apt-get install libreadline-dev ↵ libncurses5-dev`	
`$ sudo apt-get install sqlite libsqlite3-dev`	kann entfallen, falls SQLite schon läuft

↵ … direktes Weiterschreiben in der Kommando-Zeile

Die nächsten Kommando's laden die aktuelle Version von SQLite Studio herunter und installieren dann das Programm. Die Versions-Nummern müssen jeweils angepasst werden, da das Projekt ständig weiterentwickelt wird.

`$ sudo apt-get install qt5-default qtscript5-↵ dev qttools5-dev libqt5svg5-dev`	
`$ mkdir -p ~/projects/sqlitestudio/`	
`$ cd ~/projects/sqlitestudio/`	
`$ wget http://sqlitestudio.pl/files/sqlite↵ studio3/complete/tar/sqlitestudio-3.1.1.tar.gz`	Nummern ev. durch aktuelle / andere Version ersetzen
`$ tar xf sqlitestudio-3.1.1.tar.gz`	bei Problemen beim Übersetzen später, lieber auf die letzte verfügbare 3.0.x Version zurückgreifen
`$ mkdir -p output/build/`	
`$ cd output/build/`	
`$ qmake ../../SQLiteStudio`	
`$ make -j4`	-j4 für Rasp Pi3, sonst ohne
`$ make -j4 install`	

↵ … direktes Weiterschreiben in der Kommando-Zeile

Starten von SQLiteStudio unter Raspian /

Nun kann eine neue Datenbank angelegt werden (→ 3.1.6.1.3.2. eine Datenbank erstellen).

Links / Quellen:
http://www.raspberry-pi-geek.de/Magazin/2016/05/Arbeiten-mit-SQLiteStudio

SQLite Studio auf einem Mac-Rechner

SQLiteStudio gibt es auch als Mac-Version. Den Download findet man unter → . Die Installation läuft Mac-typisch ab. Das ist sicher auch die einfachste Variante.
Will man mit dem IoStick arbeiten tut sich ein großes Problem auf. Alle Programme des IoStick, wie auch das Menü-System sind nur für Windows-Rechner gedacht.
Es gibt aber verschiedene Möglichkeiten aus der Patsche zu kommen.
Man benötigt ein Installations-Medium für Windows und optimalerweise eine gültige Lizenz. Ansonste kann man nur die 6 Wochen Testzeitraum nutzen. Für einmal SQL sollte das reichen.

Möglichkeit 1: Nutzung einer virtuellen Maschine (virtueller Rechner)

Vorteile:

- leicht wieder zu entfernen (Löschen der virtellen Maschinen, Löschen des Managers für die virtuellen Maschinen → fertig)
- einfache Übertragung von Daten zwischen Mac und Win
- Naht-loses Umschalten zwischen dem Arbeiten in Mac und Win
-

Nachteile:

- Installation eines zusätzlichen Software-Paketes notwendig (Manager für die virtuellen Maschinen)
- großer Festplatten-Bedarf für die virtuellen Maschinen
- Leistungs-Verlust (Rechner wird langsamer; virtuelle Maschine ist langsamer und Leistungs-begrenzt)
-

Möglichkeit 2: Nutzung einer Parallel-Installation (Multiboot)

Vorteile:

-

Nachteile:

-

1. Installation / Nutzung von VirtualBox

4. Installation / Nutzung von Parallels Desktop

Vorteile:

-

Nachteile:

-

4. Installation / Nutzung von Vmware Fusion

Vorteile:

-

Nachteile:

-

2. Installation / Nutzung von Windows parallel zum MacOS (Apple Bootcamp)

Vorteile:

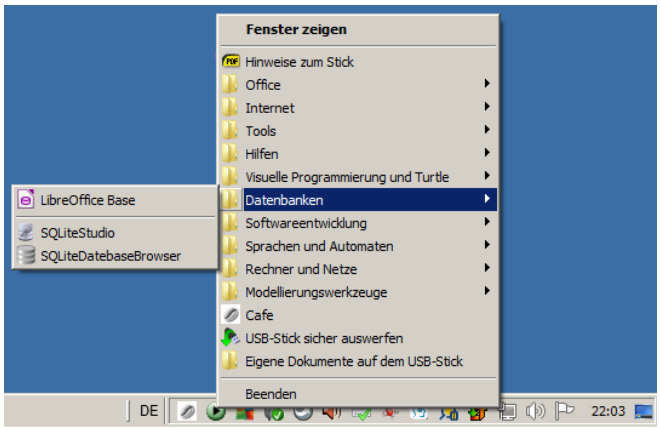
-

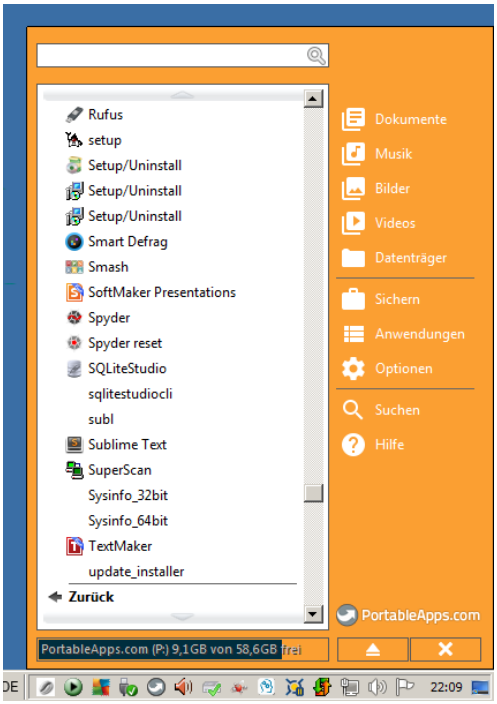
Nachteile:

-

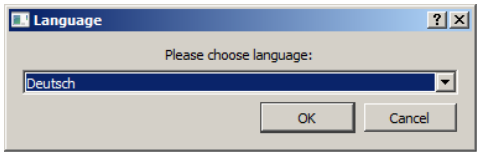
3.1.7.1.3.1. SQLiteStudio starten

Die IoStick-Benutzer finden einen Menü-Eintrag von SQLiteStudio" unter "Datenbanken" (s.a. Abb. rechts).
Wer die ZIP-Datei von SQLiteStudio in einen portApps-Ordner eines funktionierenden PortableApps-System entpackt hat, findet SQLiteStudio im Prgrammordner "Sonstige" (s.a. Abb. unten).

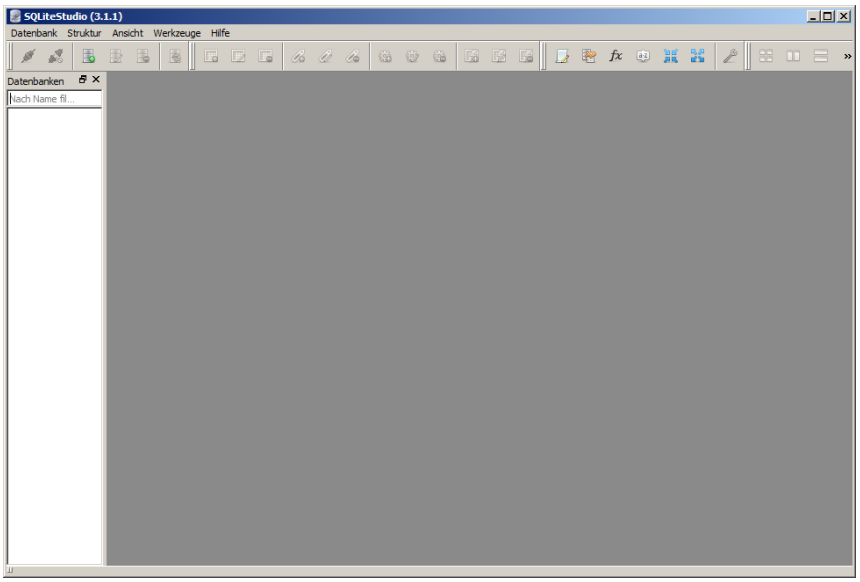




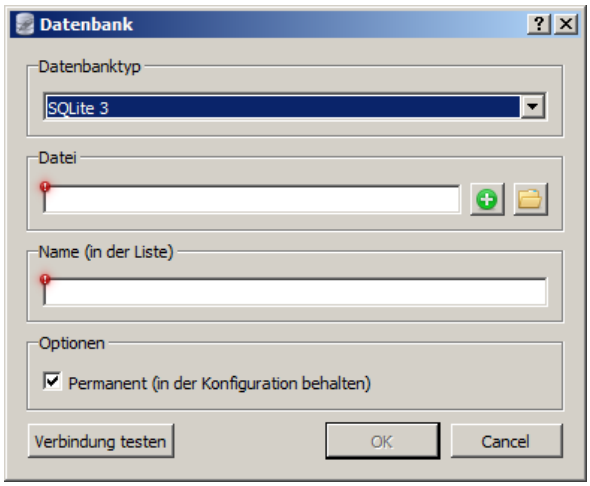
Bei der Erstbenutzung werden wir nach der gewünschte Sprache gefragt. Obwohl nicht alle Programm-Elemente übersetzt sind, hilft eine Sprachauswahl auf "Deutsch" bei der ersten Arbeit mit dem Programm. Später kann man dann bei "American English" bleiben.



Das leere Programm-Fenster zeigt uns de derzeit leeren Zustand unseres Datenbank-Management-Systems an.



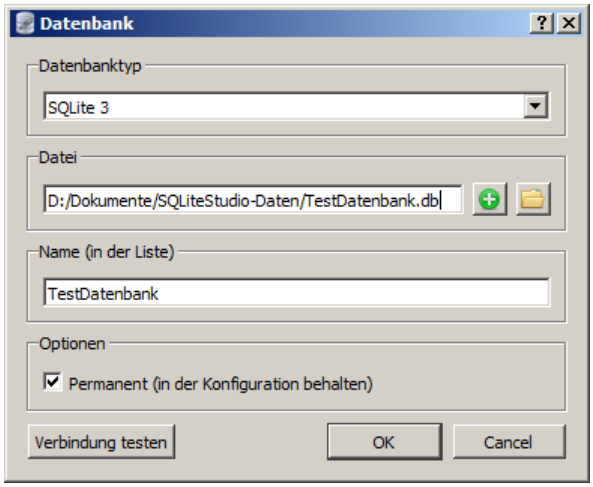
3.1.7.1.3.2. eine Datenbank erstellen



"Datenbank" "Datenbank hinzufügen"
oder [Strg] [o]

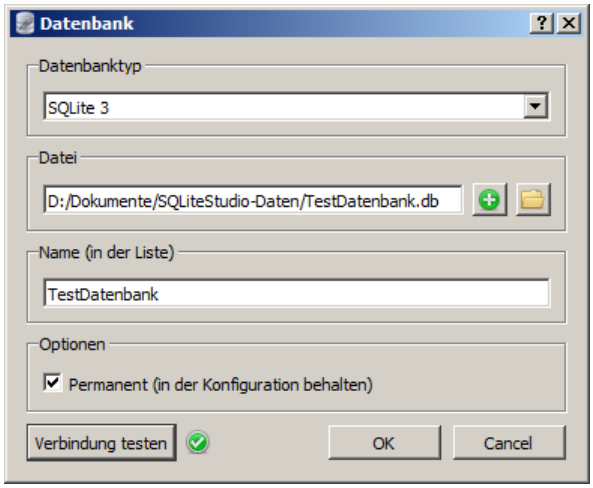
die kleinen roten Ausrufezeichen markieren die notwendigen Minimal-Eingaben

Die Datei in der die gesamte Datenbank gespeichert werden soll, legen wir über die Schaltfläche mit dem "grünen Plus"-Zeichen fest
alternativ können wir eine bestehende Date über das "Ordner"-Symbol öffnen



aus dem Dateinamen leitet SQLiteStudio einen Datenbank-Namen ab, den man aber ändern kann

die Option "Permanent" bestimmt, ob unsere Datenbank dauerhaft in der Datenbank-Liste links aufgeführt werden soll
das ist für das Weiterarbeiten beim nächsten Mal sehr praktisch



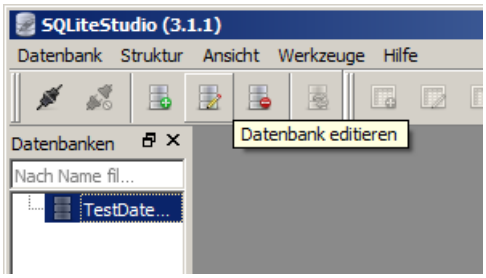
zu guter Letzt testen wir, ob die Verbindung zum internen SQL-Server steht

mit "Verbindung testen" geht das schon vor der abschließenden Bestätigung über die Anlage der Datenbank

das grüne Häkchen bestätigt die Funktionsfähigkeit

Sachlich kann eine SQL-Datenbank auch in anderen Anwender-Programmen genutzt werden. Das entspräche dann schon mehr einer Server-Client-Datenbank.

nachträglichen Ändern der Datenbank-Anbindung ist möglich über Symbolleiste oder Menü "Datenbank" "Datenbank editieren"

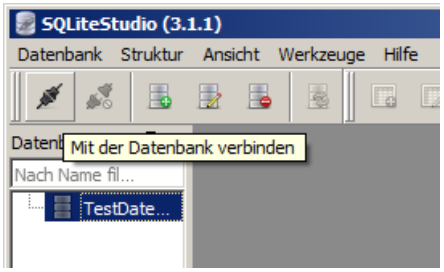


ebenso das Löschen einer Datenbank
z.B. über "Datenbank" Datenbank entfernen"

Mit der Datenbank verbinden

zum Arbeiten mit der Datenbank muss man sich mit der Datenbank verbinden

wir arbeiten sozusagen mit unserem Programm als Client und brauchen eine Verbindung zum eigentlichen (hier Programm-internen) SQL-Server

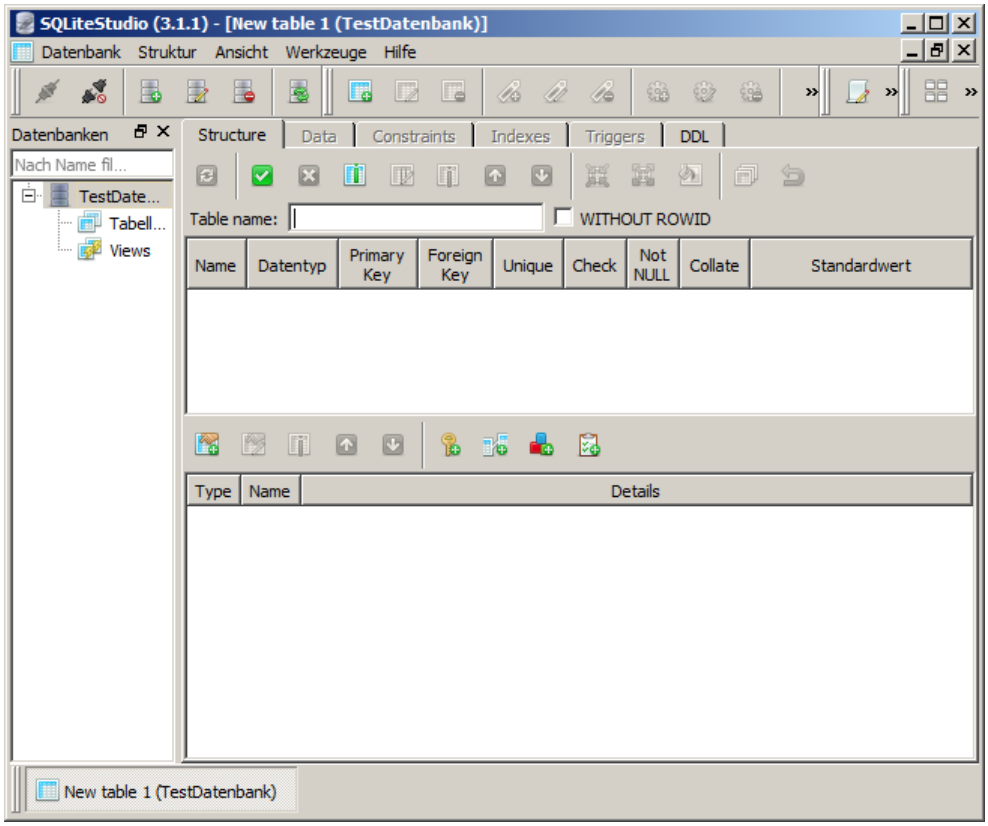


Stecker-Button in der Symbolleiste oder "Datenbank" "Mit der Datenbank verbinden"
Hat die Verbindung geklappt, dann sehen wir unter dem Datenbank-Eintrag in derListe die Struktur unserer Datenbank

es gibt zwei Bereiche, einmal die Tabelle und zum Zweiten die Views (Abfragen, Sichten)

die Tabellen enthalten die eigentlichen Daten

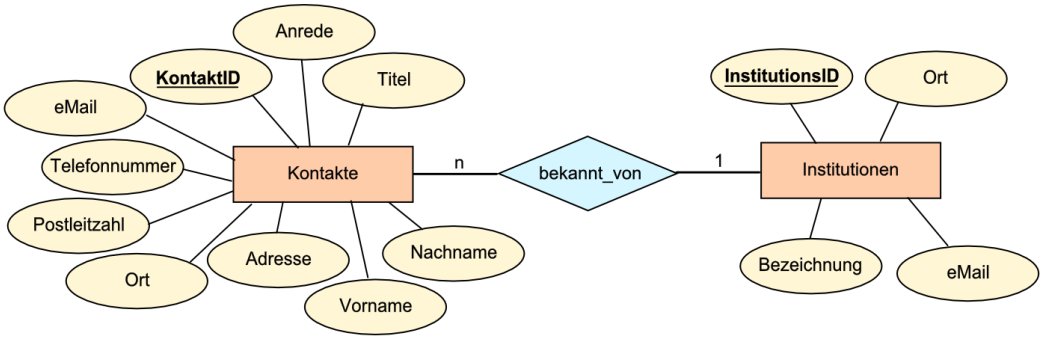
mit den Views bekommen wir bestimmte Sichten auf unsere Datenbank bzw. auf Teile / Ausschnitte davon.



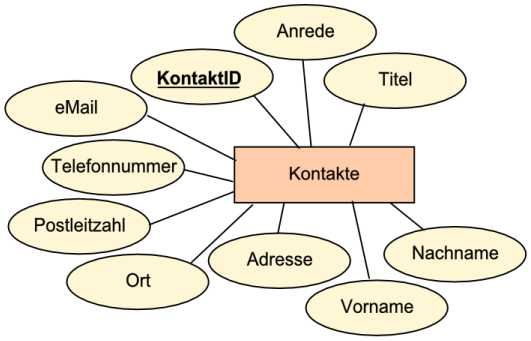
Aufgaben:

1. *Erstellen Sie sich im SQLiteStudio eine Datenbank "TestDatenbank", speichern die zugehörige Datei in Ihrem Arbeits- / Home-Ordner!*
2. *Verbinden Sie sich dann mit der Datenbank!*

Realisierung der gleichen Datenbank, wie bei den anderen Systemen. Das zu realiserende Entity-Relationship-Modell sieht so aus:



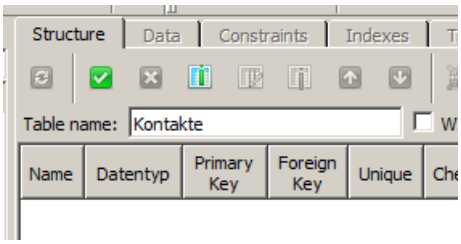
die erste zu erstellende Tabelle sind die Kontakte



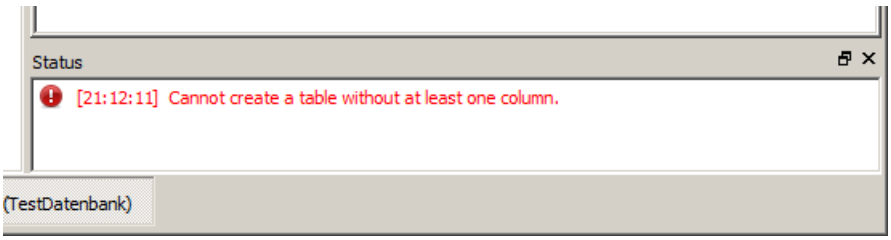
Unter dem Reiter "Structure" erstellen wir jetzt Spalte für Spalte unsere Tabellen-Struktur – also die Spalten-Überschriften (Attribute) und legen auch deren Datentyp fest.

Zuerst bestimmen wir den Namen der Tabelle. Laut derm ERD soll die Tabelle "Kontakte" heißen.

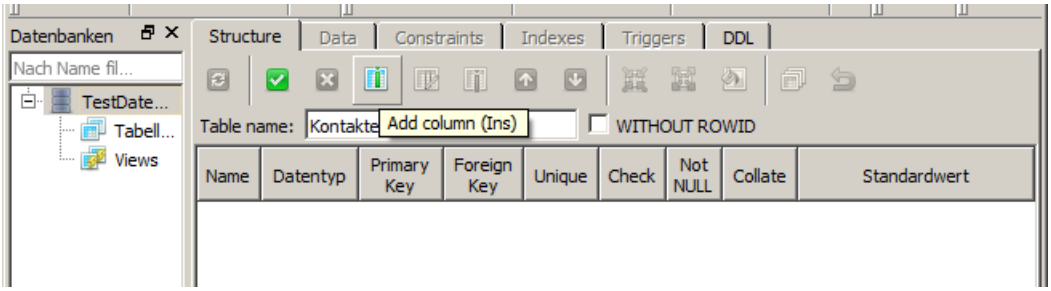
Erst, wenn alle Angaben zur Tabelle eingetragen sind, bestätigen wir die Struktur mit dem "Grünen Häkchen" ("Commit structure changes")



Geschieht die Bestätigung verfrüht, dann bekommt man i.A. eine Fehlermeldung im Fensterchen "Status".



Zum Anlegen einer Spalte wählt man das Symbol mit der Tabelle und der grünlichen Spalte. Der Hilfe-Text (in engl.) "Add column (Ins)" ist da sicher verständlich.



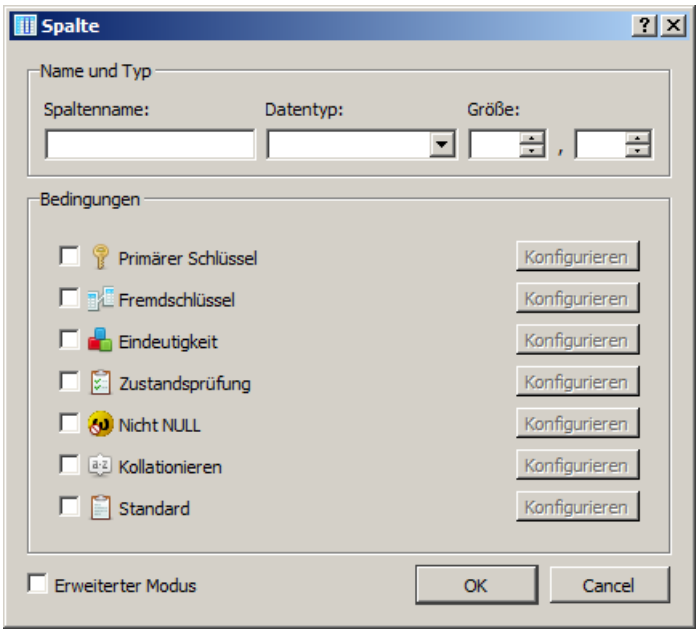
Für jede Spalte sind nun diverse Angaben zu machen. Dabei sind der Spaltenname und der Dateityp zwingend erforderlich.

bei anderen Angaben richtet sich die Angabe-Pflicht mehr nach dem Modell unserer Datenbank

ein planmäßiges, genaues und zielgrerichtetes Arbeiten ist hier aber notwendig

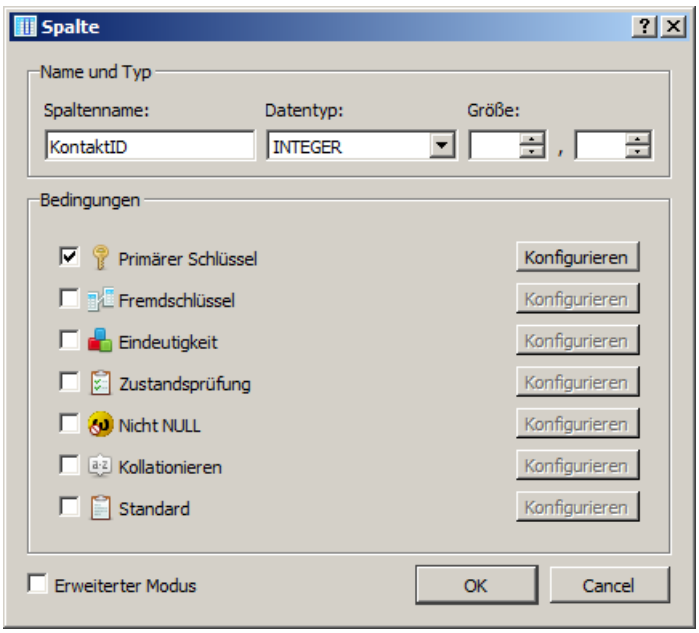
zwar sind einzelne Optionen auch nachträglich veränderlich, aber so etwas ist in Datenbanken allgemein nicht zu empfehlen

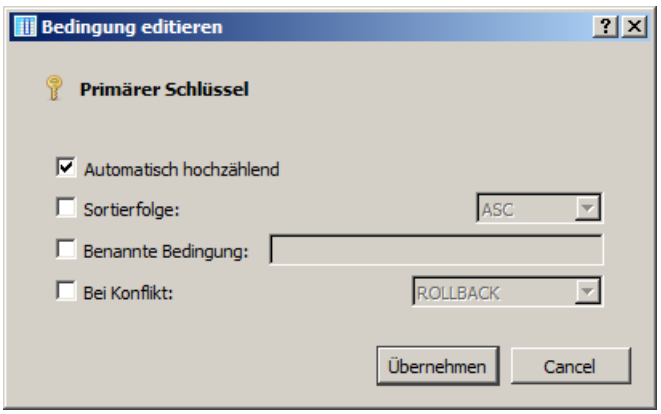
alle Bedingungen / Details lassen sich "Konfigurieren", d.h. hier sind zusätzlichen Einstellungen / Optionen möglich



Bedingungen

- **Primärer Schlüssel** (Primary Key)
  Spalte enthält den Primär-Schlüssel dieser Tabelle
- **Fremdschlüssel** (Foreign Key)
  Spalte enthält einen Fremd-Schlüsse
- **Eindeutigkeit** (Unique)
  Werte in der Spalte müssen eindeutig sein und dürfen sich somit nicht wiederholen
- **Zustandsprüfung** (Check condition)
  Festlegung von Prüf-Bedingungen für die Eingaben
- **NichtNULL** (Not NULL)
- **Kollationieren** (Collate)
- **Standard** (Default)
  voreingestellter Belegungs-Wert

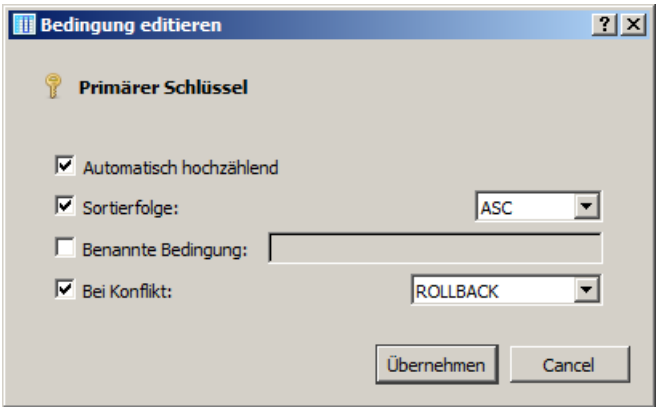




Konfigurationen / Optionen für den Primär-Schlüssel

- **Automatisch hochzählend** (Autoincrement)
  SQLite übernimmt das Erstellen neuer Primärschlüssel-Werte
- **Sortierfolge** (Sort order)
  legt fest, wo der neue Datensatz eingeordnet werden soll
- **Benannte Bedingung** (Named contraint)
- **Bei Konflikt** (On conflict)
  Konflikt-Behandlung; Was soll Widersprüchen, fehlerhaften oder Grenz-überschreitenden Werten passieren?

minimal sollte man hier "Automatisch hochzählend" setzen, dann braucht man sich um den Primärschlüssel nicht mehr weiter kümmern.
recht praktisch ist auch das Setzen der Optionen "Sortierfolge:" und "Bei Konflikt:". Ein ROLLBACK verhindert die Aufnahme nicht regulärer Datensätze in die Tabelle

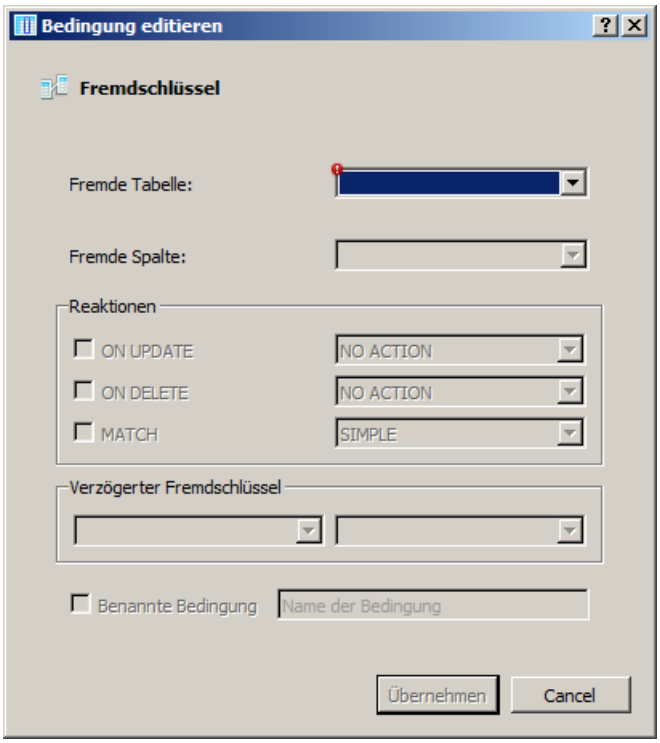


Hinweis: Für unsere Tabelle "Kontakte" mit vorgegebenen Schlüsselwerten geht das nicht!

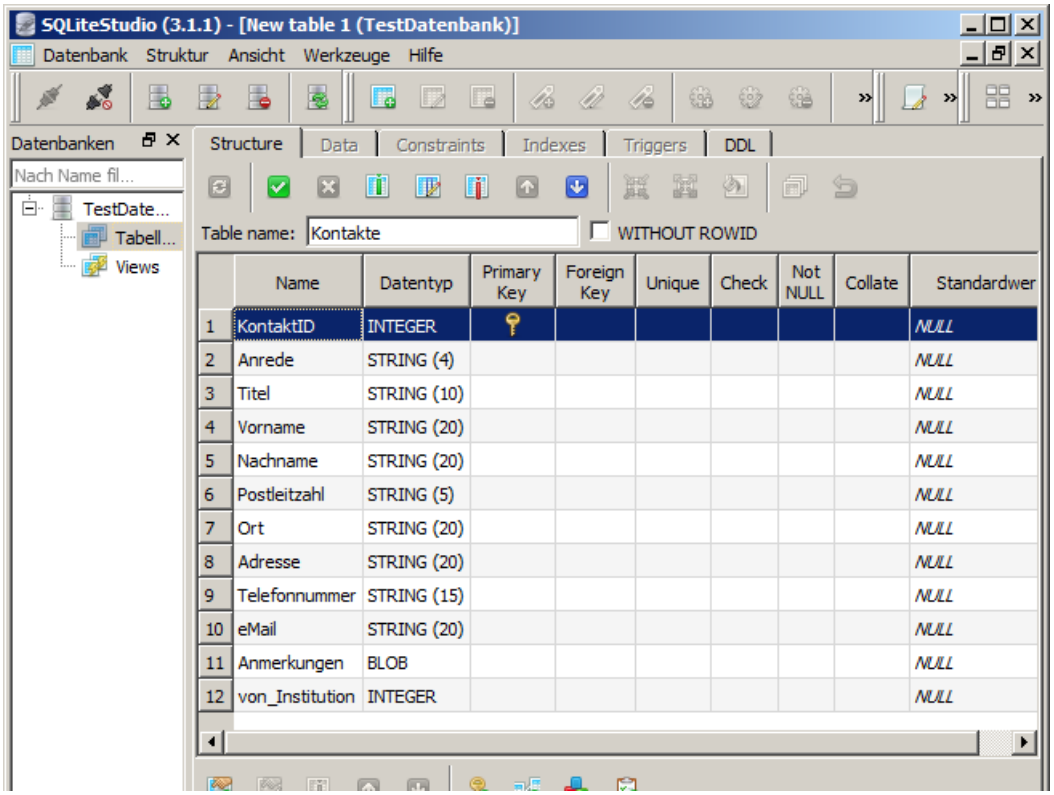
die letzte Spalte soll einen Fremdschlüssel

das bekannte rote Erinnerungs-Ausrufezeichen verweist uns auf die notwendige Konfiguration des Fremdschlüssels

auch hier zeigt das Ausrufezeichen auf eine notwendige Angabe. Und da haben wir ein Problem: Wir haben noch gar keine Tabelle zum Verweisen.
hier müssen wir die Konfiguration des Fremdschlüssels also abbrechen ("Cancel")
die Verbindung dieser Tabelle mit der noch zu erstellenden Tabelle "Institutionen" müssen wir später noch nachholen



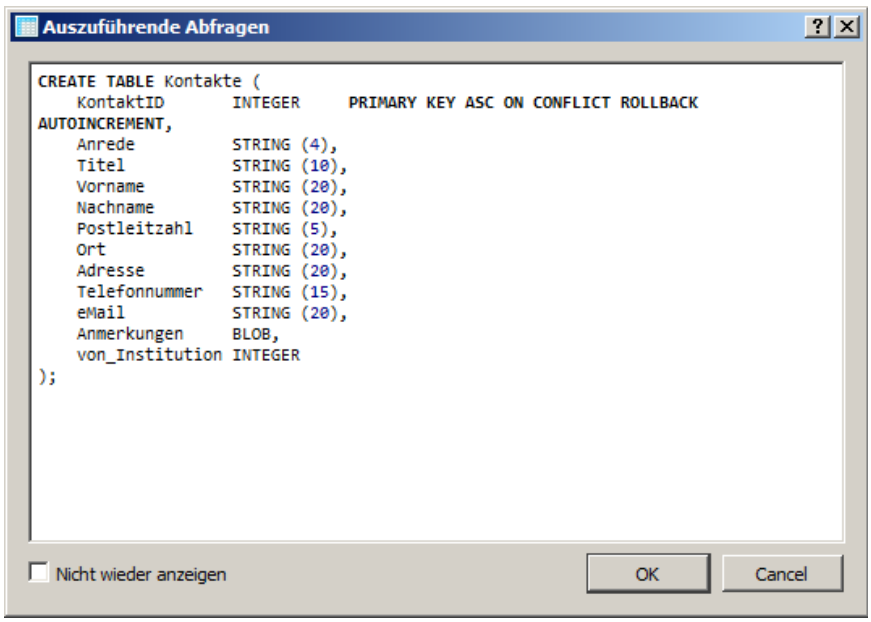
Um also Tabellen insgesamt mit möglichst wenig Verweis-Problemen zu erzeugen, ist es sinnvoll die gesamte Tabellen-Struktur rückwärts anzulegen.
In unserem Beispiel also zuerst die Tabelle "Institution" und dann die eben besprochene Tabelle.



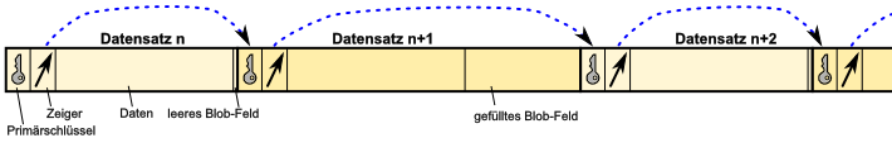
Dem aufmerksamen Leser ist sicher aufgefallen, dass ich hier wieder eine zusätzliche Spalte mit definiert habe – die "Anmerkungen". Die Anlage eine Spalte für irgendwelche Bemerkungen usw. würde ich immer empfehlen. In diese Spalte kann man für sich selbst oder andere Nutzer Hinweise hinterlassen oder Daten speichern, für es keine reguläre Spalte gibt. Der ungewöhnliche Datentyp "BLOB" steht für ein beliebig großes Textfeld. Steht nichts im Feld, dann wird auch praktisch kaum Speicherplatz gebraucht.

Auch später hat das Feld immer nur den Speicherplatz-Bedarf für soviele Textzeichen, wie wirklich eingetragen sind.

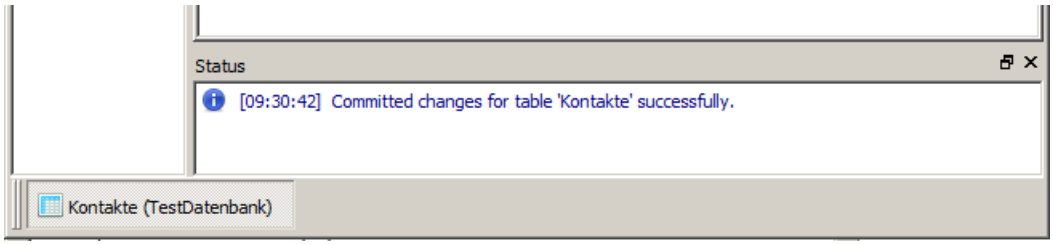
Auf den ersten Blick könnte man jetzt annehmen, dass Blob-Felder sehr günstig für Datenbanken sind. Das gilt aber nur für den reinen Speicherplatz. Durch sehr viele Blob-Felder könnte eine Tabelle aber langsamer durchsuchbar werden, da ja jedesmal der Beginn des nächsten Feldes / Datensatzes berechnet werden muss.



Die Erreichbarkeit der nachfolgenden Datensätze wird über Zeiger (Pointer, Sprung-Adressen) realisiert.

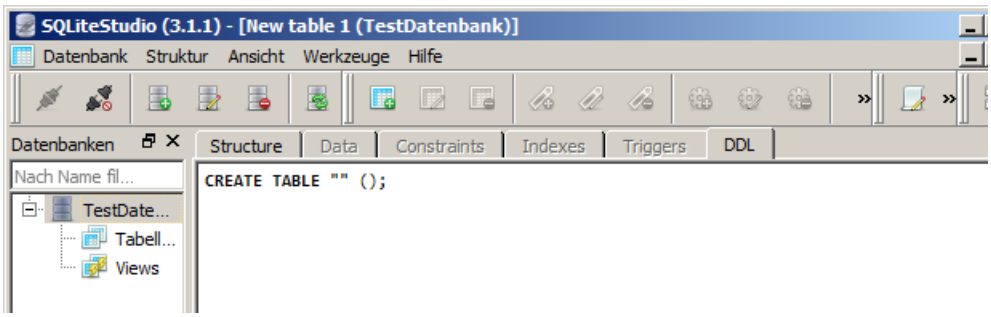


Deshalb ist es egal, ob ein Blob-Feld leer oder gefüllt ist.
Nach dem Bestätigen erhalten wir die Anzeige des kreierten SQL-Befehls für unsere zusammengeklickte Tabellen-Definition.
Die nachfolgende Status-Information ist unser Ziel.

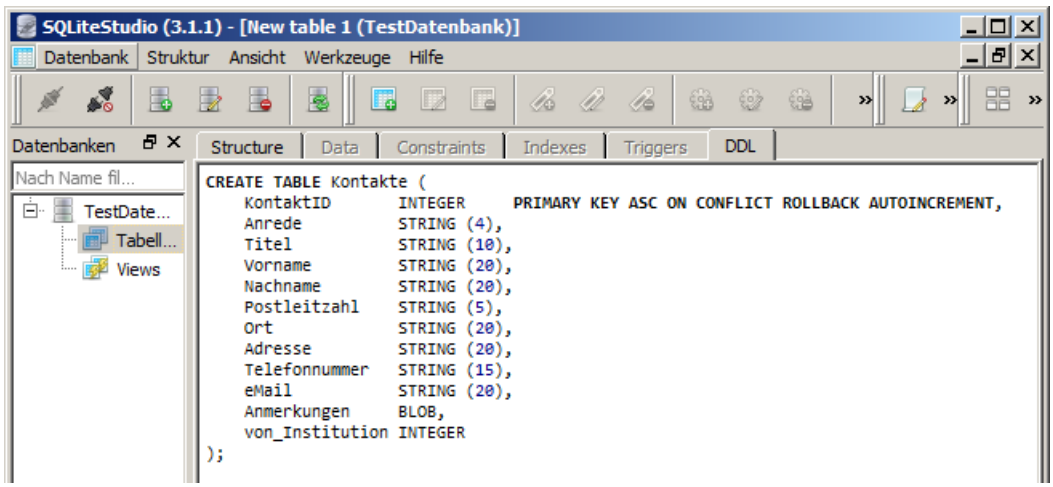


Dann ist alles ok. und wir können uns um das nächste Problem kümmern.

Unter dem Reiter "DDL" (Data Defination Language / Date Description Language) lässt sich die Struktur der Tabelle auch über einen SQL-Befehl erzeugen.



Jeder Definitions-Schritt wird gleich in SQL-Code umgesetzt, so dass am Ende der oben schon gesehene SQL-Befehl herauskommt. Hier könnten wir aber noch in die Befehls-Gestaltung eingreifen und Änderungen vornehmen.

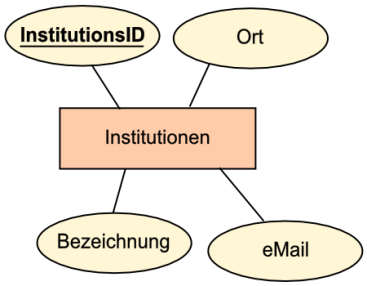


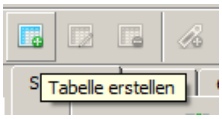
Aufgaben:

1. Erstellen Sie die Tabelle "Kontakte" mit den angezeigten Optionen!

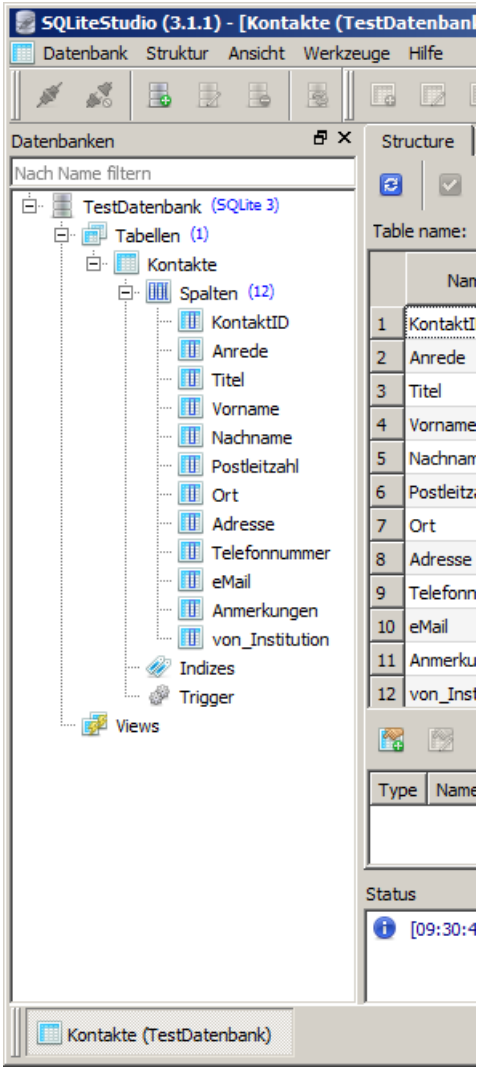
2. Erläutern Sie, was die Ausdrücke in der Zeile "KontaktID …" bedeuten!

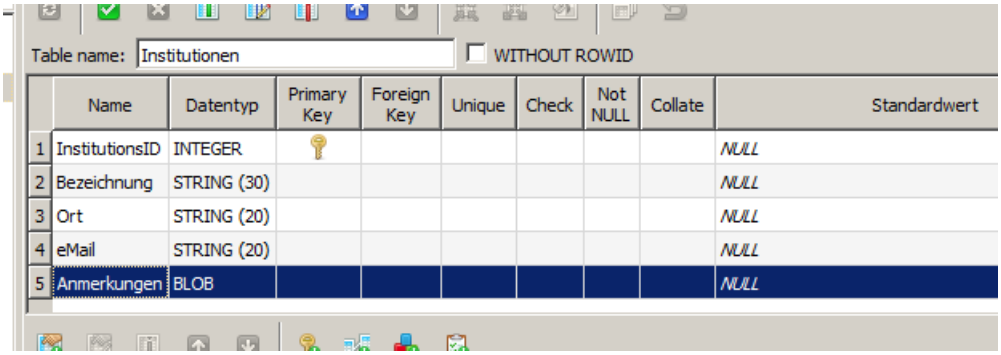
Die ordnungsgemäße Erstellung unserer Kontakte-Tabelle wird jetzt auch in der Datenbank-Struktur-Anzeige links sichtbar. Der Baum lässt sich an den Plus-Symbolen aufspreizen. Die Informationen sind dann schnell verfügbar und wir brauchen dann unsere Papier-Definition nicht mehr.
Als nächstes brauchen wir die Institutions-Tabelle. Das ERD sieht so aus:



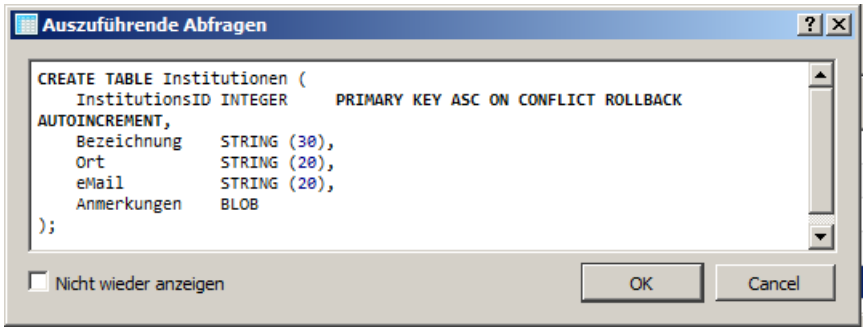


Mittels "Tabelle erstellen" wiederholen wir das eben besprochene Verfahren. Die Details sind hier weniger umfangreich.





Der SQL-Befehl wird uns wieder angezeigt. Praktisch kann man diese Anzeige zukünftig auch verhindern. Dazu muss man nur "Nicht mehr anzeigen" anklicken. In der Phase des Erlernen von SQL bzw. den ersten Arbeiten mit Datenbanken und SQL sollte man sich die Texte aber ruhig ansehen.



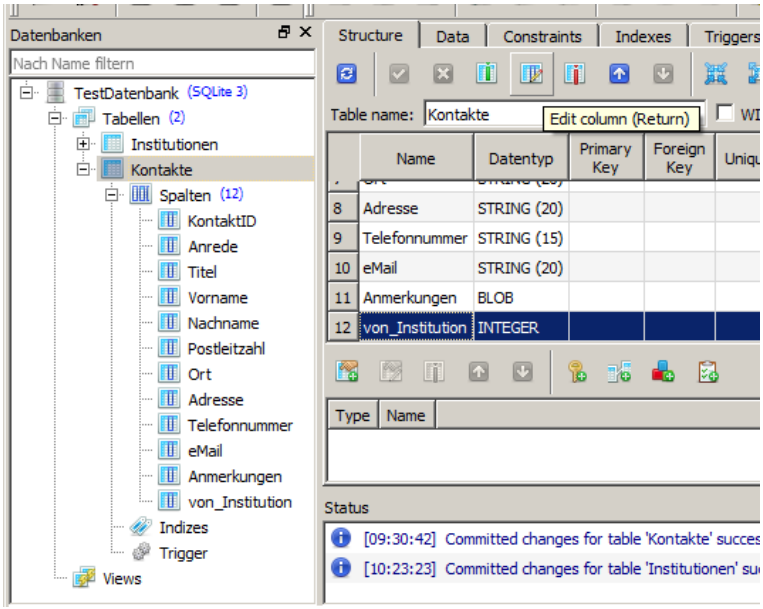
Man bekommt recht schnell ein Gefühl für die Struktur der Befehle und mit auch schon geringen Englisch-Kenntnissen kann man die Wirkung der erkennen. Das hilft auch bei der Suche nach Fehlern in SQL-Befehlen.

Aufgaben:

1. Erstellen Sie sich nun die Tabelle "Institutionen", wie oben gezeigt!

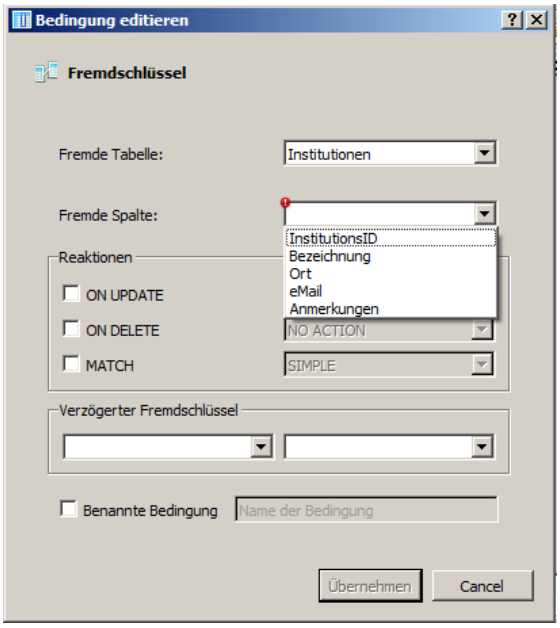
2.

Tabellen verknüpfen – Fremdschlüssel (nachträglich) einrichten

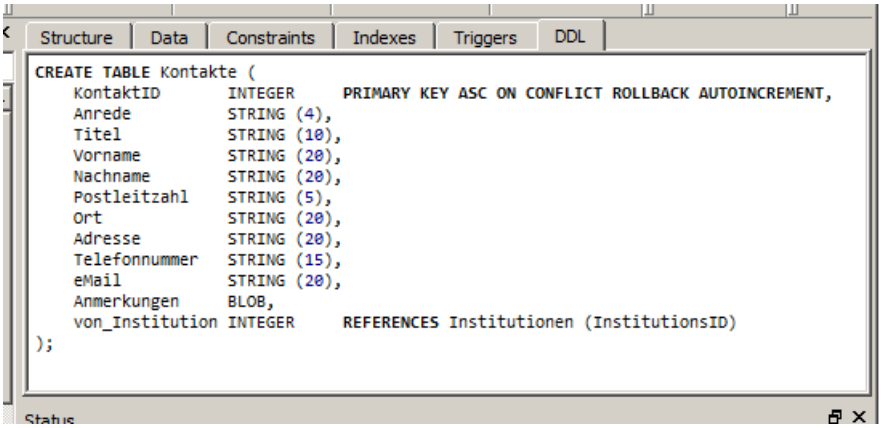


Nach der Auswahl der (zukünftigen) Femdschlüssel-Spalte kommt man entweder über das Editier-Symbol ("") oder mittels [Enter]-Taste in die Detail-Anzeige.

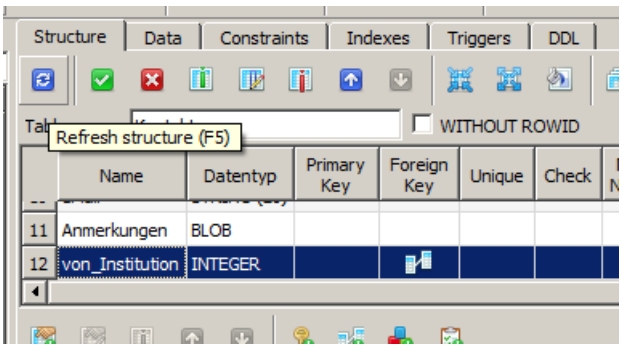
In der Konfiguration muss nun zwingend die verknüpfte Tabelle eingetragen werden. Ausdieser können wir dann die passende Primärschlüssel-Spalte auswählen. Ihre Werte werden dann später in der Kontakte-Tabelle als Verweis auf eine Institution verwendet.



Interessant sind die Möglichkeiten, auf bestimmte Vorgänge / Veränderungen einzugehen. Dabei kann man definieren, was passieren soll, wenn die Daten aktualisiert werden (ON UPDATE), wenn sie gelöscht werden (ON DELETE) und / oder, wenn sie mit einem Wert übereinstimmen (MATCH).
Der Rest ist für die Profi's und für uns nicht wirklich wichtig.
Unter dem Reiter DDL sehen wir nun die Definition unserer Tabelle.
Allerdings ist das der SQL-Befehl für die "Ersterstellung" der Tabelle. Das haben wir ja schon erledigt. Hier soll ja "nur" die Struktur angepasst werden.
Ein Struktur-Refresh [F5] ist nicht die Lösung unseres Aktualisierungs-Problems!



Beim Refresh wird die derzeit gültige Struktur in die Anzeige gebracht. Also jetzt **nicht** machen, sonst muss die Einrichtung des Fremdschlüssels noch einmal erledigt werden!
Hier ist eine Neudefinition der Tabelle "Kontakte" notwendig.
Wie aufwendig das ist zeigt der SQL-Befehl, der nach dem Aktivieren des Grünen Häkchen vorgeschlagen wird.

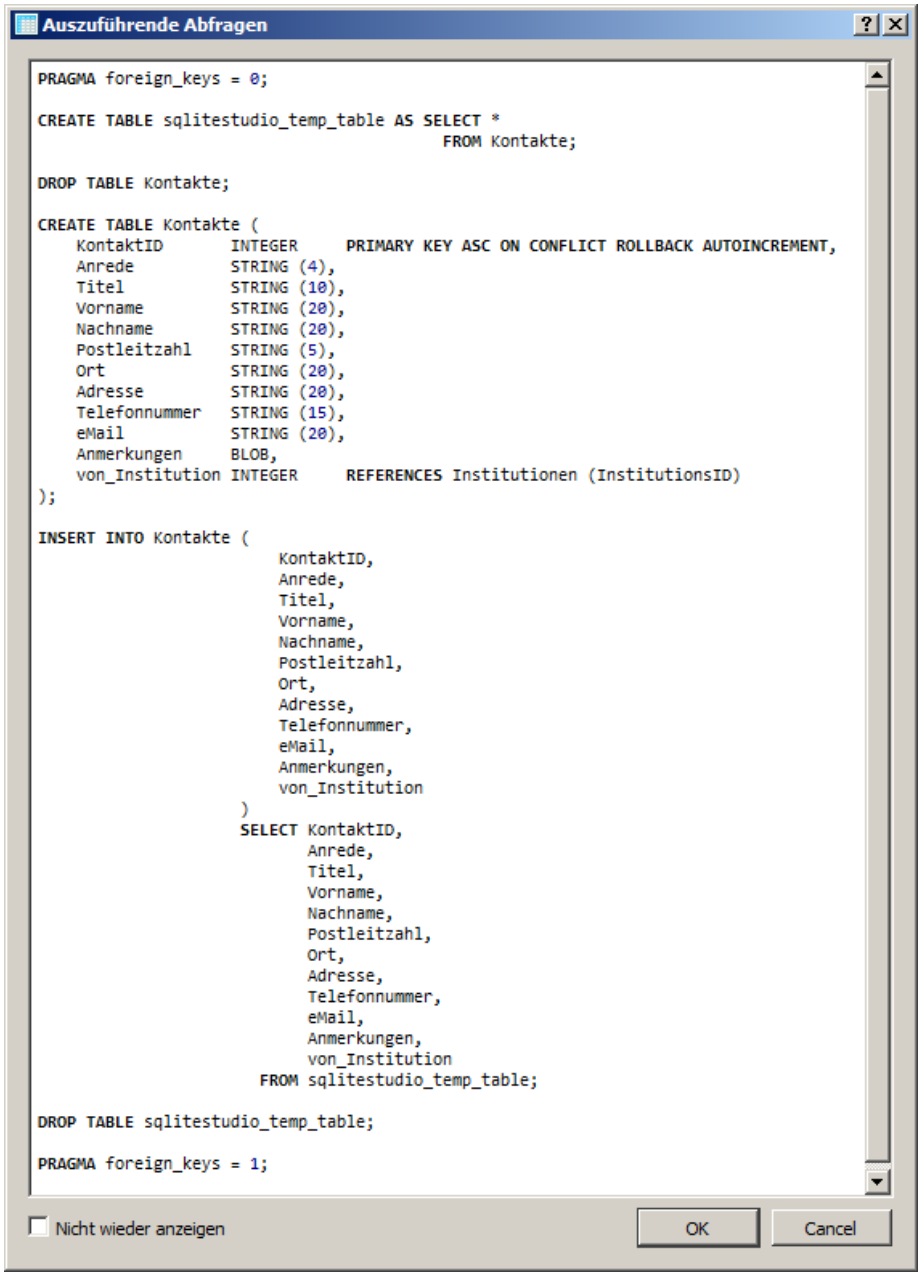


Wie komplex eine einfache Änderung an der Tabellen-Struktur werden kann, sieht man schön an dem zugehörenden SQL-Statement.

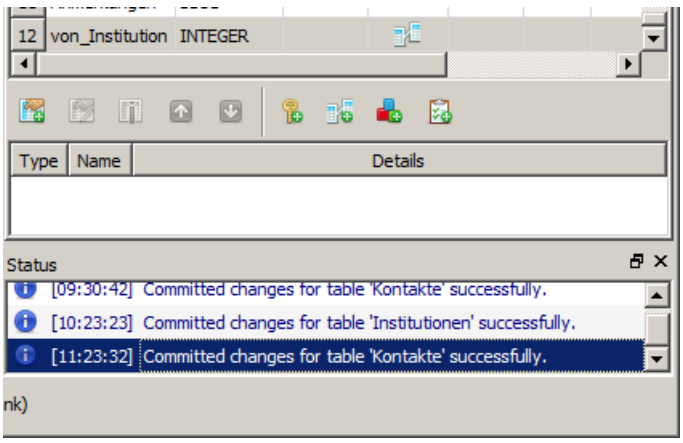
Aufgaben:

1. ***Stellen Sie die Verknüpfung zwischen den Tabellen "Kontakte" und "Institutionen" her!***
2. ***Warum benutzt man hier eigentlich eine zusätzliche Tabelle für die "Institutionen"? Man hätte die Daten doch gleich in die Tabelle "Kontakte" eintragen können. Erklären Sie ausführlich!***

Dieser SQL-Konstrukt ist wegen der lapidaren Änderung um die Erstellung eines Fremd-Schlüssels notwendig.
Was hier passiert ist auch ohne fortgeschrittene SQL-Kenntnisse interessant. Kurz erklärt geht das System so vor:
Es wird zuerst dafür gesorgt, dass die nachfolgenden Befehle ungestört hintereinander abgearbeitet werden können. Als Nächstes wird von der alten Tabelle eine Kopie als temporäre Tabelle erstellt. Sie dient dann später als Daten-Quelle für die neue Kontakte-Tabelle. Vorher wird die alte Tabelle Kontakte gelöscht.
Nach dem Erstellen der neuen Kontakte-Tabelle mit dem Fremdschlüssel folgt das Spalten-weise Kopieren der Daten aus der temporären Tabelle in die neue. Das in unserem Fall noch gar keine Daten enthalten waren spielt hier keine Rolle. Zum Schluss wird die temporäre Tabelle noch gelöscht.



In der Status-Anzeige bekommen wir ein positives Feedback.

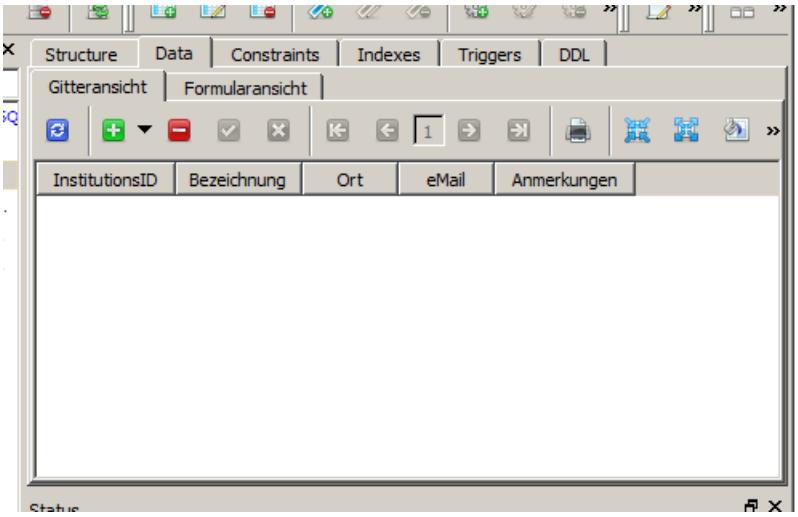


3.1.7.1.3.3. Daten in die Tabellen eingeben

manuelle Eingabe von Daten

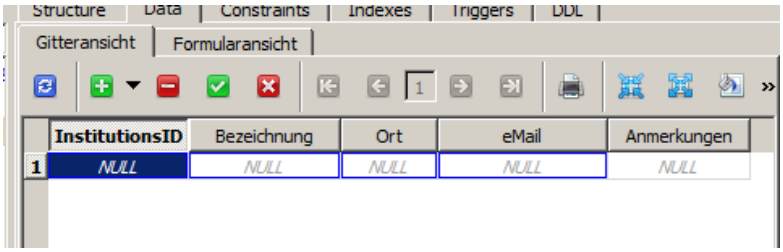
Jede Datenbank lebt von den Daten, die in den Tabellen vorhanden sind. Nur sie sind nachher für den Nutzer interessant. Die genutzte Struktur ist ihm in vielen Fällen völlig egal.
Da wir nun schon einiges über das Vorhandensein bestimmter Basisdaten vor der Nutzung in anderen Tabellen, begi´nnen wir hier bei der Dateneingabe mit der Institutions-Tabelle.

Den Zugriff auf die Tabellen-Inhalte erhält man über den reiter "Data".
Wir haben zwei Möglichkeiten Daten einzutragen. Die erste ist die sogenannte "Gitteransicht" ("Grid view"). Das ist praktisch eine ganz normale Tabellen-Ansicht.
Die zweite EingabeForm ist die "Formularansicht" ("Form view"). Diese besprechen wir gleich weiter hinten.
Über das grüne Plus-Symbol legen wir einen neuen Datensatz (eine neue Zeile) in unserer Tabelle an.

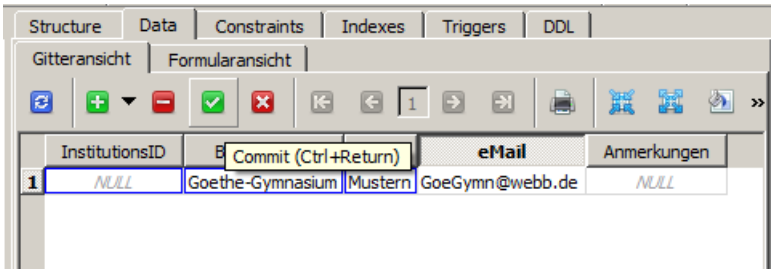


Eingabe in der "Gitteransicht"

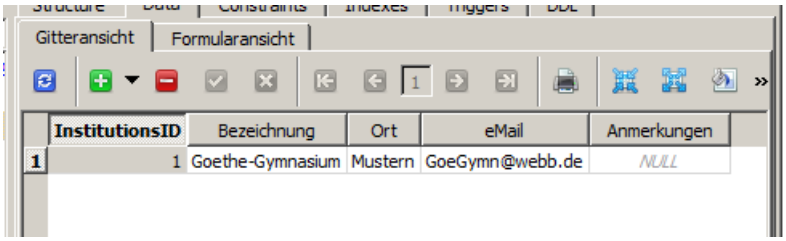
Die Eingabe-Zeile beginnt gleich mit der "InstitutionsID". Da ist man gleich verleitet, jetzt eine ID zu vergeben. Dies tun wir aber nicht, weil diese Spalte von uns als Primärschlüssel mit automatischer Hochzählung definiert wurde.

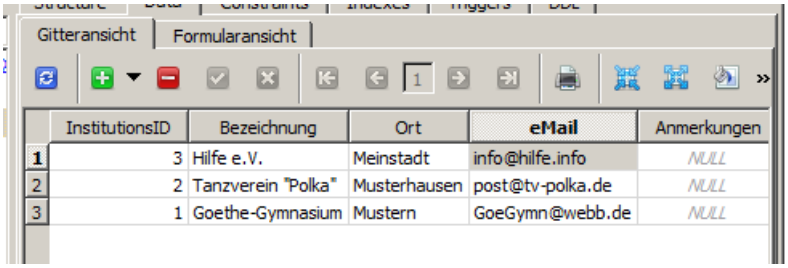


Diesen Sachverhalt muss man sich merken. Für einen Fremd-Nutzer ist das sicher ein Problem. Eine – einigermaßen gute - Verabredung ist es, die Primärschlüssel-Spalten immer mit ID enden zu lassen.
Wir beginnen also regulär mit der "Bezeichnung".



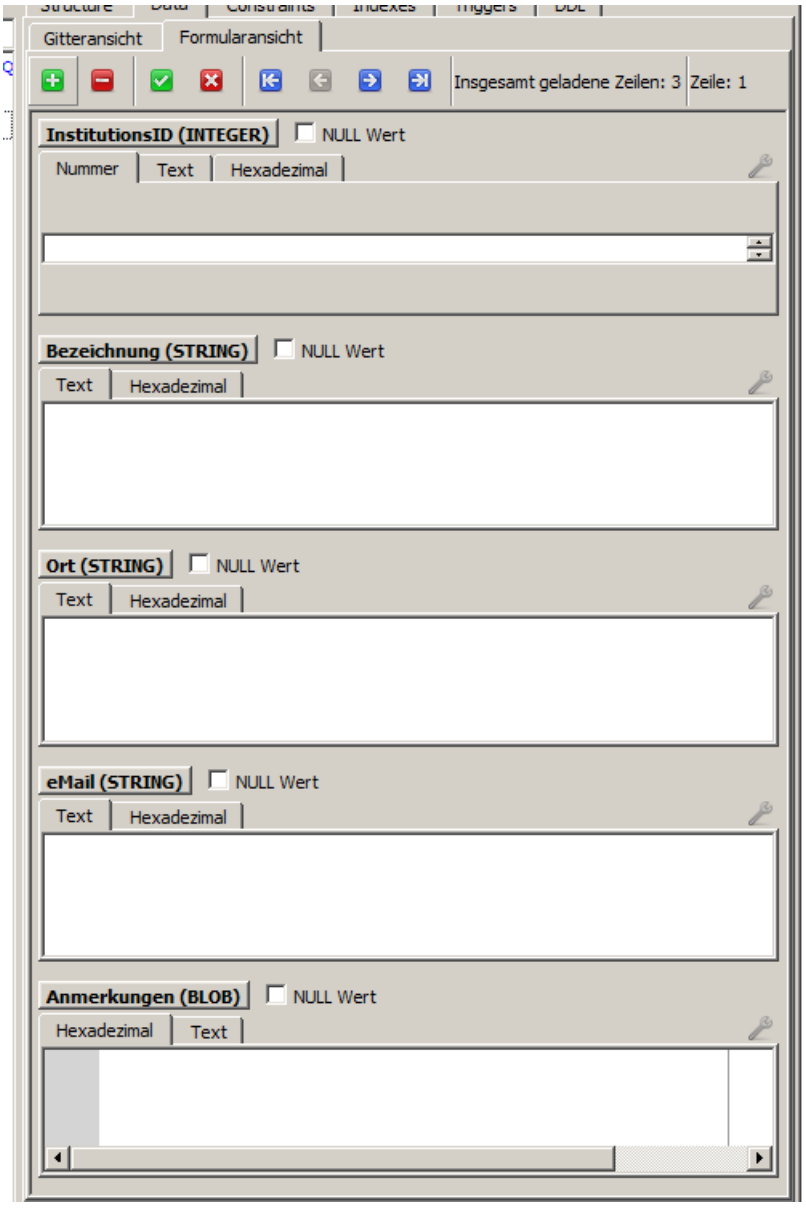
Wenn alle Daten (eines Datensatzes) eingetragen sind, bestätigen wir die Gültigkeit dieses Datensatzes durch einen Klick auf das grüne Häkchen. Wie im kleinen Hilfe-Text angezeigt geht auch die Tasten-Kombination [Strg] [Enter].
Der Datensatz bekommt dann automatisch die nächste Primärschlüssel-Nummer – hier eben die "1".
So kann man relativ schnell Tabellen mit Daten füllen.





Eingabe in der "Formularansicht"

Die Formularansicht verspricht eigentlich eine schönere Eingabe. In der schematischen Umsetzung innerhalb von SQLiteStudio ist sie aber eher unübersichtlich und nur für kleine Tabellen sinnvoll.
Ein Wechsel von Attribut zu Attribut mit [Tab] wird hier durch Zwischen-Stopp's bei den Formular-Element-Reitern aufwendiger.
Auch NULL-Werte muss man anlegen, was den Eingabe-Aufwand nochmals vergrößert.
Zu guter Letzt spricht auch noch die hier notwendige Angabe der ID für Verwirrung. Bei Nicht-Eingabe gibt es eine Fehler-Meldung im Status-Bereich. Der Nutzer kann aber kaum wissen, welche Nummer als nächstes benutzt werden kann.
Zu erwähnen ist aber der Datensatz-Navigator mit den blauen Sprung-Schaltflächen. Mit ihm kann man schön durch die einzelnen Datensätze durchnavigieren.



Wichtig ist auch hier, dass neue Datensätze oder die Änderungen in Felder zumindestens am Schluss über das grüne Häkchen (oder [Strg] [Enter]) bestätigt werden müssen!
Die Übernahme eines Datensatzes in die innere Datenbank (Database) erfolgt nur als Ganzes. Entweder wird die gesamte Transaktion ausgeführt oder eben nicht.

Aufgaben:

1. ***Geben Sie die angezeigten Daten manuell in die Tabelle ein!***
2. ***Probieren Sie auch die Formular-Ansicht aus, in dem Sie drei weitere "Institutionen" im obigen Stil (neue imaginäre Einrichtungen mit den vorgegebenen Orten!) ergänzen!***

Import von Daten

Vielfach stehen Daten schon in irgendeiner digitalen Form vor. Solche Daten möchte man nicht wirklich noch einmal abtippen. Die meisten Datenbanken bestehen ja nicht gerade aus einigen kleinen Tabellen mit wenigen Zeilen.
Die meisten modernen Programme bieten zum besonderen Speichern oder Exportieren bestimmte Datei-Formate an. Sehr verbreitet sind CSV-, XML- und JSON-Dateien. SQLiteStudio kann z.B. CSV-Dateien importieren. Diese lassen sich gut mit einem Editor (z.B. NotePad) oder Tabellenkalkulationen (wie microsoft EXCEL oder libreoffice / openoffice CALC) erzeugen.
Unsere Kontakte-Tabelle wollen wir nun per Import füllen. Voraussetzung ist eine CSV-Datei. Der grundsätzliche Aufbau ist leicht erklärt. Das CSV-Format beschreibt eine reine Text-Datei. Jede Zeile stellt einen Datensatz dar und muss mit einem Enter bzw. abgeschlossen werden. Die einzelnen Felder in einer Zeile sind durch ein definiertes Trennzeichen separiert. Klassischerweise benutzt man das Semikolon (";") oder einen Tabulator(-Sprung). Texte werden in Anführungszeichen gesetzt, die Zahlen und z.B. Datum- und Zeit-Angaben werden direkt notiert.
Eine Semikolon-separierte CSV-Datei mit den Daten für die "Kontakte"-Tabelle könnte so aussehen:

kontakte.csv

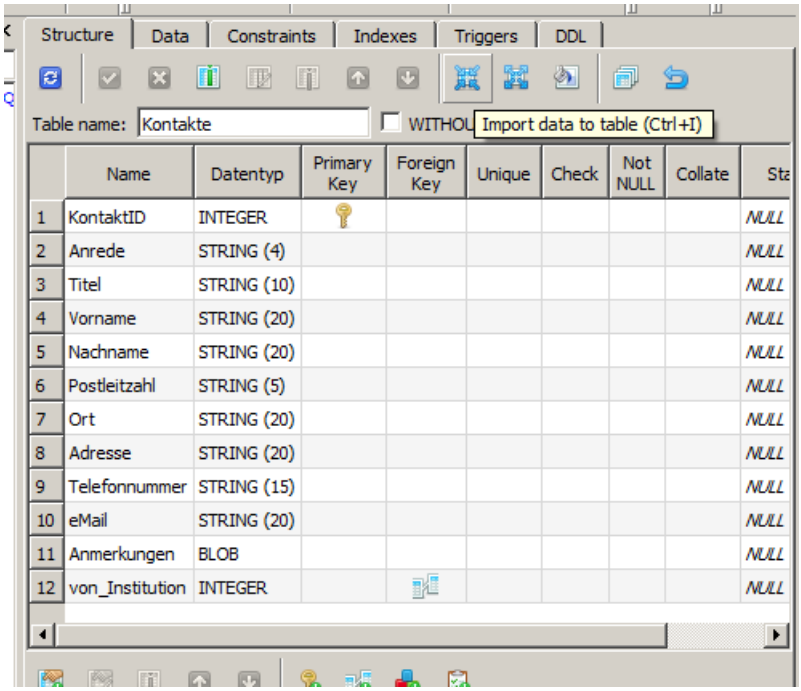
```
KontaktID,Anrede,Titel,Vorname,Nachname,Postleitzahl,Ort,Adresse,Telefonnummer,eMail,Anmerkung,von_Institution✓
1,Herr,Dr.,Klaus,Mustermann,12345,Musterhausen,Musterstr.13,0123456789, mustermann@webb.de,,1✓
2,Frau,,Monika,Mustermann,12345,Musterhausen,Musterstr. 13,0123456789, musterfrau@webb.de,,2✓
3,Herr,,Hans,Fehler,34343,Glückstadt an der Pech,Blumenweg 48,088088088, H.Fehler@webb.de,,1✓
4,Frau,,Maria,Muster,23456,Mustern,Am Musterweg 3,0234567890,m.muster@tee-online.de,,4✓
5,Herr,,Christian,Bauer,67890,Berg am Fluß,Waldallee 78,04567890901, Chr.Bauer@geemx.de,,5✓
6,Herr,,Lucas,Müller,54321,Bedorf,Feldweg 7,09998765,Mueller@tee-online.de,,1✓
7,Frau,,Tara,Zander,23456,Mustern,Hauptstr. 6e,0777711,Ta.Zan@webb.de,,6✓
8,Frau,Prof.,Hertha,Ziesow,88888,St. Muster,An der B777,0543861,Prof.H.Ziesow@tee-online.de,,4✓
12,Frau,,Maria,Berndt,67890,Berg am Fluß,Hans-Wald-Str. 4,0456783067,,,3✓
37,Herr,,Henriette,Krüger,76543,Meinstadt,Feldweg 7,06543210,post@h-krueger.de,,1✓
```

Zur Sicherheit habe ich die Positionen für das Enter-Zeichen mit einem besonderen blauen Zeichen versehen. Normalerweise ist das Zeichen nicht sichtbar. In einem Hex-Editor sieht man dann die Sequenz: **0D 0A**. Das steht für **CR** und **LF**.

Aufgaben:

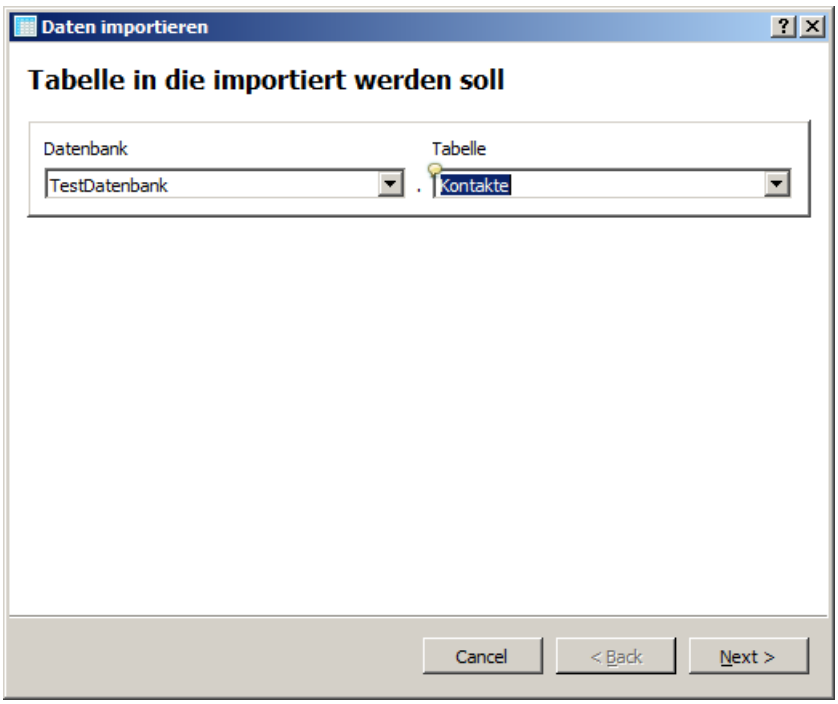
1. ***Informieren Sie sich über die Bedeutung / ursprüngliche Verwendung von CR und LF!***
2. ***Vergleichen Sie die Codierung von Texten mit ASCII- und Unicode-Zeichensatz!***
3. ***Welche Bedeutung hat die Kenntnis über die Codierung einer Text-Datei für einen Import in Datenbank-Systemen? Beachten Sie dabei auch, dass die meisten Datenbank-Systeme auf unterschiedlichen Betriebssystemen laufen (müssen)!***

Die Symbolleiste der Struktur-Ansicht enthält den Import-Button. Die vier blauen nach innen gerichteten Pfeile sind als Symbol wohl sehr passend.

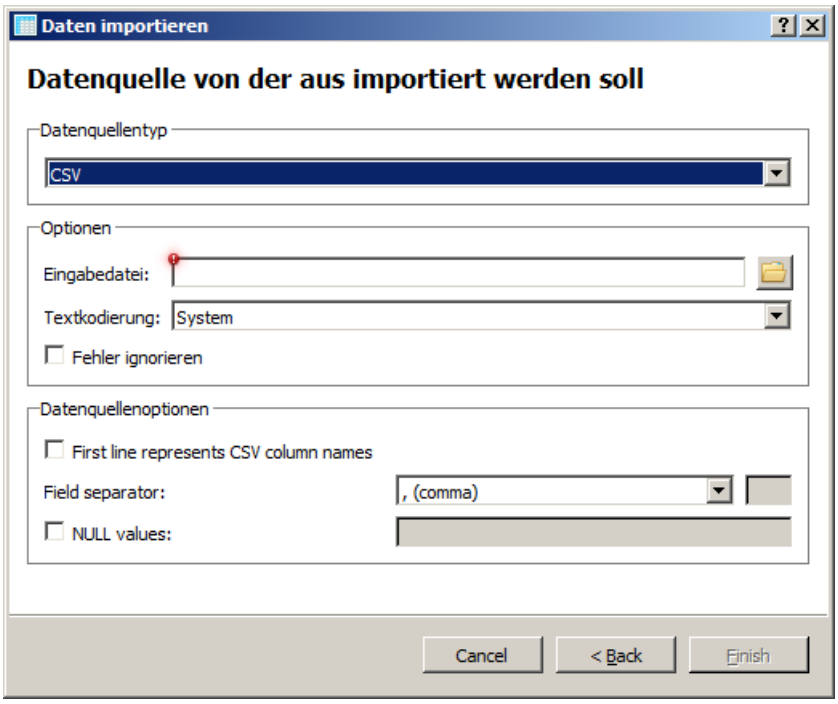


	Name	Datentyp	Primary Key	Foreign Key	Unique	Check	Not NULL	Collate	Sta
1	KontaktID	INTEGER							NULL
2	Anrede	STRING (4)							NULL
3	Titel	STRING (10)							NULL
4	Vorname	STRING (20)							NULL
5	Nachname	STRING (20)							NULL
6	Postleitzahl	STRING (5)							NULL
7	Ort	STRING (20)							NULL
8	Adresse	STRING (20)							NULL
9	Telefonnummer	STRING (15)							NULL
10	eMail	STRING (20)							NULL
11	Anmerkungen	BLOB							NULL
12	von_Institution	INTEGER							NULL

Es startet ein kleiner Assistent, der vor und zurück durchgearbeitet werden kann.
Viele Daten sind notwendig. Als erstes müssen wir z.B. die Ziel-Tabelle spezifizieren. Im Normalfall ist das die, aus deren Struktur-Ansicht heraus man den Import-Assistenten aufgerufen hat.
Das gelbe Lämpchen ist nur noch mal ein Hinweis auf die Kontrolle durch den Nutzer.
Mittels "Next"-Schaltfläche geht es zum nächsten Assistenten-Schritt.

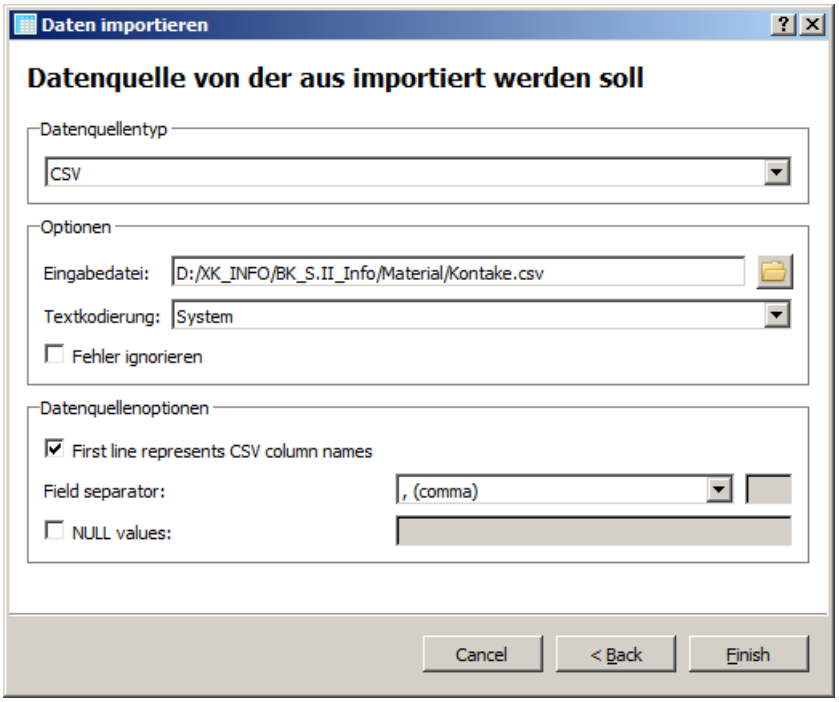


Hier stellen wir nun die Details zum Import ein.
Bei "Datenquellentyp" kann man zwischen einer reinen Daten-Datei im CSV-Format und einer SQL-Datei ("RegExp") auswählen.
Da unsere Daten – wie schon vorgestellt – im CSV-Format kommen, bleiben wir bei der Vorgabe.
Die "Eingabedatei" wird über das "Ordner"-Symbol rechts ausgewählt.
Die Textcodierung kann eine Herausforderung sein. Vor allem, wenn man Daten aus anderen betriebssystemen importiert, dann lohnt hier eine andere Wahl.

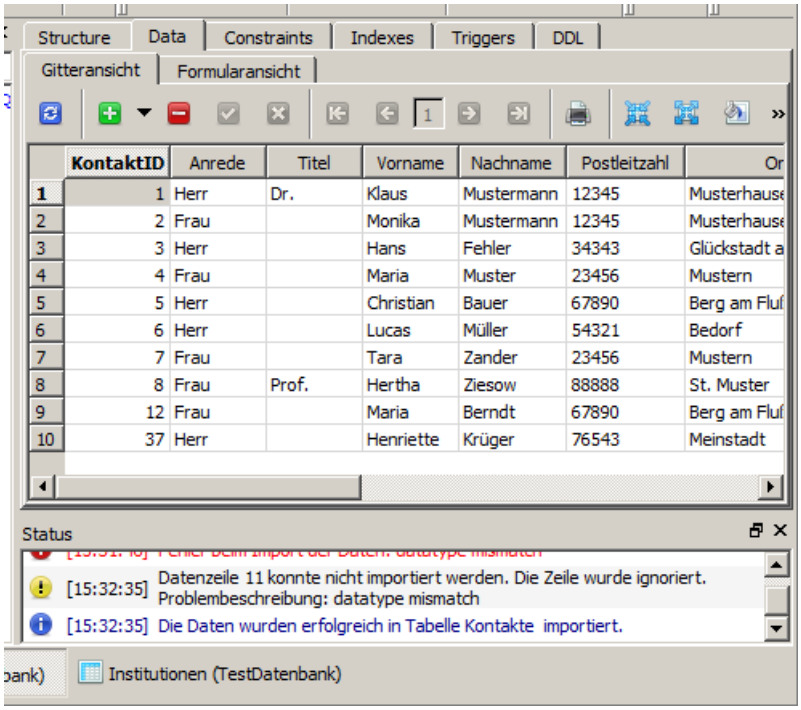


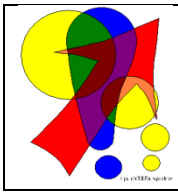
Mit "System" wird hier die Codierung genommen, die auf dem aktuellen Betriebssystem gültig ist. Da meine CSV auch im gleichen Betriebssystem erstellt wurde, bleibt die Vorgabe auch hier erhalten.

Bei den "Datenquelloptionen" muss man nun die CSV-Datei oder die Herstellungs-Optionen genauer kennen.
Aus dem Abdruck oben können wir erkennen, dass die erste Zeile Spalten-Namen enthält ("First line represnts CSV column names"). Der Feld-Seperator ist ein Komma.
Sollte andere Zeichen benutzt worden sein, dann wird hier die entsprechende Wahl getroffen.
Am Schluss bestätigen wir bei "Finisch" ("Beenden").
Sollte eine Fehler-Meldung auftauchen, dann empfiehlt sich eine Interpretation der Meldung und ev. die Überprüfung der Optionen.



Eine gute Möglichkeit ist die Option "Fehler ignorieren" zu setzen, um das System etwas Fehler-toleranter zu machen.
So sind z.B. Leer-Zeilen – wie sie häufig als letzte Zeile – in einer CSV-Datei vorkommen (abschließendes [Enter]) ein Problem für den Import-Filter von SQLiteStudio.
Wenn alles geklappt hat, dann erhalten wir im Status-Feld eine passende Meldung.
Die Daten stehen sofort für weitere Arbeiten zur Verfügung.





Bemerkungen zur führenden Null in Texten (z.B. PLZ und Telefonnummern):
Sowohl der Import – als auch die manuelle Eingabe oder Korrektur – ergibt keine führenden Nullen in Text-Feldern. Dies ist ein bekannter Bug, der sicher in einer der nächsten Versionen behoben wird.
Als Alternativ-Eingabe schlagen die Entwickler einen Notierung in einfache Hochkommata vor.

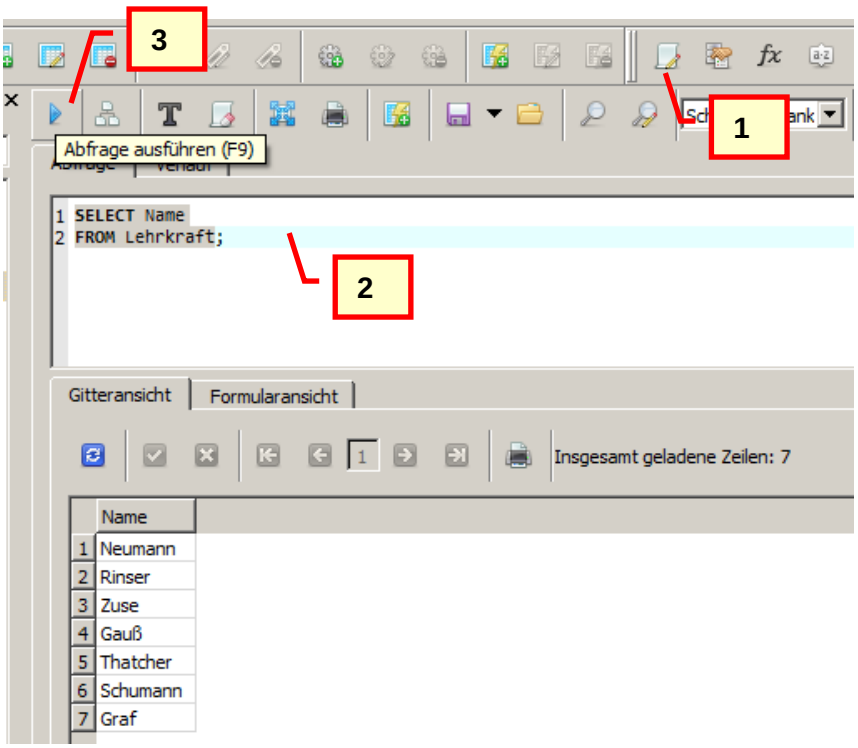
Aufgaben:

1. *Erstellen Sie sich in einem Editor eine CSV-Datei "Kontakte" mit den angegebenen Daten!*
2. *Importieren Sie diese dann in die zugehörige Tabelle im SQLiteStudio!*

3.1.7.1.3.4. Erstellen von Abfragen

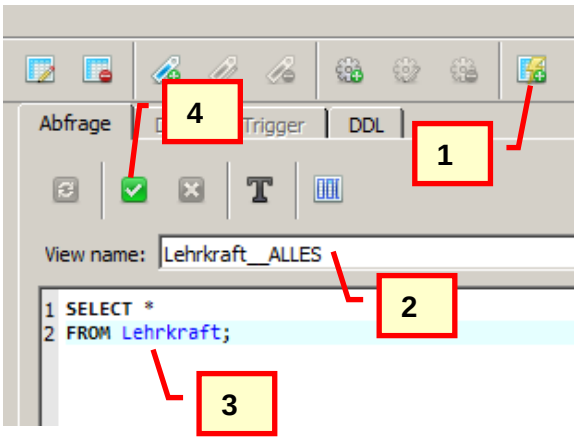
einfache Variante (temporäre Arbeits-Abfrage)

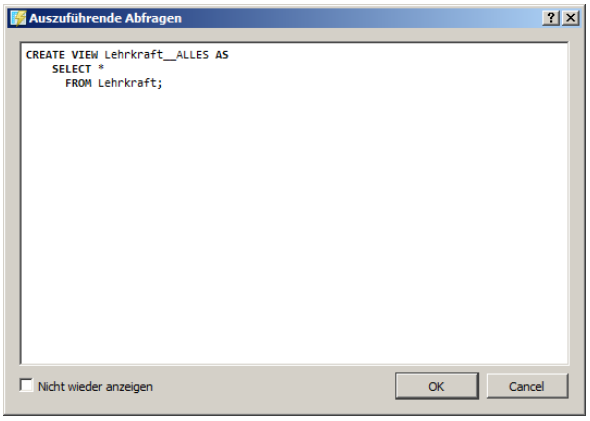
Dazu wählt man zuerst "Open SQL &editor" (**1**) aus. Unter dem Reiter "Abfrage" kann jetzt ein SQL-Statement eingetragen werden (**2**) und dann sofort beim blauen Dreieck (**3**; "Abfrage ausführen")
Das Ergebnis erscheint dann etwas tiefer in der Gitter- oder Formular-Ansicht. Letztere ist aber etwas gewöhnungsbedürftig.
Wer Tastatur-betont aberbeiten möchte benutzt zum Öffnen des SQL-Editors [Alt] + [E] und zum Ausführen [F9].



Variante mit Abspeichern / Sichern der Abfrage

Hierfür wählen wir zuerst einmal "Create a view" (**1**). Die Sicht bekommt nun einen sinnvollen Namen (**2**).
Für den View-Namen sollte man nur Buchstaben, Ziffern und den Unterstrich verwenden.
Dann gibt man das SQL-Statement ein (**3**) und führt beim grünen Häckchen aus (**4**).

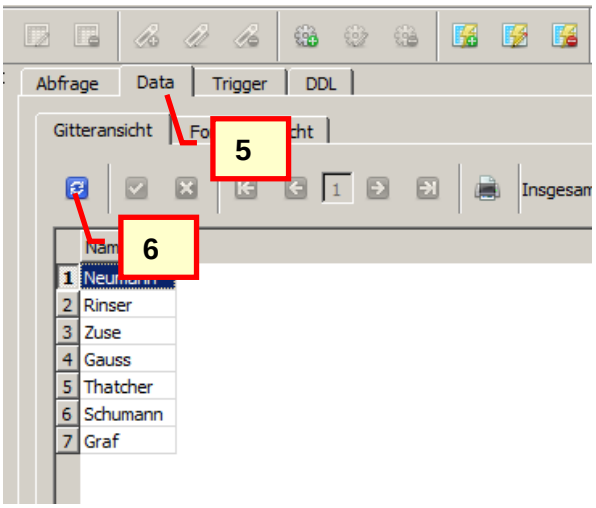




Zwischenzeitlich kann der vollständige SQL-Text – als "Auszuführende Abfragen" – für die Sicht erscheinen (Abb. links).

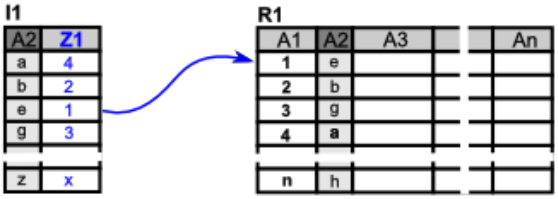
Unter "Data" sehen wir wieder das Ergebnis.

Soll eine definierte Abfrage neu angewendet werden, dann wechselt man zum Reiter "Data" (**5**) und macht hier eine "Aktualisierung der Tabellendaten" (**6**; blaues Symbol mit Kreis-Pfeilen).
Das Tastatur-Kürzel hierfür ist [F5].



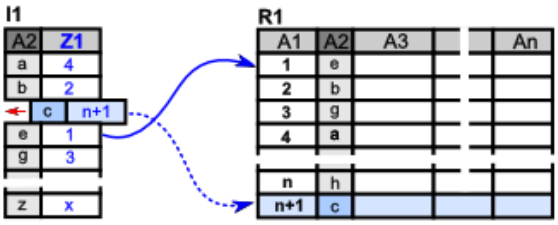
3.1.7.1.3.5. Erstellen von Indizes

Die Suche in großen Datenbanken kann schnell zum Problem werden. Man denke z.B. an eine Suche im Telefonbuch. Wir würden bei der Suche nach einem Eintrag sicher gleich viele Seiten überspringen, um dichter an den gesuchten Eintrag zu kommen. In einer Datenbank sind die Daten normalerweise ja nicht sortiert. Sie wuden dort eingetragen, wie sie kommen. Also müsste beim Suchen vorne anfangen und sich dann Datensatz für Datensatz durch die Tabelle hangeln, bis man den passenden Eintrag gefinden hat. In großen Datenbanken funktioniert dieses Vorgehen nur, wenn man beliebig Zeit hat, und wer hat das schon.
Eine erste Möglichkeit zur Lösung des Problems wäre es, die Tabelle nach jedem neuen Eintrag neu zu sortieren. Auch dies wird bei großen Tabellen kaum realisierbar sein. Außerdem ist nicht immer klar, nach welcher Spalte / nach welchem Attribut sortiert werden soll.
Abhilfe schaffen sogenannte Index-Strukturen. Sie stellen kleine Hilfs-Tabellen dar. Statt hier die Daten zu speichern, werden außer dem Ordnungs- oder Sortier-Wert noch ein Verweis auf den regulären Datensatz gespeichert. Die Verweise heißen in der Informatik Zeiger. In der Datenbank-Praxis ist das der Primärschlüssel des verwiesenen Datensatzes.



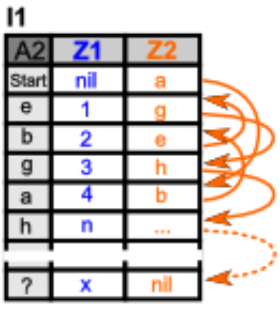
Prinzip eines tabellarischen Index

Vereinfacht kann man sich das so vorstellen: Für die Tabelle 1 (Relation1 = R1) wird z.B. ein tabellarischer Index 1 (I1) bezüglich des Attributes A2 erzeugt. Wir gehen vereinfacht davon aus, das sich die Attribut-Werte von A2 nicht wiederholen. In der Index-Tabelle ist jetzt jedem Attribut-Wert von A2 – also hier den Kleinbuchstaben von **a** bis **z** – ein Zeiger auf den zugehörigen Datensatz in Tabelle R1 zugewiesen.



Aktualisierung der Index-Tabelle nach Hinzufügen eines Datensatzes in der Verweis-Tabelle

Kommt nun ein neuer Datensatz in Tabelle R1 dazu, dann wird dieser im Normalfall angehängt.
Für den neuen Datensatz wird ein neuer Index-Eintrag erzeugt und – in einer einfachen Variante – an die richtige Sortier-Position eingefügt.
Noch effektiver wäre eine Zwei-Zeiger-Index-Tabelle. Der eine Zeiger verweist auf den Datensatz in R1 und der zweite Zeiger ist für die Gestaltung der Sortierreihenfolge in I1 verantwortlich.In so einer Tabelle bräuchten statt der aufwändigen Umsortierung ab Eintrag e in I1 nur noch zwei Zeiger umgestaltet ("verbogen") werden und schon wäre auch der Index fast sofort aktualisiert.
Diese kleinen Tabellen lassen sich sehr schnell durchsuchen, sortieren, umorganisieren und notfalls auch neu anlegen.

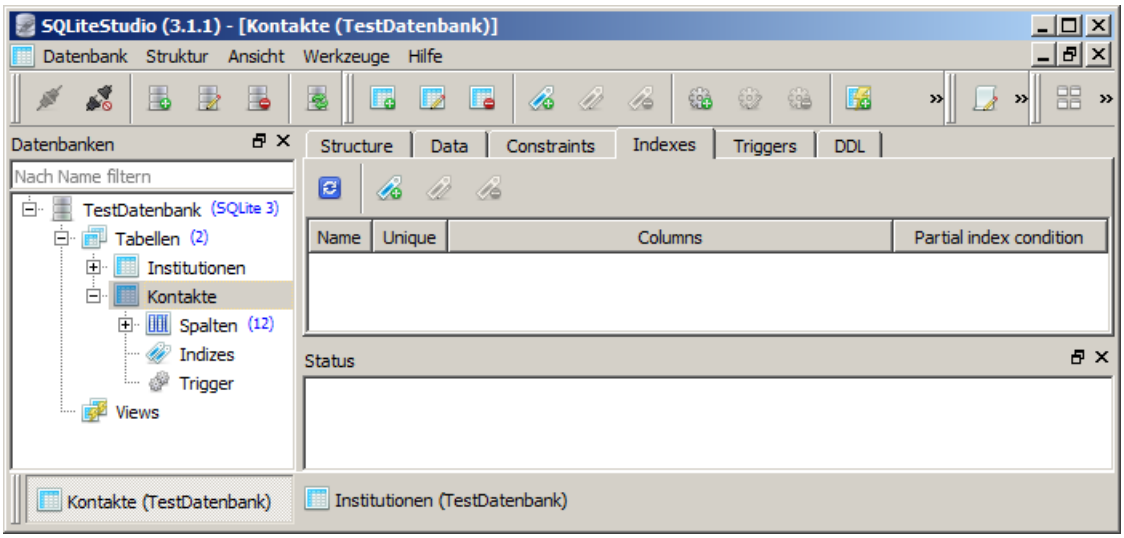


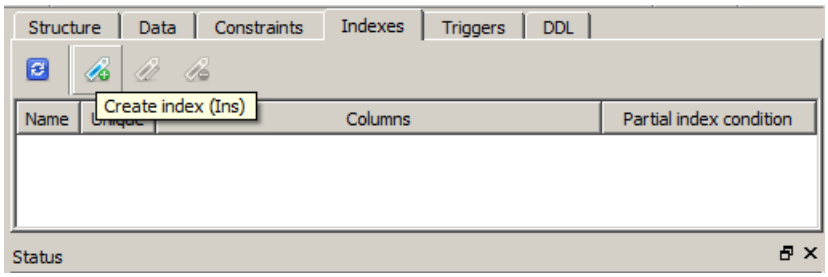
tabellarischer 2-Zeiger-Index

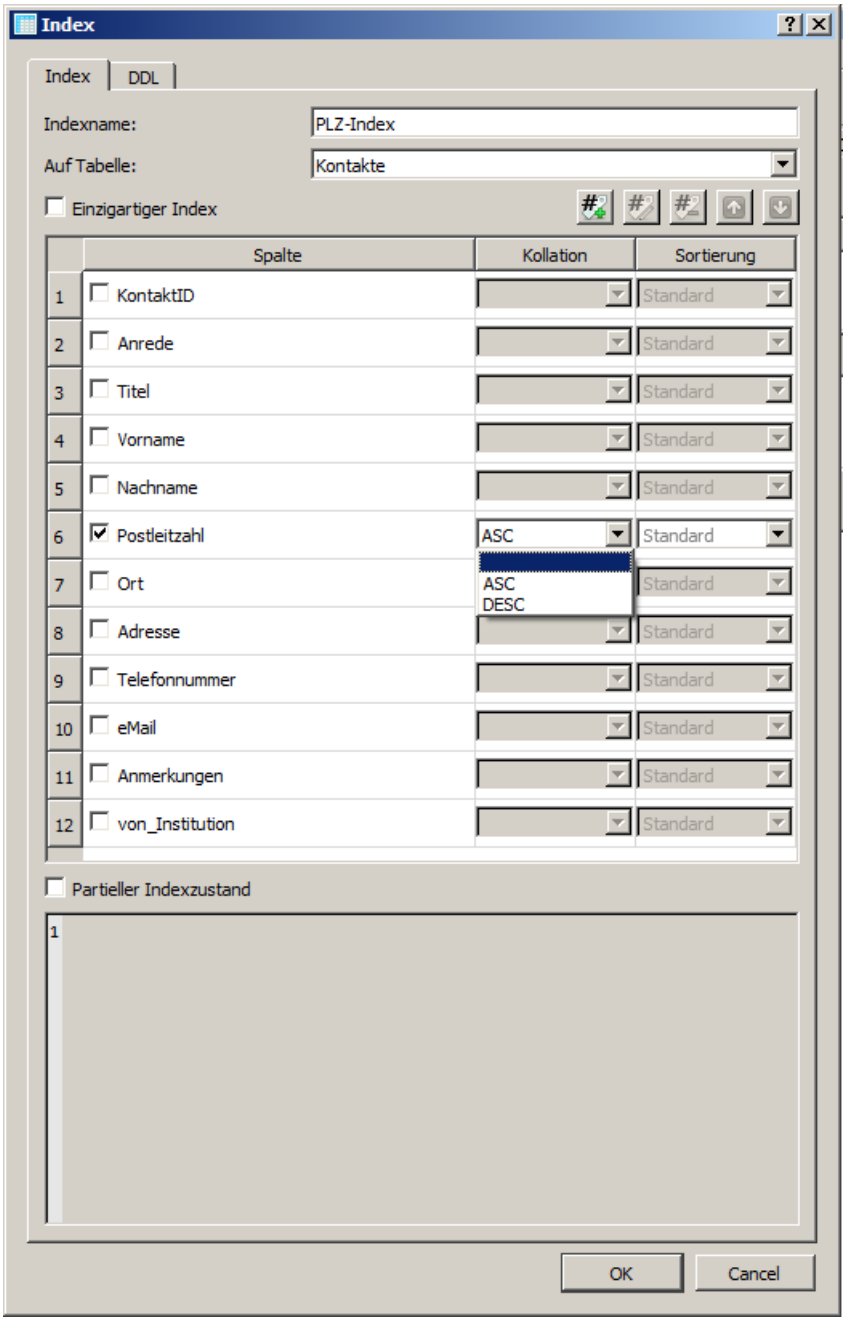
Aufgabe zwischendurch (für die gehobene Anspruchsebene):

1. *Welche Veränderungen müssen am 2-Zeiger-Index vorgenommen werden, wenn ein neuer Datensatz (wie oben: n+1) in R1 aufgenommen wird? Schätzen Sie den Zeitaufwand im Vergleich zum 1-Zeiger-Index oben ab!*

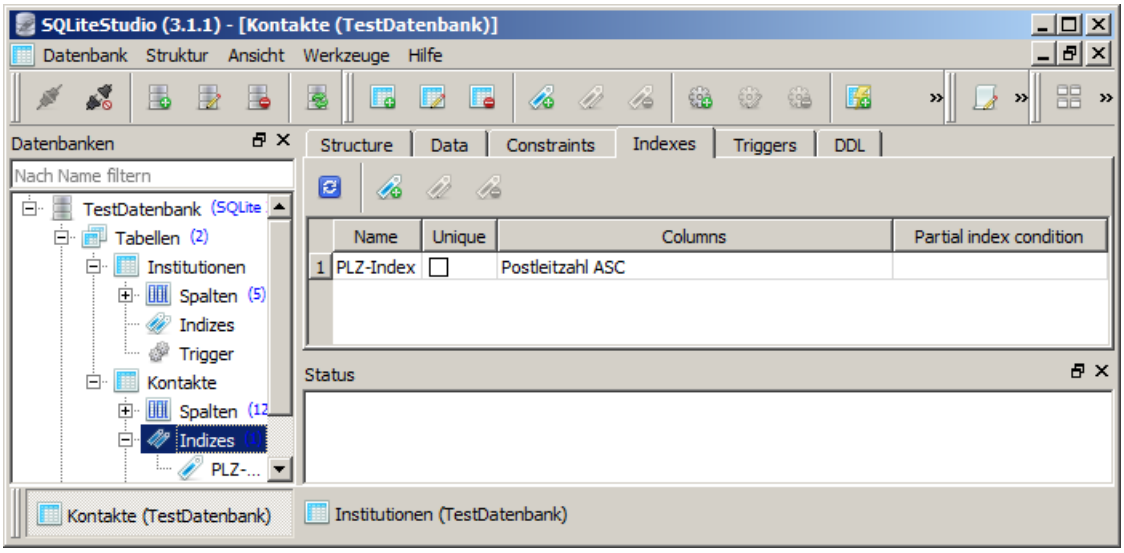
In modernen Systemen werden zudem statt Tabellen weitaus effektivere Datenstrukturen, wie z.B. Bäume (genau: B^+-Bäume), benutzt. Die Aktualisierung eines Index kann zudem parallel zur Eingabe / Veränderung anderer Einträge erfolgen. Dabei werden Stillstand- oder Nacht-Zeiten besonders effektiv ausgenutzt. Die zunehmende Hardware-Parallelisierung in der Verarbeitung von Daten verbessert die Arbeit mit Indizes nochmals. Je größer Daten-Tabellen sind und nach je mehr Attributen sie durchsucht / sortiert werden sollen, umso effektiver sind Indizes.
Vorteilhaft ist, dass man sich zu einer Tabelle beliebig viele Indizes anlegen kann. So kann ein Index die Sortierung der Namen enthalten, während ein anderer die Sortierung der Telefonnummern verwaltet. Es lassen sich auch Spalten-Kombinationen (Attribut-Kombinationen) indizieren, was z.B. für Shop- und Personen-Suche-Systeme interessant ist.
Leider gibt es keine verbindlichen Standard's für Indizes in SQL. Jedes Datenbank-System hat da seine eigenen Vorgehensweisen. Was hier zählt ist ja auch die Effektivität bei der praktischen Arbeit. Nur ganz elementare Befehle haben sich als "Praxis-Übereinkunft" durchgesetzt. Dazu gehören CREATE und DROP INDEX.
Indizes haben aber auch Nachteile. Werden Datensätze verändert, dann müssen eben auch immer die Indiezes angepasst werden. Das bedeutet zusätzlichen Rechenaufwand. Zudem sind Indezes oft nur im Speicher vorhanden. Dadurch wächst der Speicher-Bedarf solcher Systeme oft dramatisch an. Hier muss zwischen Aufwand und Nutzen abgewägt werden.






Index
Index
DDL
Indexname:
PLZ-Index
Auf Tabelle:
Kontakte
Einzigartiger Index
Spalte
Kollation
Sortierung
1 KontaktID
2 Anrede
3 Titel
4 Vorname
5 Nachname
6 Postleitzahl
7 Ort
8 Adresse
9 Telefonnummer
10 eMail
11 Anmerkungen
12 von_Institution
Standard
ASC
ASC
DESC
Partieller Indexzustand
1
OK
Cancel


Auszuführende Abfragen
CREATE INDEX [PLZ-Index] ON Kontakte (
Postleitzahl ASC
);
Nicht wieder anzeigen
OK
Cancel

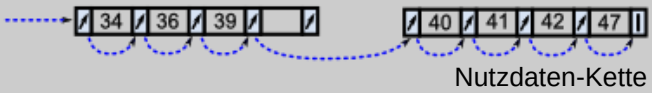


Aufgaben:

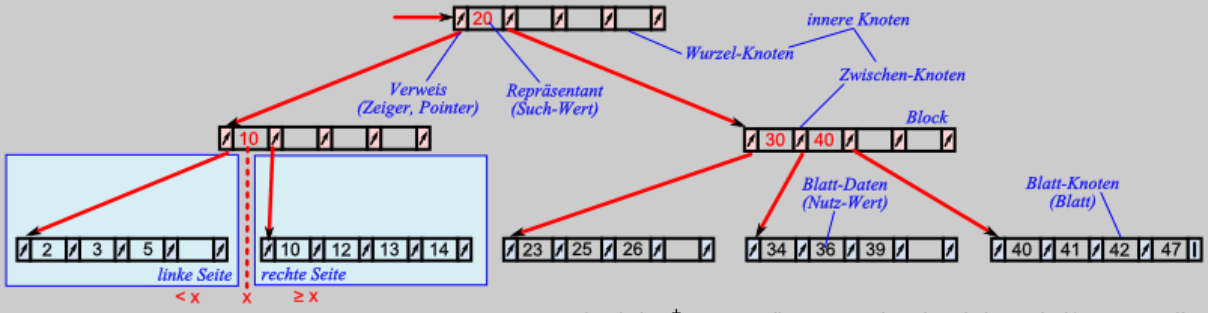
1. *Erstellen Sie den Index "PLZ-Index", wie oben vorgegeben!*
2. *Definieren Sie einen Index "Orts-Index" mit umgekehrter Sortier-Reihenfolge!*
3. *Erstellen Sie einen weiteren Index und erläutern Sie, warum dieser z.B. in einer großen Daten-Tabelle einen Sinn macht!*

Exkurs: B$^+$-Bäume

B$^+$-Bäume – früher auch B*-Bäume genannt – sind zuerst einmal typische Baum-Strukturen. Besonderheiten sind einmal die einheitliche Länge der Wege zu den Daten. Dabei wird auch auf eine möglichst geringe Verzweigungs-Tiefe Wert gelegt, um die Daten möglichst schnell zu erreichen. Es sind mehr als zwei Zweige möglich. Man spricht deshalb auch von einem Höhen-balancierten Mehrweg-Baum (Multiweg-Baum). Als zweite charakteristische Besonderheit gilt die ausschließliche Speicherung der Daten (Nutz-Werte) in den Blatt-Knoten (End-Blättern). Alle End-Blätter stellen zudem noch eine geordnete Liste dar. Blatt-Knoten haben somit auch keine Nachfolger mehr.
Diese interne Verketteung wird im folgenden Beispiel-B+-Baum aus Übersichtlichkeits-Gründen nicht mehr angezeigt.



Nutzdaten-Kette



Beispiel-B$^+$-Baum (k=2, n=2k=4) mit beschrifteten Teilen

Neben den Blatt-Knoten gibt es noch den Wurzel-Knoten und ev. weitere Zwischen-Knoten. Zusammen werden sie auch innere Knoten genannt. In einem kleinen B+-Baum kann der Wurzel-Knoten aber auch gleichzeitig das End-Blatt sein.
Wurzel- und Zwischen-Knoten enthalten Entscheidungs-Kriterien, die im Gegensatz zu den anderen Binär-Bäumen aber auch andere Werte, als solche aus der Daten-Liste enthalten können.
Ein Knoten kann mehrere Werte aufnehmen. Desweiteren sind im Knoten mehrere Zeiger (Referenzen, Pointer) auf andere Knoten-Elemente oder andere Knoten eingebaut. Man spricht bei solchen strukturierten Knoten auch gerne von einem Block.
B+-Bäume sind ausgesprochen dynamische Strukturen, die nach dem Lösen oder Hinzufügen von Werten immer mal wieder umgeordnet werden. Das oberste Ziel ist es dabei, die Verzeigungs-Länge immer klein und für alle Blatt-Elemente gleich lang zu gestalten.
Sie existieren nur während der Arbeitszeit eines Systems im Speicher.

allgemein gilt:

- Blatt-Knoten enthalten nur Nutz-Daten (Datenwerte), die in sich als sortierte Kette angeordnet sind
- innere Knoten (Wurzel- + Zwischen-Knoten) enthalten nur Such- / Schlüssel-Werte und zugehörige Verweise auf tieferliegende Zwischen-Knoten oder Blatt-Knoten
- der linke Verweis eines inneren Knotens zeigt auf innere Knoten oder Blätter, die kleinere Nutz-Werte repräsentieren
- der rechte Verweis eines inneren Knotens zeigt auf innere Knoten oder Blätter, die größere oder gleichgroße Nutz-Werte repräsentieren

für n=4 gilt
(n=2k)

- alle Knoten haben max. 4 Suchwerte (n) und max. 5 Verweise (n+1)
- die Wurzel hat mind. 1 Wert und 2 Verweise (Minimal-Index → ein Eintrag)
- Zwischen-Knoten haben mind. 1 Suchwert und 2 Verweise

- Blätter haben mind. 2 Datenwerte und 3 Verweise (Verkettung)

!!!Achtung, ab hier Datensammlung, z.T. auch über B-Bäume!!!

höhenbalancierter Mehrweg-Baum
sehr dynamische Struktur (nur im temporär im Arbeits-Speicher, solange mit der entsprechenden Tabelle gearbeitet wird)
durch wenige Höhenstufen ist eine minimaler Zugriffs-Aufwand garantiert
nur die Blatt-Knoten enthalten Nutzdaten und haben keinen Nachfolger
Zwischen-Knoten (innere Knoten und der Wurzel-Knoten) enthalten Hilfsdaten, um Zugriffe zu den Blatt-Knoten bzw. weiteren Zwischen-Knoten zu realisieren
Ordnung eines b-Baumes ist k
ein Baum der Ordnung 4 enthält maximal 3 Schlüsselwerte je Knoten
interne Knoten enthalten mindestens k-viele und (???) maximal 2k-viele Schlüsselwerte
Tiefe des Baumes (Baum-Höhe h) immer gleich für alle Zweige (Weg zu den Daten gleich lang)
Intervall-Grenzen werden durch die Schlüsselwerte bestimmt
es sind mehr als zwei Zweige möglich (Multiweg-Baum)

Teilbäume werden Seiten genannt

Wurzel ist entweder Blatt oder hat zwei Zweig-Knoten

Anzahl der Blätter an einem Knoten (auch Block genannt) wird durch k () begrenzt
Knoten hat neben dem Wert (Repräsentant) noch linken und rechten Zeiger
linker Zeiger eines inneren Knotens verweist auf ein Blatt mit Nutzdaten, in dem nur Nuzdaten mit kleinerem Schlüselwert enthalten sind
rechte Zeiger der inneren Knoten verweist auf ein Blatt mit einem größeren Schlüsselwert
rechte Zeiger der End-Knoten (Daten-Blätter) ergeben eine geordnete Liste der Elemente

Lesen:
Suche startet in der Wurzel; anhand von Vergleichen (kleiner des ersten Wurzel- oder Zwischen-Elementes / Zweigen oder zwischen zwei Elementen / Zweigen

Schreiben:
Suche, wie bei Lesen; im Fall eines Überlaufes wird Block gesplittet und der Index-Knoten um den Repäsentanten / Wert ergänzt

?Einfügen: (aus anderer Q)
wenn Platz vorhanden, dann an freier Stelle einfügen
wenn kein Platz (im Blatt-Knoten) vorhanden (→ Überlauf), dann wird Knoten geteilt
wenn kein Platz in Blatt und Index-Knoten, dann Aufteilen des Index-Knoten
dabei wird mittleres Index-Element in übergeordneten Index-Knoten eingetragen
falls dieser nicht existiert, dann wird neue Wurzel erzeugt

Löschen:
Suche, wie beim Lesen; im Fall eines Unterlaufes (Block mit zuwenig Elementen) werden benachbarte Knoten zusammengelegt

Anzahl der Zugriffe beim Lesen von der Höhe des Baumes bestimmt
Anzahl der Zugriffe beim Schreiben zwischen 1 und max. 2*h + 1
Anzahl der Zugriffe beim Löschen zwischen 1 und max. h + 1 bei durchlaufenden Unterlauf bis zur Wurzel

Speichereffizienz:
höherer Speicher-Bedarf, da Schlüsselwerte doppelt gespeichert werden
Füllgrad der Blätter zu 50% durch Struktur-Vorschriften gewährleistet

Links:
http://slady.net/java/bt/ (Java-Applet:B^+-Baum-Demo)

Aufgaben:

1. *Wieviele Nutz-Werte kann der im Exkurs als Beispiel angegebene Beispiel-B^+-Baum bei einer Höhe von 2 maximal aufnehmen? Erklären Sie!*
2. *?*
3. *?*

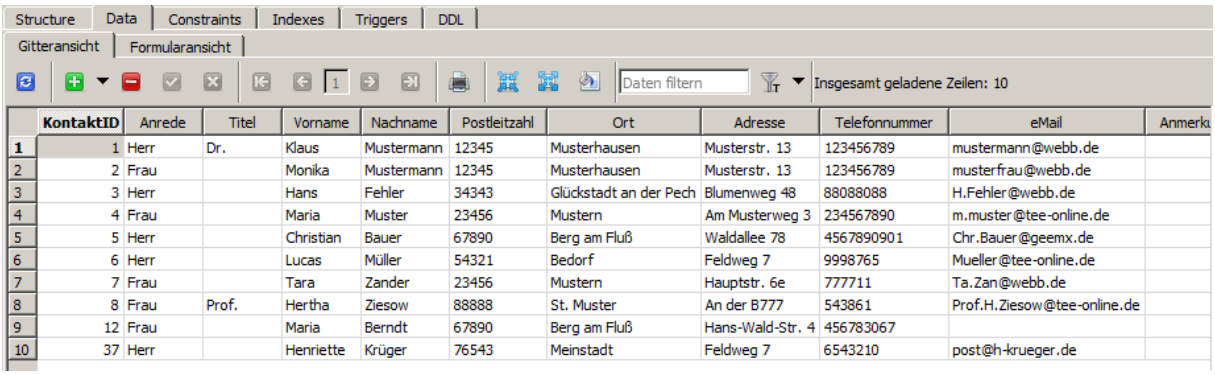
3.1.7.1.3.6. Erstellen von View's / Sichten / Abfragen

Programm-interne (Anzeige-)Möglichkeiten

betrifft nur die aktuelle Anzeige und nur den aktuellen Nutzer
Filter wirken nur innerhalb der Gitter-Anzeige
keine Weiternutzung in anderen Filtern etc. möglich
im Sinne von DBMS sind diese Ansichten keine echten Sichten (View's, Abfragen), weil sie wirklich nur der temporären Darstellung auf dem Bildschirm dienen
echte View's können als Defintion gespeichert werden und dann in der Datenbank für die weitere Nutzung aufgerufen werden, sie stellen praktisch Daten-Filter für die riesigen Daten-Bestände dar,
die Daten können aus mehreren Quell-Tabellen stammen, werden unter bestimmten Regeln (Tabellen-Algebra) (neu) zusammengestellt und dann wiederum als (Ergebnis-)Tabelle bereitgestellt
ev. werden noch Kenngrößen, wie Mittelwerte, Summen, Anzahlen mitermittelt.
sie sind temporär und nur für den aktuellen Aufruf eines View's gültig
für Nutzer ist nicht erkennbar, ob es sich um ursprüngliche oder um eine zusammengestellte Tabelle handelt
können als (eigene) Tabelle abgespeichert werden
dann werden die Kenngrößen meist beim nächsten Aufruf der gespeicherten Tabelle neu ermittelt

bei den Programm-internen Ansichten spielt sich alles in der ursprünglichen Tabelle ab, wir sehen einfach nur einen Teil davon bzw. die Tabelle andes dargestellt

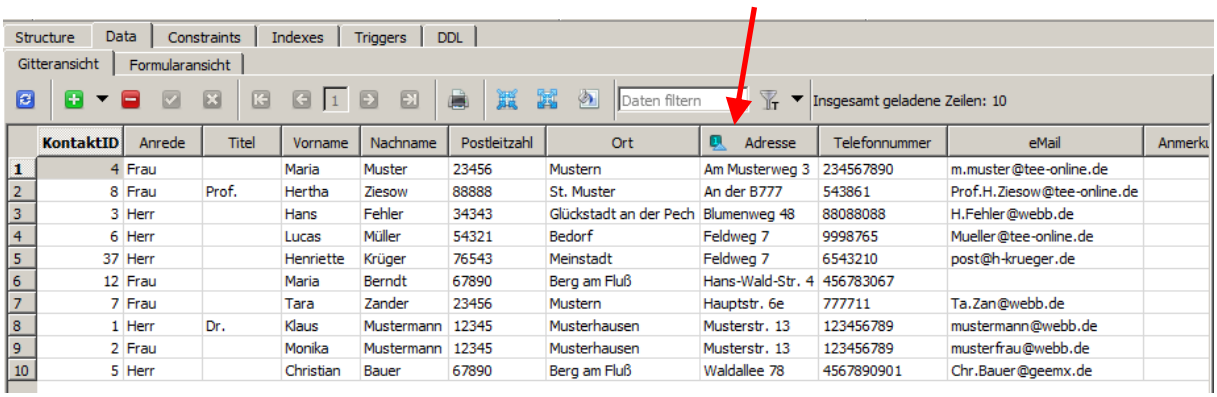
Ausgangs- / Beispiel-Tabelle für alle nachfolgenden Sortierungen usw.



	KontaktID	Anrede	Titel	Vorname	Nachname	Postleitzahl	Ort	Adresse	Telefonnummer	eMail	Anmerku
1	1	Herr	Dr.	Klaus	Mustermann	12345	Musterhausen	Musterstr. 13	123456789	mustermann@webb.de	
2	2	Frau		Monika	Mustermann	12345	Musterhausen	Musterstr. 13	123456789	musterfrau@webb.de	
3	3	Herr		Hans	Fehler	34343	Glückstadt an der Pech	Blumenweg 48	88088088	H.Fehler@webb.de	
4	4	Frau		Maria	Muster	23456	Mustern	Am Musterweg 3	234567890	m.muster@tee-online.de	
5	5	Herr		Christian	Bauer	67890	Berg am Fluß	Waldallee 78	4567890901	Chr.Bauer@geemx.de	
6	6	Herr		Lucas	Müller	54321	Bedorf	Feldweg 7	9998765	Mueller@tee-online.de	
7	7	Frau		Tara	Zander	23456	Mustern	Hauptstr. 6e	777711	Ta.Zan@webb.de	
8	8	Frau	Prof.	Hertha	Ziesow	88888	St. Muster	An der B777	543861	Prof.H.Ziesow@tee-online.de	
9	12	Frau		Maria	Berndt	67890	Berg am Fluß	Hans-Wald-Str. 4	456783067		
10	37	Herr		Henriette	Krüger	76543	Meinstadt	Feldweg 7	6543210	post@h-krueger.de	

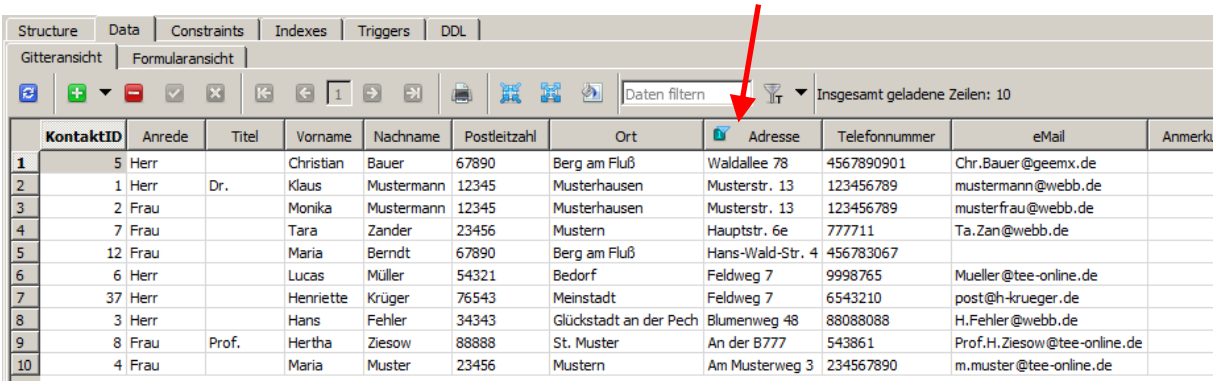
Sortierung(en)

dazu reicht ein Klick auf den Attribut-Namen (Spalten-Name)
zuerst aufsteigende Sortierung
ersichtlich am nach unten breiter werdenden grünen Dreieck in der Kopfzeile



	KontaktID	Anrede	Titel	Vorname	Nachname	Postleitzahl	Ort	Adresse	Telefonnummer	eMail	Anmerku
1	4	Frau		Maria	Muster	23456	Mustern	Am Musterweg 3	234567890	m.muster@tee-online.de	
2	8	Frau	Prof.	Hertha	Ziesow	88888	St. Muster	An der B777	543861	Prof.H.Ziesow@tee-online.de	
3	3	Herr		Hans	Fehler	34343	Glückstadt an der Pech	Blumenweg 48	88088088	H.Fehler@webb.de	
4	6	Herr		Lucas	Müller	54321	Bedorf	Feldweg 7	9998765	Mueller@tee-online.de	
5	37	Herr		Henriette	Krüger	76543	Meinstadt	Feldweg 7	6543210	post@h-krueger.de	
6	12	Frau		Maria	Berndt	67890	Berg am Fluß	Hans-Wald-Str. 4	456783067		
7	7	Frau		Tara	Zander	23456	Mustern	Hauptstr. 6e	777711	Ta.Zan@webb.de	
8	1	Herr	Dr.	Klaus	Mustermann	12345	Musterhausen	Musterstr. 13	123456789	mustermann@webb.de	
9	2	Frau		Monika	Mustermann	12345	Musterhausen	Musterstr. 13	123456789	musterfrau@webb.de	
10	5	Herr		Christian	Bauer	67890	Berg am Fluß	Waldallee 78	4567890901	Chr.Bauer@geemx.de	

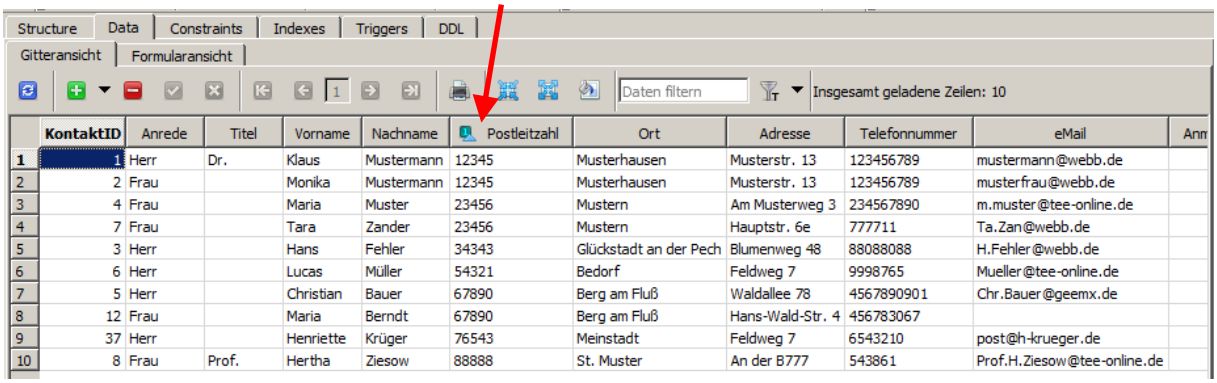
Mit einem weiterem Klick auf den Spalten-Kopf wird Sortier-Reihenfolge umgedreht.



	KontaktID	Anrede	Titel	Vorname	Nachname	Postleitzahl	Ort	Adresse	Telefonnummer	eMail	Anmerku
1	5	Herr		Christian	Bauer	67890	Berg am Fluß	Waldallee 78	4567890901	Chr.Bauer@geemx.de	
2	1	Herr	Dr.	Klaus	Mustermann	12345	Musterhausen	Musterstr. 13	123456789	mustermann@webb.de	
3	2	Frau		Monika	Mustermann	12345	Musterhausen	Musterstr. 13	123456789	musterfrau@webb.de	
4	7	Frau		Tara	Zander	23456	Mustern	Hauptstr. 6e	777711	Ta.Zan@webb.de	
5	12	Frau		Maria	Berndt	67890	Berg am Fluß	Hans-Wald-Str. 4	456783067		
6	6	Herr		Lucas	Müller	54321	Bedorf	Feldweg 7	9998765	Mueller@tee-online.de	
7	37	Herr		Henriette	Krüger	76543	Meinstadt	Feldweg 7	6543210	post@h-krueger.de	
8	3	Herr		Hans	Fehler	34343	Glückstadt an der Pech	Blumenweg 48	88088088	H.Fehler@webb.de	
9	8	Frau	Prof.	Hertha	Ziesow	88888	St. Muster	An der B777	543861	Prof.H.Ziesow@tee-online.de	
10	4	Frau		Maria	Muster	23456	Mustern	Am Musterweg 3	234567890	m.muster@tee-online.de	

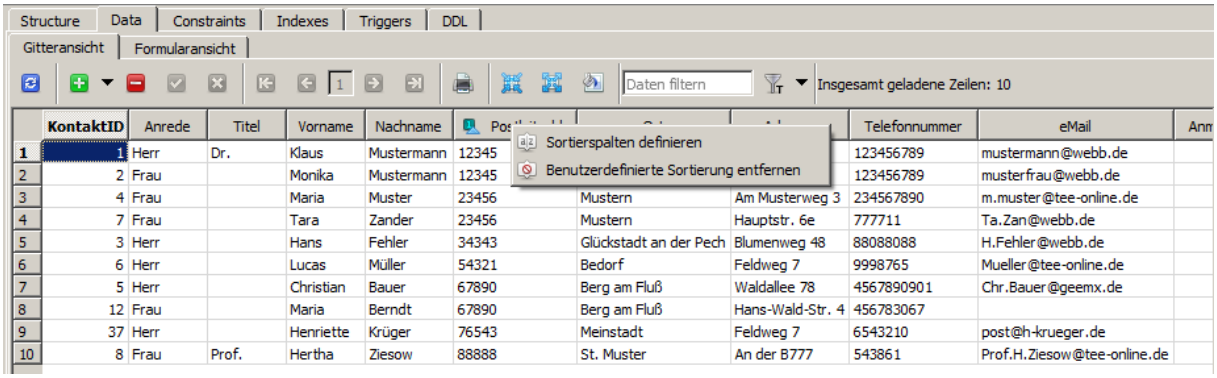
mehrfache / gestaffelte Sortierung

z.B. für eine "Adress-Tabelle" zuerst Sortierung nach Postleitzahlen



	KontaktID	Anrede	Titel	Vorname	Nachname	Postleitzahl	Ort	Adresse	Telefonnummer	eMail	Anm
1	1	Herr	Dr.	Klaus	Mustermann	12345	Musterhausen	Musterstr. 13	123456789	mustermann@webb.de	
2	2	Frau		Monika	Mustermann	12345	Musterhausen	Musterstr. 13	123456789	musterfrau@webb.de	
3	4	Frau		Maria	Muster	23456	Mustern	Am Musterweg 3	234567890	m.muster@tee-online.de	
4	7	Frau		Tara	Zander	23456	Mustern	Hauptstr. 6e	777711	Ta.Zan@webb.de	
5	3	Herr		Hans	Fehler	34343	Glückstadt an der Pech	Blumenweg 48	88088088	H.Fehler@webb.de	
6	6	Herr		Lucas	Müller	54321	Bedorf	Feldweg 7	9998765	Mueller@tee-online.de	
7	5	Herr		Christian	Bauer	67890	Berg am Fluß	Waldallee 78	4567890901	Chr.Bauer@geemx.de	
8	12	Frau		Maria	Berndt	67890	Berg am Fluß	Hans-Wald-Str. 4	456783067		
9	37	Herr		Henriette	Krüger	76543	Meinstadt	Feldweg 7	6543210	post@h-krueger.de	
10	8	Frau	Prof.	Hertha	Ziesow	88888	St. Muster	An der B777	543861	Prof.H.Ziesow@tee-online.de	

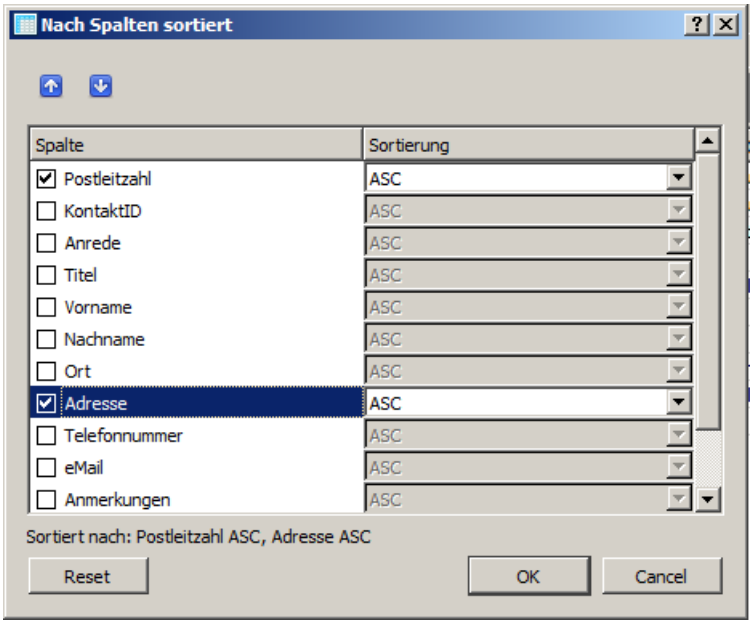
und dann noch nach den Straßennamen (Adressen)
über rechte Maustaste auf die Kopfzeile

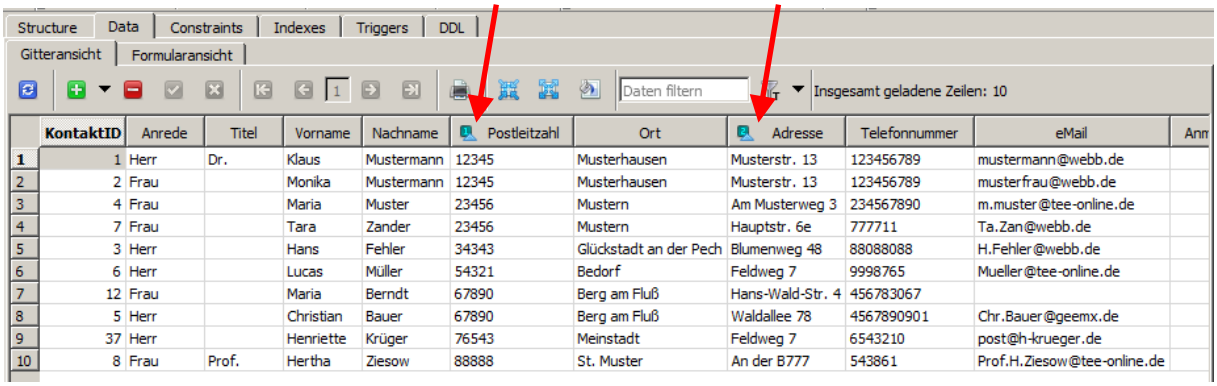


	KontaktID	Anrede	Titel	Vorname	Nachname	Pos			Telefonnummer	eMail	Anm
1	1	Herr	Dr.	Klaus	Mustermann	12345			123456789	mustermann@webb.de	
2	2	Frau		Monika	Mustermann	12345			123456789	musterfrau@webb.de	
3	4	Frau		Maria	Muster	23456	Mustern	Am Musterweg 3	234567890	m.muster@tee-online.de	
4	7	Frau		Tara	Zander	23456	Mustern	Hauptstr. 6e	777711	Ta.Zan@webb.de	
5	3	Herr		Hans	Fehler	34343	Glückstadt an der Pech	Blumenweg 48	88088088	H.Fehler@webb.de	
6	6	Herr		Lucas	Müller	54321	Bedorf	Feldweg 7	9998765	Mueller@tee-online.de	
7	5	Herr		Christian	Bauer	67890	Berg am Fluß	Waldallee 78	4567890901	Chr.Bauer@geemx.de	
8	12	Frau		Maria	Berndt	67890	Berg am Fluß	Hans-Wald-Str. 4	456783067		
9	37	Herr		Henriette	Krüger	76543	Meinstadt	Feldweg 7	6543210	post@h-krueger.de	
10	8	Frau	Prof.	Hertha	Ziesow	88888	St. Muster	An der B777	543861	Prof.H.Ziesow@tee-online.de	

bei "Sortierspalten definieren" ruft man einen einfachen Dialog auf, in dem sowohl die Sortier-Richtung, als auch die Spalten-Reihenfolge (Hierarchie) festgelegt werden kann
Dazu wählt man die Spalte, deren Rangfolge verändert werden soll aus, und bewegt diese dann mittels der blauen Pfeile nach oben oder unten.

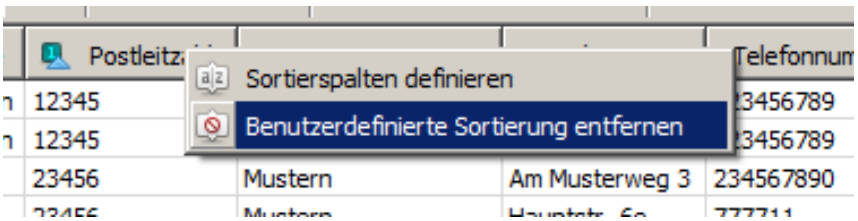
Sollte man mehrere Sortierungen irgendwie total verquer haben, dann kann man mit einem "Reset" wieder die Ausgangs-Situation herstellen und die Sortierungen neu organisieren.





	KontaktID	Anrede	Titel	Vorname	Nachname	Postleitzahl	Ort	Adresse	Telefonnummer	eMail
1	1	Herr	Dr.	Klaus	Mustermann	12345	Musterhausen	Musterstr. 13	123456789	mustermann@webb.de
2	2	Frau		Monika	Mustermann	12345	Musterhausen	Musterstr. 13	123456789	musterfrau@webb.de
3	4	Frau		Maria	Muster	23456	Mustern	Am Musterweg 3	234567890	m.muster@tee-online.de
4	7	Frau		Tara	Zander	23456	Mustern	Hauptstr. 6e	777711	Ta.Zan@webb.de
5	3	Herr		Hans	Fehler	34343	Glückstadt an der Pech	Blumenweg 48	88088088	H.Fehler@webb.de
6	6	Herr		Lucas	Müller	54321	Bedorf	Feldweg 7	9998765	Mueller@tee-online.de
7	12	Frau		Maria	Berndt	67890	Berg am Fluß	Hans-Wald-Str. 4	456783067	
8	5	Herr		Christian	Bauer	67890	Berg am Fluß	Waldallee 78	4567890901	Chr.Bauer@geemx.de
9	37	Herr		Henriette	Krüger	76543	Meinstadt	Feldweg 7	6543210	post@h-krueger.de
10	8	Frau	Prof.	Hertha	Ziesow	88888	St. Muster	An der B777	543861	Prof.H.Ziesow@tee-online.de

Die Sortierreihenfolge (Hierarchie) wird an den Nummern in den kleinen Sortier-Kennzeichnungs-Symbolen deutlich.
Werden die Sortierungen nicht mehr gebraucht, dann kann man diese wieder über einen Rechts-Klick in die Kopfzeile die "Benutzerdefinierte Sortierung entfernen".

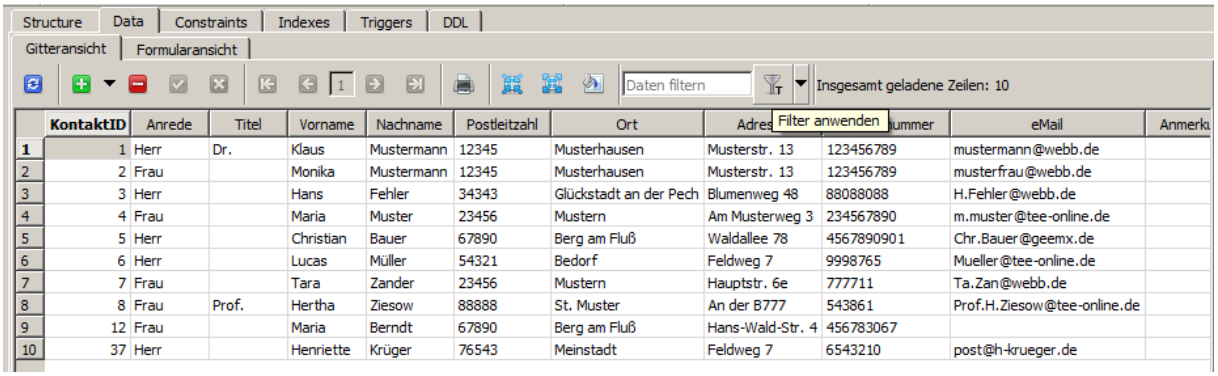


Aufgaben:

1. *Sortieren Sie Tabelle "Institutionen" alphabetisch nach den Orten!*
2. *Wie lautet der DLL-Ausdruck für die Sortierung der Tabelle "Kontakte" zuerst nach der "Anrede" (unhöfliche Sortierung) und dann nach den Telefonnummern in umgekehrter Reihenfolge?*

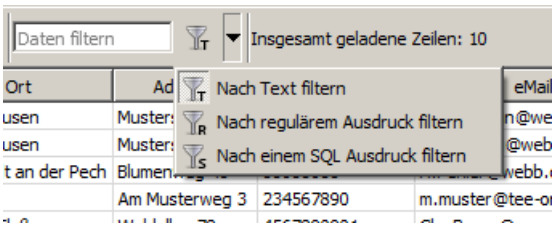
Filter-Methoden in Tabellen

Filterung dient der Auswahl der anzuzeigenden Datensätze (vor allem eben bei großen Tabellen)
gut für schnelle Datensuche; Heraussuche passender Datensätze

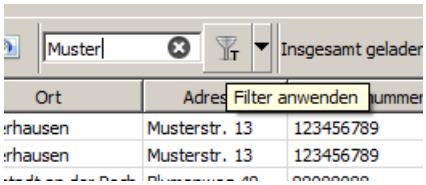


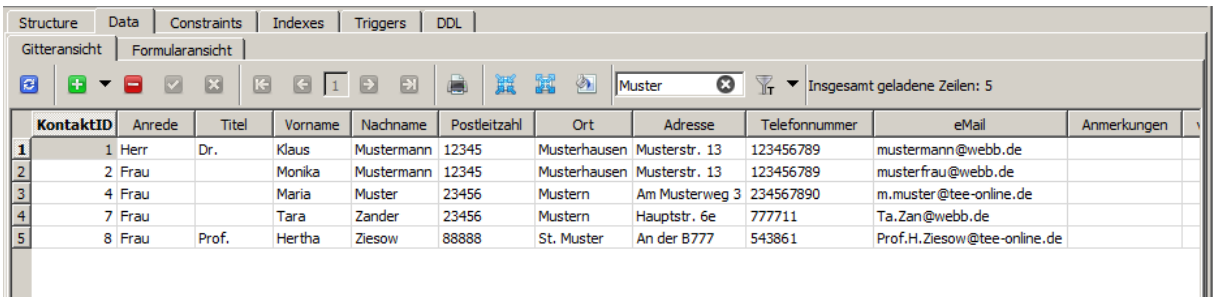
	KontaktID	Anrede	Titel	Vorname	Nachname	Postleitzahl	Ort	Adres	ummer	eMail	Anmerku
1	1	Herr	Dr.	Klaus	Mustermann	12345	Musterhausen	Musterstr. 13	123456789	mustermann@webb.de	
2	2	Frau		Monika	Mustermann	12345	Musterhausen	Musterstr. 13	123456789	musterfrau@webb.de	
3	3	Herr		Hans	Fehler	34343	Glückstadt an der Pech	Blumenweg 48	88088088	H.Fehler@webb.de	
4	4	Frau		Maria	Muster	23456	Mustern	Am Musterweg 3	234567890	m.muster@tee-online.de	
5	5	Herr		Christian	Bauer	67890	Berg am Fluß	Waldallee 78	4567890901	Chr.Bauer@geemx.de	
6	6	Herr		Lucas	Müller	54321	Bedorf	Feldweg 7	9998765	Mueller@tee-online.de	
7	7	Frau		Tara	Zander	23456	Mustern	Hauptstr. 6e	777711	Ta.Zan@webb.de	
8	8	Frau	Prof.	Hertha	Ziesow	88888	St. Muster	An der B777	543861	Prof.H.Ziesow@tee-online.de	
9	12	Frau		Maria	Berndt	67890	Berg am Fluß	Hans-Wald-Str. 4	456783067		
10	37	Herr		Henriette	Krüger	76543	Meinstadt	Feldweg 7	6543210	post@h-krueger.de	

Nach Text filtern



Am Einfachten und Allgemeinsten ist die Text-Filterung. Dazu wählt man am Besten zuerst einmal die Filter-Methode aus. Beim Klick auf den Auswähler neben dem Trichter-Symbol erscheinen die drei Möglichkeiten.
Zur Textsuche ("Nach Text filtern") steht ein kleines T neben dem Trichter-Symbol.
Nun gibt man den Suchbegriff in das "Daten filtern"-Feld ein und klickt einmal auf das Trichter-Symbol.
Der Filter wird sofort ausgeführt und in den meisten fällen verkürzt sich die Tabelle.
Wie schon gesagt bezieht sich dies nur auf die aktuelle Anzeige. An der Daten-Tabelle werden keine Veränderungen vorgenommen.



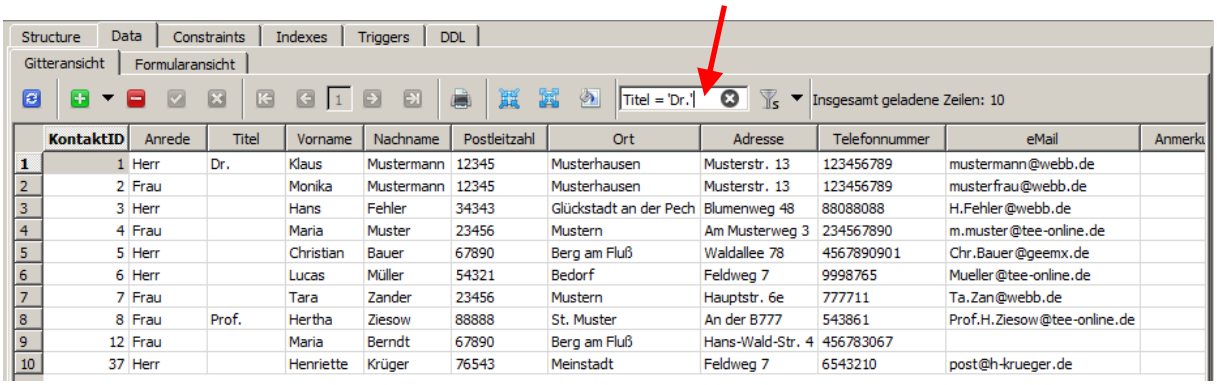


	KontaktID	Anrede	Titel	Vorname	Nachname	Postleitzahl	Ort	Adresse	Telefonnummer	eMail	Anmerkungen
1	1	Herr	Dr.	Klaus	Mustermann	12345	Musterhausen	Musterstr. 13	123456789	mustermann@webb.de	
2	2	Frau		Monika	Mustermann	12345	Musterhausen	Musterstr. 13	123456789	musterfrau@webb.de	
3	4	Frau		Maria	Muster	23456	Mustern	Am Musterweg 3	234567890	m.muster@tee-online.de	
4	7	Frau		Tara	Zander	23456	Mustern	Hauptstr. 6e	777711	Ta.Zan@webb.de	
5	8	Frau	Prof.	Hertha	Ziesow	88888	St. Muster	An der B777	543861	Prof.H.Ziesow@tee-online.de	

Das merkt man auch spätestens, wenn man die Filterung mittels des Lösch-Symbols im Suchfeld wieder ausschaltet. Alle Datensätze sind unbeschadet wieder da.

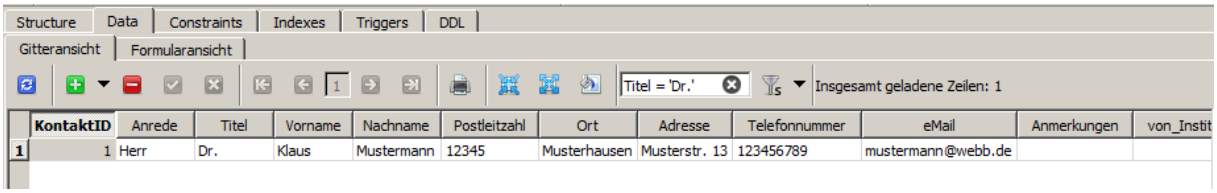
Suche mit SQL-Ausdruck

im Auswahl-Menü neben dem Trichter-Symbol wählt man "Nach einem SQL Ausdruck filtern"



	KontaktID	Anrede	Titel	Vorname	Nachname	Postleitzahl	Ort	Adresse	Telefonnummer	eMail	Anmerku
1	1	Herr	Dr.	Klaus	Mustermann	12345	Musterhausen	Musterstr. 13	123456789	mustermann@webb.de	
2	2	Frau		Monika	Mustermann	12345	Musterhausen	Musterstr. 13	123456789	musterfrau@webb.de	
3	3	Herr		Hans	Fehler	34343	Glückstadt an der Pech	Blumenweg 48	88088088	H.Fehler@webb.de	
4	4	Frau		Maria	Muster	23456	Mustern	Am Musterweg 3	234567890	m.muster@tee-online.de	
5	5	Herr		Christian	Bauer	67890	Berg am Fluß	Waldallee 78	4567890901	Chr.Bauer@geemx.de	
6	6	Herr		Lucas	Müller	54321	Bedorf	Feldweg 7	9998765	Mueller@tee-online.de	
7	7	Frau		Tara	Zander	23456	Mustern	Hauptstr. 6e	777711	Ta.Zan@webb.de	
8	8	Frau	Prof.	Hertha	Ziesow	88888	St. Muster	An der B777	543861	Prof.H.Ziesow@tee-online.de	
9	12	Frau		Maria	Berndt	67890	Berg am Fluß	Hans-Wald-Str. 4	456783067		
10	37	Herr		Henriette	Krüger	76543	Meinstadt	Feldweg 7	6543210	post@h-krueger.de	

Hinter dem Kurz-Aufruf steckt die nebenstehende SQL-Anweisung, die wir hier allerdings nicht zu Gesicht bekommen. Wer sich den Text ansieht, wird schnell erkennen, wie eine Auswahl von Daten erfolgen kann.



	KontaktID	Anrede	Titel	Vorname	Nachname	Postleitzahl	Ort	Adresse	Telefonnummer	eMail	Anmerkungen	von_Instit
1	1	Herr	Dr.	Klaus	Mustermann	12345	Musterhausen	Musterstr. 13	123456789	mustermann@webb.de		

Aufgaben:

1. *Lassen Sie nur die Datensätze anzeigen, in denen der Nachname "Fehler" enthalten ist!*
2. *Gesucht sind die nach eMail sortierten Datensätze, die den Nachnamen "Mustermann" enthalten!*

Aufgaben zu Übungs-Datenbanken:

Geo-Datenbank ""

1. !

Geo-Datenbank ""

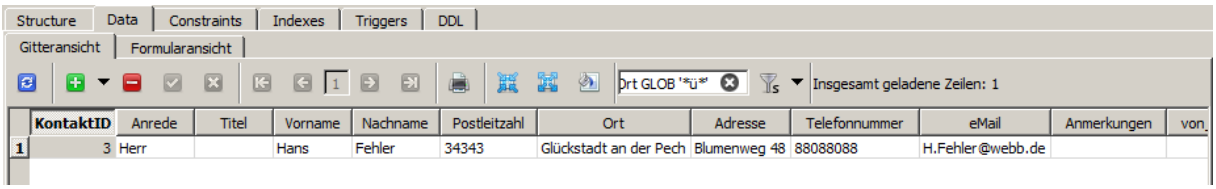
1. !

Suche mittels regulären Ausdrücken

reguläre Ausdrücke sind spezielle Formulierungen von Ausdrücken
Beschreibung von zulässigen Worten (Menge von Zeichenketten, die einem vorgegebenem Syntax entsprechen / für die vorgegebene (reguläre) Grammatik- / Bildungs-Regeln gelten)
siehe auch "Theoretische Informatik → Sprachen und Grammatiken"; Sprache vom Typ 3 nach CHOMSKY
z.B. Suche mit Wildcards (Joker-Zeichen)
mehr was für Informatik-Greeks (im Skript 🕮 **Sprachen und Automaten** gehen wir auf solche grammatikalischen Strukturen genauer ein → KLEENE-Stern)

SQL-Suchanfrage nach allen Einträgen in der Spalte "spaltenname" mit einem "ü" würde lauten:

spaltenname GLOB '*ü*'

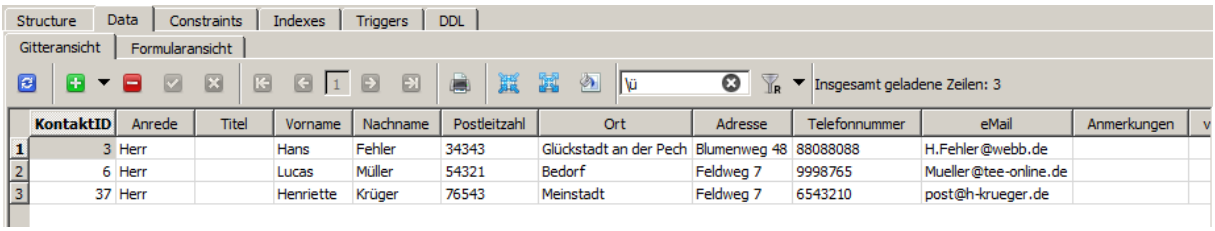


	KontaktID	Anrede	Titel	Vorname	Nachname	Postleitzahl	Ort	Adresse	Telefonnummer	eMail	Anmerkungen	von
1	3	Herr		Hans	Fehler	34343	Glückstadt an der Pech	Blumenweg 48	88088088	H.Fehler@webb.de		

für alle "ü"s in der Daten-Tabelle:

als regulärer Ausdruck ist der Sucheintrag:

\ü



	KontaktID	Anrede	Titel	Vorname	Nachname	Postleitzahl	Ort	Adresse	Telefonnummer	eMail	Anmerkungen
1	3	Herr		Hans	Fehler	34343	Glückstadt an der Pech	Blumenweg 48	88088088	H.Fehler@webb.de	
2	6	Herr		Lucas	Müller	54321	Bedorf	Feldweg 7	9998765	Mueller@tee-online.de	
3	37	Herr		Henriette	Krüger	76543	Meinstadt	Feldweg 7	6543210	post@h-krueger.de	

anderes Beispiel:
es sollen die Datensätze angezeigt werden die nur alphanumerische Zeichen enthält (???)

ltrim(lower(***spaltenname***), 'abcdefghijklmnopqrstuvwxyzäüö') = ''

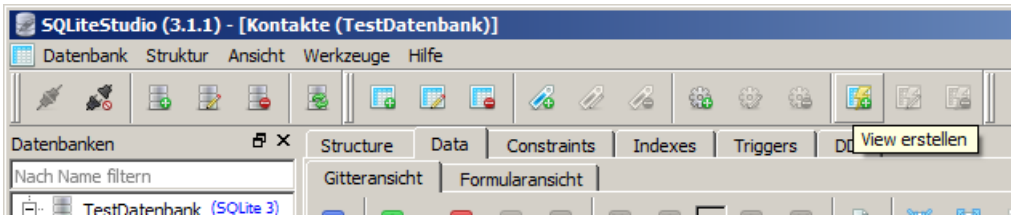
als reguläre Expression:

'[a-zA-Z]*'

(???)

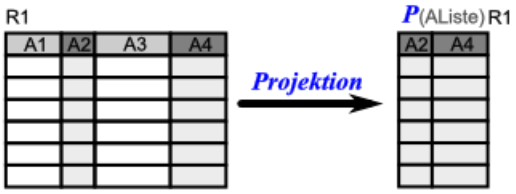
echte Abfragen / Views / Sichten

Die letzten beiden Ansichten des vorlaufenden Kapitels gehen schon stark in die Richtung, die wir uns jetzt ansehen wollen. Viele Daten müssen und sollen für den Nutzer selektiv zur Verfügung stehen. Z.B. darf nicht jeder Nutzer einer Personen-Datenbank alle persönlichen Daten sehen und das ist immer ein deutliches Beispiel: Nicht jeder darf das Gehalt einer anderen Person auslesen dürfen.
Häufig ist es auch so, dass Daten-Tabellen Unmengen von Datensätzen enthalten, die z.Z. oder überhaupt nicht mehr relevant für die Arbeit sind. Die Datensätze dürfen oft erst nach Jahren gelöscht werden, weil immer noch irgendwelche Auskunfts- oder Prüfungs-Rechte oder –Pflichten bestehen (z.B. Datenschutz-Regeln, Behörden, Aufsichtsräte, …).
Durch fertige Filter sorgt man dafür, dass die Daten, die für den nächsten Arbeitsschritt gebraucht werden, möglichst gering gehalten werden. Diese Filter werden im Allgemeinen Views, Abfragen oder Sichten genannt. Man kann sie sich immer wie rosa Brillen vorstellen. Man sieht dann alles nur noch rosa. Die Farben wurden stark reduziert.
Mit "echten" View's / Abfragen können wir auch mehrere Tabellen verknüpfend bearbeiten. Man sucht z.B. die Datensätze einer Tabelle, die bestimmte Werte enthalten, wobei diese Objekte dann auch in einer anderen Tabelle mit einem weiteren Kriterium übereinstimmen müssen. Soetwas geht bei den Programm-internen Anzeige-Abfragen eben nicht. Dort konnten wir nur eine einzelne Tabelle filtern. In der Tabellen-Algebra (Relationen-Algebra → 4.0. Relations-Algebra / Relations-Kalküle) wird das Verbinden von Tabellen JOIN genannt.
In modernen Views sind zudem noch Auswertungs-Möglichkeiten vorhanden. Früher war es oft so, dass diese Sichtweise auf die Datenbank als Bericht bezeichnet wurde.
Views lassen sich speichern und in / mit anderen Views / Abfragen / Sichten verknüpfen. I.A. sind die angelegten Views auch für andere Anwendungen / Nutzer zugänglich, wenn sie dann die notwendigen Rechte haben (Nicht jeder Angestellte darf überhaupt irgendwelche Personendaten sehen.). Die neuen View's werden dann aus den aktuellen Daten-Beständen zusammengestellt und als Tabelle zur Verfügung gestellt.
Eine weitere Nutzungs-Variante ist die Schaffung von Nutzer-spezifischen temporären Daten-Beständen, die nur in die Richtung von der Datenbank zum Nutzer funktionieren. Der Nutzer kann mit den Daten machen / probieren, was er will. Ein Zurückschreiben von Änderungen oder gar zerstörten Daten ist nicht möglich.
Zumindestens im SQLiteStudio benötigen wir für die Erstellung von Views auch einige Grundkenntnisse über die Formulierungs-Sprache SQL. Da hilft es uns schon, dass wir uns die SQL-Äquivalente der anderen – vorne besprochenen - Tätigkeiten auch mal angesehen haben. Da SQL ziemlich dicht an einem sehr abstrakten, aber einfachen, Englisch liegt, kommen wir schnell hinter die grundlegenden Prinzipien.



einfache Abfragen (Projektion(en))

In einfachen Abfragen werden nur bestimmte Spalten (Attribute) einer Tabelle ausgewählt. Diese Art der Abfrage nennt man Projektion. Sie gehört zu einem der Grund-Funktionen der Relationen-Algebra (Tabellen-Algebra).
Durch die Einschränkung der Felder werden die ursprünglichen Tabellen nun deutlich schlanker und übersichtlicher.

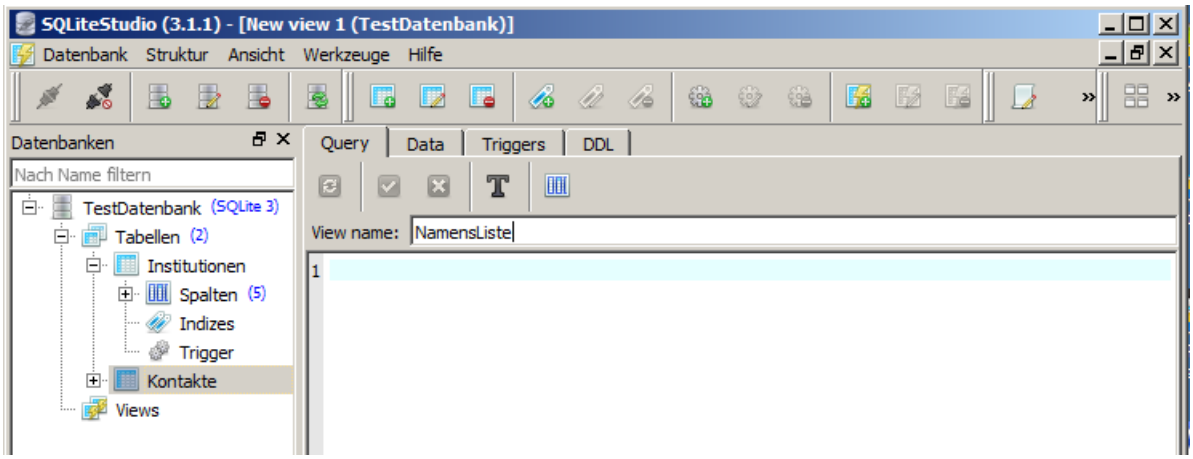


SQL-Repräsentation

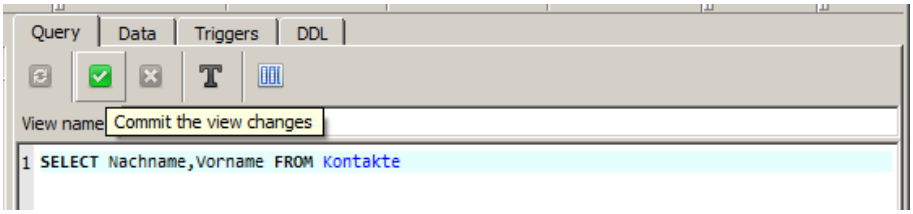
Für die tieferschürfende Arbeit mit Datenbanken kommen wir nicht um die Besprechung der SQL-Statement herum. Eine ausführliche Darstellung der SQL-Sprach-Elemente gibt es im Kapitel **(→** 5.1.11.1. Projektion (Abbildung): **)**. Hier jetzt nur die einfachen Befehle bzw. Syntax-Ausdrücke:

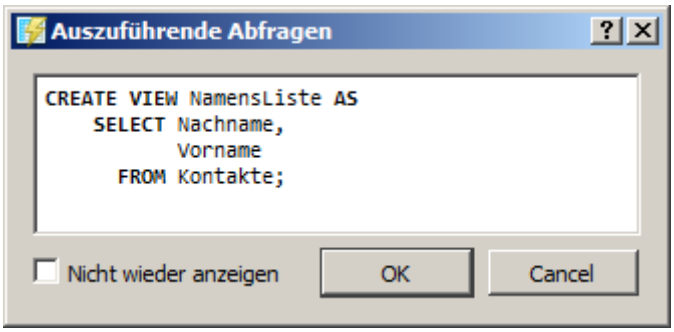
SELECT *spalteX* **{** **,** *spalteY* **}** **FROM** *tabelle***;**

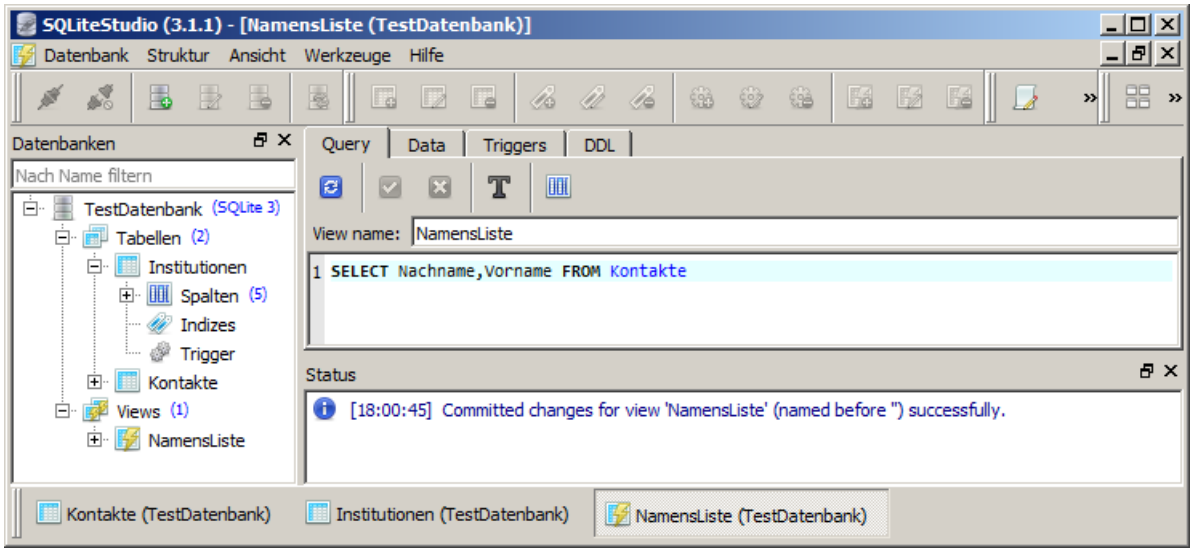
Die geschweiften Klammern (**{ }**) stehen für zusätzliche mögliche Objekte – meist im Sinne der erweiterten Wiederholung des umschlossenen Ausdruckes. Hier also der Aufzählung weiterer Spalten / Attribute. Wir stellen die Syntax-Hilfsstrukturen wie Alternativen, Optionen und Wiederholungen in roter Farbe dar, damit deren Bedeutung deutlicher wird. Die "Befehls-Wörter" der Sprache sind fett hervorgehoben. Die variablen Elemente – also bestimmte Platzhalter z.B. für Spalten- und Tabellen-Namen - sind kursiv dargestellt.



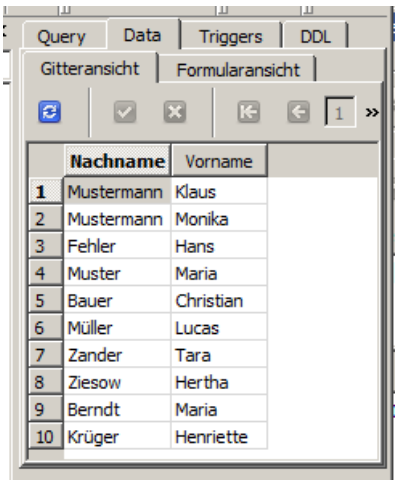
Im SQLite-Studio wird ein einfaches Syntax-Highligthning benutzt.





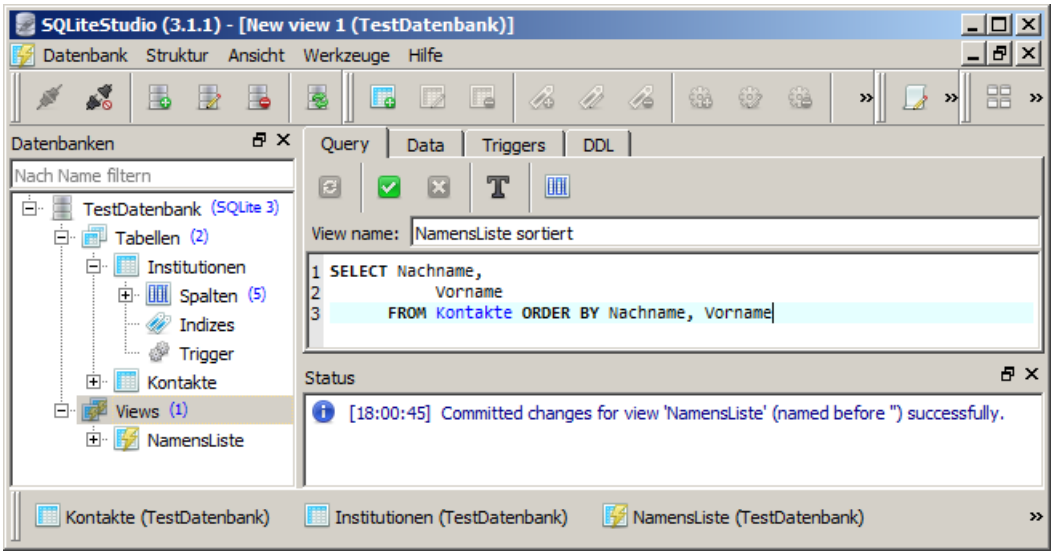


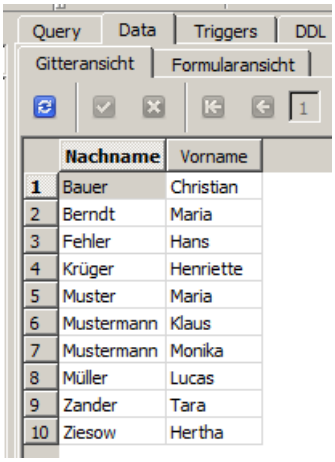
Unter dem Reiter "DDL" kann man sich auch später die zugrundeliegende SQL-Anweisung ansehen. Das ist ev. hilfreich, wenn man ähnliche Aufgaben erledigen soll und nur einzelne Sachverhalte – wie z.B. eine zusätzliche Sortierung – hinzufügen möchte.
Die SQL-Anweisung lässt sich Kopieren und z.B. in einem neuen View wieder Einfügen.
Für die Definition des View's brauchen wir nur den Teil ab SELECT und bis vor dem Semikolon. (Ein abschließendes Semikolon wird aber durch SQLiteStudio ignoriert.)
An die kopierte SQL-Teilanweisung fügen wir dann z.B. eine Sortierung hinzu.



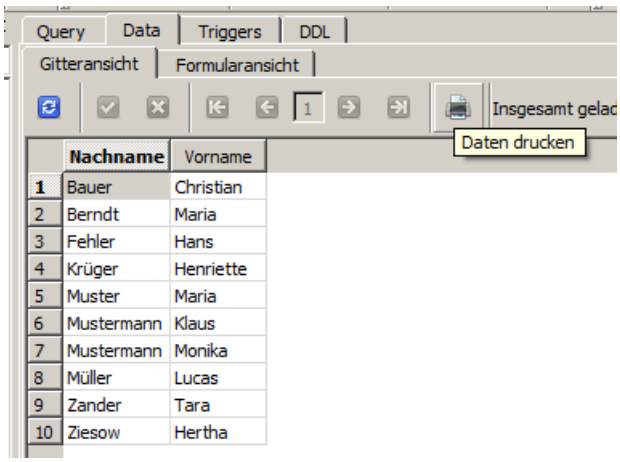
SQL-Repräsentation (→)

SELECT *spalteX* **{ ,** *spalteY* **}** **FROM** *tabelle* **ORDER BY** *spalteX* **{ ,** *spalteY* **};**





Irgendwann wird nun auch die praktische Nutzung als Papier-Ausdruck usw. interessant. Über das Drucker-Symbol in der Gitteransicht erhält man nach dem üblichen Drucker-Dialog den passenden Druck. Es entsteht ein einfacher Ausdruck der reinen Ansichten-Tabellen.
Leider scheint es derzeit noch keinen Export der Daten aus einem View zu geben.
Eine Ausnahme sind PDF-Dateien, wenn man einen PDF-Drucker installiert hat. Da es freie Versionen im Internet gibt, steht praktisch jedem diese Möglichkeit zur Verfügung.

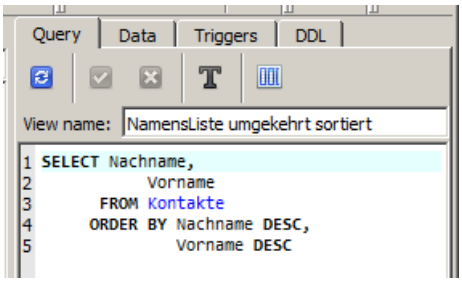


Nicht vollständig definierte View's sind manchmal intern noch präsent. Das merkt man dann, wenn auf einmal im DDL-Text zusätzlich eine DROP-Anweisung erscheint.

erweiterte Abfragen mit Sortierungen usw. usf. (Selektion(en))

Eine erste gestalterische Möglichkeit – die klassische Sortierung – haben wir eben schon bei der Erstellung eines einfachen View's besprochen. Das ORDER BY ist wohl auch nicht wirklich schwer zu verstehen. Etwas anders sehe ich hier die umgedrehte oder kombinierte Sortierung. Jetzt weden die SQL-Teil-Anweisungen – also die SELECT-Konstrukte schon komplexer und auch das eine oder andere Mal undurchsichtiger.

Aber fangen wir erst einmal mit einer umgedrehten Sortierung an. Praktisch soll die vollständig anders sortierte Liste von dem Beispiel aus dem letzten Abschnitt entstehen.



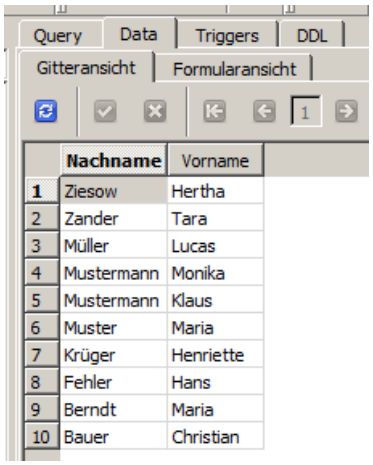
Zuerst verwenden wir den gleichen SELECT-Ausdruck, wie oben. Die umgedrehte Sortierung erreicht man mittels DESC (descending) hinter dem Spaltennamen. Bei der aufsteigenden Sortierung würde eigentlich dort ASC (ascending) stehen. Das wird aber als Standard vorausgesetzt und kann deshalb entfallen.

SQL-Repräsentation (→)

SELECT *spalteX* **{ ,** *spalteY* **}** **FROM** *tabelle*
ORDER BY *spalteX* **[ASC] | DESC] { ,** *spalteY* **[ASC] | DESC] } ;**

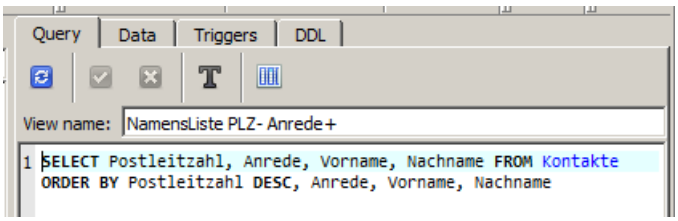
Die eckige Klammer (**[ ]**) steht optionale Bestandteile im Ausdruck. Sie können u.U. weg gelassen werden. Der senkrechte Strich (**|**) kennzeichnet Alternativen. Eine kann / muss dann ausgewählt bzw. benutzt werden.

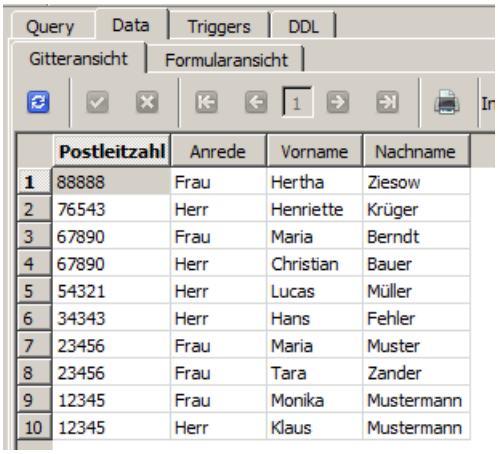
Auf die Anzeige des DDL-Statements verzichten wir mal, weil er nicht wirklich etwas neues enthält.
Auch die fertige Liste ist nicht überraschend.



Im nächsten Beispiel wollen wir die Sortier-Richtungen kombinieren. Dazu gehen wir auf unseren originalen Daten-Bestand der "Kontakte"-Tabelle zurück.

Gesucht ist nun eine Ansicht (Abfrage-Tabelle), in der die Postleitzahlen aus irgendeinem Grund rückwärts sortiert sein sollen und dann die Frauen und Männer schön und höflich (Frauen zuerst) angezeigt werden sollen.





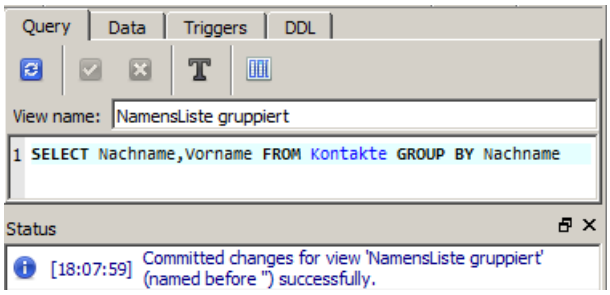
	Postleitzahl	Anrede	Vorname	Nachname
1	88888	Frau	Hertha	Ziesow
2	76543	Herr	Henriette	Krüger
3	67890	Frau	Maria	Berndt
4	67890	Herr	Christian	Bauer
5	54321	Herr	Lucas	Müller
6	34343	Herr	Hans	Fehler
7	23456	Frau	Maria	Muster
8	23456	Frau	Tara	Zander
9	12345	Frau	Monika	Mustermann
10	12345	Herr	Klaus	Mustermann

Aufgaben:

1. ***Sortieren Sie die "Kontakte" in einer neuen Sicht "NamensListe umgekehrt höflich" umgekehrt nach den "Nachnamen" und dann nach "Anrede" in der höflichen Form!***
2. ***Denken Sie sich für Ihren Nachbarn eine dreistufige Sortierung aus! Tauschen Sie die Arbeits-Aufträge aus und erledigen Sie diese!***
3. ***Drucken Sie Ergebnisse einmal aus und übergeben Sie diese zur Kontrolle an den Nachbarn!***

Gruppierungen

Bei einer größeren Anzahl von Datensätzen wünscht man sich oft ein zusätzliches Ordnungs-System. Das könnten z.B. Gruppierungen sein.
Im SQLiteStudio kann man sich zwar solche View's definieren – eine Anzeige ist aber nicht realisiert. Auch im Ausdruck bekommt man nur die übliche Tabelle.

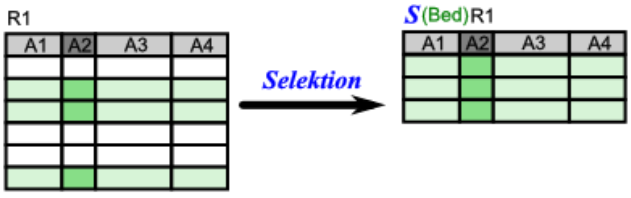


Inhalts-abhängige Abfragen (Selektion(en) / Restriktion(en))

auch als Auswahl (der Datensätze) verstanden

… WHERE *bedingung*

Ergebnis beinhaltet alle Tupel (also den vollständigen Datensatz), bei dem die Bedingung erfüllt ist



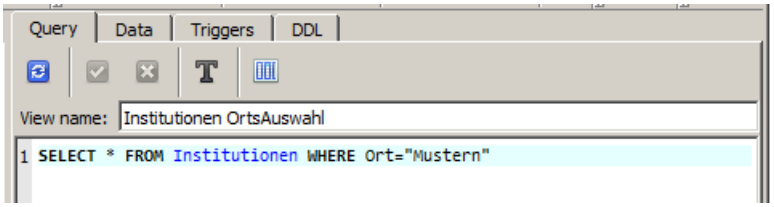
SQL-Repräsentation (→ 5.1.11.2. Selektion (Auswahl):):

SELECT * FROM *tabelle* **WHERE *bedingung*;**
bedingung kann dabei ein einfacher oder komplexer – z.B. zusammengesetzter – (SQL-)Ausdruck sein

bedingung mus als Ausdruck immer ein WAHR oder FALSCH (TRUE / FALSE) ergeben.

Die Möglichkeiten für die Formulierung der Bedingungen sind sehr komplex und umfangreich. Sie müssen ja für eine perfekte Daten-Auswahl auch sehr Leistungs-fähig sein. Aus praktischen Gründen besprechen wir hier nur einige ausgewählte Möglichkeiten. Wenn man das Prinzip verstanden hat, dann ist das "Rumspielen" mit speziellen Funktionen, Optionen und Erweiterungen nicht mehr so ein großes Problem. Eine etwas umfangreichere Darstellung der Möglichkeiten von SQL findet der Leser weiter hinten (→ 5.1.11. SELECT … FROM). Ansonsten ist hier der Zugriff auf die SQLite-Beschreibung / -Dokumentation zu empfehlen. Man kann diese direkt aus dem SQLiteStudio über "Hilfe" und "SQLite Dokumentation" erreichen. Dabei gelangt man zur offiziellen Dokumentations-Seite von SQLite (→http://sqlite.org/lang.html)

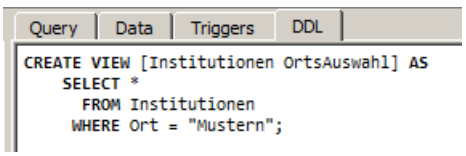
Starten wir hier mit einer einfachen Selektionen.



Gesucht seien die Datensätze aus der Tabelle "Institutionen", die den Ort Meinstadt beeinhalten.

Hinter dem WHERE – was man der Einfachheit halber mit einem **wenn** übersetzen kann – folgt der Bedingungs-Teil.

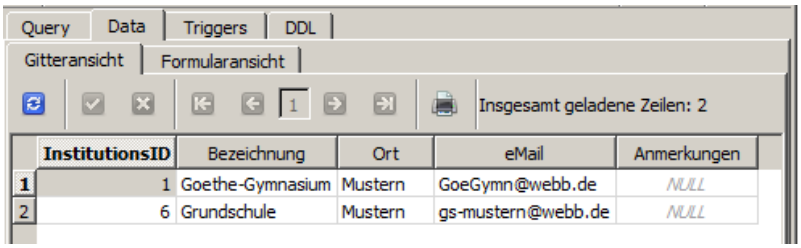
Hier soll der Ort einem vorgegebenem Text entsprechen. Das wird durch Gleichsetzung von Spaltenname und Suchtext erreicht.



Unter dem Reiter "DLL" oder vor dem Ausführen über das grüne Häkchen können wir uns den vollständigen SQL-Befehl ansehen.

In den folgenden Beispielen werden wir darauf öfter verzichten, da die wesentlichen Teile schon in der Abfrage-Beschreibung ("Query") zu sehen ist.

Das Ergebnis ist eine auf zwei Zeilen reduzierte "Institutionen"-Tabelle.

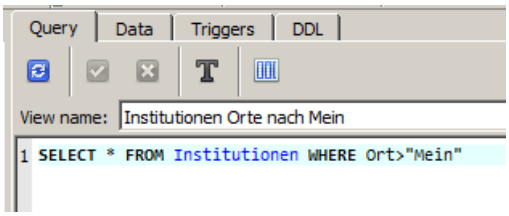


Ähnliches hatten wir ja auch schon durch eine Programm-interne Filterung erreicht (→ Filterung mittels "Text"-Muster / Such-Text).

Das Wert-Suchen in Zahlen-Feldern unterscheidet sich nicht von der Text-Suche.

Da liegt die Vermutung nicht fern, dass man auch mittels "kleiner", "kleiner-gleich", "größer-gleich" und "größer" als abfragen kann. Bei Texten wird die lexikalische Reihenfolge verwendet.

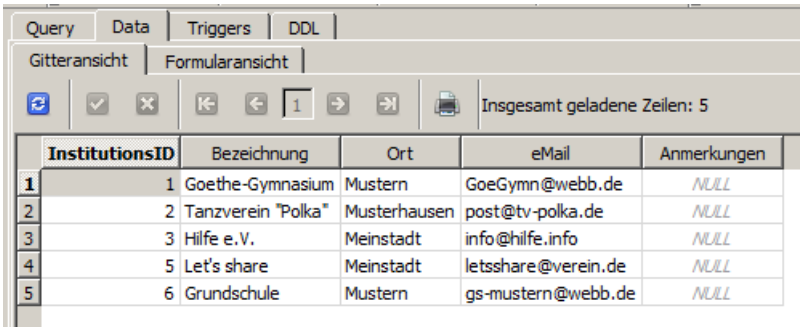
Suchen wir also alle Datensätze mit Orten, die alphabetisch nach "Mein" folgen, dann formulieren wir das mittels **Ort > "Mein"**.



Es sind auch Teil-Texte oder Einzelzeichen zulässig.

Nach dem Klick auf das grüne Häkchen bekommen wir die – gleich unten abgebildete – Tabelle.

Aber – ist hier nicht ein Fehler passiert? Meinstadt taucht auch in der Abfrage-Tabelle auf.



	InstitutionsID	Bezeichnung	Ort	eMail	Anmerkungen
1	1	Goethe-Gymnasium	Mustern	GoeGymn@webb.de	NULL
2	2	Tanzverein "Polka"	Musterhausen	post@tv-polka.de	NULL
3	3	Hilfe e.V.	Meinstadt	info@hilfe.info	NULL
4	5	Let's share	Meinstadt	letsshare@verein.de	NULL
5	6	Grundschule	Mustern	gs-mustern@webb.de	NULL

Ursache für dieses "Missverständnis" ist die Reihenfolge von Texten / Worten wie in einem Lexikon. Die kurzen Wörter kommen zuerst. Somit steht "Meinstadt" hinter "Mein".
Das SQL-System hat also völlig korrekt gearbeitet. Ev. muss man als Anfragender seine Abfrage weiter präzisieren.
Will man nun nach Text-Abschnitten suchen, die innerhalb eines Wortes liegen, müssen wir auf Joker-Zeichen zurückgreifen. Man weiss ja nicht genau, wieviele und welche Zeichen vor oder hinter dem Such-Textabschnitt liegen.
Eine beliebig lange und auch frei zusammengesetzte Zeichenkette wird mit dem Prozent-Zeichen (%) codiert. Eine solche Zeichenkette darf auch leer sein. Will man die Anzahl der Zeichen spezifizieren, dann wird das Joker-Zeichen Unterstrich (_) für immer genau ein zeichen benutzt.
Einige kleine Beispiele zur Veranschaulichung.
Man erkennt schnell, dass die Joker-Zeichen sehr mächtig sind und viele Anwendungsfälle abdecken.
Die Joker-Zeichen sind z.B. für die Suche bei deutschen Namens-Variationen, wie Meier, Maier, Meyer und Mayer hilfreich. Wer weiss schon immer exakt, wie wer genau geschrieben wurde. Eine Suche ohne die Joker-Zeichen würde immer nur einen Namen finden.

`%abc`
beschreibt alle Texte, die mit "abc" enden; dazu gehört auch "abc"

`_dorf`
entspricht z.B. "Adorf", aber nicht "Zweidorf"

`%geld%`
meint alle Texte, die irgendwo "geld" enthalten; "geld" kann auch am Anfang oder am Ende stehen

Selektion aus einem View

z.B. Suche nach "@verein" in den eMail-Adressen nur der Institutionen, die in Orten nach "Mein" kommen

Benutzung von auswertenden / berichtenden Funktionen

sum(spaltenname), max(spaltenname), min(spaltenname), … im SELECT-Teil

???trim(???), … im WHERE -Teil

statt des Feldnamens (Attributes) werden Formeln mit den Feldnamen (z.B. "Nettopreis") ohne führendes Gleichheitszeichen eingegeben (z.B. zur Berechnung des Bruttopreises: "Nettopreis" * 1,19)
es muss dann ein Alias vergeben werden, da sonst die berechnete Spalte keine Überschrift hätte, sie ist ja immer neu

Zusammenstellung ausgewählter / häufig verwendeter Operatoren usw. für SQLite

Bedingungs-Operatoren in Abfragen

Operator / Symbol(e)	Benennung	Bemerkungen / Bedeutung
=	(ist) gleich	geprüft wir exakte Identität
<>	(ist) ungleich	alle anderen Werte werden akzeptiert
<	(ist) kleiner (als)	alle Werte, die kleiner als der angegebene sind, werden akzeptiert
<=	(ist) kleiner oder gleich	erfüllt, wenn der Wert kleiner als der angegebene oder gleich groß, wie dieser, ist
>	(ist) größer (als)	alle Werte, die größer als der angegebene sind, werden akzeptiert
>=	(ist) größer oder gleich	erfüllt, wenn der Wert größer als der angegebene oder gleich groß, wie dieser, ist

logische Operatoren in Abfragen

Operator / Symbol(e)	Benennung	Bemerkungen / Bedeutung
AND	logisches UND	
OR	logisches ODER	
NOT	logisches NICHT Negation	

Platzhalter / Joker-Symbole / Wildcards

Zeichen / Symbol	Bedeutung	
%	entspricht einer beliebigen (auch leeren) Zeichenkette bzw. eines beliebigen Wertes	
_	steht für (genau) ein Zeichen in einer Zeichenkette	

ev. noch Spalten in den folgenden Tabellen / Übersichten tauschen / ersetzen

Rechen-Operatoren in Abfragen

Operator / Symbol(e)	Benennung	Bemerkungen / Bedeutung
+	Addition	
-	Subtraktion	
*****	Multiplikation	
/	Division	

Funktionen in Abfragen

Option / Funktion	Auswirkung / Bedeutung	SQL-Schlüsselwort	Bemerkungen
ohne	*keine*		
Durchschnitt	berechnet das (arithmetrische) Mittel (Mittelwert) des Feldes (Attributes)	AVG	
Anzahl	zählt die Datensätze, die in der Abfrage auftauchen	COUNT	COUNT(*) .. es werden alle Datensätze (in der Abfrage) gezählt COUNT(*Spalte*) .. es werden die Datensätze gezählt, bei denen die ***Spalte*** einen Nicht-NULL-Wert enthält
Maximum	ermittelt den größten Wert des Feldes (Attributes)	MAX	
Minimum	ermittelt den kleinsten Wert des Feldes (Attributes)	MIN	
Summe	berechnet die Summe der Feld-Werte (Attribut-Werte)	SUM	
Gruppiert	gruppiert die Datensätze nach den Werten im ausgewählten Feld (Attribut)	GROUP BY	

Filter-Bedingungen für Abfragen

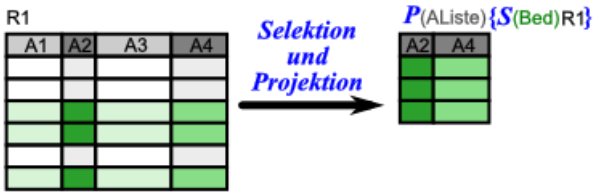
Option	Auswirkung / Bedeutung	SQL-Schlüsselwort	Bemerkungen Beispiel(e)
ohne	*keine*		
IST LEER	ist erfüllt, wenn das Feld einen NULL-Wert besitzt	IS NULL IS LEER	bei Options-Feldern (JA / NEIN) wird der unbestimmte Fall genutzt (also weder Ja noch Nein)
IST NICHT LEER	Negation von "IST LEER"; ist erfüllt, wenn das Feld einen (von NULL abweichenden) Wert besitzt	IS NOT NULL IS NOT LEER	
LIKE **WIE**	Übereinstimmung; ist erfüllt, wenn das Feld den entsprechenden Wert besitzt	LIKE	WIE 'Muster*' oder WIE 'Muster????' (liefert: z.B. "Mustermann" und "Musterfrau" zurück)
ZWISCHEN *min* **UND** *max*	im Intervall; ist erfüllt, wenn der Feldinhalt zwischen den Angaben ***min*** und ***max*** liegt	BETWEEN *min* AND *max*	
IN (*Liste***)**	Entsprechung; Enthaltung ist erfüllt, wenn das Feld mit einem Wert aus der (Semikolon-getrennten) übereinstimmt		IN (3; 6; 12; 24) IN ('Moskau'; 'Berlin'; 'New York')
NICHT …	Negation; negiert den folgenden	NOT …	

	Ausdruck		
= WAHR	Validierung; ist erfüllt, wenn der Feldinhalt ***wahr*** enthält	= TRUE	
= FALSCH	Falsifizierung ist erfüllt, wenn der Feldinhalt ***falsch*** enthält	= FALSE	
EXISTS(SELECT-Anweisung**)**	falls mind. ein Datensatz existiert, dann gibt die Funktion TRUE zurück, sonst FALSE		

kombinierte Abfragen (Projektion(en) + Selektion(en))

Verbindung von Selektion und Projektion (bezüglich einer Tabelle)
stellen die wirklichen praktisch nutzbaren Sichten in Datenbank-System und vor allem in / für Bedienoberflächen dar
zum Einen wird man so der Datenflut herr (Selektion) und zum anderen schränkt man die Daten auf den Teil ein, mit dem ein bestimmter Nutzer arbeiten darf (Projektion)

sachlich ist die Reihenfolge der Verbindung von Selektion und Projektion egal, es ergibt sich immer das gleiche Ergebnis
in SQL wird zuerst die Projektion formuliert (SELECT …) und dann folgt die Selektion (…WHERE …)



bei der Umsetzung in Datenbank-Kommandos (Datenbank-System-interne Befehle) ist eine vorlaufende Selektion meist sinnvoller, da so die Anzahl der weiter zu verarbeitenden Datensätze deutlich eingeschränkt wird. Dannach werden dann die notwendigen Selektionen vorgenommen (die Anzahl der Attribute in einem Datensatz meist deutlich kleiner als die Zahl der Datensätze einer Tabelle)
im SQL schnell aus den vorher besprochenen Anweisungen zusammenzusetzen
statt des * im SELECT-Teil der Selektionen werden nun konkrete Spalten (Attribute) genannt

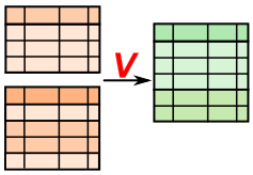
oder bei vorhandenen (ausführlichen) SELECT-Teilen wird nun ein WHERE-Statement ergänzt

vorteilhaft ist, dass man die SQL-Befehle schön teilweise ausprobieren kann und passende Teile dann zusammenkopieren / zusammenstellen kann
trotz recht langer SQL-Anweisungen ist dann die Fehlerquote relativ gering

verknüpfte Abfragen (über mehrere Tabellen / Sichten hinweg) (Join('s))

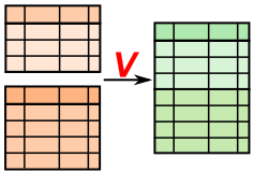
Die Benutzung der Daten aus nur einer Tabelle ist eher die Ausnahme. Sehr viel häufiger werden Daten aus mehreren Tabellen gemeinsam für bestimmte Aufgaben gebraucht.
Um die Daten effektiver zu speichern, haben wir die Tabellen-Strukturen optimiert (→ Normalisierung) und dabei bestimmte Daten in andere Tabellen ausgelagert. In den ursprünglichen Tabellen stehen jetzt ja nur noch Verweise (Fremd-Schlüssel) zu den Datensäzen in den ausgelagerten Daten.
In der Praxis brauchen wir aber die kombinierten Daten. Dazu werden Tabellen wieder miteinander verbunden. Es gibt verschiedene Verbünde (Join's).
Gehen wir aber systematisch vor. Betrachten wir zuerst Tabellen mit vergleichbarer Struktur. In der Praxis könnten das Bestands-Daten aus verschiedenen Standort-Datenbanken, die nun zusammengefügt werden sollen. Solche Tabellen könnten z.B. aus Projektionen (→) stammen – also auch schon View's sein.

R1 UNION R2
fügt die Zeilen von zwei Tabellen mit gleicher Spaltenzahl zu einer neuen Tabelle zusammen
die Namen der Spalten müssen nicht gleich sein, die Datentypen müssen aber identisch sein!



doppelte Zeilen werden automatisch unterdrückt, ist dies nicht gewünscht, dann muss UNION ALL benutzt werden

Die "einfache" Vereinigung enthält also praktisch auch noch Selektionen, was bei der vollständigen Vereinigung unterbleibt.



SQL-Repräsentation (→)

SELECT *spalteX* **{ ,** *spalteY* **}** **FROM** *tabelle*
ORDER BY *spalteX* **[ASC] | DESC] { ,** *spalteY* **[ASC] | DESC] } ;**

Aufgaben:

1. *Vergleichen Sie die Programm-internen (An-)Sichten mit den echten Sichten (View's, Abfragen)!*
2.
3.

3.1.7.1.3.6. Export von Daten

Daten austauschen ist heute eine der wichtigsten informatischen Aufgaben. Erst bei der Möglichkeit Daten in verschiedenen Programmen oder sie mehrfach zu nutzen, wird deren heutige Potenz wirksam.
Der Daten-Export aus dem einen Programm ist also genauso wichtig, wie der Import in dem anderen. Als Austausch-Formate haben sich einige sehr einfache Datei-Typen etabliert. Einige stellen wir hier im Rahmen der Export-Möglichkeiten des SQLiteStudio's vor.
Um die Daten-Repräsentation in den einzelnen Datei-Formaten deutlicher zu machen, zeigen wir den exportierten Daten-Bestand der "Institutionen"-Tabelle immer mit an. Da bieten sich schöne Vergleichs-Möglichkeiten.
Was wir in den nachfolgenden Abschnitten für den Tabellen-Export beschreiben, funktioniert auch für ganze Datenbanken und für Sichten (View's). Das Prinzip bleibt das Gleiche.

Export als CSV

Daten im CSV-Format sind sehr universell zu nutzen. CSV steht für Comma-separeted values (selten auch: Character-separeted values). Schon in einfachen Editoren lassen sie sich anzeigen und bearbeiten. Mehr Funktionen und Möglichkeiten hat man in Tabellenkalkulationen – wie z.B. microsoft®EXCEL® und / oder libreoffice / openoffice CALC). Auch für einen Austausch in andere Betriebssystem-Plattformen ist das CSV-Format hervorragend geeignet. Praktisch gibt es in jedem gängigen Betriebssystem mindestens ein Programm, dass mit CSV klar kommt. Die allgemeinste Codierung ist dann 7-bit-ASCII.
Außer Komma-getrennt gibt es viele Spezial-Optionen. So sind heute viele andere Seperatoren möglich. Das ist besonders dann wichtig, wenn die Daten selbst Komma's enthalten (z.B. Texte oder deutsche Zahlen-Formate). Hier bieten sich dann Semikolon's oder Tabulatoren als Trenner an.
Die CSV-Dateien werden manchmal auch als TXT-Dateien abgelegt. Da man auch diese mit einem Editor ansehen kann, ist das Ermitteln des internen Datei-Formates meist nicht wirklich schwierig.
Die "Institutionen"-Tabelle sieht dann (in einem Text-Editor) so aus:

```
InstitutionsID,Bezeichnung,Ort,eMail,Anmerkungen
1,Goethe-Gymnasium,Mustern,GoeGymn@webb.de,
2,"Tanzverein ""Polka""",Musterhausen,post@tv-polka.de,
3,Hilfe e.V.,Meinstadt,info@hilfe.info,
4,Traditionsverein,Cedorf,tradi@verein.de,
5,Let's share,Meinstadt,letsshare@verein.de,
6,Grundschule,Mustern,gs-mustern@webb.de,
```

Aufgaben:

1. ***Was ist eigentlich 7-bit-ASCII?***
2. ***Informieren Sie sich in Wikipedia über die Definition des CSV-Formates!***
3. ***Exportieren Sie die Tabelle "Kontakte" in eine CSV-Datei!***
4. ***Überlegen Sie sich einen Algorithmus, der einfache Daten aus einer originalen CSV-Datei imortieren kann!***

Export als PDF

Das PDF-Format (Printable Document Format) ist nicht bzw. nur teilweise Text-basiert. Im Allgemeinen kommt man an die Daten-Felder nicht heran. Dafür ist dieses Format ja auch nicht gedacht. Mit ihm sollen Daten in einer druckbaren Form weitergegeben werden, die beim Betrachter (End-Nutzer) nicht mehr verändert werden sollen / dürfen. Genaugenommen handelt es sich also auch nicht um ein Austausch-Format zum Hin- und Her-Tauschen, sondern nur zum eingeschränkten Informations-Austausch.
Aber Achtung! Nicht signierte PDF-Dateien sind manipulierbar! Ihre Verwendung ist also immer mit Vorsicht zu genießen.

Ausschnitte der PDF-Export-Datei (Ansicht aus einem Editor)

```
%PDF-1.4
1 0 obj
…
…
>>
stream
xœÝZKo7¾ï¯à¹€ÖŸ@Vâ=0, ‡ ‡ÂiR
```

```
VP7‡þýò±Ë¥4”L[´#G…
…
>>
startxref
16107
%%EOF
```

Export als HTML

HTML ist die Abkürzung von Hypertext Markup Language (Hypertext-Markierungs-Sprache). Auch hier steht der einseitige Informations-Austausch zum End-Nutzer im Vordergrund. Ein Import ist zwar möglich, dazu eignen sich ähnliche Formate, wie das XML deutlich besser.
HTML-Dateien sind reine Text-Dateien, in denen die Elemente strukturiert abgelegt werden. Hauptzweck ist die Darstellung von Texten, wie sie eben im Internet bei den verschiedenen Internet-Seiten zur Anwendung kommen. Das HTML-Format bestimmt deshalb auch vorrangig die allgemeine Darstellung auf dem Ziel-Rechner. Als Programm wird i.A. ein Browser (z.B. mozilla Firefox oder microsoft®Internet-Explorer®) benutzt.
Die Gestaltung der Daten-Anzeige wird von Tag's bestimmt, die im HTML in eckige Klammern eingeschlossen sind. Bei den meisten Tag's gibt es einen einleitenden und einen beendenden Tag. Beide lauten gleich. Der beendende Tag hat zusätzlich noch einen Schrägstrich (Slash) vor dem Tag-Text. Die Tag's <HTML> und </HTML> bilden also ein Paar und umschließen z.B. den gesamten HTML-Text. Intern kommen dann geschachtelt weitere Tag's zur Anwendung. Einer der wenigen Tag's, die nur einzeln vorkommen, ist der Zeilenumbruch
 (für break). Die Groß- und Kleinschreibung der Tag-Texte ist egal. Die meisten Systeme benutzen aber Groß-Buchstaben zur auffälligeren Kennzeichnung. Innerhalb von einigen Tag's dürfen Optionen angegeben werden. Diese werden dann eher klein geschrieben.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN" "http://www.w3.org/TR/REC-
html40/strict.dtd">
<html>
    <meta http-equiv="Content-Type" content="text/html; charset=System"/>
    <title>Exported table: Institutionen</title>
    <style type="text/css">
        table
        {
            border-style: solid;
            border-width: 1px;
            border-color: black;
```

```
            border-collapse: collapse;
…
…
        table tr td.rownum
        {
            padding: 0px 3px 0px 3px;
            border-style: solid;
            border-width: 1px;
            border-color: #666666;
            background-color: #DDDDDD;
            text-align: right;
        }

    </style>
    <body>
        <table>
            <tr class="title">
                <td colspan="6" align="center">Table: Institutionen</td>
            </tr>
            <tr class="header">
                <td align="right">
                    <b><i>#</i></b>
                </td>
                <td>
                    <b>InstitutionsID</b><br/>INTEGER
                </td>
                <td>
                    <b>Bezeichnung</b><br/>STRING
                </td>
                <td>
                    <b>Ort</b><br/>STRING
                </td>
                <td>
                    <b>eMail</b><br/>STRING
                </td>
                <td>
                    <b>Anmerkungen</b><br/>BLOB
                </td>
            </tr>
            <tr>
                <td class="rownum">
                    <i>1</i>
                </td>
                <td align="right">
                    1
                </td>
                <td align="left">
                    Goethe-Gymnasium
                </td>
                <td align="left">
                    Mustern
                </td>
                <td align="left">
                    GoeGymn@webb.de
                </td>
                <td align="left" class="null">
                    <i>NULL</i>
                </td>
            </tr>
            <tr>
                <td class="rownum">
                    <i>2</i>
                </td>
                <td align="right">
                    2
                </td>
                <td align="left">
                    Tanzverein &amp;quot;Polka&amp;quot;
                </td>
                <td align="left">
                    Musterhausen
```

```
            </td>
            <td align="left">
                post@tv-polka.de
            </td>
            <td align="left" class="null">
                <i>NULL</i>
            </td>
        </tr>
        <tr>
            <td class="rownum">
                <i>3</i>
            </td>
            <td align="right">
                3
            </td>
            <td align="left">
                Hilfe e.V.
            </td>
            <td align="left">
                Meinstadt
            </td>
            <td align="left">
                info@hilfe.info
            </td>
            <td align="left" class="null">
                <i>NULL</i>
            </td>
        </tr>
        <tr>
            <td class="rownum">
                <i>4</i>
            </td>
            <td align="right">
                4
            </td>
            <td align="left">
                Traditionsverein
            </td>
            <td align="left">
                Cedorf
            </td>
            <td align="left">
                tradi@verein.de
            </td>
            <td align="left" class="null">
                <i>NULL</i>
            </td>
        </tr>
        <tr>
            <td class="rownum">
                <i>5</i>
            </td>
            <td align="right">
                5
            </td>
            <td align="left">
                Let's share
            </td>
            <td align="left">
                Meinstadt
            </td>
            <td align="left">
                letsshare@verein.de
            </td>
            <td align="left" class="null">
                <i>NULL</i>
            </td>
        </tr>
        <tr>
            <td class="rownum">
                <i>6</i>
```

```
                </td>
                <td align="right">
                    6
                </td>
                <td align="left">
                    Grundschule
                </td>
                <td align="left">
                    Mustern
                </td>
                <td align="left">
                    gs-mustern@webb.de
                </td>
                <td align="left" class="null">
                    <i>NULL</i>
                </td>
            </tr>
        </table>
        <br/><br/>
        <i>Document generated by SQLiteStudio v3.1.1 on Mi Dez 27 19:41:39
2017</i>
    </body>
</html>
```

In einem Browser sieht der Export dann natürlich viel besser aus – zumindestens für uns Menschen. Für den zweiseitigen Datenaustausch zwischen Rechnern ist das HTML-Format ja auch nicht wirklich gedacht.

Aufgaben:

1. *Exportieren Sie die Tabelle "Kontakte" in eine HTML-Datei!*
2. *Betrachten Sie diese Datei in einem Browser!*

für die gehobene Anspruchsebene:

3. *Erstellen Sie mit einem (HTML-)Editor eine minimale HTML-Datei, das eine ganz einfache Tabelle mit den exportierten Daten zeigt!*
4. *Übertragen Sie die Datei (z.B. mittels USB-Stick oder per eMail) auf ein Gerät mit einem anderen Betriebssystem und / oder anderem Browser! Betrachten Sie die Datei dann dort in einer passenden App! Was stellen Sie fest?*

Export als XML

XML ist stark mit HTML verwandt. Allerdings geht es beim XML (Extensible Markup Language) um die strukturierte Speicherung von Daten in einem universellen Austausch-Format. XML-Dateien sind also auch Text-Dateien, in denen Daten mittel Tag's vor allem Maschinenlesbar gemacht werden sollen. Eine gewisse Lesbarkeit für Menschen sollte aber erhalten bleiben.
Die Einfachheit von XML-Dateien lässt sie zu einem universellen Daten-Format auch für zukünftige Programme und Datenbanken werden. Import- und Export-Filter sind einfach zu programmieren. Die meisten Programmiersprachen bringen gleich von Haus aus Hilfsmittel zur effektiven Bearbeitung von XML-Dateien mit.
Applikationen dürfen eigene Tag's definieren und verwenden. So könnte ein spezielles Programm für die Verwaltung unserer "Institutionen"-Tabelle sich die Tag-Paare <InstitutionsID> </InstitutionsID>, <Bezeichnung> </Bezeichnung>, <Ort> </Ort>, <eMail> </eMail> und <Anmerkungen> </Anmerkungen> definieren. Die Definition wird in einer extra DTD-Datei gespeichert, auf die in der eigentlichen XML-Datei hingewiesen wird. Diese Art der XML-Dateien sind dann für Menschen deutlich besser zu lesen. Zeilenumbrüche verbessern eben-

falls die Lesbarkeit. In der reinen Anwendung sind sie nicht notwendig und fehlen entsprechend.

```
<?xml version="1.0" encoding="System"?>
<table>
    <database></database>
    <name>Institutionen</name>
    <ddl>CREATE TABLE Institutionen (InstitutionsID INTEGER PRIMARY KEY ASC ON
CONFLICT ROLLBACK AUTOINCREMENT, Bezeichnung STRING (30), Ort STRING (20), eMail
STRING (20), Anmerkungen BLOB);</ddl>
    <columns>
        <column>
            <name>InstitutionsID</name>
            <type>INTEGER</type>
            <constraints>
                <constraint>
                    <type>PRIMARY KEY</type>
                    <definition>PRIMARY KEY ASC ON CONFLICT ROLLBACK
AUTOINCREMENT</definition>
                </constraint>
            </constraints>
        </column>
        <column>
            <name>Bezeichnung</name>
            <type>STRING</type>
        </column>
        <column>
            <name>Ort</name>
            <type>STRING</type>
        </column>
        <column>
            <name>eMail</name>
            <type>STRING</type>
        </column>
        <column>
            <name>Anmerkungen</name>
            <type>BLOB</type>
        </column>
    </columns>
    <rows>
        <row>
            <value column="0">1</value>
            <value column="1">Goethe-Gymnasium</value>
            <value column="2">Mustern</value>
            <value column="3">GoeGymn@webb.de</value>
            <value column="4" null="true"/>
        </row>
        <row>
            <value column="0">2</value>
            <value column="1">Tanzverein &amp;quot;Polka&amp;quot;</value>
            <value column="2">Musterhausen</value>
            <value column="3">post@tv-polka.de</value>
            <value column="4" null="true"/>
        </row>
        <row>
            <value column="0">3</value>
            <value column="1">Hilfe e.V.</value>
            <value column="2">Meinstadt</value>
            <value column="3">info@hilfe.info</value>
            <value column="4" null="true"/>
        </row>
        <row>
            <value column="0">4</value>
            <value column="1">Traditionsverein</value>
            <value column="2">Cedorf</value>
            <value column="3">tradi@verein.de</value>
            <value column="4" null="true"/>
        </row>
        <row>
            <value column="0">5</value>
```

```
            <value column="1">Let's share</value>
            <value column="2">Meinstadt</value>
            <value column="3">letsshare@verein.de</value>
            <value column="4" null="true"/>
        </row>
        <row>
            <value column="0">6</value>
            <value column="1">Grundschule</value>
            <value column="2">Mustern</value>
            <value column="3">gs-mustern@webb.de</value>
            <value column="4" null="true"/>
        </row>
    </rows>
</table>
```

Aufgaben:

1. *Finden Sie die Strukturfehler in der nachfolgenden abstrahierten XML-Datei!*

	Quell-Text	Kommentare
1	<Zoo>	
2	<Tiere>	
3	<Lurche>	
4	<Pfleger> Meier </pfleger>	
5	<Lurch> Moorfrosch </Lurch>	
6	<Lurch> Ringelnatter </Lurch>	
7	</Lurche>	
8	<Saeuger>	
9	<Betreuer> Dietrich </Betreuer>	
10	<Betreuer> Neumann <Betreuer>	
11	<Saeuger> Elephant </Saeuger>	
12	<Saeuger> Löwe </Löwe>	
13	<Tiere>	
14	<Gebaeude>	
15	Gehege: Rotwildgehege	
16	Käfig: Löwenkäfig	
17	<Gebaeude>	
18	<Personal>	
19	<Pfleger> Meier </Pfleger>	
20	<Pfleger> Dietrich </Pfleger>	
21	<Kassierer> Neumann </Kassierer>	
22	</Personal>	
23	</Direktor> <Direktor>	
24	>/Zoo>	

2. *Erstellen Sie mit einem Text-Editor eine XML-Datei mit den im Text erwähnten Tag's für eine Institutions-XML-Datei!*
3. *

Export als JSON

JSON-Dateien sind sehr moderne, Text-basierte Austausch-Dateien. Auch für sie gibt es oft fertige Bibliotheken zu den verschiedenen Programmiersprachen. Ursprünglich stammt das Format aus der Programmiersprache JAVA. Der vollständige Name lautet: JavaScript Object Notation.
Beim JSON-Format sind die Daten vorrangig mittels Klammern und Komma als Separator strukturiert. Je nach Datenstruktur wird ein Eintrag z.B. mittels Name und Wert verwaltet. Beide sind durch einen Doppelpunkt getrennt. Für tabellarische Daten wird dann z.B. in Kopfzeile und Datenzeile(n) unterschieden. Das JSON-Format ist vor allem wegen seines

sparsamen Einsatzes von Steuer-Zeichen beliebt. Im Allgemeinen sind JSON-Dateien kleiner als XML-Dateien mit dem gleichen Inhalt.

```
{
    "type": "table",
    "database": null,
    "name": "Institutionen",
    "withoutRowId": true,
    "ddl": "CREATE TABLE Institutionen (InstitutionsID INTEGER PRIMARY KEY ASC ON
CONFLICT ROLLBACK AUTOINCREMENT, Bezeichnung STRING (30), Ort STRING (20), eMail
STRING (20), Anmerkungen BLOB);",
    "columns": [
        {
            "name": "InstitutionsID",
            "type": "INTEGER",
            "constraints": [
                {
                    "type": "PRIMARY KEY",
                    "definition": "PRIMARY KEY ASC ON CONFLICT ROLLBACK
AUTOINCREMENT"
                }
            ]
        },
        {
            "name": "Bezeichnung",
            "type": "STRING"
        },
        {
            "name": "Ort",
            "type": "STRING"
        },
        {
            "name": "eMail",
            "type": "STRING"
        },
        {
            "name": "Anmerkungen",
            "type": "BLOB"
        }
    ],
    "rows": [
        [
            1,
            "Goethe-Gymnasium",
            "Mustern",
            "GoeGymn@webb.de",
            null
        ],
        [
            2,
            "Tanzverein \\"Polka\\"",
            "Musterhausen",
            "post@tv-polka.de",
            null
        ],
        [
            3,
            "Hilfe e.V.",
            "Meinstadt",
            "info@hilfe.info",
            null
        ],
        [
            4,
            "Traditionsverein",
            "Cedorf",
            "tradi@verein.de",
            null
        ],
        [
```

```
            5,
            "Let's share",
            "Meinstadt",
            "letsshare@verein.de",
            null
        ],
        [
            6,
            "Grundschule",
            "Mustern",
            "gs-mustern@webb.de",
            null
        ]
    ]
}
```

Aufgaben:

1. *Informieren Sie sich in Wikipedia über die Definition des JSON-Formates!*
2. *Erstellen Sie auf der Grundlage des Wikipedia-Artikels eine rudimentäre JSON-Datei (handschriftlich auf dem Papier) zur / aus der "Institutionen"-Tabelle!*
3. *Vergleichen Sie das CSV-, XML- und JSON-Datei-Format in einer Tabelle! Kennzeichen Sie Gemeinsamkeiten und Unterschiede auch zwischen den verschiedenen Format-Paaren!*

Export als SQL

Die Datenbank-Sprache SQL kann neben der eigentlichen Arbeit an einem Datenbank-Management-System auch zum Datenaustausch benutzt werden. SQL ist ja quasi die Muttersprache der relationalen Datenbanken.
SQL-Dateien sind ebenfalls Text-basiert und sowohl Maschinen- als auch sehr gut von Menschen lesbar. Zeilen mit zwei Bindestrichen sind Kommentarzeilen und können auch weggelassen werden.
In Mehr-Nutzer-Systemen muss die spätere Einspeisung von SQL-Code sicherstellen, dass nicht ein anderer Nutzer dazwischenfunkt. Deshalb sind noch einige zusätzliche SQL-Anweisungen (SQL-Statement's) notwendig. Das sind z.B. BEGIN und COMMIT TRANSAKTION.
Interessant ist für den reinen Daten-Import später nur die Sequenz der INSERT-Anweisungen. U.U. muss eine SQL-Export-Datei für spezielle Zwecke in einem Editor angepasst werden, damit z.B. eine schon vorhandene Tabelle nicht zerstört wird.

```
--
-- File generated with SQLiteStudio v3.1.1 on Mi Dez 27 16:39:54 2017
--
-- Text encoding used: System
--
PRAGMA foreign_keys = off;
BEGIN TRANSACTION;

-- Table: Institutionen
CREATE TABLE Institutionen (InstitutionsID INTEGER PRIMARY KEY ASC ON
CONFLICT ROLLBACK AUTOINCREMENT, Bezeichnung STRING (30), Ort STRING (20),
eMail STRING (20), Anmerkungen BLOB);
INSERT INTO Institutionen (InstitutionsID, Bezeichnung, Ort, eMail, Anmer-
kungen) VALUES (1, 'Goethe-Gymnasium', 'Mustern', 'GoeGymn@webb.de',
NULL);
```

```
INSERT INTO Institutionen (InstitutionsID, Bezeichnung, Ort, eMail, Anmer-
kungen) VALUES (2, 'Tanzverein "Polka"', 'Musterhausen', 'post@tv-
polka.de', NULL);
INSERT INTO Institutionen (InstitutionsID, Bezeichnung, Ort, eMail, Anmer-
kungen) VALUES (3, 'Hilfe e.V.', 'Meinstadt', 'info@hilfe.info', NULL);
INSERT INTO Institutionen (InstitutionsID, Bezeichnung, Ort, eMail, Anmer-
kungen) VALUES (4, 'Traditionsverein', 'Cedorf', 'tradi@verein.de', NULL);
INSERT INTO Institutionen (InstitutionsID, Bezeichnung, Ort, eMail, Anmer-
kungen) VALUES (5, 'Let''s share', 'Meinstadt', 'letsshare@verein.de',
NULL);
INSERT INTO Institutionen (InstitutionsID, Bezeichnung, Ort, eMail, Anmer-
kungen) VALUES (6, 'Grundschule', 'Mustern', 'gs-mustern@webb.de', NULL);

COMMIT TRANSACTION;
PRAGMA foreign_keys = on;
```

Aufgaben:

1. *Exportieren Sie die Tabelle "Kontakte" in eine SQL-Datei!*
2. *Erstellen Sie sich eine zweite Datenbank "ZweitDatenbank" im SQLiteStudio und importieren Sie die SQL-Datei in diese Datenbank!*

3.1.7.1.3.7. Import der exportierten Daten in andere Programme

Alle oben gekennzeichneten Text-Datei-Formate lassen sich mit einfachen Editoren anzeigen. Nicht immer wird dabei die eigentliche Daten-Struktur erkennbar, da die Einträge in den Daten z.T. ohne Einrückungen und zusätzliche Zeilenumbrüche auskommen. Einige Spezial-Programme verfügen aber über Programm-interne Formatierungs-Hilfen, die recht ansehnliche Ergebnisse erzeugen.
Geht es nur um die Anzeige, dann helfen bei HTML- und XML-Dateien auch Browser (z.B. mozilla Firefox oder microsoft®Internet-Explorer®) weiter.
Tabellen-Kalkulationen sind ansosnten immer gute Import-Programme für Daten-Tabellen. Sowohl microsoft®EXCEL® oder libreoffice / openoffice CALC kommen sehr gut mit den exportierten CSV-Dateien klar.
Da gilt sicher auch für andere Office-Produkte (FreeOffice von SoftMaker) oder spezielle Tabellenkalkulations-Apps. Da sie aber im Schulbereich und in der freien Arbeits-Praxis eine geringere Rolle spielen, wurde die Funktionabilität nicht weiter geprüft.
Beim Import müssen zumeist die Textcodierung, der Seperator und das Vorhandensein der Kopf-Zeilen bzw. Defintions-Strukturen angegeben werden. Stellt man hier alles ordentlich ein, dann bekommt man meist sehr ordentliche Daten-Tabellen. Diese können dann in einem Programm-internen Tabellkalkulations-Dateityp abgspeichert werden. Es stehen dann auch alle Arbeits-Funktionen und Programm-Leistungen (z.B. Diagramme) zur Verfügung.

3.1.7.1.3.8. Nutzung der Datenbank über Programmiersprachen

Die Programmierung und "Fremd"-Nutzung von Datenbanken besprechen wir weiter hinten (→ 6. Datenbanken - Programmierung). Interessanterweise sind sich die üblichen Programmiersprachen im Umgang mit SQL-Datenbanken recht einheitlich.
An dieser Stelle soll nur mal kurz ein Beispiel aus der Programmiersprache Python aufgezeigt werden.

Datennutzung / Datenzugriff mit Python

Die Kommunikation zwischen Datenbank(-Server) und Client (unser Python-Programm) läuft über SQL-Code. Das funktioniert praktisch bei allen Datenbanken. Eine entsprechende SQL-Schnittstelle gehört heute zu den Minimal-Funktionen. Niemand will für ein neues Datenbanksystem alle seine Daten neu eingeben oder seine Hilfs-Scripte neu programmieren. Sachlich gibt es auch wenige Gründe, weshalb man kein SQL nutzen sollte. Diese Gründe sind vornehmlich für Profi's interessant und spielen in normalen Datenbank-Welten kaum eine Rolle.

```
import sqlite3

conn = sqlite3.connect('daten/Kontakte.dat')
curs = conn.cursor()
curs.execute("CREATE TABLE personen(PID, Vorname, Name, eMail)")

DatenListe=[(1,"Monika","Musterfrau","musterfrau@webb.de"),
            (2,"Klaus","Mustermann","muma@tee-online.de"),
            (3,"Prof. Lisa","Klug","Prof.L.Klug@uni-mustern.de)]

for Elem in DatenListe:
    curs.execute("Insert INTO personen VALUES (?,?,?,?)", Elem)

…

curs.close()
conn.close()
```

curs.fetchone() liefert eine Ergebnis-Zeile zur Anfrage als Tupel
curs.fetchmany(n) liefert n Ergebniszeilen zur Anfrage als n-Tupel von Tupeln
curs.fetchall() liefert alle Ergebniszeilen zur Anfrage als Tupel von Tupeln

zusätzliche Informationen und Programm-Beispiele gibt es im Programmierungs-Kapitel (→)
oder auch im Skript 🕮 **Python**

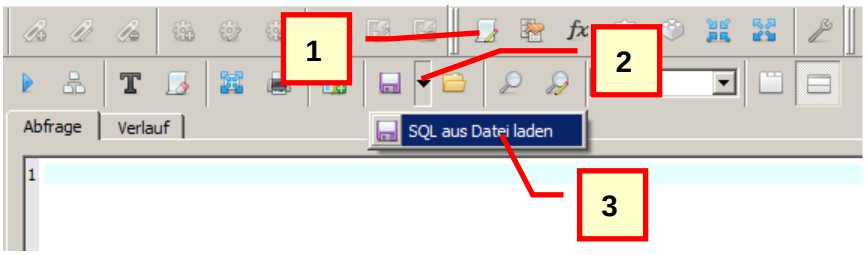
Links:
https://wiki.sqlitestudio.pl/index.php/User_Manual (online-Hilfe, Bedienungsanleitung)

3.1.7.1.3.9. SQL-Datenbanken importieren

ganze Datenbanken im SQL-Format werden am Besten direkt über sqlite importiert.
→ 3.1.7.1.4. SQL-Datenbanken in SQLite importieren

zuerst Erstellen einer Datenbank-Datei in SQLite-Studio

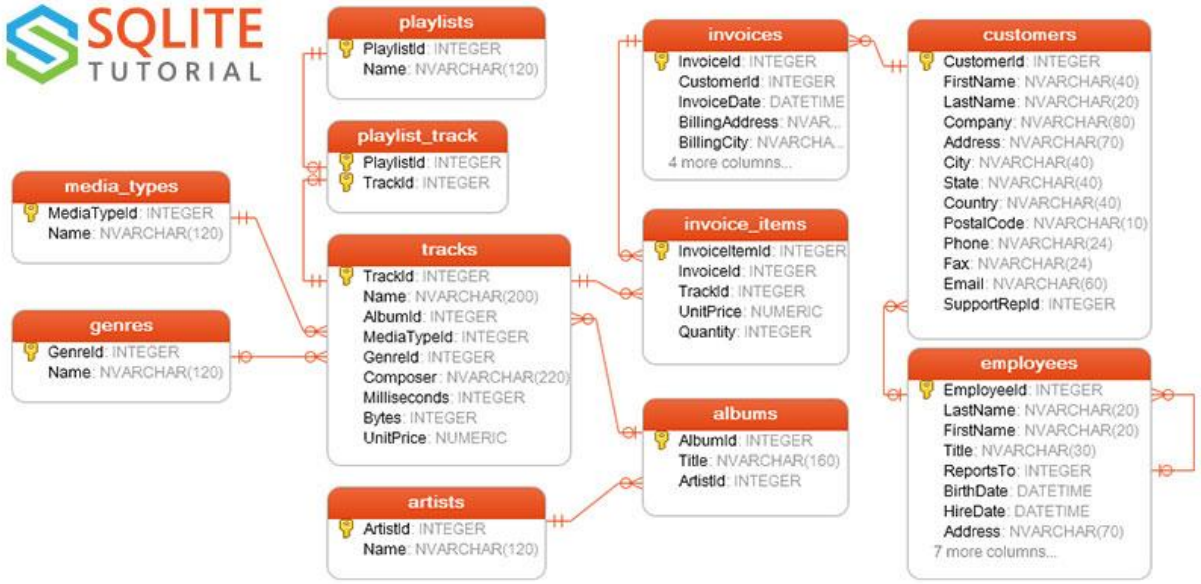
Scheinbar falsch! (auch schon falsche Symbolik) hier scheinbar speichern gemeint
Öffnen des SQL-Editors und dort neben dem Disketten-Symbol den Auswähler anklicken und "SQL aus Datei laden"
("Load SQL from file")



??? SQLiteStudio unterstützt nur Import von Daten in vorhandene Tabellen

Beispiel-Datenbanken:

chinook



Q: https://www.sqlitetutorial.net/sqlite-tutorial/sqlite-sample-database/

zum Integrieren mittel Kommandozeile ins Verzeichnis wechseln, in dem SQLite installiert ist
dort dann eingaben

…sqlite> **sqlite3 *Pfad*\chinook.db**

…sqlite> **.tables**

werden die importierten Tabellen angezeigt

SQLiteStudio - Tastatur-Kürzel (original (lassen sich bei den "Einstellungen" anpassen))

Aktion	Tastenkombination
Editor für Textwerte in Zellen	
Gewählten Text kopieren	Ctrl+C
Gewählten Text ausschneiden	Ctrl+X
Gewählten Text löschen	Del
Von der Zwischenablage einfügen	Ctrl+V
Wiederholen	Ctrl+Y
Rückgängig	Ctrl+Z
Ergebnisansicht	
Änderungen der Zellenninhalte speichern	Ctrl+Return
Kopiert Zelleninhalt(e) in die Zwischenablage	Ctrl+C
Markierte Datenzeile löschen	Del
Fügt einen leeren Wert in die selektierte(n) Zelle(n) ein	Alt+Backspace
Neue Datenzeile einfügen	Ins
Inhalt der markierten Zelle im separaten Editor öffnen	Alt+Return
Fügt Zelleninhalt(e) von der Zwischenablage ein	Ctrl+V
Änderungen der Zelleninhalte zurücknehmen	Ctrl+Backspace
Fügt den NULL Wert in die selektierte(n) Zelle(n) ein	Backspace
Ergebnisansicht (tabellarisch und Formular)	
Daten aktualisieren	F5
Zur Formularansicht wechseln	Ctrl+.
Zur tabellarischen Ergebnisansicht wechseln	Ctrl+,
Formularansicht der Ergebnisse	
Änderungen der aktuellen Zeile speichern	Ctrl+Return
Derzeitige Zeile löschen	Ctrl+Del
Springe zur ersten Zeile dieser Seite	Ctrl+Alt+PgUp
Neue Zeile einfügen	Ins
Springe zur letzten Zeile dieser Seite	Ctrl+Alt+PgDown
Springe zur nächsten Zeile	Ctrl+Alt+Right
Springe zur vorherigen Zeile	Ctrl+Alt+Left
Änderungen der aktuellen Zeile zurücknehmen	Ctrl+Backspace
Hauptfenster	
Statusfeld verbergen	Esc
Nächstes Fenster	Ctrl+PgDown
Konfigurationsdialog öffnen	F2
CSS Konsole öffnen	F11
Debug Konsole öffnen	F12
SQL Editor öffnen	Alt+E
Vorheriges Fenster	Ctrl+PgUp
Liste der Datenbanken	
Datenbank hinzufügen	Ctrl+O
Filter zurücksetzen	Esc
Gewählte Einträge kopieren	Ctrl+C
Gewählten Eintrag löschen	Del
Von der Zwischenablage einfügen	Ctrl+V
Schema aktualisieren	F5
Alle Schemas aktualisieren	Shift+F5
Alles auswählen	Ctrl+A
Neues Fenster zufügen	
Neuen Trigger zufügen	Ins
Gewählten Trigger löschen	Del
Gewählten Trigger bearbeiten	Return
Springe zum nächsten Reiter	Alt+Right
Springe zum vorherigen Reiter	Alt+Left
Aktualisiere View Triggerliste	F5

Aktion	Tastenkombination
Report-Verlaufsfenster	
Gewählten Eintrag löschen	Del
SQL Editor Eingabefeld	
Code-Assistenten anfordern	Ctrl+Space
Gewählten Text kopieren	Ctrl+C
Selektierten Textblock kopieren und unterhalb einfügen	Ctrl+Alt+Down
Selektierten Textblock kopieren und oberhalb einfügen	Ctrl+Alt+Up
Gewählten Text ausschneiden	Ctrl+X
Gewählten Text löschen	Del
Aktuelle Zeile löschen	Ctrl+D
Suche im Text	Ctrl+F
Nächster Fund	F3
Vorheriger Fund	Shift+F3
Format-Inhalte	Ctrl+T
Selektierten Textblock eine Zeile nach unten verschieben	Alt+Down
Selektierten Textblock eine Zeile nach oben verschieben	Alt+Up
Inhalte aus einer Datei laden	Ctrl+O
Von der Zwischenablage einfügen	Ctrl+V
Wiederholen	Ctrl+Y
Ersetze im Text	Ctrl+H
Inhalte in eine Datei speichern	Ctrl+S
Gesamten Editorinhalt auswählen	Ctrl+A
Toggle comment	Ctrl+/
Rückgängig	Ctrl+Z
SQL Editor-Fenster	
Abfrage ausführen	F9
Execute "EXPLAIN" query	F8
Tastatureingabe-Fokus in das obere SQL Editorfenster setzen	Alt+PgUp
Tastatureingabe-Fokus in das untere Ergebnisfenster setzen	Alt+PgDown
Wechsel von der aktuellen Datenbank zur nächsten in der Liste	Ctrl+Down
Wechsel von der aktuellen Datenbank zur vorherigen in der Liste	Ctrl+Up
Gehe zum nächsten Editor-Reiter	Alt+Right
Gehe zum vorherigen Editor-Reiter	Alt+Left
Tabellenfenster	
Neue Spalte zufügen	Ins
Neuen Index zufügen	Ins
Neue Tabellenbedingung zufügen	Ins
Neuen Trigger zufügen	Ins
Gewählte Spalte löschen	Del
Gewählten Index löschen	Del
Markierte Tabellenbedingung löschen	Del
Gewählten Trigger löschen	Del
Gewählte Spalte bearbeiten	Return
Gewählten Index bearbeiten	Return
Markierte Tabellenbedingung bearbeiten	Return
Gewählten Trigger bearbeiten	Return
Tabellendaten exportieren	Ctrl+E
Daten in die Tabelle importieren	Ctrl+I
Springe zum nächsten Reiter	Alt+Right
Springe zum vorherigen Reiter	Alt+Left
Aktualisiere Tabellenindexliste	F5
Aktualisiere Tabellenstruktur	F5
Aktualisiere Tabellentriggerliste	F5

Beispiel: Literatur-Datenbank (LiDaBa)

Hilfs-Quellen:
https://www.geschichte.tu-darmstadt.de/index.php?id=1123

Aufgaben-Beschreibung:

Für ein Forschungsgebiet und zu erstellende Dokumentationen (konkret: Facharbeit, Diplomarbeit, Hausarbeit, Bachelorarbeit, Promotion, …) soll eine Sammlung der verfügbaren Literaturquellen zusammengestellt werden. In der Datenbank soll später z.B. nach Autoren, Titeln, Schlüsselbegriffen, Verlagen, ISBN-Nummern usw. gesucht werden und die Daten strukturiert zusammengestellt werden.

Aufgaben:

1. *Analysieren Sie, welche Daten Sie für die Erstellung einer Literatur-Datenbank – wie oben beschrieben – benötigen!*
2. *Untersuchen Sie, welche Daten mit großer Wahrscheinlichkeit mehrfach in der Datenbank gespeichert werden!*
3. *Schlagen Sie ein ER-Modell für die LiDaBa vor auf A4-Blatt!*
4. *Diskutieren Sie die verschiedenen Vorschläge und leiten Sie aus den Vorschlägen eine (zumindestens theoretisch) optimale Lösung ab!*
5. *Erstellen Sie ein – für den Kurs gemeinsam akzeptiertes – Entity-Relationship-Diagramm in einem Zeichenprogramm! Exportieren Sie das Diagramm als nutzbare Graphik-Datei (z.B.: PNG. JPG, GIF oder in der Not BMP)*
6. *Legen Sie eine Projekt-Dokumentation an! Übernehmen Sie Aufgabenstellung und dokumentieren Sie die Entwicklungsschritte Ihrer Modell-Erstellung und –Umsetzung! (Spätestens am Ende der Unterrichtstunden immer kurz führen!)*
7. *Setzen Sie die Datenbank vollständig in Ihr favorisiertes Datenbank-System um!*

3.1.7.2. SQL-Datenbanken in SQLite importieren

SQLite bietet viele Möglichkeiten
Voraussetzung ist aber eine echte SQLite-Installation

`sqlite` *DatenbankDatei* `-init` *SQL_Datei*

`sqlite3.exe` *DatenbankDatei* `".read` *SQL_Datei*`"`

von einem SQLite-Prompt (siege dazu auch: → 3.1.7.3. Nutzung von SQLite mittels Kommandozeilen-Befehlen):

```
sqlite> .read datenbank.sql
```

oder

```
cat datenbank.sql | sqlite3 datenbank.db
```

! wichtig ist, dass die SQL-Statements mit einem Semikolon enden

3.1.7.3. Nutzung von SQLite mittels Kommandozeilen-Befehlen

Voraussetzung ist aber eine echte / lokale SQLite-Installation
SQLite-Studio bringt nur eine sqlite3.dll mit, diese enthält zwar die Funktionen, sie lassen sich aber so nicht starten

im Installations-Verzeichnis von SQLite gibt es die sqlite3.exe
diese ist das Kommandozeilen-Programm

direkter Start:
```
sqlite3
```

Start mit Datenbank
```
sqlite3
```

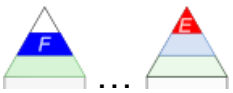
alle Befehle beginnen mit einem Punkt (.), deshalb auch dot-Kommando's genannt
dahinter ev. Optionen, die mit einem Bindestrich beginnen oder passende Argumente, wie z.B. die benutzte Datenbank usw. usf.
Anzeige der verfügbaren Kommando's mit
```
.help
```

zurück auf die "normale" Kommando-Ebene:
```
.exit
```

oder Beenden mit dem System-typischen End-of-File-Signals
Windows:
Linux: [Strg] + [D]

weiterführende Links:
https://www.sqlite.org/draft/cli.html (engl. Beschreibung zu vielen sqlite-Kommando's)

3.1.8. Datenbanken mit PROLOG



PROLOG ist eigentlich ein System zum Programmieren und Nutzen von Wissens-Datenbanken. Auch wenn der realtionale Ansatz nicht ganz fern ist, sind Datenbanken in PROLOG doch ganz anders strukturiert.
Datenbanken – besser Fakten-Sammlungen – bestehen aus Atomen. Ein Atom stellt dabei einen Datensatz dar. Der Atom-Name ist als Tabellen-Name zu verstehen.

Tabelle				
Attribut1	Attribut2	Attribut3	**…**	AttributN

tabelle(attribut1, attribut2, attribut3, …, attributN).

Es fällt gleich auf, dass es keine Attribut-Namen gibt. Nur die Daten und der Tabellen-Name ist in PROLOG sichtbar. Die Attribut-Namen können wir natürlich in Kommentaren mit angeben. Das erhöht die Verständlichkeit der Fakten-Sammlung.
Der größte Unterschied zu relationalen Datenbanken ist die fehlende Schlüssel-Struktur. PROLOG unterstützt keine Primär-Schlüssel. Diese können wir zwar nachbilden, da aber jeder Nutzer die Daten-Basis jederzeit beliebig ändern kann, ist die Einmaligkeit von Primär-Schlüseln nicht gewährleistet. Automatismen zum Vergeben von fortlaufenden oder einmaligen Schlüsseln gibt es in PROLOG nicht. Für die exakte Vergabe von Schlüsseln ist der Nutzer vollständig selbst verantwortlich.

```
%% Fakten → Schüler
%% schueler(Name,Vorname,weiblich,GebTag,GebMonat,GebJahr)
schueler("Bauer","Marie",true,10,5,2002).
schueler("Droste","Alex",false,1,08,2000).
schueler("Müller","Melanie",true,7,10,2001).
schueler("Müller","Guido",true,17,3,2002).
schueler("Zander","Erico",false,13,6,2001).
```

Eine **Projektion**, wie sie z.B. durch die SQL-Anweisung:

```
SELECT Name, GebJahr FROM Schueler;
```

realisiert werden würde, könnte in PROLOG durch:

```
?- schueler(Name, _, _, _, _, GebJahr).
```

dargestellt werden. Dadurch erhält man für die obige Daten-Basis:

Eine **Selektion**, die z.B. durch die folgende SQL-Anweisung repräsentiert werden würde:

```
SELECT * FROM Schueler WHERE GebJahr = 2001;
```

sollte dann in PROLOG durch:

?- schueler(Name, Vorname, Weiblich, GebTag, GebMonat, 2001).

oder:

?- schueler(Name. Vorname, Weiblich, GebTag, GebMonat, X), X = 2001.

ersetzbar sein. Das Ergebnis ist wenig überraschend:

Natürlich lassen sich auch **Projektion und Selektion** verknüpfen. Entsprechend zu:

SELECT Vorname FROM Schueler WHERE GebTag > 15;

ist der PROLOG-Konstrukt:

?- schueler(_, Vorname, _, X, _, _), X > 15.

Da ergibt sich für die obige "Tabelle":

Bei den Selektionen interessieren uns noch UND- bzw. ODER-Verknüpfungen, wie z.B.:

SELECT Name, Vorname FROM Schueler
WHERE Weiblich = false AND GebJahr = 2002

Sie sind ebenfalls Problem-los in PROLOG abbildbar:

?- schueler(Name, Vorname, false, _, _, 2002).

Für ODER sähe die Umsetzung von:

SELECT Name FROM Schueler WHERE GebMonat = 7 OR GebMonat = 8;

in PROLOG so aus:

?- schueler(Name, _, _, _, 7, _); schueler(Name, _, _, _, 8, _).

Bleibt die Verknüpfung von zwei Tabellen in einem Join. Als 2. Tabelle sei hier die Wort-Ersetzung der Monate gegeben.

```
%% Fakten → Monate
%% monat(Monat,MonatsName).
monat(1,'Januar').      monat(2, 'Februar').    monat(3, 'März').
monat(4,'April').       monat(5, 'Mai').        monat(6, 'Juni').
monat(7,'Juli').        monat(8, August').      monat(9, September').
monat(10,'Oktober').    monat(11, 'November').  monat(12, 'Dezember').
```

Für eine Anzeige wollen wir nun den Geburts-Monat als Text angegeben haben. Die SQL-Anweisung könnte so aussehen:

```
SELECT schueler.Name, monat.MonatsName
FROM INNER JOIN schueler, monat
WHERE schueler.GebMonat=monat.Monat
```

Das PROLOG-Äquivalent ließe sich so ausdrücken:

```
schueler(Name, _, _, _, _X, _), monat(_X,GebMonat).
```

Ein weiteres Problem taucht auf, wenn wir unsere Daten-Basis ändern wollen. Die Daten werden beim Konsultieren in den Speicher geladen und sind hier unveränderlich. Erst nach einem Ändern im Quell-Text und erneutem Konsultieren besitzen wir eine geänderte Daten-Basis im Speicher. Für rein lesenden Zugriff auf Daten ist das ok.
Es gibt aber die Möglichkeit in PROLOG dynamische Strukturen anzulegen. Dazu müssen die Atome entsprechend deklariert werden.

```
:- dynamic funktor/stelligkeit.
```

In unserem Schüler-Beispiel würde das so aussehen:

```
:- dynamic schueler/6
```

Die Angabe der Atome bleibt so, wie oben aufgezeigt. Mit dem **listing**-Befehl können wir uns den aktuellen Daten-Stapel (praktisch wohl eine Listen-Struktur) ansehen:

```
?- listing(schueler/6).
```

So ergibt sich die bekannte Daten-Basis:

PROLOG stellt noch weitere Funktionen bereit, die dem Handling der Daten-Struktur dienen:

Funktionen für dynamische Daten-Strukturen (in PROLOG)

- **asserta(fakt).** fügt den Fakt an den Anfang der dynamischen Daten-Struktur ein
- **assertz(fakt).** hängt den Fakt an das Ende der dynamischen Daten-Struktur an
- **retract(fakt).** löscht den angegebenen Fakt aus der dynamischen Daten-Struktur (quasi Löschen eines Datensatzes)
- **retractall(fakt).** löscht alle Fakten zum angegebenen Funktor (quasi Löschen der gesamten Tabelle)

Ergänzend kann eine eigene Regel zum Einfügen in die Daten-Struktur definiert werden:

```
einfuegen(N,V,G,T,M,J):- assertz(schueler(N,V,G,T,M,J)).
```

Diese macht aber nur dann Sinn, wenn der Quelltext für noch mehr Personen lesbar sein soll, oder die interne Fakten-Struktur gestaffelt oder stärker strukturiert ist.

Beispiel für strukturierte Fakten:

```
einfuegen(N,V,G,T,M,J):- assertz(schueler(N,V,G,gebDat(T,M,J))).
```

```
abfrage_NamensListe:- schueler(Name, Vorname, _, _, _, _),
                      write(Vorname), write(' '), write(Name), nl, fail.
```

Interessant ist die Verwendung von fail am Ende der Regel-Definition. Dieses fail sorgt dafür, dass die Daten-Basis nach weiteren passenden Fakten durchsucht wird und auch diese der Anzeige (/ der Regel) zur Verfügung gestellt werden. Fail schlägt immer fehl und erzwingt damit den nächsten Durchlauf.
Aufgerufen wird diese Regel durch:

```
?- abfrage_NamensListe.
```

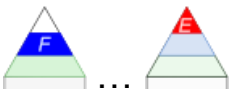
und liefert als Ergebnis eine übersichtliche Liste:
Nun fehlt noch eine Überschrift über die Liste. Auch das ist kein Problem:

```
abfrage_NamensListe:- write('Klassenliste'),nl,
                      ( schueler(Name, Vorname, _, _, _, _),
                        write(Vorname), write(' '), write(Name), nl, fail
                      ).
```

Aufgaben:

1.
2. *Was würde eigentlich passieren, wenn man statt fail das Schlüsselwörtchen cut in die Regel einbaut? Stellen Sie Voraussagen auf! Probieren Sie es aus!*
3. *Setzen Sie unsere Test-Datenbank (Tabellen "Kontakte" und "Institutionen" einschließlich Verknüpfung) in PROLOG um!*

3.1.8.1. fortgeschrittene Datenbank-Techniken mit PROLOG



Hier sind jetzt nicht nur die Datenabk-Operationen etwas anspruchsvoller, sondern auch die PROLOG-Umsetzungen.
Zuerst wollen wir bestimmte Attribute aus der Daten-Basis heausziehen und sie in einer Liste für die weitere Verwendung bereitstellen.
Als Beispiel wählen wir hier mal eine gewünschte Liste aus den Vornamen der Mädchen.
Das Prädikat zum Finden aller Passungen ist findall/3. Das erste Argument ist die Variable für das rauszuziehende Element gefolgt von dem Ausdruck, der ausgewertet werden soll. Als 3. Argument müsen wir noch eine Variable angeben, in der die Ergebnis-Liste aufgebaut werden soll.

```
?- findall(Elem, schueler(_, Elem, _, _, _, _), Liste).
```

Dieser Ausdruck liefert uns zuerst einmal alle Vornamen. Mit können wir uns die Liste auch ansehen:

Damit eventuelle Bedingungen eingebaut werden können, muss das mittlere Argument um die Bedingung (weiblich) erweitert werden. Der gesamte (mittlere) Ausdruck muss in eine Klammer gefasst werden, da die Gesamt-Anzahl der Argumente für findall natürlich bei drei bleiben muss.

```
?- findall(Elem, (schueler(_, Elem, W, _, _, _), W = true), Liste).
```

So erhalten wir die gewünschte Liste:

Links:
https://www.tinohempel.de/info/info/prolog/datenbank.htm
https://www.philipphauer.de/info/info/datenbanken-in-prolog/

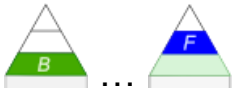
3.1.9. Datenbanken mit Snap!

Hin und wieder taucht ein Artikel oder ähnliches zu diesem Thema auf. Praktisch ist Snap aber kein für uns gebräuchliches Datenbank-Management-System. Dadurch fliegt es an dieser Stelle aus der Kapitelfolge heraus.
Beim Programmieren von Datenbanken bietet es aber interessante Möglichkeiten. So kann vor allem den Abfrage- / View-Bereich sehr schön abarbeiten.
Die Verbindung liegt hier zwischen einer visuell ansprechenden Block-orientierten Programmierung und der Nutzung einfacher SQL-Abfragen.

Die Block-Orientierung hilft sehr gut beim Zusammenstellen von Abfragen. Diese können immer direkt ausgeführt und geprüft werden. Übergibt man die Abfrage-Ergebnisse einer Variablen, dann können danach viele weitere Operationen darüber ausprobiert werden.
Super praktisch sind die einfachen Verknüpfungs-Möglichkeiten einfacher Programmiertechniken mit Datenbank-Abfragen. Da wird das Nutzungs-Potential von Programmierung und Datenbanken sehr schön sichtbar.
Auf einige Programmier-Möglichkeiten bezüglich von SQL-Abfragen gehen wir später noch etwas ausführlicher ein (→ 6.5. Datenbank-Programmierung mit Snap!).
Ein recht interessantes Projekt "Supermarkt" kann im Skript weiterverfolgt werden. Natürlich bietet sich hierfür auch der Kurs "Informatik für Abfänger (…)" auf → https://open.sap.com an.
In der dritten Woche wird in den Video's 5 und 6 auf Datenbanken eingegangen.
Abgelaufene Kurse können hier im Selbststudium nachverfolgt werden.

4. Datenbanken – fortgeschrittene Nutzung

4.0. Relationen-Algebra / Relationen-Kalküle



auch relationale Algebra
nicht richtig wäre Relationen-Algebra, da diese zur abstrakten Algebra gehört und sich mit BOOLEschen Ausdrücken beschäftigt
im Datenbank-Jargon wird das manchchmal nicht sauber getrennt
unter einer Algebra versteht man ein Zusammensetzungs- bzw. Teilungs-System
Verknüpfung mathematischer Strukturen; Lehre von den Gleichungen
mehr Rechen-Vorschriften
Verallgemeinerung der Arithmetrik auf Zahlen und Zeichen, die Zahlen repräsentieren unter Verwendung definierter (arithmetrischer) Regeln / Vorschriften
z.B.: BOOLEsche Algebra

Definition(en): Algebra
Eine Algebra ist eine (abgeschlossene) Menge von Regeln und Operationen zur Verarbeitung / Verknüpfung / Umwandlung von Zahlen, Variablen und anderen mathematischen Objekten.
.

Kalküle (bedeutet soviel wie Berechnungen bzw. Überlegungen)
System von Regeln aus der Logik und Mathematik zur systematischen Lösung von mathematischen Problemen
aus gegebenen Axiomen oder Aussagen werden weitere Aussagen gewonnen
ab- bzw. einschätzende Berechnungen (Wahrscheinlichkeits-behaftet; keine sichere / eindeutige Ergebnisse / Lösungen)
mehr Lösungs-Vorschriften / Sammlung von Regeln und deren Notation
z.B.: Notierungen mit logischen Symbolen

Tupel-Kalkül
Bereichs-Kalkül (Domänen-Kalkül)

auch nur realtionale Algebra genannt
betrachtet als Mengen-Element ein Tupel (Datensatz, Zeile)
je nach Anzahl der Spalten sprechen wir vom n-Tupel, also gilt für eine Tabelle mit 5 Spalten das 5-Tupel (auch: Quin-Tupel) als Mengen-Element
prozeduale Sprache

Definition(en): relationale Algebra / Relationen-Algebra

Die Relationen-Algebra ist eine (abgeschlossene) Menge von Regeln und Operationen zur Verarbeitung / Verknüpfung / Umwandlung von Tabellen (Relationen).
Das Ergebnis der Regeln und Operationen ist wiederum eine Tabelle.

.

Benennung der Tupel:

num. Schr.	Benennung
0-Tupel	leeres Tupel
1-Tupel	Singel
2-Tupel	Dupel; geord. Paar
3-Tupel	Tripel
4-Tupel	Quadrupel

num. Schr.	Benennung
5-Tupel	Quintupel
6-Tupel	Sextupel
7-Tupel	Septupel
8-Tupel	Oktupel
9-Tupel	Nonupel

num. Schr.	Benennung
10-Tupel	Dekupel
20-Tupel	
100-Tupel	Centupel

E.F. CODD (1923 - 2003) definierte Algebra für Relationen (1970), die auf 6 Operatoren basiert
die Zahl der Operatoren wird auch mal mit 5 oder 8 angegeben, insgesamt ist die Struktur und Zuordnung sehr undurchsichtig
aus einer oder mehrerer Tabellen (Relation(en)) entsteht immer wieder eine Tabelle (, die aber auch leer sein darf!)
dabei sind 5 primitive Operatoren und drei komplexe Operatoren beschrieben worden
heute weiss man, dass bestimmte – wünschenswerte – Verknüpfungen mit den CODDschen Operatoren nicht realisert werden können
deshalb wurde die Relationen-Algebra später auch immer wieder erweitert und erweitert
selbst CODD hat seine Algebra-Regeln und –Operatoren mehrfach korrigiert, dadurch fällt es schwer eine saubere Trennung vorzunehmen (Hierzu wäre eine tiefgreifende historische Aufarbeitung der Entwicklungs-Geschichte der CODDschen Algebra notwendig.).

(Mengen-)Operationen auf Relationen nach CODD

- **Projektion (π)** unär
- **Selektion (σ) (Restriktion)** unär
- **Kreuz-Produkt (x) (Kartesisches Produkt)** binär
- **Vereinigung** binär
- **Differenz (Δ)** binär
- **Umbenennung (ρ)** unär

weitere komplexe Operationen, die sich auf die Grund-Operationen zurückführen lassen:

- **Join (⋈)**
  (Verbund)

- **symmetrische Differenz**

- **Schnittmenge**
  (Intersection)

- **Division**

Erweiterung der klassischen Algebra um …:

- **Null-Werte**

- **Gruppierungs-Operatoren**

- **Aggregat-Funktionen**

Symbole für spezielle Join's
??? ⋊ **Semi-Join** ⋉

- **Identität**
  identische Relation — gemeint ist hier die Objekt-Gleichheit, nicht die Gleichheit der Strukuren und Inhalte

- **Inversion**
  Transponierung
  inverse Relation
  transponierte Relation — Spiegeln einer Matrix / Tabelle entlang ihrer Diagonale

- **Relationen-Produkt**

leere Relation ø
Relation, bei der alle Elemente miteinander in Beziehung stehen: 1 := A x A

minimale Menge der Operationen (Grundoperationen)

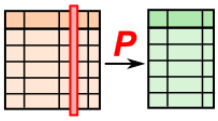
- **Projektion**
- **Selektion**
- **Kreuzprodukt**
- **Vereinigung**
- **Differenz**
- **Umbenennung**

mit diesen lassen sich alle Auswahlen treffen, sei entsprechen auch dem Algebra-System von CODD

einige SQL-Operationen (z.B. GROUP BY oder HAVING) lassen sich durch algebrische Operationen nicht abbilden, deshalb werden manchmal einige Erweiterungen zur relationalen Algebra hinzugefügt → mehr theoretisches Problem

Projektion

Ergebnis sind neue Tupel mit ausgewählten Attributen

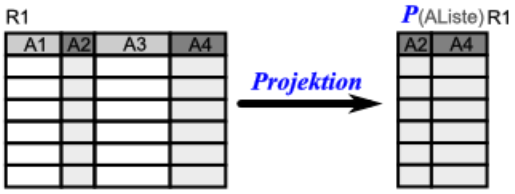


Projektionen werden in Termen mit **P** oder π verschlüsselt.

unäre Operation, die also nur mit einer Tabelle / Relation arbeitet

R1 sei eine Relation
mit den Attributen A1, A2 usw. usf.
notieren wir dies in der Form:

R1={A1,A2,…,An}



Operatoren (Funktionen) notieren wir im Folgenden mit fetten Groß-Buchstaben oder griechischen Klein-Buchstaben bzw. den definierten Symbolen

R1

A	B	C
a	b	c
d	e	f
g	h	i

R2 = R1[A,C]

A	C
a	c
d	f
g	i

eine Projektion der Attribute A und C aus der nebenstehenden orangenen (Quell-)Tabelle R1 führt z.B. zur grünlichen (Ergebnis-)Tabelle R2, die nur die Attribute A und C enthält.

R2 = **P**(R1[A,C]) oder: R2 = π(R1[A,C])

In R3 möchten wir nur die Spalte B haben. Die passende Projektion sähe dann so aus:

od. z.B.:

R3= R1[B]

B
b
e
h

R3 = **P**(R1[B]) oder: R3 = π(R1[B])

In einigen Büchern usw. findet man auch die folgenden Notierung:

R3 = $\mathbf{P}_B$(R1) oder: R3 = π_B(R1)

oder eben für die mehr-attributige Projektion zu R2:

R2 = $\mathbf{P}_{A,C}$(R1) oder: R2 = $\pi_{A,C}$(R1)

Diese Notation ist bei einfacheren Ausdrücken übersichtlicher. Wenn aber die Anzahl der Attribute steigt und diese auch längere Namen haben, dann wird die Notation über mehrere Zeilen sehr schnell unübersichtlich. Wir bleiben deshalb bei obiger Notation im immer gleichgroß geschriebenen Symbolen.
Projektionen sind also praktisch immer Spalten-Auswahlen (Attribut-Beschränkung)

SQL-Repräsentation (→ 5.1.11.1. Projektion (Abbildung):):
SELECT *spalteX* **{ ,** *spalteY* **}** **FROM** *tabelle*;

Hier ein **wichtiger Hinweis!**:
Der SQL-Befehl SELECT steht umgangssprachlich für "Auswahl" und nicht für "Selektion". Selektionen sind bestimmte algebraiische Operationen!

Aufgaben:

1. Geben Sie die folgenden Projektionen als Ergebnis-Tabellen an!

a) R4 = P(R1[A]) ***b)*** R5 = P(R1[B,C]) ***c)*** R6 = P(R2[C])

2. Gegeben sind die drei folgenden Tabellen.

R1

A	B	C
I	II	III
IV	V	VI
VII	VIII	IX

R2

D	E
z	y
x	w
v	u
t	s
r	q
p	o

R3

F	G	H	J
11	12	13	14
21	22	23	24

a) Geben Sie die folgenden Projektionen an!

aa) R4 soll die Spalten A und B von R1 enthalten

ab) die neue Relation R5 soll aus den Spalten D und E von R2 bestehen

ac) in die Relation R7 werden die Attribute H, G und F von R3 projiziert

b) Geben Sie für die Projektion aus der Teil-Aufgabe ac) die Ergebnis-Tabelle an!

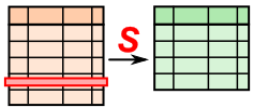
c) Handelt es sich bei den folgenden Ausdrücken um eine Projektion? Begründen Sie jeweils Ihre Entscheidung!

ca) R6 = P(R3[F,G]) ***cb)*** R5 = P(R1[B,A,A])

cc) R8 = P(R2[E],R3[G,J]) ***cd)*** R9 = P(R3[H,G,J,F])

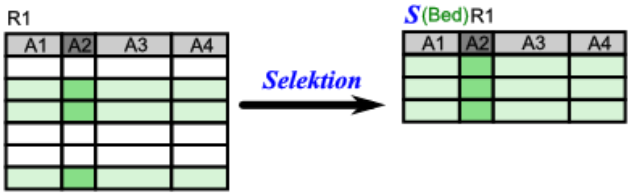
Selektion / Restriktion

auch als Auswahl (der Datensätze) verstanden



R WHERE Bedingung
Ergebnis beinhaltet alle Tupel, die die Bedingung(en) erfüllen

Selektionen werden in Termen durch
S oder σ (griech.: sigma) verschlüsselt.



R2 = **S**(R1[B=b]) oder: R2 = σ(R1[B=b])

Genauso könnte R3 durch andere Projektion gebildet werden

R3 = **S**(R1[A<g]) oder: R3 = σ(R1[A<g])

(bei Annahme geordneter Attribut-Werte von a nach j)

praktisch also Zeilen- bzw. Datensatz-Auswahl (bezüglich von Bedingungen) und Spalten-Auswahl (Einschränkung der Attribute)
die algebraiischen Ausdrücke werden ja üblicherweise von innen nach außen abgearbeitet

R1

A	B	C
a	b	c
d	e	f
g	b	h
i	j	b
b	b	b

od. z.B.:

R2= R1[A,B=b,C]

A	B	C
a	b	c
g	b	h
b	b	b

R3= R1[A<g,B,C]

A	B	C
a	b	c
d	e	f
b	b	b

SQL-Repräsentation (→ 5.1.11.2. Selektion (Auswahl):):
SELECT * FROM *tabelle* **WHERE *bedingung*;**
bedingung kann dabei ein einfacher oder komplexer – z.B. zusammengesetzter – (SQL-)Ausdruck sein

Hier ein **wichtiger Hinweis!**:
Der SQL-Befehl SELECT steht umgangssprachlich für "Auswahl" und nicht für "Selektion". Selektionen sind die gerade besprochenen algebraiische Operationen, bei denen eine Zeilen- / Datensatz-Auswahl erfolgt!

Aufgaben:

1. *Prüfen Sie, ob die folgenden Selektionen zulässig sind und welche Ergebnisse entstehen!*

   a) S(R1[A>a]) *b)* S(R1[B=b,C=b]) *c)* S(R2[A<u,B=A,C>a])

2. *Gegeben sind die drei folgenden Tabellen.*

R1

A	B	C
I	II	III
IV	V	VI
VII	VIII	IX

R2

D	E
z	y
x	w
v	u
t	s
r	q
p	o

R3

F	G	H	J
11	12	13	14
21	22	23	24

a) Geben Sie die folgenden Selektionen an!

aa) R4 soll die Datensätze aus R1 enthalten, bei denen C>II ist

ab) die neue Relation R5 soll aus R2 alle Datensätze enthalten, bei denen die Werte aus der Spalte E größer ist als die Werte aus E

ac) in die Relation R7 werden die Zeilen aus R3 herausgesucht, für die F gößer 11 und H kleiner gleich 23 ist

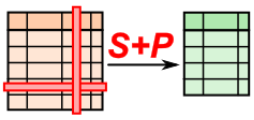
b) Geben Sie für die Selektion aus der Teil-Aufgabe ac) die Ergebnis-Tabelle an!

c) Handelt es sich bei den folgenden Ausdrücken um eine Selektion? Begründen Sie jeweils Ihre Entscheidung!

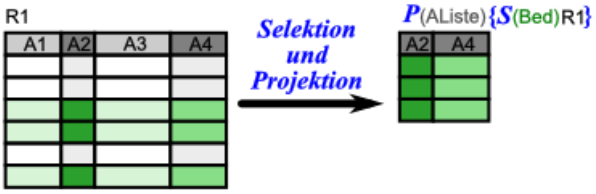
ca) R6 = S(R3[F=11]) *cb)* R5 = R(R1[B,A,A])

cc) R8 = S(R2[E>u],R3[G>10]) *cd)* R9 = S(R3[H=13,G=12,J=14,F=24])

Verbindung von Selektion und Projektion (bezüglich einer Tabelle)



serielle Abarbeitung von zwei unären Operation
kann als eine **unäre** Operation betrachtet werden



R2 = **P**(**S**(R1[B=b])[B,C]) oder: R2 = π(σ(R1[B=b])[B,C])

Es kommt zu einer Schachtelung, die zur Veranschaulichung durch verschiedene Farben gekennzeichnet wird:

R2 = **P**(**S**(R1[B=b])[B,C]) oder: R2 = π(σ(R1[B=b])[B,C])

Dabei ergibt die "blaue" Selektion eine imaginäre Zwischen-Tabelle Ri,

Ri = **S**(R1[B=b]) oder: Ri = σ(R1[B=b])

auf die dann die "rote" Projektion angewendet wird.

R2 = **P**(Ri[B,C]) oder: R2 = π(Ri[B,C])

R1

A	B	C
a	b	c
d	e	f
g	b	h
i	j	b
b	b	b

od. z.B.:

R2= R1[B=b,C]

B	C
b	c
b	h
b	b

R3= R[A<g]

A
a
d
b

Für die andere Ziel-Tabelle R3 kann die folgende Projektion genutzt werden:

R3 = **P**(**S**(R1[A<g])[A]) oder: R3 = π(σ(R1[A<g])[A])

(bei Annahme geordneter Attribut-Werte von a nach j)
sachlich ist die algebraische Abarbeitungs-Reihenfolge von Selektion und Projektion gleichwertig, so dass auch gilt:

R2 = **P**(**S**(R1[B,C][B=b]) oder: R2 = σ(π(R1[B,C] [B=b])

bzw.:
R3 = **P**(**S**(R1[A][A<g])) oder: R3 = σ(π(R1[A][A<g]))

An dieser Stelle zeigen wir noch einmal die Notation mit indizierten Attributen. Als Beispiel betrachten wir R2 von oben:

$R2 = PS_{R1B=b,C}$

praktisch also Zeilen- bzw. Datensatz-Auswahl (bezüglich von Bedingungen) **und** Spalten-Auswahl (Einschränkung der Attribute)
die algebraischen Ausdrücke werden ja üblicherweise von innen nach außen abgearbeitet

SQL-Repräsentation (→ 5.1.11.2. Selektion (Auswahl):):
SELECT *spalteX* **{ ,** *spalteY* **}** **FROM** *tabelle* **WHERE** ***bedingung*;**

die SQL-Systeme nehmen meist zuerst die Selektion und dann die Projektion vor, was bezogen auf das SQL-Statement bedeutet, dass zuerst der hintere Teil (z.B. WHERE) bearbeitet wird und dann der vordere (SELECT)

Aufgaben:

1.

2. Gegeben sind die drei folgenden Tabellen.

R1

A	B	C
I	II	III
IV	V	VI
VII	VIII	IX

R2

D	E
z	y
x	w
v	u
t	s
r	q
p	o

R3

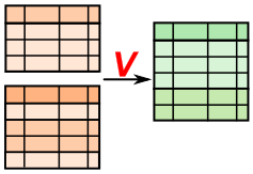
F	G	H	J
11	12	13	14
21	22	23	24

3.

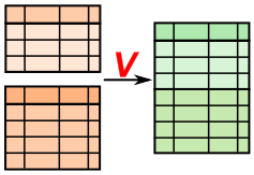
Kommen wir nun zu den **binäre**n Operationen. Diese Operationen arbeiten immer mit zwei Relationen. Dabei dürfen beide Relationen bei einzelnen Oprationen auch gleich sein. In einigen Fällen ist die Reihenfolge der Verwendung der beiden Tabellen wichtig (z.B. Differenz). Im Normal-Fall bearbeiten wir aber zwei unterschiedliche Tabellen. Ziel ist es, Informationen neu zu kombinieren oder zu erweitern.

Vereinigung / Union Join

R1 UNION R2
fügt die Zeilen von zwei Tabellen mit gleicher Spaltenzahl zu einer neuen Tabelle zusammen
die Namen der Spalten müssen nicht gleich sein, die Datentypen müssen aber identisch sein!



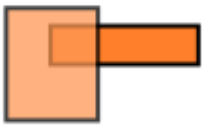
doppelte Zeilen werden automatisch unterdrückt, ist dies nicht gewünscht, dann muss UNION ALL benutzt werden



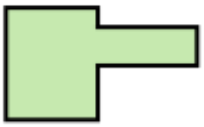
Die "einfache" Vereinigung enthält also praktisch auch noch Selektionen, was bei der vollständigen Vereinigung unterbleibt.

Mengen-mäßig betrachtet ist die Vereinigung die Menge, die alle Elemente aus den (Ursprungs-)Mengen M1 und M2 (orange). Die Ergebnis- bzw. Vereinigungsmenge ist grün ausgefüllt.

In der Mengen-Lehre ist eine Vereinigung, wie wir sie gerade mit UNION ALL vorgestellt haben, nicht möglich. In Mengen kommen – per Definition – Elemente nur einmalig vor.



Original- / Ursprungs-Mengen



Vereinigungs-Menge

Bei Relationen müssen die Datensätze (Zeilen, Tupel) als Element der Menge betrachtet werden. Entsprechend tauchen in der Ergebnis-"Menge" nun alle Datensätze einmal auf. Die doppelten (rot gekenzeichnet) aus der Menge 2 (hier R2) werden nicht noch einmal in die Ergebnis-Menge übertragen.

R1

A	B	C
a	b	c
d	e	f
g	h	i

R2

D	E	F
d	e	f
b	c	d
g	h	i
j	k	l
m	n	o

R3= R1 V R2

A	B	C
a	b	c
d	e	f
g	h	i
b	c	d
j	k	l
m	n	o

SQL-Repräsentation:
SELECT * FROM tabelle1 UNION SELECT * FROM tabelle2;

Aufgaben:

1.

2. Gegeben sind die folgenden Tabellen mit gleichen Datentypen in allen Spalten. Erstellen Sie die Vereinigungen von a) R1 und R2 als R4 ; b) R2 und R1 als R5; c) R2 und R3 als R6 und von d) R3 und R1 als R7!

R1

A	B	C
1	2	3
4	5	6
7	8	9
10	11	12

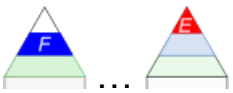
R2

D	E	F
1	2	3
2	3	4
3	4	5
4	5	6
3	8	9
10	12	13

R3

G	H	I	J
1	2	3	4
4	5	6	7
7	8	9	10

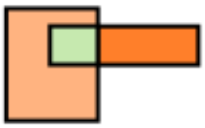
3. Erstellen Sie die vollständige Vereinigung (UNION ALL) für die Tabellen R2 und R1! (Probleme mit Schlüsseln übergehen wir hier!)



Schnitt(menge) / Durchschnitt / Intersection

R1 INTERSECT R2
Ergebnis sind nur die Tupel, die in beiden Tabellen vorkommen

In der Relationen-Algebra betrachten wir ja die Tupel als Mengen-Elemente. Somit müssen beim Ermitteln der Schnitt-Menge die Tabellen gleichviele Spalten enthalten.
Es werden alle Tupel ausgewählt, die sowohl so in der Realtion R1 als auch in der Relation R2 vorkommen.


Schnitt(-Menge)

R1

A	B	C
a	b	c
d	**e**	**f**
g	**h**	**i**

R2

D	E	F
d	**e**	**f**
b	c	d
g	**h**	**i**
j	k	l
m	n	o

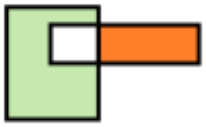
R3= R1 ∧ R2

A	B	C
d	e	f
g	h	i

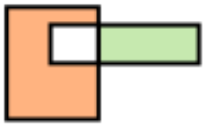
SQL-Repräsentation:
SELECT * FROM tabelle1 INTERSECT SELECT * FROM tabelle2;
SELECT * FROM tabelle1 WHERE spalte IN (SELECT spalte FROM tabelle2);

Differenz / Minus

R1 MINUS R2
Ergebnis sind alle Tupel aus R1, die nicht in R2 vorkommen
(aus Tabelle R1 werden alle Tupel aus R2 gelöscht)



Differenz(-Menge)
(M1 – M2)



Differenz(-Menge)
nach Operanden-Tausch
(M2 – M1)

R1

A	B	C
a	b	c
d	e	f
g	h	i

R2

D	E	F
d	e	f
b	c	d
g	h	i
j	k	l
m	n	o

R3 = R1 \ R2

A	B	C
a	b	c

R4 = R2 \ R1

A	B	C
b	c	d
j	k	l
m	n	o

SQL-Repräsentation:
SELECT * FROM tabelle1 MINUS SELECT * FROM tabelle2;
SELECT * FROM tabelle1 WHERE spalte NOT IN (SELECT spalte FROM tabelle2);

??? symetrische Differenz

Division / Quotient

R1 DEVIDEBY R2
Ergebnis enthält alle Attribute aus R1, die nicht in R2 vorkommen und die Tupel aus R1, die hinsichtlich eines gemeinsamen / gleichen Attributs-Wertes auch in R2 vorkommen

Nicht zu verwechseln mit der nummerischen Division von zwei Zahlen. Diese kann in SQL-Statement's mit einem / erledigt werden.
Hier ist die Division zweier Mengen bzw. der Quotient beider gemeint.

abgeleitete Operation:
R1 / R2 = P(R1) – P((P(R1) x R2) – R1)

gesucht sind bei der Division die (Teil-)Datensätzen aus R1, die für jeden Datensatz in R2 einen Eintrag haben

R1

A	B	C	D
a	b	c	d
e	**f**	g	**h**
i	j	k	l
a	m	c	n
e	**o**	g	**p**
i	q	k	r

R2

B	D
f	**h**
o	**p**

R3=
R1 / R2

A	C
e	g

SQL-Repräsentation:
SELECT * FROM tabelle1 DEVIDEBY SELECT * FROM tabelle2;

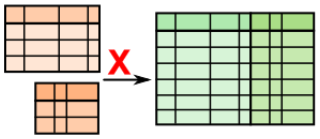
Aufgaben:

1.
2.
3.

Kreuz-Produkt / kartesisches Produkt

auch Cross Join

Kombination jeder Zeile der einen Tabelle (**R1**) mit jeder Zeile der anderen (**R2**)



Ergebnis besteht aus allen möglichen Tupel-Kombinationen von **R1** und **R2**
es entstehen sehr große Tabellen (neue_Zeilenzahl = ZeilenzahlTab1 * Zeilenzahl_Tab2)
Verwendung meist fraglich → ev. für DataMining
oft als Daten-Quelle für weitere Operationen benutzt

die etwas dunkleren grünen Felder in der Ergebnis-Tabelle R3 sollen nur die Wiederverwendung der ursprünglichen Datensätze in der Kombination zeigen

R1

A	B	C	D
a	b	c	d
e	f	g	h
i	j	k	l

R2

E	F	G
1	2	3
4	5	6

R3 = R1 X R2

A	B	C	D	E	F	G
a	b	c	d	1	2	3
e	f	g	h	1	2	3
i	j	k	l	1	2	3
a	b	c	d	4	5	6
e	f	g	h	4	5	6
i	j	k	l	4	5	6

SQL-Repräsentation:
SELECT * FROM *tabelle1* **CROSS JOIN** *tabelle2*;
SELECT * FROM *tabelle1, tabelle2*;

Aufgaben:

1. *In den Tabellen wurden vom Datenbank-Administrator die Spalten A und E als Schlüssel festgelegt. Welche Spalte(n) sind in der Tabelle R3 Schlüssel? Erklären Sie!*
2. *Prüfen Sie, ob R4 = R2 X R1 gleich der Tabelle R3 ist!*
3. *Was ändert sich in der Tabelle R3, wenn ein ev. unbedachter Datenbank-Administrator*
   a) die Spalten D und G als Schlüssel festgelegt hat
   b) die Schlüsselwerte in den Schlüsselspalten A und E gleich belegt hat.
4. *Gegeben sind die folgenden drei Tabellen. Stellen Sie die kartesischen Produkte für R1 X R3, R2 X R3 und R1 X R2 X R3 auf!*

R1

A	B	C
I	II	III
IV	V	VI
VII	VIII	IX

R2

D	E
z	y
x	w
v	u
t	s
r	q
p	o

R3

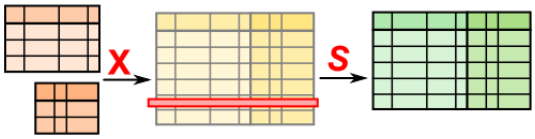
F	G	H	J
11	12	13	14
21	22	23	24

3.

Verbund / Verknüpfung (Join)

hintereinander ausgeführte Operationen "kartesisches Produkt" und "Selektion"

aus der Kreuz(-produkt)-Tabelle wird durch Selektion eine Auswahl von Datensätzen getätigt



als Sonderfall des Verbundes / Join's gilt der Equal-Join (Gleich-Verbund; → innerer / äquivalenter Verbund / Inner Join / Equivalent Join), bei dem in den Datensätzen der Kreuz-(produkt)-Tabelle nach bestimmten übereinstimmenden Feldern gesucht wird und diese dann eleminiert werden

R3 ist Kreuz-Produkt als Zwischen-Ergebnis und vorrangig gelblich dargestellt

R1

A	B	C	D
a	b	c	d
e	f	g	h
i	j	k	l

R2

E	F	G
1	2	3
4	5	6

R3 = R1 X R2

A	B	C	D	E	F	G
a	b	c	d	**1**	2	3
e	f	g	h	**1**	2	3
i	j	k	l	**1**	2	3
a	b	c	d	4	5	6
e	f	g	h	4	5	6
i	j	k	l	4	5	6

Selektion dahingehend, dass in E ein Wert z.B. kleiner als 3 (passende Werte sind rot und fett gekennzeichnet, wobei eben nur der 2. Datensatz die Bedingung erfüllt und damit aussortiert wird (rötlich unterlegt))

R4 = R3[A,B,C,D,E<3,F,G]

A	B	C	D	E	F	G
a	b	c	d	4	5	6
e	f	g	h	4	5	6
i	j	k	l	4	5	6

Aufgaben:

1. *Erstellen Sie aus R1 und R2 eine Relation R5, die als kartesisches Produkt keine Datensätze enthält, bei denen F mit 2 belegt ist!*
2. *Gibt es eine Verbund-Relation aus R1 und R2, die in G Werte über 8 enthält? Erläutern Sie Ihre Antwort!*

innerer / äquivalenter Verbund / Inner Join / Equivalent Join

auch Equi-Join; Sonderfall des "normalen" Verbundes (Join, → Verbund / Verknüpfung (Join))

R1 **INNER JOIN** R2
Ergebnis sind alle möglichen Kombinationen der Tupel aus beiden Tabellen, die bestimmte Attribut(-Werte) gleich haben
(Verknüpfung über Fremdschlüssel-Primärschlüssel-Beziehung

R3 ist Kreuz-Produkt als Zwischen-Ergebnis und vorrangig gelblich dargestellt

R1

A	B	C	D
a	b	c	d
e	f	g	h
i	j	k	l

R2

E	F	G
x	b	c
y	f	c
z	j	c

R3 = R1 X R2

A	B	C	D	E	F	G
a	**b**	c	d	x	**b**	c
e	f	g	h	x	b	c
i	j	k	l	x	b	c
a	b	c	d	y	f	c
e	**f**	g	h	y	**f**	c
i	j	k	l	y	f	c
a	b	c	d	z	j	c
e	f	g	h	z	j	c
i	**j**	k	l	z	**j**	c

jetzt Selektion dahingehend, dass der Wert in B gleich dem Wert in F ist (passende Werte sind rot und fett gekennzeichnet, womit 3 Datensätze eleminiert werden (rötlich unterlegt)

R4 = R3[A,B=F,C,D,E,F,G]

A	B	C	D	E	F	G
a	b	c	d	x	b	c
e	f	g	h	y	f	c
i	j	k	l	z	j	c

SQL-Repräsentation:
SELECT * FROM tabelle1 **INNER JOIN** tabelle2 ON tabelle1.spalte = tabelle2.spalte;
ersatzweise auch möglich / Entsprechung:
SELECT * FROM tabelle1 **JOIN** tabelle2 ON tabelle1.spalte **=** tabelle2.spalte;
SELECT * FROM tabelle1, tabelle2 **WHERE** tabelle1.spalte **=** tabelle2.spalte;

Aufgaben:

1. *Gesucht ist eine Relation R6, die sich über einen inneren Verbund bildet, wobei in C und G auf gleiche Werte geprüft werden soll!*
2. *Wie sieht das Ergebnis für den Fall aus, dass das Kreuz-Produkt aus R2 X R1 gebildet wird und dann die Bedingung (C und G gleich) angewendet wird? Erläutern Sie Ihr Ergebnis!*

natürlicher Verbund / Natural Join

???: R1 JOIN R2
??? Ergebnis sind alle möglichen Kombinationen der Tupel aus beiden Tabellen, die bestimmte Attribut(-Werte) gleich haben

Natural Join ist ein Equi-Join (innerer Verbund) mit einer nachfolgenden Projektion (Spaltenausblendung)
natürlicher Verbund ist kommutativ und assoziativ
bei der Abrarbeitung können diese Eigenschaften genutzt werden, um die zu bearbeitende Datenmenge klein zu halten

(Verknüpfung über Fremschlüssel-Primärschlüssel-Beziehung mit Ausblendung () äquivalenter Spalten (z..B. Fremdschlüssel-Primärschlüssel-Verbindungen ())

R3 ist Kreuz-Produkt als Zwischen-Ergebnis und vorrangig gelblich dargestellt

R1

A	B	C	D
a	b	c	d
e	f	g	h
i	j	k	l

R2

E	F	G
a	b	c
i	j	k

R3 = R1 X R2

A	B	C	D	E	F	G
a	b	c	d	**a**	b	c
e	f	g	h	a	b	c
i	j	k	l	a	b	c
a	b	c	d	i	j	k
e	f	g	h	i	j	k
i	j	k	l	**i**	j	k

jetzt Projektion unter Ausblendung der zu A identischen Spalte E (rötlich unterlegt)

R4 = R3[A=E,B,C,D,F,G]

A	B	C	D	F	G
a	b	c	d	b	c
i	j	k	l	j	k

SQL-Repräsentation:
SELECT tabelle1.*, tabelle2.spalte INNER JOIN tabelle2 ON (tabelle1.spalteX = tabelle2.spalteX AND tabelle1.spalteY = tabelle2.spalteY);

Aufgaben:

1.
2.
3.

voller Inklusionsverbund / Full (Outer) Join

???: R1 FULL R2
Kombination aus linkem und rechtem Inklusionsverbund, leere Felder werden mit NULL belegt

R1

A	B	C	D
a	b	c	d
e	f	g	h
i	j	k	l

R2

E	F	G
a	2	3
i	5	6

R3 = (R1[A] = R2[E]), R1[B,C,D], R2[F,G]

A	B	C	D	F	G
a	b	c	d	2	3
e	f	g	h	NULL	NULL
i	j	k	l	5	6

SQL-Repräsentation:
SELECT * FROM tabelle1 FULL JOIN tabelle2 ON (tabelle1.spalteX = tabelle2.spalteX AND tabelle1.spalteY = tabelle2.spalteY);

/ Semi Join

???: R1 SEMI R2
Kombination aus einem natürlichen Verbund und einer anschließenden Projektion auf die Attribute der ersten Tabelle

R1

A	B	C	D
a	b	c	d
e	f	g	h
i	j	k	l

R2

E	F	G
a	2	3
i	5	6

R3 = (R1[A] = R2[E]), R1[B,C,D]

A	B	C	D
a	b	c	d
i	j	k	l

SQL-Repräsentation:
SELECT tabelle1.* FROM tabelle1 INNER JOIN tabelle2 ON (tabelle1.spalteX = tabelle2.spalteX AND tabelle1.spalteY = tabelle2.spalteY);

/ Theta Join / Non-Equivalent Join

???: R1 FULL R2
Verallgemeinerung des inneren Verbundes
neben der Gleichheit → innerer Verbund können auch andere Vergleiche / Formeln

SQL-Repräsentation:
SELECT * FROM tabelle1 INNER JOIN tabelle2 ON (tabelle1.spalteX [= | < | <= | > | >= | <>] tabelle2.spalteX AND tabelle1.spalteY = tabelle2.spalteY);

/ Self Join

???: R1 SELF R2
beliebiger Verbund einer Tabelle mit sich selbst

SQL-Repräsentation:
SELECT alias1.spalteX, alias1.spalteY, alias2.spalteX FROM tabelle alias1, tabelle alias2 WHERE alias1.spalteZ = alais2.spalteX (+);

linker Inklusionsverbund / Left (Outer) Join

???: R1 LEFT R2
die (ausgewählten) Daten aus der ersten (linken) Tabelle werden mit ev. vorhandenen Daten aus der zweiten (rechten) Tabelle ergänzt, leere Felder werden mit NULL belegt

SQL-Repräsentation:
SELECT * FROM tabelle1 LEFT JOIN tabelle2 ON (tabelle1.spalteX = tabelle2.spalteX AND tabelle1.spalteY = tabelle2.spalteY);
(SELECT * FROM tabelle1, tabelle2 WHERE (tabelle1.spalteX = tabelle2.spalteX (+) AND tabelle1.spalteY = tabelle2.spalteY (+));)

mit (+) sind die Spalten gekennzeichnet, bei denen Platz für NULL-Werte freigehalten werden muss (auf der rechten Seite)

rechter Inklusionsverbund / Right (Outer) Join

???: R1 RIGHT R2
die (ausgewählten) Daten aus der zweiten (rechten) Tabelle werden mit ev. vorhandenen Daten aus der ersten (linken) Tabelle ergänzt, leere Felder werden mit NULL belegt

SQL-Repräsentation:
SELECT * FROM tabelle1 RIGHT JOIN tabelle2 ON (tabelle1.spalteX = tabelle2.spalteX AND tabelle1.spalteY = tabelle2.spalteY);
(SELECT * FROM tabelle1, tabelle2 WHERE (tabelle1.spalteX (+) = tabelle2.spalteX AND tabelle1.spalteY (+) = tabelle2.spalteY);)

mit (+) sind die Spalten gekennzeichnet, bei denen Platz für NULL-Werte freigehalten werden muss (auf der linken Seite)

symmetrische Differenz

A	B	C	D
a	b	c	d
e	f	g	h
i	j	k	l

E	F	G
1	2	3
4	5	6

A	**B**	**C**	**D**	**E**	**F**	**G**
a	b	c	d	1	2	3
e	f	g	h	1	2	3
i	j	k	l	1	2	3
a	b	c	d	4	5	6
e	f	g	h	4	5	6
i	j	k	l	4	5	6

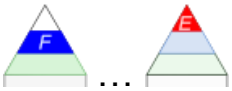
durch die referenzielle Integrität wird abgesichert, das zu jedem Fremd-Schlüssel auch ein Primär-Schlüssel einer anderen Tabelle vorhanden ist (/ passt)

Methoden der analytischen Informationssysteme

- **Online Analytical Processing (OLAP)** — Hypothesen-basierte Analyse von Daten (Bestätigung oder Ablehnung von vordefineirten Hypothesen)
- **Data-Mining** — Suche nach Zusammenhängen (Querverbindungen) zwischen Daten; Erzeugen neuer Daten(-Kombinationen); Ausgraben versteckter Daten
- **Data-Warehouse** — Zusammenfassung von Daten in einem einheitlichen Format (Datenauslage, Datenbereitstellung, Datenausbreitung)

kleine "Regel-Sammlung" zur Relationen-Algebra

"Regel"	Grob-Funktion
Projektion ::= Relation - Attribut(e) = TeilRelation Projektion ::= TeilTabelle = Tabelle - Spalten	Spalten-Auswahl
Selektion ::= Relation = TeilRelation1 + TeilRelation2 Selektion ::= Tabelle - Zeilen	Zeilen-Auswahl
Produkt = Relation X Relation KartesischesProdukt = Tabelle X Tabelle	Multiplikation
Join ::= Produkt + Selektion Verbund ::= Kartesisches Produkt - Zeilen	
EquiJoin = Join + Selektion GleichVerbund ::= KartesischesProdukt - Zeilen	
NaturalJoin = EquiJoin + Projektion natürlicherVerbund ::= GleichVerbund - Spalte(n)	



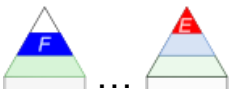
algebraische Darstellung der Relationen-Eigenschaften

R ist reflexiv $\iff$ $I \subseteq R$
R ist irreflesiv $\iff$ $I \cap R = \emptyset$
R ist symmetrisch $\iff$ $R = R^T$
R ist asymmetrisch $\iff$ $R \cap R^T = \emptyset$
R ist antisymmetrisch $\iff$ $R \cap R^T \subseteq I$
R ist transitiv $\iff$ $R \circ R \subseteq R$, also $R^2 \subseteq R$

Rechnen-Regeln der Relationen-Algebra

$R^{TT} = R$
$Q \circ (R \circ S) = (Q \circ R) \circ S$ (Assoziativität des Relations-Produktes)
$I \circ R = R \circ I = R$ (I als neutrales Element des Realtions-Produkts)
$(R \circ S)^T = S^T \circ R^T$ ()

$R^T = R^{-1}$, wenn gilt: $(b,a) \in R^T \iff (a,b) \in R$



Wiederholung / Rückgriff

algebraïsche Elemente aus der Mengenlehre

Benennung	**Symbolik**	**Bedeutung / Beschreibung**
Leere Menge	{} Ø	Menge enthält keine Elemente
Menge	Großbuchstaben: A, B, …, M, N, … oder mit Zusatzziffern bzw. Indizies: M1, M2, … ; M_1, M_2, … für besondere Beziehungen zwischen Mengen: M_A, M_B, …	
Komplement	$\bar{A}_B$ $\bar{A}_B$	:= {x | x ∉ A ∧ x ∈ B}
Mächtigkeit	|M|	Anzahl der Elemente in der Menge
Potenz-Menge	$\mathcal{P}$(A)	:= {X | $X \subseteq$ A}
Gleichheit	A = B	:⇔ ∀x (x ∈ A ⇔ x ∈ B)
Teil Teilmenge	A ⊆ B	:⇔ ∀x (x ∈ A ⇒ x ∈ B)
Disjunkte Mengen	A ∩ B = Ø	Mengen ohne gemeinsame Elemente
Vereinigung	A U B	:= {x | x ∈ A ∨ x ∈ B}
Schnitt Durchschnitt	A ∩ B	:= {x | x ∈ A ∧ x ∈ B}
Differenz	A \ B	:= {x | x ∈ A ∧ x ∉ B}
Symmetrische Differenz	A Δ B	:= {(x | x ∈ A ∧ x ∉ B) ∨ (x | x ∉ A ∧ x ∈ B)}
Kartesisches Produkt	A x B	:= {(a,b) | a ∈ A ∧ b ∈ B}

Beispiel:

Tabellen-Bestand (Datenbank):

Personal

PID	Name	Vorname	Ort	Vorgesetzter	Gehalt
1	Ahrend	Karl	Glücksort	3	2600
2	Koch	Anton	Adorf	9	2300
3	Zander	Michael	Hochberg	NULL	5000
4	Mayer	Frieder	Niederhagen	9	3100
5	Dietrich	Johannes	Brandhagen	9	2900
6	Meier	Claudia	Neu Schönau	3	1800
7	Bauer	Regina	Hochberg	9	2000
8	Tietze	Claudia	Brandhagen	9	2500
9	Lehmann	Monika	Lussow	NULL	5400
10	Schulz	Manne	Niederschönau	9	3400

Kunde

KID	Name	PLZ	Ort	Straße
1	Autohaus Günther	12345	Adorf	Hauptstr. 27
2	Autoverwertung 2000	23456	Niederburg	Kirchenweg 7
3	Autohandel Diamant	55555	Gleichstätt	Lessingstr. 4
4	Gebrauchtwagen 24/7	23456	Niederburg	Pappel-Allee 124
5	Zander-Gebrauchte	99999	Pechhagen	Nordstr. 23
6	Meier-Autohaus Adorf	12345	Adorf	Waldweg 9
7	Fahrzeuge Schmidt	12321	Mittelhausen	Goethe-Platz 2
8	Meier-Autohaus Gleichstätt	55555	Gleichstätt	Pechhagener Weg 33

Auftrag

AID	Auftragsdatum	KID	PID
1	03.06.2012	1	4
2	20.12.2012	5	1
3	30.08.2013	8	4
4	01.02.2014	2	1
5	23.11.2014	3	10
6	02.01.2015	1	4

5. SQL – die Datenbank-Sprache

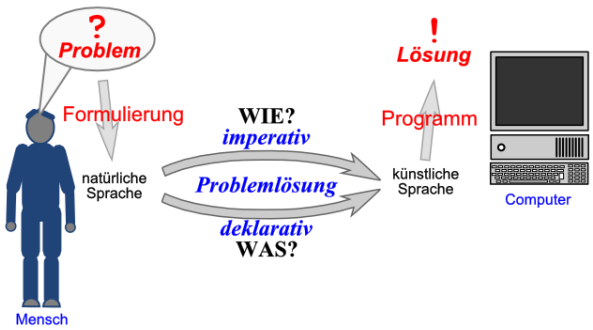
Structured Query Language
textorientierte Datenbank-Sprache zur Erzeugung und Manipulation von Tabellen und / oder Daten

Daten-Abfrage- und –Definitions--Sprache

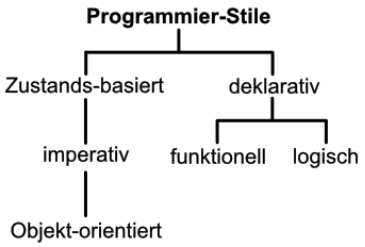
zur Verwaltung von Daten, Datenbanken und Zugriffsrechten und zur Datenmanipulation gedacht
sollte möglichst auch von weniger Programmier-affinien Personen nutzbar sein
herausgekommen ist die wohl leichteste Datenbank-Anfragesprache
für englisch-sprechende Personen mit geringen Computer- und Programmier-Kenntnissen lesbar und verständlich
mit wenig Übung anwendbar

gilt auch very high abstracly programming language (sehr hoch abstrahierende / abstrahierte Programmiersprache)

deklarative Sprache, d.h. wir müssen nicht spezifizieren, wie etwas gemacht werden soll – also den Algorithmus umsetzen – sondern wir sagen, **was** das Ergebnis sein soll



Bedeutung von Kenntnissen in SQL wird auf dem Job-Markt hoch eingestuft. Praktisch liegt die Häufigkeit von Bedarfen für einen angebotenen Job an zweiter Stelle direkt hinter JAVA.
Das liegt auch daran, dass viele Programmiersprachen SQL eingebettet benutzen. Man kommt also praktisch als Informatiker oder Programmierer nicht mehr an SQL und Datenbanken vorbei.



Übersicht über Programmier-Stile

Vorteile des deklarativen Konzepts:

- einfach zu erlernen, oft auch für weniger professionelle Nutzer gut nutzbar
- Computersystem selbst optimiert die Umsetzung und Ausführung des Programms → sind sehr schnell / effektiv
- i.A. wesentlich kleinerer Quelltext
-

Nachteile des deklarativen Konzepts:

- Daten-Zusammenhänge gehen für den außenstehenden Nutzer ev. verloren
-

DQL Data Query Language (Daten-Abfrage-Sprache)

DDL Data Description Language / Data Definition Language (Daten-Definitions-Sprache)
DML Data Manipulation Language (Daten-Manipilations-Sprache / Daten-Zugriffs-Sprache)
DCL Data Control Language (Daten-Kontroll-Sprache)
DRL Data Retrieval Language (Daten--Sprache)

Anfragen werden in sogenannte QEP's (sprich: quecks) zerlegt. Die **Q**uery **E**xecution **P**lan's stellen optimierte Arbeits-Schema's dar, umkomplxe Aufgaben überhaupt sinnvoll und zeitrelevant zu lösen

erste Version 1986 (ANSI-Standard)

SQL89

zwei mögliche Ebenen des umgesetzten / realisierten Sprachumfangs

- **Level 1**
- **Level 2**

SQL92 / SQL2

vier mögliche Ebenen des umgesetzten / realisierten Sprachumfangs

- **Entry Level**
- **Transitional**
- **Intermediate Level**
- **Full Level**

SQL2008 / SQL3

aktuelle Version SQL:2008 ISO/IEC 9075:2008

Ebene	typische SQL-Anweisungen (Datendefinitionen)	
extern	CREATE VIEW DROP VIEW	
konzeptionell	CREATE TABLE ALTER TABLE DROP TABLE CREATE DOMAIN ALTER DOMAIN DROP DOMAIN	
intern	CREATE INDEX ALTER INDEX DROP INDEX	

böse Frage zwischendurch:

Woher kommt die Unterscheidung der Ebenen in der obigen Tabelle und was beinhalten die Ebenen?

Alternativ zu diesem Skript kann man auch den SQL-Kurs (→ https://www.w3schools.com/sql/default.asp) auf der Entwicker-Lernplattform w3schools.com nutzen. Der Kurs ist teilweise übersetzt und führt durch die wesentlichen Inhalte. Die Test's sind anspruchsvoll, aber leider auch immer wieder mal in englischer Sprache. Am Schluß kann man auch ein Zertifikat erreichen.
w3schools wird auch von anderen Lernplattformen (z.B. AppCamps.de) und Tutorial's benutzt, um die praktischen Übungen zu realisieren. Auch über diese Wege ist ein anders orientierter Lern-Zugang möglich.
Im Abschnitt → 5.2. SQL lernen bei w3schools.comgehen wir auf einige Aspekte der Arbeit mit w3schools.com ein.

Daten-Typen in SQL

- **Integer** / **Int** — ganze Zahl
- **Smallint** — je nach DBMS verkleinerter Integer-Bereich
- **Numeric(*gs*,*ns*)** — Dezimal- / Gleitkomma-Zahl mit genau ***gs*** Stellen und mit ***ns*** Nachkomma-Stellen
- **Decimal(*ms*,*ns*)** — Dezimal- / Gleitkomma-Zahl mit mindestens ***ms*** Stellen und mit ***ns*** Nachkomma-Stellen
- **Real** — Dezimal- / Gleitkomma-Zahl mit sogenannter einfacher Genauigkeit
- **Double Precision** — Dezimal- / Gleitkomma-Zahl mit sogenannter doppelter / höherer Genauigkeit
- **Float(*st*)** — Dezimal- / Gleitkomma-Zahl mit einer Genauigkeit von ***st*** Stellen
- **Character** / **Char** — einzelnes Zeichen
- **Character(*n*)** / **Char(*n*)** — Zeichenkette aus genau ***n*** Zeichen
- **Varchar(*n*)** — Zeichenkette mit maximal ***n*** Zeichen
- **Boolean** — Wahrheitswert
- **Date** — Kalender-Datum
- **Time** — (Uhr-)Zeit

Links:

https://www.w3schools.com/sql/default.asp (Lern-Plattform / relativ gut teil-übersetztes Tutorial zu SQL mit Test's und Zertifikat)

https://appcamps.de/ (freie Lern-Plattform für Schulen zu vorwiegend informatischen Themen (durchgehend deutsch))

5.1. wichtige SQL-Anweisungen

Abarbeitung der wichtigsten in der üblichen Gebrauchs-Reihenfolge
z.T. Gruppierung nach theoretischen Konzepten dieses Skriptes
von einfachen Nutzungen zu immer spezielleren
keine Unterscheidung nach Sprachteil-Gruppen
nicht vollständig, nur die üblichen Schul-relevanten Anweisungen werden besprochen
Hier geht es noch viel weniger um das Er- oder Auswendig-Lernen einer Sprache, als beim Erlernen einer Programmiersprache im schulischen Kontext.
Elementare Anweisungen mit typischen Anwendungen.

Hier wird jetzt vielleicht auch deutlich, warum wir uns immer die SQL-Repräsentationen von irgendwelchen Operationen auf unseren Datenbanken angesehen haben. Da haben wir schon ein Gefühl für die Ausdrucks-Stärke von SQL-Anweisungen bekommen.
Besonders praktisch ist die quasi parallel nutzbare graphische Anwendung in Datenbanken, wie ACCESS und BASE und die Text-basierte Notation der Anweisungen in klassischen SQL-Datenbanken (wie z.B. SQLite →). Vielfach kann man die Arbeits-Veriante wählen, die einem am Meisten zusagt oder die das Problem am Schnellsten lösen hilft.

Verbindung zum Datenbanksystem / Datenbank-Konsole
z.B. für eine mySQL-Installation:

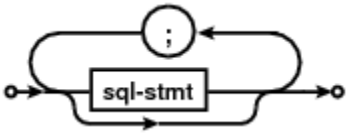
```
> mysql –h localhost –u root –p
```

Legende / Schreib-Konventionen:

Notation	Beschreibung / gemeintes Objekt	Beispiel(e)	Hinweise / Bemerkungen
fett	SQL-Anweisungen oder Syntax-Elemente Terminal	**CREATE** … **;**	
fett, rot	Sytax-Strukturen Wiederholung Alternative Option	**{ }** **|** **[ ]**	
fett, kursiv	Platzhalter für SQL- bzw. untergeordnete Strukturen Nichtterminal		
kursiv	Platzhalter für Elemente der Datenbank Variable		
fett, blau	besondere Strukturen, …		

Die Syntax-Diagramme (nebenstehender Stil) sind von der Webseite von SQLite (→ https://www.sqlite.org/syntaxdiagrams.html). Im folgenden nur noch mit der verlinkten Kurz-Referenz /###/ gekennzeichnet.
Was bedeutet ein solches Diagramm? Wie liest man es?
Oben steht der Name des SQL-Ausdrucks. Dieser heißt hier **sql-stmt-list**, was für SQL-Anweisungs-Liste steht. Beim

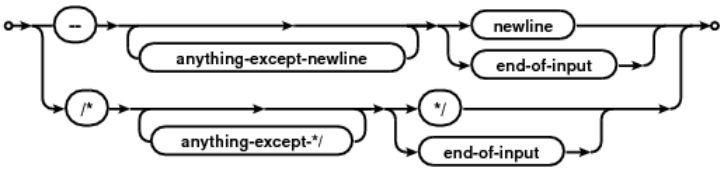
sql-stmt-list:



/###/

Lesen beginnt man am linken Punkt, den man sich als Verbinder / Kontaktstelle vorstellen kann.
Man folgt dann der Pfeilrichtung. Hier gibt zwei Möglichkeiten. Entweder man geht zu einem **sql-stmt** (SQL-Statement (SQL-Anweisung)) oder direkt zum Endpunkt rechts. Hier folgt dann ein anderes Syntax-Diagramm mit einem linken Verbinder.
Da sql-stmt von einem Rechteck umrandet ist, handelt es sich hier um einen Namen für ein weiteres folgendes Syntax-Diagramm mit dem Namen **sql-stmt** (SQL-Anweisung).
Nach einem **sql-stmt** kann man aber auch zum Anfang zurückkehren. Um z.B. ein weiteres **sql-stmt** folgen zu lassen ist aber zwangsläufig ein Semikolon (**;**) notwendig. Solche notwendigen Zeichen – auch Terminale genannt, werden in abgerundeten (ovalen) Rechtecken oder Kreisen notiert. Solche Namen, wie **sql-stmt** sind nur Bezeichner (Platzhalter, Variablen) für andere Syntax-Diagramme. Sie heißen auch Nicht-Terminale. In SQL-Anweisungen kommen sie so nicht vor. Sie müssen immer durch Terminale und / oder Variablen und / oder Namen (z.B. von Tabellen usw.) ersetzt werden.
Diese konkreten Namen für Tabellen usw. werden in den Syntax-Diagrammen *kursiv* gesetzt.
Syntax-Diagramme basieren auf den sogenannten BACKUS-NAUR-Formen zur Darstellung von Sprachen (s.a. → 🕮 **Sprachen und Automaten**).

comment-syntax:



/###/

5.1.1. CREATE DATABASE – Erstellen einer Datenbank

Erzeugen einer neuen Datenbank:
CREATE DATABASE *datenbankname*;

⋮

/###/

weiter z.B.:
→ 5.1.6. CREATE TABLE – Erstellen einer Tabelle
→ 5.1.2. CREATE USER – Erstellen eines Nutzers

5.1.2. CREATE USER – Erstellen eines Nutzers

Erstellen eines neuen Nutzers:
CREATE USER *nutzername'@'localhost* **IDENTIFIED BY** *anmeldename*;

⋮

/###/

weiter z.B.: (Voraussetzung sind vorhandene Tabellen usw. in der Datenbank)
→ 5.1.6. CREATE TABLE – Erstellen einer Tabelle
sonst:
→ 5.1.3. GRANT – Setzen von Zugriffsrechten
→ 5.1.4. REVOKE – Entziehen von Zugriffsrechten

5.1.3. GRANT – Setzen von Zugriffsrechten

Setzen von Zugriffs-Rechten auf die Datenbank oder Teile oder der aufgerufenen Funktionabilität dazu:

GRANT ALL ON *datenbank* **TO** *nutzername***;**
statt ALL können auch die zulässigen Befehle angegeben werden, also z.B.: SELECT, DELETE, UPDATE, REFERENCES
Gegenstück ist **REVOKE** (Entzug der Zugriffs-Rechte)

GRANT SELECT, DELETE, UPDATE, REFERENCES(*nummer***) ON** *tabelle* **TO** *nutzername***;**

5.1.4. REVOKE – Entziehen von Zugriffsrechten

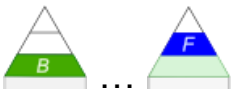
REVOKE DELETE ON *tabelle* **FROM** *nutzername***;**

die Vergabe von Zugriffs-Rechten erfolgt mit **GRANT**

⋮

/###/

5.1.5. USE DATABASE – Verbindung zu einer Datenbank herstellen

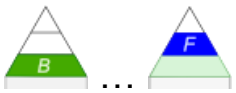


USE DATABASE *datenbank***;**

⋮

/###/

5.1.6. CREATE TABLE – Erstellen einer Tabelle



Erstellen einer Tabelle einschließlich ihrer Struktur (ohne eigentliche Daten):
CREATE TABLE *tabelle* **(** ***attributTyp*** **{ , *attributTyp* } , PRIMARY KEY (*attribute*)) { , FORREIGN KEY (***attribut***) REFERENCES** *referenz_tabelle* **(***referenz_attribut***) });**

CONTRAINT …		
… PRIMARY KEY	setzt Spalte als Primär-Schlüssel	
… REFERENCES	definiert Spalte als Fremd-Schüssel	
… NOT NULL	erzwingt Daten-Eingabe (Wert kann nicht leer sein!)	
… CHECK	realisiert eine Prüfung der Eingabe	

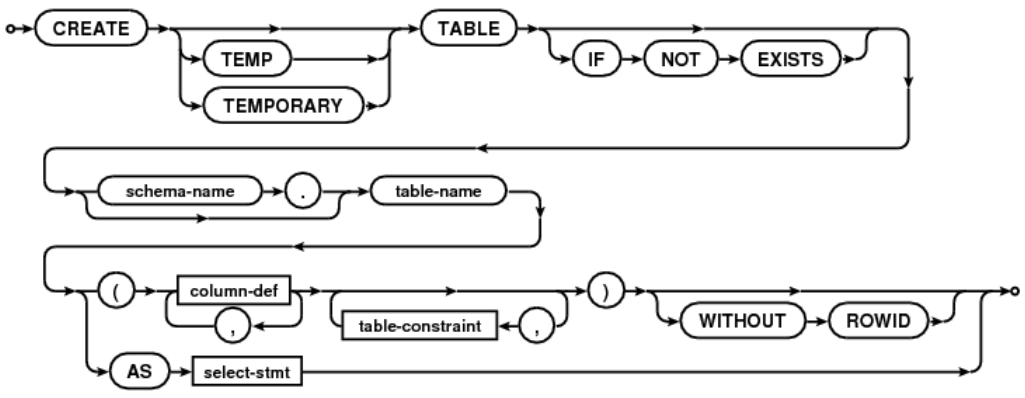
mit **UNIQUE(**spalte**)** wird festgelegt, das die Werte in der Spalte einmalig sein müssen
bei **UNIQUE(**spalte1, spalte2**)** muss die Kombination aus beiden Spalten-Werten einmalig sein

CREATE SEQUENCE variable **INCREMENT BY** 1 **MINVALUE** 1;
INSERT INTO *tabelle* **(***spalte1* **{** , *spalteN* **}**) **VALUES** (*variable***.NEXTVAL { ,** wertN **})**;

komplexes Beispiel:

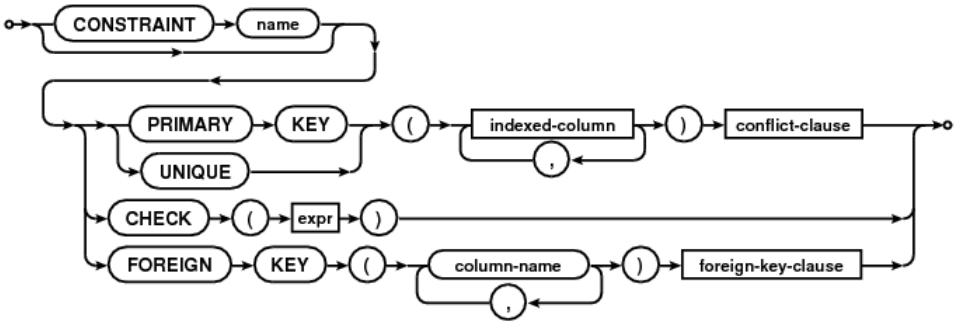
```
CREATE TABLE Nutzer ( NID INT CONSTRAINT Loginname PRIMERY KEY,
   Nachname VARCHAR(30) CONSTRAINT Nutzer_Name NOT NULL,
   Vorname VARCHAR(40) CONSTRAINT Nutzer_Vorname NOT NULL,
   GeburtsJahr INT,
   Geschlecht CHAR(1) CONSTRAINT Nutzer_Geschlecht CHECK ( TYP IN ('w', 'm') ),
   GebLand VARCHAR(20) ….. CHECK(GebLand="Deutschland")
   Groesse INT ….. CHECK((Groesse>=40) AND (Groesse<=250))
   Konto VARCHAR(25) CHECK(Konto LIKE "DE%")
   UNIQUE( NID ),
   UNIQUE( Nachname, Vorname ) );
```

create-table-stmt:



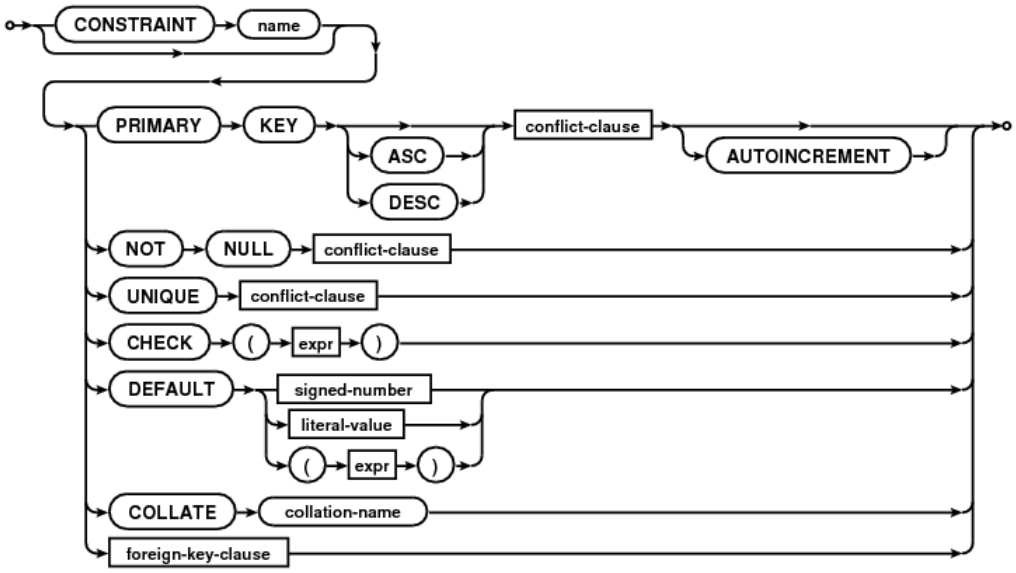
/###/

table-constraint:



/###/

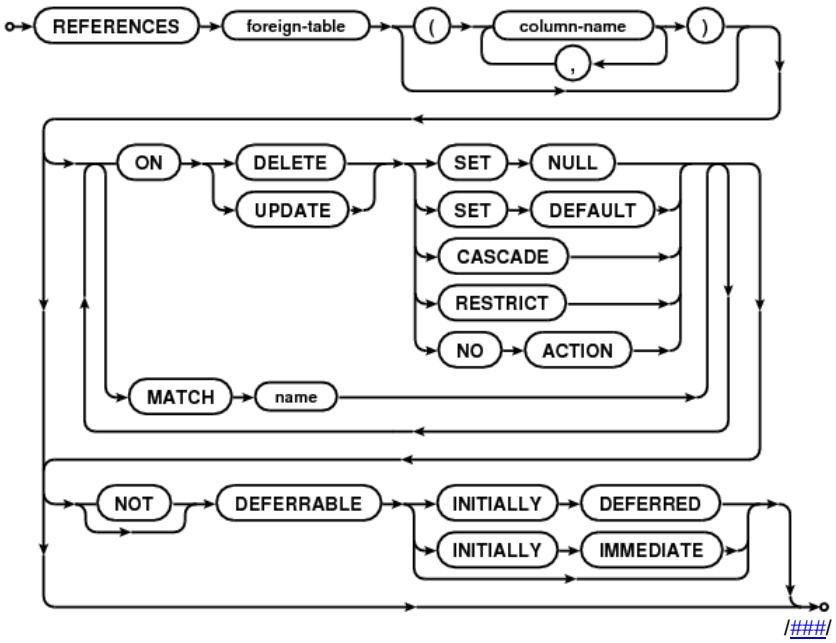
column-constraint:



/###/

5.1.6.1. Tabellen mit Fremdschlüsseln erstellen

foreign-key-clause:



/###/

Beispiel-Konstrukt:
Artikel:=(…; ↗ ArtikelGruppe; …)
ArtikelGruppe:=(AGrID; …)

Optionen bei der Definition von Fremd-Schlüsseln:

ON DELETE RESTRICTED:
ON UPDATE RESTRICTED:
es werden alle Lösch- bzw. Aktualisierungs-Anweisungen in der verknüpften Tabelle (hier: ArtikelGruppe) verweigert, solange es noch es noch Datensätze in der "übergeordneten" Tabelle (hier: Artikel) gibt, die den zu löschenden Fremd-Schlüssel enthalten

ON DELETE CASCADE:
ON UPDATE CASCADE:
alle Lösch- oder Aktualisierungs-Anweisungen (also Primär-Schlüssel-Aktualiserungen) in der verknüpften Tabelle (hier: ArtikelGruppe) wird an die übergeordnete Tabelle (hier: Artikel weitergegeben und alle Datensätze in diese Tabelle gelöscht / aktualisiert

ON DELETE SET NULL:
ON UPDATE SET NULL:
Löschung oder Aktualisierung eines Primär-Schlüssels in der verknüpften Tabelle (hier: ArtikelGruppe) bewirkt einen NULL-Wert in der übergeordneten Tabelle (hier: Artikel)

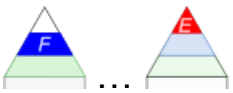
ON DELETE SET DEFAULT:
ON UPDATE SET DEFAULT:
Löschung oder Aktualisierung eines Primär-Schlüssels in der verknüpften Tabelle (hier: ArtikelGruppe) bewirkt ev. eine Neu-Setzung des Schlüssel-Wertes auf den / einen Vorgabe-Wert in der übergeordneten Tabelle (hier: Artikel)

die beiden letzten Varianten zerstören die referenzielle Integrität
ON CASCADE sollte nur mit Bedacht benutzt warden, da so schnell viele Kategorien / … schnell aus der datenbank verschwinden (und bei Bedarf wieder neu eingegeben werden müssen)

Beispiel aus der Warenhaus-Welt:

```
CREATE TABLE Artikel
(       AID             Interger        NOT NULL,
        ANummer         VarChar(15)     NOT NULL,
        ABezeichnung    Var0Char(30)    NOT NULL,
        …
        APreis          Float(2)        NOT NULL DEFAULT 0,

        PRIMARY KEY(AID),

        FOREIGN KEY(LID) REFERENCES Lieferant
        ON UPDATE CASCADE
        ON DELETE NO ACTION,

        …

        CHECK(APreis >= 0)
)
```

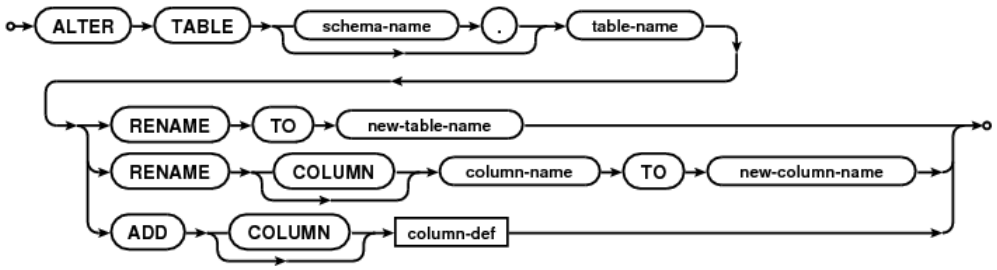
5.1.7. ALTER TABLE – Ändern der Tabelle(n-Struktur)



verändern der Tabellen-Struktur, z.B. Verändern der Feld-Größe:
ALTER TABLE *tabelle* **MODIFY (***attribut* **NUMERIC(***neueLänge***));**

oder Hinzufügen einer neuen Spalte:
ALTER TABLE *tabelle* **ADD (***attribut* **CHAR(***länge***));**

alter-table-stmt:

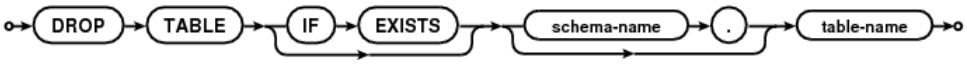


/###/

5.1.8. DROP TABLE – Löschen einer Tabelle

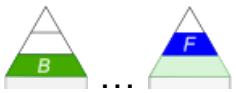
löschen einer ganzen Tabelle (einschließlich des Inhalts):
DROP TABLE *tabelle* **INCLUDING CONTENTS;**

drop-table-stmt:



/###/

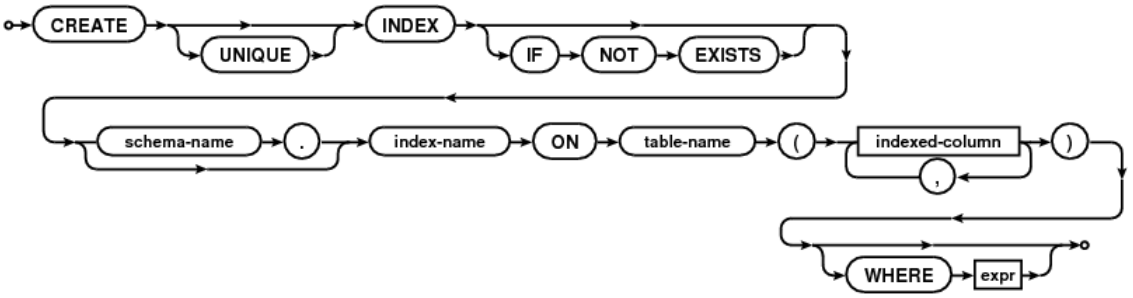
5.1.9. CREATE / ALTER / DROP INDEX – Erstellen, Ändern und Löschen eines Index



Es gelten die selben Prinzipien, wie bei **TABLE.**

Erstellen eines neuen Index:

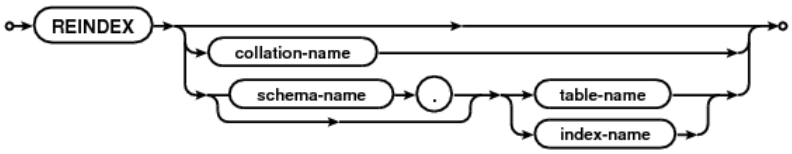
create-index-stmt:



/###/

Neuerstellen / Aktualisieren eines Index

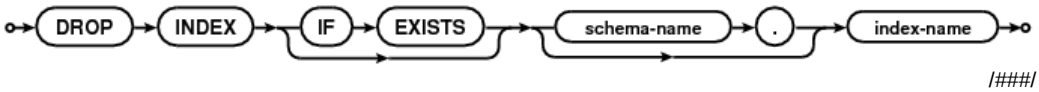
reindex-stmt:



/###/

Löschen eines Index

drop-index-stmt:



/###/

5.1.10. INSERT – Einfügen eines Datum's

Einfügen von Daten in eine Tabelle:
INSERT INTO *tabelle* **(***attribut* **{ , ***attribut* **}) VALUES (***wert* **{ , ***wert* **})**

:

/####/

5.1.11. SELECT … FROM – Auswählen von Spalten und / oder Zeilen

spalten ::= attribut { , attribut }
aggregation ::= aggregationsfunktion(spalte)
bedingung ::= vergleich

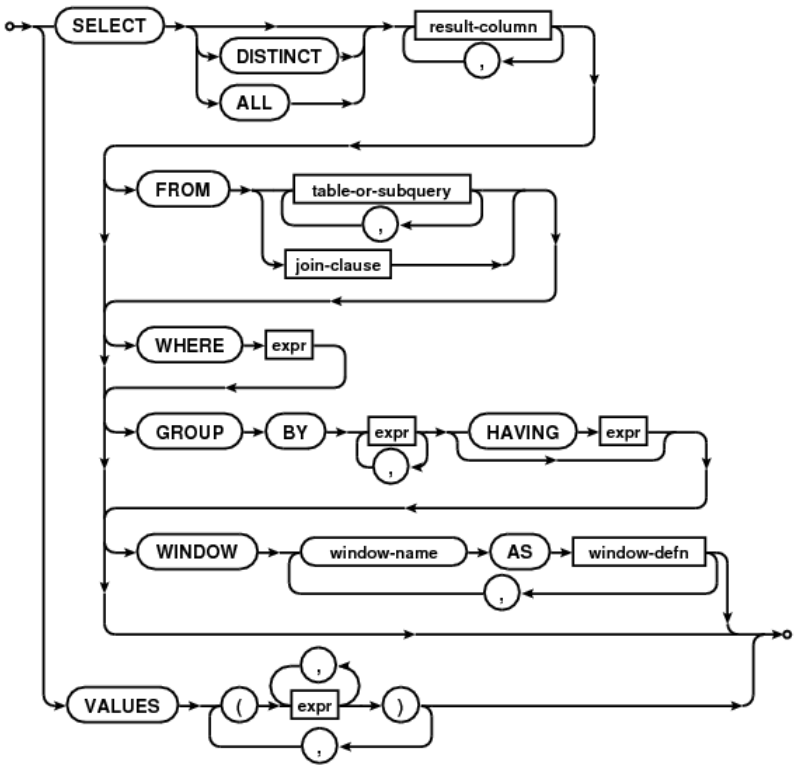
bedingung mus als Ausdruck immer ein WAHR oder FALSCH (TRUE / FALSE) ergeben.

Syntax:
SELECT ***spalten*** **|** ***aggregation*** **FROM** ***tabelle*** **[WHERE** *bedingung* **] [GROUP BY** *spalten* **|** *index* **] [HAVING** *bedingung* **] [ORDER BY** *spalten* **[ASC] | DESC]**

ASC .. (ascending) aufsteigende / ansteigende Reihenfolge / Sortierung
DESC .. (descending) absteigende Reihenfolge / Sortierung

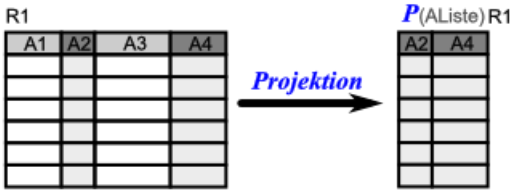
Darstellung einer Tabelle / Anzeige aller Daten einer Tabelle
SELECT * FROM *tabellenname*;

select-core:



/###/

5.1.11.1. Projektion (Abbildung):



Bei der Projektion werden von der Original-Tabelle oder einer Sicht nur bestimmte Spalten abgebildet. Eine Auswahl der Datensätze erfolgt nicht.
Es werden praktisch Sichten erzeugt.
Projektionen dienen z.B. dazu die angezeigte / verfügbare Daten-Menge zu reduzieren oder den Datenschutz-Bedingungen anzupassen.

SELECT *attribut* **{ ,** *attribut* **}** **FROM** *tabelle***;**

Anzeige ausgewählter Spalten einer Tabelle
SELECT *spaltenname1, spaltenname2* **{**, *spaltennameN***}** **FROM** *tabellenname*;

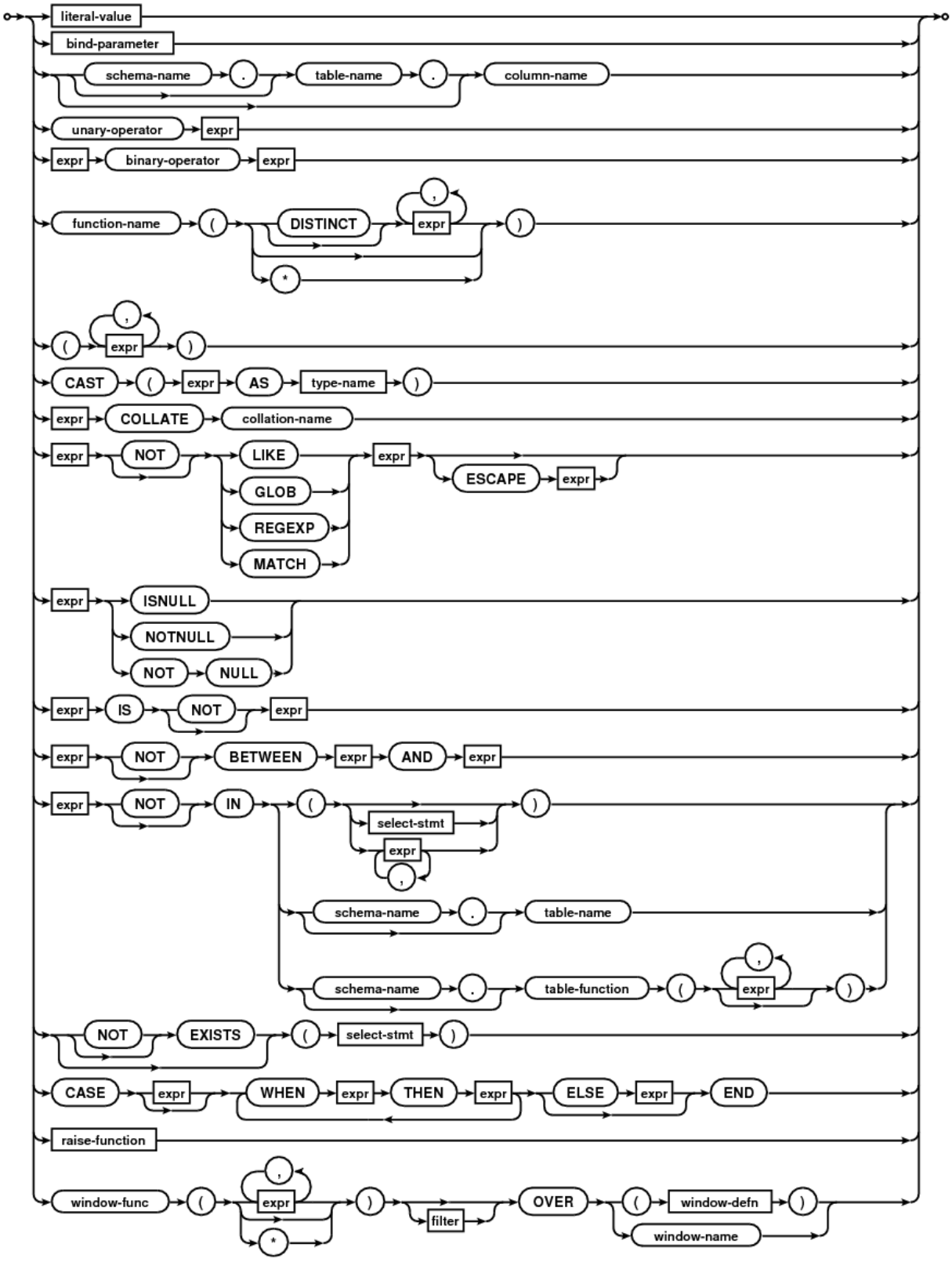
Anzeige ausgewählter Spalten (nach ihrer Positionsnummer in der Tabelle)
Zählung beginnt bei 1!
SELECT 1, 2 **{**, N**}** **FROM** tabellenname;

nach bestimmten Spalten sortierte Ausgabe ausgewählter Spalten einer Tabelle (meint standardmäßig aufsteigende Sortierung)
SELECT *spaltenname1, spaltenname2* **{**, *spaltennameN***}** **FROM** *tabellenname* **ORDER BY** spaltennameX **{**, spaltennameZ**}**;

… betont aufsteigende Sortierung
SELECT *spaltenname1, spaltenname2* **{**, *spaltennameN***}** **FROM** *tabellenname* **ORDER BY** *spaltennameX* **{**, *spaltennameZ***}** **ASC**;

… mit absteigenden Sortierung:
SELECT *spaltenname1, spaltenname2* **{**, *spaltennameN***}** **FROM** *tabellenname* **ORDER BY** *spaltennameX* **{**, *spaltennameZ***}** **DESC**;

expr:


literal-value
bind-parameter
schema-name
.
table-name
.
column-name
unary-operator
expr
expr
binary-operator
expr
function-name
(
DISTINCT
expr
,
*
)
(
expr
,
)
CAST
(
expr
AS
type-name
)
expr
COLLATE
collation-name
expr
NOT
LIKE
GLOB
REGEXP
MATCH
expr
ESCAPE
expr
expr
ISNULL
NOTNULL
NOT
NULL
expr
IS
NOT
expr
expr
NOT
BETWEEN
expr
AND
expr
expr
NOT
IN
(
select-stmt
expr
,
)
schema-name
.
table-name
schema-name
.
table-function
(
expr
,
)
NOT
EXISTS
(
select-stmt
)
CASE
expr
WHEN
expr
THEN
expr
ELSE
expr
END
raise-function
window-func
(
expr
,
*
)
filter
OVER
(
window-defn
)
window-name

/###/

5.1.11.1.1. Projektion für Fortgeschrittene:

Projektion nur eines Teils aus der Spalte (Teilstring):
SELECT SUBSTR(*spaltenname, startPosition, anzahlZeichen*) **FROM** *tabellenname***;**

Projektion mit Verbindung (Konkatenation) von Spalten zu einer neuen (hier mit Komma-Trennung):
SELECT *spaltenname1* **| |** ', ' **| |** *spaltenname2* "*neuerSpaltenname*" **FROM** *tabellenname***;**

Zusammenfassung gleicher Attribut-Werte (Spalten-Inhalte) zu einer Zeile:
SELECT DISTINCT *spaltenname* **FROM** *tabelle***;**
damit verhandert man das mehrfache Aufführen gleicher Attribut-Werte im Ergebnis

Nutzung von Aggregat-Funktionen (); hier: Zählung mit **COUNT()**
SELECT COUNT(*) "*Anzahl*" **FROM** *tabelle***;**

weitere Aggregat-Funktionen sind:

Aggregat-Funktion	Aufgabe	Bemerkungen
MIN()	Minimum	
MAX()	Maximum	
SUM()	Summe	
AVG()	Durchschnitt	

Gruppierung gleicher Werte einer Spalte (hier Spalte1) und gruppierte Summierung der Werte aus Spalte2:
SELECT *spaltenname1,* **SUM(***spaltename2***) FROM** *tabelle* **GROUP BY** *spaltenname1***;**

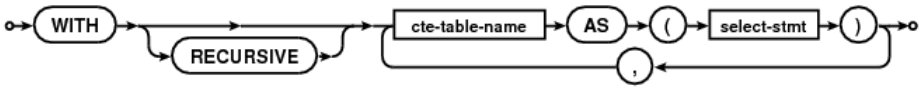
begleitete Daten-Typen-Konvertierung
SELECT TO_CHAR(*spaltenname, format***)** "Anzahl" **FROM** *tabelle***;**
SELECT TO_CHAR(*Datum, 'YYYY'***)** FROM tabelle;
weitere Daten-Typen-Konvertierungen: **TO_NUMBER()**, **TO_DATE()**

NULL-Werte sollen durch spezielle Texte etc. ersetzt werden:
SELECT *spaltenname1,* **NVL(** *spaltenname2, '???'***) FROM** *tabelle***;**

SELECT *Name*, **NVL(** *GebDatum, '???'***) FROM** *tabelle***;**

die ersten n Zeilen einer sortierten Auswahl anzeigen:
SELECT * FROM (SELECT ROWNUM *nummer, resultatTabelle.** **FROM (SELECT * FROM** *tabelle* **[WHERE** … **] ORDER BY** …) *resultatTabelle*) **WHERE** *nummer* = *n***;**

with-clause:



/###/

Beispiel(e) aus der Warenhaus-Welt:

den billigsten Artikel(-Preis) anzeigen:
SELECT MIN(APreis) FROM Artikel

die Daten für den billigsten Artikel anzeigen:
```
SELECT * FROM Artikel
WHERE APreis = (    SELECT MIN(APreis)
                    FROM Artikel
               )
```

den billigsten Artikel(-Preis) in jeder ArtikelGruppe anzeigen:
```
SELECT AGrID, MIN(APreis)
FROM Artikel
GROUP BY AGrID
```

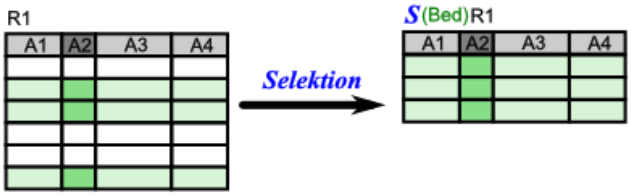
den billigsten Artikel in jeder ArtikelGruppe (mit seinen Daten) anzeigen:
```
SELECT Tab1.*                                          Tab1 ist ein Alias
FROM Artikel Tab1                                      hier Def. des Alias
WHERE Tab1.APreis = (     SELECT MIN(Tab2.APreis)            Tab2 ist Alias
                          FROM Artikel Tab2                  Def. Alias
                          WHERE Tab1.AGrID = Tab2.AGrID
                    )
```

die Anzahl der Artikel ermitteln:
```
SELECT COUNT(*)
FROM Artikel
```

den durchschnittlichen Preis in jeder ArtikelGruppe berechnen:
```
SELECT AGrID, AVG(APreis)
FROM Artikel
GROUP BY AGrID
```

5.1.11.2. Selektion (Auswahl):

Aufgrund bestimmter Kriterien, Bedingungen, Werte, … werden bestimmte Datensätze ausgewählt.
Besonders typisch ist dazu der WHERE-Teil / -Anhang in SELECT-Anweisungen. Aber es gibt auch andere Möglichkeiten Datensätze auszuwählen.

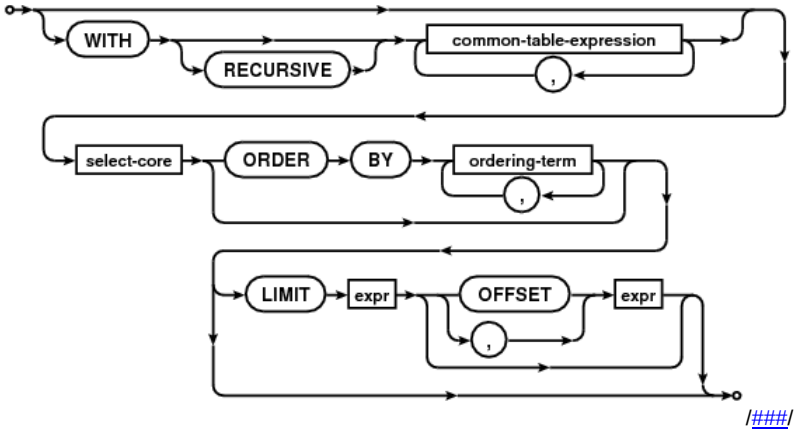


SELECT * FROM *tabelle* **WHERE *bedingung*;**
bedingung kann z.B. der Vergleich mit einem Wert sein; bedingung muss dann den spaltennamen ein Vergleichzeichen und den Vergleichswert enthalten
z.B.: spaltenname = wert oder spaltenname < wert oder spaltenname >= wert bzw. spaltenname <> wert usw. usf.

weiterhin geht für ***bedingung*** auch:
spaltenname **BETWEEN** *kleinsterWert* **AND** *größterWert*
wir suchen also aus einem Bereich heraus

Texte werden in einfache Hochkommata (**' '**) eingeschlossen
um z.B. die Groß- und Klein-Schreibung bei Texten nicht einzeln zu prüfen können Texte z.B. auf Groß-Buchstaben umgestellt werden mit **UPPER()**
(bei ACCESS wird die Funktion UCASE() benutzt)

simple-select-stmt:



/###/

SELECT * FROM *tabelle* **WHERE** *bedingung1* **{ AND** *bedingungN* **};**

SELECT * FROM *tabelle* **WHERE** *bedingung1* **{ OR** *bedingungN* **};**

SELECT * FROM *tabelle* **WHERE** *spaltenname* **LIKE** *suchausdruck*;
suchausdruck kann z.B. 'M%' sein, dann werden alle Ausdrucke mit einem beginnenden großen B akzeptiert
enthält der suchausdruck Unterstriche, dann stehen diese für genau ein beliebiges Zeichen

SELECT * FROM *tabelle* **WHERE** *spaltenname* **IN** *suchmenge*;
suchmenge wird in runden Klammern angegeben; also z.B.: (1, 3, 5, 7)

SELECT * FROM *tabelle* **WHERE** *spaltenname* **IS NULL**;

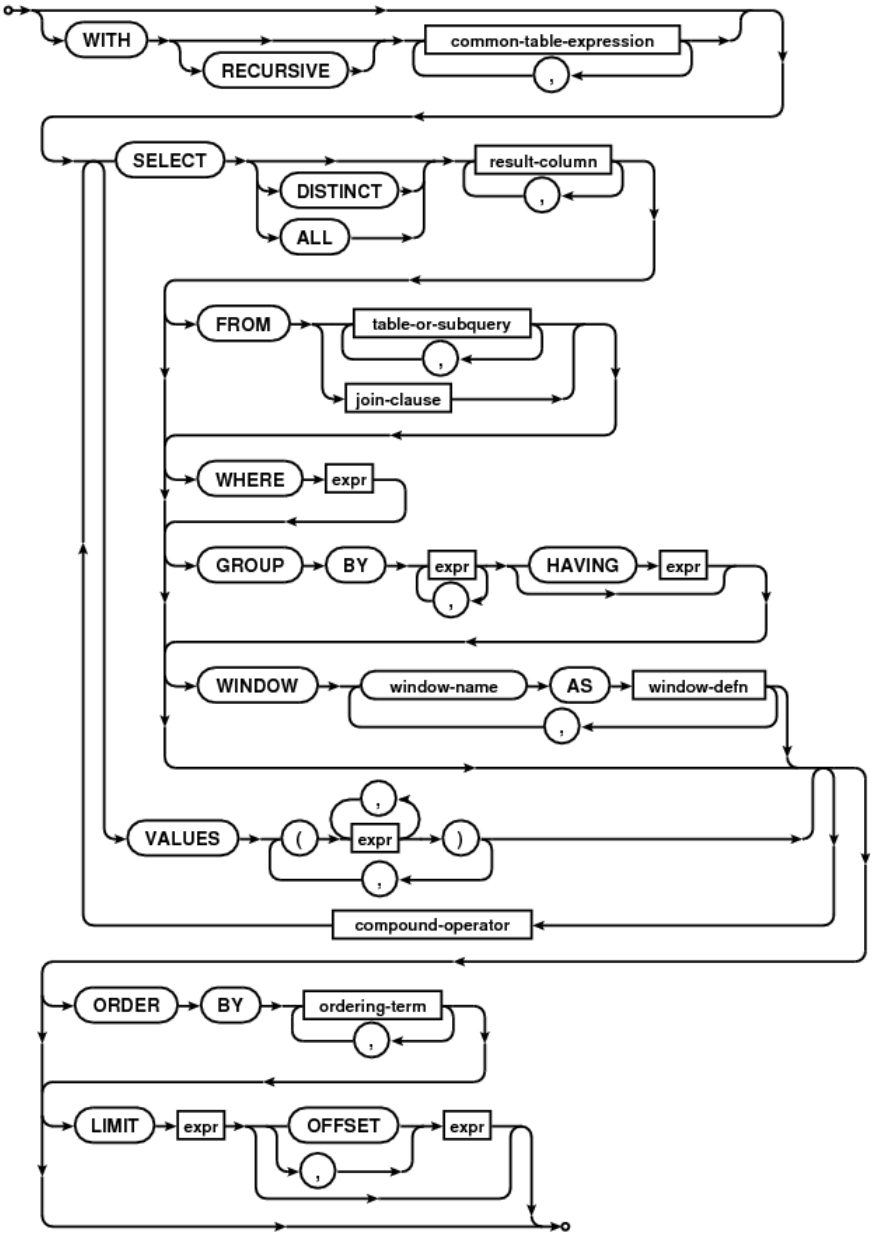
SELECT * FROM *tabelle* **WHERE** *spaltenname* **IS NOT NULL**;

5.1.11.2.1 Selektion für Fortgeschrittene:

Datum-bezogene Auswahl (hier bestimmte Anzahl Tage zurück):
SELECT * FROM *tabelle* **WHERE** *spaltenname* **>=** (**SYSDATE** – *anzahlTage*)**;**
SELECT * FROM *tabelle* **WHERE** *spaltenname* **>=** (**NOW()** – *anzahlTage*)**;**
die System-Datums-Funktion ist in den DB-Systemen unterschiedlich realisiert

SELECT * FROM *tabelle* **WHERE** *spaltenname1* **IS NOT NULL AND** *spaltenname2* **IS NULL;**

select-stmt:



/###/

das Schlüsselwort **DISTINCT**

Syntax
SELECT DISTINCT *spalte* **{**, *spalte***}** **FROM** *tabelle* **WHERE** ….

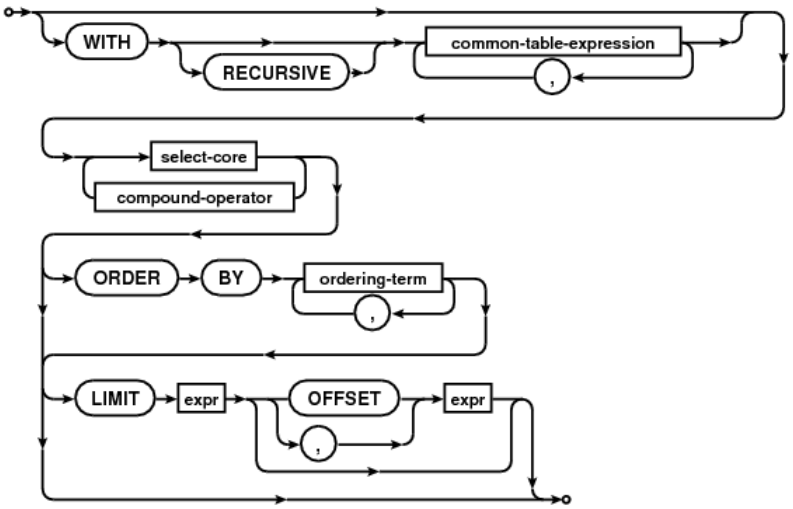
verhindert Wiederholung gleicher Ergebnisse / Ergebnis-Zeilen bezüglich der aufgezählten Attribute (Spalten)

oft gebraucht ist das Auswerten der so kompromierten Tabelle, z.B. dahingehend, wieviel verschiedene Einträge (Werte) nun auftauchen
SELECT COUNT(DISTINCT *spalte* **) FROM** *tabelle* **WHERE** ….

in ACCESS kann dieser Konstrukt nicht funktionieren. Dann bietet sich der folgende Ausdruck an:
SELECT COUNT(*) AS *hilfstabelle* **FROM (SELECT DISTINCT** *spalte*
FROM *tabelle* **WHERE** …**)**.

/###/

factored-select-stmt:



/###/

mit LIMIT begrenzt man die Anzahl der Tupel (Datensätze) z.B., wenn es darum geht die 10 besten Schüler oder die 3 schnellsten Auto's oder die 5 größten Länder usw. usf.
natürlich lassen sich bei umgekehrter Sortierung auch die schlechtesten, langsamsten und kleinsten finden

komplexere Beispiel(e) aus der Warenhaus-Welt:

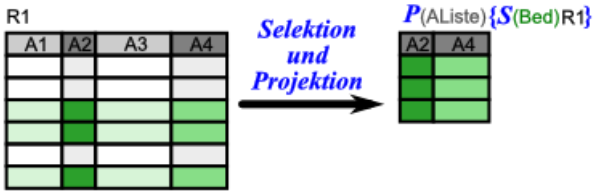
die ArtikelGruppe mit der Preis-Summe anzeigen, bei denen die Preis-Summe einen bestimmten minimalen Wert hat:

```
SELECT AGrID, SUM(APreis)
FROM Artikel
GROUP BY AGrID
HAVING SUM(APreis) >= 10
```

für jeden Kunden die Anzahl seiner Bestellungen und die Anzahl der bestellten Artikel anzeigen:

```
SELECT KID, COUNT(*) AS AnzahlBestellungen, SUM(Menge) AS ArtikelAnzahl
FROM Bestellungen
GROUP BY KID
```

5.1.11.3. Verbund (Join):



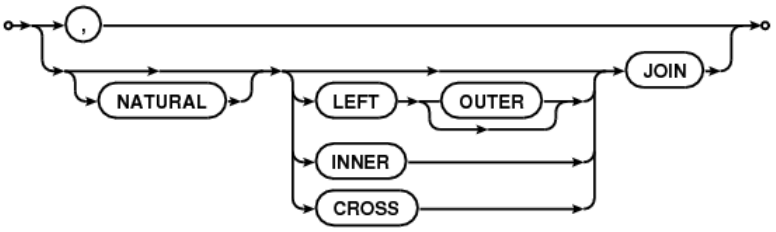
Verbindung von Projektion und Selektion. Hauptsächlich wird dabei in der **SELECT**-Anweisung der Teil vor dem **FROM** dazu benutzt, um die Spalten abzubilden (auszuwählen). Mittels **WHERE**-Abschnitt wird die eigentliche Auswahl vorgenommen.
Hierbei werden echte Sichten z.B. für Formulare / Berichte usw. erzeugt. Sie enthalten nur noch kleinste Daten-Mengen, aber geanu die, die gebraucht / gewünscht / angefordert wurden.

SELECT *
FROM Tabelle1, Tabelle2

liefert das relationale Produkt

Tabelle1 **X** Tabelle2

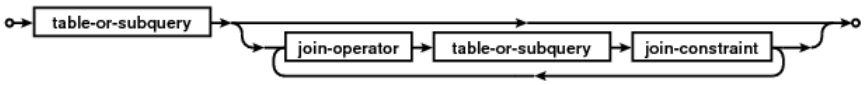
join-operator:



/###/

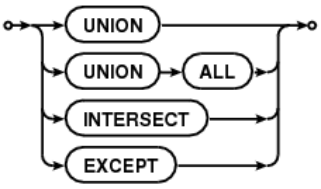
das entspricht einem CROSS JOIN (→ Kreuz-Produkt / kartesisches Produkt)
auch als kartesisches bzw. Kreuz-Produkt bezeichnet

join-clause:



/###/

compound-operator:



/###/

Beispiel(e) aus der Warenhaus-Welt:

:
SELECT
FROM

5.1.11.3.1 Verbund für Fortgeschrittene:

praktisch alle Kombinationen von einfachen und speziellen Projektionen und Selektionen möglich

hohe Kunst der Daten-Auswahl

SQL-Anweisungen werden dann auch schnell unübersichtlich und teilweise auch schwer lesbar, da jetzt die Syntax-Vorschriften dominieren und die Nutzer-Verständlichkeit zurückbleibt.
ev. durch Aufeinanderfolge von einzelnen SQL-Anweisungen / -Sequenzen zu vereinfachen

⋮

/###/

5.1.11.4. Verbund (Equi Join):

SELECT * FROM *tabelle* **WHERE** *spaltenname* **IN (*selektion*);**
selektion ist dabei wieder ein eigenständiger vollständiger Selektions-Befehl (aber ohne inneres, abschließendes Semikolon
quasi kaskadierte Selektion

SELECT * FROM *tabelle1, tabelle2* **WHERE** *tabelle1.attribut = tabelle2.attribut*

⋮

/###/

5.1.11.5. Tabellen-Erstellungs-Abfrage

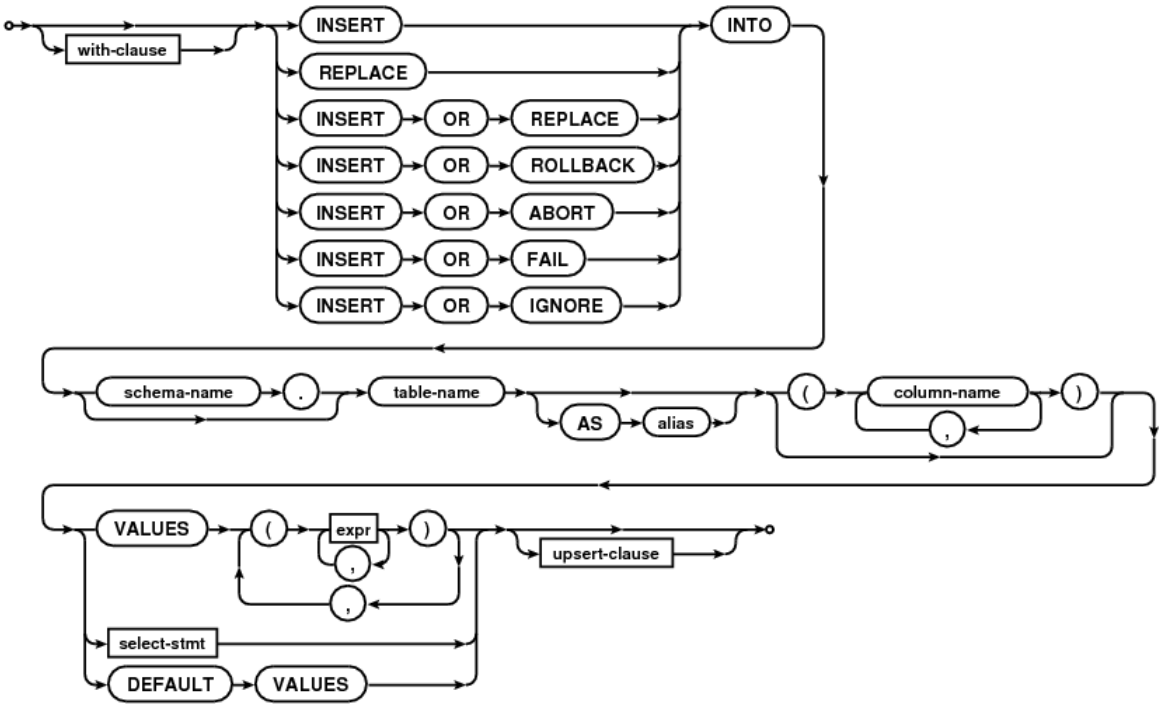
Syntax
SELECT *spalten* **INTO** *neue_tabelle* **FROM** *tabelle* **[WHERE *bedingung*];**

⋮

/###/

5.1.12. INSERT INTO – Einfügen von …

insert-stmt:



/###/

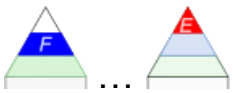
5.1.12.1. Anfüge-Abfrage

Syntax
INSERT INTO *tabelle* **{** ***spalten*** **}** **VALUES** **{** ***werte*** **}** **;**
INSERT INTO *tabelle* **{** ***spalten*** **}** **SELECT** ***spalten*** **FROM** *quell_tabelle* **[** **WHERE** ***bedingung*** **]** **;**

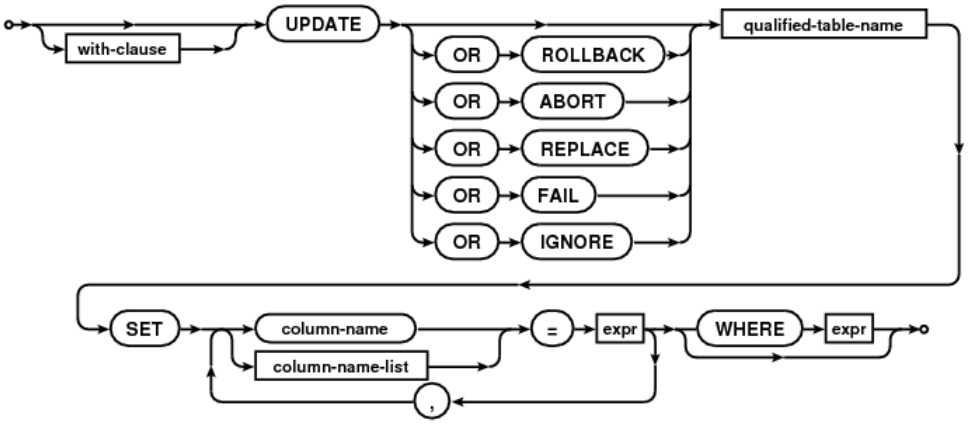
:

/###/

5.1.13. UPDATE – Aktualisieren von …



update-stmt:



/###/

5.1.13.1. Aktualisierungs-Abfrage

Ändern von Inhalts-Werten augrund einer zutreffenden Bedingung:
UPDATE *tabelle* **SET** *attribut = neuer_wert* **{**, *attribut = neuer_wert***}** **[WHERE *bedingung*];**

Syntax
UPDATE *tabelle* **SET** *spalte = ausdruck* **|** *wert* **WHERE *bedingung*;**

:

/###/

5.1.14. DELETE FROM – Löschen von …

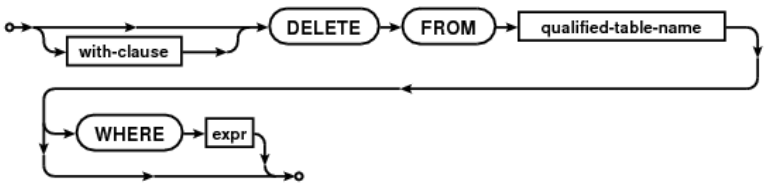
5.1.14.1. Lösch-Abfrage

Löschen von ausgewählten Datensätzen aus einer Tabelle:
DELETE FROM *tabelle* **WHERE *bedingung*;**

DELETE *tabelle*
→ **DROP** *tabelle*

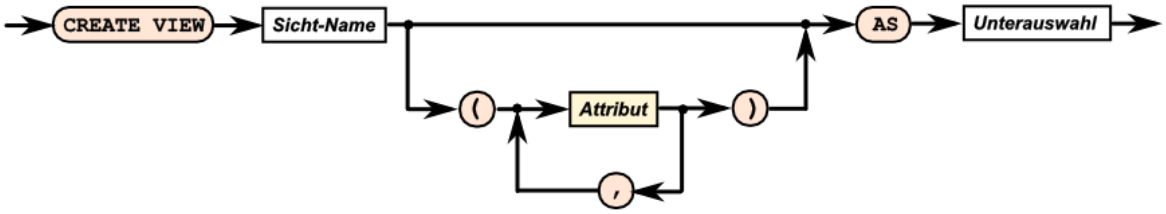
ganze Tabelle löschen:
DROP TABLE *tabelle*

delete-stmt:



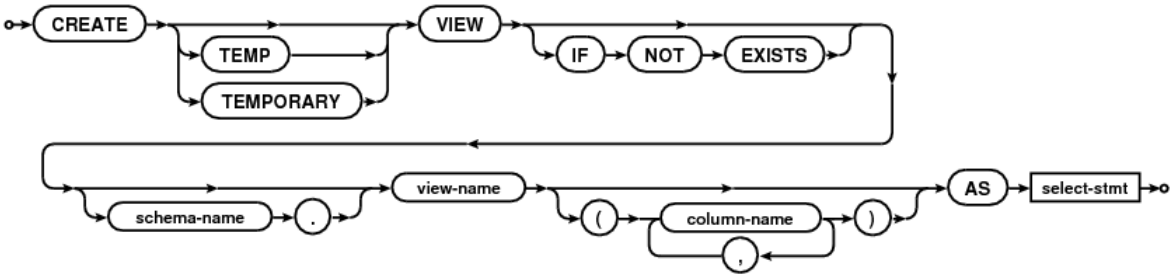
/###/

5.1.15. CREATE / ALTER / DROP VIEW – Erstellen, Ändern und Löschen einer Sicht (Abfrage)



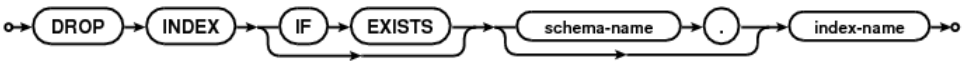
Syntax-Diagramm für CREATE VIEW (!!! noch prüfen!)

create-view-stmt:



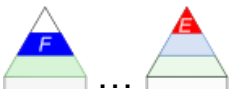
/###/

drop-view-stmt:

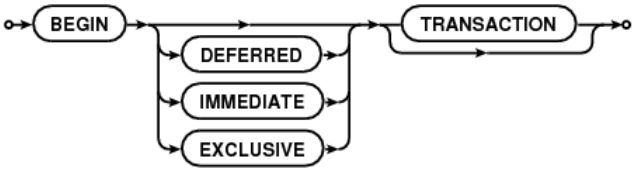


/###/

5.1.16. Transaktionen



begin-stmt:



/###/

commit-stmt:

/###/

wirkliche Ausführung aller DML-Anweisungen seit der letzten COMMIT-Anweisung
COMMIT;

Rücknahme aller DML-Anweisungen seit der letzten COMMIT-Anweisung
ROLLBACK;
immer dann notwendigm wenn eine Transaktion nicht vollständig durchgeführt werden konnte

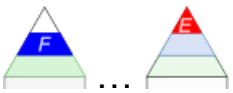
rollback-stmt:

/###/

Sperrung (Einstellung der Unveränderbarkeit) einer Tabelle bis zum nächsten COMMIT bzw. ROLLBACK
LOCK TABLE *tabelle* **IN EXCLISIVE MODE NOWAIT;**

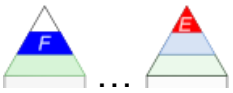
Links:
https://www.sqlite.org/syntaxdiagrams.html (Quelle der SQLite-Syntax-Diagramme in Kapitel 5.x und fehlende Diagramme (verw. Kurz-Referenz: ###))

5.1.17. Berechnungen in Datensätzen



```
SELECT ArtikelID, ArtikelName, BestellMenge, NettoEinzelPreis
                (NettoEinzelPreis)*1,19 AS BruttoEinzelPreis,
                (NettoEinzelPreis)*1,19*(BestellMenge) AS GesamtPreis
FROM Bestellung;
```

5.1.18. Berechnungen über Tabellen und / oder Abfragen

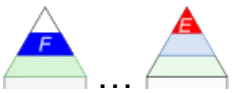


entspricht einer Berichts-Erstellung
Veränderung zu erwarten, wenn sich ein Wert in der Tabelle bzw. den betreffenden Tabellen (für eine Sicht / Abfrage) geändert hat

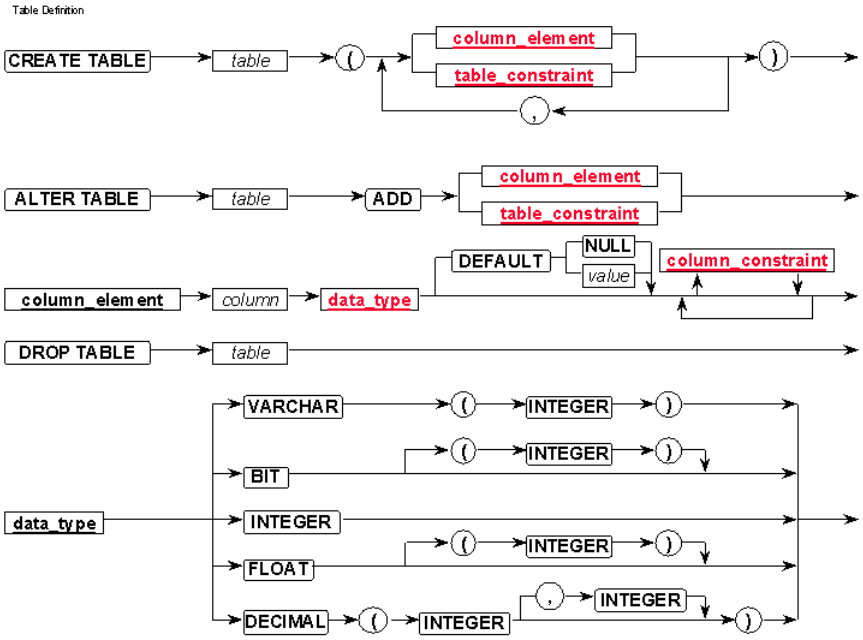
```
SELECT KundenID, BestellID,
                SUM(GesamtPreis) AS RechnungsPreis
FROM Bestellung;
```

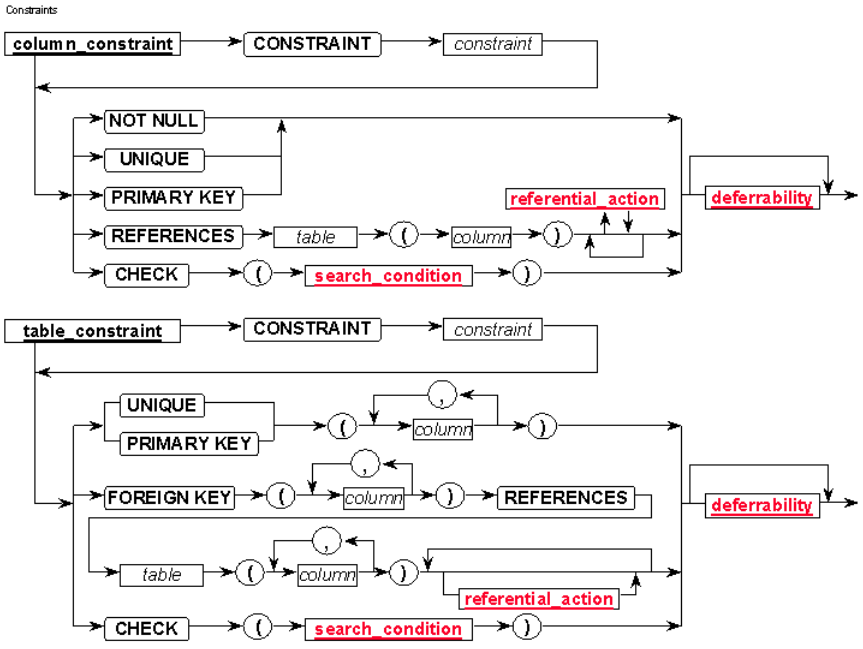
Syntax-Diagramme für SQL'92

(Q: http://www.dbs.ethz.ch/education/infosys/syntax_diagramme/sld001.html)

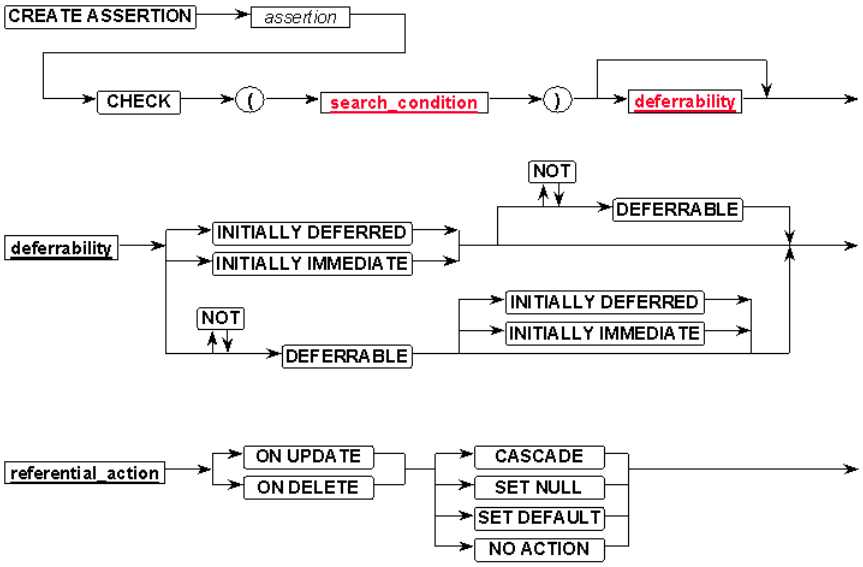


Dieses Syntax-Diagramm ist mit Absicht aufgenommen, um ein zusammenhängendes System für Aufgaben, als Hilfs-Blatt usw. usf. zu haben. Sachlich entspricht das weitesgehend den detaillierten Diagrammen aus SQLite, die wir in den vorlaufenden Abschnitten benutzt haben.

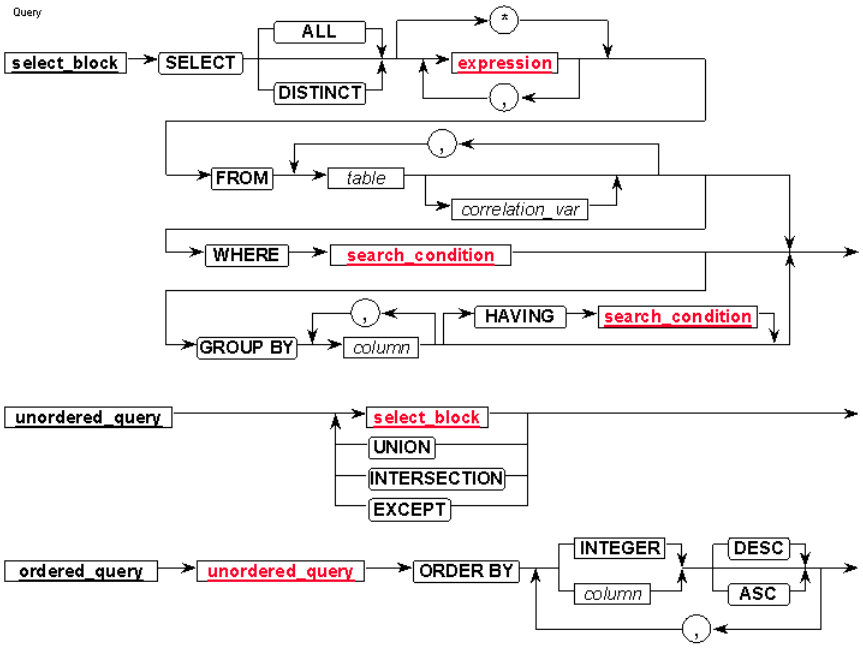




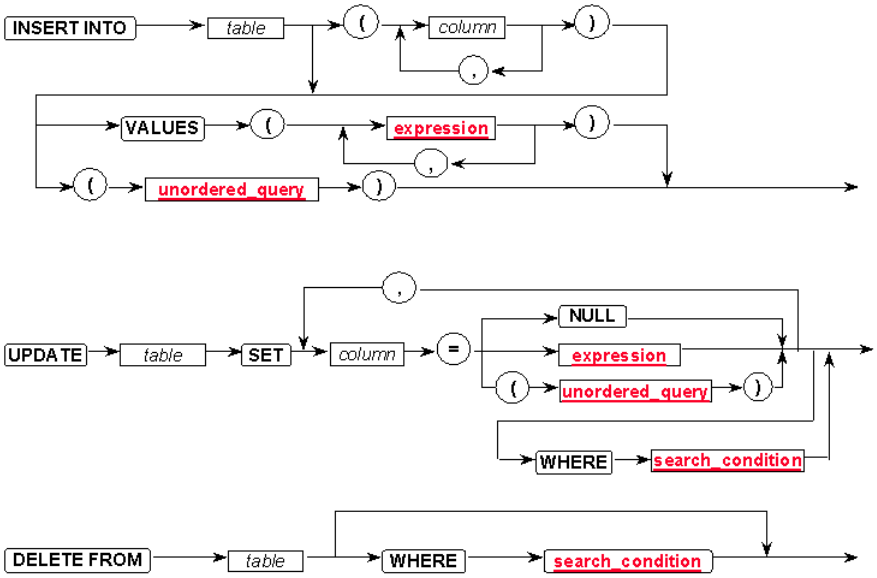
Assertion, Deferrability,
Referential Action



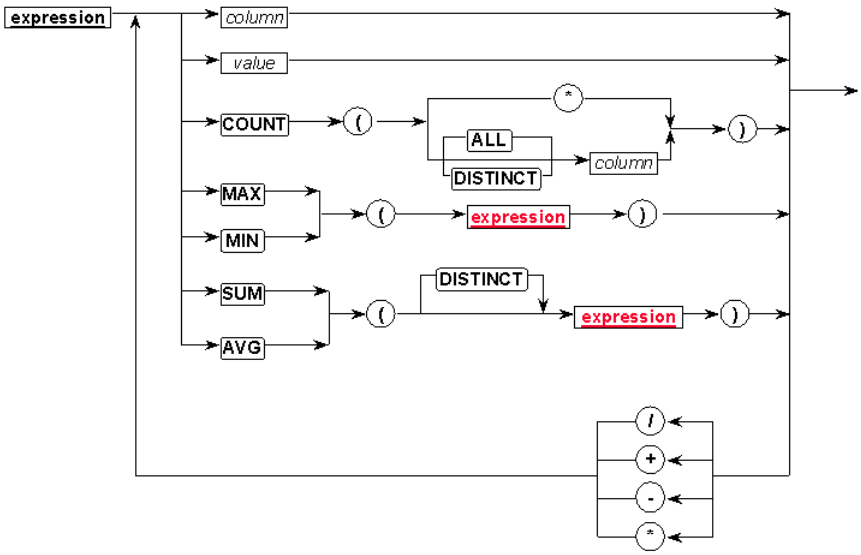
Query



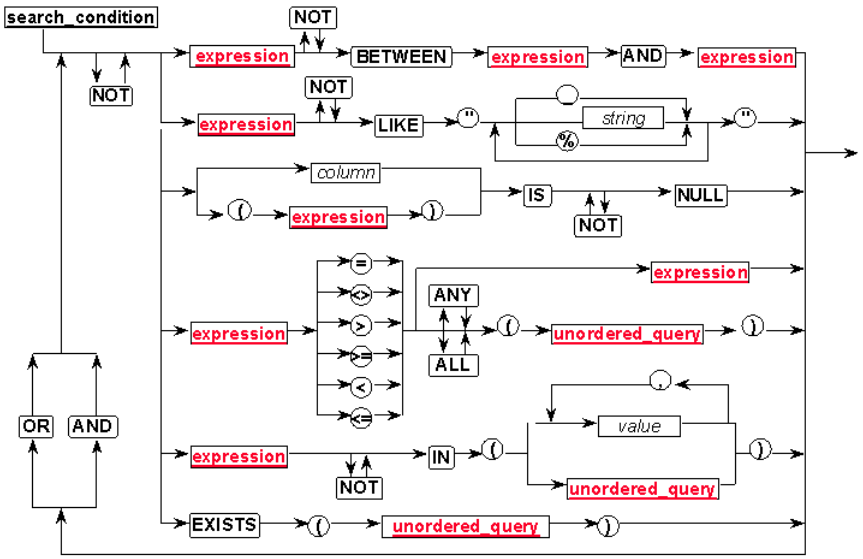
Data Manipulation



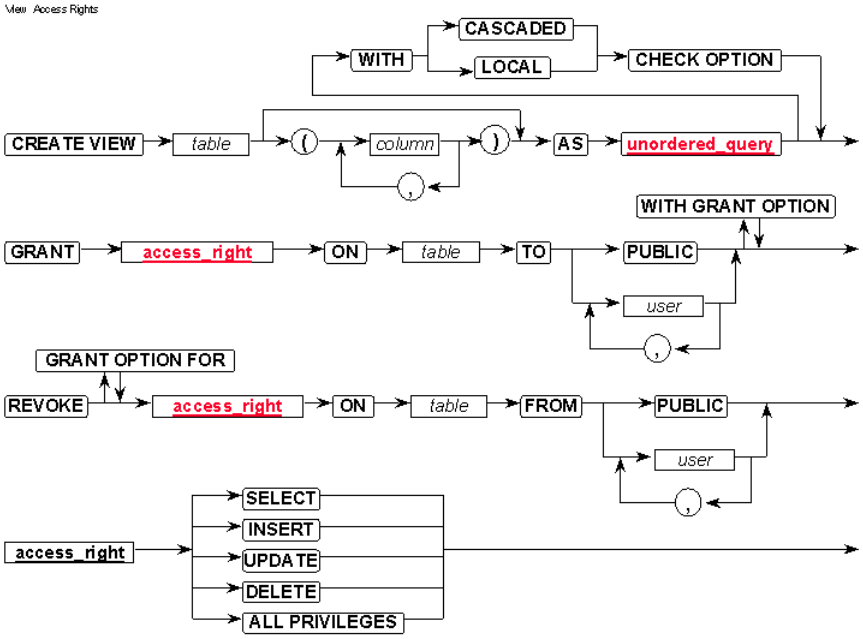
Expression

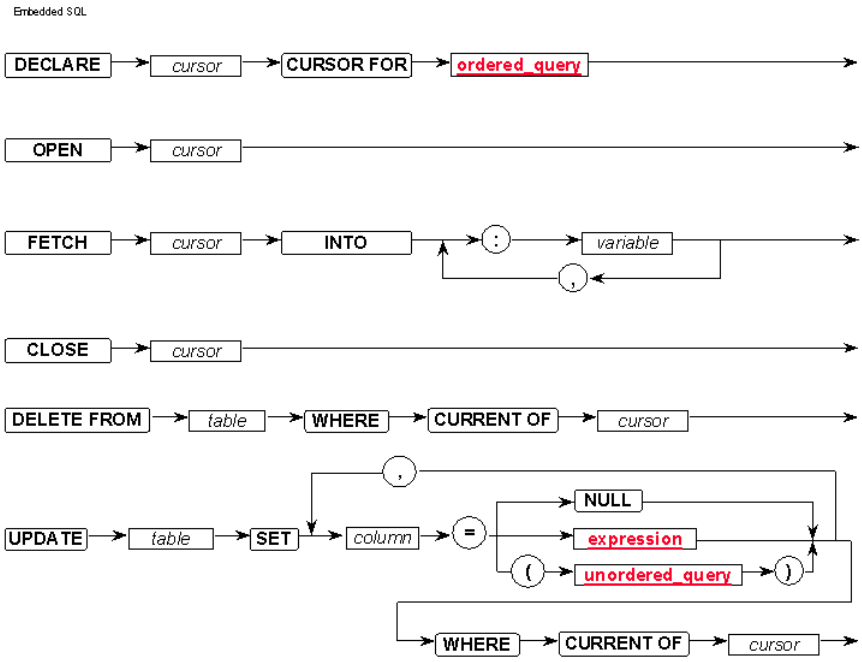


Search Condition



View Access Rights





Aufgaben:

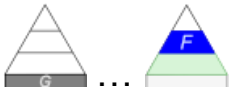
1.

2. Prüfen Sie anhand des Syntax-Diagramms, ob die folgenden Ausdrücke als SQL-Anweisungen zulässig sind!

a) CREATE TABLE Schuhe;
b) CREATE DATABASIS Webshop;
c) WHERE CREATE tabelle FROM attribut1 ORDER BY attribut2;
d) Fahrraeder SELECT Radgroesse FROM "26" WHERE Color="blau";
e)
f)
g)
h)

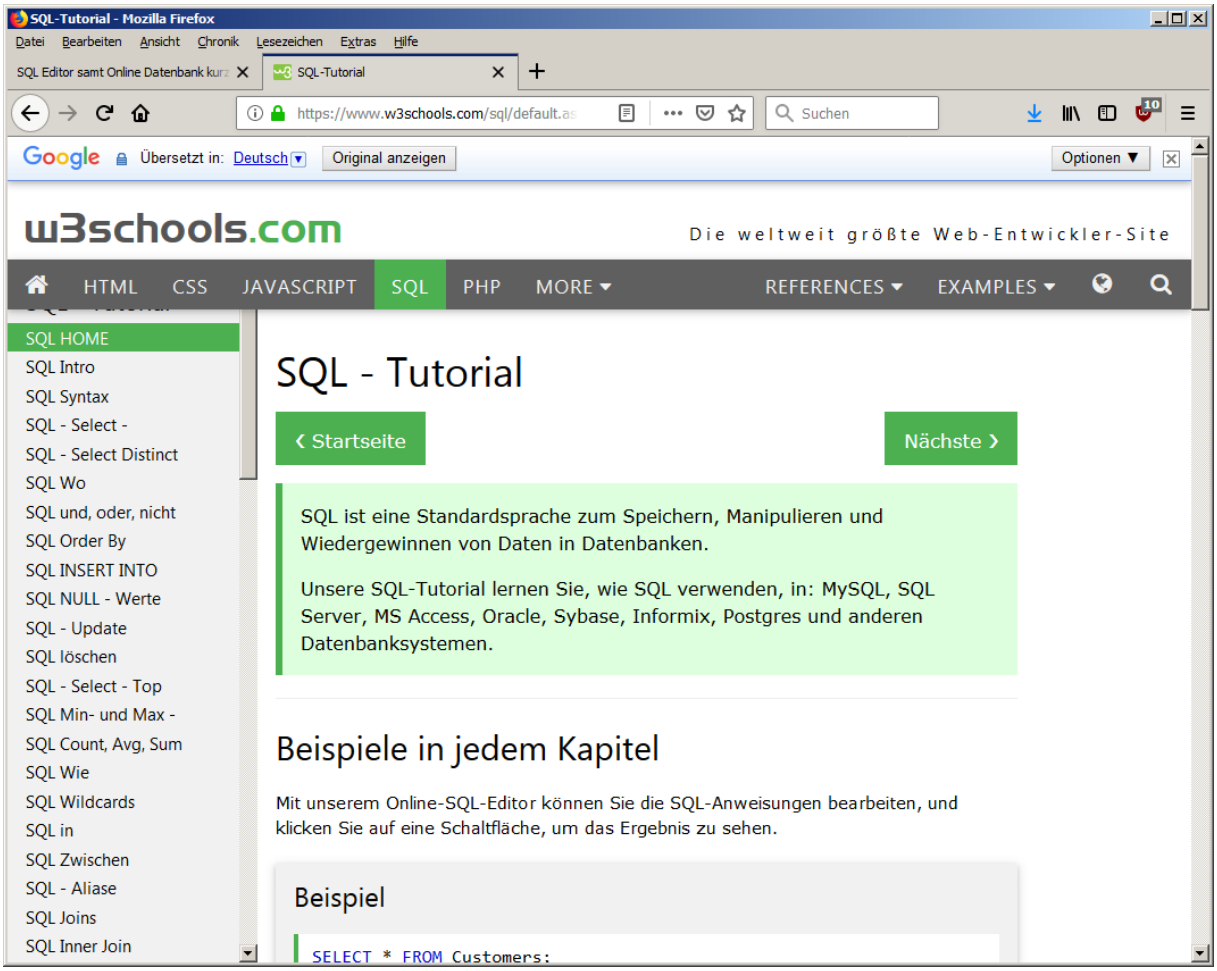
3.

5.2. SQL lernen bei w3schools.com



Wenn der Arbeitsrechner kein Datenbank und / oder SQL-System bereitstellt, dann kann man getrost auf eine online-Plattformen zurückgreifen.
Für Kurse, die Abitur-orientiert haben, können onlinePlattformen nur nebenläufig genutzt werden. In der Prüfungs-Situation steht ja praktisch keine Internet-Verbindung zur Verfügung.
Hier wird meist eine Datenbank mit vielen Daten(-sätzen) und ein SQL-Editor zur Verfügung gestellt.
Somit lassen sich Datenbanken und SQL mit sehr einfachen Smartphone's oder Tablet's erkunden und ausprobieren.
Zerschossener oder fehlerhafter SQL-Code ist kein Problem, man kann i.A. immer wieder von vorne anfangen. Die digitalen Spuren bleiben im Internet.
Der Start-Zustand der Datenbank lässt sich jederzeit wiederherstellen.

online Tutorial bei w3schools.com (→ https://www.w3schools.com/sql/default.asp)
Registrierung nicht notwendig
Arbeits-Stand wird über Cookies gespeichert

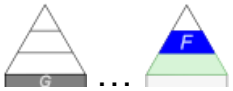


vorrangig englisch; teilweise übersetzt – auch deutsch verfügbar (nach Klick auf das Globus-Symbol erscheint ein Menü mit möglichen Sprachen. in die die Webseite übersetzt wird.
Es wird der google-Übersetzer verwendet. Der ist befriedigend bis gut, kann aber auch verführerisch oder verwirrend sein

Links:
http://www.inf-schule.de/information/datenbanksysteme (Tutorial mit eigenem einfachen Editor)

https://appcamps.de/unterrichtsmaterial/unterrichtsmaterial-zu-datenbanken-und-sql/ (Tutorial und Material von AppCamps)

5.3. Arbeiten mit Übungs-Datenbanken



5.3.1. Nutzung von Datenbanken aus online-Quellen

Prinzipiell haben wir verschiedene Möglichkeiten online-Datenbanken zu nutzen. Die erste Möglichkeit ist eine direkte Nutzung der online-Datenbank über das Internet. Voraussetzung ist natürlich eine ausreichend Leistungs-fähige Internet-Leitung und die Nutzungs-Möglichkeit des Internet's. Z.B. kann die Nutzung während der Durchführungs von Prüfungen eingeschränkt sein.
Meist ist in den online-Datenbanken nur eine lesende Nutzung möglich. Wir können also i.A. nur Anfragen an die Datenbank stellen. Die Datenbanken sind gegen Überschreiben und Löschen von Daten abgesichert. Die online-Nutzer haben keine entsprechenden Rechte. Man braucht sich keine Sorgen machen, dass man durch fehlerhafte SQL-Anfragen einen Schaden anrichtet. Dies darf aber nicht als Aufforderung verstanden werden, die Grenzen der Sicherheit auszutesten.
Die Nutzung selbst hat schon etwas Magisches, wenn man bedenkt, dass man gerade die Daten von einem ev. weit entfernten Rechner runterholt.
Sicherer in der Handhabung sind lokale Datenbanken. Dies setzt aber voraus, dass man die online-Datenbank irgendwie downloaden konnte. Leider sind viele der Datenbestände dann auch wieder sehr gross und unübersichtlich.
Ein entscheidender Vorteil ist, dass man eine zerstörte oder nachhaltig veränderte Datenbank einfach löschen kann und durch die downgeladte originale Datenbank ersetzen kann. Da ist man i.A. innerhalb von Sekunden wieder auf dem Ausgangs-Zustand. Das ist im schulischen Umfeld von großem Vorteil.
Viele Bereitsteller von Datenbanken bieten diese in verschieden Formaten an. Für uns sind die SQLite-Versionen (Datei-Endungen *.db und *. am Besten geeignet.
Natürlich ist die Bereitstellung einer Datenbank als SQL-Anweisungs-Sammlung die Lösung, die am universellsten ist. Schließlich können die von uns besprochenen Programme ja gerade SQL. Die meisten Programme bieten auch eine Import-Möglichkeit an.
Ein Import der SQL-Datenbanken setzt i.A. voraus, dass man im Ziel-Programm zuerst einmal eine neue, leere Datenbank angelegt hat. Erst danach können die Tabellen importiert werden.

Links:

http://www.oberstufeninformatik.de/Datenbanken/ (u.a. SQL-Datenbanken zum Downloaden)

https://dbup2date.uni-bayreuth.de/blocklysql/index.html (Wetter- und Fußball-Bundesliga-Datenbank + Anleitung(en))

5.3.2. Nutzung von Datenbanken aus Abitur-Aufgaben

aus rechtlichen Gründen wird hier nur kurz erläutert, wie man ev. an die Datenbanken kommt
Lehrer aus MV können in einer Moodle-Gruppe auf dem Landes-eigenem Bildungs-Server ua. auch auf die Abitur-Daten zurückgreifen
die Links in diesem Abschnitt funktionieren nur lokal. D.h. man muss diese Datenbanken in dem Ordner dieses Skriptes gespeichert haben. Ich selbst kann diese Datenbanken nicht über den eigenen Schulgebrauch hinaus verteilen!

aus Abitur Mecklenburg-Vorpommern 2017

praktikum.sqlite ("Praktikum")
zu Aufgabe 5

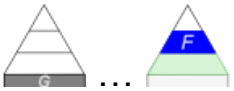
aus Abitur Mecklenburg-Vorpommern 2018

b1.sqlite ()
generiert mit einem Fake-Name-Generator (→ www.fakenamegenerator.com) (Lizenz für die Daten)

aus Abitur Mecklenburg-Vorpommern 2019

Surfen.sqlite (Surfen)
zu Aufgabe A1

5.4. "How to" für SQL / SQL-Koch-Anleitungen



Vielfach habe ich beobachtet, das war die einzelnen SQL-Anweisungen bekannt sind, aber das sich beim Zusammenstellen einer Abfrage ein riesiges Problem auftut.
Da Abfragen auch die Hauptnutzung von SQL in den Datenbank-Kursen (in Schulen) ist, beschränke ich mich hier zuerst einmal auch nur auf die Abfragen. Trotz alledem empfehle ich jedem Anfänger sich einen eigenen SQL-Spicker zu erstellen. Je nach Kurs-Art und Bewertung-Anforderungen ist der Spicker dann vielleicht auch durch den Kurs-Leiter für Lern-Kontrollen freigegeben.
Das "How to" ist für Anfänger / Einsteiger gedacht. Es werden nur typische und häufig benutzte Problemstellungen beachtet. Für komplexere Probleme bietet sich die Beschreibung der SQL-Befehle als Informations-Quelle an. Im "How to" werden die SQL-Anweisungen Tätigkeits- / Handlungs- / Ziel-orientiert vorgestellt. Eine systematische angelegte Vorstellung der SQL-Anweisungen erfolgte bereits in der Referenz (→ 5.1. wichtige SQL-Anweisungen).
U.U. hilft ein Nachlesen / Informieren an dieser Stelle weiter, als die Vorstellung im "How to".
Ein ausführlicheres Beispiel zur Konstruktion eines SQL-Statement's wird bei → 5.4.1.3. Erstellen einer (einfachen Daten-)Tabelle (Beispiel) aufgezeigt. Ev. kann auch der → EXKURS am Ende des "How to" genutzt werden.

Legende: **Beispiel(e)**

SQL-Bestandteile		
müssen exakt so geschrieben oder angeordnet werden (entspricht Terminalen)		
SELECT	obligatorische(r) SQL-Abweisung(steil); für das Problem notwendig	**CREATE TABLE**
SELECT	vor- oder nachlaufender SQL-Anweisung(s-Teil); zeigt die Lage einer obligatorischen Struktur in einem größeren Ausdruck an	
Variablen / Platzhalter		
müssen aus der Datenbank-Struktur stammen und wie dort geschrieben werden		
Tabelle	ein Objekt (Tabelle, Attribut, …) aus der genutzten Datenbank	
Ausdrücke / Formeln		
Meta-Symbole (Nicht-Terminale), die als Variablen / Konstruktions-Vorschriften von Ausdrücken dienen; müssen aus SQL-Bestandteilen, Variablen und / oder Werten zusammengestellt werden		
Bedingung	ein Objekt (Tabelle, Attribut, …) aus der genutzten Datenbank oder ein zusammengesetzter Ausdruck (aus Operation und Objekten)	**BETWEEN** 1 **AND** 100 ***Gehalt*** > 3000
Metazeichen		
dienen nur der verkürzten Schreibweise in den Ausdrücken; sie stellen Optionen, Wiederholungen oder Alternativen dar; sie werden nicht mitgeschrieben		
[…]	es kann weitere Teile der SQL-Anweisung geben; optionale Teile	
{…}	Wiederholungen (z.B. mehrere Tabellen)	***Tabelle*** **{ , *Tabelle* }**
… | …	Alternative (entweder der linke oder der rechte Ausdruck	

5.4.1. Erstellen von Datenbanken, Tabellen und Indizes sowie Daten-Eingabe

5.4.1.1. Erstellen einer Datenbank

CREATE DATABASE *Datenbankname* [...]

Erläuterung:
Die ***Datenbank*** wird dem Datenbank-Management-System bekanntgegeben (Deklaration).

5.4.1.2. Verbinden mit … / Nutzen einer Datenbank

USE DATABASE *Datenbankname* [...]

Erläuterung:
Die angegebene ***Datenbank*** wird zum Arbeiten geöffnet.

5.4.1.3. Erstellen einer (einfachen Daten-)Tabelle

CREATE TABLE *Tabellenname* (*Struktur*)

CREATE TABLE *Tabellenname* (*Attribut* { , *Attribut* } PRIMARY KEY (*Spaltenname*))

CREATE TABLE *Tabellenname* (*Attribut* { , *Attribut* }
PRIMARY KEY (*Spaltenname* { , *Spaltenname* }))

Erläuterung:
CREATE TABLE erzeugt die ***Tabelle*** mit der in Klammern angegebenen ***Struktur***. Die ***Struktur*** ist eine Liste von ***Spaltennamen*** und ev. zugehörigen ***Bedingungen***.

typische Struktur (Attribut-Liste) ist:

- ***Attribut* { , *Attribut* }**

jedes ***Attribut*** besteht aus:

- ***Spaltenname Datentyp* [*Bedingung*]**

gültige ***Datentypen*** sind:

- **INT** für ganze Zahlen (z.B. gut für eine (zusätzliche) Schlüssel-Spalte geeignet)
- **DOUBLE oder REAL** für Zahlen mit Nachkommastellen (Gleitkomma-Zahlen)
- **DATE / TIME** für Kalender- bzw. Zeit-Daten
- **VARCHAR(*Zeichenanzahl*)** für Texte mit der Anzahl von Zeichen
- **BOOLEAN** für logische / Wahrheits-Werte (**TRUE** und **FALSE**)
- **BLOB** für zusätzliche – nicht genau spezifizierte – Daten (!! ohne feste Speicher-Reservierung)
- **…**

mögliche ***Bedingungen*** sind:

- **PRIMARY KEY** definiert die Schlüssel-Spalte(n)
- **NOT NULL** bestimmt, das in der Spalte immer Daten angebenen werden müssen (ein Leer-Lassen ist nicht zulässig)
- **FORREIGN KEY … REFERENCES** definiert einen Fremd-Schlüssel (siehe 5.4.1.4.) aus / zu einer anderen Tabelle
- **…**

Hinweis(e):
Bei der Definition des Primär-Schlüssels (Schlüssel-Attribut) sowie der Attribute können diverse weitere Bedingungen festgelegt werden. Siehe dazu in der SQL-Referenz.
Für praktische Tabellen emphiehlt sich die Anlage eines eigenständigen Primär-Schlüssel.
Die Änderung einer vorhandenen Tabellen-Struktur ist mit der Anweisung **ALTER TABLE** realisierbar (→ SQL-Referenz).
Oft wird empfohlen für Geld-Beträge den Daten-Typ **DECIMAL(8,2)** zu verwenden, um Rundungs-Problemen aus dem Weg zu gehen. DECIMAL(8,2) bedeutet, dass hier eine Festkommazahl mit insgesamt 8 Ziffern-Stellen bestimmt wird. Von den 8 Ziffernstellen sind 6 Vorkomma- und 2 Nachkomma-Stellen.

Beispiel für die schrittweise Zusammenstellung einer SQL-Anweisung (in diesem "How to"):

Ziel-Tabellen-Struktur:
Personen = (**PID**, Name, Vorname, GebDatum)

nur die schwarz gedruckten Teile des SQL-Ausdrucks werden in den Editor eingegeben!
man kann zu Anfang alle Ausdrucksteile auf einem Zettel mit Beistift schreiben, wegradieren und durch Inhalte ersetzen; gut funktioniert dies, wenn man für jeden Ausdruck (mit zugehörigen Meta-Zeichen) eine Zeile einplant (s.a. EXKURS am Ende des "How to")

CREATE TABLE *Tabellenname* (*Struktur*)

ersetzen der Variable "Tabellenname" durch den (Ziel-)Tabellen-Namen "Personen" und weiter beim Ausdruck "Struktur"

CREATE TABLE Personen **(*Struktur*)**

ersetzen des allgemeinen Ausdruck's "Struktur" durch den verfeinerten Ausdruck "Attribut {, Attribut }"

CREATE TABLE Personen **(*Attribut* { , *Attribut* })**

ersetzen des erstenHilfs-Ausdruck's "Attribut" durch eineSpezifikation aus "Spaltenname", "Datentyp" und ev. noch notwendigen "Bedingungen"

CREATE TABLE Personen **(*Spaltenname Datentyp* [*Bedingung*] { , *Attribut* })**

ersetzen der Ausdrücke "Spaltenname", "Datentyp" und ev. noch notwendigen "Bedingungen" durch die konkreten Angaben für die erste Spalte (1. Attribut)

CREATE TABLE Personen **(** PID INT PRIMARY KEY { , *Attribut* } **)**

nun weiter solange noch Attribute notwendig sind; der Ausdruck "Attribut" wird gleich durch die Konkreten Angaben ersetzt!

CREATE TABLE Personen **(** PID INT PRIMARY KEY, Name VARCHAR(30), **{ , *Attribut* })**

CREATE TABLE Personen **(** PID **INT PRIMARY KEY,** Name **VARCHAR(30),**
Vorname **VARCHAR(30), { , *Attribut* })**

CREATE TABLE Personen **(** PID **INT PRIMARY KEY,** Name **VARCHAR(30),**
Vorname **VARCHAR(30),** GebDatum **DATE)**

da der Nutzer jetzt beliebig viele leere Datensätze ohne Name, Vorname und Geburtsdatum erzeugen kann, sollte über eine Zwangs-Angabe von ev. Name und Vorname mit NOT NULL nachgedacht werden

CREATE TABLE Personen **(** PID **INT PRIMARY KEY,** Name **VARCHAR(30) NOT NULL,**
Vorname **VARCHAR(30) NOT NULL,** GebDatum **DATE)**

5.4.1.4. Erstellen einer Tabelle mit Verknüpfung zu einer anderen Tabelle / Erstellen einer Tabelle mit einer Referenz

```
CREATE TABLE Tabellenname ( Struktur )
```

```
CREATE TABLE Tabellenname (Attribut { , Attribut }
   PRIMARY KEY ( SpaltennameX )
   FOREIGN KEY ( SpaltennameY )
      REFERENCES ReferenzTabellenname( ReferenzSpaltenname )
)
```

```
CREATE TABLE Tabellenname (Attribut { , Attribut }
   PRIMARY KEY ( SpaltennameX )
   FOREIGN KEY ( SpaltennameY )
      REFERENCES ReferenzTabellenname( ReferenzSpaltenname )
   { , FOREIGN KEY ( Spaltenname? )
      REFERENCES ReferenzTabellenname( ReferenzSpaltenname ) }
)
```

Erläuterung:
Die Erzeugung einer Referenz (Fremdschlüssel) zu einer anderen (Referenz-)Tabelle erfolgt immer als Erweiterung einer typischen Tabellen-Definition (→ 5.4.1.3.; einschließlich eines Primärschlüssel's).

Hinweis(e):
Die Fremd-Schlüssel-Spalte sowie die Primär-Schlüssel-Spalte der referenzierten Tabelle müssen den gleichen Datentyp haben.
Die referenzierte Tabelle muss vorhanden sein. Für die Erstellung der Tabellen einer Datenbank ist also mit einer "Bottom up"-Strategie zu arbeiten.

5.4.1.x. einen Datensatz in eine Tabelle eingeben / importieren

5.4.1.x. einen Datensatz aus einer Tabelle entfernen / löschen

5.4.1.x. einen Datensatz in einer Tabelle verändern / modifizieren / einzelne Daten verändern

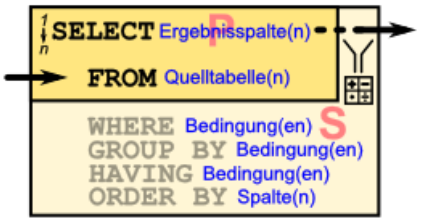
5.4.1.x. Erstellen eines Index

5.4.2. Abfragen

5.4.2.0. Grund-Schema für eine Abfrage

minimales Abfrage-Schema	komplexeres Abfrage-Schema
SELECT … **FROM …**	**SELECT …** **FROM …** **WHERE …** **GROUP BY …** **ORDER BY …**

Das minimale Schema kann immer gleich in das Eingabe-Feld für die SQL-Anweisung reingeschrieben oder reinkopiert werden.
Es empfiehlt sich für Anfänger anhand eines Synthax-Diagramm's (z.B.: → 5.1.11.2.1 Selektion für Fortgeschrittene:) die Wege durchzugehen und die notwendigen Terminale exakt zu übernehmen. Abweichungen von der Pfeil-Richtung sind nicht zulässig.



Die nachfolgende Reihenfolge ist Aufgaben-orientiert. Wenn man mit dem minimalen Abfrage-Schema beginnt, macht man nichts falsch.

5.4.2.1. Daten aus einer Tabelle verwenden

SELECT … FROM *Tabellenname* […]

Erläuterung:
Hinter **FROM** wird die ***Tabelle*** genannt, aus der die Daten für die Selektion stammen sollen.

notwendiger Vorlauf:
Auswahl von Tabellen-Spalten (Attributen)
- → 5.4.2.3. alle Spalten einer Tabelle auswählen / anzeigen ,
- → 5.4.2.4. einzelne Spalten einer Tabelle auswählen / anzeigen (Projektion) ,
- → …

mögliche Fortsetzungen:
- → 5.4.2.7. identische Ergebniszeilen verhindern (Pseudo-Selektion) ,
- → 5.4.2.8. bestimmte Datensätze / Zeilen auswählen (Selektion) ,
- → 5.4.2.14. die Datensätze / Zeilen gruppieren ,
- → 5.4.2.18. die Datensätze sortieren
- → …

5.4.2.2. Daten aus mehreren Tabellen verwenden

SELECT … FROM *Tabellenname* { , *Tabellenname* …} […]

Erläuterung:

Hinter **FROM** folgt eine Komma-getrennte Liste von ***Tabellen***, genannt, aus denen die Daten für die Selektion stammen sollen.

Hinweis(e):
Im **SELECT**-Teil müssen die ***Spalten*** (***Attribute***) eindeutig bezeichnet werden! Zu empfehlen ist immer die Schreibung: ***Tabellenname.Spaltenname*** für jedes einzelne ***Attribut***.

5.4.2.3. alle Spalten einer Tabelle auswählen / anzeigen

SELECT * **FROM** …

SELECT ALL FROM …

Erläuterung:
Das Sternchen (*) hinter **SELECT** umfasst alle (beliebige) Spalten der hinter **FROM** aufgezählten Tabelle. Das Schlüsselwort **ALL** umfasst ebenfalls alle Spalten.

5.4.2.4. einzelne Spalten einer Tabelle auswählen / anzeigen (Projektion)

SELECT ***Spaltenname*** **{ , *Spaltenname* }** **FROM** …

Erläuterung:
Das Sternchen (*) hinter **SELECT** umfasst Komma-getrennten aufgezählten ***Spalten*** der hinter **FROM** aufgezählten Tabelle.

5.4.2.5. einzelne Spalten aus mehreren Tabellen auswählen / anzeigen (Projektion)

SELECT ***Tabellenname.Spaltenname*** **{ , *Tabellenname.Spaltenname* }** **FROM** …

Erläuterung:
Das Sternchen (*) hinter **SELECT** umfasst alle (beliebige) ***Spalten*** (in Punktschreibweise mit Angabe der ***Tabelle***) der hinter **FROM** aufgezählten ***Tabellen***.

5.4.2.6. Spaltennamen in der Ergebnis-Tabelle anpassen

SELECT ***Spaltenname*** **AS** ***neuerName*** **{ , *Spaltenname* AS *neuerName* }** **FROM** …

Erläuterung:
Mit dem Schlüsselwort **AS** kann jeder ***Tabellenspalte*** ein neuer ***Name*** vergeben werden.

Hinweis(e):
Die Hinweise bezüglich der eindeutigen Auswahl von Spalten (s.a. 5.4.5.) gelten weiterhin.

Eine Verwendung der Umbenennung mit AS ist nicht mit der vollständigen Auswahl aller Spalten mittels Sternchen (*****) möglich.

5.4.2.7. identische Ergebniszeilen verhindern (Pseudo-Selektion)

SELECT DISTINCT ***Tabellenname.Spaltenname*** **{ , *Tabellenname.Spaltenname* }**
FROM ***…***

SELECT DISTINCT ***Spaltenname*** **{ , *Spaltenname* }**
FROM ***…***

SELECT DISTINCT * **FROM** ***…***

SELECT DISTICT ALL FROM ***…***

Erläuterung:
Kommen in der Ergebnis-Tabelle Zeilen mit identischen Inhalten heraus, dann werden die Wiederholungen mittels **DISTINCT** unterdrückt und eine Zeilen-reduzierte Ergebnis-Tabelle ausgegeben.

Hinweis(e):
Bei Angabe von ***Spaltennamen*** müssen diese eindeutig einer ***Tabelle*** zuordnetbar sein! Es wird empfohlen bei der Nutzung mehrerer ***Tabellen*** die ***Spalten*** immer mittels Punkt-Schreibweise eindeutig zu charakterisieren.

5.4.2.8. bestimmte Datensätze / Zeilen auswählen (Selektion)

… **FROM** ***…*** **WHERE** ***Bedingung*** **[…]**

… **FROM** ***…*** **WHERE** ***Bedingung*** **{ , *Bedingung* } […]**

… **FROM** ***…*** **WHERE** ***Bedingung*** **{** ***logischeOperation Bedingung*** **} […]**

Erläuterung:
Mittels **WHERE** wird eine Auswahl der Datensätze vorgenommen. Die Datensätze müssen die ***Bedingung(en)*** erfüllen.

typische ***Bedingungen*** sind:

- Vergleiche (z.B.: ***Spaltenname*** **=** 'Meier'); weitere Vergleichszeichen: **<**, **<=**, **>=**, **>** bzw. **!=** (für ungleich)
- Identität-Suche mit **IS** ***Wert*** (z.B.: ***Spaltenname*** **IS** "Meier")
- Bereichs-Suche mit **BETWEEN** ***Bereichsausdruck*** (z.B.: ***Spaltenname*** **BETWEEN** 100 **AND** 1000)
- Suche in einer Menge mit **IN** ***Menge*** (z.B.: ***Spaltenname*** **IN (**'Meier**',** 'Meyer**')**
- Ähnlichkeiten-Suche mit **LIKE** ***Filterausdruck*** (z.B.: ***Spaltenname*** **LIKE** "M??er")

logische Operatoren / Verknüpfungen sind:

- **AND** logische UND-Verknüpfung (beide Bedingungen müssen wahr sein)
- **OR** logische ODER-Verknüpfung (mindestens eine Bedingung muss wahr sein)
- **NOT** logische Negation der nachfolgenden Bedingung / Aussage (aus wahr wird falsch, aus falsch wird wahr)

Filterausdrücke können die folgenden Joker-Zeichen enthalten:

- **%** steht für beliebige Zeichen und Zeichenketten
- **?** steht für genau ein beliebiges Zeichen

Wert kann:

- ein beliebiger Daten-Eintrag (Feld) sein (sachlich ähnlich zu Gleichheit)
- eine leere Zelle / ein leeres Feld sein → **NULL**

ein ***Bereichsausdruck*** ist durch zwei ***Werte*** als Grenzen und das dazwischenliegende **AND** charakterisiert

5.4.2.9. Datensätze zählen (Aggregation)

```
SELECT COUNT(*) FROM …
SELECT COUNT( Spaltenname ) FROM …
SELECT COUNT( Spaltenname ) AS neuerSpaltenname FROM …
```

Erläuterung:
Es werden die Zeilen der Ergebnis-Tabelle gezählt. Im Allgemeinen ist die Angabe eines Namens für die Zähl-Ergebnis-Spalte sinnvoll.

5.4.2.10. das Minimum in einer Spalte ermitteln

```
SELECT MIN( Spaltenname ) FROM …
SELECT MIN( Spaltenname ) AS neuerSpaltenname FROM …
```

Erläuterung:
Es wird der kleinste Wert / Inhalt / das **Minimum** in der ***Spalte*** der Ergebnis-Tabelle ermittelt. Im Allgemeinen ist die Angabe eines Namens für die Ergebnis-Spalte (mit dem Minimum) sinnvoll.

5.4.2.11. das Maximum in einer Spalte ermitteln

```
SELECT MAX( Spaltenname ) FROM …
SELECT MAX( Spaltenname ) AS neuerSpaltenname FROM …
```

Erläuterung:
Es wird der grösste Wert / Inhalt / das **Maximum** in der ***Spalte*** der Ergebnis-Tabelle ermittelt. Im Allgemeinen ist die Angabe eines Namens für die Ergebnis-Spalte (mit dem Maximum) sinnvoll.

5.4.2.12. den Durchschnitt / Mittelwert einer Spalte ermitteln

```
SELECT AVG( Spaltenname ) FROM …
SELECT AVG( Spaltenname ) AS neuerSpaltenname FROM …
```

Erläuterung:
Es wird der **Durchschnitt** / **arithmetrische Mittelwert** der ***Spalte*** der Ergebnis-Tabelle ermittelt. Im Allgemeinen ist die Angabe eines Namens für die Ergebnis-Spalte (mit dem Durchschnitt) sinnvoll.

5.4.2.13. die Werte einer Spalte summieren

```
SELECT SUM( Spaltenname ) FROM …
SELECT SUM( Spaltenname ) AS neuerSpaltenname FROM …
```

Erläuterung:
Es wird der **Durchschnitt** / **arithmetrische Mittelwert** der ***Spalte*** der Ergebnis-Tabelle ermittelt. Im Allgemeinen ist die Angabe eines Namens für die Ergebnis-Spalte (mit der Summe) sinnvoll.

5.4.2.14. die Datensätze / Zeilen gruppieren

```
… FROM […] … GROUP BY Spaltenname […]
… FROM […] … GROUP BY Spaltenname { , Spaltenname } […]
```

Erläuterung:

Dannach können z.B. noch Sortierungen folgen → .

5.4.2.15. die Datensätze / Zeilen in einer Gruppe zählen

```
SELECT COUNT(*) FROM … [ WHERE … ] GROUP BY Spaltenname
SELECT Spaltenname { , Spaltenname }  COUNT(ZählSpaltenname)
FROM … [ WHERE … ] GROUP BY ZählSpaltenname
```

Erläuterung:
Mit **COUNT()** gleich hinter **SELECT** wird eine Ergebnis-Tabelle für alle (wegen Sternchen (*****)) bzw. die angegebenen ***Spalten*** mit einer Zähl-Spalte angelegt angelegt. Diese ist wird nach gleichartigen Einträgen in der genannten ***Spalte*** (hinter **GROUP BY**) gruppiert.

5.4.2.16. die Werte einer Gruppe summieren

```
SELECT COUNT(*) FROM … [ WHERE … ] GROUP BY Spaltenname
SELECT Spaltenname { , Spaltenname } COUNT(ZählSpaltenname)
FROM … [ WHERE … ] GROUP BY ZählSpaltenname
```

Erläuterung:
Mit **COUNT()** gleich hinter **SELECT** wird eine Ergebnis-Tabelle für alle (wegen Sternchen (*****)) bzw. die angegebenen ***Spalten*** mit einer Zähl-Spalte angelegt angelegt. Diese ist wird nach gleichartigen Einträgen in der genannten ***Spalte*** (hinter **GROUP BY**) gruppiert.

5.4.2.17. in einer Gruppe von Datensätzen / Zeilen den Mittelwert ermitteln

```
… FROM […]  …
```

Erläuterung:

5.4.2.18. die Datensätze sortieren

```
SELECT … FROM … WHERE … SORT BY Spaltenname Bedingung
SELECT … FROM … WHERE …
   SORT BY Spaltenname Bedingung { , Spaltenname Bedingung }
```

Erläuterung:
Die Definition einer Sortierung erfolgt praktisch am Ende der SQL-Anweisung. Neben der ***Spalte*** – nach der die Ergebnis-Tabelle sortiert werden soll – kann als ***Bedingung*** die Sortier-Richtung angegeben werden. Es sind mehrere (gestaffelte) Sortierungen möglich.

Hinweis(e):
Bedingung: Es wird eine aufsteigende Sortier-Richtung angenommen (, diese kann aber auch mit angehängtem **ASC** extra bestimmt werden). Die umgekehrte Sortier-Richtung erreicht man mit angehängtem **DESC**.

HAVING

als "Alternative" zu WHERE

5.4.2.x. zwei Tabellen vereinen / einen inneren Verbund zwischen zwei Tabellen herstellen

JOIN

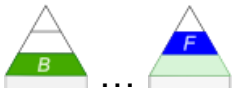
Exkurs: SQL-Anweisungen erstellen für Nicht-Affine

Dieser Exkurs ist vorrangig für solche SQL-Anwender gedacht, denen die hintereinander weggeschriebenen Statements zu unübersichtlich sind. Mit der folgenden Arbeitstechnik kann man sich als nicht so Affiner das Leben etwas leichter machen. Wir greifen auf die gleichen Definitionen zurück, die wir gerade im "How to" aufgezeigt haben. Als erstes Arbeitsmittel verwenden wir ein Blatt Papier, einen Radierer und einen Bleistift. Wer will kann am Schluß die fertigen Statements mit einem kräftigeren Stift nachschreiben.
Zuerst suchen wir uns die Definition für unser zu bearbeitendes Problem heraus. So könnte unser Problem sein, Name und Vorname aus einer Personen-Tabelle herauszusuchen, wobei nur die Namen zwischen M und R dabei sein sollen. Die Ergebnistabelle soll umgekehrt nach den Namen sortiert werden.
Statt es nun so auch auf's Papier zu schreiben, zerlegen wir es in sinnvolle / syntaktisch passende Zeilen.

Unser erstes Problem ist also das Anzeigen Daten aus einer Tabelle → 5.4.2.1. Wir übernehmen also die dort vorgeschlagene Struktur (s.a. Abschnitts-Anfang) Zeilen-weise auf's Papier.	SELECT FROM ***Tabellenname***
Tabellenname ist von uns zu ersetzen, also radieren wir Tabellenname weg und notieren dafür: Personen Als nächstes war angegeben, dass nur einzelne Spalten angezeigt werden sollen. Dafür ist 5.4.2.4. geeignet.	SELECT FROM Personen
Natürlich könnte man auch gleich bei 5.4.2.4. starten. Die notwendigen Änderungen liegen zwischen SELECT und FROM, also tragen wir sie dort ein. Zuerst einmal als Meta-Ausdruck.	SELECT ***Spaltenname*** **{** ***, Spaltenname*** **}** FROM Personen
Diesen können wir wegradieren und durch die geforderten (anzuzeigenden) Spalten ersetzen. An dieser Stelle können wir das SQL-Statemant auch schon mal ausprobieren.	SELECT Name, Vorname FROM Personen
Ev. offenbaren sich hier Tipp-Fehler oder falsche Bezeichnungen.	
Die Auswahl von Datebnsätzen wird im Punkt 5.4.2.8. beschrieben. Es handelt sich um eine WHERE-Erweiterung hinter dem FROM-Abschnitt. Diesen Konstrukt übernehmen wir wieder.	SELECT Name, Vorname FROM Personen WHERE ***Bedingung***
Nun suchen wir etwas in einem Bereich. Dazu kann man z.B. den Bedingungs-Ausdruck mit BETWEEN benutzen. Dieser ersetzt den rauszuradierenden Bedingungs-Ausdruck. Das Einrücken dient einer übersichtlicheren Darstellung der zusammengehörenden Strukturen und einem leichteren Herausradieren.	SELECT Name, Vorname FROM Personen WHERE ***Spaltenname*** **BETWEEN** ***Wert*** **AND** ***Wert***
Auch die beiden Wert-Audrücke müssen noch konkretisiert werden. D.h. wir geben die untere und obere Grenze – also M und R an. →	SELECT Name, Vorname FROM Personen WHERE Name BETWEEN ***Wert*** **AND** ***Wert***

Da bis hierhin alle Ersatz-Ausdrücke aufgelöst sind, kann man auch wieder einen Versuch und das Statement testen. Würde das Zwischen-Ergebnis nicht passen, dann wäre ein neuer Ansatz zu empfehlen.	SELECT Name, Vorname FROM Personen WHERE Name BETWEEN 'M' AND 'R'
Bleibt als letzte Forderung die Sortierung nach dem Namen. Der Blick in die Lösungs-Vorschrift 5.4.2.18. besagt ein Anhängen der SORT BY-Anweisung an den WHERE-Bedingungs-Teil. Die Bedingung gehört zwar syntaktisch gleich hinter den Spaltennamen. Dann würden wir aber zwei zu radierende Ausdrücke gleich hintereinander haben. Da könnte es Überschneidungen geben.	SELECT Name, Vorname FROM Personen WHERE Name BETWEEN 'M' AND 'R' SORT BY ***Spaltenname*** ***Bedingung***
Zuerst ersetzen wir Spaltenname. Da nach den (Personen-)Namen sortiert werden soll ist das schnell realisiert. SORT BY lässt auch noch Bedingungen zu. Das sind die Sortier-Reihenfolgen / -Richtungen. Eine normale – aufsteigende – Sortierung bräuchten wir nicht weiter zu spezifizieren. Wer will kann auch betont ein ASC angeben.	SELECT Name, Vorname FROM Personen WHERE Name BETWEEN 'M' AND 'R' SORT BY Name ***Bedingung***
Die umgekehrte – absteigende – Sortierung muss aber unbedingt durch DESC gekennzeichnet werden. Somit wären wir fertig. Das Statement kann original so in den SQL-Editor übertragen werden. Diesen stören Zeilen-Umbrüche und Einrückungen nicht. Der hintereinanderweg geschriebene Text funktioniert genauso, ist aber deutlich schwerer zu lesen.	SELECT Name, Vorname FROM Personen WHERE Name BETWEEN 'M' AND 'R' SORT BY Name DESC
SELECT Name, Vorname FROM Personen WHERE Name BETWEEN 'M' AND 'R' SORT BY Name DESC	
Auch weitere Änderungen / Verbesserungen / Erweiterungen lassen sich an dem Zeilen-strukturierten Ausdruck deutlich besser vornehmen.	

5.5. Anleitung zum Lesen / Interpretieren von SQL-Anweisungen



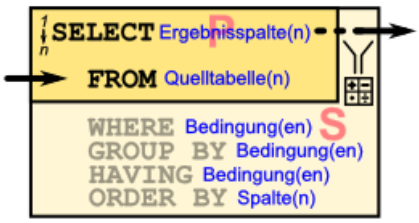
Eigentlich sollte die Struktur von SQL auch wenig geschulte (englisch-sprachige) Anwender in die Lage versetzen, den Inhalt bzw. den Zweck eines Ausdruck's zu verstehen. Das sollte zum Einen für das Lesen von SQL-Statement's gelten, als auch für die Erstellung. Eine vorgebildete (englisch-sprachige) Sekretärin oder Sachbearbeiterin sollte SQL-Ausdrücke formulieren können. Mit der immer größer werdenden Komplexität heutiger Datenbanken und der Sprache SQL hat dieser Anspruch schon deutlich gelitten. Mit den "Problemen" der Erstellung haben wir uns ja schon im Abschnitt (→) beschäftigt.
Hier wollen wir nun einen Ansatz liefern, wie man SQL-Statement's in verständliche – mehr oder weniger umgangs-sprachliche – Aussagen umsetzen kann.

> Wenn manche Lehrer wissen würden,
> wie schwer ihre Aufgaben für Schüler sind
> und wenn man diese ohne ihr eigenes Zusatzwissen
> bzw. vorgegebene Lösungs-Blätter lösen muss,
> dann würden sie diese nicht so stellen.

Besteht die Möglichkeit, ein SQL-Statement an der originalen Datenbank ablaufen zu lassen, dann hilft das ungemein. Aus dem Vergleich der Ausgangs-Tabellen und der Ergebnis-Tabelle (der Abfrage) lassen sich viele Wirkungen von speziellen SQL-Ausdrücken viel leichter erkennen.
Ein SQL-Statement ohne Datenhintergrund zu verstehen ist z.T. wirklich schwierig.
Praktisch tauchen Probleme mit (normalen) SQL-Ausdrücken nur im Zusammenhang mit Abfragen auf. Andere Anweisungen sind i.A. gut zu lesen und zu verstehen.
Wir beschränken uns hier deshalb auf die Zerlegung / Analyse von Abfragen. Das Grund-Prinzip lässt sich aber auch auf andere SQL-Strukturen übertragen.

5.5.1. Strukturieren der SQL-Anweisung / Abfrage

Zuerst schreiben wir die SQL-Anweisung so in mehrere Zeilen um, dass immer ein Schlüsselwort vorne steht. In guten SQL-Programmen und von umsichtigen SQL-Anwendern erstelle SQL-Statements sind das vielleicht schon. I.A. kann man im Editor (der DB-Management-Software) die SQL-Anweisungen mit [Enter] ohne Gefahr "umgestalten".
Solche Strukturierungen ignoriert die Datenbank und für uns Menschen werden die Ausdrücke deutlich besser lesbar. Besonders bei **AS**-Ausdrücken ist das zuerste sehr hilfreich. Sie müssen vielfach erst ganz zum Schluß mit einbezogen werden.
Auch Einrückungen mit Leerzeichen für untergeordnete Strukturen werden von den meisten Editoren "überlesen".



originaler Beispiel-SQL-Ausdruck:

SELECT FROM WHERE

umstrukturierter Beispiel-SQL-Ausdruck:

SELECT
FROM
WHERE

5.5.2. "Übersetzen" der Ausdrücke

Im zweiten Schritt übersetzen wir die einzelnen Zeilen so, dass die Funktion verständlich wird. Vielfach ist das ja schon von sich aus leicht möglich. Bei einigen Strukturen müssen wir aber Obacht geben.

Übersetzungs-Hilfen /-Floskeln / …

SQL-Ausdruck	Übertragung / Übersetzung	Bemerkungen

Übersetzung des Beispiel-SQL-Ausdruck's

Versuchen wir nun diese allgemeinen Formulierungen auf unser konkretes Beispiel anzuwenden.

SQL-Ausdruck	Übertragung / Übersetzung	Bemerkungen
SELECT		
FROM		

5.5.3. Abgleich der Angaben mit der Datenbank-Struktur / Ableiten der Teil-Struktur der Datenbank

5.5.4. Erstellen einer fach- oder umgangs-sprachlichen Beschreibung

6. Datenbanken - Programmierung

Die Programmierung von und mit Datenbanken gehört heute zu den wichtigsten Aufgaben der Programmierer für Anwender- und Internet-Software.

Grundsätzlich unterscheidet man zwei grundsätzlich verschiedene Programmierungs-Arten. Da ist zum Einen die Erstellung von Software-Lösungen aus dem Datenbank-System selbst heraus. Dabei wird z.B. eine im System enthaltene Makro- und / oder Programmiersprache genutzt. Microsoft ACCESS gehört z.B. zu diesem Programmier-Typ. Mit der Programmier-Umgebung lassen sich alle Komponenten wie Tabellen, Abfragen, Formulare, Berichte und auch die Makros nutzen. Für den Anwender entsteht der Eindruck, er arbeitet mit einem speziellen Programm. Die Datenbank als solche ist für den Nutzer kaum sichtbar.
Die zweite Art ist die Programmierung einer Software mit quasi einer frei gewählten Programmiersprache. Die relativ komplexe Daten-Verwaltung lagert der Programmierer an ein Datenbank-System aus. Über eine spezielle Software-Schnittstelle für die gewählte Programmiersprache kann auf eine oder mehrere Datenbank-Schnittestellen zugegriffen werden. Die meisten verwenden **SQL** (→ 5. SQL – die Datenbank-Sprache). Vereinfacht kann man sich das so vorstellen, dass ein spezieller Output-Befehl – nennen wir in hier **writeSQL** genutzt wird, um der Datenbank etwas mitzuteilen. Mit einem äquivalenten Input-Befehl – hier z.B. **readSQL** – fragt man bei der Datenbank an und erhählt auswertbare Daten für das zu erstellende Programm.
Quasi eine Kombination aus beiden Varianten stellen Web-Datenbanken und zugehörige Applikationen dar. Sie basieren auf Datenbanken unterschiedlichster Art. Das geht bei einfachen CSV- und XML-Dateien los und geht weiter bis zu SQL- und NoSQL-Datenbanken mit professioneller Ausrichtung.
Zur Programmierung der Applikationen werden Script-Sprachen, wie JavaScript, PHP, Perl oder Ruby genutzt. Seit einigen Jahren nimmt auch die Programmierung über HTML5 immer schnellere Fahrt auf.
Wir werden uns zuerst ganz kurz die erste Variante an. Sie ist immer sehr speziell, da die Programmiersprache des Datenbank-System oder z.B. der Office-Suite benutzt wird (→ 6.1. Datenbank-Programmierung mit ACCESS-VBA). Hier gehen wir auch auf die standardisierten Schnittstellen (→ 6.2. universelle Datenbank-Schnittstellen) – z.B. eben ODBC – ein (→ 6.2.1. die ODBC-Schnittstelle).
Die zweite Herangehensweise mit den klassischen Programmiersprachen, wie z.B. Python (ausführlich: → 🕮 **Programmieren mit Python**) und JAVA (ausführlich:→ 🕮 **Strukturierte und Objekt-orientierte Programmierung mit JAVA**) folgen dann anschließend. Für die meisten Kurse wird sicher nur eine der beiden Sprachen Python (→ 6.3. Datenbank-Zugriff mit Python) oder JAVA (→ 6.4. Datenbank-Zugriff mit JAVA) interessant sein.
In einem gesonderten Abschnitt folgen einige Ausführungen zu den Web-Datenbanken und –Applikationen (→ 7. Web-Datenbanken). Hier ist das Feld riesig. Eine weit verbreitete Kombination von mehreren aufeinander abgestimmten Programmen ist XAMPP (→ 7.1. Arbeiten mit XAMPP). Diese lässt sich auch über den IoStick (→ https://tinohempel.de/info/info/IoStick/index.html) von T. HEMPEL nutzen.

6.0. Vorüberlegungen / Grundlagen

In den konzeptionellen Besprechungen (→ 2.1.2. übergreifende / übergeordnete Modelle) haben wir schon mehrere Schicht- oder Ebenen-Modelle vorgestellt. Sie dienten immer einer passenden Modellierung bzw. der Abtrennung von Aufgaben / Funktion.
Will man eine Datenbank programmieren, dann wird man bald von der Komplexität der Aufgaben überwältigt. Hier hat sich, wie fast überall in der Informatik und speziell in der Programmierung eine Aufteilung von Aufgaben in Schichten (sogenannten Tier) bewährt. Man spricht auch von einer Schicht-Architektur. Selbst einzelne Schichten sind meist noch so komplex, dass weitere Aufteilungen erfolgen müssen.
Programmierer und Techniken konzentrieren sich auf einzelne Schichten und füllen diese entsprechend aus. Für Übergänge zu anderen Schichten, werden Schnittstellen geschaffen. Die Kommunikation zwischen den Schichten läuft nur über diese Schnittstellen. Da jede Schicht für sich eigenständig gehändelt werden, lassen sie sich gut testen, ersetzen, aktualisieren oder auch völlig neu konzeptieren.
Die Anwender / Nutzer der Schichten brauchen nur die Schnittstellen kennen, die üblicherweise auch ausführlich dokumentiert sind. Wie die Programmierer oder Techniker eine Schicht intern realisieren, bleibt deren Kompetenz.

Single-Tier

In Single-Tier-Programmen werden alle Funktionen in einem Komplex realisiert. Das macht bei kleinen und sehr speziellen Lösungen Sinn. Für größere Projekte ist eine Aufteilung der Aufgaben in spezielle Schichten unanbdingbar.

Two-Tier

Zu den Zwei-Schicht-Architekturen gehören die klassischen Client-Server-Systeme. Client's und Server stellen dabei eigene Schichten dar. Die Client's stellen Anfragen an die Server. Diese ermitteln eine Antwort und senden diese an die Client's zurück.
In Datenbank-Systemen übernehmen die Server die Aufgaben der Daten-Haltung (Speicherung), Administration und Daten-Zusammenstellung. Die Client's übernehmen die zusammengestellten Rohdaten und bereiten sie für die Nutzer auf. Das ist zumeist die Fein-Darstellung (Formate, …) und Präsentation (Diagramme, Tabellen, …).
Client und Server müssen keine eigenständigen Rechner sein. Sie können als Software auf einem Rechner nebeneinander funktionieren.

Three-Tier

Die klassische Drei-Schicht-Architektur umfasst die Schichten:

- **data-server tier** **back end** **Datenhaltungs-Schicht**	enthält die eigentliche Datenbank (gespeicherte Daten) übernimmt das Indexieren der Datenbestände realisiert den Datenschutz beinhaltet die Datensicherung stellt (große Mengen von) Daten zur Verfügung
- **application-server tier** **middle tier; business tier** **enterprice tier** **Logik-Schicht**	verknüpft die (großen) Daten-Mengen sinnvoll miteinander beinhaltet die eigentliche Daten-Verarbeitung
- **client tier** **front end** **Präsentations-Schicht**	verantwortlich für die Nutzer-gerechte Darstellung der Daten realisert Daten-Ein- und –Aus-gabe

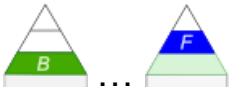
Jedwede Kommunikation und Daten-Transporte erfolgen immer über die – in der Mitte liegende – Logik-Schicht (middle-tier)
Client-Tier ← → Middle Tier ← → Database Tier

Aufgaben:

1. *Welche Schichtungen / Schicht-Modelle der Informatik kennen Sie? Stellen Sie diese ganz kurz vor!*
2. *Informieren Sie sich unter:*
   http://www.dfpug.de/konf/konf_1998/09_tier/d_tier/d_tier.htm
   über die Vor- und Nachteile von Tree-Tier-Programm-Systemen! Notieren Sie sich die Informationen in kompakter Form!
3.

6.1. Datenbank-Programmierung mit ACCESS-VBA

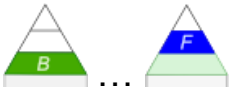
Die Nutzung von Datenbank-Komponenten aus Office-Produkten wird im neuen Rahmenplan "Informatik" (Sek.II; für MV) abgelehnt.
Für Einsteiger-Projekte sind sie aber sehr gut geeignet!



VBA ist die BASIC-artige Programmiersprache der Office-Programme von Microsoft-Office. Dabei handelt es nicht etwa um eine billige BASIC-Variante, sondern um ein sehr Leistungsfähiges Programmier-System.
Das gesamte System zu besprechen, würde dieses Skript bei weitem sprengen. Da gibt es sehr gute Spezial-Literatur. Viele Bücher besprechen nur einzelne Teil-Sprachen für eines der Office-Produkte. Man braucht auch nicht immer unbedingt ein Buch für die aktuellste Version. So ein, zwei Versionen zurück betrachtet, hat sich meist nicht viel getan. Für spezielle Leistungen, der erst in der aktuellsten Version dazugekommen sind, ist natürlich dann auch nur die aktuellste VBA-Version interessant.
Für die älteren Versionen erhält man oft in speziellen online-Buchhandlungen (z.B. terrashop.de oder Jokers.de) stark verbilligte Bücher. Diese sind oft als Mängelexemplare deklariert, aber nicht wirklich dieser Bezeichnung wert.

6.1.1. Arbeiten mit Makro's

Die Nutzung von Datenbank-Komponenten aus Office-Produkten wird im neuen Rahmenplan "Informatik" (Sek.II; für MV) abgelehnt.
Für Einsteiger-Projekte sind sie aber sehr gut geeignet!



Ein guter Einstieg ist die Benutzung von Makro's in einem Office-Programm. Makro's sind Sequenzen von Tätigkeiten in einem Office-Programm, die man sehr häufig braucht.
Der erste Zugang ist vielleicht, ein vorhandenes Makro zu nutzen. Über eine einfache Tasten-Kombination wird dann eine Sequenz von Befehlen in einem Ritt abgearbeitet. Mit der Schrittfolge hat der einfache Nutzer nichts direkt zu tun.
Irgendwann erstellt man ein erstes Makro. Dabei werden z.B. Maus- oder Tasten-Befehle aufgezeichnet. Das Makro erhält einen Namen und man kann ihm eine bestimmte Tasten-Kombination zuordnen. Immer wenn dann später in dem Programm diese Tasten-Kombination gedrückt wird, kommt es zur Abarbeitung der aufgezeichneten Befehls-Folge.
Die Befehl-Folge kann man sich ansehen und sie auch manipulieren / ändern. Das Ansehen bringt schon ein gewisses Verständnis für die Arbeitsweise des VBA-Systems. Kleine Änderungen lassen sich dann auch noch gut ausprobieren.
Das ist quasi die dritte Nutzungs-Ebene von Makro's. Die vierte Ebene schließt dann das Erstellen eines Makro's bzw. eines Programm's über einen Text-Editor ein. Einen solchen findet man im Office. Die höchsten Weihen sind dann Datenbank-Anwendungen, bei denen ein unbedarfter Nutzer gar nicht merkt, dass es sich um eine Office-Anwendung handelt.

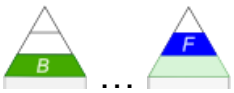
6.2. universelle Datenbank-Schnittstellen

Aufgabe von Datenbank-Schnittstellen ist es, den inneren Aufbau und irgendwelche Implementierungs-Details vor fremden Anwendungen und Anwendern zu verstecken. Die Schnittstelle bietet einen standardisierten Zugriff auf das Datenbank-Management-System (DBMS).
Diese Schnittstelle wird mehr oder weniger offen gestaltet und veröffentlicht. Durch die Nutzung der Schnittstelle können die eigene Anwendung und das DBMS völlig unabhängig entwickelt werden, die Funktionalitäten sind stabil und dauerhaft über die Schnittstelle definiert.
Für die eigene Anwendung ist es nicht wichtig, wie und wo die Daten gespeichert sind. Das ist und bleibt die Domäne des Datenbank-Management-Systems.
Eigentlich jedes DBMS bietet eine eigene Schnittstelle für Programmierer (auch die aus der eigenen Firma und eigene Datenbank-Anwendungen) an. Diesen bieten den maximalen Nutzeffekt. Man kann praktisch auf alle Leistungen des DBMS zurückgreifen.
Dieser Vorteil wird dann zum Nachteil, wenn Nutzer andere DBMS benutzen wollen oder müssen. Das können z.B. alte Daten-Bestände sein oder finanzielle Gründe. Heute gibt es schließlich viele DBMS auch in abgespeckter Version für freie Zwecke.
Die meisten “normalen“ Anwendungen benötigen gar nicht den vollen Funktions-Umfang einer Hochleistungs-Datenbank. Vielmehr ist in Zeiten vielen Unbeständigkeiten (Pleiten, Aufkäufe, …) eine Kontinuität, Ersetzbarkeit und Flexibilität gefragt. Dafür wurden schon frühzeitig sehr universelle Schnittstellen erschaffen, von denen wir hier die ODBC-Schnittstelle (→ 6.2.1. die ODBC-Schnittstelle) ausführlicher vorstellen werden.
Als Gegen-Part haben Borland, IBM, Novell und WordPerfect Corp. die IDAPI-Schnittstelle (Integrated Database Application Programming Interface) entwickelt.
Weitere – zum Teil auch breiter angewendete – Schnittstellen sind BDE (Borland Database Engine), die Perl DBI (Perl Database Independet) und für microsoft Visual BASIC die ADO (ActiveX Data Objects).

6.2.0. Grundlagen der Datenbank-Schnittstellen

Im Allgemeinen benötigt man bei Datenbanken zwei unterschiedliche Zugriffe. Das ist zum Einen die Kontakt-Aufnahme zur Datenbank einschließlich der Authentifizierung und zum Zweiten die eigentliche Datenbank-Arbeit.
Man kann sich den Verbindungs-Aufbau wie ein Stecker-System vorstellen. Wenn der Stecker passt, dann können die Daten über die Kontakte ausgetauscht werden.

6.2.1. die ODBC-Schnittstelle



Die ODBC-Schnittstelle ist eine der verbreitesten Schnittstellen-Definition. Sie basiert auf SQL und wurde von Microsoft entwickelt. ODBC steht dabei für Open Datenbase Connectivity.
Viele Datenbank-Management-Systeme bieten eine entsprechende Schnittstelle an. Man kann deshalb ODBC auch als einen allgemeingültigen Standard betrachten.
Wie schon oben schon kurz beschreiben, wird bei ODBC eine Trennung zwischen der Verbindung und dem Daten-Austausch vorgenommen.
Bei der Verbindungs-Definition wird ein Client auf dem Nutz-System eingerichtet. Dieser enthält alle – z.T. auch sehr sensiblen – Daten für die Verbindungs-Aufnahme zur Datenbank. Dazu gehören z.B. der Speicherort oder die Zugangs-Daten.
Über den – meist nur einmalig einzurichtenen – Client kann die Nutzer-Software die Verbindung zur – irgendwo liegenden – Datenbank aufnehmen und dann Daten austauschen.
Der – später folgende – eigentliche Daten-Austausch erfolgt über SQL-Anweisungen.

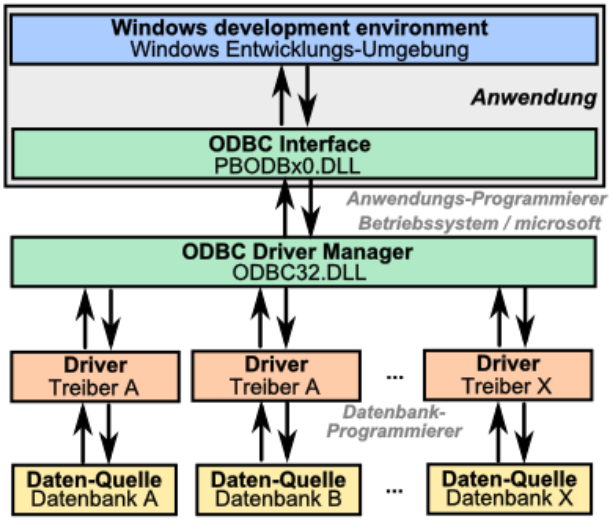
Funktionalitäts-Stufen bei ODBC

- **Core** beinhaltet die Basis-Funktionalität
- **Level 1**
- **Level 2**

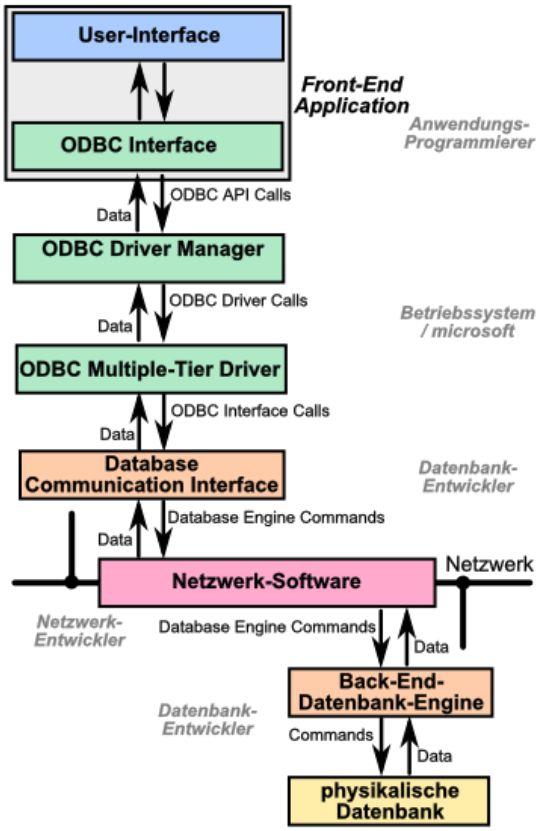
besteht aus "Front-end"- oder Client-Treiber für die Applikations-/Nutzer-Seite und einem Treiber für das DBMS – dem sogenannten "Back-end"- oder Server-Treiber

die Front-end-Seite wird auch ODBC-Client genannt
daneben gehören noch der ODBC-Server und der DBMS-Server zu den drei notwendigen Komponenten für ein funktionierendes ODBC-System
die beiden letzten Komponenten werden typischerweise vom DBMS installiert bzw. bereitgestellt

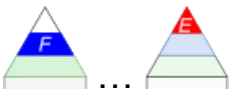
der ODBC-Server gehört aber zu den ODBC-Teilen, die microsoft entwickelt hat, nur der DBMS-Server muss vom Entwickler des DBMS zur Verfügung gestellt werden
für den praktischen Anwender ist eigentlich nur die Einrichtung des ODBC-Clients wichtig, da auf modernen System die ODBC-Treiber meist schon installiert sind



Die Netzwerk-Software beinhaltet dann die ISO/OSI-Schichten. Die Anzahl der genutzten Schichten ist also schon gewaltig. Umso überraschender ist es, dass die Kommunikation doch so reibungslos abläuft. Das spricht für die Qualität der Schnittstellen und der dahinter liegenden Programme.

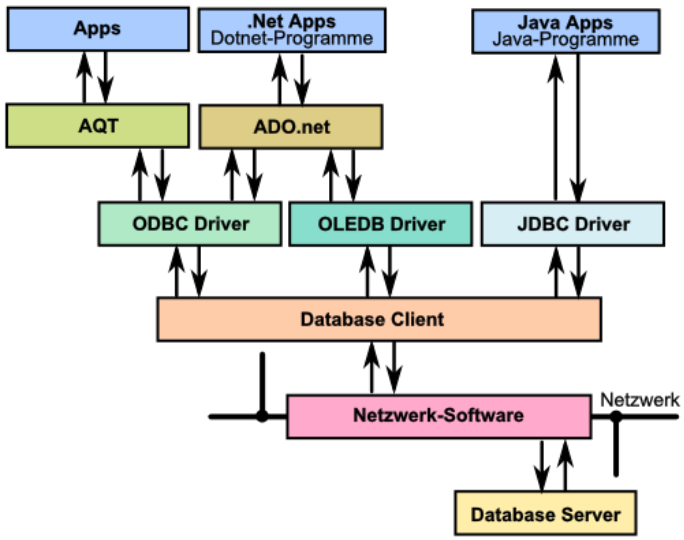


6.2.2. weitere offline-Datenbank-Schnittstellen



den Begriff offline sollte man hier nicht so ernst nehmen
gemeint sind hier Schnittstellen, die für direkte Computer-Verbindungen gemacht worden sind

moderne Fortsetzung / Weiterentwicklung ist die OLEDB, die mit dem dotNet-Framework in Windows-Systeme einzug gehalten hat
praktisch parallel zum ODBC-Treiber



IDAPI (Integrated Database Application Programming Interface) wurde als Alternativ- bzw. Kongurrenz-Produkt von Borland, IBM, Novell und Lotus/Wordperfect entwickelt (1992)

ähnlicher Aufbau und Prinzip
verbesserte Funktionalität, kann z.B. auf mehrere / unterschiedliche Datenbanken gleichzeitig zugreifen
hohe Performanz
in der Praxis ist IDAPI eher eine Erweiterung von ODBC als ein eigener Standard

bietet auch Konnektivität zu Datenbanken, die auf dem Standard xbase basieren

6.2.3. weitere online-Datenbank-Schnittstellen

hier verstehe ich solche Schnittstellen, die auf Internet-Verbindungen basieren
das dürfen natürlich auch interne TCP/IP-Verbindungen sein

6.2.3.x. CKAN

Beispiel:
SQL-Ausdruck

```
SELECT bezeichnung, strasse_name, hausnummer, hausnummer_zusatz
FROM "0c2c4996-afad-4ebe-9a3d-a3a7d7047a4d"
WHERE ST_Within(ST_Transform(geometrie,25833), ST_Buffer(ST_Transform((
    SELECT geometrie
    FROM "e614d725-4fad-4447-820d-4475ca55cff5"
    WHERE bezeichnung = 'Deutsche Med'),25833),300))
```

passender CKAN-Aufruf:
https://www.opendata-hro.de/api/action/datastore_search_sql?sql=SELECT bezeichnung, ⮐
strasse_name, hausnummer, hausnummer_zusatz ⮐
FROM "0c2c4996-afad-4ebe-9a3d-a3a7d7047a4d" WHERE ⮐
ST_Within(ST_Transform(geometrie,25833),ST_Buffer(ST_Transform((SELECT ⮐
geometrie FROM "e614d725-4fad-4447-820d-4475ca55cff5" WHERE bezeichnung =⮐
'Deutsche Med'),25833),300))

Links:
http://docs.ckan.org/en/ckan-2.7.3/api (Beschreibung / Dokumentation der API (engl.))

6.3. Datenbank-Zugriff mit Python

Bei Python steht mehr der schnelle Skript-orientierte Zugriff auf Datenbanken im Vordergrund. Natürlich kann man auch Fenster- und / oder Objekt-orientierte Programme erstellen.

6.3.0. Vorarbeiten / Installation

Eine der Bibliotheken für den Datenbank-Zugriff unter Python wird Connector genannt

für MySQL z.B. unter http://dev.mysql.com/downloads/connector/python/ zu downloaden
vorher Python-Version erkunden

import sys
print(sys.version)

od. bei Start des Python-Systems in der obersten Zeile ablesen
passende Version von der Download-Seite runterladen

mit Admin-Rechten installieren

6.3.1. Vorbereitung

Um eine Datenbank nutzen zu können, müssen einige Vorarbeiten getätigt werden. Dabei handelt es sich im Wesentlichen um den Verbindungs-Aufbau und die Authentifizierung bei der Datenbank.

```
import mysql.connector # neue Bibliothek

Servername = 'localhost' # Rechnername (localhost ist der eigene Rechner)
Benutzer   = 'mustermann'  # Nutzeraccount bei der Datenbank
Passwort   = 'sehrgeheim'  # passendes Passwort
Datenbank  = 'InfoDB'      # Name der Datenbank, die genutzt werden soll

# Verbindung mit der Datenbank
con = mysql.connector.connect(host=Servername)
con.cmd_change_user(username = Benutzer, password = Passwort)
con.database = Datenbank
```

In unserem obigen Code-Schnipsel ist con die Variable, die für eine konkrete Datenbank-Anbindung steht. Über diese Variable läuft dann später der gesamte Daten-Verkehr zwischen der eigenen Applikation (mit den internen Daten-Strukturen) und der Datenbank (mit ihrer internen Daten-Struktur).

6.3.1.1. Zugriff auf entfernte Datenbanken

Sachlich ist die Anbindung einer externen Datenbank – irgendwo im Internet – nicht anderes, als die Verbindung zu einer lokalen. Ändern tut sich eigentlich nur die Host-Adresse.

```
con = mysql.connector.connect(host=Servername)
```

Port 3306 muss ev. bei Firewall's freigegeben werden

```
con.cmd_change_user(username = Benutzer, password = Passwort)

con.database = Datenbank
```

oder alles kombiniert:

```
con = mysql.connector.connect(
            host     = Servername,
            user     = Benutzer,
            password = Passwort,
            database = Datenbank )
```

6.3.2. Daten-Zugriff über SQL-Befehle

Für das Arbeiten mit den Daten einer Datenbank braucht man nur die Verbindung. Ob die Datenbank lokal oder entfernt ist, bleibt für den Nutzer verborgen. Nur wenn sein Netzwerk oder Internet nicht funktioniert, wird er Probleme mit dem Arbeiten bekommen. Dann läuft praktisch nichts mehr. Gute Datenbank-Anwendungen werden die Datenbank oder Teile davon z.B. in der Nacht repliziert haben und als lokale Datenbank irgendwo lokal abgelegt haben. Dann kann der Nutzer mit dieser Datenbank arbeiten und in der nächsten Replikations-Phase werden Änderungen zwischen den entfernten und lokalen Datenbank abgeglichen.
Das verdeutlicht die Vorteile der Trennung von Verbindungs-Aufbau und dem eigentlichen Daten-Austausch.
Es muss aber am Schluß unbedingt die Verbindung zur Datenbank noch geschlossen werden! (→ 6.3.4. Nachbereitung)

6.3.2.1. Daten-Abfrage

```
# SQL-Befehl ausführen
cursor = con.cursor()
SQLBefehl = 'SELECT Name, Einwohner FROM kontinent'
cursor.execute(SQLBefehl)


# Durchlaufen der Ergebnisse
row=cursor.fetchone()
while (row!=None):
  print(row[0], row[1])
  row = cursor.fetchone()
```

fetchone holt einen (passenden) Datensatz, der dann im Folgenden ausgewertet werden kann
beim nächsten Aufruf von fetchone ist dann der nächste (passende) Datensatz gemeint
fetchone lässt sich also gut in Schleifen-Strukturen nutzen
Der Datensatz (hier als Zeile (row) interpretiert) wird als Liste aufgefasst und kann über Listen-Befehle oder Index-Zugriffe manipuliert werden

oder mit Feld-Namen:
und Daten in ein Directory

```
cursor = con.cursor(dictionary=True)
SQLBefehl = "SELECT Name, Einwohner FROM kontinent"
cursor.execute(SQLBefehl)

row=cursor.fetchone()
while (row!=None):
  print(row['Name'], row['Einwohner'])
  row = cursor.fetchone()
```

6.3.2.2. Veränderung von Tabellen

hier kommen die SQL-Befehle UPDATE und INSERT zum Tragen
s.a. → 5. SQL – die Datenbank-Sprache

Struktur-Veränderungen von Tabellen sind immer sehr aufwändig und Risiko-behaftet (s.a. → 6.3.2.2.3. Verändern der Tabellen-Struktur). Besser ist natürlich immer eine vorausschauende Planung einer Datenbank (→ 2.1. Datenbank-Modellierung).

6.3.2.2.1. Erstellen einer neuen Tabelle

```
[...]
SQLBefehle = """
CREATE TABLE IF NOT EXISTS schueler (
  SNR     integer NOT NULL,
  Vorname varchar(30),
  Name    varchar(50) NOT NULL,
  PRIMARY KEY (SNR)
);
CREATE TABLE IF NOT EXISTS fahrstunde (
  SNR    integer,
  Datum  date,
  Preis  integer,
  Zeit   time,
  PRIMARY KEY (SNR, Datum),
  FOREIGN KEY (SNR)
        REFERENCES Schueler(SNR)
);
"""
cursor = con.cursor()

try:
  for einzelCursor in cursor.execute(SQLBefehle, multi=True):
    print(einzelCursor._executed.decode('UTF-8'))


except mysql.connector.Error as err:
  print("Fehler bei der SQL-Ausführung: %s" % (err))
[...]
```

Anweisungs-Struktur	Erklärung	Beispiel / Hinweise
Attributname **varchar(***Länge***)**	definiert ein Attribut mit dem angegebenen Attributnamen -> Datentyp ist Zeichenkette mit einer angegebenen *Länge*	
Attributname **integer**	… -> Datentyp ist Ganzzahl	
Attributname **real**	… -> Datentyp ist Fließkommazahl	
Attributname **date**	… -> Datentyp ist Datum	
Attributname **time**	… -> Datentyp ist Zeit	
Attributname **blob(***Größe***)**	… -> Datentyp ist unbestimmt; es wird Speicherplatz für maximal *Größe* Byte reserviert	
*Schlüsse*l **integer NOT NULL**	NOT NULL bestimmt, dass dieses Attribut nicht leer sein darf → sonst Fehlermeldung	
PRIMARY KEY (*Schlüssel***)**	legt Attribut Schlüssel als Primär-Schlüssel fest	
FOREIGN KEY (*Schlüssel***) REFERENCES** *Tabellenname(Schlüssel_dort)*	legt Attribut *Schlüssel* als Sekundär-Schlüssel fest (dieser befindet sich in der Tabelle mit dem angegebenen Tabellennamen und heißt in dieser *Schlüssel_dort*)	

```
print("Neuanlage eines Fahrschülers")
print("============================")

neuName    =     input("Name   : ")
neuVorname =     input("Vorname: ")

# höchste Nummer finden
cursor = con.cursor()
SQLBefehl = 'SELECT MAX(SNR) FROM schueler'
cursor.execute(SQLBefehl)
row=cursor.fetchone()

neuSNR = int(row[0])+1


SQLBefehl = "INSERT INTO schueler (SNR, Vorname, Name) VALUES
(%i,'%s','%s')" \
          % (neuSNR, neuName,neuVorname)
cursor.execute(SQLBefehl)
print(cursor._executed.decode('UTF-8'))
cursor.close()
```

bei MySQL ist spezielle Anweisung für die Erzeugung eines Schlüssel verfügbar, der die Schlüssel-Nummerierung automatisiert (AUTO_INCREMENT)

```
CREATE TABLE IF NOT EXISTS schueler (
  SNR          integer NOT NULL AUTO_INCREMENT,
  Vorname      varchar(30),
  Name         varchar(50),
  BevorzugtFNR integer,
  PRIMARY KEY (SNR)
)
```

```
print("Neuanlage eines Fahrschülers")
print("============================")

neuSNR     = int(input("SNR    : "))
neuName    =     input("Name   : ")
neuVorname =     input("Vorname: ")


SQLBefehl = "INSERT INTO schueler (SNR, Vorname, Name) VALUES
(%i,'%s','%s')" \
          % (neuSNR,neuName,neuVorname)

cursor = con.cursor()
cursor.execute(SQLBefehl)
# print(cursor._executed.decode('UTF-8'))
cursor.close()

con.commit()
```

```
; Ändert den Namen des Schülers mit der Nummer 11.
UPDATE schueler
   SET name="Konda"
 WHERE SNR = 11

; Löscht alle schueler, deren SNR kleiner als 20 ist.
DELETE FROM schueler
 WHERE SNR < 20
```

Aufgaben:

1. ***Erstellen Sie ein Python-Programm, dass die Datensätze einer vorgegebenen Tabelle aus einer Datenbank anzeigt!***
2. ***Erweitern Sie das Fahrschul-Programm um die Möglichkeit einzelne Fahrstunden einzugeben und in der Datenbank zu speichern!***
3. ***Konzipieren und realisieren Sie ein Programm, dass einen einfachen Menü-basierten Umgang mit einer Datenbank ermöglicht! Einigen Sie sich im Kurs auf einen minimalen Funktions-Umfang! (Die Kursteilnehmer mit dem erhöhten Anspruch realisieren zusätzliche Funktionen!)***
4.

6.3.2.2.2. Löschen einer Tabelle

6.3.2.2.3. Verändern der Tabellen-Struktur

```
ALTER TABLE schueler
  ADD COLUMN GebDat date DEFAULT NULL
```

6.3.2.3. Abarbeitung großer SQL-Skripte

```
  for einzelCursor in cursor.execute(SQLBefehle, multi=True):
    print(einzelCursor._executed.decode('UTF-8'))
```

SQLBefehle ist im obigen Programm eine vorher zu definierende und zu belegende Liste mit SQL-Statements

6.3.3. SQL-Fehler abfangen

dazu benutzt man die Exception-Strukturen von Python

```
cursor = con.cursor()
SQLBefehl = 'SELECT Name, Einohner FROM kontinent'
try:
  cursor.execute(SQLBefehl)

  # Abrufen der Ergebnisse
  [...]

except mysql.connector.Error as err:
  print("Fehler bei der SQL-Ausführung: %s" % (err))
```

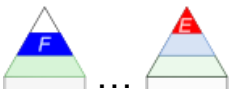
Für einfache Versuche mit Datenbanken sind solche Strukturen oft zu aufwändig. In echten "Arbeits"-Anwendungen sind sie aber elementar wichtig. Ohne Absicherungen kann es zum Absturz der Applikation und damit auch zum Datenverlust kommen.

6.3.4. Nachbereitung

Die Verbindung zu einer Datenbank (→) muss am Ende unbedingt ordnungsgemäß geschlossen werden. Ansonsten wäre eine externe Manipulation unter dem angemeldeten Account möglich.

```
# Ende der Verarbeitung
cursor.close()
# Abmelden
con.disconnect()
```

6.3.5. Gefahren beim programmtechnischen Umgang mit SQL



SQL-Injection

ein Benutzer versucht durch Manipulation Sicherheitslücke oder Anwendungs-Fehler für eigene Zwecke zu benutzen

z.B. wird eine Eingabe (z.B. ein Text durch SQL-Sequenzen ergänzt / ersetzt

Geben Sie einen Namen ein: Meier**' OR '1'='1**

diese werden in die benutzte SQL-Anweisung mit eingebaut und dann ausgeführt
z.B. DELETE "Meier" **OR 1 = 1**

Da "1 = 1" immer zutrifft, wird der SQL-Befehl ausgeführt. In diesem Fall das Löschen für alle Datensätze. Solche Unzulänglichkeiten in Programmen (! und nicht in den Datenbanken!) lassen schnell einen Alptraum wahrwerden. Ganze Datenbanken lassen sich u.U. so löschen.
Die nächste Situation zeigt dieses Horror-Szenario mit einem einfachen SQL-Statement in einer Eingabe.

Geben Sie einen Namen ein: M%**'; DROP TABLE** *Tabellenname1***; DROP TABLE** *Tabellenname2***;**

Maßnahmen zur Reduzierung / Verhinderung von Datenbank-Angriffen

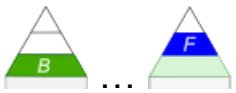
- Beschränkung der Rechte des Anwender (der Applikation) bzw. der Applikation bezüglich / innerhalb der Datenbank
- Kontrolle von Nutzereingaben (soll z.B. nur ein Datensatz mit DELETE gelöscht werden, dann kann durch Mitzählen (cursor.rowcount) erfasst werden, dass mehrere Aktionen ausgeführt worden → dann kann man mit con.rollback() statt con.commit() die gemachten Änderungen sofort wieder rückgängig machen)
- Vermeiden vom Multi-Cursor's

6.3.6. Projekt: Darstellung von Positionen in einer geographischen Karte

```
# Konstanten: Grenzen der Karte in Grad
cNord = 55.1
cSued = 47.2
cWest = 5.5
cOst  = 15.5

def InPixelNS(bild, grad):
  """ Umrechnung Geokoordinate in Bildkoordinate. Übergabe: Bild (für des-
sen Höhe) und Breitengrad """
  erg = bild.height()-round((grad-cSued)*bild.height()/(cNord-cSued))
  return erg

def InPixelWO(bild, grad):
  """ Umrechnung Geokoordinate in Bildkoordinate. Übergabe: Bild (für des-
sen Breite) und Längengrad """
  erg = round((grad-cWest)* bild.width()/(cOst-cWest))
  return erg
```

Q: https://www.inf-schule.de/information/datenbanksysteme/zugriff/pythonzugriff/projekt_karte

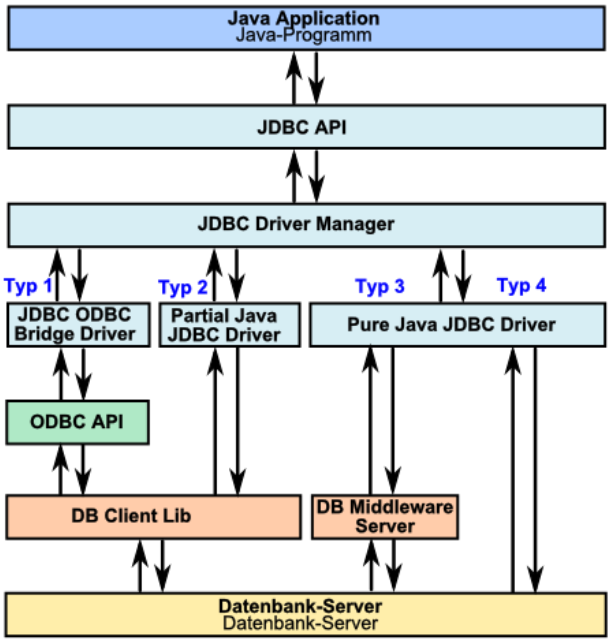
6.4. Datenbank-Zugriff mit JAVA



Bei JAVA steht von Anfang an der Objekt-orientierte Zugang im Vordergrund.

von Sun entwickelt Java Database Connectivity (JDBC)
entspricht im Prinzip der ODBC-Schnittstelle
ist universell verwendbar, aber schon stark auf die Nutzung in JAVA orientiert

Prinzip der Teilung in Verbindungs-Aufbau und SQL-Daten-Verarbeitung



Treiber-Typen bei JDBC

- **Typ-1** kommuniziert über eine sogenannte JDBC-ODBC-Brigde mit der Datenbank, d.h. für die Datenbank muss ein ODBC-Treiber vorhanden und installiert sein
  sollte nur verwendet werden, wenn für die Datenbank eine ODBC-, aber keine JDBC-Schnittstelle gibt

- **Typ-2** hier läuft die Kommunikation mit der Datenbank über eine Plattform-abhängige Programm-Bibliothek auf dem Client

- **Typ-3** setzt die JDBC-Schnittstellen-Befehle (API-Befehle) auf dem Client in generische (originale) Befehle des genutzten DBMS um, diese werden dann über das Netzwerk an einen speziellen Zwischen-Server (Middleware) übertragen, der dann die eigentliche Datenbank-Nutzung realisiert
  eventuelle Ergebnisse einer Datenbank-Abfrage werden, ausgehend vom Middleware-Treiber, über das Netzwerk an den Client übertragen

- **Typ-4** kommuniziert direkt über die JDBC-API-Funktionen mit dem DBMS, der Typ-4-Treiber kommuniziert direkt über das Netzwerk mit dem Datenbank-Server

Anbindung per ODBC (z.B. für MS-Access)

1. Java-Anwendungen sollten auf SQL-Datenbanken nicht per ODBC, sondern möglichst nur per direktem JDBC-Type-4-Treiber zugreifen. Ein Zugriff über die ODBC-JDBC-Bridge ist wesentlich aufwändiger und langsamer.
2. Zur Einrichtung von ODBC und DSN siehe SQL-ODBC.
3. Connection (siehe unten 'Programmierbeispiele'):

```
Class.forName( "sun.jdbc.odbc.JdbcOdbcDriver" );
Connection cn = DriverManager.getConnection( "jdbc:odbc:" + sDsn,
dbUsr, dbPwd );
```

Anbindung an MySQL

1. Zur Installation von MySQL siehe: mysql.htm#InstallationUnterWindows.
2. MySQL-JDBC-Type-4-Treiber (z.B. '`mysql-connector-java-5.1.16-bin.jar`' aus '`mysql-connector-java-5.1.16.zip`') downloaden von: http://www.mysql.com.
3. CLASSPATH muss JDBC-Treiber beinhalten.
4. Connection (siehe unten 'Programmierbeispiele'):

```
Class.forName( "com.mysql.jdbc.Driver" );
cn = DriverManager.getConnection(
"jdbc:mysql://MyDbComputerNameOrIP:3306/myDatabaseName", dbUsr, dbPwd
);
```

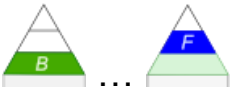
Anleitung: Einfaches Programmierbeispiel zum Auslesen einer Tabelle aus einer SQL-Datenbank

Link: www.tortsen-horn.de/techdocs/*

Links:

http://www.lehrer.uni-karlsruhe.de/~za220/htm/kurse/java/Client-Server/Datenbanken.htm
http://www.highscore.de/java/aufbau/datenbankanbindung.html
https://www.java-forum.org/thema/ms-sql-datenbankzugriff-per-jdbc.26835/

6.5. Datenbank-Programmierung mit Snap!

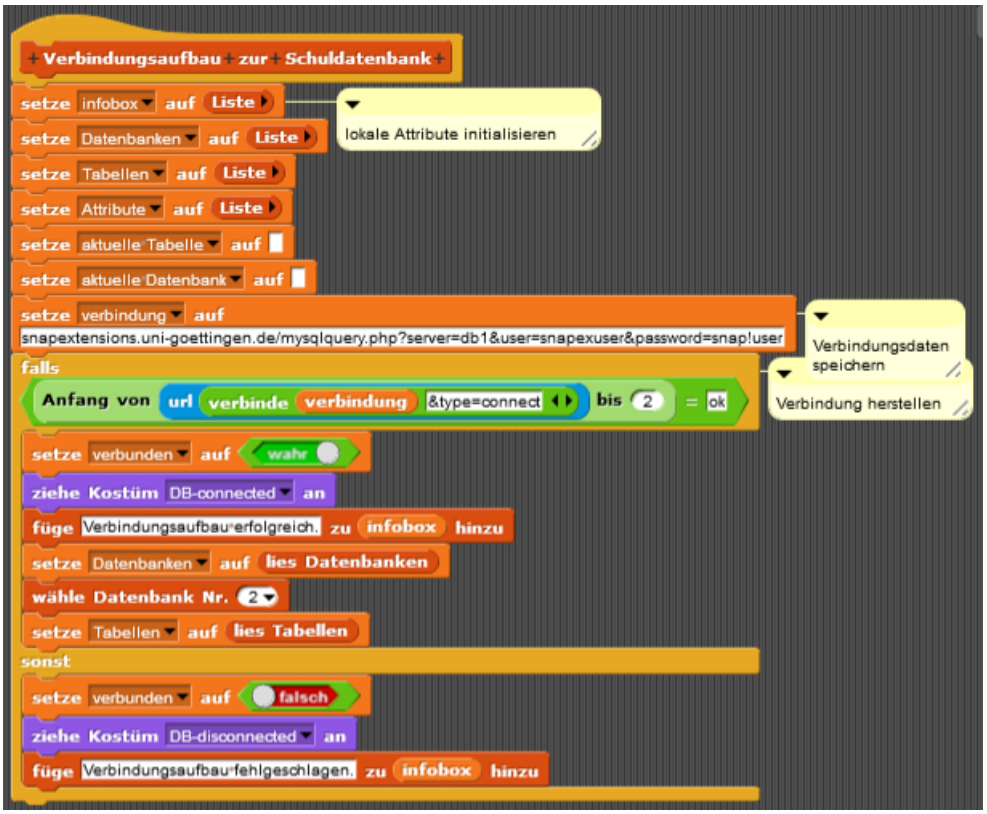


Auch bei Snap! wird die Programmierung von Datenbank-Zugriffen ganz klassisch realisiert. Die Kommunikation erfolgt über SQL-Statements. Diese werden im Programm zusmmengestellt und dann an den SQL-Server abgeschickt. Dieser antwortet auf das Statement. Die Antworten reichen von einfachen Bestätigungen und enden bei der Rücklieferung großer Datenmengen. Diese können dann wieder vom eigenen Programm in gewünschter Form aufbereitet werden.

notwendig sind zusätliche Blöcke
Importieren der SQL-Blöcke

zuerst Verbindungs-Aufbau zu einer Datenbank
Schon dieser – einfach scheinende – Block hat es in sich.

Genauer aufgelöst sieht er so aus.
Es handelt sich um einen speziellen Verbindungs-Aufbau zu einer online-Datenbank, die von MODROW für Schulungs-Zwecke und Projekte ins Internet gestellt wurde.
Für erste Geh-Versuche reicht uns das auch erst einmal.



Unter dem angegebenen Server (mySQL) sind mehrere Datenbanken verfügbar.
Die Auswahl erfolgt mit dem "wähle Datebbank Nr. ()".
Datenbank 2 ist eine kleine "Schuldatenbank"
Die Datenbank mit der Nummer 5 enthält Verkehrszeichen.

Leider ist der genutzte Verbindungs-Aufbau nicht ganz typisch.

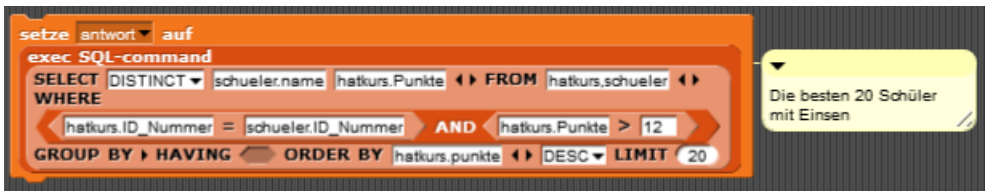
Anzeige der vorhandenen Tabellen (in der gerade verbundenen Datenbank) dann Block "Lies Tabellen"

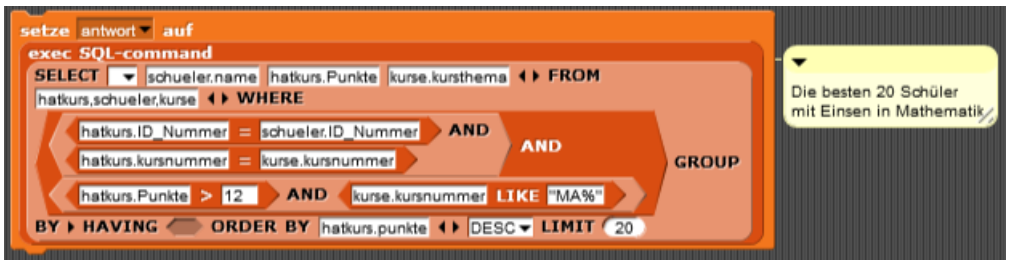
Anzeige von Attributen
Block "Lies Attribute von Tabelle …"

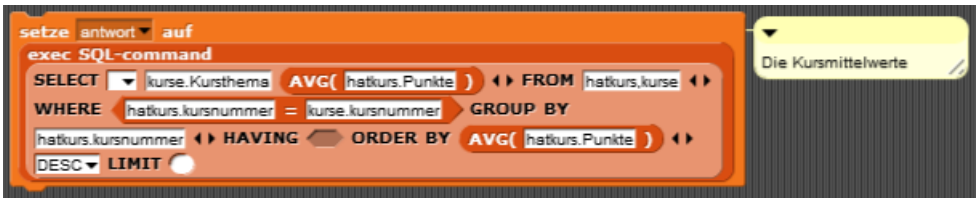
SELECT-Anweisung (einfache Variante) nur mit FROM und WHERE
muss einem SQL-Ausführ-Block übergeben werden.
Ergebnisse von SQL-Anfragen können auf eine Variable gelegt werden

komplexer SELECT-Block

ausgewählte Beispiele (aus open.SAP-Kurs "Einführung in die Informatik" mit Prof. MODROW)







Die Daten aus der Variable lassen sich mit **split** dann in verarbeitbare Datensätze (Tupel) zerlegen. Zugriff ist z.B. über Index möglich. Elemente (der Tupel) sind Komma-getrennt.

hier stößt man dann auch schnell an die Grenzen graphischer / Block-orientierter Programmierung

6.x. Anwendungs-Projekt "Super-Markt 'Kauf-ein' "

Flaschen-Kasse
scant die Flaschen oder Kästen
ermittelt Pfand-Betrag und entscheidet über weiteren Weg der Flaschen (Recycling (Einweg) oder Mehrweg)

Kassen
mit Bar-Code-Lesern zum Erfasen des Artikels und dann Berechnung des zu bezahlenden Betrages
KartenZahlung mit Kredit-Karte (Prüfung der Karten-Nummer)
Einziehen des Betrages von einem (Kunden-)Konto

Obst- und Gemüse-Abteilung
intelligente Waage (erkennt mit Kamera die Obst und Gemüse-Sorten)
zählt sie oder wiegt sie, je nach Preis-Auszeichnung

Lager
Warenwirtschaftssystem mit Produkten, Lieferanten und Kunden
Lager-Positionen für bestimmte Artikel
Nachbestellung, wenn Lager-Bestand unter das Limit gefallen ist

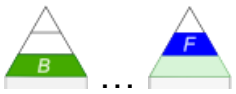
Werbe-Abteilung
die aus den Rennern und Pennern eine Mischung für die Werbe-Anzeige macht

Parkhaus
Kamera's erfassen Autonummer und errechnen aus Ankunft und Abfahrt die Park-Gebühren

Aufgaben:

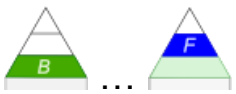
1.
2.
3.

7. Web-Datenbanken



7.0. theoretische Vorbetrachtungen

7.1. Arbeiten mit XAMPP



erste Erläuterungen und Anleitungen im Script: → 🕮 **dynamische Webseiten und Web-Datenbanken**

7.1.0. Vorbereitungen

Download von xampp.org
für einfache Projekte am Besten geeignet ist die portable ZIP-Version (unter weitere Versionen zu finden!)
kann auf einem USB-Stick etc. entpackt werden (also transportabel und ohne Admin-Rechte nutzbar)
aktualisieren und e.v. schnell mal neu installieren geht damit auch am Einfachsten
entpacken in das Wurzel-Verzeichnis eines Laufwerkes (z.B. USB-Stick)

hier auch gut in Kombination mit dem "portable Apps"-System zu nutzen
enthält praktische alle wichtigen Anwendungen, alle frei nutzbar, werden ständig gepflegt, lassen sich bei jedem Start aktualisieren (wenn notwendig und gewollt), auf jedem WINDOWS-Rechner benutzbar (außer bei gesperrtem USB)
Download über portableApps.com
zuerst das Menü-System und dann später die gewünschten Applikationen

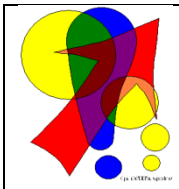
für echte Anwendungen (z.B. Intranet) sollte ev. "normale" Version genutzt werden

7.1.1. Einrichtung und Konfiguration

7.1.x.

gute Quelle; sehr viele Info's und Detail's → www.inf-schule.de

8. Datenbanken aus der Sicht von Wirtschaft und Wissenschaft



Bemerkungen zum folgenden Abschnitt:
Die Inhalte sind hier noch sehr locker und teilweise recht wirr zusammengetragen. Da dieser Bereich nicht direkt ein Kern-Thema eines Informatik-Kurses in der Sek.II betrifft, wird dieser Abschnitt auch als Anhängsel betrachtet.
Interessant könnten aber diverse Detail für Schüler sein, die sowohl Informatik als auch Wirtschaft belegen. Auch für praktische Programmier-Aufgaben gibt es viele Ansatzpunkte.
Nach und nach werden die Inhalte hier weiter ergänzt und schrittweise geordnet und betextet.

viele Informationen, Ideen und Strukturen basieren auf einem online-Kurs "Big Data Analytics" von Prof. Dr. E. MÜLLER bei openHPI (→ open.hpi.de) sowie dem OpenHPI-online-Kurs "Data Science und Data Engineering - Klarheit in den Schlagwort-Dschungel" mit Prof. F. NAUMANN (Jan./Feb. 2020)
diese Kurse können zum Selbststudium genutzt werden und sind sicher sauberer konzeptioniert als die nachfolgenden Seiten
eine echte Kurs-Teilnahme ist nicht mehr möglich, vielleicht wird der Kurs wieder aktiviert, das muss tagaktuell geprüft werden

Beispiel: Data-Warehouse

Schicht / Ebene		
externe Ebene	hohe (Zeit-, Rechen- u. Daten-aufwändige) Anforderungen durch die Nutzer	
konzeptionelle Ebene	redundanzfreie Basis-Tabellen als Dimensions-, Fakten- u. / od. Lookup-Tabellen	
interne Ebene	Rohdaten, häufig in denormalisierter (stark redundanter Form)	

in der Nacht werden die riesigen Aggregations-Tabellen / -Strukturen der externen Ebene erzeugt, diese sind dann hoch verfügbar und können extrem schnell benutzt werden
Aggregationen erzeugen Datenbestände, die bis zu sechsmal so groß sind, wie die Basis- bzw. Roh-Daten, Aggregations-Tabellen befinden sich als gespeicherte Daten (meist Tabellen) dann auch auf der internen Ebene

Definition(en): Data Warehouse
Unter Data Warehouse versteht man ein Informations-System zur Analyse von Unternehmen- und Institutions-Daten.

Data Warehouse's sind aus der immer größeren Datenmange geboren, die von Datenbanken und im täglichen Arbeiten von Informations-Systemen erzeugt und bereitgestellt werden.

bei der Nutzung (z.B. Abfrage einer Tabelle) kann es ein Problem mit der ständigen Aktualisierung von Daten-Bestände geben
Trennung der transaktionalen Datenbank (mit dem online Datenstrom) von der Auswertungs-Datenbank – eben dem Data Warehouse
i.A. ist das Data Wharehouse auch wieder eine – z.B. relationale – Datenbank
auf Daten-Analyse und –Darstellung spezialisiert (aktuelle Verkaufs-Trends, Tages-Berichte, …)
Übertragung der Daten aus der transaktionalen Datenbank in das Data Warehouse z.: Nachts, in einer Geschäfts-schwachen Zeit, zum Feierabend, …
Prozess der Übertragung nennt sich ETL (Extract Transform Load) beinhaltet eben die Vorgänge:

- Extraktion der Daten aus der transaktionalen Datenbank
- Transformation der extrahierten Daten in die Data Warehouse-Struktur
- Laden / Speichern der transformierten Daten im Data Warehouse (ev. mit Filterung und Normalisierung)

vor allem die ersten beiden Vorgänge sind sehr aufwändig
zusätzlich kann es auch noch eine weitere Extraktion der Daten für einen Daten-Markt (Data Market) geben, in dem nur die wichtigsten Daten-(Teil-)Mengen verfügbar sind, hier dann aber effektivere Zugriffe möglich sind

Vorteile des Data Warehouse-Konzepts

- keine Beeinträchtigung des laufenden Betriebes durch Daten-Analyse
- starke Spezialisierung auf Analyse und Präsentation
- es sind Daten-Analysen zu Zusammenhängen möglich, die man sich tagaktuell ausdenken kann, und die nicht von der eigentlichen Struktur der Daten in der transaktionalen Datenbank angelegt waren
- Verknüpfung mit anderen Datenbanken (z.B. Geodatenbank)
-

Nachteile des Data Warehouse-Konzepts

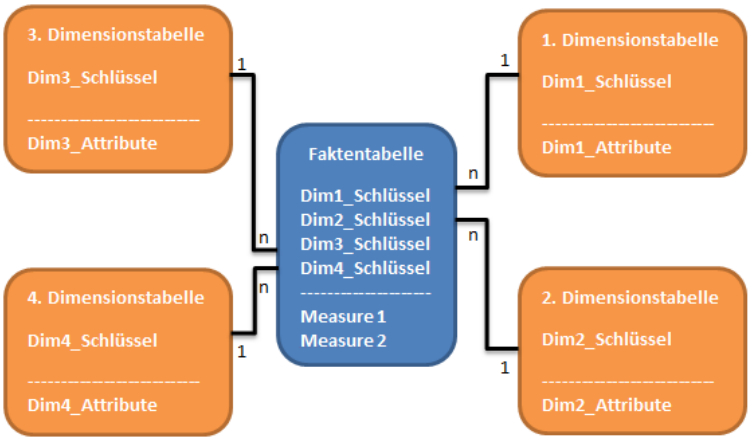
- Daten im Data Warehouse sind immer (etwas) veraltet
- Daten haben im Data Warehouse meist andere – oft eine einfachere – Struktur
- meist noch mal Vergrößerung der Datenmenge durch Umstrukturierung
-

Data Warehouse umfasst Komponenten zur:

- Aufbereitung von Daten
- Daten-Haltung
- Informations-Analyse

sowie den

- Data Warehouse- und Metadaten-Manager



Star- od. Schneeflocken-Schema
Q: www.datenbanken-verstehen.de

Definition(en): Business Intelligence
Unter Bussiness Intelligence werden Methoden und Prozesse zur Erhebung, Speicherung, Verarbeitung und systematischer Auswertung elektronischer Daten zusammengefasst.

Bereiche:
Daten-Analyse (Erfassung und Auswertung von Unternehmens- und Markt-Daten)
Advanced Analytics (frühzeitiges Erkennen von Markt-Entwicklungen, Prognosen)

Technologien der Bussiness Intelligence:

- **Data Warehouse**
- **Online Analytical Processing (OLAP)**
- **Data Mining**

Definition(en): Queries
Queries (sprich:) sind Abfrage-Mechanismen in Datenbanken.
Queries sind Anfragen an Datenbanken.

Definition(en): Schema
Ein Schema (i.S. von Datenbanken) ist eine Sammlung von Metadaten, die die Struktur von Relationen (Tabellen) und ihre Verbindungen (Verknüpfungen, Kardinalitäten) beinhalten.

Definition(en): Skalierbarkeit
Die Skalierbarkeit ist die Fähigkeit Leistungen / Funktions-Umfänge / Algorithmen / … auf kleinere oder größere bzw. größer werdende Datenbestände anzuwenden.

durch horizontale Skalierbarkeit wird erreicht, dass in Datenbanken auf JOINs (→ 4.0. Relations-Algebra / Relations-Kalkül) und Transaktionen verzichtet werden
besonders bei NoSQL-Datenbanksystem bringt das Performance-Vorteile und der organisatorische / administrative Aufwand reduziert sich

aus System- bzw. Betriebs-ökonomischer Sicht werden auch noch die folgenden Faktoren betrachtet

Definition(en): Gesamtbetriebskosten
Sind die absoluten oder relativen Aufwendungen für den Betrieb einer Datenbank in einem bestimmten Zeitraum. Als Kostenfaktoren kommen in Frage: Adminstration, Datensicherungen, Handware, Software, Updates, Energie, …

Definition(en): Leistung
Die Leistung einer Datenbank ist ein Maß für die Menge manipulierter Daten, die produzierten Ergebnisse, Arbeitsdurchläufe, … pro Zeit-Einheit.
Als Leistungs-Ausdruck kann auch der veränderte Aufwand bei gleichbleibenden Resultaten verwendet werden.

je geringer die (Betriebs-)Kosten bei gleichbleibenden Ergebnissen, umso besser / höher ist die Leistung der Datenbank

8.x.y. BigData, Data Science und Data Engineering

Q: basierend auf den Open-HPI-Kursen:
"Big Data Analytics" von / mit Prof. MÜLLER ()
"Data Engineering und Data Science – Klarheit in den Schlagwort-Dschungel" von / mit Prof. NAUMANN (Jan/Feb 2020)
→ Kurse stehen i.A. zum Selbststudium auf open.hpi.de zur Verfügung!

8.x.y.0. Historisches

Daten wurden schon immer gesammelt und in irgendeiner Form dargestellt
Geschichten von Jagden, Bau-Aufwendungen bei den Pyramiden, Rezepte für Heilmittel auf Papyrus, Arbeits-Anweisungen für die Mumifizierung, …

aus der Daten-Wissenschaft erwachsen

185? Analyse des Cholera-Ausbruches in London vom John SNOW
Technik der Aggregation (Aufsummierung und graphische Darstellung in einer Karte)
einfache Daten analysieren
eine Hypothesen gebildet

derzeit Flut an Daten; Menge so groß dass wir als Menschen sie gar nicht mehr und selbst Computer kaum alle Informationen aus ihnen herausholen können
heute sind Daten auch besonders vielgestaltig

US Library of Congress: 20 TB Text
Amazon: 42 TB
YouTube: 45 TB
National Energy Research Scientific Computing Center (NSERC): 2,8 PB
World Data Centre for Climate (WDCC): 330 TB + 6 PB auf Magnetbändern
LexisNexus / ChoicePoint: 250 TB (Personen-Daten!)

Data mining
Finden von Regeln (besonders WENN … DANN … Regeln), Zusammenhängen, Klassen (in Klassen einsortieren), Clustern (Gruppieren, Gruppen bilden), Mustern usw.

Klassiker der Regel-Zusammenhänge: die Windel-Bier-Regel
besonder zum Wochenende hin und kurz vor Geschäftsschluß werden Windeln und Bier zusammen gekauft → Produkt-Anordung im Supermarkt darauf hingehend angepasst

Daten-getriebene Anwendungen
(meist recht einfache) Anwendungen zur Analyse von (sehr) großen Daten-Mengen oder Daten-Strömen
typische Nutzungs-Bereiche derzeit:

- Produkt-Management (Begleitung von Produkten über ihres Lebenszeitraum → Erkennung von Problemen)
- Heim-Automatisierung (SmartHome → Lernen der Gewohnheiten)
- Gesundheitswesen (Epidemien, Krankheits-Verläufe, Medikamentationen, …)
- Wasser-Management ()
- Wissenschaft ()

- Markt-Forschung ()
- Informations-Marktplätze (Kauf und Verkauf von Daten)
- Verkehrs-Management (Straßen-Planung, Ampel-Schaltungen (→ grüne Welle), …)
- Energie-Management ()
- Bildungswesen ()
- Politik (→ Beeinflussung von Wahlen, Wähler-Mobilisierung, …)
- …

Maschinelles Lernen (Machine Learning)
Verallgemeinerung / Zusammenfassung / … der heute bekannte Techniken, um aus (vorhandenen) Daten zu lernen, und ev. zu neuen Daten Voraussagen zu machen
Bildung von Modellen

positive Beispiel-Nutzungen für Big Data:

- Vorhersage (Wetter, Natur-Katastrophen, Sensoren- / Maschinen-Ausfälle, Krankheiten / Epidemien, …)
- Optimierung (Verkehrs-Flüsse, Maschinen-Nutzung, Straßen-Verläufe, Positionen von Sendemasten, Logistik, …)
- Personalisierung (Medikamentation, Ausbildung, Produkt-Empfehlungen, …)
- Komfort (Heim-Automatisierung, Anmeldung am Smartphone, (teil-)autonomes Fahren, …)
- Intelligenz (automatische Übersetzung von Texten, Computer-Gegner in Spielen, Robotik, …)
-

(Big) Nudging

(z.T. unterbewußte) Beeinflussung / Anregung des Handelns von Menschen

z.B.:
noch 1 Artikel verfügbar
heute 50 % Rabatt, Mond-Preise
hohe Nachfrage
in der letzten Zeit haben 10 andere Personen auch in diesem Hotel gebucht
Personen haben auch diese Produkte gekauft
Bewertungs-Sterne, Label, …
Platzierung von Waren in Regalen
Platzierung von Essen in einer Mensa (z.B. Salate nach vorn)
Standard-Einstellungen von Programmen / Apps (Meldung des Nutzer-Verhaltens an die Hersteller)
Fake News, Nachrichten-Kanäle (eher solche, die an der eigenen Weltanschauung orientiert sind)
Werbung (ganz allgemein)

negative Beispiele für Big Data:

- Eindringen (in die Privatspäre → Auffenthaltsort über's Handy, Gesichter-Erkennung über die Video-Überwachung, Smart Home, …)

- Klassifizierung (Kaufverhalten, Social Scoring, Gewährung von Freigängen / Bewährungen im Justizwesen, …)
- Fehl-Informationen (Filter Bubble (man sieht nur eine Auswahl von Nachrichten; Fake News))
- Einschreiten (Zensur, Drohnen-Einsatz für die Tötung von Personen)
-

- Beeinflussung von Wahlen (z.B. https://www1.wdr.de/mediathek/av/video-der-fall-cambridge-analytica--102.html)
- Social Scoring (Kontrolle, Bewertung und Sanktionierung der Staatsbürger; z.B. China)
- Überwachung des Internet-Daten-Verkehrs / eMail-Verkehrs / Telefon-/Handy-Gespräche (→ https://de.wikipedia.org/wiki/PRISM ; https://www.sueddeutsche.de/digital/ueberwachung-am-de-cix-betreiber-des-frankfurter-internet-knoten-verliert-klage-gegen-den-bnd-1.3996859 ; https://www.zeit.de/digital/datenschutz/2015-09/gchq-karma-police-internet-ueberwachung ; ...)

8.x.y.0.1. Historie der Datenbanken

zuerst proprietäre Daten-Sammlungen
Information Management Systems (IMS) / Pointer-basierte Systeme
1960er Jahre
Zeiger-orientierter Zugriff auf Daten-Gruppen
sehr an den Betriebssystemen und vorhandenen Datei- und Speicher-System orientiert
neu verfügbare Hardware (Groß-Rechner od. Mainframe's) wurde für die Daten-Verwaltung genutzt
schwache Trennung von physischer Datenspeicherung und dem Daten-Modell
Programmierer haben die Daten so gespeichert, wie sie angefallen sind oder sie sie für besonders günstig erachtet haben
IMS von IBM wurde für die Teile-Verwaltung des Apollo-Programms entwickelt und wird heute noch (natürlich weiterentwickelt) von der NASA verwendet
kommt auch immer noch bei Banken und Versicherungen zur Anwendung

relationale Datenbanken
Speicherung in definierten und standardisierten Tabellen u.a. mit Fremdverweisen und standardisierten Datentypen
Trennung von Datenspeicherung und Daten-Modell; Nutzer hat keinen Zugriff auf die Daten-Speicherung mehr!

Edgar Frank "Ted" CODD (1923 - 2003):
prägt Begriff des Daten-Modell als mathematisch und systematisch fundiertes Prinzip der Daten-Speicherung
entwickelt genial einfaches Prinzip des relationalen Daten-Modell's

erstes relationales System war System A von IBM
mit SIQUEL/SQL als Anfragesprache
es folgte dann INGRESS von Michael STONEBREAKER (1970er)

heute XML- und JSON-Daten-Modelle (???)
immer mehr Datentypen, die verwaltet werden können

neben einer Tendenz zu immer größeren Datenbanken mit immer größeren zu verwaltenden Daten-Mengen, kommen nun auch immer kleinere Datenbanken dazu (quasi Mini-Datenbanken, Eingebettete Datenbanken) für kleine und kleinste Anwendungen, die aber trotzdem das Datenbank-Konzept als Grundlage nutzen

ab 2010 verteilte Datenbanken und Schlagwörtern wie Big Data, Big Scale, …

"in Memory"-Datenbanken
Speichern und verwalten von Daten nach dem Datenbank-Prinzip direkt im Speicher
z.B. auch zur online-Analyse von Datenströmen

8.x.y.1. Was sind den nun "Big Data"?

viele Daten oder große Daten-Mengen sind schon lange bekannt
z.B. Volkszählungen
astronomische Daten (Kalendarien, Mond-Phasen, Mond- und Sonnen-Finsternisse, …)

GARTNER-Report (2012) erste breit akzeptierte Definition für BigData

Charakteristika von Big Data (die großen 3 V's)

- **Volume**
  (Umfang / Menge / Daten-Volumen)
  - auch als 1. Dimension betrachtet

  hierauf bezieht sich ursprünglich das "big"
- **Velocity**
  (Geschwindigkeit)
  - auch als 2. Dimension betrachtet
- **Variety**
  (Vielfalt / Vielzahl / Verschiedenheit / Abwechslung / Unterschiedlichkeit)
  - auch als 3. Dimension betrachtet

die drei Dimensionen werden auch als 3-V-Modell betrachtet

begleitet von innovativer Informations-Verarbeitung

ermöglicht tiefere Einblicke in die Inhalte / Strukturen der Daten und verbessert die Möglichkeiten der Prozess-Automatisierung z.B. durch automatisierte Algorithmen

Big Data heute mehr aus dem Bezug auf innewohnende Eigenschaften der Datensammlung bezogen
interessante Eigenschaften, Zusammenhänge, Probleme, … in den Daten werden gesucht

Charakteristika von Big Data durch IBM (die großen 4 V's / the four V's)

- **Volume**
  (Umfang / Menge / Daten-Volumen)
- **Velocity**
  (Geschwindigkeit)
- **Variety**
  (Vielfalt / Vielzahl / Verschiedenheit / Abwechslung / Unterschiedlichkeit)
- **Veracity**
  (Wahrhaftigkeit / Vertraulichkeit / Sicherheit)

vielfach als Big Data werden solche Daten verstanden, die zu groß (zu viele) sind, zu schnell (verarbeitet) werden müssen und / oder zu kompliziert sind, um sie mit gängigen Verfahren zu verarbeiten
als gängige Verarbeitung werden hier die üblichen Datenbank-Systeme verstanden

sehr zukunfts-offene Begriffs-Bestimmung

moderne Definitionen sehen BigData breiter angelegt

<u>Charakteristika von Big Data (die großen 7 V's)</u>

- **Volume**
  (Umfang / Menge / Daten-Volumen)
  - auch als 1. Dimension betrachtet

  hierauf bezieht sich ursprünglich das "big"
- **Velocity**
  (Geschwindigkeit)
  - auch als 2. Dimension betrachtet
- **Variety**
  (Vielfalt / Vielzahl / Verschiedenheit / Abwechslung)
  - auch als 3. Dimension betrachtet
- **Value**
  (Wert)

  Was ist der Mehrwert der Daten?
- **Variability**
  (Variabilität)

  Wie veränderlich sind die Daten (z.B. weltweit)?
- **Veracity / Validity / Volability**
  (Zuverlässigkeit / Gültigkeit /)
- **Visualisation**
  (Visulisierung)
-

von einzelnen Autoren werden noch weitere V's angegeben:

- Viscosity (Viskosität (gemeint Leichtigkeit, mit der die Daten ins System gelangen))
- Venue (Wo sind die Daten?)
- Vocabularity (Vokabular (gemeint sind z.B. unterschiedliche Begrifflichkeiten))
- Virality (Viralität (gemeint ist hier die ungleichmäßig Ausbreitung unterschiedlicher Daten innerhalb des Datenbestandes / eines Netzes / …))
- vagueness (Unklarheit / Unbestimmtheit / Unschärfe / Verschwommenheit)
- ???

vielfach sind diese aber auch gut anderen / den großen 7 V's zuzuordnen
Frage der exakten Begrifflichkeiten

massives Sammeln und strukturiertes Auswerten der Daten mit hoher Geschwindigkeit

- Kauf-Vorschläge (online-Werbung, Bannerwerbung,)
- automatische Such-Begriffs-Ergänzung (Such-Vorschläge)
- Marktforschung
- Web-Statistiken / Tracking (Nutzer-Verfolgung)
- Risiko-Bewertung und Beitrags-Anpassung bei Versicherungen
- digital price discrimination (Angebote an einen potentiellen Kunden mit einem Preis, den dieser wahrscheinlich gerade noch bezahlen würde)
- Bonitäts-Prüfung (Big Data Scoring)
- Entdeckung von Unregelmäßigkeiten bei: (Fraud-Detection)
  - Finanz-Verkehr
  - Aktien-Handel
  - Daten-Verkehr

- Server-Anfragen / System-Angriffe
-
- Bezahl-Systeme für Telekommunikation
- Energie-Verbrauchs-Überwachung und –Steuerung (Smart Metering)
- Geheimdienste (Bewegungs- und Nutzungs-Profile; → gläserner Mensch)
- Personal-Beschaffung / -Einstellung
- Vorhersage von Epidemien
- Panik-Forschung
- Wetter-Vorhersage
- Erdbeben und Vulkanausbruch-Vorhersage

nach und nach treten immer mehr Probleme (für die klassische Datenverarbeitung) auf
z.B. großes **Volumen**:
wenn man einer großen Datei – z.B. mit sortierten Einträgen – irgendwo in der Mitte einen Eintrag einfügen möchte, dann müssen alle nachfolgenden Einträge auf dem Datenträger weiter nach hinten verschoben werden
beim Löschen eines Eintrags in der Mitte müssen alle nachfolgenden Einträge nach vorne verschoben werden
was bei kleinen Dateien praktisch nicht auffällt, wird bei großen Dateien zum Hemmnis
Beispiel: Walmart verarbeitet pro Stunde mehr als 1'000'000 Transaktionen

Daten passen u.U. irgendwann nicht mehr auf einen Datenträger, auf einen Rechner, auf einen Server-Verbund

klassisches Durchsuchen vom ersten bis zum passenden Datensatz dauert bei großen Datenbanken zu lange

Sortieren von großen Datenmengen wird zum fast unlösbaren Problem

z.B. zu schneller Daten-Input (**Daten-Eintritts-Geschindigkeit** / Velority)
klassisches Beispiel ist die Daten-Verarbeitung an der Börse oder in Kontrollzentren von Chemie-Anlagen, Verkehrs-Führung (z.B. Eisenbahn), Karten-Verkauf für beliebte Konzerte, Handels-Plattformen (z.B. am Black Friday)

autonomes Fahren

Bildung von Puffern / Warteschlange in den meisten Fällen nicht möglich
beim welt-umspannenden Handel gibt es praktisch keine Ruhezeiten mehr
als passender Vergleich könnte man die Unannehmlichkeiten nehmen, die z.B. beim Video-Streaming auftreten, wenn die Daten zu langsam

Netzwerk-Überwachung (z.B. zum Erkennen von Hack's) kann nicht mit Verzögerung passieren
Überwachung des Finanzwesens (Kreditkarten-Betrug, Erkennung gespeerter Karten, Geldwäsche, keine Transfers in Embargo-Staaten, …)

Video-Überwachung mit Gesichts-Erkennung nutzt nur dann etwas, wenn gesuchte Personen sofort und sicher erkannt werden

große Daten-Mengen beim Beobachten von Atombomben-Test's od. ä.
Versuche am CERN beim Zusammenschießen von Teilchen
Daten entstehen hier in extrem kurzen Zeiträumen und müssen zumindestens gespeichert oder ausgefiltert, ev. auch ausgewertet werden

einfließende Daten müssen praktisch immer sehr zeitnah verarbeitet werden

Aufgaben:

1. *Beobachten Sie mit einer geeigneten App (z.B.) die Sensor-Daten Ihres Smartphone's (z.B. Lautstärke, …)!*
2. *2.*

zu unterschiedliche Daten (**Heterogenität der Daten** / Variety)
auch (multi-)modale Daten genannt / gemeint
unterschiedliche Sprachen (z.B. engl. und die Muttersprache, Slang's)
unterschiedliche Formate für die Darstellung eines Tages-Datum's, nicht nur in verschiedenen Ländern, sondern auch in einem Land selbst

Aufgaben:

1. *Tragen Sie für sich selbst die verschieden Formate für da heutige Datum zusammen! Wer kennt die meisten?*
2. *2.*

zu ungenaue / unzuverlässige Daten (**Daten-Qualität** / Wahrhaftigkeit / Sicherheit / Fehlerhaftigkeit)

zu komplizierte Eingabe-Masken mit ähnlichen Inhalten / Eigenschaften
hängen nicht selten von Daten-Eingeber ab (Subjektivität → Augenfarbe / Haarfarbe / …)

Unvollständigkeit von Daten ganz allgemein

in unklaren Fällen werden irgendwelche "Ersatzdaten" eingetragen, die von vielen Systemen oder nach Datentransfers nicht mehr als Ersatz-Daten erkannt werden
(z.B. Postleitzahl: 99999 in Deutschland, …)
fehlende Informationen und bestimmende Eingabe-Felder (z.B. bei einer Datenbank das Feld "weiblich": Problem bei fehlerder Information → Ist derjenige nun "männlich" oder fehlt die Information nur?
Eingabe-Masken haben zu wenige Felder, es müssen aber noch andere Daten mit erfasst werden, Daten werden dann in anderen Feldern mit eingetragen, was die Daten in diesem Feld nicht mehr sicher analysierbar macht

auch die maschinelle Übertragung / Umwandlung von Daten ist fehler-anfällig (sehr häufig Programmierfehler beim Umgang mit den Grenzen)

fehlerhafte Sensoren (verschmutzte Oberflächen, abweichende Arbeits-Temperaturen, …)

Aktualität der Daten (wenn z.B. die aktuellen Daten bei einer Klima-Auswertung fehlen, dan kann ein falscher Trend oder eine falsche Voraussage für das Sommerwetter od. ä. abgeleitet werden

nicht zuletzt geschwärzte Texte aus Archiven (Text-Analyse nur begrenzt sinnvoll, weil man nicht weiss, was für Wörter oder die Art von Wörtern unkenntlich gemacht worden)

Herausforderungen

- **Verarbeitung vieler Datensätze**

- **Verknüpfung vieler Datenfelder**
- **schneller Import großer Datenmengen**
- **schnelle (Echzeit-)Verarbeitung der Daten**
- **kurze Latenz- / Antwort-Zeiten**
- **Skalierbarkeit von kleinen bis großen Mengen / Problemen**
- **parallele Abarbeitung vieler gleicher, ähnlicher oder andersartiger Abfragen / Anforderungen**
- **sehr variable / unterschiedliche Daten-Typen auswerten**

→ Herausforderungen führen zu NoSQL-Datenbanken, da SQL solche Anfordeungen nur mit sehr großem technischen Aufwand erfüllen könnte

- Apache Hadoop Framework
- MongoDB
- Aster Data
- Greenplum
-

arbeiten nach MapReduce-Ansatz

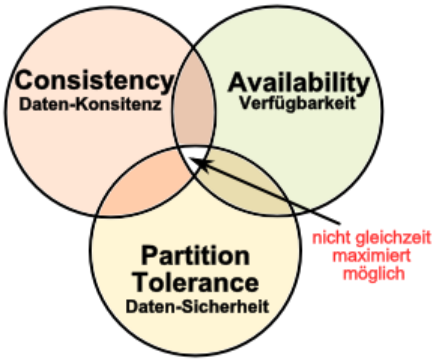
heute geschätzt, für die heute weltweit gesammelten Daten gilt

- 23 % der Daten sind wirklich nutzbar (97 % der daten aber nicht nützlich, da sie nicht indiziert sind)
- 0,5 % der Daten werden tatsächlich genutzt
- 35 % sind schützenswert
- 20 % werden wirklich geschützt (mit Datensicherungen und Datenschutz-Maßnahmen)

Die Zielrichtungen und das Selbstverständnis von Big Data und der Datenschutz sind potentielle und reale Widersprüche.

im Datenschutz gibt es kein "belangloses Datum"
an dem z.B. alte Daten gelöscht werden müssten oder vollkommen anonymisiert
selbst klassisch anonymisierte Daten lassen sich (mit Big Data-Methoden) ent-anonymisieren

CAP-Theorem
Leit- und Lehrsatz, der besagt, dass es nicht gleichzeitig möglich ist, alle drei Eigenschaften (Consistency (Daten-Konsistenz), Availability (Verfügbarkeit) und Partition Tolerance (Datensicherheit)) zu maximieren.



Daten-Quellen

mögliche Daten-Quellen

- **offene Daten**

  strukturierte Daten
  frei zugängliche Daten
  z.B. aus dem Internet
  - **Linked Open Data** (bewußt freigegebene Daten von Organisationen, Institutionen, Regierungen, …
    -
  - konkret: **Goverment Data**
    - statistische Daten der Länder
    - Protokolle von Regierungs-Sitzungen, …
    - geöffnete Archive
    - …
  - …

  z.B. Hidden Web
  - auf Webseiten abgelegte Daten-Bestände (z.B. als Dateien)
  -

  z.B. Wissenschaftsdaten
  - Klima- und Wetter-Daten
  - Sternen-Kartierung
  - Teilchen-Beschleuniger (CERN)
  - Protein-Strukturen
  - DNA-Sequenzen
  - Chemikalien-Informationen / Gefahrgut-Daten
  -

  unstrukturierte Daten
  z.B. Textdaten
  - Webseiten
  - veröffentlichte Dokumente (z.B. PDF-Dateien)
  - Zeitungs-Artikel
  - social media (tweet's, Like's, …)
  - Rechnungen
  - Zeitschriften
  - Patente
  - eMails
  - Produkt-, Hotel- usw. -Bewertungen

- **interne Daten**

  z.B. Geschäftsdaten
  - master data management
  - Bilanzen
  -

  z.B. Transaktions-Daten
  - Einkäufe in Webshop's
  - Überweisungen von Geldbeträgen
  - Währungs-Handel
  - Börsenhandel
  -

  z.B. Protokoll-Daten
  - Login's

- Verbindungs-Aufbau, -Dauer, -…
- Nutzer-Verhalten auf Webseiten
-

z.B. Sensor-Daten
- Strom-Verbrauch über moderne Stromzähler (Smarte Zähler)
- Smarthome
- autonomes Fahren
-

-

Anforderungen an Linked Open Data
- Datenbestand soll verlinkt sein (mit eindeuter ID → URI ID)
- URI ID möglichst im html-Format
- Hintergrund-Informationen
- möglichst weitere Links einarbeiten (→ Erzeugen eines semantischen Daten-Netzwerks)
- möglichst für Menschen lesbar, aber immer für Maschinen!

auch Tabellen auf Webseiten lassen sich selektieren (?erqualen / Quals)

Daten-Formate

- **XML**
- **JSON**
- **RDF**
- **SQL**
-

Beispiele für Daten-Quellen

allgemeine / diverse Daten

- **DBpedia** wikipedia-Daten
-
https://www.kaggle.com/datasets
-
https://opendatainception.io/
-
https://cloud.google.com/public-datasets/
-
https://github.com/awesomedata/awesome-public-datasets
-
https://registry.opendata.aws/
-
-

Wissenschaftsdaten / Science

Data

- Sloan Digital Sky Survey
  https://www.sdss.org/
- Large Synoptic Survey Telescope
  https://www.lsst.org/scientists/understanding-simulations-and-data
- CERN
  http://opendata.cern.ch/
- Menschliches Genom
  https://www.ncbi.nlm.nih.gov/genome/guide/human/
- Genome weiterer Organismen
  https://www.ncbi.nlm.nih.gov/guide/howto/dwn-genome/
- Protein-Strukturen
-
-

Regierungsdaten / Goverment Data

- **Europäische Union**
  https://data.europa.eu/euodp/en/data/
- **Weltbank**
  https://data.worldbank.org/
-
  https://www.who.int/gho/database/en/
- **(USA)**
  https://catalog.data.gov/dataset
- FBI-Welt-Fakten-Buch
- **Weltgesundheitsorganisation**
  https://www.who.int/gho/database/en/
- **NASA**
  https://data.giss.nasa.gov/
- **Statistisches Bundesamt (BRD)**
  https://www.destatis.de/DE/Service/Datenbanken/_inhalt.html
-
-

Definition(en): Big Data

Unter Big Data versteht man die Daten-Mengen / -Bestände (Massendaten), die durch auffallende Größe, hohe Komplezität, Schnellebigkeit und eine schwache Struktur gekennzeichnet sind.

interessante Links:
https://lod-cloud.net/ (Animation zur Entwicklung des "Linked Open Data"-Netzwerkes

8.x.y.2. Was genau ist "Data Engineering"?

Definition(en): Data Engineering

8.x.y.3. Was genau ist nun "Data Science"?

Verbindung von Domänen-Wissen (Wissen eines Anwendungs-Gebiet's), Daten-Analyse und Daten-Management

klassischer Wissenschafts-Ansatz "Voraussagen – Modellieren – Testen" (hypothesize, model, test) wird teilweise in Frage gestellt

nach gibt es ein viertes Paradigma der Wissenschaft
Paradigma = Denkweise
heute wird darunter in der Wissenschaft eine allgemein anerkannte, zusammengehörende aufeinander abgestimmte Sammlung von Regeln, Gesetzen, Methoden, … verstanden, die historisch einen längerfristigen Bestand hat
notwendige / überfällige Paradigmen-Wechsel befördern die weitere Entwicklung der Wissenschaft

anders kann man die Paradigmen einer Wissenschaft auch als ihre Entwicklungs-Phasen verstehen

Paradigmen / Entwicklungs-Phasen von Wissenschaften

- **empirisch experimentell** — emperical science
  Sammeln und Notieren / Beschreiben von Phänomenen
  Erfassen, Klassifizieren, Systematisieren von Daten
  Durchführen von (ungerichteten) Experimenten
- **theoretisch modellierend** — theoretical science
  Ableiten von Regeln und Gesetzen
  Bildung von Theorien, Modellen
  experimentelle Prüfung von Theorien / Voraussagen
- **berechnend simulierend** — computational science
  Erstellen und Nutzung von Simulationen
- **Daten-intensiv KI-basiert** — Nutzung von großen Mengen an (auch älteren) empirischen Daten, um neue und mehr Erkenntnisse aus ihnen abzuleiten
  (extrem bis sehr) komplexe Modelle der Realität (, die u.U. auch einzelne Fachbereiche / (Teil-)Wissenschaften überwuchern)
  quasi ein maschinelles Nachvollziehen der Wissenschafts-Entwicklung für ein Problem / eine Teilwissenschaft / …

mir erscheint das 4. Paradigma etwas stark Hype-geprägt bezüglich der derzeitigen Entwicklung von Bid Data usw. zu sein
wenn, dann befinden wir uns auch erst in der Anfangsphase, die auch ersteinmal objektiv reflektiert / begleitet werden muss
Guru's des vierten Paradigma's sehen schon das Ende der klassischen Wissenschaften
sie behaupten teilweise, dass man keine (anderen / klassischen) Modelle / Theorien mehr braucht, weil die Daten selbst die Welt / Realität / das Problem ausreichend beschreiben (Korrelation ist genug)
dabei wird aber vergessen oder unterschlagen, dass eine Daten-Analyse / Auswertung unbedingt Theorien / Modelle / … braucht, um den Daten einen Informations-Gehalt zu geben
Korrelation impliziert noch keine Kausalität

es besteht auch eine große Gefahr mit der Digitalisierung eine Wissenschaft nur aus der Sicht der digitalisierten bzw. digital erfassten Daten zu sehen
die traditionellen / klassischen / analogen Daten werden einfach ignoriert
einfach deshalb, weil sie für die digitalen Native's nicht leicht erfassbar ist
Gefahr des Ableiten falscher Schlüsse (da nur eine kleine / eingeschränkte Daten-Breite genutzt wird)

Beispiele für Fake-Korrelationen:

- Anzahl nicht-kommerzieller Raketen-Start's und der Anzahl von Promotionen in der Sozialogie
- Anzahl der Personen, die in Schwimmingpool's ertranken und der Anzahl von Filmen die Nicolas CAGE produziert hat

Definition(en): Data Science

Künstliche Intelligenz

AI … Artificial Intelligence

offiziell Teil der Robotik
mittlerweise getrieben durch die Problemkreise "maschinelles Lernen" (Machine Learning) und "Deep Learning" mehr losgelöst von der Verkopplung mit mechanischen Aktoren od.ä.
derzeit Sammel-Wissenschaft, die Erkenntnisse der Neurowissenschaften, Mathematik, Informatik (→ Theoretische Infromatik), Logik, Psychologie, Kommunikations-Wissenschaften, Philosophie, Linguistik, … nutzt und teilweise in neue (Wissenschafts-übergreifende) Zusammenhänge bringt
beschäftigt sich mit Wissens-basierten Systemen (Experten-Systeme), Muster-Analyse und –Erkennung, Muster-Vorhersage, Robotik, Künstliches Leben, Modellierung anhand künstlicher Entropie-Kraft, …

problematisch ist, dass es an einem übergreifenden Begriff oder einer Definition von Intelligenz mangelt
oft sehr frei interpretiert

starke KI sind Systeme, die auf dem gleichen Niveau, wie Menschen arbeiten
zur schwachen KI zählt man solche Anwendungen, die konkrete Einzel-Probleme lösen

typische Anwendungen:

- Suchmaschinen
- Gesichtserkennung
- Data-Mining
- Schrifterkennung
- Bilderkennung
- Texterkennung
- autonome Waffen
- Bots
- maschinelle Übersetzung
- humanoide Roboter
- Sprach-Assistenten
- Spracherkennung
- persönliche Assistenten
- autonome Fahrzeuge
- Computer-Vision-Systeme (Großraum-Video-Überwachung)
- Gruppen- und Verhaltenssimulationen
- Computer-Algebra-Systeme
- semantische Suchmaschinen
- Informationsrückgewinnung
- Biometrie
- wissensbasierte System
- Avatare / Gegenspieler

eigenständiges Lernen
Reagieren auf neue Situationen auf der Basis von bekannten Regeln

TURING-Test (1950)
ein Mensch (Probant / zutestendes System) kommuniziert mit einem anderen System (Maschine oder Mensch) und dieser/s kann nicht unterschieden mit welcher Art von Gegenüber er kommuniziert

wenn der TURING-Test bestanden wird, dann wird dem System eine (äquivalente) Intelligenz zugesprochen

Stufen der KI

- **schwache KI** — Lösung von konkreten Anwendungs-Problemen
  z.B.: Sprach-Assistenten (Siri, …), Bilderkennung
- **starke KI**
  allgemeine KI — Lösen mehrerer (komplexerer) Probleme auf Augenhöhe mit dem Menschen
  Anforderungen:
  - Ziehen logischer Schlüsse
  - Allgemeinwissen
  - Planungsfähigkeiten / Zielorientierung
  - Lernfähigkeit
  - Sprach-Verständnis
  - ähnliche Sensor-basierte Umwelterfassung
  - ähnliche Interaktion mit der Umwelt
  -
- **Super-KI** — KI ist intelligenter, als ein Mensch
  ev. selbstreproduzierend
  Problem der Singularität
  KI begreift ev. Mensch als Störfaktor, sind schneller im Fällen von Entscheidungen, nicht Emotions-getrieben

Singularität in der System-Theorie der Punkt, in dem eine kleine Veranderung / Ursache eine große Wirkung hat
hier gemeint der Punkt, an dem die / eine KI die Menschheit übertrumpft und damit die Zukunft der Menschheit danach unbestimmt / ungewiss ist
vielfach auch der Punkt verstanden, an dem eine KI so etwas wie ein Bewußtsein erlangt
(sie könnte aber auch clever genug sein, dieses Ereignis zu verstecken)
erste Konzepte der technologischen Singularität gehen auf Stanislaw ULAM (1965) zurück
nach einem bekannten Computer-Wissenschaftler und Zukunftsforscher Ray KURZWEIL ist mit der Singularität um 2045 zu rechnen

Daten-Kompetenz

Data Literacy

vielfach auch mit dem Codieren und Decodieren von Daten in Zusammenhang gebracht

das Codieren umfasst dabei das Sammeln, Aufbereiten und Analysieren der Daten
das Nutzen dieser Daten wird dem Decodieren zugeordnet

Teilziele:

- Daten-Kultur erreichen (in der Gesellschaft)
- Identifizieren von Daten-(getriebenen)Anwendungen
- Koordinieren von Daten-Anwendungen
- Ermöglöichen einer Daten-Bereitstellung / Daten-Modellierung
- Beachten des Datenschutz
- Reinigung der Daten
- Daten-Integration
- Klären Daten-Herkunft
- Analysieren von Daten
- Verbalisieren / Beschreiben von Daten
- Visualisieren / Präsentieren
- Interpretieren / Erklären von Daten
- Ableiten von Handlungen / Reaktionen
-

Definition(en): Data Literacy / Daten-Kompetenz
Daten-Kompetenz umfasst die Fähigkeiten Daten kritisch zu erfassen, zu verwalten (managen), zu bewerten und anzuwenden auf konkrete und / oder allgemeine Anwendungsfälle.
Unter der Daten-Kompeten werden solche Fertigkeiten und Fähigkeiten, Daten Sinn-entsprechend und Ziel-orientiert zu codieren und zu decodieren.

ethische Aspekte der Daten-Nutzung

in der Informatik sind die Daten sehr dicht an der Realität; weiter entfernt sind z.B. Hardware-Entwickler; erfordert erhöhte Sorgfalt

mit großer Macht-Fülle (Daten-Menge) ist auch immer eine große Verantwortung verbunden
Daten und Daten-Anwendungen sollen mehrfach genutzt werden (dual use)

vermeiden von fragwürdigem oder Sinn-entstellende / manipulierendem Umgang bzw. eine entsprechende Nutzung von Daten

Orientierung darauf, eine negative / fragwürdige Nutzung der Daten zu verhindern

eine gut entwickelte Daten-Kompetenz bewirkt eine Stärkung der positiven Nutzungen von Daten

Einhaltung des Datenschutzes (Gesetzes-konform, vorausschauend, verantwortungsbewußt)

Privatspäre

Wer hat u.U. Zugriff auf meine Daten?

- Sie selbst
- Familien-Mitglieder
- Ihre aktuellen und abgelegten Freunde (→ ev. jetzt Feinde)
- Ihr Internet-Service-Provider
- eigene oder fremde Regierungen / Geheimdienste
- ev. Jeder
    - z.B. bei schlechten Datenschutz-Einstellungen
    - Mitschnitt und Verkauf von Daten
    - durch Hacks und Datenlecks
- Archive speichern ev. über Jahre / Jahrzehnte hinweg
- …

Schutz-Möglichkeiten

- Daten(-Erhebung), … vermeiden
- strenge Gesetzgebung
- Daten-Kompetenz
- statistisches Grundverständnis
-

Themenfeld: Medizin-Daten

(erwartete) Vorteile durch Big Data

- **personalsierte Medizin**
- **Arzneimittel-Planung**
- **Risiko-Sensoren**
- **Forschung**
- **Genom-Analyse**
-

(erwartete) Nachteile durch Big Data

- **Anonymität**
- **"Big Brother"-Effekt**
  Überwachungs-Effekt
- **Korrelation statt / vor Kausalität**
- **Entscheidungs-Kompetenz bei Maschine**
- **elektronischer Code / Algorithmen statt humanistischer Werte**
-

"Nationale Kohorte"

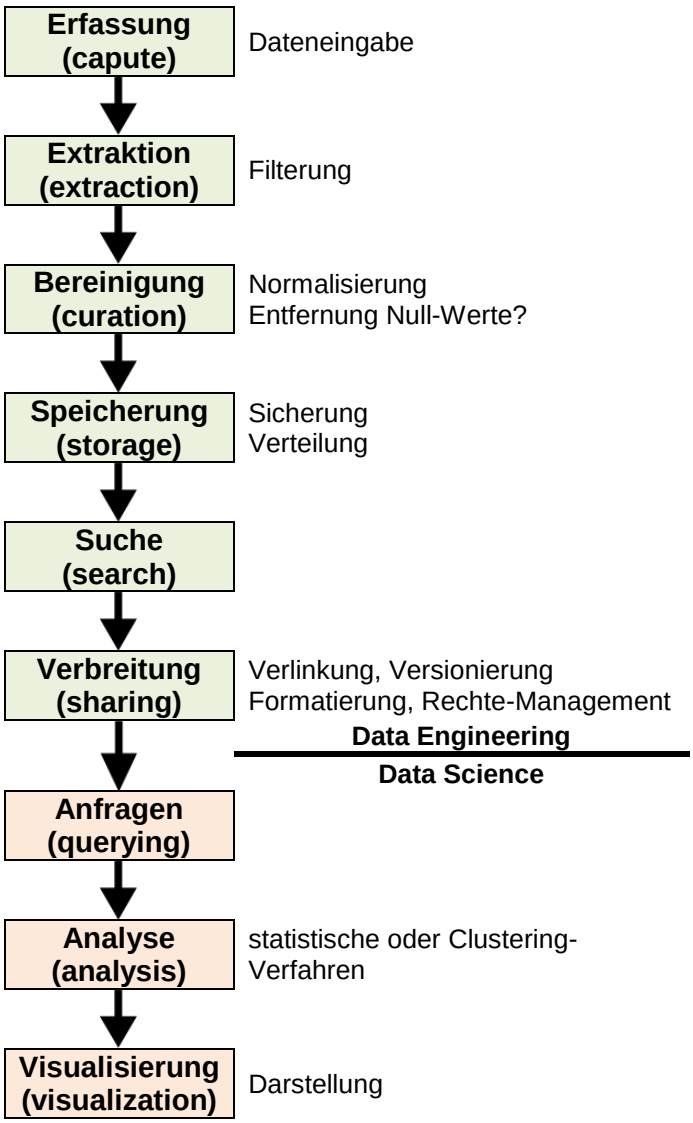
google Flu Trends
Versuch epidermiologische Tendenzen aus Such-Anfragen zu generieren
Projekt bei goggle aufgegeben
von nationalem Interesse (z.B. Planung Impfmittel)
wird derzeit akademisch weiter beforscht (unter Einbeziehung (anderer) sozialer Medien)

8.x.y.4. Data Science

"Data Science"-Pipline

Data Engineering grünlich hinterlegt

reien Data Science rötlich hinterlegt



Empfehlungs-Systeme / kollaboratives Filtern

Recommander

kollaborative filtering

praktisch umgesetzt in Musik-, Buch-/Lese- oder Kauf-Empfehlungen, Kontakt-Vermittlungen z.B. auch im Ranking von Webseiten (in der Suchmaschine google)

"andere Kunden haben auch dieses Produkt gekauft"

Problem bei neuen Kunden / Produkten (cold start problem)
→Anlage von Profilen / Default-Einstellungen / zufällige Werte

Problem langweilige / doppelte Angebote (nach Kauf eines Fernsehers bekomme ich weitere Empfehlungen für gleichartige Fernseher)
→ Erzeugen "zufälliger Entdeckungen" (Serendipität)

Empfehlungs-Systeme können gut validiert werden (weil z.B. die tatsächlichen (zusätzlichen) Käufe beobachtet werden) und dann wiederum verbessert zu werden

8.x.y.3. Data Mining

Wissens-Extraktion / Daten-Schürfen
Extraktion interessanter Daten (nicht-trivial, abgeleitet, vorher unbekannt)

Data Mining-Kategorien

- **Prognosen** Finden von Trend's im Datenzeit-Bezug
- **Assoziation** Suchen nach Abhängigkeiten zwischen Daten-Objekten
- **Segmentierung** Schaffen einheitlicher, homogener (zusammenhängender) Objekt-Teilmengen
- **Klassifikation** Aufteilen der Daten-Objekt in vordefinierte Klassen

geht über SQL und normale Daten-Auswertung nach menschlicher Vorgabe hinaus
Computer übernimmt die Analyse und auch die Auswahl der Methoden
Bildung von Brücken zwischen erhobenen Daten und der Entscheidungs-Ebene (Mensch)

klassische Methodik:
- Beobachtung machen (Observation)
- Modelle bilden (Modelling)
- Vorhersagen ableiten (Prediction)
- Entscheidungen fällen (Decision Making)

moderner Grob-Ablauf (BigData Systems)
1. Daten-Integration
2. DataMining
3. Daten-Visualisierung

Wissens-Entdeckung in Daten

z.B. KDD-Prozess (Knowledge Discovery in Databases)
- Daten-Integration und –Bereinigung → DataWarehouse
- durch Transformation, Selektion und Projektion extrahieren der relevanten Daten (Data Mining) → Muster
- Wissen entsteht erst Kopf des Betrachter / Nutzers
- ev. interativer Rückgriff auf die ersten Schritte des KDD-Prozesses
- ev. Nutzung von Visualisierungsmethoden auf die Rohdaten und frühe (abgeleitete) Daten-Strukturen

Daten-Bereinigung und –Integration beansprucht 60 – 90 % des Analyse-Aufwands

Projektion → Auswahl der Spalten
Selektion → Auswahl der Zeilen
Transformation → Ändern des Daten-Typs / Normierung / Diskretisierung / Aggregat-Bildung / Differenzen-Bildung / …

Methoden:
Cluster-Bildung (Clustering, Gruppen-Bildung; Segmente finden;)
Klassifikation (Classification)
Häufig gemeinsam auftretende Beziehungen (Frequent Itemset Mining)

typische OLAP-Operatoren
Daten-Auswahl (z.B.: SELECT * FROM fakten;)

- Roll up (drill-up) → Aggregation (Daten zusammenfassen)
  - z.B.: SELECT … FROM resultate GROUP BY kriterium;
- Drill down (roll down) → Feingranulierung
  - z.B.: … GROUP BY unterkriterium, zusatzkriterium;
- Slice and dice (Scheiben- / Schichten-/ Kategorie-Auswahl)
  - z.B.: SELECT … FROM … WHERE kriterium;
- Pivot (rotate) → Rotation / Tausch von Spalten und Zeilen
- …

Bewertung der Daten:

- SelfExp: Maß für die Abweichung vom Erwartungswert eines Datums
  - Berechnung als Quotient aus der Differenz zwischen Datum und dem Erwartungswert sowie der (Gesamt-)Standardabweichung
- InExp: Maß für die Lage des Datum unterhalb dem Erwartungswert / einer anderen Zelle
  - Berechnung: ist Maximum der SelfExp für die nächste Feingranulierung
- PathExp: Maß für die Abweichung vom Erwartungswert hinsichtlich einer bestimmten Feingranulierung
  - Berechnung: ist Maximum der SelfExp aller Zellen bei einer speziellen Feingranulierung

statistische Methoden / Verfahren für BigData

Was gibt es überhaupt für Daten?

Werte / Daten			
kategorisch (qualitativ)		**numerisch (quantitativ)**	
nominal / ungeordnet	**ordinal / geordnet**	**diskret**	**kontinuierlich / analog**
ohne innere Ordnung	**innere Ordnung / Ränge / Reihenfolge**	**abgezählt**	**unzählbar**
Farben: gelb, rot, blau, …	Kleider-Größen: XS < S < M < L < XL < XXL	Ganze Zahlen in einem Intervall	Zeit / Zeitreihen
Namen: Klaus, Monika, Bert, …	Grade / Noten: 1 > 2 > 3 > 4 > 5 > 6 (…)	Artikelnummer	Blutdruck Außentemperatur

Dimensionen von Daten:

eindimensional	**zweidimensional**
einfache Zahlenreihen	Punkte im Koodinatensystem
dreidimensional	**multidimensional**
Raumdaten	
hochdimensional	**ohne Dimension**
Zeit-Reihen von Meßdaten	Protein-Faltungs-Strukturen

Wie kann man welche Daten statistisch verarbeiten?

diskretive Statistik:
Mittelwert(e), Median, Maximum, Minimum, Spannweite, Abweichung, Quantile, Ausreißer, Mode (häufigster Wert), …

Gesetzmäßigkeiten / Verarbeitungsregeln / Berechnungsregeln:

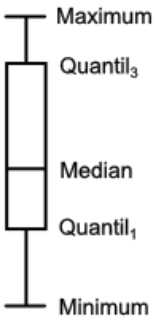
Distributive Maße:
Anzahl(N_1 **U** N_2) = Anzahl(N_1) + Anzahl(N_2)
Summe(N_1 **U** N_2) = Summe(N_1) + Summe(N_2)
Produkt(N_1 **U** N_2) = Produkt(N_1) * Produkt(N_2)
Maximum(N_1 **U** N_2) = Maximum(Minimum(N_1) , Maximum(N_2)
Minimum(N_1 **U** N_2) = Minimum(Minimum(N_1) , Minimum(N_2)

Algebraische Maße (nicht-distributiv!)
Durchschnitt(N) = Summe(N) / Anzahl(N)
Standardabweichung( )
Durchschnitt(N_1 **U** N_2) = Summe(N_1 **U** N_2) / Anzah(N_1 **U** N_2)
Varianz( )

Holistische Maße (nicht distributiv berechenbar)
Median( )
Modalwert() / HäufigsterWert( )
Rang( )

weitere Maße
InterQuantilRang( ) / IQR( ) = $Q_3 – Q_1$
Standardabweichung( )
Ausreißer()

Visualisierung von statistischen Maßen mittels BoxPlot
Darstellung von 5 charakterisierenden Daten / Maßen in einem intuitiv verständlichen System
z.B.: Maximum, 3. Quantil, Median, 1. Quantil, Minimum oder 0%, 25%, 50%, 75% und 100% (entspricht 25 Perzentilen)
weiterhin Visualisierung von Ausreißern, Interquantilsabständen und Streuungen möglich



Visualisierung in Scatterplot Matrizen
bei mehrdimensionalen Daten werden für alle Paare von Dimensionen zweidimensionale Diagramme erstellt → gut für Mustersuche / Beziehungen zwischen den Daten, die über einfache Korrelationen hinausgehen

Parallele Koordinaten-Systeme
ähnlich Netz-Diagrammen
mehrere y-Achsen werden parallel angeordnet und die einzelnen Objekte als verbundene Linien angezeigt → gut für Muster-Erkennung / Gruppierung

Wahrscheinlichkeitsraum ist ein Tripel (Ω, F, P) wobei Ω die Menge der möglichen Erreignisse, F die Potenzmenge der Ereignisse (Ereignisraum) und P die Funktion zur Berechnung des Auftretens einer Wahrscheinlichkeit.

Korrelationen

Spaß-Korrelationen
http://tylervigen.com/spurious-correlations

Daten clustern

K-Means Clustering-Algorithmus

1. Festlegung der (gewünschten / erwarteten) Clusteranzahl n
2. zufällige Festlegung von n Cluster-Zentroiden
3. Wiederholung:
   - Zuordnung der Elemente zu den Zentroiden (geringster Abstand)
   - Berechnung eines neuen Zentroiden aus den gesammten Elementen
4. solange, bis sich Zentroide nicht mehr verändern

Bewertung der Cluster-Bildung mittels Silhouetten-Koeffizienten

- Berechnung der durchschnittlichen Distanz zum Zentroide → a(o)
- Berechnung der durchschnittlichen Distanz zu einem alternativen Cluster / Zentroiden (Unähnlichkeit zu anderem Cluster / Zentroiden) → b(o)
- Silhouette ist 0, wenn
  sonst Quotient aus der Differenz b(o)-a(o) und dem Maximum(a(o),b(o))
- Silhouetten-Koeffizient: Summe(aller Silhouetten) und nachfolge Normierung
- +1 sehr gute Clusterung; -1 sehr schlechte Clustering

statt einem Zentroiden kann auch ein anderer Cluster-Repräsentant verwendet werden (z.B. Medoid)

Dichte-basiertes Clustern

DBSCAN (Density Based Spatial Clustering of Applications with Noise)
braucht keine Vorgaben hinsichtlich der erwarteten Cluster-Zahlen
Ausreißer werden aussortiert und nicht mit in die Cluster einbezogen
Cluster können auch extravagante Formen haben
funktioniert nicht bei hierarchischen / verschachtelte Cluster

DBSCAN – ein Dichte-basierter Cluster-Algorithmus

1. Festlegen des Parameter ε (Radius der ε-Umgebung / max. Abstand)

2. Festlegen des Parameters MinPts (Minimalanzahl Nachbarn in der e-Umgebung um in die wirklich zu clusternden Punkte (Kern-Punkte) zu gelangen
3. Für jeden Punkt:
    - ermittle die Punkte, die innerhalb eines bestimmten Abstandes (ε-Umgebung) liegen → Dichte-Wert
    - Auswählen als Kern-Punkt (core objects), wenn der Punkt mindestens so viele Nachbarn in der ε-Umgebung hat, wie bei MinPts vorgegeben wurde

für jedes Objekt o in der Ausgangsmenge D
    wenn o noch nicht klassifiziert wurde dann
        wenn o ein Kern-Objekt ist dann
            fasse alle Dichte-erreichbaren für o in einem neuen Cluster zusammen
        sonst
            füge o zu den Ausreißer-Objekten hinzu

Um die Parameter ε und MinPts festzulegen gibt es folgende Heuristik:
ε = Knick in der Kurve des Abstand des 4- oder 3-nächsten Nachbarn gegen die (geordneten) Objekte (x-Achse)
MinPts = 2 * d -1 (d … Dimension der Daten)

hierarchische Clusterung

agglomeratives Vorgehen

1. Initialisierung, indem alle Objekt für sich einen Cluster bilden
2. Paarweise Suche der beiden nächstgelegenen Objekte bzw. Cluster und zusammenfassen zu einem Cluster (eine Ebene drüber)
3. entfernen der verschmolzenen Objekte / Cluster aus der Objekt-Menge
4. weiter bei 2 bis alle Objekte / Cluster nur noch ein Cluster bilden (oberste Ebene)

Bestimmung des Abstandes zwischen Cluster / Objekte und Clustern über:
Single-Link (kleinster/minimaler Abstand zwischen den Objekten der einen Gruppe zu den der anderen Gruppe)
Complete-Link (maximaler/größter Abstand zwischen den Objekten der einen Gruppe zu den der anderen Gruppe)
Average-Link (durchschnittlicher Abstand aller Paare aus beiden Gruppen)

diversives Vorgehen

Dichte-basiertes hierarchisches Clustern

adaptives ε

OPTICS (Ordering Points To Identify the Clusteing Structure)
Suche nach dem am nächsten liegenden Punkt bei möglichst kleinem ε
Cluster unterscheiden sich dann durch Wechsel von kleinen ε zu großen und dann wieder zu kleinen ε

Kern-Distanz (core-distance) sucht das e, dass den kleisten Wert darstellt für den vorgegebenen Parameter MinPts

für alle Objekte o aus der Datenbank D
    festlegen o.bearbeitet = falsch
für alle Objekte o aus der Datenbank
    wenn o.bearbeitet = falsch ist dann
        füge Objekt o in die Kontroll-Liste ein
    solange Kontroll-Liste nicht leer ist tue
        wähle das erste Element (o, r-dist) aus der Kontroll-Liste
        retrieve $N_\varepsilon(o)$ und bestimme c_distance = core_distance(o)
        setze o.bearbeitet auf wahr
        schreibe o, r_distance, c_distance in eine Datei
        wenn o ein Kern-Objekt ist mit einer Distanz <= e dann
            für alle Kern-Objekte {p $\in$ $N_\varepsilon(o)$}, die noch nicht bearbeitet wurden
                bestimme r_distanz = reachability_distance(Kern-Objekt, Objekt)
                ist Kern-Objekt noch nicht in der Kontroll-Liste {(p,_) $\notin$ ContolList} dann
                    füge Kern-Objekt {(p, r_distance)} zur Kontroll-Liste hinzu
                sonst wenn {(p,old_r_distance)} Element der Kontroll-List und (r_distance < old_r_distance)
                    aktualisere {(p. r_distance)} in der Kontroll-Liste

Klassifizierung

Ziel ist die Beurteilung von neuen Objekten, um sie bestimmten Klassen zuzuodnen. Die Klassen wurden unter der Nutzung der Objekt-Eigenschaften-Kombinationen bestimmt und dynamisch angepasst. Oft ist nicht bekannt, welche Eigenschaften-Kombinationen genau eine bestimmte Ziel-Klassifizierung bedingt, z.B. die Festlegung, ob ein Kunde eine Hochrisiko-Kunde bei einer Versicherung. Klassifizierung in diesen Sinn ist kontrolliertes Lernen (des Systems).
Das entscheidende ist das Erlernen der Unterscheidung der Klassen.

Klassifikation → binär oder diskret
Vorhersage → numerisch, analog

Klassifizierungs-Verfahren:

- Decision trees (Lernen von Unterscheidungs-Bäumen)
- k-nearst neighbor
- Bayes classifier
- Linear discremant function & SVM

Eigenschaften von Klassifizierern

- Korrektheit; Akkuratheit; geringe Fehlerzahl; hohe Güte
- Interpretierbarkeit; Verständlichkeit
- Effizienz (in der Trainings- und in der Nutzungs-Phase
- Skalierbarkeit
- Robustheit

m-fold Cross Validation ()
klassifizierter Datenbestand wird in m Teil-Datenbestände aufgeteilt
aus m-1 Teil-Beständen wird der Klassifizierer bestimmt
dieser wird auf dem übrig gebliebenen Teildaten-Bestand angewendet und mit der originalen Klassifikation geprüft
das wiederholt man für alle m Teil-Datenbestände

leave-one-out ()
wie m-fold Cross, allerdings bei m = n (es wird nur ein Objekt gegengetestet)

Korrektheit ist Quotient aus der Anzahl der richtigen Klassifikationen und der Gesamtmenge
Fehler-Rate ist Quotient aus der Anzahl der unrichtigen Klassifikationen und der Gesamtmenge
Klassifikations-Fehler (in Abhängigkeit von der Eigenschaften-Anzahl (Parameter)) angewendet auf Trainings-Daten (wird kleiner) oder alternativ auf die Test-Daten (wird größer)

Decision trees
Unterscheidung der Daten-Objekt über nacheinander angewendete Beschneidung der Menge anhand von Grenzen
Grenzen müssen lernend angepasst werden, bis sie zur Klassifikation der Test-Daten passen
dazu benutzt man Klassen-reine Teil-Datenbestände

Algorithmus (Decision trees)
ID3(Examples, TargetAttr, Attributes):
create a Root node for the tree;
if all Examples are positive, return Root with label=+;
if all Examples are negative, return Root with label=-;
if Attributes = 0, return Root with label = most common Value of TargetAttr in Examples;
else
A = the best dicision attribute for next node;
assign A as decision attribute for Root;
for each possible value v_i of A:
generate branch coresponding to test A = v_i;
$Examples_{vi}$ = examples that have value $_{vi}$ for A;
if $Examples_{vi}$ = 0 add leaf node with label = most common value of TargetAttr in Examples;
else add subtree ID3(Examplesvi, TargetAttr,Attributes\A);

Vorteile:
- hierarchisch
- lineare Dimensionen
- schnell auf diskreten und numerischen Daten
- schnelles / effizientes Lernen / Klassifizierung
- gute Genauigkeit der Bäume → gefunden Klassifizierung
- für Menschen gut verständliche Klassifizierung

Nachteile:
- nicht stabil: kleine Veränderungen in den Daten können große Veränderungen im Baum erzeugen

Nearest Neighbor Classifiers
Suche nach dem ähnlichsten (vergleichbaren) Datensatz; Berechnungen von Distanzen
Bewertung / Klassifizierung dann entsprechend diesem Nachbarn
Instanz-basiertes Lernen; Lazy evaluation
gesamte Arbeit erfolgt gleich in der Klassifizierung (kein vorheriges Training notwendig)
geht auch über vorherige Bestimmung von Clustern-Repräsentanten (z.B.: Zentroid)
hohe Klassifizierungs-Güte
gute Anpassung an große Datenmengen

man braucht:

- eine Distanz-Funktion
- Entscheidungs-Menge (Decision set) (empirisch: $1 << k < 10$
- Entscheidungs-Regeln (Decision rule) (z.B. nach Mehrheit aus Entscheidungs-Menge; Normierung mit Kehrwert der Distanz möglich; Häufigkeiten; kleinste Distanz)

nachteilig sind:
relativ naiv (kein hinterlegtes Modell)
zur Bewertung ev. gesamter Datenbestand anzufassen; umgehbar durch Indizierung / Sortierung
bei höherer Dimension in den Daten-Beständen gehen immer mehr irrelevaten Daten mit ein
ev. Festlegung relevanter Attribute notwendig

Bayesian Classifier
Beurteilung über Wahrscheinlichkeiten der Zugehörigkeit zu den Klassen
probalistisch
Klassen müssen vorher analysiert werden (Lern-Phase)
Berechnung der bedingten Wahrscheinlichkeit
schwierig bei hoch-dimensionalen Daten

Linear Classifier
Berechnung einer Trenn-Linie (Gerade) bzw. einer Trenn-(Hyper-)Ebene zwischen den verschiedenen Klassen
sehr einfach
läßt auf nicht-lineare Klassifikation erweitern

empfindlichen gegenüber Ausreißern; also nicht stabil
sehr optimistische Annahme. dass Klassen linear zu trennen sind
bei höheren Dimension recht rechenaufwändig

Support Vector Machines (SVMs)
stabiles Lernen der trennenden Hyper-Ebene
nicht-lineare Trennung möglich
ev. Transformation in einen linear Raum
durch Kernel.Methoden wird Aufwand reduziert

Ensemble Classification
Suche nach dem / einen optimalen Klassifizierer
Kombination mehrerer Klassifizierer
z.B.: Bagging
Durchschnitt mehrerer Klassifizierer über zufällig gebildete Teil-Datensätze aus einem Trainings-Datenbestand
z.B.: Boosting
Kombination mehrerer schwacher Klassifizierer zu einem starken
neu hinzukommende Klassifizierer sollen sich auf die Objekt mit bisher fehlerhaften Zuordnung konzentrieren

Frequent Itemset Mining
Suche nach wiederkehrenden / gehäuften Mustern
in Transaktions-Datenbanken (z.B. Einkäufe)
z.B. Kombinationen von bestimmten Produkten kaufen → Produkt-Vorschläge für weitere Einkäufe (recommendation systems)

erste dokumentierte Beziehung in Amerika, am Freitag Nachmittag gab es gehäufte Einkäufe von Windeln und dazu Bier → Ausnutzung dadurch, dass neben Windel-Aufsteller gleich auch eine Palette mit Bier steht (und umgekehrt)

Apriori-Prinzip
Apriori-Algorithmus
Es wird für jede Anzahl-Element-Kombinationen eine Kandidaten-Menge generiert und diese analysiert (auf Häufigkeit (geforderte Minimal-Häufigkeit)
die häufigen Kandidaten werden in eine Lösungs-Menge übernommen
aus den Lösungs-Kandidaten werden nur Kombinationen mit einem zusätzlichen Element generiert und in die Kandidaten-Menge der Anzahl+1-Element-Kombination übernommen
usw. usf. bis keine Kombination mehr die Minimal-Häufigkeit erfüllt

Frequent Pattern Tree
Elemente werden nach Häufigkeit sortiert und ev. gegen eine Minimal-Häufigkeit abgeschnitten
aus Transaktionen wird unter Beachtung der Häufigkeiten (innerhalb der Transaktionen) ein bedingter Baum mit Häufgkeitszählern in den Knoten konstruiert

Assoziations-Regeln

support(A → B) = Anzahl_der_Transaktionen(A, B) / Gesamt_Anzahl_Transaktionen
(Häufigkeit de Beziehung)
support(A → B) = support(A,B)

confidence(A → B) = Anzahl_der_Transaktionen(A, B) / Anzahl_der_Transaktionen(A)
(Stärke der Beziehung)
confidence(A → B) = support(A,B) / support(A)

unexpected(A → B)
(Unerwartungswert)

actionable(A → B)
((Aktions)Nutzbarkeit)

Lift = Häufigkeit(A,B) / Häufigkeit(A)*Häufigkeit(B) = Häufigkeit(A → B) / Häufigkeit(A)
= Häufigkeit(B → A) / Häufigkeit(B)

Outlier Mining

Ausreißer-Suche und -Klassifizierung
Anwendung:

- Erkennung von Anomalien
  - z.B.: bei Erkrankungen
- Kreditkarten-Mißbrauch
- Wartungs-Zyklen für Bauteile / Ausfallhäufigkeiten
- Auffinden von Rausch-Daten
-

Def. z.B. über kleine Cluster oder seltene Objekte, die scheinbar einem anderen Modell entstammen oder bei klassifizierten Objekten, die keiner Standard-Klasse zugeordnet sind
große Abstände zu anderen Cluster; spärlich besetzte Cluster
Objekte mit geringer Häufigkeit
Objekte an den Rändern von Cluster / Häufigkeits-Verteilungen
völlig neue Daten

Distanz-basierte Suche nach Ausreißern

Auffinden von gehäuften Randwerte bezogen auf eine GAUß-Verteilung z.B. jeweils 2,5 % der Daten, d.h. die restlichen 95 % werden als regulär betrachtet
setzt voraus das es sich bei den Daten um eine GAUß-Verteilung handelt (stimmt selten!)
versagen bei unterschiedlichen Dichten

Dichte-basierte Suche nach Ausreißern

suche / finden von lokalen Ausreißern
Objekte würden nur zur normalen Gruppe gehören, wenn (deutlich) veränderte Parameter benutzt werden müsste
k-nächste-Nachbar-Distanz ist deutlich größer / deutlich größer als die k-nächste-Nachbar-Distanz der bei k Nachbarn (des Prüf-Objektes)
man erhält Liste der k-nächste-Nachbar-Distanzen, die ausgewerte werden kann
z.B. kann der Nutzer die Grenze anhand der sortierten Liste festlegen (Ranging-Grenze)
z.B. über die lokale Erreichbarkeits-Dichte (Bewertung der Dichte um ein Objekt)

lokale Erreichbarkeits-Dichte = 1 / durchschnittliche Erreichbarkeits-Distanz(minPunkte)
LOF Lokaler Outlier Factor = 1 → Inlier; >>1 → Outlier; wenig >1 → lokale Outlier

Ausreißer in multi-dimensionalen Räumen

Subspace Mining

aufgrund verschiedener Sichten auf die Daten ergeben sich unterschiedliche Gruppen
Suche von Clustern und Ausreißern in Teilgruppen der Attribute
relaiv offen hinsichtlich der ausgewählten Attribute
erster Ansatz sind Auswertungen von zufälligen Projektionen

Einsatz von "Monte Carlo"-Algorithmen

Data Mining bei kontinuierlichen Datenströmen

unendliche Daten(-Ströme) / Zeit-Reihen; Gesamtdatenbestand nicht greifbar
z.B. Kursdaten von Aktien
auch Daten aus verschiedenen Kategorien
Finden von Anomalien
z.B. bei Transaktionen (Kreditkarten); Sensordaten (Auto's, Reaktoren, …)

Data Mining bei Graph-Daten / in Graphen

z.B. soziale Netzwerke (Knoten = Personen; Kanten sind Verbindungen)
Problem, dass Anzahl der Kanten deutlich größer ist als die Anzahl der Knoten

semantische Beziehungen in Daten / Wissen (Wissens-Graph) (z.B. auch Bookmarking)

biologische Daten
z.B. Protein-Protein-Interaktionen; Substrat-Protein-Interaktionen; Metabolismen-Netzwerke

attributierte Graphen = Graphen mit Knoten, die mehrdim. Attribute besitzen

Suche nach homophilen Gruppen bzw. korrelierten Attributen

Q: http://www.cs.uiuc.edu/~hanj/bk3
(ältere Version: http://web.engr.illinois.edu/~hanj/bk2/slidesindex.htm)

Links:
http://www.dataminingbook.info (ZAKI, Mohammed J.; MEIRA, Wagner jr.: "Data Mining and Analysis")
https://hpi.de/mueller/tutorials/graph-exploration-sigmod.html ()

Supermarkt-Kette benutzte Kundenkarten
analysierte Kauf-Verhalten von schwangeren Frauen
Interesse lag auch in der Frage, mit welchen Coupon's man Kunden eher glücklich macht

Analyse, was die Kunden neu kauften (z.B. Parfüm-freie Körperlotion, bestimmte Lebensmittel-Ergänzungen (Mg, Ca, Zn)
und was sie nicht mehr kauften (Zigaretten und Alkohol)

es wurden 25 Artikel beobachtet und die junge Frau kam auf 23 Treffer

das System ermittelte eine Wahrscheinlichkeit von 87 %, dass sie schwanger ist und konnte den Geburts-Termin (ungefähre Dekade im Monat) voraussagen

Anekdote:
Ein amerikanischer Vater regt sich bei der Geschäftsleitung der Supermarkt-Kette "Target" darüber auf, dass seine Tochter seit kurzem verstärkt Werbung (eMail's) für Artikel erhält, die etwas mit einer Schwangerschaft zu tun haben. Sie gehe schließlich noch zur Highschool und ob man sie zu einer Schwangerschaft ermutigen wolle?
Der – den Fall bearbeitende – Manager hatte keine Ahnung und kontrollierte die eMails. Sie enthielten tatsächlich Werbung für Mutterschafts-Kleidung und Kinderzimmermöbel sowie lächelnde Baby's. Ein paar Tage später entschuldigte sich dann der Manager für die "nicht-angepassten" eMail's.
Später hat der Vater seine Tochter zur Rede gestellt und diese gab beschämt zu, dass sie schwanger sei und im August gebären werde.
Der Mann entschuldigte sich dann später bei der Supermarktkette für seine unberechtigte Beschwerde. Offensichtlich hätte es in seinem Haus Aktivitäten gegeben, von denen er nichts mitbekommen habe.

Ausnutzung von Beziehungen in Daten (z.B. gemeinsamer Kauf von Windeln mit Bier durch Männer → z.B. Ausnutzung durch gemeinsame Produkt-Angebote / Produkt-Präsentation / Produkt-Plazierung im Markt)

8.x.y.z. Data Mining an Texten – Text-Analysen, …

Texte sind beliebte / zuverlässige Wissen-Speicher
besonders interessant, weil viele Daten in Text-Form vorliegen

- Bücher
- Zeitungen
- Zeitschriften
- eMail's
- social-media-Beiträge (tweets, likes, …)
- Patente
- Lexika
- wikipedia-Seiten
- Webseiten

Text Mining

Text Mining-Bereiche

- **Information Retrieval** **Text-Analyse**	Statistik
- **Natural Language Processing** **Verarbeitung von (natürlicher) Sprache**	Grammatik Text-Struktur-Erkennung
- **Knowledge Extraction** **Wissens-Extraktion**	Semantik Zusammenhänge von Text-Informationen

TF-IDF (term frequency – inverse document frequency)

term frequency … Term-Häufigkeit (Wort-Häufigkeit)
inverse document frequency … umgedrehte Häufigkeit eines Terms in den Dokumenten
Grund-Maß des Text-Mining's
Maß für die Relevanz von Worten in einem Text-Korpus

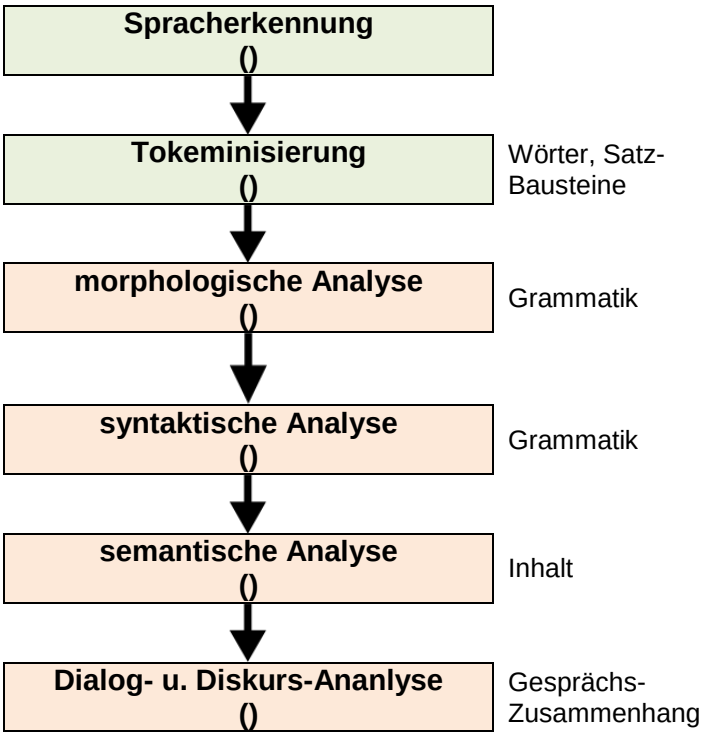
Worte mit einer sehr hohen Dokumenten-Häufigkeit sind wahrscheinlich ohne größere Bedeutung und werden als Stop-Worte verstanden (z.B. Bindeworte, Artikel, …)

ein hoher TF-IDF-Wert besagt, dass ein Wort in einem bestimmten Dokument besonders häufig vorkommen, aber in allen Dokumenten zusammen eher selten ist → Wort mit besonderer Bedeutung

Computer-Linguistik

Natural Language Processing

wenn akustische Signale in geschriebene Texte umgesetzt werden sollen



Tokenisierung zerlegt die Sätze in einzelne Einheiten – meist Wörter
zuerst müssen aber z.B. die Sätze selbst selektiert werden
ev. auch Zerlegung von zusammengesetzten Substantiven od.ä. in einzelne Begriffe, um sie besser einordnen zu können

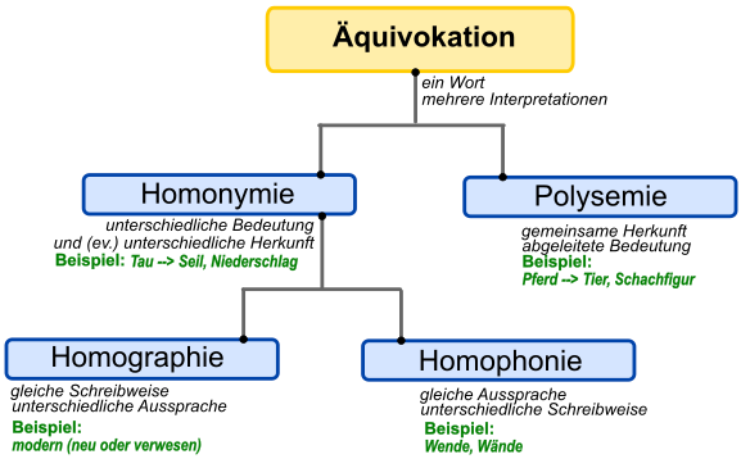
die Grammatik der Sätze steht bei der morphologischen Analyse im Vordergrund
deklinierte Worte werden dann in die Grund-Form – oder sogar nur den Wort-Stamm – zurück übertragen, um sie im Weiteren einfacher auszuwerten

die syntaktische Analyse bestimmt nun die Satz-Bausteine / -Teile mit ihrer grammatikalischen Form
z.B. müssen ja Objekt und Subjekt klar voneinander unterschieden werden können

den Satz-Teilen wird dann in der semantischen Analyse eine Bedeutung zugewiesen
hier sind jetzt auch Bezüge zu anderen (vorlaufenden) Sätzen oder Weltkenntnisse notwendig, um einen Satz inhaltlich verstehen zu können

z.B.: Ein Junge wandert die Straße entlang. Michael singt leise vor sich her.
Das semantische Erkennungs-System muss hier z.B. erkennen, dass Michael ein Junge ist und derjenige ist, der im Satz davor gewandert ist.

besonders schwierig wird das, wenn ein Wort sehr viele unterschiedliche Bedeutungen haben kann und sich die gemeinte Bedeutung erst aus dem anderen Satz-Teilen oder dem gesamten Text ergibt.
weitere Probleme beim Übergang von Namen zu Spitznamen od.ä., indirekten Personen-Bezügen, …
z.B. Hamburger Menü
kann ein spezielles Menü in Gaststätten von Hamburg sein, es kann sich aber auch um ein Menü mit einem Hamburger (einer gebratenes Hackfleisch-Scheibe) handeln
eine weitere Verwendung hat der Begriff bei den modernen Apps für die gestapelten Menü-Einträge, die sich hinter einem Symbol mit meist drei dickeren



Strichen versteckt

Aufgaben:

1. *Wählen Sie sich ein Wort (zusammengesetztes Wort usw.) aus und tragen Sie dazu besonders viele Bedeutungen zusammen! Wer findet das Wort mit den meisten unterschiedlichen Bedeutungen!*
2. *2.*
3. *3.*

Beispiel für eine semantische Suchmaschine ist "WOLFRAM alpha" (https://www.wolframalpha.com/)
hier kann man englischsprachige Fragen und Wortgruppen eingeben und erhält passende Antworten und / oder Suchergebnisse

Named Entity Recognition

Benennung von erkannten Objekten / Begriffen / Identitäten

einfachter Ansatz ist ein Wörterbuch-basierter Ansatz
dort sind die Worte mit ihrer möglicher Bedeutung verzeichnet

heute versucht man mittels maschinellen Lernen die Analyse flexibler und problem-orientierter durchzuführen
als Trainings-Texte können z.B. gut Lexika oder auch Hypertexte (HTML-Seiten) genutzt werden, da hier viele Entitäten als Links auf andere Seiten / Hinweise auf andere Stichworte ausgeführt sind
besonders gut funktioniert dies z.B. mit wikipedia-Texte, die diese sowohl als Lexika und als HTML-ähnlicher Text fungiert
außerdem sind viele Seite kategorisiert

Named Entity Disambiguation

disambiguieren
nach der Erkennung von Worten soll jetzt eine Identifikation (konkrete Bedeutungs-Zuordnung) durchgeführt werden

semantische, maschinen-lesbare Datenbanken:

- **wikidata** https://www.wikidata.org/
- **DBpedia** Datenbank mit semantischen Informationen aus wikipedia https://wiki.dbpedia.org/
- **Yago** https://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/yago/

Relationship Extraction

Herstellen von gesicherten Beziehungen zwischen den Entitäten
prüfen von bindenden Wörtern zwischen verschiedenen Entitäts-Typen
ermitteln von Richtungen von Beziehungen (Wer liefert was? Was ist die Ursache, was die Wirkung? Wer ist der Erwachsene und wer das Kind? …)

Linguistische Maße

Anzahl der Wörter

durchschnittliche Wortlänge

durchschnittliche Anzahl von Silben in den Wörtern

Anzahl der Substantive, Verben, …

Anzahl der Sätze mit bestimmten Bestandteilen

Anzahl der Nebensätze

Analyse der durchschnittlichen Satzlänge und deren Visulisierung:
https://www.uni-konstanz.de/mmsp/pubsys/publishedFiles/KeOe07.pdf

Verse length (Vers-Länge)

Hapax Legumena
Anzahl der Worte, die einmal im Text vorkommen

je mehr solcher Worte in einem Text vorkommen, um so komplizierter / ausschweifender / phantastischer ist dieser

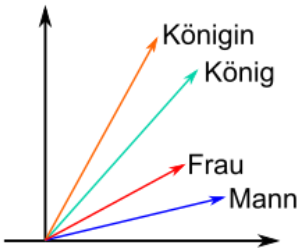
Hapax Dislegumena
Anzahl der Worte, die genau zweimal im Text vorkommen

darüber typische Charakterisierung von Texten und damit auch von Autoren möglich

Word Embedding (Wort-Einbettung)

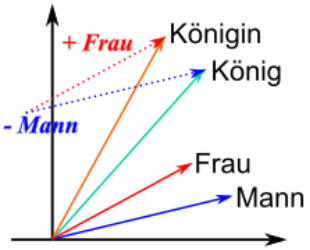
Anordnung von Wörtern in einem mehrdimensionalen Raum in der Form, dass Worte mit ähnlicher Bedeutung näher in diesem Raum stehen als solche, bei denen keine Beziehung besteht

z.B. Erkennen oder Abfragen von Synonymen (diese liegen im Raum dicht beieinander)
läuft praktisch vollständig automatisiert und wird stark durch maschinelles Lernen getrieben



praktisch kann man dann auch mit den Vektoren arbeiten und rechnen
daraus lassen sich dann wieder semantische Beziehungen herstellen

z.B.: gegeben sind die Vektoren von Mann, Frau, König und Königin
über die "Vektor.Berechnung" König – Mann + Frau erhält man dann Königin (oder einem synonymen Begriff (oder zumindestens einen in dessen Nähe))



Sentiment Analysis (Stimmungs-Analyse)

erkunden der Stimmung, Gefühle, Meinung, Haltung, Parteilichkeit, … eines Textes

z.B. Produkt-Bewertungen, Post's in Social Media usw. eintufen

google-mail hat mal getestet, die eMail's vor dem Abschicken zu analysieren und gegebenenfalls empohlen, den Text erst am nächsten Tag abzusenden

z.B. über Stichwort-Listen

auch gute Anwendung für maschinelles Lernen
z.B. aus Kombinationen aus textuellen Bewertungen und vergebenen Sternen

praktisch heute schon in vielen Social Media angewendet → Filter-Blase
Nutzer bekommen nur noch positiv gestimme / anregende / interessante / … Beiträge zu sehen, um ihn weiter im Medium zu halten
praktisch werden vor allem die eigenen Meinungen usw. usf. wieder angezeigt, so dass man den Eindruck hat, dass wäre die Welt-Meinung
echter Pluralismus so nicht (mehr) möglich

8.x.y.z. BigData zum selber Ausprobieren

8.x.y.z.1. google BigQuery

BigQuery ist eigentlich Kosten-pflichtiger google-Dienst

mit einem google-Konto aber 60 Tage zum Testen Kosten-frei

cloud.google.com/bigquery

mit einer Meta-Programmiersprache auch digital und automatisiert nutzbar

"BigQuery ML" basiert weitesgehend auf SQL

8.x.y. Smart Data

Verarbeitung großer Datenmengen, die sich aus der Vernetzung von Geräten ergeben
Bereitstellung von "kleinen" effektiven / benutzbaren Daten aus Big Data

???y
welche Daten sind für die aktuelle Station wichtig
welche Daten sind für das Netz relevant, wovor muss gewarnt werden, worauf hingewiesen

8.x.y. Skalierbares Daten-Management

8.x.y.0. Parallelisierung

großer Bedarf an gleichzeitigen Abarbeitungen (Zeit-gleiche Bestellungen, …)

Parallelisierungs-Arten

- **Aufgaben-Parallelismus** **task parallelism**	unabhängige / separate Task's werden erledigt
- **Anweisungs-Parallelismus** **instruction(-level) parallelism**	unter Nutzung von mehreren Prozessoren (CPU's) / Prozessor-Kernen (Core's) / Graphik-Prozessoren (GPU's) werden die gleichen Bearbeitungs-Schritte zur gleichen Zeit erledigt Daten müssen weitgehend unabhängig sein (zumindestens während der Verarbeitung)
- **Daten-Parallelismus** **data parallelism**	verschiedene Daten werden verschiedenen Datenträgern gespeichert sie können aber ohne weiteres gleichzeitig und gleichartig bearbeitet werden typisch für Graphik-Prozessoren (GPU's)

Grenzen

Blade → Rack → Data Center / Rechenzentrum
Ausfälle

- ein typisches Beispiel für ein Rechenzentrum (nach Jeff DEAN (2009)):
  - innerhalb von 2 Jahren 1x eine Überhitzung → Herunterfahren aller Rechner innerhalb von 5 min notwendig und das schrittweise Hochfahren der einzelnen Rechner über 1 bis 2 Tage
  - 1x jährlich Ausfall eines Strom-Verteilers → 500 – 1000 Rechner sind aus; Reparatur- und Wieder-Aktivierungs-Zeit um die 6 Stunden (ev. große Daten-Verluste, weil Daten nur im Hauptspeicher waren)
  - 1x jährlich in einem Bereich fällt die Kommunikation aus, oder funktioniert nicht mehr richtig → rund 6 Stunden um die Geräte zu prüfen und ev. alle neu zu starten
  - alle 2 Monate gehen einzelne Racks in undefinierte / verwirrte Zustände über; meist bei unklaren Ursachen (z.B. im Betriebssystem, …) → bei vielen Maschinen ist mit Daten-Verlusten zu rechnen
  - 1000 Rechner fallen insgesamt aus (praktisch 3 pro Tag)
  - Tausende an Hardware-Komponenten fallen aus (schlechter Hauptspeicher, zulangsame oder überlastete Festplatten, …)
  - Datenleitungen, die von Tieren angefressen werden / bei Baggerarbeiten durchgetrennt werden / …

Energie-Verbrauch steigt expotentiell

AHMDAHLs Gesetz

die Beschleunigung eines Programmes ist durch den nicht-parallelisierbaren Anteil beschränkt
je größer der nicht-parallelisierbare Anteil ist, umso weniger läßt sich das Programm insgesamt beschleunigen

$$S_{max} = \frac{1}{(1-f)+\frac{f}{p}}$$

p … Anzahl der Prozessoren
f … parallelisierbarer Anteil

angenommener Anteil von parallelisierbaren Code-Abschnitten liegt bei 90 % (was allgemein sehr hoch ist (nur bei Graphik-Berechnungen typisch))
der Beschleunigungs-Effekt S liegt z.B. bei 10 statt 1 Rechner nur bei einem Faktor von 5,3 und nicht 10, wie man eigenlich gehofft hat

z.B.:
f = 0,9
bei 10 Prozessoren

$$S_{10} = \frac{1}{(1-0{,}9)+\frac{0{,}9}{10}} \approx 5{,}3$$

bei 20 Prozessoren

$$S_{20} = \frac{1}{(1-0{,}9)+\frac{0{,}9}{20}} \approx 6{,}9$$

8.x.y.z. OLAP

Online Analytical Processing
große Daten-Mengen analytisch auswerten
wenige Anfragen pro Zeiteinheit mit sehr großen Ergebnis- bzw. betroffenen Daten-Mengen
meist sehr aufwändig / rechen-intensiv / viele Arbeitsschritte (z.B. statistische Auswertungen, Visualisierungen, …)

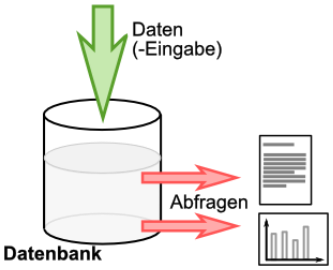
verteilte (Datei-)Systeme

ein Rechner ist der Name-Node (Namens-Knoten) er hat die Funktion eines Server's
bei ihm passiert die Datei-Speichung nur virtuell, er gibt nur "seinen guten Namen" dafür her
praktisch sorgt der Name-Node für die Verteillung der Daten (Dateien oder Datei-Segmente) auf mehreren Data-Note's (Daten-Knoten)
typische Daten-Segmente haben in der Praxis eine Größe von 128 MB
in verteilten Systemen sind die Daten häufig mehrfach (typischerweise dreifach) vorhanden, um Ausfällen vorzubeugen und damit eine hohe Datensicherheit zu garantieren
weiterhin ergeben sich Möglichkeiten zum Lasten-Ausgleich
Nachteile der mehrfachen Speicherung sind erhöhte Hardware-Aufwändungen und eine hohe Daten—Redundanz
die genaue Verteilung und Absicherung der Konsistenz ist ein technisches Problem, was in speziellen verteilten Datei-Systemen realisiert wurde und für den Nutzer nicht sichtbar wird
z.B.: google-Datei-System, S3-System von amazon, HADOOP
heute übernehmen die Daten-Knoten auch noch Bearbeitungs-Aufgaben zu den abgelegten Daten, damit spart man sich den teuren Transport der Daten über das Netz
statt dessen werden nur die Ergebnisse zum Namens-Knoten geschickt
die Aufgaben werden von Client's an den Namens-Knoten gestellt, dieser verteilt sie an die Daten-Knoten (→ verteilte Berechnungen) und die schicken eben die Ergebnisse an den Namens-Knoten zurück
dieser bedient dann die Client's
Beispiel: BOINC-Projekte (verteiltes Berechnungs-System mit Bildschirm-Schoner-Funktion) (→ boinc.berkeley.edu)

Lösung wissenschaftlicher Probleme (Berechnung von Protein-Strukturen, Klima-Modellen, …; Suche nach außerirdischen Funksignalen, Gravitations-Wellen, …; Lösen von mathematischn und kryptologischen Problemen)

Daten-Verarbeitungs-Pinzipien

Batch-Verarbeitung



arbeiten auf einem sehr großen / dem gesamten Daten-Bestand
charakterisiert durch sehr komplexe Programme, die nacheinander (als Stapel (Batch)) abgearbeitet werden (müssen)

meist ohne Nutzer-Interaktionen

Ergebnisse sind große Analysen mit sehr komplexen Visualisierungen usw.
typisch Quartal's- oder Tages-Auswertungen, …
begrenzender Faktor ist hier die Größe des Speicher's; praktisch nur Volume-Problem (→ Big V's)
für die komplexen Analyse-Programme ist aber auch Leistungs-fähige Hardware notwendig

Datenstrom-Verarbeitung

(data)stream processing
Daten-Verarbeitung beschränkt sich auf einen aktuellen Daten-Ausschnitt (laufende Bestellungen oder Bestellungen der letzten Stunde usw.usf.; Sensor-Daten, Log's, …)

neben Volume ist hier auch Velosity interessant

meist nur einfache Programme (z.B. Grenz-Wert-Überwachung)

Ergebnisse sind flüchtige Kennzahlen für die betrachteten Zeiträume
meist nur einfache Visualisierungen; z.B. Kauf-Vorschläge für Artikel, die man sich länger / genauer angesehen hat
Effektivität und die Grenzen werden durch die Rechen-Kapazität bestimmt; Speicher interessiert kaum

an die Datenstrom-Analyse kann dann auch die Speicherung und eine weitere / unabhängigge Batch-Verarbeitung anschließen

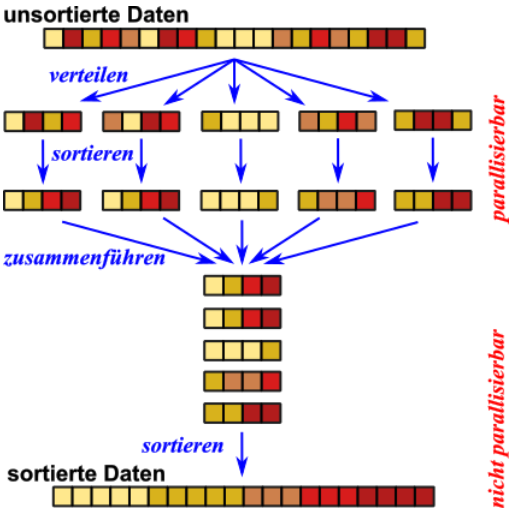
Problem: Skalierung

Wie reagieren meine Systeme, wenn sich die Menge der Daten im größeren Stil verändern?
Was passiert, wenn Geräte ausfallen? Schaffen die anderen dann die Aufgaben?
Was passiert, wenn neue Daten(-banken) mit eingebunden werden sollen?

→ Skalierungs-Muster

???

- Phase 0: Daten verteilen
- Phase 1: Berechnungen auf Teilmengen (der verteilten Daten)
- Phase 2: Zusammenführen der Ergebnisse aus der Teilmengen-Analyse zu einer Gesamt-Analyse (auf die betrachtete, unverteilte Daten-Menge)



die Vorsortierung kleinerer Daten-Pakte auf den verteilten Systemen ist deutlich effektiver, als ein vergleichsweises gemeinsames Sortieren auf nur einem Rechner. Das Vorsortieren lässt sich zudem parallelisieren.
die vorsortierrten Daten-Blöcke werden dann an einer Stelle – meist ist dies der Name-Node – zu einer Gesamt-Sortierung zusammengesetzt.
Dieser Teil ist nicht parallelisierbar, weil jetzt Abhängigkeiten zwischen allen Daten bestehen.
im letzten Schritt kann aber auch wieder Verteilen eingearbeitet werden.
So könnte der Name-Node immer jeweils alle gleichwertigen Element auf einen der Data-Node's schieben und mit den anderen Elementen genau so verfahren. Aus der Sicht des Name-Nod's kann dann über einen Zugriff auf die Data-Note's in der richtigen Reihenfolge die sortierte (Gesamt-)Reihenfolge erzeugt werden.
Neben Sortierungen eignen sich auch Gruppierungen, Klassifikationen und Aggregationen gut für das obige Verfahren. Wichtig ist, dass der parallelisierbare Anteil groß genug ist.
Ein klassisches Beispiel für eine Aggregation ist die Summen-Bildung. Hier kann man auf den verteilten System z.B. Teil-Summen bilden und nur die dann an den Name-Node zurückschicken. Dieser kann aus den teil-Summen dann eine Gesamt-Summe bilden.
Ähnlich funktionier dies für Produkte (Multiplikationen), bestimmte Mittelwerte und Zählungen.

Map/Reduce

ist ein Programmier-Modell für die parallele Abarbeitung von Aufgaben
entwickeln von Jerrey DEAN und Sanjay GHEMAWAT (google Inc.)
gearbeitet wird eine Rahmen aus zwei Funktionen – eben Map und Reduce
innerhalb von Map und reduce können wiederum andere Funktionen aufgerufen werden

Map $(k_1 \times v_1) \rightarrow (k_2 \times v_2)$	nimmt Schlüssel-Wert-Paar (key, value) entgegen, wandelt diese irgendwie geartet um und gibt ein neues Schlüssel-Wert-Paar aus (diese Operationen können vollständig parallel für jedes Schlüssel-Wert-Paar gemacht werden, diese immer unabhängig voneinander sind!)
Reduce $(k_2 \times list(v_2)) \rightarrow (v_3)$	nimmt einen Schlüssel sowie eine Liste mit allen Werten, die genau diesen Schlüssel haben und erzeugt aus dieser Liste einen neuen Wert (z.B. die Summe) hier kann wieder für alle Schlüssel (k_2) parallel gearbeitet werden (weil für diesen wieder alle Werte unabhängig von den anderen sind)

zwischen Map und Reduce liegt aber ein Daten-zusammenfassender Schritt, der sich nicht parallelisieren läßt

die im Map-Teil berechneten Schlüssel-Wert-Paare müssen neu angeordnet / umsortiert / zusammengefasst / gruppiert werde, damit sie dann in der Reduce-Teil wieder parallel weiter verarbeitet werden können

typische Aufgabe: gleiche Wörter-Zählen
(praktisch das Hello-Welt-Beispiel für Map-Reduce)

```
map(dateiname, zeile){
    for each (wort in zeile)
        emit(wort, 1);}
```

aus:	wird:
O Romeo! Warum denn Romeo? Verleugne deinen Vater, deinen Namen, Romeo!	O, 1 Romeo, 1 Warum, 1 denn,1 Romeo, 1 Verleugne, 1 deinen, 1 Vater, 1 deinen, 1 Namen, 1 Romeo, 1

```
reduce(wort, nummern){
    int summe = 0;
    for each (eintrag in nummern){
        summe += eintrag;}
    emit(wort, summe);}
```

aus:	wird:
O, 1 Romeo, 1 Warum, 1 denn,1 Romeo, 1 Verleugne, 1 deinen, 1 Vater, 1 deinen, 1 Namen, 1 Romeo, 1	O, 1 Romeo, 3 Warum, 1 denn, 1 Verleugne, 1 deinen, 2 Vater, 1 Namen, 1

auch der einzelne Text kann parallelisiert verarbeitet werden, vielleicht z.B. so verteilt:

für den Programmier ergibt sich keine veränderung, er schreibt die gleichen Befehle auf

```
map(dateiname, zeile){
    for each (wort in zeile)
        emit(wort, 1);}
```

das System (Compiler, Systemsteuerung) entscheidet, wie in den verteilten Systemen die Teilaufgaben ausgeführt werden

	aus:	wird:
Rechner 1	O Romeo! Warum denn Romeo?	O, 1 Romeo, 1 Warum, 1 denn,1 Romeo, 1
Rechner 2	Verleugne deinen Vater, deinen Namen, Romeo!	Verleugne, 1 deinen, 1 Vater, 1 deinen, 1 Namen, 1 Romeo, 1

auch den Reduce-Teil programmiert man genau so wie bei einem Rechner, ein ev. verteiltes System wird intern mit den paralelelisierbaren Teil-Aufgabe versorgt

```
reduce(wort, nummern){
    int summe = 0;
    for each (eintrag in nummern){
        summe += eintrag;}
    emit(wort, summe);}
```

	aus:	wird:
Rechner 1	O, 1 Romeo, 1 Warum, 1 denn,1 Romeo, 1	O, 1 Romeo, 2 Warum, 1 denn, 1
Rechner 2	Verleugne, 1 deinen, 1 Vater, 1 deinen, 1 Namen, 1 Romeo, 1	Verleugne, 1 deinen, 2 Vater, 1 Namen, 1 Romeo, 1

	aus:	wird:
Rechner 1	O, 1 Romeo, 1 Warum, 1 denn,1 Romeo, 1	O, 1 Romeo, 3 Warum, 1 denn, 1 Verleugne, 1 deinen, 2 Vater, 1 Namen, 1
Rechner 2	Verleugne, 1 deinen, 1 Vater, 1 deinen, 1 Namen, 1 Romeo, 1	

Beispiel: gemeinsame Bekannte finden
Problem z.B.: 1,4 Mrd. Facebook-Nutzer (Stand: 2016) mit durchschnittlich 155 Freunden ergibt 979'999'999'300'000'000 Paare zum Prüfen:

```
map(person, freundesliste){
    for each (freund in freundesliste)
        if(freund < person)
            emit(<freund, person>, freundesliste);
        else
            emit(<person, freund>, freundesliste);}

reduce(<person1, person2>, freundesliste){
    emit(person1, person2>, freundesliste[1] ∩  freundesliste[2]);}
```

Hadoop

Hadoop Distributed File System (HDFS) von der Apache Software Foundation betreut

Q: Apache Software Foundation

erste Version 2006; 2011 die Version 1.0.0 freigegeben

Hadoop Map/Reduce Engine
besteht aus:

- Map/Reduce-Master: Job-Tracker
- Map/Reduce-Slave: Task-Tracker

Arbeits-Verfahren:

1. Input-Phase: Aufteilen der eingegangenen Daten in das HDFS
2. Map-Phase: Ausführen des Map-Teil's
3. Soft & Shuffle Phase: Sortieren und Neuverteilen der vom Map-Teil gelieferten Daten-Paare
4. Reduce-Phase: Kombination zusammengehörender Daten-Paare und Ausführen des Reduce-Teil's
5. Output-Phase: Ausgeben der vom Reduce-Teil berechneten Daten in das HDFS

8.x.y.z. OLTP

Online Transaction Processing
Management von einzelnen Transaktionen (speichern, abzuschließen, prüfen der Vollständigkeit (Ab- und Aufbuchungen), …)
Verarbeiten von sehr vielen Transaktionen innerhalb kurzer Zeiträume – praktisch sofort
Aktualisieren von Lagerbeständen, Aktivieren von Auslieferungen, Umbuchungen, …
viele Anfragen pro Zeiteinheit mit wenigen einzelnen abzuarbeitenden Schritten

verteilte Transaktionen

durchgeführte Operationen, z.B.:

- Senden eines Post's
- Hochladen eines Bildes
- Überweisen eines Geld-Betrages
- Verkauf / Bestellungen eines / des letzten Artikel's
- Mitbieten bei ebay

an einem verteilten System, soll weltweit sichtbar sein
auf viele Daten soll mit rel. kleinen Transaktionen zugegriffen werden

Absicherung der Vielzahl von Operationen, die durch die Transaktion ausgelöst wurden, als Ganzes und vollständig, nur dann ist die Transaktion abgeschlossen, ev. sonst Fehlermeldung oder Abbruch des Vorgangs (z.B. Bestellung)

beim Anmelden eines japanischen Nutzers soll auch eher ein japanischer Server genutzt werden
manche Informationen können verzögert verteilt werden, z.B. japanische Facebook-Einträge müssen nicht sofort auf jedem Server weltweit vorhanden sein

Ziel ist auch Erhöhung der Datensicherheit durch Replikation

ACID und BASE

Säure und Base
Konzepte, die vollständige Transaktionen beschreiben

ACID (Atomicity, Consistency, Isolation, Durability)
deutsch: Atomitizität, Consitenz, Isolation und Dauerhaftigkeit

wichtige Kriterien von Datenbank-Management-Systemen

- **Atomicity** Atomizität
  eine Transaktion kann nur als Ganzes ausgeführt werden
  eine Teilung ist nicht zulässig
  z.B. nur mit Ab- und Aufbuchung ist eine Finanz-Transaktion vollständig (nur ein Teil geht nicht!)

- **Consistency** Consistenz
  Passung aller Daten zueinander
  z.B. Ausschluß von zwei Nutzern mit der gleichen Kennung, eMail-Adressen, …; auch bei nachträglichen Änderungen und Fehlerhaften Korrekturen
  Aktualisierung von Lager-Beständen bei Verkäufen oder Nachlieferungen (Wareneingängen)

- **Isolation** Isolation
  jede Transaktion ist unabhängig von den Anderen
  z.B. bei der Reservierung eines Sitzplatzes im Flugzeug ist der Platz solange für andere Buchungen gesperrt, bis der reservierende Nutzer sich entschieden hat

- **Durability** Dauerhaftigkeit
  abgesicherte und ev. replizierte Speicherung der Daten von Transaktionen

in verteilten Systemen sind alle dieser Kriterien kaum vollständig einzuhalten
z.B. Verkauf des letzten Buches auf einem Server in Japan und einem in Europa zur gleichen Weltzeit

da die ACID-Eigenschaften praktisch nicht 100%ig zu schaffen sind, wurde als alternatives Konzept BASE entwickelt
basiert auf NoSQL-Datenbank-System
Base **A**vailable, **S**oft-state, **E**ventually consistent
garantiert nicht mehr die harten Kriterien von ACID
skaliert deutlich besser; kann sehr viele Transaktionen bearbeiten

wichtige Kriterien von Datenbank-Management-Systemen

- **Base Availabe** Verfügbarkeit besonders wichtig
  keine Transaktion soll verzögert werden, weil andere noch in der Bearbeitung sind

- **Soft-state** keine festen Zustände
  leichte Inkonsistenz wird tolleriert

- **Eventually consistent** an der Herstellung der Konsistens wird laufend gearbeitet

das CAP-Theorem

verteilte System werden von drei Eigenschaften bestimmt:
- Konsistenz (Consistency)
- Verfügbarkeit (Availability)
- Partitions-Tolerenz (Partition tolerance)

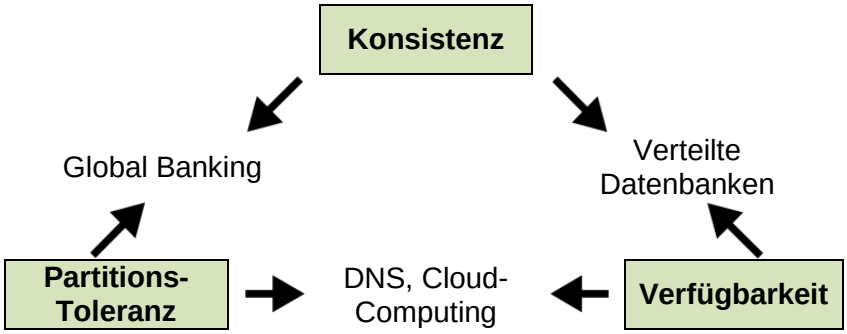
von diesen können aber immer nur maximal 2 auch erfüllt werden

Konsistenz z.B. bei Geld-Geschäften besonders wichtig
daraus folgt, dass nun entschieden werden muss:

- Soll das System immer verfügbar sein?

oder

- Soll die Partitions-Toleranz maximiert werden?



Daten-Partitionierung

Daten-Replikation

Frage, wie oft Daten gespeichert werden sollen?
Verfahren der Replikation

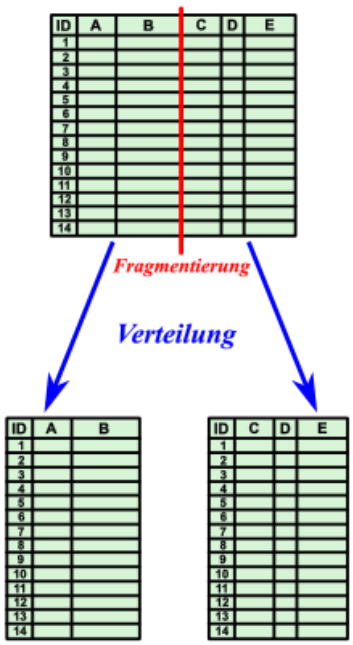
1. Fragmentierung (In welche Stücke sollen die Daten zerlegt werden?)
2. Allozierung (Wohin soll gespeichert werden? Wie oft sollen die Daten gespeichert werden?

Ziele

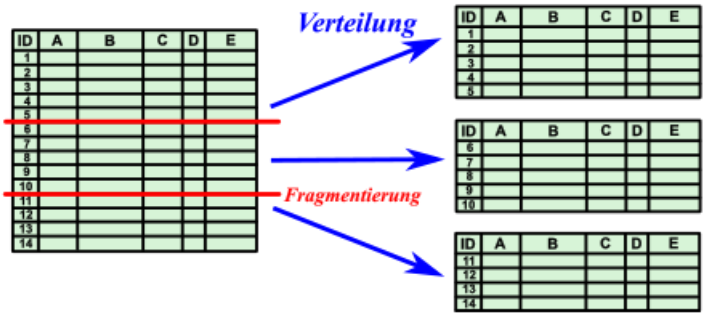
- gleichmäßige Verteilung
- praktische Verarbeitbarkeit (Daten für ein Problem möglichst auf einem Slave)
- Senkung der Daten-Transport-Kosten
- Lasten-Ausgleich (auch für nachgelagerte OLAP-Operationen)
- optimieren der Verfügbarkeit
- anpassen der Granularität der Daten nach Verwendung

Daten Allozierung

vertikale Partitionierung
Teilung der Datensätze / Transaktionen / … nach ihren Attributen
(z.B. Kunden-Daten einer Bestellung kommen auf den einen Slave, die Waren-Daten auf einen anderen
praktisch Tabellen durch vertikale / senkrechte zerlegt



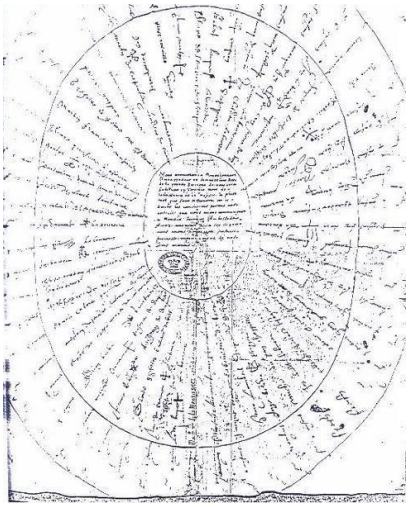
horizontale Partitionierung
Teilung der Daten in Gruppen von Datensätzen / Transaktionen / …
(z.B. die ersten 100 Bestellungen kommen auf den 1. Slave, die nächsten auf den 2. usw. usf.
praktisch Tabellen durch horizontale / waagerechte Linien zerlegt



Round-Robin-Verfahren

auch Rundlauf-Verfahren
Datensätze / Tarnsaktionen / … werden immer reihum auf die Slave's verteilt
z.B. einzeln, als Gruppe oder auch als pationierte Daten, weil Datensatz sehr groß ist

ein runder Robin war ein Beschwerde-Brief, der von seinen Unterstützern rundherum unterschrieben wurde
(alle sind somit gleichrangig beim Unterzeichnen, ein Hauptverantwortlicher (Rädelsführer) ist nicht zu identifizieren)
im 17.Jahrhundert in Frankreich verbreitet



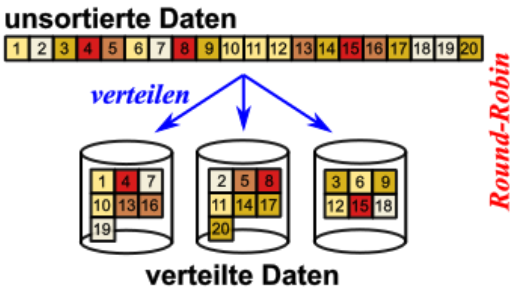
Runder Robin des
Jessé DE FOREST von 1621
Q: de.wikipedia.org (Paul Iee6977)

Vorteile:

- einfache Implementierung
- guter Lasten-Ausgleich
- gleichmäßige Verteilung der Daten

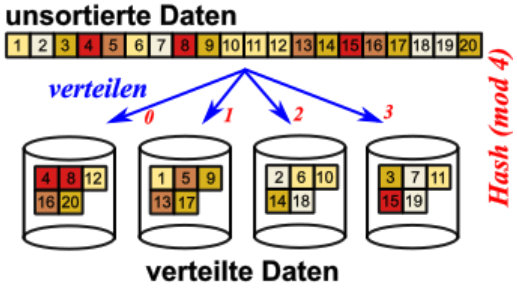
Nachteile:

- schlechte Performane bei Bereichs-Anfragen (Daten müssen dann erst zusammengesucht werden)



Hash-Verfahren

über den Datensatz wird ein Hash berechnet und nach dem berechneten Wert wird dann die Verteilung auf die Slave's / Bucket's (Eimer / Speicher für die gehashten Daten) vorgenommen
relativ einfache Hash-Funktion ist die Modulo-Berechnung
als Operator wird dann die Anzahl der Slave's benutzt



Vorteile:

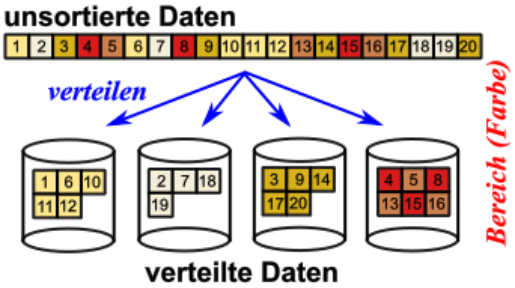
- normalerweise gleichmäßige Verteilung, weil meist zufällige Werte
- Daten mit gleichen Schlüsseln (hier gemeint die Ausgangs-Werte für die Berechnung der Hash-Funktion) landen auf dem gleichen Slave

Nachteile:

- zuzätzlicher Rechen-Aufwand für die Hash-Funktion
- ungleichmäßige Verteilung bei schlechter Hash-Funktion
- bei einem Datenverlust sind wieder viele Berechnungen oder Umsortierungen notwendig
- Bereichs-Anfragen mit erhöhtem Such-Aufwand verbunden

Bereichs-Partitionierung

die Verteilung erfolgt basierend auf bestimmten Attributs-Werte-Bereichen
das können Geschlecht, Körpergröße, Material-Eigenschaften, … sein
in der nebenstehenden Abbildung wurden Farben zu Unterscheidung herangezogen



Vorteile:

- optimal für Bereichs-Analyse bezüglich des Partitionierungs-Kriterium's
- Daten mit ähnlichen Schlüsseln (hier die Farbe) landen auf dem gleichen Slave

Nachteile:

- ungünstig für Bereiche anderer Attribute (, die nicht als Partitionierungs-Kriterium dienten)
- ungleichmäßige Verteilung bei schlechter Wahl des Kriterien
- ev. ungleiche Last-Verteilung
- Daten müssen vorbewertet werden

Consensus-Protokolle

wenn mehrere – relativ unabhängige – Systeme arbeiten, dann können Unstimmigkeiten entstehen
dann muss mittels eines sicheren Verfahrens ein Konsenz zwischen den Systemen hergestellt werden
einfache Beispiele: Zeit-Abgleich, Reihenfolgen

Consensus-Verfahren

- **Two Phase Commit** (z.B.: Hochzeit)
  wenn beide Bteilige sich sicher sind, dann kann eine (gemeinsame) Erklärung verbreitet werden
  wenn Bezahl-Bestätigung (Ab- und Aufbuchung über exter-

nen Bezahl-Dienst) kommt, dann kann die Bestellung weiter ausgeführt werden

- **Tree Phase Commit** zuerst wird Bereitschaft (für das Verfahren) geprüft dann werden die Bedingungen abgeglichenund zuletzt das Ergebnis festgelegt (und synchon umgesetzt)
- **Paxos** mit fünf verschiedenen Rollen (Aufgaben) im Verfahren Client, Acceptor, Proposer, Learner, Leader
- **Blockchain** Einigung über Veränderungen oder Besitztümer
- **Proof of Work**
- **Proof of Stake**
- **Proof of Activity** Mischung aus Proof of Work und Proof of Stake
- **Proof of Elapsed Time**
- **Proof of Burn**
- **Proof of Capacity**
- **Raft**
-

8.x.y.z. Cloud-Computing

Scale up
Hoch-Skalieren
Anschaffung größerer und Leistungs-fähigerer Hardware (Rechner, Speicher, …)
hohe Anschaffungs-Kosten
ev. häufige Wechsel der Hardware notwendig (relativ unflexibel)
umfangreiche Veränderungen der Software oft ebenfalls notwendig (vieles Spezial-Systeme)

Scale out
Hinaus-Skalieren
Anschaffung weiterer (gleichartiger) Hardware (Rechner, Speicher, …)
meist preiswerter, deutlich flexibler
Anschaffung genau nach bedarf notwendig, Hardware muss nicht ständig erneuert werden
Software leicht anpassbar oder Standard-Systeme

Scale in
Hinein-Skalieren
effektiveres Nutzen der vorhandenen Hardware z.B. durch Virtualisierungen (Simulation mehrerer Rechner auf einer Hardware (meist schon so Lasten-Ausgleich möglich)
Hardware (Speicher, Prozessor) wird optimaler ausgenutzt

z.B. auch gemeinsame Nutzung eines (angeschaften) Datenbank-Management-Systems (nur 1x Kosten) für mehrere Kunden (die haben eigene Tabellen, …, ohne Kontakt zu den Daten der anderen Nutzer)

Techniken lassen sich praktisch einzeln und mehrfach kombinieren

Abstraktionen

Legacy-Systeme
geringe Flexibiltät, geringe Virtualisierung (Art der Abstraktion)
Rechenzentren mit spezieller Hard- und Software
Hardware-orientiert

Virtualisierung
mittlere Flexibiltät, mittlere Virtualisierung
Client-/Server-Virtualisierung, für Desktop-Systeme
mehrere / viele Anwendungen auf einem System (Multi-Tasking)
Software-orientiert

Cloud-Virtualisierung
hohe Flexibiltät, hohe Virtualisierung
für Web-Anwendungen
Service- / Dienst-orientiert

Eigenschaften von Cloud-Systemen:

- on-demand access (Zugriff nur auf Bedarf)
- Zugriff unabghängig vom Standort / globaler Zugriff möglich
- Anbieter können Ressourcen sehr flexibel und breit gefächert nutzen
- hohe Elastizität (bei steigenden Anforderungen lässt sich schnell mehr von dem Dienst buchen / Bedarfs-Anpassung)
- wirtschaftlich (meist nur das bezahlen, was man wirklich nutzt)
    - bessere Auslastungen
    - spezialisiertes und professionelleres Personal kann eingestellt werden
    - Strom-Kosten (Verluste bei Strom-Transport teurer, als Verlust-Aufgleich bei Daten)
    - …
- …

off promise cloud
ein separater Cloud-Anbieter mit vielen verschiedenen Nutzern

on promise cloud
Cloud-Struktur innerhalb eines großen Unternehmens
Anschaffung eines Rechenzentrums / Servers für viele / alle Abteilungen

fork-computing
erweitert die Cloud-Systeme um die Leistungsfähigkeiten der Client's zu einer Gesamt-Leistung

Everythink as a Service

Infrastructure as a Service (IaaS)
Cloud-Anbieter stellt nur die Infrastruktur (Hardware, Leitungen, …) zur Verfügung
Nutzer kann die Cloud völlig frei nutzen, muss sie selbst administrieren, pflegen, …
eigene Betriebssysteme, Anwendungen, … möglich

Platform as a Service (PaaS)
Cloud-Anbieter stellt Hardware z.B. mit passendem Betriebssystem bereit
meist wird ein vorinstalliertes, vorkonfiguriertes System angeboten, mit dem der Nutzer wieder machen kann, was er will
eigene Anwendungen möglich

z.B. Server-Hosting bei Strato, …

Software as a Service (SaaS)
völlig vorgefertigtes System mit Anwendungen
Anwendung steht im Vordergrund
Nutzer merkt nichts mehr vom Betriebssystem (meist sehr kosten-günstige ausgewählt (z.B. Linux))
Nutzer verwendet – meist über den Browser – Standard-Software (Textverarbeitung, Tabellenkalkulation, Präsentations-Software, eMailing, Kalender, Aufgaben-Planung, Kontakte, …)

z.B.: google-office, web-Mailing, Prezi, Kahoot, …

8.x.y. Daten-Aufbereitung

8.x.y.z. Informations-Qualität

einfachste Definition für Qualität ist "die Eignung für den Gebrauch"

> Auch wenn wir Qualität nicht definieren können, wissen Sie schon, was gemeint ist.
> Robert PIRSIG
> (Even though quality cannot be defined, you know what it is.)

Komplizierteste Begriffs-Bestimmung listet 179 Dimensionen für den Begriff auf.
Eine recht praktische Merkmals-Liste zeigt die folgenden (15) Dimensionen / Kriterien für Informations-Qualität auf:

- Exaktheit (Accuracy)
- Objektivität (Objectivity)
- (Believability)
- Reputation (Reputation)
- Konsistenz (Consistency)
- Sicherheit (Security)
- Relevanz (Relevance)
- Aktualität (Timeliness)
- Wertschöpfung (Value-Added)
- Vollständigkeit (Completeness)
- (angemessener) Umfang (Amount of Data)
- eindeutige Auslegbarkeit (Interpretability)
- Verständlichkeit (Understandability)
- Zugänglichkeit (Accessibility)
- Übersichtlichkeit (Concise Representation)

Definition(en):

8.x.y.z. Dimensionen der Daten-Qualität

Kategorien der Informations-Qualität und ihre Dimensionen

- **inhärente (Dimensionen)** ➔ **Inhalt**	- (hohes) Ansehen - Fehlerfreiheit - Objektivität - Glaubwürdigkeit
- **system-unterstützende (D.)** ➔ **System**	- Zugänglichkeit - Bearbeitbarkeit
- **darstellungs-bezogene (D.)** ➔ **Darstellung**	- Verständlichkeit - Übersichtlichkeit - einheitliche Darstellung - eindeutige Auslegbarkeit
- **zweck-abhängige (D.)** ➔ **Nutzung**	- Aktualität - Wertschöpfung - Vollständigkeit - angemessener Umgang - Relevanz

nach: Deutsche Gesellschaft für Informationsqualität (dgiq)

Vollständigkeit und Fehlerfreiheit sind die für informatische Zwecke besonders wichtigen Dimensionen der Daten-Qualität

8.x.y.z.1. System-Unterstützung

Zugänglichkiet (Accessibility)

Hier steht die Frage im Raum, ob man auf einfache / zumutbare Art und Weise an die Daten herkommen kann.
Probleme können z.B. durch Schutz-Interessen von Firmen oder Daten-Umstrukturierungen (neues Warenwirtschafts-System) auftreten

Bearbeitbarkeit (Ease of manipulation)

Daten müssen in Formaten zur verfügung stehen, die dem Inhalt entsprechen und den Zweck der Nutzung unterstützen
Negativ-Beispiele: Email-Adressen als JEPG-Datei, Datum's-Formate

8.x.y.z.2. Inhärität

8.x.y.z.3. Darstellungs-Bezug

Kann ein Mensch die Daten anschauen und mit ihnen (praktikabel) weiter arbeiten?

Verständlichkeit (Understandabilty)

Daten in passenden Datentypen (Wohnort als Name statt als GPS-Daten)
keine lesbaren Produkt-Namen, stattdessen nur Nummern oder (kryptische) Kürzel
hilft beim Identitifizieren von Fehler-Quellen

Übersichtlichkeit (Concise representation)

möglichst kurze / knappe Darstellung der Daten
zu viele sowie unnötige Daten verschlechtern die Übersichtlichkeit

einheitliche Darstellung (Consistent representation)

besonders bei der Zusammenführung von Daten ein Problem, da kommen verschiedenste Zahlen- Währungs- und Datum-/Zeit-Formate daher
z.B. werden Geschlechter völlig unterschiedliche erfasst, gespeichert und dargestellt (w, m, d; weiblich: ja oder nein; Anreden; …)
unterschiedliche Einheiten-Systeme (SI oder cgs) sowie unterschiedliche Einheiten (°C und K) oder auch Zahlen-Dimensionen für die Einheiten (mg oder g) sorgen hier für unbedingt zu beachtende und wahrscheinlich zu korrigierende schlechte Daten-Qualität

eindeutige Auslegbarkeit (Interpretability)

der hinter den gespeicherten Daten (Zahlen, …) steckende Inhalt mus eindeutig sein
21 kann alles sein: Temperatur, Anzahl, Uhrzeit oder Laufzeit (z.B. Stunde oder eben Stunden)

z.B. kann in einer Tabelle "Kunden" völlig verschiedene Arten von Kunden erfasst werden, die sind u.U. nicht gleichwertig für eine nachfolgende Bearbeitung / Auswertung
News-Letter-Leser sind anders, als Kauf-Kunden oder Support-Kunden oder Produkt-Interessierte zu verarbeiten und zu verstehen

z.B. auch andere Begrifflichkeiten sehr unterschiedlich auslegbar (vor allem bei internationalen Daten)
z.B. werktags, …
hier hilft nur eindeutige Begriffs-Bestimmung oder die Festlegung von Grenzen

8.x.y.z.4. Zweck-Abhängigkeit

die Qualität der Daten wird erst über die Nutzung / Nutzbarkeit bestimmt

Aktualität (Timeliness)

stehen die Daten rechtzeitig und schnell genug für die weitere Bearbeitung zur Verfügung
z.B. Aktien- oder Wechsel-Kurse

Wertschöpfung (Value-added)

kann man aus den Daten und ihrer Auswertung einen (neuen / zusätzlichen) Wert schöpfen

Vollständigkeit (Completeness)

sind die Daten schon vorbereitet / gefiltert worden
sind Daten von bestimmten Kategorien schon entfernt worden

angemessener Umfang (Appropriate amount of data)

liegen überhaupt genügend Daten vor (z.B. für komplexere Statistiken) oder sind Datenmengen zu groß (für ein bestimmtes Verfahren / einen bestimmten Algorithmus)
handelt es sich um eine ausgewogene Teilmenge z.B. für Verfahren des maschinellen Lernen's
es können aber auch relevante Daten für eine (bestimmte) Analyse fehlen

Relevanz (Relevancy)

werden die Daten für das Verfahren wirklich gebraucht
müssen die Daten gespeichert / verarbeitet werden
sind die Daten für den Zweck notwendig
unpassende Daten-Genauigkeit (z.B. zu viele oder zu wenige Nachkomma-Stellen bei Messwerten)

8.x.y.z. Auswirkungen schlechter Qualität

Die Qualität einer Analyse kann maximal so gut sein, wie die Qualität der eingenenden Daten.

> Wenn man Müll hinein tut,
> dann bekommt man auch nur Müll hinaus.
> (Garbage in, garbage out.)

fehlerhafte Preise im Einzelhandel (80 % der Barcode-Fehler gehen zu Lasten der Verbraucher)
nicht richtig eingearbeitet Angebots-Preise, …

50 – 80 % der Einträge im US-Strafregistern falsch, ungenau oder fehlerhaft
nicht eingehaltene Lösch-Termine

rund 7 % der Massenpost-Sendungen nicht zustellbar, weil die Adressdaten nicht exakt waren

anekdotisches Beispiel: in Irland hat der Autofahrer Prawo Jazdy Unmengen von Ticket's bekommen, die konnten aber niemanden zugeordnet werden
heraus kam, dass hier von der Polizei nicht der Name sonder die polnische Bezeichnung für Führerschein erfasst wurde

Rechnungen (z.B. von Telefon-Anbietern) oder Überweisungen (z.B. Arbeitsamt) in Millionen-Höhe

Folge können rechtlich, ökonomisch, … sein
zusätzlich schnell Image-Schaden wegen schlechter Presse

Beispiele für niedrige Daten-Qalität

8.x.y.z. Daten-Vandalismus

absichtliche Daten-Manipulation, um bestimmte Effekte zu erreichen
häufigste Form ist das Löschen von Daten

politisch motivierte Manipulation

8.x.y.z. Messen der Daten-Qualität

nur gegen die Realität zu prüfen
praktisch sehr aufwändig

mögliche Qualitäts-Kriterien

- beim Subjekt
  - Relevanz
  - Glaubwürdigkeit
  - Verständlichkeit
  - …
- im Prozess
  - Verfügbarkeit
  - Antwort-Zeiten
  - …
- beim (Daten-)Objekt
  - Verfügbarkeit
  - Vollständigkeit (bezogen auf Gesamtheit)
  - fehlende Attribute / Daten-Dichte
  - Aktualität
  - …

8.x.y.z. Daten-Aufbereitung

Data preparation
Daten-Vorbereitung, -Reinigung, -Aufbereitung, -Präparation
altes Problem
Vorrang hat die Umsetzung der Roh-Daten in eine Form, in der sie weiter verarbeitet werden können

typische Probleme:

- Erkennen des Zeilen-Endes
- Erkennen des Endes eines Attributes (Feld-Trenner) od.ä.
- unterschiedliche Zeichen-Sätze (fehlerhaft umcodierte Sonderzeichen, Umlaute, …)
- unterschiedliche Schreibweisen von Zeitangaben (besonders Datum)
- einzelne Fehler in Datensätzen von großen Daten-Beständen
- fehlende Atrribute (die aber für die weitere Verwendung notwendig sind)

viel händischer / Personal-Aufwand
teilweise automatisierbar
heutige Daten-Wissenschaftler (data scientist's) beschäftigen sich zu rund:

- 60 % mit dem Daten-Reinigen und –Strukturieren
- 19 % mit der Daten-Beschaffung
- 9 % Suche, Bearbeitung, Korrektur fehlender Daten(-Teile)
- 4 % Verbessern, Anwenden von Algorithmen
- 5 % anderes

fast mit gleichen Anteilen werden die unbeliebten Tätigkeiten beschrieben:

- 57 % Daten-Reinigen und –Strukturieren
- 21 % Daten-Beschaffung

- 10 % Herstellen von Trainings-Daten für das Maschinelle Lernen
- 4 % Verbessern, Anwenden von Algorithmen
- 3 % Suche, Bearbeitung, Korrektur fehlender Daten(-Teile)
- 5 % anderes

Präparation auf Datei-Ebene

alt ASCII (auf DOS-Ebene weitgehend standardisiert)

gute – alte – und weit verbreitete Standrds für Zeichen sind UTF-8 und UTF-16

modern Unicode
aktuelle Version 12.1 beinhaltet rund 138'000 Zeichen und unterstützt 150 Sprach-Systeme
auf Windows-Rechnern i.A. über Eingabe als 4-Ziffern-Code bei gedrückter ALT-Taste (Zeichen erscheint erst nach dem Freigeben der Alt-Taste)
in alten Windows.System und DOS reicht die Eingabe als 3-Ziffern-Code (erzeugt Zeichen aus der Code-Page 850)

ungünstige (aber sehr beliebte) Austausch-Formate / Daten-Layout's:
z.B. EXCEL: mehrere Tabellen auf einem Tabellen-Blatt

Präparation auf Werte-Ebene

falsch oder schlecht formatierte Daten
- z.B. pseudo-markierte Überschriften / Hinweis-Texte mit Feld-Trennern
- z.B. Überschriften

fehlerhafte Werte (leeres Feld oder NULL oder Null oder WAHR oder FALSCH oder …)

Erkennen von Feld-Trennern und gleich-artigen Normalzeichen
z.B. Komma's als Feld-Trenner und Kommata in Texten (, die eigentlich durch Anführungsstriche umschlossen sind)
Erkennung von Anführungs-Strichen innerhalb von Texten

gleiche Bedeutungen von verschiedenen Zeichen:
z.B. tiefe und hohe Anführungs-Zeichen,

unterschiedliche Bedeutung gleicher Zeichen:
z.B. Anführungs-Zeichen bei Betonungen, wörtlicher Rede oder Ironie

Daten-Reinigung

Daten-Reinigung mit externen Quellen

Nachschlagen von (Wohn-)Adressen in zentralen Datenbanken (z.B. bei der Deutschen Post)

ähnliches für Produkte / Bauelemente / … möglich

Geocoding
Umwandlung einer Adresse in Geo-Daten, meist verbunden mit einer einheitlichen / standardisierten Darstellung

Nutzung von Wissens-Datenbanken (Knowledge Database's)
"Welt-"Wissen ist in strukturierter Form vorhanden, Abgleich mit eigenen Daten möglich
auch Anreicherung der Daten oder Ergänzung (fehlender Einträge)

teilweise auch möglich:
wikipedia, wikidata

Lösung des Matching-Problem's
Fehlerhafte Schreibweisen, überzählige Leerzeichen, Umsetzung von Umlauten, …

Imputation

Ergänzung fiktiver Werte an Fehl-Positionen
ev. Benutzung des Mittelwertes, Median (in der Mitte liegender Wert) oder Modus (häufigster Wert), um Statistiken möglichst wenig zu beeinflussen

problematisch bei vielen fehlenden Daten

Daten-Reinigung mittels Abhängigkeiten

Prüfen der Einhaltung von bestimmten Daten-Eigenschaften
z.B. einmalige Vergabe eines Schlüssel's in einer Daten-Tabelle
ev. korrigierbar durch Vergabe eines neuen Schlüssel's für einen Datensatz (vorher ev. Duplikats-Prüfung notwendig)
Problem bei Verwendung in anderen Tabellen (als Fremd-Schlüssel)

funktionalle Abhängigkeit
interne Bedingungen für Daten
nur einmaliges Vorkommen von bestimmten Attributen (Rangfolge, Teilnehmer bei einem Wettbewerb, …)
Problem die Fehler in den anderen Attributen zu finden
ev. Gegen-Prüfen der Daten auf Plausibilität
Ziel bei unterschiedlichen Fehlern bezüglich bestimmter Daten durch passende Korrektur auf weniger Fehler zu kommen

Beispiel (aus HPI-Kurs):
Bedingung: jeder Sportler soll nu in einer Sport-Art geführt werden

Name	Sport-Art	Distanz	Zeit	Medallie
Michael Phelps	Schwimmen	100 m	49 s	Gold
Usain Bolt	Schwimmen	100 m	11 s	Siber
Usain Bolt	Sprint	200 m	19 s	Gold

für uns offensicht, dass Usain Bolt wohl nicht beim Schwimmen dabei war
für Daten-Verarbeitungs-System aber kein klarer Fakt
Usain Bolt könnte der Name von zwei Sportlern sein
es könnte Sprint oder Schwimmen falsch sein

Gegen-Prüfen mit weiteren Daten und Abhängigkeiten
Wenn Usain Bolt beim Schwimmen über die Distanz von 100 m und Siber gewonnen hat und dabei weniger Zeit gebraucht hat, als der Gold-Gewinner (Michael Phelps), dann ergibt sich ein weiteres Problem.
Wird dagegen das Schwimmen als Fehler erkannt, lösen sich die Probleme auf. Der richtige Eintrag muss ev. nachrecherchiert werden. Die Bedingung der Datenbank ist erfüllt und die Inkonsistenzen der Daten entfallen.

weitere Abhängigkeiten:
- Matching
- Inklusion

Duplikate

typisch:
verschieden geschriebene Adressen für eine Person und eigentlich nur einer Adresse
doppelte Rechnungen (bei verschiedenen Rechnungs-Stellern in einem Dienstleistungs-Unternehmen; doppelte Daten-Übertragungen)

Prüfen, ob zwei Datensätze usw. in einer Sach-Tabelle nur zu einem Realwelt-Objekt gehören und den gleichen Sachverhalt beschreiben
→ Bestimmung der Ähnlichkeit

z.B. Sprachen-Verlinkung bei Wikipedia
Ausgangs-Punkt z.B. "Piotr"

Problem der großen Daten-Mengen / große Komplexität
z.B. Finden von graphischen Duplikaten
praktisch ist Vergleich jedes Objektes mit allen anderen notwendig

Aufgaben:

1. *Gehen Sie zur Wikipedia-Seite "Poitr" und welchseln Sie dann systematisch die Sprachen (englisch, russisch, spanisch, französisch, italienisch) durch (ev. wieder zum Anfang zurückgehen)! Notieren Sie die Sprache und den angesprungenen Begriff! Stellen Sie den Zusammenhang als Graph dar!*
   Wechseln Sie ausgehend vom angesprungenen Begriff wieder weiter die gleichen Sprachen (jetzt auch deutsch) und erweitern Sie den Graphen!
2. *Welche Probleme ergeben sich bei solchen "Übersetzungen"?*
3. *Erstellen Sie einen Begriffs-Graphen für die Musik-Richtung "Easy Listening" oder einer "Joint Stock Company" in den verschiedensprachigen Wikipedia-Versionen!*
4. *Wählen Sie einen Begriff, für den Sie Begriff-Verschiedenheiten in unterschiedlichen Sprachen kennen und erstellen Sie einen Begriffs-Graphen, wie oben beschrieben!*
5.

Duplikat-Erkennung

Ähnlichkeit

Suche nach einem Wert, der die Ähnlichkeit beschreibt
Prüfen des Ähnlichkeits-Wert's an einem Schwellen-Wert

Edit-Distanz
minimale Anzahl von Editier-Operationen, um das eine Wort in das andere umzuwandeln

- Austausch eines Zeichens ist eine Operation
- Löschen eines Zeichens ist eine Operation
- Einfügen eines Zeichens ist eine Operation

Problem bei Namens-Änderungen nach Heirat

unterschiedlich bei verschieden häufigen Begriffen / Namen / …
bei seltenen Begriffen / Namen / … ist es eher wahrscheinlich, dass es sich um ein und das selbe Objekt (der Realwelt) handelt, auch wenn sich viele Attribute unterscheiden
bei häufigen Begriffen / Namen / … ist die Ähnlichkeit / Übereinstimmung nur dann gegegeben, wenn viele / alle anderen Attribute übereinstimmen

Ähnlichkeit beim Klang
z.B. für Namen (Meier, Meyer, Maier, Mayer, Mayr, …)

Anzahl gemeinsamer Worte in Sätzen / Texten
Wort-Überlappungen

Beziehungs-basierte Ähnlichkeit

benutzen anderer Datenbestände (Tabellen), um Ähnlichkeiten zu finden
Artikel-Tabelle enthält mehrere zu prüfende Datensätze
für jeden Artikel gibt es in einer anderen Datenbank / Tabelle eine Stück-Liste, dann kann auch über gleiche oder sehr ähnliche Stück-Listen eine Ähnlichkeit der betrachteten Artikel gefunden werden

LEVENSHTEIN-Distanz

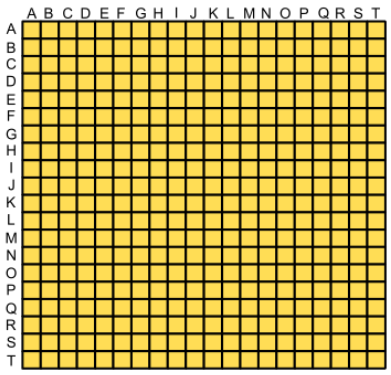
moderne Klassifikation über maschinelles Lernen
Training mit bekannten Duplikaten

Link:
https://phiresky.github.io/levenshtein-demo/ (online-Rechner)

Algorithmen

da die Daten-Menge sehr groß, muss bei den Algorithmen sehr effektiv gearbeitet werden
da jeder zusätzliche Arbeits-Schritt eben tausend- oder millionen-fach ausgeführt werden muss

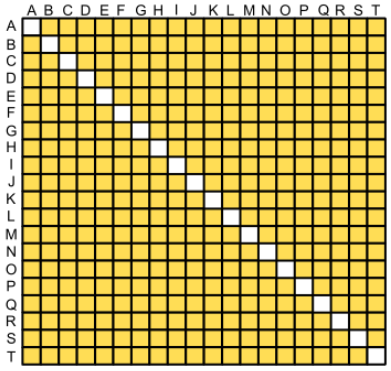
Vorauswahl von Kandidaten, dann genauere Prüfung der Kandidaten



Einsparung bei Bedachtung bestimmter Eigenschaften und Bedingungen

angenommen (nur) 20 Datensätze sollen auf Ähnlichkeit geprüft werden

für schlechten Algorithmus wären 20 x 20 = 400 Vergleiche notwendig (da hier Datensätze betrachtet werden, sind intern immer noch die Attribute einzeln zu vergleichen)



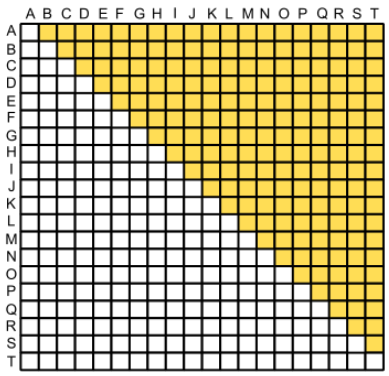
Daten-Satz sowie jedes Daten-Attribut muss nicht mit sich selbst getestet werden

damit spart man auf 20 Datensätzen schon mal 20 Vergleiche
macht 5 % Einsparung aus

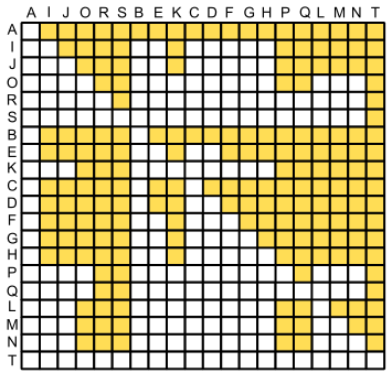
i.A. gilt die Symmetrie der Ähnlichkeit (wenn A ähnlich zu B ist, dann ist (sehr sehr wahrscheinlich) auch B ähnlich zu A

diese Annahme erspart uns die Hälfte der übriggebliebenen Vergleich

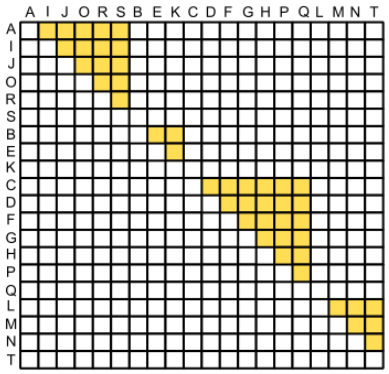
nur noch 380 / 2 = 190



Aufteilung der Datensätze in Gruppen (Partitionierung) → Vergleiche nur innerhalb der Partitionen (Gruppen)
Kriterien der Partitionierung müssen so gewählt werden, dass passend große Gruppen gebildet werden, aber Duplikate nicht übersehen werden können
z.B. Aufteilung eines Landes nach Postleitzahlen (erste oder die ersten beiden Ziffern)



innerhalb der Partitionen (hier durch den Alphabet-Bruch erkennbar) werden die gleichen Verbesserungen der Algorithmen, wie oben vorgenommen, so dass nur noch ein kleiner Teil von Vergleichen übrig bleibt
hier (wegen der Gruppierung) nur noch 15 + 3 + 21 + 6 = 45 Vergleiche



im Beispiel also Reduktion auf fast ein Zentel der ursprünglich notwendigen Vergleiche

→ Findet aber nicht Probleme bei der PLZ
dann nutzt man zwei oder drei weitere Partitionierungen nach anderen Kriterien
tauchen dann mögliche Duplikate in mehreren Ergebnis-Listen auf, dann sind die Duplikate ziemlich sicher und man findet auch weniger wahrscheinliche, ganz spezielle Duplikate

Daten-Fusion

Problem nach dem Finden der Duplikate ist das Löschen
Welcher Datensatz soll erhalten bleiben? Welche können – ohne Daten-Verlust (!) – gelöscht werden?

Möglichkeit ist die Kombination von Duplikaten zu einem reichhaltigeren Datensatz

Kriterien können sein:

- gleicher Attribut-Wert (können gefahrlos übernommen werden)
- größere Länge (meist ist das längere Attribut das mit Information)
- leeres Attribut kann durch ein gefülltes ersetzt / ergänzt werden (Zusammentragen von Ingformationen)
- Nutzung von Minimum, Maximum, Durchschnitt, …
- Anhängen nicht zuordbarer Attribute / Inhalte
- Aktualität (neueres Datum)
- Vertrauenswürdigkeit
- …

ev. nehält man alle Datensätze / Duplikate

macht einen (den informtivsten) zum Haupt- oder Kopf-Datensatz und die (anderen) Duplikate werden als Neben-Datensätze zugeordnet
damit werden z.B. Bestell-Historien erhalten

8.x.y. Informations-Integration

gemeint sind die Probleme, die sich unterschiedliche Strukturen der Daten ergeben

Welche Schwierigkeiten gibt es mit der Informations-Intergration?

Datenbank sollten eigentlich die Lösung des Problem's der vielen Daten in speziellen, dezentralen, individuellen, proprietären Dateien sein
viele Daten-Formate
Ziel war zentrale Datensammlung für alle (Abteilungen / Nutzer / …) mit hoher Aktualität und geringer Redundanz
neue Daten-Typen (Graphiken, Geodaten, …)

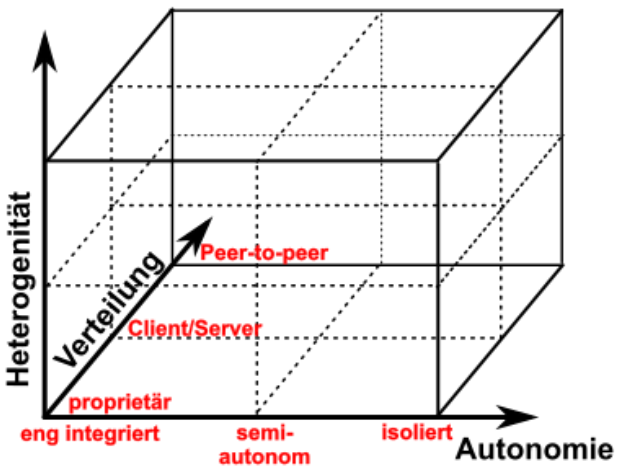
Problem-Felder

- **System-Probleme** — technische Aspekte
  Datei-Formate; Betriebssysteme mit unterschiedlichen Datei-Strukturen
- **soziale Probleme** — inhärenter Widerwille von Mitarbeitern / Abteilungen / …
  ihre Daten für eine breite Verwendung (an andere Mitarbeiter / in anderen Abteilungen / …) bereitzustellen
- **logische Probleme** — unterschiedliche Daten-Modelle
  unterschiedliche Modellierung von Daten
  verschiedene Definitionen von Dingen / Prozessen / …

8.x.y.z. Autonomie und Heterogenität von Daten-Quellen

Autonomie ist hier bezüglich der Eigenständigkeit der verschiedenen Systeme gemeint
sie werden eigenverantwortlich gepflegt, aktualisiert und ev. auch abgeschaltet
Daten-Strukturen können sich ändern, ev. kommen Erweiterungen dazu

daraus folgt Heterogenität der Implementierung, Design der Daten, Daten-Qualität, Zugriffs-Rechten, …



enge Integration
hier wird die Autonomie aufgegeben, um eine breite Integration in verschiedene eigene und fremde Systeme und Anwendungen zu ermöglichen
der Daten-Eigner stellt eine offene Schnittstelle zu seinen Daten bereit, gibt ev. Informationen zum Daten-Modell bekannt, modelliert seine Daten ev. auch um
Ergebnis ist i.A. eine Datenbank mit einem höhreren Mehrwert

semiautonom
der Eigner erlaubt den Zugriff auf seine Daten
behält sich aber Änderungen an seinem Modell, den Daten-Strukturen usw. usf. vor

isoliert
Daten-Quelle hat keine Kenntnis darüber, dass sie Teil einer Integration ist
z.B. irgendwelche Web-Daten, sie stehen einfac so zur Verfügung und werden – ev. auch ohne Kenntnis des Eigners – in andere Projekte eingebunden

Client-Server-Verteilung
Daten stehen auf einem System (Server) zur Verfügung und können von beliebig vielen anderen System (Client's) genutzt / heruntergeladen werden

Peer-to-peer
hier werden die Daten gleichmäßig im Netz auf verschiedenen – gleichberechtigten – System gehalten, die sich in irgendeiner Form gegenseitig aktualisieren

Arten der Heterogenität

- **syntaktisch** betrifft Formate, Codierungen, …
- **strukturell** betroffen siend hier die Schemata, Modelle, …
- **semantisch** Bedeutung / Interpretation von Daten

syntaktische Heterogenität
Hardware-Heterogenität: technische Aspekte (verbaute Technik, Übertragungungs- und Übersetzungs-Komponenten)
Software-Heterogenität: unterschiedliche Betriebssysteme mit z.B. unterschiedlichem Handling von Dateien, Ordnern usw.
verschiedene Protokolle und deren Umsetzung
Software-Versionen
Schnittstellen-Heterogenität: Zugriff auf verschiedene Felder; Implementierungen von Logiken
Logik-Verständnis von Formularen
gebundene Felder (Felder die unbedingt ausgefüllt werden müssen)

strukturelle Heterogenität
unterschiedliche Darstellung eigentlich gleicher Daten:
z.B.: Personen und Geschlechts-Charakterisierung
Person(ID, Nachname, Vorname, männlich, weiblich)
Person(ID, Nachname, Vorname, Geschlecht)
Männer(ID, Nachname, Vorname) ; Frauen(ID, Nachname, Vorname)

Beispiel (einer Analyse des Objektes "Company" in Info-Boxen bei wikipedia)
es wurden von Nutzern 1083 unterschiedliche Attribute zugeordnet, davon kamen 499 nur einmalig vor
in 39 Atrribut-Bezeichnungen kam der Teilstring "name" vor
für 273 Unternehmen war aber nicht einmal etwas bei enem "name"n eingetragen

→ Probleme!
Welches Attribut kennzeichnet nun den "echten" Firmen-Namen?
Was passiert, wenn in vielen Feldern etwas steht? Welches Feld hat Vorrang?
wass passiert, wenn gar kein "name"-Feld ausgefüllt ist?
…
Bei welcher Verwendungs-Zahl beschneidet man die Anzahl von Attributen?
Was passiert mit neuen Attributen, die zuerst ev. nur kleine Vertreterzahlen haben?
…

in Datenbanken gibt es einen ausgeprägten Hang zum Chaos bei Schemata

Schema-Mißbrauch über Homonyme (gleiche Begriffe mit / für unterschiedliche Bedeutungen)

semantische Heterogenität
mit den größten Problemen verbunden, da Computer hier die wenigste Unterstützung gewähren können
Computer können zwar Begriffe erkennen, aber deren Bedeutung kaum
vielfach Bedeutung nur aus dem Sachzusammenhang erschließbar
oft erweiterte Interpretation notwendig

Beispiel:
Andrea und Gina stehen lange mit ihrem Auto am Fluß.
Am nächsten Tag schaut er allein von der anderen Seite zurück.

8.x.y.z. Schema Matching

ist die Verknüpfung unterschiedlicher Daten-Schemata /-Modelle
Probleme durch unterschiedliche Sprachen(first name, Vorname) oder Homonyme usw. usf.
Ergebnis ist i:A. eine Zurdnungs-Tabelle mit Attributs-Paaren (Korrespondenzen) aus beiden Schemata

die Tabellen / Attribute des einen Schema's werden Tabellen / Attributen in einem anderen Schema zugeordnet
ev. händisch
moderne Werkzeuge funktionieren graphisch orientiert, ev. mit Umcodierungen
Hochleistungs-Produkte versuchen das Matching automatisch → erstellen Vorschlag
benutzen gleiche Attributs-Bezeichnungen, übersetzte Attributs-Namen, Vergleiche über die Atrributs-Werte (semantische Vergleiche)

funktioniert besonders gut bei bekannten Duplikaten

eigentliche Daten-Übertragung erledigt das Schema Mapping

8.x.y.z. Schema Mapping

ist das Verfahren der Übertragung und Anpassung der Daten nach dem Matching Schema notwendig bei unterschiedlichen Formaten, Codierungen

8.x.y.z. Materialisierte Integration

das Ergebnis einer Daten-Integration – also eine Datenbank nach dem Schema-Mapping – wird bei der materialisierten Integration lokal gespeichert
die Arbeiten mit dem integrierten Daten passiert ausschließlich auf den zwischen-gespeicherten Daten

die zwischen-gespeicherte Datenbank wird als materialisiertes Integrations-Produkt verstanden (es ist zusätzlicher Speicher notwendig)

Probleme:

- da nicht kontinuierlich intehriert wird, können lokale Daten beraltert sein
- zusätzlicher Speicher-Aufwand
- zusätzlicher Bearbeitungs- / Transformations-Aufwand
- häufig bietet die Quelle nicht den gesamten Datenbestand (zum downloaden) an

8.x.y.z.1. Data Warehouses

entstanden aus der Herausforderung komplexer Analysen über die Daten
außerdem sollen laufende (online-)Daten-Verarbeitungen nicht durch die Analyse-Verfahren gestört werden

praktisch Daten-Lager mit großer Angebots-Fläche
ist ein materialisertes Integrations-Ergebnis
Daten sind i.A. bereinigte Daten in einer für viele Analysen passenden lockeren / flachen Struktur
ev. werden ETL-Prozesse angeschoben
es sind meis riesige Daten-Mengen vorhanden, da man aber im Voraus noch nicht weiss, welche Anfragen kommen werden, werden alle Daten verfügbar gehalten

Problem der Aktualität kann durch geschickte Aktualisierungs-Zeiten reduziert werden

technisch gesehen sind es relationale Datenbanken mit SQL-Zugriff
meist Integration mehrerer unabhängiger Daten-Quellen zu einem großen Daten-Bestand, die als Big data charakterisiert werden kann

nach der Daten-Aufbereitung werden die Daten mehr-dimensional strukturiert, um möglichst kurze Zugriffe auf sie zu haben (und nicht mehrfach geschachtelte Fremdschlüssel, wie in "normalen" relationalen Datenbanken üblich)
Veranschaulicht durch Würfel (Data cube, OLAP-Würfel, Daten-Würfel), praktisch aber viel mehr Dimensionen

Daten werden häufig in ein Starschema gebracht und gespeichert

im Zentrum des Sterns liegt eine Basis-Fakten-Tabelle und darum herum sind diverse Detail-Tabellen angeordnet

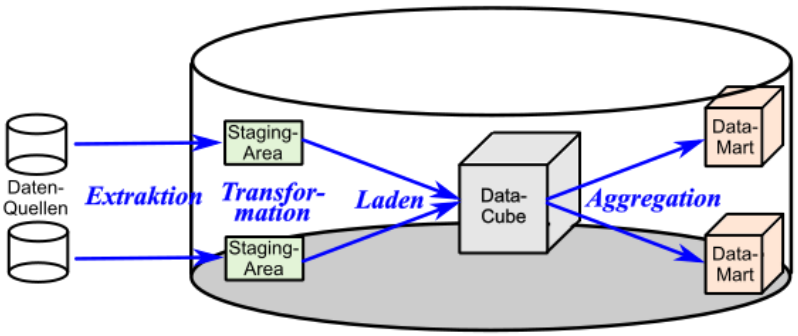
Zugriff auf den großen mehr-dimensionalen Datenbestand durch Slicing und Dising (Herauschneiden von Scheiben / Schichten / kleineren Würfeln aus dem Data Warehouse "Würfel")
Data Warehouse bietet ev. kleine Data cube's an, die einige spezielle Dimensionen (Atrribute) umfassen oder nach Nutzungs-Zweck zusammengestellt wurden
heißen Data mart (Daten-Markt)
in diesen sind kleinere, schnellere, effektivere Analysen möglich

weiter mit 5-10

8.x.y.z. ETL-Prozesse – Extract-Transform-Load

Prozess der Materialisierung der Daten während der Integration

Prozess der Umsetzung der externen Daten in den internen Daten-Bestand (Data-Cube).



Extraktion beschreibt alle Prozesse des Selektierens und Herunterladens der externen Daten von ausgewählten Daten-Quellen
hier muss mit verschiedenen Daten-Strukturen, unterschiedliche Schnittstellen und sehr großen Daten-Mengen gerechnet werden

die Transformation umfasst alle Vorgänge der Daten-Bereinigung und –Strukturierung der heruntergeladenen Daten-Bestände
u.U. auch Filtern, Überprüfen von Daten, …
dazu stellt die Staging Area viele verschiedene Tool's zur Verfügung

diese transformierten Daten werden dann im Lade-Prozess in die interne Datenbank (Ziel-Datenbank) eingespeist Dieser Daten-Bestand wird wegen seinen flachen Struktur aus vielen Dimensionen Daten-Kubus (Data-Cube) genannt

damit sind die eigentlichen ETL-Prozesse beendet und es kann sich noch Aggregations-Prozesse anschließen, die zu kleineren – besser handhabbaren – Daten-Beständen , den sogenannten Daten-Märkten (Data-Mart) führen
hierfür wird vorrangig auf Slicing und Dising ((in Scheiben) schneiden und Würfeln) zurückgegriffen

8.x.y.z. Business Intelligence - BI

Geschäfts-Intelligenz
für
im Unterschied zum Data Mining geht es beim BI mehr um die Nach-Verfolgung und Auswertung der Daten ("nackte" Analyse, vielfach sogar ohne Bewertungen)
Data Mining wird in diesem Zusammenhang eher mit dem Finden neuer Zusammenhänge usw. in verbindung gebracht
sachlich, technisch und informatisch sind BI und DM aber dicht verwandt

statistische Auswertung und Präsentation der Daten; Erstellen von Berichten
umfangreiche Klassifizierungen und Differenzierungen
Trends finden und darstellen
Personal-Analysen (z.B.: Wer verkauft wievel?)
Herstellen neuer Zusammenhänge
zunehmende Aktualität

8.x.y.z. Data Lake's / Data Reservoir's

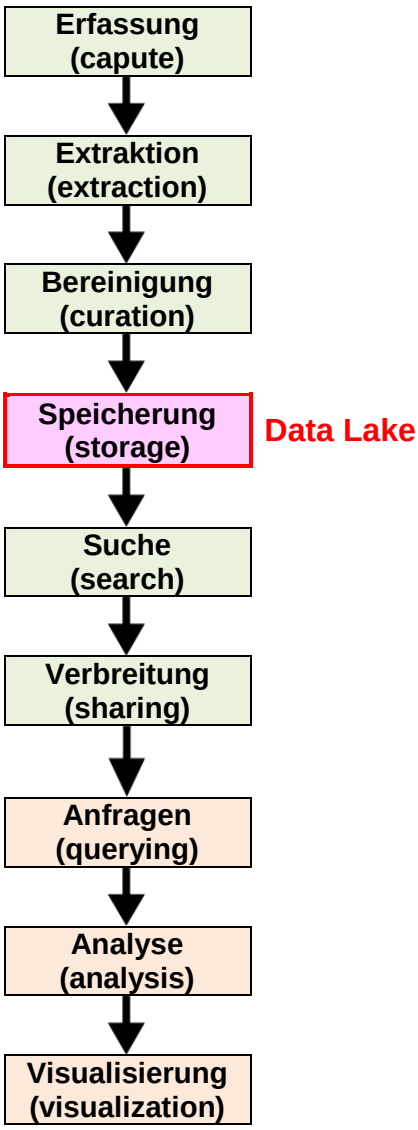
modernes Schlagwort, derzeit leicht gehypt
quasi eine Steigerung des Big Data
Fortsetzung der "Jäger- und Sammler-Mentalität" auf die Ebene der Digitalisierung
da (sehr) viele Daten sehr einfach zugänglich sind und auch billig gespeichert werden können, werden diese auch für weitere / spätere Verwendungen aufgehoben
(Man weiss ja nicht, wofür es später mal gut sein könnte!)

massenhafte Sammlung von unstrukturierten Daten
möglichst in Reinform, um wirklich wenig verfälschte daten zur Verfügung zu haben
die strukturiierten Daten-Bestände in den klassischen Big-Data-Datenbanken werden mit einbezogen

quasi Sammlung von Daten aus hoch-strukturierten Datenbanken, halb-strukturierenten Daten z.B. in JSON- oder XML-Dateien sowie Binär-Daten, wie Bilder, Video's, …

Beispiele microsoft Azure Data Lake, Amazon S3, Apache HADOOP, …

in der Data-Science-Pipeline
findet man Data Lakes



Data Lake mehr im Zusammenhang mit eigenen / internen Daten-Beständen verstanden
Data Hub meint eher Datensammlungen für die Veröffentlichung / den externen Gebrauch

wichtige Aspekte sind die Quellen-Nachweise und die Durchsuchbarkeit für die Daten-Bestände
meist hängen großen Meta-Datenbanken an den Rohdaten
erweiterte Verschlagwortung, nicht nur der Daten in den Tabellen sondern auch für die Tabellen selbst

Log-Daten beschreiben schon vollzogene Zugriffe und ev. auch Transformationen usw. usf.
bei schlechter Arbeit mit den Daten-Beständen und einem fehlendem Konzept können dann auch Data-Swamp's (Daten-Sümpfe) entstehen, mit denen man im schlimsten Fall gar nichts mehr anfangen kann

8.x.y.z.1. Daten-Herkunft

wo kamen die Daten (ursprünglich) her, warum stehen die Daten in dieser Form zur Verfügung
Data origin, Data provenance, Data …

wichtig auch für die Bewertung der Daten-Qualität, für nachlaufende Bewertungen der Daten und Analysen
Analyse der Daten-Pfade und durchlaufenen (vielleicht auch ungünstigen / ungeeigneten) Veränderungen

offensichtliche Passung reicht nicht, da oft bei Quellen-Prüfungen Widersprüche gefunden werden
gerne Schätzungen benutzt oder Quellen verwendet, die gleiche Zahlen (trotz eigentlich unterschiedlistzer Ausgangs-Bedingungen) liefern

WHERE-Provenance
Wo leigen meine (verwendeten) Daten?

WHY-Provenance
Warum entsteht dieses (Analyse-)Ergebnis? (Wie enststand das Ergebnis?)

WHY-NOT-Problematik
Warum sehe isch ein bestimmtes – von mir erwartetes – Ergebnis nicht?
Was fehlt hier? Was ist bei der Analyse schief gelaufen? …

8.x.y.z. virtuelle Integration

quasi Gegen-Stück zur Materialisierung der Daten während der Integration

Nachteile der Materialisierung:
- großer Speicher-Bedarf
- vollständige Aktualität nicht gegeben (Daten sind immer (etwas) veraltet)

Ziel der Virtualisierung:
- Daten sollen bei der Quelle bleiben
- es entstehen gefühlte / temporäre Daten-Bestände

- hohe Aktualität
- Daten-Quellen sollen / wollen Autonomie behalten

da die Daten nicht wirklich existieren, müssen Anfragen usw. wieder zurück an die originalen Quellen gegeben werden, erneut analysiert und dargestellt werden
Daten-Quellen können autonom bleiben

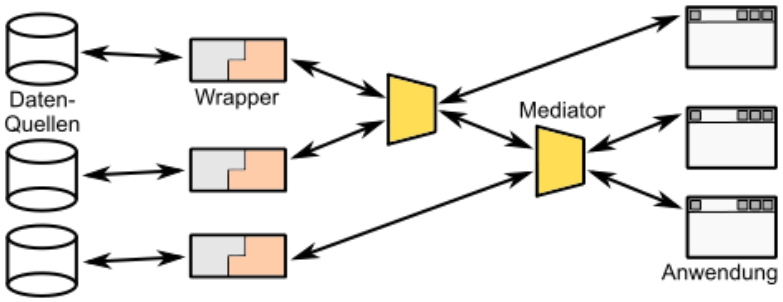
Nachteile der Virtualisierung:

- längere Verarbeitungs-Zeiten
- da Zeit kritische Diemension ist, können Daten nur wenig Transformiert, Bereinigt usw. usf. werden
- man verlässt sich sehr stark auf die Zuverlässigkeit der Daten-Quelle
- hohe Daten-Übertragungs-Mengen
- durch Autonomie der Quellen ist starke Abhängigkeit von deren Beständigkeit gegeben

8.x.y.z.1. Mediatoren / Wrapper

Wrapper sind Verpackungen / äußere Hüllen von Daten-Quellen
haben Schnittstellen-Funktion zwischen Daten-Quellen und Mediatoren
sammelt die Daten zusammen, bereit sie so auf, dass sie einem internen Daten-Modell entsprechen
entscheiden ev. auch über die Auswahl der Quellen

Mediatoren bereiten die Daten auf und stellen sie bedarfsgrecht den Anwendungen zur Verfügung
geben z.B. passende Anfragen an die Quellen weiter (z.B. gefilterte und sortierte Daten) und geben die gelieferten Ergebnisse an die Anwendungen weiter

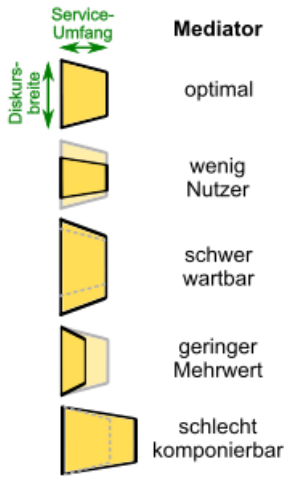


können die Quellen dagegen nicht sortieren oder gefilterte Daten bereitstellen, dann übernimmt diese Aufgabe der Mediator

die Anwendung erhält immer gleichartige Daten(-Strukturen und –Qualitäten)

Diskursbreite (domain scope) = Vielfalt der Daten, die eingelesen werden können

Service-Umfang (service scope) = Umfang der intergrierten Leistungen im Mediator



dünne und dicke Mediatoren
(nach WIEDERHOLD (1999))

8.x.y.z. das Deep Web

beinhaltet solche Daten-Quellen, die nicht ihren gesamten Datenbestand verfügbar machen, sondern nur Teile zugänglich / abrufbar sind
Daten nicht (direkter) Download abrufbar
Daten sind nur durch Such-Anfragen zugänglich
z.B. Katalog von Amazon ist nicht als solcher erreichbar (weil er auch so gar nicht existiert!), wohl aber Details nach einer entsprechenden Anfrage

auch Hidden Web, Invisible Web

meist besonders gut gepflegt
meist professionell erstellt
wenn man die bereitgestellten Daten gut (aus den Anfrage-Formularen) auslesen kann, dann hat i.A. sehr gute Daten

soll – wie bei einem Eisberg – den unter Wasser liegenden Teil – des Internet's ausmachen
oberster Teil wird dementsprechend Surface Web genannt
kurz unter der Wasser-Oberfläche befindet sich das Shallow Web, das im Wesentlichen aus den dynamischen Webseiten besteht, die individuell für den Nutzer zusammengestellt wurden und nicht wirklich irgendwo abgespeichert werden bzw. existieren

unterster Teil des Eisberg's ist dann das Dark Web

8.x.y. Data Mining, Statistik, Maschinelles Lernen

8.x.y.1. Statistik

> Traue keiner Statistik,
> die du nicht selbst gefälscht hast.
> Winston CHURCHILL

Obwohl dieses Zitat Winston CHURCHILL und manchmal auch Joseph GOEBBELS zugeordnet wird, ist es wohl nur eine Umformulierung einer Aussage in einem Zeitschriften-Artikel von Hanns-Erich HAACK: "… So viel haben sie schon gelernt, daß sie nur den Statistiken glauben, die sie selbst gefälscht haben."
Hier im Skript geht aber nicht darum Statistiken zu fäschen, sondern aufzuzeigen, wo die Fallstricke liegen und wie man Manipulationen erkennen kann.

explorative Statistik
erforschende Analyse der Daten z.B. in Visualisierungen (z.B. durch Hereinlegen einer Gerade od.ä.)

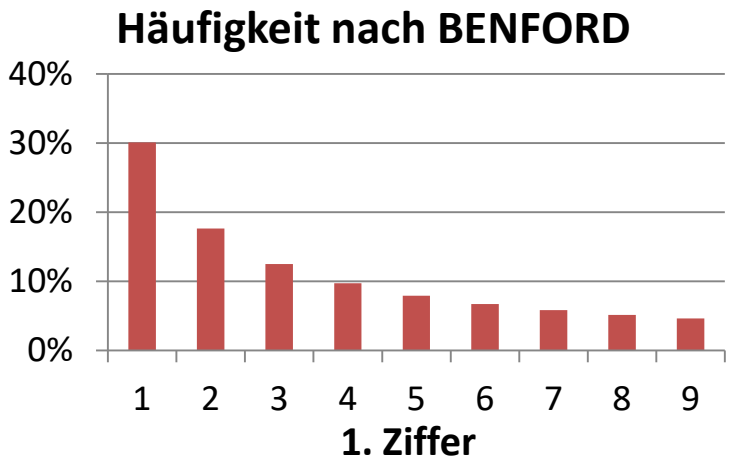
deskriptive Statistik
beschreiben die Daten (als Gesamtheit) z.B. über den Mittelwert
benennen von Kurven-Formen

induktive Statistik
Ableitung von Merkmalen der Gesamtheit aus einer Stichprobe
basiert stark auf Wahrscheinlichkeits-Rechnung

interessante Links:
unstatistik.de / http://www.rwi-essen.de/unstatistik/ (Beispiele für manipulierende Statistiken)

BENFORDsches Gesetz

lt. BENFORD beginnen Zahlen besonders häufig mit einer 1, etwas selterner mit einer 2 usw. usf.
Die Verteilung folgt der Formel:



$$P(d) = \lg(d+1) - \lg(d) = \lg\left(1 + \frac{1}{d}\right)$$

besser ist eigentlich Regel für diese Beziehung
nicht allgemeingültig, gilt aber für sehr viele Daten
besonders für Zahlen gültig, die in einem sehr großen Zahlen-Bereich (über mehrere Zehner-Potenzen) kommen

lässt sich auch für die Prüfung auf Daten-Manipulationen verwenden
selbst ausgedachte Zahlen, z.B. bei Bilanz-Fälschungen folgen nicht dieser Regel
oder im anderen Extremfall werden Zahlen so manipuliert, dass sie optimal zur Regel passen (wenn der Fälscher die Regel kennt)
wird teilweise sogar vor Gericht verwendet

Schummeln mit Statistik

verschiedene Interpretation für "Durchschnitt"

"Durchschnitt"	Definition	Berechnung	Bemerkungen
arithmetisches Mittel	Quotient aus der Summe aller Werte und der Anzahl der Werte	$\bar{x} = \frac{\sum x_i}{n}$	üblicher Durchschnitt
geometrisches Mittel			
harmonisches Mittel			gleicht Extremwerte aus
Median	ist der Wert der in der geordneten Reihe der Werte genau in der Mitte liegt (bzw. das arithmetrische Mittel der beiden mittleren Werte)		meist besser geeignet, da Ausreißer das Ergebnis weniger verfälschen
Modus	ist der am Häufigsten vorkommende Wert		

Visualisierung

Ziele:

- Darstellung komplexer Zusammenhänge auf einfache, verständliche Art und Weise
- Darstellung zusammenfassender Informationen
- Exploration von Daten
-

Problem:

- sind praktisch immer vereinfachsender Natur
- nur wenige Dimensionen möglich
- unbekannte Zusammenhänge können verloren gehen, weil man sie nicht vermutet und deshalb nicht in die Visualisierung eingeschlossen hat

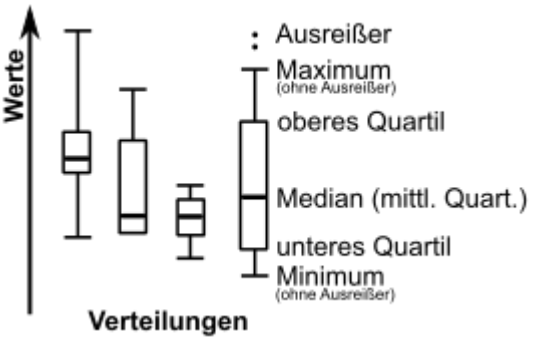
Diagramm-Arten:

- Balken- / Säulen-Diagramm
- Kreis-Diagramm
- Netz-Diagramm
- VENN-Diagramm
- Box-Plot's
- Violin-Plot
- Radar-Diagramm
- Punkt-Diagramme (x-y(-z)-Diagramme)
- Flächen-Diagramm
- …

zusätzlich 3D-Versionen z.B. Torten-Diagramm (statt Kreis-Diagramm) oder Zylinder-Diagramm (statt Säulen-Diagramm) ohne Zusatz-Information in der 3. Dimension (nur grafischer Effekt!)

Boxplot

Kasten-Diagramm
für große Mengen an Zahlen mit Maximum und Minimum (T-Grenzen)
ev. mit Ausreißer die als Einzel-Punkte auf der Ebene der Streuungs-Linie eingezeichnet
wird Streuungs-Linie gestrichelt gezeichnet
Kasten zeigt den Werte-Bereich zwischen oberen und unteren Quartil und damit die mittleren 50 % der Werte (in der geordnenten Liste)



stellt weiterhin z.B. den Median dar (als mittlere (dicke) Linie im Kasten)
gut für den Vergleich mehrerer Verteilungen

Arten der Achsen

lineare Achse
bieten gute Übersicht
Erwartung der Normal-Nutzer

logarithmische Achsen
bietet bessere Übersicht bei Werte-Bereichen über mehrere Dimensionen / Zehner-Potenzen
verbessert die Detail-Ansicht im Bereich kleiner Zahlen

reziproke Achsen

Schummeln mit Visualisierung

z.B. durch andere Achsen-Skalierungen
Ausschnitte auf x- und / oder y-Achse mit ev. variablem Werte-Bereich
fehlerhafte Diagramm-Typen für die Daten

Risiko-Kompetenz

Zahlen- und Kennwerte-Verständnis

Beispiel:
durch die Einführung eines diagnostischen Verfahrens sei die Sterblichkeit um 24 % gesunken

Fehl-Interpretationen:
von 1'000 untersuchten Personen sterben 240 weniger
von 1'000 untersuchten Personen sterben 240

richtig ist:
bei 1'000 untersuchten reduziert sich die Sterblichkeit vom Ursprungswert zum Neuwert um 24 %
→ Normal-Sterblichkeit liegt angenommen z.B. bei 10 % (also 100 von 1'000 Personen), dann liegt sie durch das diagnostische Verfahren jetzt nur noch bei 7,6 % (also rund 8 Personen von 1'000)

SIMPSON-Paradoxon

Gegeben sind z.B. die nebenstehenden Daten von Fußball-Spielern.
Die Frage ist nun, welchen Spieler sollte man von den beiden kaufen? Wer ist der bessere Spieler?

Saison	Spieler	Tore	Torschüsse	Quote
2017/18	Robert	5	20	25 %
	Nils	90	300	30 %
2018/19	Robert	120	330	36 %
	Nils	28	70	40 %

Q: Daten aus HPI-Kurs (fiktiv)

Da Nils in beiden Saison's die höhere Quoto wird man schnell verleitet, Nils auch als den besseren Spieler zu halten.
Betrachtet man aber beide Spielzeiten zusammen, dann ergibt sich anderes Bild. Hier stellt sich Robert als der bessere Spieler heraus, weil er insgesamt eine höhere Quote hat.

Saison	Spieler	Tore	Torschüsse	Quote
2017/18/19	Robert	125	350	36 %
	Nils	118	370	32 %

Q: Daten aus HPI-Kurs (fiktiv)

SIMPSON-Effekt beschreibt Veränderung von statistischen Kennwerten bei variablen Grundgesamtheiten. Die Einzel-Ergebnisse gehen mit unterschiedlichen Gewichten in die Gesamt-Bewertung ein.

beschrieben wurde der Effekt zuerst von Edward Hugh SIMPSON ()

Beispiel (Q: de.wikipedia.org)

	weiblich			männlich		
	bestanden	**gesamt**	**Durchfall-Quote**	**bestanden**	**gesamt**	**Durchfall-Quote**
1. Tag	7	8	12,5 %	1	1	0,0 %
2. Tag	1	2	50,0 %	2	3	33,3 %

An beiden Tagen wurde für die Frauen eine deutlich höhere Durchfall-Quote ermittelt. Betrachtet man aber die Gesamtheit (für beide Tage), dann dreht sich Bild:

	weiblich			männlich		
	bestanden	**gesamt**	**Durchfall-Quote**	**bestanden**	**gesamt**	**Durchfall-Quote**
1. Tag	8	10	20,0 %	3	4	25,0 %

Diskriminierungs-Klage gegen die University of California in Berkeley

Die Zahlen stammen aus dem Herbst 1973 und zeigen eine deutliche Ungleich-Verteilung. Auch ein Signifikanz-Test bestätigte, dass diese Zahlen nicht zufällig sein können.

	Bewerber	zugelassen
weiblich	4321	35 %
männlich	8442	44 %

Bei der genauen Analyse ergab sich aber ein weitaus differenziertes Bild. Von 101 Fakultäten gab es 16 in dem sich nur ein Geschlecht beworben hatte. Bei den restlichen mit Bewerbern beider Gesdchlechter ergab sich, dass in 4 Fakultäten die Männer signifikant höhere Erfolgs-Quoten hatten. Bei 6 Fakultäten ergab sich eine signifikant höhere Chance für Frauen.

8.x.y.2. Data Mining und Machine Learning

Ziele von Data Mining:

- Aussagen über vorhandene Daten treffen → Statistiken, Visualisierung
- unbekannte Zusammenhänge finden → Cluster-Analyse, Klassifizierungen, Visualisierungen, Assoziations-Regeln, Regressionen, Ausreißer-Erkennung
- Aussagen über die zukünftige Entwicklung der Daten machen → Trend's
- …

Themen zum maschinellem Lernen:

-

deskriptive und prädikative Analyse

deskriptive Analyse ist beschreibend
vorhandene Daten werden analysiert und Kennwerte bestimmt
Was ist passiert?

ev. erweitert durch die
diagnostische Analyse
sucht die Ursachen für die Daten / Werte
Warum ist etwas passiert

prädikative Analyse ist vorausschauend
vorhandene Daten werden dazu benutzt um Voraussagen für die Zukunft zu machen
Was wird vermutlich passieren?

ev. erweitert durch die
präskriptive Analyse
beeinflussen der weiteren Entwicklung der Daten / Werte
Wie kann man es bewirken?

Assoziations-Regeln

aus der Mitte der 90iger Jahre
typisch Warenkorb-Analyse
Was haben die Kunden gekauft?

Analyse von Warenkorben:
K1 = {Cola, Wasser, Whisky}
K2 = {Saft, Cola}
K3 = {Wasser, Bier}
K4 = {Wasser}
K5 = {Bier, Cola, Wasser}
K6 = {Cola, Bier}

K7 = {Bier, Wasser}
K8 = {Wein, Wasser}
K9 = {Wasser, Bier}

R1: WENN K_i = *{Wasser}* DANN K_i = *{Bier}* (Wenn im Warenkorb Wasser ist, dann ist auch Bier enthalten!)

Assoziations-Regel: {Wasser} → {Bier} heißt im OpenHPI-Kurs: Prozentsatz der Warenkörbe mit Wasser, die auch (besser wahrscheinlich: "schon") Bier enthalten
{Gin} → {Tonic}: Warenkörbe indenen sowohl Gin, als auch Tonic enthalten ist

Support:
die Regel X → Y hat einen Support s, falls s% aller Warenkörbe sowohl X als auch Y enthalten

Support s für R1 und die Beispiel-Warenkörbe K1 … K9:
bei 3 Körben von insgesamt 9 ist s = 3/9 = 33,3 %

großer Support bedeutet große Häufigkeit der Kombination, daraus können Maßnahmen abgeleitet werden: z.B.:

- gemeinsam bewerben
- gemeinsam präsentieren
- Kombinations-Pakete anbieten
- …

Konfidenz:
die Regel X → Y hat eine Konfidenz c, falls c% aller Warenkörbe, die X enthalten auch Y enthalten

Konfidenz c für R1 und die Beispiel-Warenkörbe K1 … K9:
es enthalten 7 Körbe Wasser, aber nur bei 3 auch Bier: c = 3/7 = 42,9 %

gesucht sind dann Regeln, die einen Mindest-Support sowie / und / oder eine Mindest-Konfidenz haben

Mining von Assoziations-Regeln

A-Priori-Algorithmus (Apriori-Algorithmus)

1. Finde alle Produkt-Teilmengen mit dem Mindest-Support
2. Leite daraus Assoziations-Regeln mit der Mindest-Konfidenz ab

Problem sind die riesigen Produkt- Teilmengen
praktisch 2^n Möglichkeiten

Support kann bei steigender Produkt-Menge nur sinken

→ Wenn eine Produkt-menge den Mindest-Support nicht erreicht, dann kann es auch keine Obermenge.
→ Bottum-up Kandidaten-Generierung

für die Beispiel-Warenkörbe bedeutet das, dass es bei so seltenen Einkäufen mit Whisky, dessen Support hier sehr wahrscheinlich unter einem sinnvollen Wert liegt, so dass keine Kombinationen Whisky mit Cola usw. usf. untersucht werden müssen

Ableitung der Assoziations-Regeln nur noch aus dem Rest der Warenkörbe

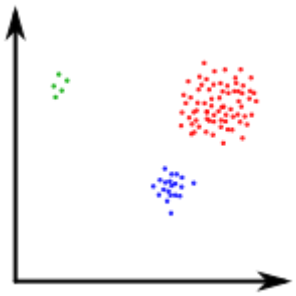
A-Priori-Algorithmus
→ http://www.vldb.org/conf/1994/P487.PDF

Clustering

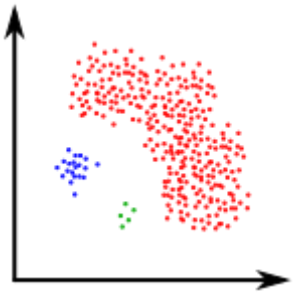
Gruppen-Bildung
Partitionieren, Clustern
sind meist entdeckerische, nicht-überwachte (Lern-)Verfahren

meist werden die Anzahl der gesuchten Cluster und das Ähnlicjkeits-Maß vorgegeben

Ähnlichkeit der Objekte innerhalb einer Gruppe soll sehr groß sein und die Abweichung / Nicht-Ähnlichkeit zu allen anderen Gruppen möglichst groß
als Maß wird die Distanz (Punkt-Abstand) benutzt
→ Objekte mit kleiner Distanz zueinander gehören zu einem Cluster, wenn auch die Distanz zu den anderen Cluster möglichst groß wird / ist
es gehen aber auch andere Ähnlichkeits-Merkmale → s.a. Duplikat-Erkennung

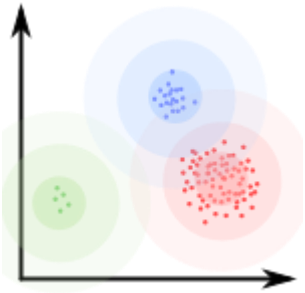


Problem bei Gruppen, die nicht Kreis-förmig sind
hier kann Abstand zwischen zwei zugehörigen Objekten größer sein, als der zu einem anderen Cluster



Dichte-basiertes Clustern
Bildung von Clustern, in den zu mindestens einem anderen Objekt eine große Ähnlichkeit besteht (und zu dem Nächsten Anderen der anderen Cluster möglichst größere Abstände)

Wahrscheinlichkeit-basiertes Clustern / weiches Clustern
für alle Objekt werden die Wahrscheinlichkeiten berechnet, wie sie zu einem Cluster gehören (über die Ähnlichkeit bezüglich des Cluster-Zentrums)



k-Means-Clustering

k steht für die Anzahl der Cluster, die herausgesucht werden sollen
Mean … Mittelwert

Verfahren:

1. zufälliges Auswählen von k Objekten als angenommene Cluster-Zentren
2. Ermittle für jedes andere Objekt den Abstand zu den Cluster-Zentren und ordne es dem dichtesten zu
3. Bildung eines neuen Cluster-Zentrums (Suche des Objektes im Zentrum) aus den in einem gesammelten Cluster Objekte
4. Wiederhole solange bei 2. bis keine Neuzuordnungen von Objekten mehr stattfinden

im originalen k-Means-Verfahren werden nicht Objekte als Cluster-Zentrum benutzt, sondern zufällige Positionen
Verfahren findet WORONOI-Zellen (auch VORONOI-Zellen)

Probleme:

- Anzahl der gesuchten Cluster muss vorgegeben werden
- bei ungünstigen Start-Zentren kann das Verfahren recht lange dauern, da sich die Abstände (Objekt - Zentrum) nur wenig unterscheiden
- Ausreißer werden immer zugeordnet

Verbesserungen sind:

- k-Median
- k-Means++

überwachtes und unüberwachtes Lernen

Ziel ist die Klassifizierung von (neuen) Objekten
Zuordnung eines (neuen) Objekt's zu vorhandenen Gruppen / Klassen

un-überwachtes Lernen (Unsupervised Learning / Training)

Verfahren funktionieren von sich heraus
es werden alle Daten für das Verfahren genutzt
es gibt keine vorgegebenen Trainings-Daten
keine

optimal sind besonders viele Attribute (oder abgeleitete Werte) einzubeziehen
ist aber etsprechend rechenaufwendig, also tendenz zu möglichst wenigen – aber klar funktionierenden Attributen (oder abgeleiteten Werten)

überwachtes Lernen (Supervised Learning / Training)

hier müssen Nutzer-Annotationen (Klassifizierungs-Informationen durch Nutzer) dazu kommen
verlangt also Trainings-Daten (oder ein Teil der Roh-Daten muss vorbearbeitet werden)

Problem ist die aufwendige Trainings-Phase, ev. werden echte Experten gebraucht
ev. können Wissens-Datenbanken benutzt werden

aktives Lernen (Active Learning)
System schlägt besonders wichtige / interessierende Atribute vor, die für den Lern-Prozess wichtig sind
damit Experten / Trainer nicht unnötig viele Daten beurteilen müssen, werden vom System nur die problematischen Objekte vorgeschlagen, die dann vom Experten annotiert werden sollen / müssen

Trainings-Daten / Test-Daten

Ziel ist die Verallgemeinerbarkeit des zugrundelegenden Modell's
fehlerfreies Funktionieren des Modell's bei neuen Daten
sichere Klassifizierung neuer Objekte

im Normalfall werden mehrere Modelle verglichen
sollte eine zeitliche Trennun möglich sein, dann sollte diese erfolgen
(z.B. bei Hass-Kommentaren nur mut Daten trainieren, die vor einem Stichtag erschienen sind, getestet wird dann mit Daten nach dem Stichtag)

Schritte beim Training eines Neuronalen Netzwerkes

- zufällige Initialiserung der Gewichte
- Vorhersage auf ein Trainings-Beispiel
- Berechnung der Abweichungen der Vorhersage zur Annotation

Trainings-Daten

sind (der größere) Teil der annotierten Daten, die für das Anlernen des System benutzt werden
sie werden benutzt, um die Parameter des Modell zu optimieren
ev. in mehreren Durchläufen / Interationen

Test-Daten

sind (ein kleinerer) Teil der annotierten Daten, die nur für den abschließenden Test des System benutzt werden
typischerweise rund 20 % der zur Verfügung stehenden annotierten Daten
diese Daten sind in der Entwicklungs- und Optimierungs-Phase des Modell's nicht einzusetzen

bei der Cross-Validierung (Kreuz-Validierung, f-fold cross validation)
Aufteilung der annotierten Daten auf k annähernd gleich große Partitionen
benutzt man z.B. einen Teil (meist rund 10 %) der Daten für den abschließenden Test, die anderen 90 % werden als Trainings-Daten benutzt
also Training mit k-1 Partitionen
man nimmt dann aus dem gleichen Gesamt-Datenbestand wieder einen Umfang von 90 % zum Trainieren und den restlichen Teil für das Testen
Qualitäts-Test mit den restlichen Partitionen
usw. usf.

weitere Verbesserung möglich, wenn neben den üblichen 20 % Test-Daten noch mal rund 20 % Validierungs-Daten abgetrennt werden
mit den restlichen rund 60 % wird trainiert
mit den Validierungs-Daten wird das System selbst feingetunt / optimiert
hier kann man auch noch mit der Cross-Validierung arbeiten

Overfitting (Überanpassung)

passiert, wenn das Modell zu stark an die Trainigs-Daten angepasst ist
tritt häufig dann auf, wenn das Modell sehr komplex ist und sehr viele Attribute für die Klassifizierung eine Rolle spielen

System erkennt etwas anderes, weil Trainings-Daten nicht optimal ausgewählt wurden, oder auch wenn Trainings-Daten auch zum Testen benutzt werden

im Extremfall ist es so, dass das System für jedes Trainings-Objekt genau seine Zuordnung gelernt hat

Beispiel: Erkennung von Wolfs- und Haski-Bildern
System sollte beide Hunde-Artigen unterscheiden lernen, was mittels Trainings- und Test-Daten auch perfekt geklappt hat.
Bei einem Praxis-Test erkannte das System die Wölfe aber nicht richtig, bzw. einige Haski's als Wölfe.
Bei der Suche nach der / den fehler-Quellen stellte sich heraus, dass das System nicht Hunde und Wölfe unterschied, sondern Bilder mit Schnee und ohne.
Die Wölfe waren immer auf Bildern mit Schnee abgebildet, und genau dass hatte das System gelernt.

Ähnliches Beispiel aus dem Militär-Bereich:

System sollte feindliche und eigene Panzer unterscheiden lernen. Klappte auch wieder perfekt. Im Praxis-Test und der nachfolgenden Fehler-analyse stellete sich heraus, dass das System nur Schönwetter-Foto's der eigenen Panzer, von den Schlechtwetter-Foto's der feindlichen Panzer unterscheiden konnte.

Klassifizierung

Aufgabe ist es, ein neues Objekt in eine der vordefinierten / erlernten Klassen einzuordnen

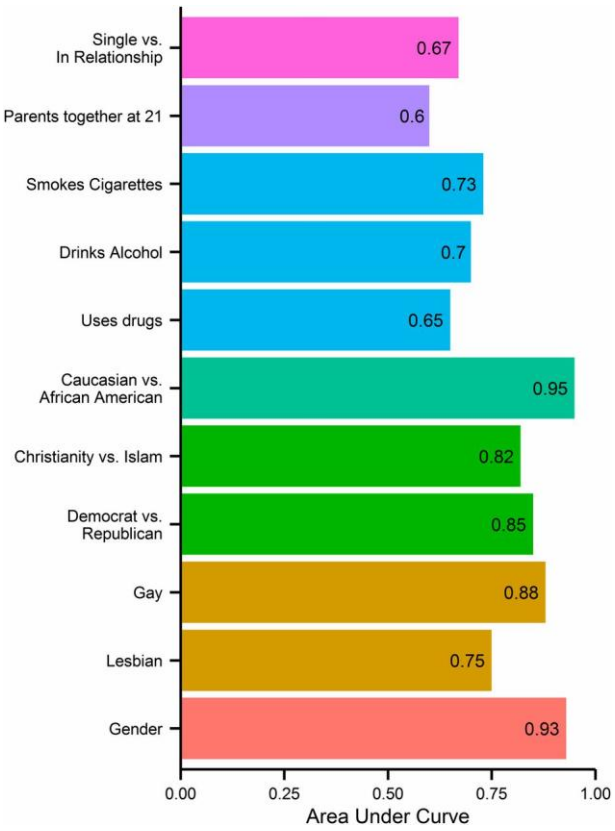
typische Beispiele:

- Bild-Erkennung
- Betrugs-Erkennung
- Personen
- Tumore
- Auto-Kennzeichen
- autonomes Fahren
- Duplikat-Erkennung
- Stimmungs-Erkennung
- Hass-Kommentare
- Stimm- und Sprach-Erkennung
- Finger-Abdrücke
- Bonitäts-Prüfung
- Erkennung von Risiko-Fahrern
- Empfehlungs-Systeme
- …

Beispiel: Klassifizierung von Facebook-Nutzern anhand der Likes zu Webseiten oder Produkten (nicht zu anderen Personen!)
Aufgabe war die Ableitung anderer Eigenschaften / bzw. eine neuartige Klassifizierung in Gruppen, zu denen kein Attribut-Bezug bestand

es wurden nur binäre Attribute betrachtet

die Wahrscheinlichkeit für die Exaktheit der vorausgesagten Eigenschaften / Klassifizierungen sind in der Abb. dargestellt
im

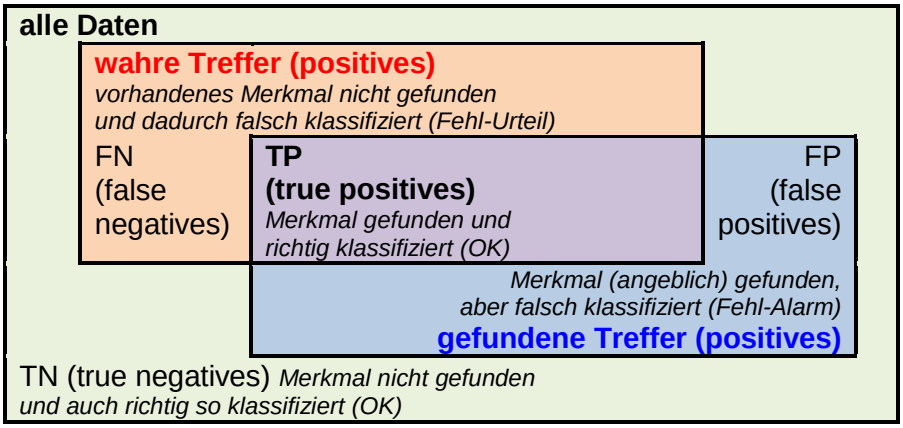


Q: https://www.pnas.org/content/110/15/5802

Erfolgs-Maße

für die binäre Klassifizierung, also Zuordnung zu der einen oder der anderen Klasse

gemeint immer ein Merkmal (ist weiblich, hat Tumor, ist Hund, …) und das Gegenstück dazu (ist nicht weiblich, hat kein Tumor, ist kein Hund, …)



Mensch	System		Realität
Bewertung der Beurteilung / Klassifizierung	**Beurteilung Klassifizierung Klasse**	**Gruppe**	**Attribut Merkmal Eigenschaft**
OK	Merkmal erkannt / angezeigt / zugeord.	**TP** r_p	**Merkmals-Träger**
Fehl-Alarm / -Urteil		**TN** r_n	**kein Merkmal**
Fehl-Urteil	Merkmal nicht erkannt / angezeigt / zugeord.	**FP** f_p	**Merkmals-Träger**
OK		**FN** f_n	**kein Merkmal**

Akkuratheit / Korrektheit / Treffergenauigkeit / Vertrauens-Wahrscheinlichkeit / Korrekt-Klassifikations-Rate / :

Accuracy = ACC = (TP + TN) / (TP + TN + FP + FN) = (TP + TN) / n = (TP + TN) / (P + N)

n = P + N = TP + TN + FP + FN

Präzision / Genauigkeit / Spezifität / Richtig-Negativ-Rate / Relevanz / Wirksamkeit : positiver Vorhersage-Wert, positiver prädikativer Wert
Specificity, correct rejection rate, true negative rate, positive predicative value (PPV)

Precision = PPV = TP / (TP + FP) = 1 – FDR

TNR = TN / (TN + FP) = TN / N = 1 – FPR **=** Specificity = $r_n / (r_n + f_p)$

Vollständigkeit / Erkennungs-Rate / Sensitivität / Richtig-Positiv-Rate / Treffer-Quote:
Sensitivity, true positiv rate (TPR), hit rate

Recall = TPR =TP / (FN + TP) = 1 – FNR **=** Sensitivity = $r_p / (r_p + f_n)$
= TP / P

Forderung: hohe Korrektheit bzw. hohe Genauigkeit bei hoher Vollständigkeit
→ Ziel-Konflikt
zur Auflösung wird als Maß das harmonische Mittel (F-Maß, F_1-Score) aus Precision und Recall berechnet

F-Measure = (2 * Precision * Recall) / (Precision + Recall))
F1 = (2 * PPV * TPR) / (PPV + TPR) = (2 * TP) / (TP + TN + FP + FN)

Ziel ist möglichst hoher Wert für F-Measure
harmonisches Mittel bestraft schon geringfügige fehlerhafte Präzisionen und Vollständigkeiten (deutlicher als das arithmetrische Mittel)

		Gold-Standard		
		Gold positiv	Gold negativ	
Predicted Entity	Pred positiv	**wahr positiv** true positive	**falsch positiv** false positive	**Präzision** precision
	Pred negativ	**falsch negativ** false negative	**wahr negativ** true negative	
		Rückruf recall		

weitere Maße:

Falsch-Negativ-Rate / :
false negative rate (FNR) / miss rate

= FNR = FN / (FN + TP) = 1 – TPR = FN / P **=** $= f_n / (r_p + f_n)$

Falsch-Positiv-Rate / Ausfall-Rate:
false positive rate (FPR) / fallout

Fallout = FPR = FP / (FP + TN) = FP / N = 1 – TNR **?** $= r_n / (r_n + f_p)$

False discovery rate (FDR):

FDR = FP / (FP + TP) = 1 – PPV

False omission rate (FOR):

FOR = FN / (FN + TN) = 1 – NPV

Trenn-Fähigkeit / Segreganz / negativer Vorhersage-Wert:

negative predicative value (NPV)

= NPV = TN / (TN + FN) = 1 – FOR = = r_n / (r_n + f_n)

Falsch-Klassifikations-Rate / Klassifikations-Fehler:

= (f_p + f_n) / (r_p + f_p + r_n + f_n) = (f_p + f_n) / n = 1 - Accuracy

likelihood ratio (LR) / Likelihood-Quotienten-Test:

LRpositiv = Sensitivität / (1- Spezifität)

LPnegativ = (1- Sensitivität) / Spezifität

MATTHEWS-Korrelations-Koeffizent (Matthews correlation coefficent):

MCC = (TP * TN – FP * FN) / Wurzel((TP + FP) * (TP + FN) * (TN + FP) *(TN + FN))

Threat score (TS) / Critical Sucess Index (CSI):

TS = CSI = TP / (TP + FN + FP)

Bookmaker Informedness (BM) / Informedness:

BM = TPR + TNR – 1

Markedness (MK):

MK = PPV + NPV – 1

Effektivitäts-Maß (E):

E = 1 / (α(1 / P) + (1 – α)/R
bei α = 1 → Genauigkeit bei α = 0 → Treffer-Quote
0 … beste Effektivität 1 … schlechteste Effektivität

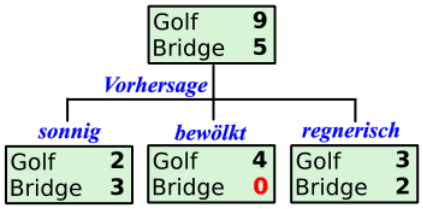
interessante Links:
https://www.spiegel.de/wissenschaft/mensch/medizinische-tests-und-statistik-denken-sie-immer-falsch-positiv-a-1087042.html (Tücken der Statistik (fiktives Scharlach-Beispiel))
https://www.spiegel.de/gesundheit/diagnose/viele-aerzte-verstehen-statistiken-zu-diagnosen-nicht-a-844210.html (falsches Statistik-Verständnis unter Ärzten)

Entscheidungs-Bäume (decision trees)

Klassifizierungs-Verfahren

Beispiel (von OpenHPI)
Was machen wir heute? Gehen wir raus golfen oder spielen wir drinnen Brigde?

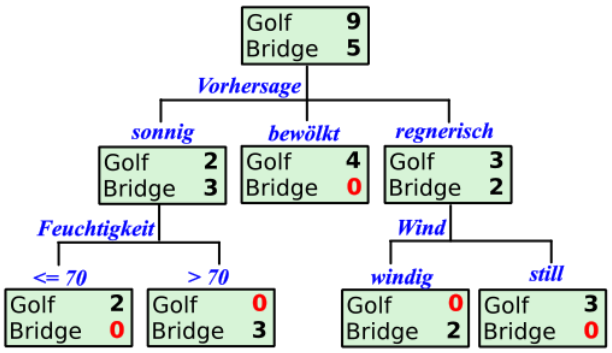
In einem ersten Verfahren könnte man z.B. versuchen eine Entscheidung nur über die Wetter-Vorhersage zu machen. Dabei ergäbe sich der folgende Baum:



Trainings-Daten

Vorhersage	Temp. (°F)	Feucht. (%)	Wind	Aktivität
sonnig	85	85	still	Bridge
sonnig	80	90	windig	Bridge
bewölkt	83	78	still	Golf
regnerisch	70	96	still	Golf
regnerisch	68	8	still	Golf
regnerisch	65	70	windig	Bridge
bewölkt	64	65	windig	Golf
sonnig	72	95	still	Bridge
sonnig	69	70	still	Golf
regnerisch	75	80	still	Golf
sonnig	75	70	windig	Golf
bewölkt	72	90	windig	Golf
bewölkt	81	75	still	Golf
regnerisch	71	80	windig	Bridge

Für "bewölkt" ergibt sich hier schon eine eindeutige Klassifizierung zu "Golf", da "Bridge" hier mit 0 Datensätzen in den Trainings-Daten vertreten ist.
Nun müssen die Gruppen "sonnig" und "regnerisch" noch weiter unterschieden werden. Bei der Wahl der Feuchtigkeit und einem Trigger-Wert von 70 lässt sich eine saubere Zweitteilung erzielen, wobei jede Teilgruppe eine 0-Gruppe enthält.
Die Gruppen "regnerisch" wir mittels des Attribut's "Wind" in die zwei Teilgruppen "windig" und "still" geteilt, die ebenfalls eine 0-Gruppe enthalten.
Damit ist ein Entscheidungs-Baum aufgebaut worden.



Vorteile:

- für Menschen gut bachvollziehbar
- anschaulich
- einfach zu implementierbarer Algorithmus

Ist dieser Baum aber der günstigste / schnellste / effektivste?
Bei mehr-dimensionalen Ansätzen kommt man zu Entscheidungs-Wäldern, in deren Ergebnis meist Wahrscheinlichkeiten für Klassifikationen stehen.
Support-Vektor-Maschine (Support vector machine) betrachten die Objekte in einen vieldimensionalen Raum und ziehen eine Trenn-Linie / - -Ebene / -???, um die Objekte jeweils der einen oder anderen Seite zuzuordnen.
Beim "Naive Bayes"-Verfahren (Verfahren der naiven Basis) werden die Wahrscheinlichkeiten für die Klassifikation berechnet und dann angewendet. Das Verfahren geht "naiv" davon aus, dass alle Attribute unabhängig voneinander sind.

neuronale Netze

versuchen mittels digitaler Technik das assoziative Denken den Mensch nachzubilden / zu simulieren

bestehen aus kleisten Elementen, den "Neuronen"
sind vernetzte Rechen-Elemente, die über Rück-Kopplungen die Bedeutung der verschiedenen Eingangs-Signale erlernt und dann an Ausgabe-"Neuronen" dann die Klassifikationen ausgeben

neuronale Netze sind i.A. vielschichtig, jedes "Neuron" ist mit jedem "Neuron" der nächsten Schicht verbunden
die Signal-Stärke, mit der es die einzelnen Eingaben / Eingabe-Neuronen beachtet wird über die Gewichte gelernt, dazu dient die Rück-Kopplung (Back-Propagation))

Neuron bestimmt zusätzlich seine Ausgangs-Signal-Stärke über die Aktivierungs-Funktion

Die Topologie eines Neuronalen Netzwerkes wird durch die Anzahl der Schichten / Ebenen und durch die Anzahl der Neuronen pro Schicht charakterisiert.
Die Kapazität eines Neuronalen Netzwerkes wird durch dessen Größe / Komplexität bestimmt.
Bei einer sehr großen Kapazität eines Neuronalen Netzwerkes besteht wieder die Gefahr, dass das Netzwerk die Trainings-Daten "auswendig" lernt und nicht wirklich für neue Klassifikationen taugt.

Deep Learning

derzeit stark gehypt
bietet sehr viele (neue) Möglichkeiten
besonders für große Daten-Mengen und komplexe Daten (z.B. Bilder-Inhalte) geeignet

typische Anwendungs-Gebiete:

- Bilder-(Inhalte-)Erkennung
- Erkennung von Tier-Arten
- Kennzeichen-Erfassung
- Personen-Verfolgung
- Gesichter-Erkennung
- Authentifizierungs-Verfahren
-

Schichten, die zwischen Eingangs- und Ausgangs-Schicht liegen sind i.A. für den Benutzer verdeckt (hidden layer's).
verfolgt man, was genau in den einzelnen Schichten passiert, dann stellt man fest, dass i.A. die ersten Schichten sehr abstrakte Klassifizierungen vornehmen (z.B. linke obere Seite des Bildes ist heller usw. usw.)
mit jeder weiteren Schicht werden die Klassifikationen –bezogen auf die Ziel-Klassifikation immer konkreter
hier werden jetzt konkrete Formen / Umrisse / Gebilde / Farbverläufe usw. usf erkannt, die dann in der letzten Schicht zur Ziel-Klasse kombiniert werden

Training erfolgt mit vielen Bildern und Foto's vom Objekt und zufälligen Start-Gewichten und Aktivierungs-Funktionen
Gegen-Training mit alternativen Bildern / Foto's

Fairness / systematische Abweichung

interne Autonomie des Lernens / Training's der Neuronalen Netzwerke sowie deren eigenständiges Agieren in der Nutz-Phase bergen diverse Gefahren:

- Anpassung von Schwell-Werten / Triggern in von Menschen nicht gewollte Richtungen (rassistisches / sexistisches Agieren; Grenzwerte bei Geld-Überweisungen (vielleicht ist Überweisung von 30'000 Euro schon ein zeichen für Geldwäsche / Terrorismus-Unterstützung oder nur ein Auto- oder Haus-Kauf)
-

typisches Beispiel:
Compass-Skandal
System sollte die Entscheidung fällen, ob eine inhaftierte Person auf Bewährung frei kommen soll
es sollte die Wahrscheinlichkeit der Rückfälligkeit ermittelt werden
System wurde so erstellt, dass es weiße Personen bevorteilte und farbige benachteiligte, obwohl sie andere vergleichbare Attribut-Werte hatten

Ursache war das Training des Sstem mit früheren Fällen und Entscheidungen von Richtern (, die i.A. zu Ungunsten von Farbigen entscheiden (statistisch gesichert))
mehr Farbige in den Trainings-Daten

das System hat nur den vorhandenen Rassismus gelernt und weiter angewendet

chinesische Kriminellen-Erkennung (an Gesichtern):
Trainings-Daten waren Foto's von Krimminellen und Foto's von anderen Personen aus dem Internet

systematische Fehler:
Kriminellen-Foto's eher düster, deprimiert, wütend, abweisend, aggressiv, …
Internet-Protrait-Foto's sind eher gut gelaunt, liebenswürdig, lustig, kontaktfreudig, aufgeschlossen, …
dazu kommt die Subjektivität der Richter (Vorverurteilung von dreckigen, düsteren, widerspenstigen, … Personen) einschließlich politisch motivierter Urteile, die aber gar nicht wirklich kriminelle Hintergründe / Wahrheiten beinhalten

Terroristen-Erkennung durch Geheimdienste:
Trainings-Daten bestanden aus 7 Telefon-Daten von Terroristen gegen 100'000 andere Telefon-Verhaltensweisen

Problem Korrelation gegen Kausalität (→)

erklärbare KI

explainable AI (XAI)
ist der Versuch und die Wissenschaft um die Erkundung und Erklärung der Vorgänge in der KI und besonders in Neuronalen Netzen (NN)

Nachvollzug von Entscheidungen der KI / Neuronalen Netzen

Schaffung von Rechtssicherheit (z.B. bei Personen-Klassifizierung (Krininelle, Terroristen, …))

funktioniert gut bei Entscheidungs-Bäumen (→) aber deutlich schlechter bei NN oder Support Vektor Maschinen

derzeit gibt es gute Erkenntnisse auf dem Gebiet der Bild-Erkennung
ebenfalls bei Stimmungs-Anlaysen (für texte), weil man die bedeutsamen Worte anzeigen lassen kann

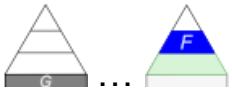
Problem-Beispiel aus der Bild-Erkennung
System lernt Pferde-Bilder und andere "Nicht-Perde"
System versagte in der Praxis
Trainings-Daten hatten auf jedem Foto eine copyright-Kennung, die Nicht-Pferd-Bilder nicht
letztendlich hatte das System nur das copyright-Zeichen gelernt

9. Datenbanken und Datenschutz

10. Suchmaschinen

https://open.hpi.de/courses/searchengine2017 (abgelaufener OpenHPI-Kurs zum Selbststudium)

10. komplexe und Übungs-Aufgaben



Aufgaben:

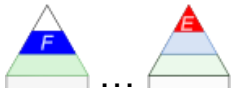
1.

2.

3. Bestimmen Sie die Kardinalitäten der nachfolgenden Beziehungen! Erläutern Sie Ihre Entscheidung!

a)	Schüler	-----	hat	-----	Lehrer
b)	Auto	-----	hat	-----	Motor
c)	Auto	-----	hat	-----	Reifen
d)	Schüler	-----	erhält	-----	Zeugnis
e)	Student	-----	darf_arbeiten_an	-----	PC
f)	Student	-----	besucht	-----	Kurs
g)	Schüler	-----	ist_befreundet_mit	-----	Schüler
h)	Leser	-----	leiht_aus	-----	Buch
i)	Schüler	-----	hat	-----	Zeugnis

4.



Aufgaben für die gehobene Anspruchsebene:

1.

2. Testen Sie Ihr Datenbank-Wissen! Was können Sie davon schon?

http://gisbsc.gis-ma.org/GISBScL4/de/html/GISBSc_VL4_VL_WC.html

3.

Literatur und Quellen:

/1/
ISBN

/2/
ISBN

/3/
ISBN

/4/
ISBN

/5/ GIERHARDT, Horst:
Datenbanken: Entity-Relationship-Model.-Bad Laasphe, Städtisches Gymnasium, 2014.-horst@gierhardt.de
(http://www.oberstufeninformatik.de/Datenbanken/ERMTheorie.pdf)

/10/ SELLE, Stefan:
Künstliche Neuronale Netzwerke und Deep Learning.-Saarbrücken (2018)

Die originalen sowie detailliertere bibliographische Angaben zu den meisten Literaturquellen sind im Internet unter http://dnb.ddb.de zu finden.

Kurz-Referenz auf Quelle

/###/ Syntax-Diagramme für SQLite
https://www.sqlite.org/syntaxdiagrams.html

Internet-Seiten, etc.

/A/ Wikipedia
http://de.wikipedia.org

Abbildungen und Skizzen entstammen den folgende ClipArt-Sammlungen:

/A/ 29.000 Mega ClipArts; NBG EDV Handels- und Verlags AG; 1997

/B/

andere Quellen sind direkt angegeben.

Alle anderen Abbildungen sind geistiges Eigentum:

/I/ lern-soft-projekt: drews (c,p) 1997 - 2022 lsp: dre

verwendete freie Software:

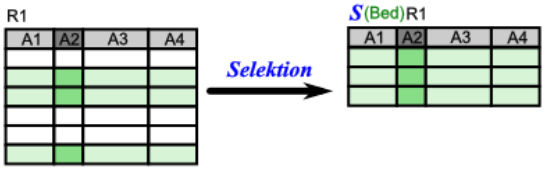
- **Inkscape** von: inkscape.org (www.inkscape.org)
- **CmapTools** von: Institute for Human and Maschine Cognition (www.ihmc.us)

⊞- (c,p) 2015 - 2022 lern-soft-projekt: drews -⊟
⊞- drews@lern-soft-projekt.de -⊟
⊞- http://www.lern-soft-projekt.de -⊟
⊞- 18069 Rostock; Luise-Otto-Peters-Ring 25 -⊟
⊞- Tel/AB (0381) 760 12 18 FAX 760 12 11 -⊟

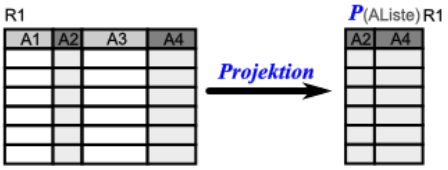
derzeit Aussortiertes

Methoden für eine Tabelle (oder Abfrage)

- **Selektion** — Auswahl von Zeilen (Tupeln) aus einer Relation aufgrund einer oder mehrerer Bedingungen

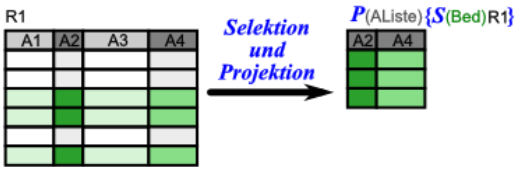


- **Projektion** — Auswahl (/ Einschränkung) von Spalten (Attribute) aus einer Relation aufgrund einer Auswahlliste



möglich auch Kombinationen, praktisch sinnvoll um die anzuzeigende / bereitgestellte Daten-Menge noch weiter zu reduzieren:

- **Selektion und Projektion** — Auswahl von Zeilen (Tupeln) und auch (Einschränkung) von Spalten (Attributen) aus einer Relation



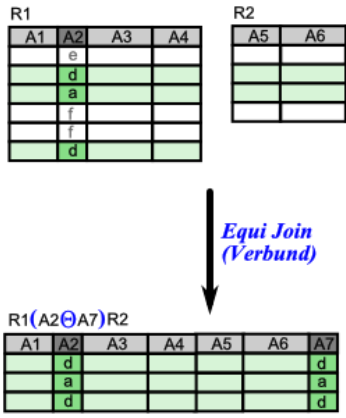
-

-

Methoden für mehrere Tabellen und / oder Abfragen

- **Verbund**
  Join
  (Equi Join)

  Verknüpfung von Relationen (Tabellen) aufgrund von Attributs-Beziehungen



- **Mengen-Operation**

  Bildung von Vereinigung, Differenz und / oder Durchschnitt auf Relationen mit gleicher Struktur