

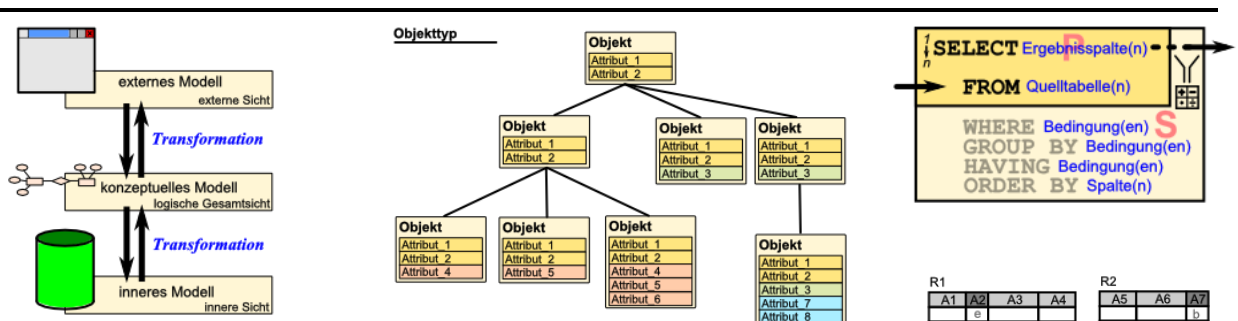
Informatik

für die Sekundarstufe II

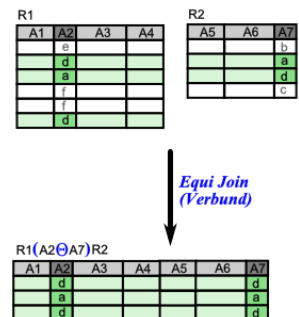
- Datenbanken -

Teil 1: Einführung, Modelle, Theorie

Autor: L. Drews

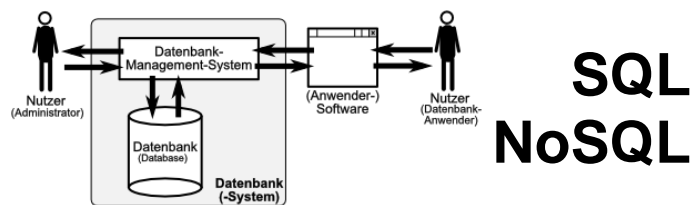


ACCESS
AND | OR | XOR
BASE



Wie jagt ein typischer Programmierer afrikanische Elefanten?

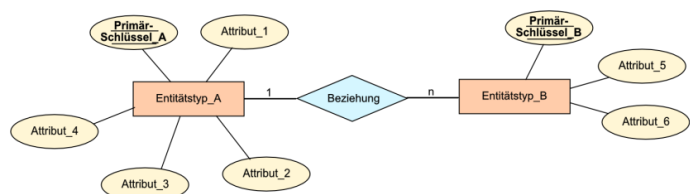
```
{
  Gehe nach Afrika
  Beginne am Kap der guten Hoffnung
  Durchkreuze Afrika von Süden nach Norden
  bidirektional in Ost-West-Richtung
  Für jedes Durchkreuzen tue
  {
    Fange jedes Tier, das Du siehst
    Vergleiche jedes Tier mit Elefant
    Halte an bei Übereinstimmung
  }
}
```



SQL
NoSQL

Wie jagt ein SQL-ler afrikanische Elefanten?

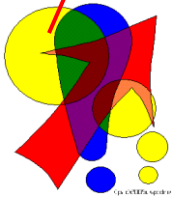
```
SELECT elefant
FROM afrika
```



V. 0.12a (2023)

Legende:

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



Nutzungsbestimmungen / Bemerkungen zur Verwendung durch Dritte:

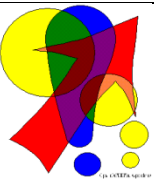
- (1) Dieses Skript (Werk) ist zur freien Nutzung in der angebotenen Form durch den Anbieter (lern-soft-projekt) bereitgestellt. Es kann unter Angabe der Quelle und / oder des Verfassers gedruckt, vervielfältigt oder in elektronischer Form veröffentlicht werden.
- (2) Das Weglassen von Abschnitten oder 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)	
creativecommons (cc) 	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	6
Definition(en): Informatik	7
Definition(en): Information	7
1. Datenbanken – Erkundungen	8
Problem-Fragen für Selbstorganisiertes Lernen	8
Definition(en): Datenbank (allgemein)	9
Orientierungs-Beispiel:	13
die größten Datenbanken der Welt (pauschal; ohne klares Ordnungs-Kriterium)	14
1.1. von der Karteikarte zur Computer-Datenbank	16
Problem-Fragen für Selbstorganisiertes Lernen	16
1.1.0. Karteikarten, Karteien und Register.....	17
Daten – Was ist das eigentlich?	21
Definition(en): Daten	21
1.1.1. Daten in Dateien	23
Definition(en): Datei.....	27
1.1.2. lokale Datenbanken.....	29
Definition(en): Dateisystem	31
1.1.3. globale Datenbank-Systeme.....	32
Wo geht es hin? Was passiert derzeit?	35
(kurze) Genealogy der rationalen Datenbank-Systeme	38
2. Datenbank-Entwurf – von der Theorie zum Konzept	39
Problem-Fragen für Selbstorganisiertes Lernen	39
2.0. Grundbegriffe	40
2.0.1. Daten, Informationen und Nachrichten.....	40
Definition(en): Information	41
Definition(en): Daten	42
Definition(en): Nachrichten	42
2.0.2. Datenbanken und Datenbank-Systeme	43
Definition(en): Datenbank(-System) / Datenbank i.w.S.....	43
Definition(en): Datenbank-Management-System	47
Definition(en): Datenbasis / Datenbank i.e.S.	51
2.0.3. allgemeine Merkmale / Charakteristika von Datenbanken.....	52
2.0.3.1. Redundanz.....	54
Definition(en): Redundanz.....	54
Exkurs: "Zauber"-Trick	56
Definition(en): Redundanz 1. Ord.....	57
Definition(en): Redundanz 2. Ord.....	58
Definition(en): Redundanz 3. Ord.....	59
2.0.2.2. Daten-Integrität.....	60
Definition(en): Integrität	60
2.0.2.3. Daten-Konsistenz	65
Definition(en): Konsistenz.....	65
2.0.2.4. Authentizität.....	68
Definition(en): Authentizität	68
2.0.2.5. Vertraulichkeit.....	71
Definition(en): Vertraulichkeit	71
2.1. Datenbank-Modellierung	73
2.1.1. Datenbank-Modelle	75
Definition(en): Datenbank-Modell	76
Definition(en): Daten-Modellierung	77
2.1.1.1. Datenmodell	80
Definition(en): Daten-Modell.....	80
2.1.2. übergreifende / übergeordnete Modelle	80

3-Ebenen-Modell	80
Definition(en): konzeptionelles Daten-Schema	86
Definition(en): logisches Daten-Schema	86
Definition(en): physisches Daten-Schema	86
2.1.3. das Entity-Relationship-Modell (ERM)	88
Definition(en): Entität / Entity	89
Definition(en): Attribute	90
Definition(en): Domäne / Attributs-Ausprägung	91
Definition(en): Entitäts-Typ	91
Definition(en): Beziehungen / Relationship's	91
Definition(en): Kardinalität / Beziehungs-Typ	92
Definition(en): Relation	92
Definition(en): Entity-Relationship-Modell (ERM)	93
(informatische) Begriffs-Welten	94
2.1.4. Entity-Relationship-Diagramme (ERD)	96
Definition(en): Entitätsschlüssel / Schlüssel / Schlüsselattribut	101
Definition(en): Primärschlüssel	101
Definition(en): Fremdschlüssel / Foreign key	102
2.1.5. Erweiterungen des Entity-Relationship-Modells	108
2.1.6. Transformation eines ERM (ERD) in eine relationale Datenbank	109
Definition(en): Relationen-Schema	111
2.1.6.1. Weiterentwicklungen des Entity-Relationship-Modells	118
2.1.7. Überführung von Tabellen in eine relationale Datenbank durch	
Normalisierung	120
2.2. relationales Daten(bank)-Modell	121
2.2.0. Grundbegriffe und Allgemeines	121
Definition(en): relationales Daten-Modell	121
Definition(en): Relationales Datenbank-Management-System (RDBMS)	122
2.2.0.1. Sind Tabellen und Relationen ein und das Selbe?	123
Definition(en): Relation (allg.)	124
2.2.0.2. Grund-Begriffe, -Elemente und -Verfahren in relationalen Datenbank-	
Modellen	125
Definition(en): flache Tabellen / flache Relationen	125
Schlüssel-Kandidaten	125
Definition(en): Schlüssel-Kandidat	125
Primär-Schlüssel	126
Definition(en): PrimärSchlüssel	126
Fremd-Schlüssel	126
Definition(en): Fremd-Schlüssel	126
referenzielle Integrität	127
Definition(en): referentielle Integrität	127
Relationen-Schema	133
2.2.1. Überführung der Entity-Relationship-Modell's in ein relationales	135
Exkurs: CODD's Regel zur Spezifikation einer relationalen Datenbank	136
CODD's Regel für eine ideale Datenbank	136
2.2.2. Normalisierung	138
Definition(en): Normalisierung	139
Definition(en): funktionale Abhängigkeit	141
2.2.2.1. Nullte Normalform	142
2.2.2.2. Erste Normalform	144
2.2.2.3. Zweite Normalform	148
2.2.2.4. Dritte Normalform	151
2.2.2.5. weitere Normalformen	154
2.2.2.5.1. BOYCE-CODD-Normalform (BCNF)	154
2.2.2.5.2. die 4. Normalform	155
2.2.2.5.3. die 5. Normalform	155
2.2.2.6. Algorithmen der Normalisierung	156

2.2.2.7. Zusammenfassung,	157
Definition(en): relationale Datenbank	157
2.2.2.8. mathematisch orientierte Darstellung des relationalen Datenbank-Modell .	160
2.3. andere Daten(bank)-Modelle	162
2.3.1. hierarchisches Datenbank-Modell.....	164
2.3.2. Baum-artige Daten(bank)-Modelle.....	168
2.3.2.1. binäre Bäume	168
2.3.2.2. weitere Baum-Strukturen	169
2.3.3. netzwerkartiges Daten(bank)-Modell	170
Definition(en): CAP-Theorem	172
Definition(en): ACID	172
2.3.4. Objekt-orientiertes Daten(bank)-Modell.....	173
Definition(en): Objekt-orientierte Datenbank.....	174
2.3.5. Dokument(en)-orientiertes Daten(bank)-Modell	175
2.3.6. Graph-Datenbanken	177
Definition(en): Graph-Datenbank-Management-System	180
Fluss-Überquerungs-Problem als Graph-Modell.....	180
2.4. Transaktionen und das Transaktions-Modell.....	184
Definition(en): Transaktion	185
2.5. Daten-Verteilung und -Sicherheit in Datenbanksystemen	186
Definition(en): Dauerhaftigkeit / Durability	194
Definition(en): Replikation	194
Literatur und Quellen:	195
Kurz-Referenz auf Quelle.....	195
Internet-Seiten, etc.....	195
derzeit Aussortiertes	619

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 Video's anschauen und die Quize lösen. Leider gibt es keinen Zugriff mehr auf die Hausaufgaben und die abschließende Testung. Aber das wird 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 vorbeischaun. 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 hierbei 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 sich 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 sein. 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 untergeordneten Niveau schon die Grenze des Lehrplans sind, aber es könnten auch Themen des – von mir subjektiv eingestuft – 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 ausschließlich 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 eignen sich auch für Projekte, Grob-Orientierungen für Schüler-Vorträge usw. usf. Wenn 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>

<https://www.kstbb.de/informatik/rdb/index.html> (sehr umfangreiche Schritt-für-Schritt-Aufarbeitung der Datenbank-Entwicklung)

nur für den internen Gebrauch!:

<https://books.google.de/books?id=5HXM-IFIXXcC>

Definition(en): Informatik
Informatik ist die Wissenschaft von der Verarbeitung von Informationen.

Definition(en): Information
Informationen sind Daten mit einer Bedeutung.

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?

Wie werden Daten / Datenbanken analysiert?

Was macht ein Datenbank-Administrator?
Warum komme ich nicht an alle Daten einer Datenbank heran?

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.

Eine Datenbank ist eine Sammlung logisch zusammenhängender Daten, die von einem Verwaltungssystem (Datenbank-Management-System) organisiert werden.

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 gespeichert 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!*

Kriterien für eine allgemeine Analyse / Charakterisierung einer Datenbank

- Größe des verwendeten Speichers (z.B. in Byte)
- Anzahl gespeicherter Datensätze
- Anzahl und Verteilung der Server
- historische Entwicklung (z.B. des verwendeten Speicher's ; ...)
- Thema / Inhalt der DB / gespeicherte Objekte
- verwendete Datenbank- bzw. Programmier-Sprache(n)
- verwendetes Datenbank-System
- Nutzungs-Möglichkeiten
- Benutzer-Gruppen (z.B. unangemeldete/anonyme Nutzer / Jederman; registrierte Nutzer; eingeschränkter Personenkreis)
- Quelle der Daten (automatisiert; händische Eingabe; ...)
- Zugriffs- und Nutzungs-Möglichkeiten ((Internet-)Adresse)
- Schnittstellen (Verfügbarkeit; Datenbank- bzw. Programmier-Sprache(n); kostenpflichtig?; ...)
- ...

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	texas.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 / ...
<ul style="list-style-type: none">• 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
<ul style="list-style-type: none">• Personen-Karteien Personen-Register	Daten der Ortsämter Fahndungs-Karteien der Polizei usw. usf.
<ul style="list-style-type: none">• Register für Firmen und Institutionen	Firmen-Register (Handels-Register) Vereins-Register
<ul style="list-style-type: none">• Warenwirtschaftssysteme	SAP
<ul style="list-style-type: none">• Bilderdatenbanken	www.flickr.com www.instagram.com www.pixabay.com
<ul style="list-style-type: none">• Multimediatdatenbanken	www.youtube.com commons.wikimedia.org
<ul style="list-style-type: none">• Personendatenbank	Schulverwaltungsprogramme (WinSchool, Magellan, ...)
<ul style="list-style-type: none">• Soziale Medien sociale media	flickr.com facebook.com youtube.com twitter.com
<ul style="list-style-type: none">• Versions-Verwaltungen	github
<ul style="list-style-type: none">• Software-Sammlungen	sourceforge.org (Sammlung von Python-Programmen und – Bibliotheken)
<ul style="list-style-type: none">••	

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:

- spezielle Web-Seiten
- besondere Nutzer-Leistungen
- ungewöhnliche Darstellung der Daten
- ...

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 Scientific 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 Computer-Datenbank



Problem-Fragen für Selbstorganisiertes Lernen

Wie sind Datenbanken (Daten-Sammlungen) allgemein strukturiert?

Wie hat man früher größere Daten-Bestände organisiert? Wie konnte man darin sinnvoll arbeiten?

Welche ersten Datenbank-Systeme gab es für Personal-Computer?

Wie sind heutige Datenbanken (z.B. im Internet) aufgebaut?

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 Detektiven) oder Skizzen-Blöcke (z.B. von Künstlern) als spezielle Datenbanken gesehen werden. Die großen Archive, Galerien, Museen und Bibliotheken 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 also 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 reele Buch wird durch die Karteikarte als informatives 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.

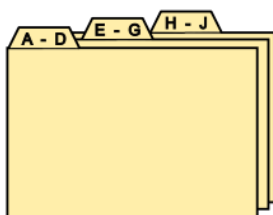
Als Nächstes interessieren uns die Informationen zum Buch. Das sind also die bibliographischen Angaben und der Standort in der Bibliothek.

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.

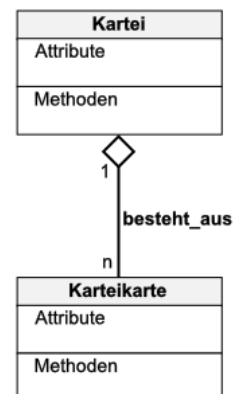
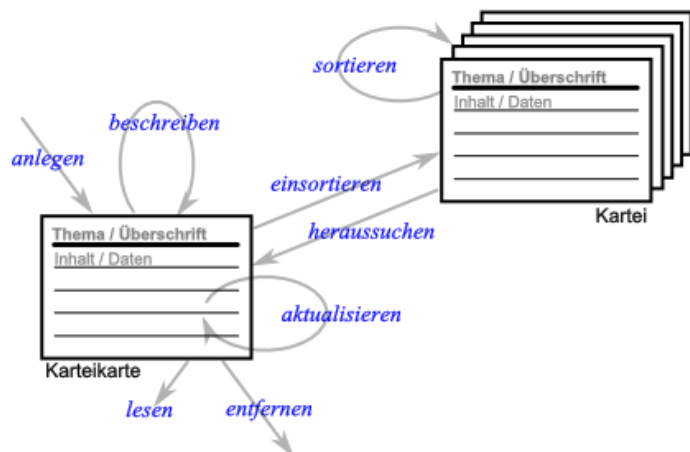
Wir sprechen allgemein von den **Methoden** zu den Objekten.

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 Kartei besitzt dann eben keine einzige Karteikarte. Um z.B. das Suchen / Finden etwas effektiver zu gestalten nutzte man Reiter zum Gruppieren der Karteikarten. Heute verwenden wir z.B. Indizes dafür.

Besonders anspruchsvolle Register haben an den Rändern der Karteikarten ein Lochungs- und Kerben-System. In der nebenstehenden Abbildung ist das z.B. der Anfangs-Buchstabe des Schlüsselwort's. Mit Hilfe der dort verschlüsselten Merkmale konnte man große Karten-Stapel recht effektiv nach den (nicht) "eingekerbten" Informationen durchsuchen.



Dazu wurde z.B. eine Stange durch den Karten-Stapel geschoben. Alle nicht gesuchten Merkmale blieben an der Stange hängen. Die Karten mit dem Suchmerkmal (in der Abb. Merkmal "D") fallen aus der Kartei raus.

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 verschiedenen Standorten wurde kaum realisiert, weil der organisatorische Aufwand enorm war. Eher führte man nur Teilregister in den einzelnen Fachbibliotheken.

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

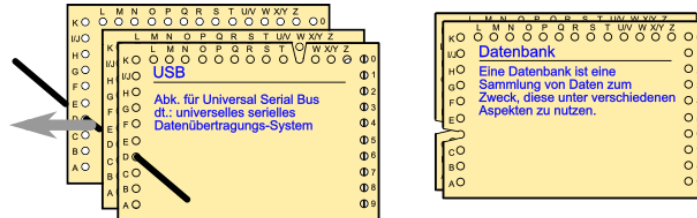
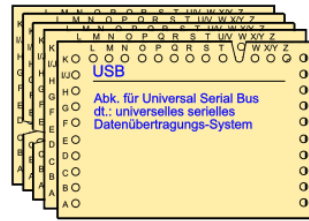
Für die Volkszählung 1890 in den USA entwickelte er eine Datenverarbeitungs-Maschine nach der Idee der programmierbaren Webstühle.

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

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

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 (International **B**usiness **M**achines).



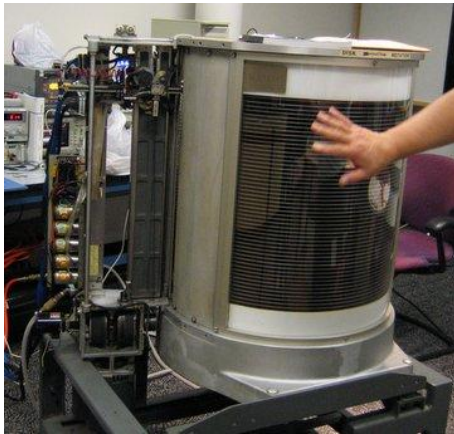
1	2	3	0	2	4	10	On	S	A	C	E	a	c	e	g	EB	SB	Ch	Sy	U	Sh	Hk	Br	Rm
2	2	4	1	3	E	15	Off	IS	B	D	F	b	d	f	h	SY	X	Fp	Cn	R	X	Al	Cg	Kg
3	0	0	0	0	W	20		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	1	1	1	1	0	25	A	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
B	2	2	2	2	5	30	B	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
C	3	3	3	3	0	3	C	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
D	4	4	4	4	1	4	D	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
E	5	5	5	5	2	C	E	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
F	6	6	6	6	A	D	F	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
G	7	7	7	7	B	E	G	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7
H	8	8	8	8	A	F	H	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8
I	9	9	9	9	b	c	I	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9

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



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))

Konrad ZUSE entwickelte mit seiner "Zuse 3" die erste elektrische Rechenmaschine (1941); basierend auf Relais; Größe ungefähr wie ein Kleiderschrank
Rechenkapazität ungefähr so groß wie ein heutiger einfacher Schul-Taschenrechner (damit ist kein wissenschaftlicher Taschen-Rechner gemeint)
wenig Permanent-Speicher



Festplatten-Speicher RAMAC 350
Q: en.wikipedia.org (vnunet.com)

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
Winchester-Platten

große Daten-Speicher waren Voraussetzung für die Entwicklung der Datenbank-Technologie im Bereich der elektrischen Daten-Verarbeitung
große Daten-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).



auswechselbare Plattenstapel;
hinten Band-Laufwerke
Q: en.wikipedia.org (Deutsche Fotothek)

Festplatten 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
dementsprechend verdoppelt sich die Speicher-Kapazität ungefähr alle 2 Jahre
derzeit geht Entwicklung noch schneller voran; als Regel ist das MOORESche Gesetz aber
gut verwendbar und eigentlich war es immer nur eine Regel (die so 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 so viele 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:

<ul style="list-style-type: none">• Welche Daten werden gespeichert? Welche Daten müssen gespeichert werden?	Auswahl notwendiger und sinnvoller Daten
<ul style="list-style-type: none">• Wie werden die Daten gespeichert?	Datenträger; permanent oder temporär
<ul style="list-style-type: none">• Wie kommt man an die gespeicherten Daten wieder heran? Wie können die Daten genutzt werden?	SQL als Datenbank-Anfragesprache
<ul style="list-style-type: none">• Wie arbeitet man sicher und effizient?	effektive und geprüfte Algorithmen

14122015 sind z.B. Daten

kann z.B. ein Kalender-Datum sein (14.12.2015)

können aber auch Messwerte (fortlaufend: 1 4 1 2 2 0 1 5 od. 14 12 20 15 od. gruppiert (14;12), (20,15)

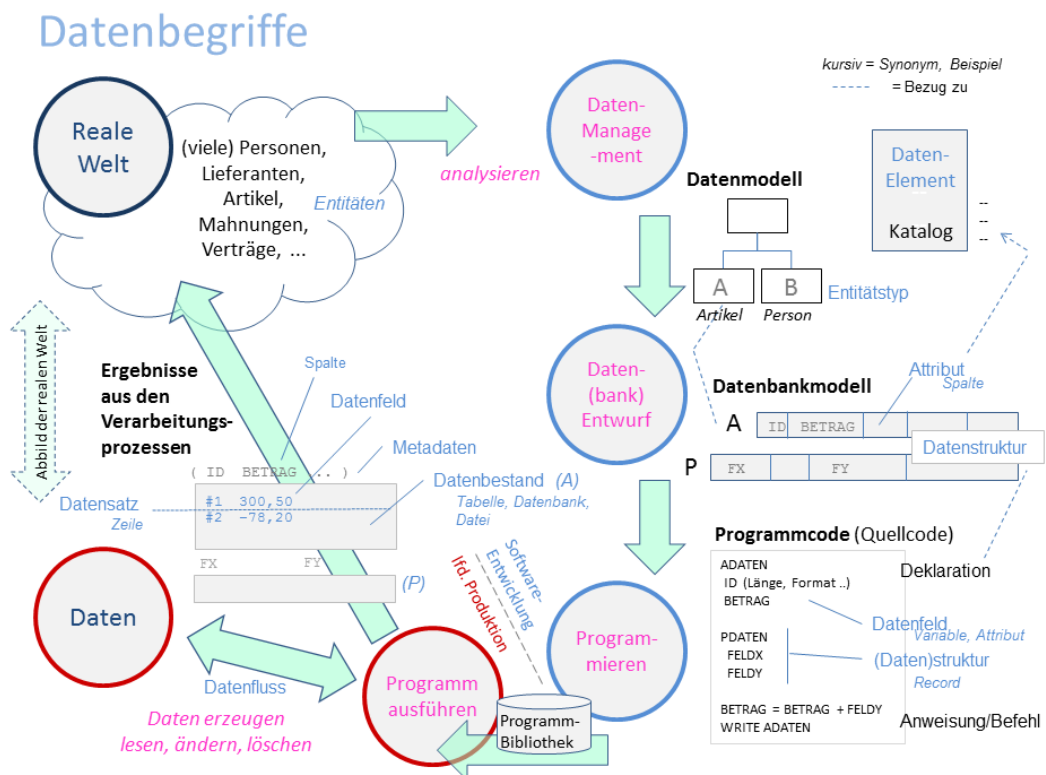
Daten lassen sich kategorisieren:

nach der Höhe der Komplexität:

• 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, ...)



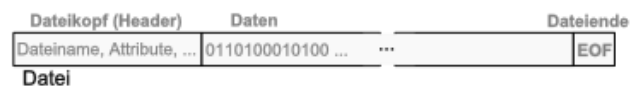
Entstehung von Daten und dazugehörige Begriffe
 Q: de.wikipedia.org (VÖRBY)

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.

Diese Form der Daten-Verarbeitung kann man ab den 60er Jahren des 20. Jahrhunderts in Filmen und Foto's beobachten: Bis heute ziehen sich die Band-Laufwerke und Festplatten durch die IT-Technik, wobei der Anteil aber extrem gesunken und die Technik natürlich leistungsfähiger geworden 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 erfolgt das praktisch direkt. Über einen magnetischen Schreib- und Lese-Kopf wurden die Daten auf einen mit Eisen beschichteten 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. Moderne Bandlaufwerke arbeiten mit Band-Kassetten, die größten zwischen Tonband- und Video-Kassetten haben.



IBM 360 Bandlaufwerke
Q: de.wikipedia.org (Erik Pitti)

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.



Floppy-Laufwerke 8, 5,25 und 3,5"
Q: de.wikipedia.org (Michael Holley)

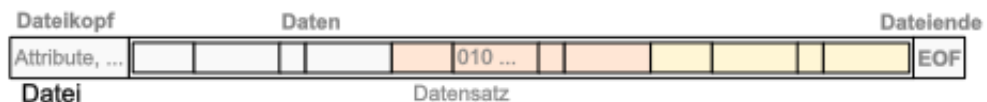
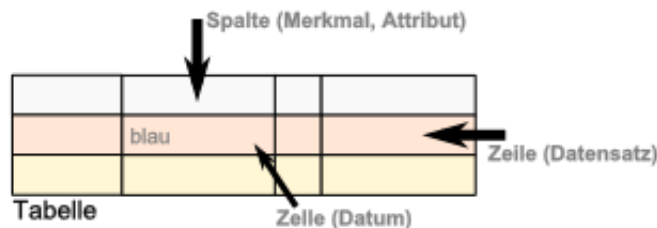
böse Fragen zwischendurch:

Wie sind die Daten auf einer CD bzw. DVD angeordnet?

Auf welcher der drei Floppy-Disketten konnte man die größte Datenmenge speichern? Recherchieren Sie nach!

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 unmerklich 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 Nutzung der Daten ohne das Erzeuger-Programm möglich.

Die Daten des einen Programms sind nicht mit denen eines anderen Programms kompatibel. So waren z.B. Zeitdaten in den Programmen ganz unterschiedliche gespeichert. In dem einem Programm wird das Geburtsdatum vielleicht in der Form 12.03.2001 gespeichert, während es in einem anderen (z.B. aus dem amerikanischen Bereich stammende) im Format 2001-03-12 gespeichert wird.

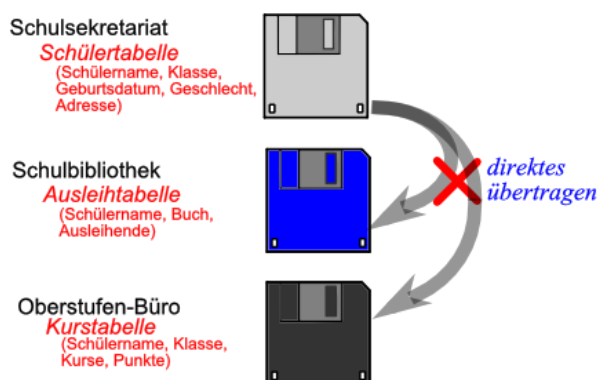


Selbst an einer Arbeitsstelle war nur eine begrenzte universelle und Bereichs-übergreifende Nutzung der Daten möglich.

Für die Programmierer bestand kein Bedarf für irgendwelche anderen Programme kompatible Daten zu erzeugen. Meist ging das 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



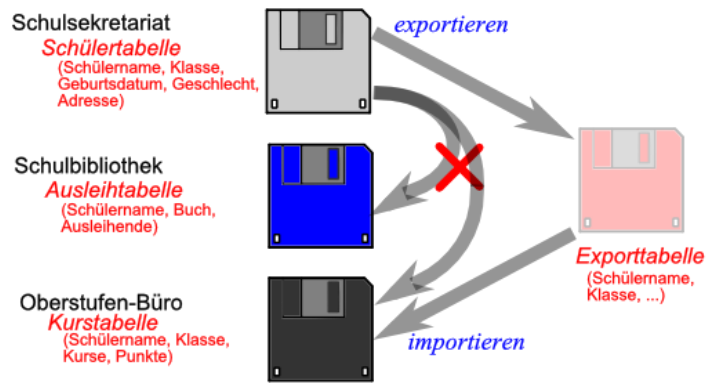
Die Daten müssen / mussten von Hand von einem Programm / Computer zum anderen transportiert werden. Der Datenaustausch erfolgte z.B. über Disketten.

Viele Importe scheiterten, weil die Datenformate (z.B. des Datum's) 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 gar nicht 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 bestimmte Datenstrukturen und -Prozesse verstehen, dann sind sie meist ähnlich (einfach), wie in ihrem Büro organisiert.

Nachteile

- Daten-Inkonsistenz 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 andere Programme 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 beim einfachen Anhängen von Daten an eine bestehende Datei)
- 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 ermöglichen. 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 vergrößerte 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
Arbeitsprinzip wird **O**n**I**ne **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-Systeme 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.

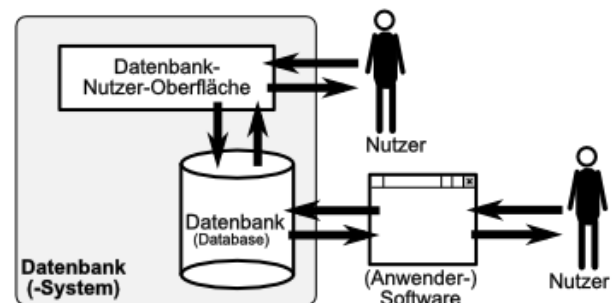
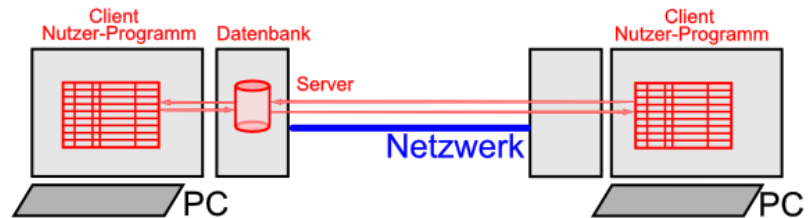
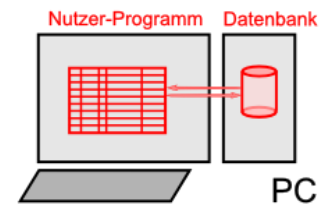
Zeitlich ist diese Entwicklung den 70er Jahren des letzten Jahrhunderts zuzuordnen. Hier kam es zu einer explosionsartigen Entwicklung relationaler Datenbanken. Ursächlich dafür waren theoretische Vorarbeiten von Edgar (Ted) Codd (1970) zu Konzeption von Datenbanken.

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 Datenauswertung war dann auch extern in speziellen Programmen (z.B. "Crystal Reports") möglich. Sie machten sehr komplexe Berichte möglich. Formulare waren für ein Nutzerbezogenes Daten-Handling gedacht.

Vorteile:

- Daten (Datenbank) noch von Hand händelbar (z.B. kopieren, sichern, ...)
- effektives Speichern und Manipulieren der Daten
- viele fertig integrierte 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
- Ressourcen-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.

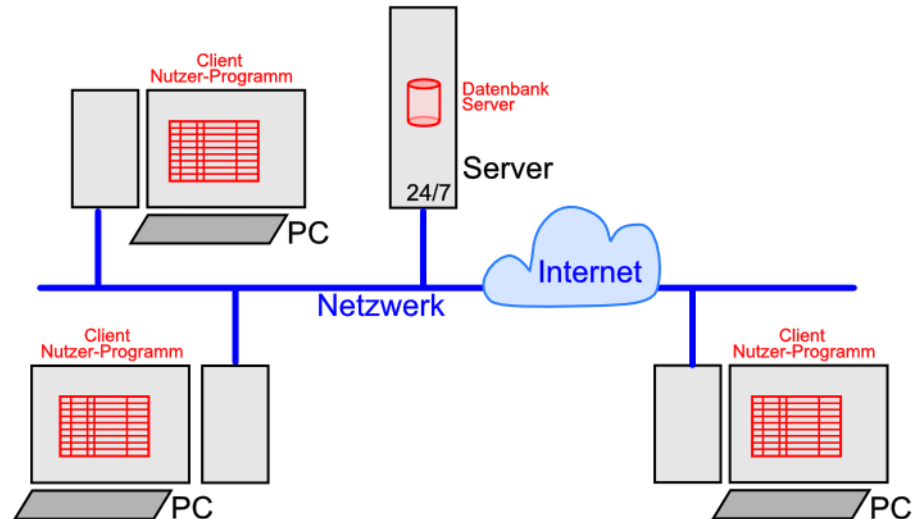
Beispiele für Dateisysteme:

Bezeichnung	Grob-Beschreibung	Verfügbarkeit / Realisierung im Betriebssystem	Verwendungsempfehlungen / 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 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 hochprofessionellen Bereich (bei speziellen Distributionen) 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
- tiktok.com
- itslearning.com
- ...
- Netflix
- de.wikipedia.org
- gelbe-seiten.de
- 11880.de
- google.com
- commons.wikimedia.org
- github.com
- sourceforge.org
- schule.cloud
- ...

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: 7'235 Exabyte), die wir täglich mehrfach kontaktieren und mit unseren Anfragen bombardieren.

Täglich wächst steigt 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- professionelle Bestreung der Datenbank durch speziellen (Datenbank-)Administrator
 - ...

Nachteile:

- System-unabhängiges Handling 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 überdimensioniert, 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 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

- 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 so etwas 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 von extern
- 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 gleichzeitig) 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 aus 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 relativ 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 Zugriffszeiten zufrieden geben, das ist aber günstiger, als alle Daten ständig z.B. parallel 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.

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 Big Data-Systemen werden z.B. in den Massen einlaufender Sensor-Daten (Stichwort IoT (Internet of Things)) nach auffälligen Mustern gesucht. Auch die Kunden-Karten-Systeme (z.B. Deutschland-Card, Payback, ...) nutzen die anfallenden Kaufdaten für weitere Daten-Analysen.

In den letzten Jahren kommen vermehrt Machine-Learning- und Künstliche Intelligenz-Systeme mit dazu.

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 unbekanntem 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

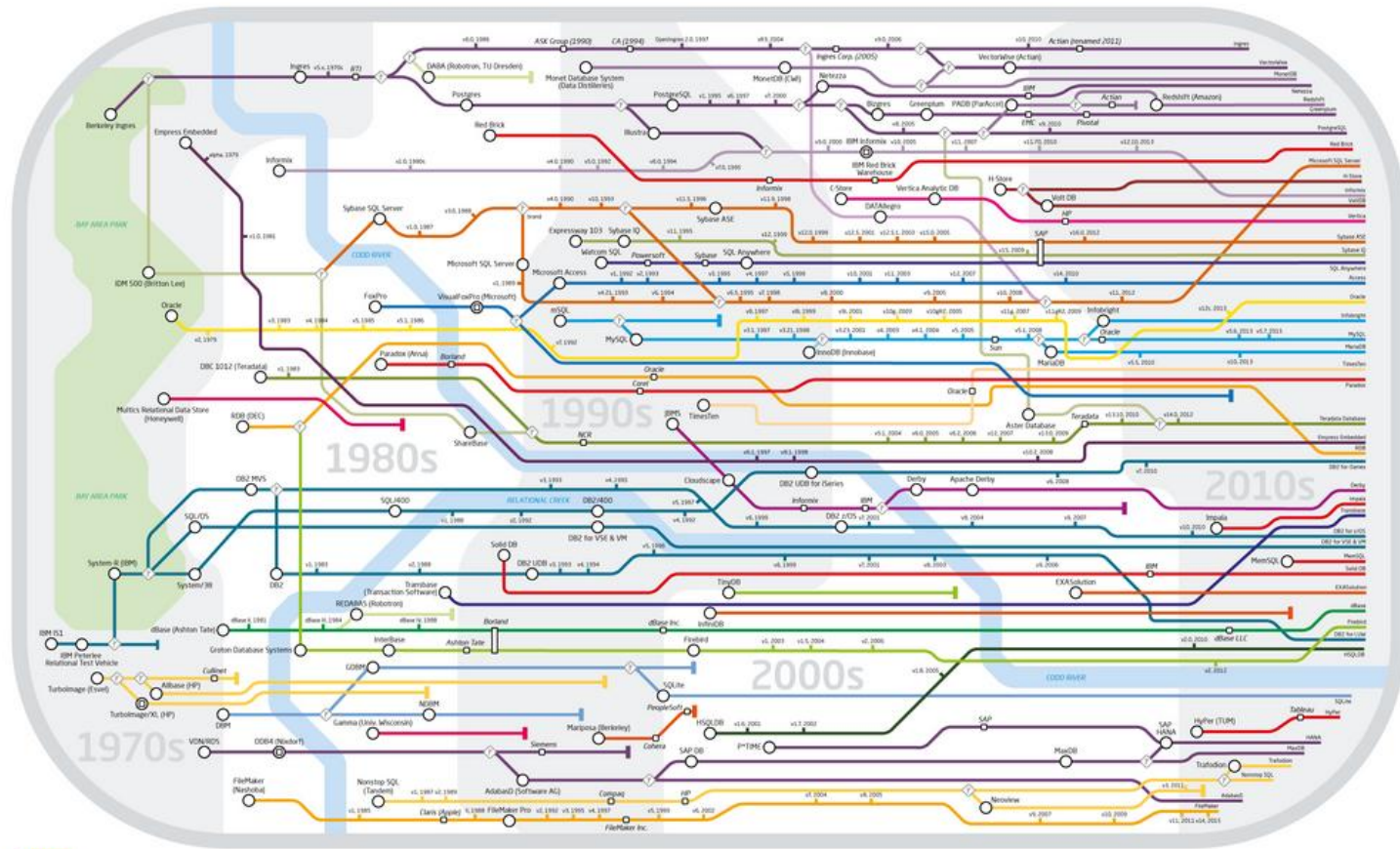
Q: u.a.: SQL-Kurs (NAUMANN; OpenHPI; 2013)

Zeitraum / Jahr	technischer Stand	Bemerkungen / Hinweise
	•	
	•	
60er Jahre	<ul style="list-style-type: none"> • Datenbank-Systeme auf der Basis von hierarchischen Modelle • Einbeziehung von Netzwerk-Technologien • informatische(r) Technologien / Trends / Stand <ul style="list-style-type: none"> ○ Zeiger (Pointer) auf Daten(-sätze) ○ schwache Trennung von physischer und konzeptioneller Ebene ○ navigierende Abfragesprachen 	<p>→ 2.3.1. hierarchisches Datenbank-Modell</p> <p>→ 2.3.3. netzwerkartiges Daten(bank)-Modell</p>
70er Jahre 80er Jahre	<ul style="list-style-type: none"> • relationale Datenbank-Systeme <ul style="list-style-type: none"> ○ Daten in Tabellen (Relationen) ○ ausgeprägte Trennung in physische und konzeptionelle Ebene (3-Ebenen-Konzept) ○ deklarative Abfragesprachen 	<p>→ 2.2. relationales Daten(bank)-Modell</p>
80er Jahre 90er Jahre	<ul style="list-style-type: none"> • viele Entwicklungen zu kleineren und größeren Datenbank-Systemen (Skalierung) • Objekt-orientierte Datenbank-Systeme • ... 	<p>→ 2.3.4. Objekt-orientiertes Daten(bank)-Modell</p>
2000er Jahre	<ul style="list-style-type: none"> • Spezialisierungen auf neue / speziellere Datentypen • ... 	<p>z.B.: → 2.3.5. Dokument(en)-orientiertes Daten(bank)-Modell</p>
2010er Jahre	<ul style="list-style-type: none"> • Web-basierte und skalierbare Datenbank-Systeme • NoSQL • neue Hardware (Nutzung von GPU's, Main-Memory, SSD's, auch virtuelle Server, ...) • ... 	<p>→ 7. Web-Datenbanken</p>
2020er Jahre	<ul style="list-style-type: none"> • Big Data, Data Science • Nutzung Künstlicher Intelligenz • Optimierung der Hardware (Energie-Effizienz, weniger Klima-schädlich) • ... 	

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

(kurze) Genealogy of relational Datenbank-Systeme

Genealogy of Relational Database Management Systems



Key to lines and symbols

- DBMS name (Company)
- ◻ Acquisition
- 📅 Versions
- ⏹ Discontinued
- ◊ Branch (intellectual and/or code)
- ✂ Crossing lines have no special semantics

Felix Naumann, Jana Bauckmann, Claudia Cramer, Jan-Peer Rudolph, Fabian Truchschütz
 Contact: Hasso Plattner Institut, University of Potsdam, felix.naumann@hpi.de
 Design: Alexander Savitskiy, GraphiQ Design, Hamburg
 Version 6.0 - October 2018
https://hpi.de/naumann/projects/rdbms_genealogy.html

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

Datenbank-Entwicklung

- Analyse
- Entwurf
- Realisierung

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. Claude Elwood 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.



Die Störungen sind eigentlich nicht Teil des Modell's, sind aber ein wichtiger reglementierender Faktor.

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 nicht-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 oder andere 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 uns dann Null.

Somit können wir feststellen:

Informationen müssen also eine Aussage enthalten (, also dürfen nicht leer (nichtsagend) 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 entsteht 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 – rotvioletten Stoff's 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.



Gemisch eines farbigen Salzes und Wasser bei verschiedenen Temperaturen

links: seltene (Ausgangs-)Situation mit hohem Informations-Gehalt
rechts: typische (Normal-)Situation (nach dem Auflösen) mit geringem Informations-Gehalt

Die völlig ungeordnete Verteilung des gelösten Stoffes hat praktisch keine 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 zuerst einmal auch Eingeben, Speichern und das Ausgeben mit ein.

Daten-Arten


- **unstrukturiert** 70 -90% der verfügbaren Daten
hoher Aufwand bei der Verarbeitung
z.B. BMP-Dateien
aber auch Texte, Video's, Ton-Aufnahmen
- **strukturiert** bestimmte Interpretations-Muster vorhanden (z.B. Kalender-Datum; Tabellen, ...)
(sehr) geringer Verarbeitungs-Aufwand
formale Struktur → Beschreibungs-Muster
z.B. CSV-Dateien
- **semistrukturiert** äußere Struktur vorhanden, im Inneren aber wieder unstrukturiert
mittlerer Verarbeitungs-Aufwand
z.B. HTML-, JSON- und XML-Dateien / -Daten

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 Betrachtung 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 übertragen werden.

Aufgaben:

- 1. Informieren Sie sich, wie Bitmap-Daten gespeichert werden. Warum handelt es sich hier um unstrukturierte Daten?*
- 2. Ordnen Sie JPG-Daten einer Daten-Art zu und begründen Sie diese Zuordnung!*

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 eine "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.) bezeichnen. 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.

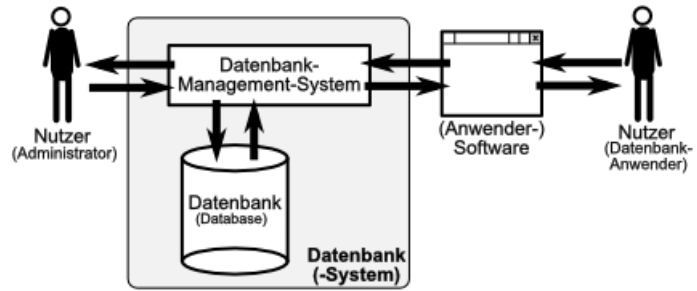
Vergegenwärtigen wir uns noch einmal wozu wir eigentlich Datenbanken brauchen:

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.

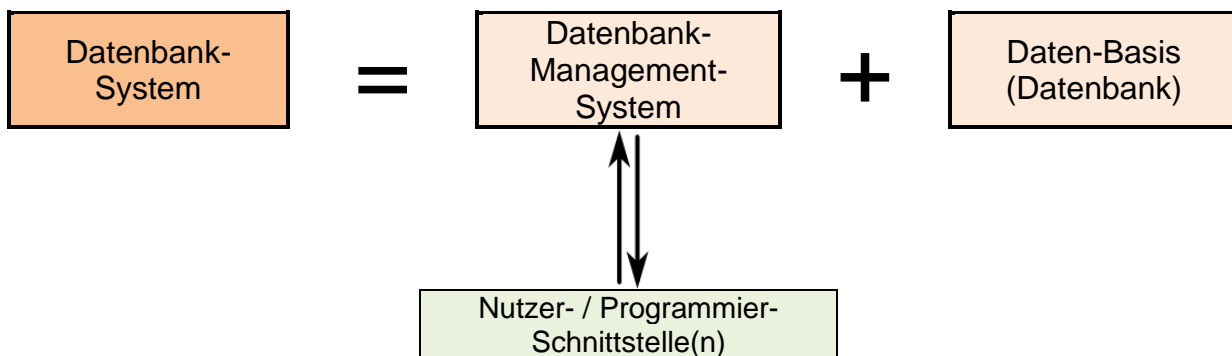


Alle Nutzer greifen auf die Daten-Basis nur über das Datenbank-Management-System (DBMS) zu.

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)) erfolgen immer über die Schnittstellen des Datenbank-Management-Systems.

Die Verfügbarkeit von SQL-Schnittstellen ist kein Muss – hat sich aber als ein Quasi-Standard herauskristallisiert.

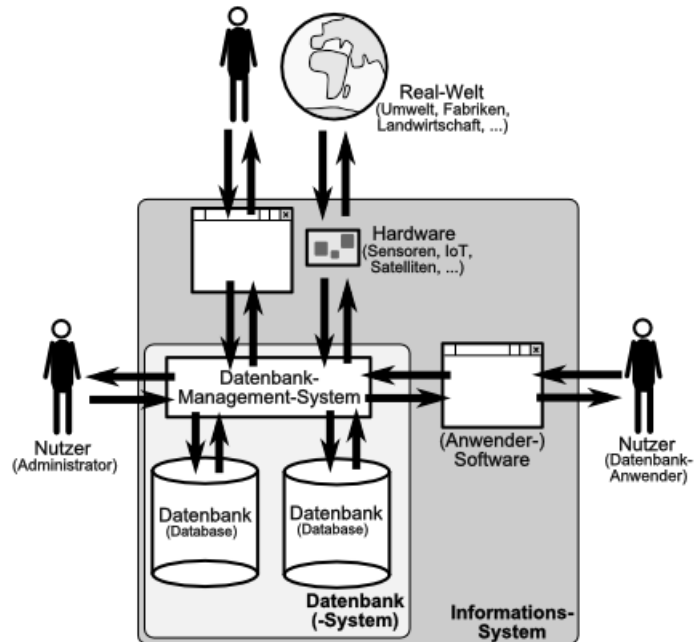
Geht man den nächsten Schritt nach außen, dann kommen neben den Anwenderprogrammen auch noch die automatisierten Systeme dazu. Hierzu zählen wir den gesamten Bereich der IoT (Internet of Things) oder die Daten-Erfassung mittels Satelliten. Sie stellen quasi eine direkte Verbindung von Real-Welt zu unserem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

Wir sprechen jetzt von Informationssystemen.

Besonders im Bereich der Produktion, des Handels und Militärs 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.

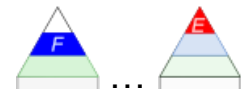
Moderne Datenbank-Systeme sind sehr komplex. Sie bestehen aus vielen Komponenten, die z.T. unabhängig voneinander funktionieren / verwaltet und weiter entwickelt werden. Einige Komponenten bzw. Aspekte sind in der umseitigen Tabelle kurz vorgestellt.



Probleme mit Schnittstellen und Abhängigkeiten (z.B. vom Betriebssystem) werden erst dann nach Außen auffällig sichtbar, wenn System-Ausfälle auftreten.

Haribo → Problem durch Wechsel des Warenwirtschaftssystems → große Liefer-Probleme

McDonald's (Bericht in der OZ)



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-Verwaltungssystem	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 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, dass 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-Management-System

Ein Datenbank-Management-System ist eine Software (im Wesentlichen Programme) 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 Systeme 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 Lotus)
- 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
- ...

Rang			DBMS	Datenbankmodell	Punkte		
Dez 2022	Nov 2022	Dez 2021			Dez 2022	Nov 2022	Dez 2021
1.	1.	1.	Oracle	Relational, Multi-Model	1250,31	+8,62	-31,43
2.	2.	2.	MySQL	Relational, Multi-Model	1199,40	-6,14	-6,64
3.	3.	3.	Microsoft SQL Server	Relational, Multi-Model	924,35	+11,84	-29,67
4.	4.	4.	PostgreSQL	Relational, Multi-Model	617,97	-5,18	+9,76
5.	5.	5.	MongoDB	Document, Multi-Model	469,33	-8,57	-15,34
6.	6.	6.	Redis	Key-value, Multi-Model	182,57	+0,52	+9,03
7.	8.	7.	IBM Db2	Relational, Multi-Model	146,61	-2,95	-20,56
8.	7.	8.	Elasticsearch	Suchmaschine, Multi-Model	144,93	-5,40	-12,80
9.	9.	10.	Microsoft Access	Relational	133,83	-1,20	+7,84
10.	10.	9.	SQLite	Relational	132,44	-2,19	+3,76
11.	12.	17.	Snowflake	Relational	114,77	+4,61	+43,73
12.	11.	11.	Cassandra	Wide column	114,65	-3,47	-4,55
13.	13.	12.	MariaDB	Relational, Multi-Model	100,93	-3,98	-3,43
14.	14.	13.	Splunk	Suchmaschine	90,79	-3,44	-3,53
15.	15.	16.	Amazon DynamoDB	Multi-Model	83,85	-1,55	+6,22
16.	16.	14.	Microsoft Azure SQL Database	Relational, Multi-Model	81,98	-1,68	-1,27
17.	17.	15.	Hive	Relational	77,89	-4,00	-4,04
18.	18.	18.	Teradata	Relational, Multi-Model	65,89	+0,66	-4,40
19.	19.		Databricks	Multi-Model	60,74	-0,15	
20.	20.	19.	Neo4j	Graph	57,34	+0,04	-0,70
21.	22.	24.	Google BigQuery	Relational	55,69	+1,55	+9,88
22.	21.	22.	FileMaker	Relational	53,86	-0,45	0,00
23.	23.	21.	SAP HANA	Relational, Multi-Model	50,20	-1,25	-4,38
24.	24.	20.	Solr	Suchmaschine, Multi-Model	48,26	-3,07	-9,46
25.	25.	23.	SAP Adaptive Server	Relational, Multi-Model	42,76	-0,83	-8,63
26.	26.	25.	HBase	Wide column	40,04	-0,38	-5,50
27.	27.	26.	Microsoft Azure Cosmos DB	Multi-Model	37,95	-1,80	-1,77
28.	29.	29.	InfluxDB	Time Series, Multi-Model	30,25	+0,28	+1,86
29.	28.	27.	PostGIS	Spatial DBMS, Multi-Model	29,20	-1,57	-3,24
30.	30.	28.	Couchbase	Document, Multi-Model	26,71	-1,91	-1,74
31.	31.	32.	Amazon Redshift	Relational	25,54	-1,50	+1,17

Rangier der DBMS
 Q: <https://db-engines.com/de/ranking>

Nach dem Arbeits-Prinzip unterscheidet man transaktionale Datenbanken und sogenannte Data Warehouse's.

Ein Data Warehouse stellt seinen Daten-Bestand relativ ungefiltert und breit zur Verfügung. Der Nutzer kann die Daten nach seinen eigenen Interessen kombinieren und verwerten. Data Warehouse-Datenbestände sind für das Maschinelle Lernen eine wichtige Grundlage.

Bei transaktionalen Datenbanken werden die Daten im Wesentlichen auf der Basis von Daten-Sätzen genutzt. Mit Hilfe von abgeschlossenen Transaktionen werden Datensätze gespeichert, gelesen und manipuliert. Die Datensätze können mit Datensätzen aus anderen Tabellen oder Datenbanken kombiniert und zu neuen Daten-Beständen zusammengefügt werden. Das könnte dann z.B. auch eine Data Warehouse sein.

	transaktionale Datenbanken	Data Warehouse's
Arbeits-Prinzip	Online Transaction Processing (OLTP)	Online Analytical Processing (OLAP)

Zweck	Transaktions-Verwaltung	Daten-Analyse, Entscheidungs-Hilfe
Merkmale	<ul style="list-style-type: none"> • viele Daten-Veränderungen • viele kurze Anfragen • mehr / viele Schreib-Operationen • kurze Reaktions-Zeiten • Ergebnisse landen wieder in der Datenbank (neue Datenbank-Situation) 	<ul style="list-style-type: none"> • 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 Abrechnungssystemen)
- Kunden-Management (z.B.: Ärzte, Händler, ...)
- Lohn-Abrechnung
- Daten-Analyse (Firmen-Beratung) (→ Big Data, ...)
- Versicherungen
- 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 spe-

ziellen 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 Bestandteilen.

Berichte sind heute meist schon bei den Anwender-Programmen jenseits der Schnittstelle zum Datenbank-Management-System angesiedelt. Programme, wie z.B. Cristal Reports, ermöglichen eine sehr komfortable Zusammenstellung von Auswertungen und Berichten.

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 so etwas realisieren.

Heute erfolgt der Zugriff auf Datenbank-Daten über die bereitgestellten Schnittstellen und speziell entwickelte Anwender-Software. Auch Web-Anwendungen bzw. App's sind durch die Nutzung der verfügbaren Schnittstellen zu Datenbanken möglich.

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 verfügbare 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-Systemen noch Datenstrom-Systeme dazu.

Eine (klassische) **Datenbank** arbeitet in etwa nach dem folgenden 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 unabhängig 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.

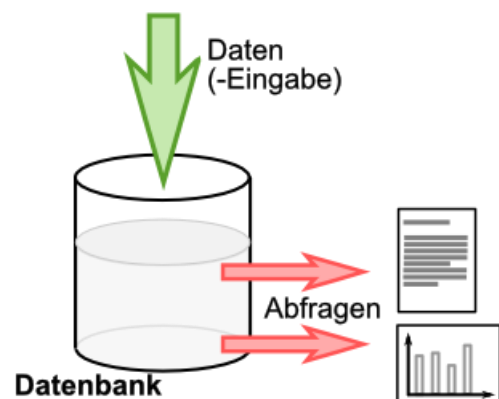
Aus Datenbanken lassen sich nur in sehr geringem Umfang dynamische Anzeigen (laufende Verkäufe, Lagerbestände) erzeugen. Man muss dazu immer eine Abfrage starten, die praktisch den gesamten Daten-Bestand durchforstet.

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 deshalb auch nicht wieder rekonstruieren.

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 zum 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 gleichzeitig (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 Administration** die Verwaltung der Nutzerrechte, Datensicherungen, Datenreparaturen, Updates, ... erfolgen von einer Stelle aus und werden von wenigen (hoch-)qualifizierten, (hoch-)berechtigten 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:

- **Persistenz** beschreibt die Dauerhaftigkeit und Verweildauer der Daten in einer Datenbank / in einem System (z.B. nach dem Abschalten der Strom-Versorgung)
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 Entwurfs 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/

Nachrichten und Informationen aus Literatur, Presse und Internet:

Ostseezeitung

McDonalds

.

/Q: IN: Ostsee-Zeitung –, S. /



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 redundant.

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 der ev. 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 auch wieder mehrfach gesichert und ev. auch wieder 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.

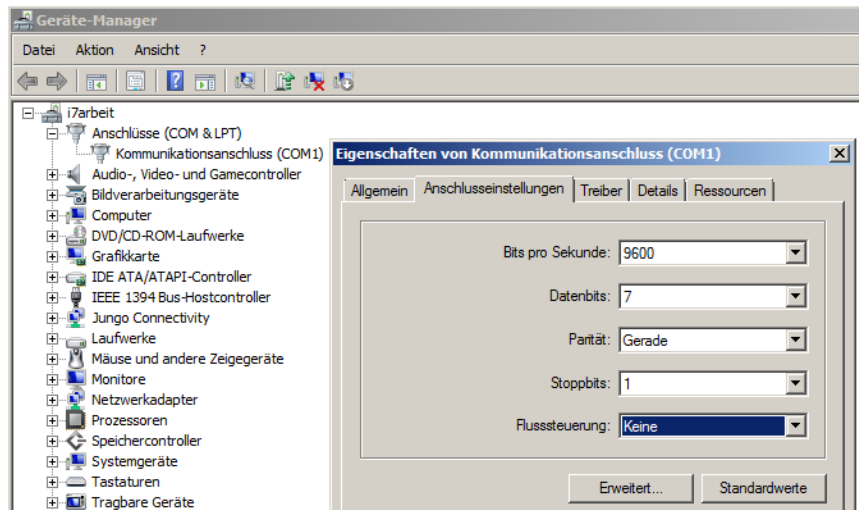
Daten-Bit-Sequenz	Anzahl Einsen	Paritäts-Bit (even-Parität)	Sendesequenz
0000000	0	0	00000000
0100110	3	1	01001101
0110000	2	0	01100000
1001100	3	1	10011001
1111111	7	1	11111111

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.



Die Redundanz kann gut für Fehler-Korrekturen genutzt werden. Zwar können wir mit einer einfachen Paritäts-Prüfung noch keinen Fehler genau lokalisieren, aber wir wissen u.U., dass ein Fehler vorliegt. Mit weiteren Sicherungs-Verfahren, wie z.B. dem CRC lassen sich dann Fehler schon genauer lokalisieren und manchmal sogar beheben.

allgemeine Berechnungs-Formel:

$$\text{Redundanz} = \frac{\text{Anzahl der übertragenen Bits}}{\text{Anzahl der Nutz-Bits}}$$

Exkurs: "Zauber"-Trick

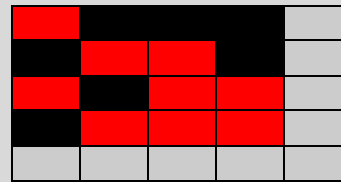
Gebraucht werden Karten, Bilder od.ä., die sich durch ein binäres Merkmal unterscheiden. Das könnten z.B. die Farbe im Französischen Skat-Blatt (also rot oder schwarz) oder Zahl bzw. Bild sein. Aber auch Vorder- und Rückseite sind eine Möglichkeit. Hier im Beispiel gehen wir von roten und blauen Karten aus.

Der Proband soll nun eine Reihe von z.B. 4 Karten legen. Die 5. Karte legt der "Zauberer" "zufällig" selbst dazu. Das kann auch später machen.

Nachdem der Proband 4 Reihen gelegt hat, ergänzt der Zauberer noch eine 5. Reihe. Z.B., um die Sache "noch komplizierter" zu machen.

Der Zauberer dreht sich dann um und lässt den Probanden eine beliebige Karte umdrehen.

Der Zauberer erkennt nach dem Umdrehen sehr schnell, welche Karte dies war.



Wie macht er das?

Der Zauberer nutzt das Paritäts-Prinzip. Er zählt z.B. die roten Karten einer Reihe (Zeile oder Spalte) durch und immer, wenn es eine ungerade Anzahl ist, ergänzt er eine rote Karte um die gerade Parität zu erreichen.

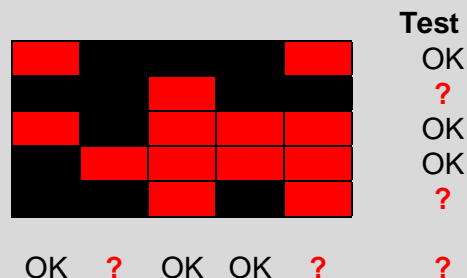
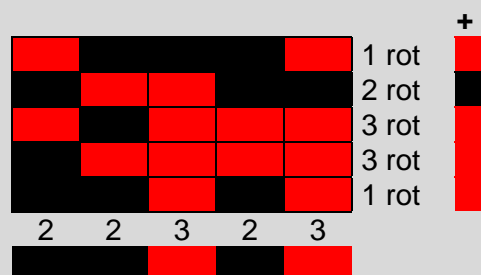
In diesem Muster kann jetzt der Proband eine beliebige Karte wechseln.

Die Veränderung lässt sich durch nun die fehlerhaften Paritäts-Karten leicht ermitteln.

Mit einem bisschen "Nachdenken" und "Spekulieren" kann das Ganze noch etwas spannender gemacht werden.

Um den Trick undurchsichtiger zu machen kann man natürlich auch viel mehr Karten nutzen. Mit einem Wechsel des Paritäts-System für Spalten und Zeilen verschleiert man den Trick noch zusätzlich.

Dann muss man sich nur die Bewertung der Ecke (Kreuzung der Paritäts-Reihen) merken. Eine weitere Möglichkeit ist das Legen eines Rechteck's als Grundform.

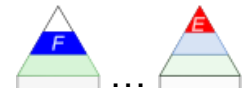


nach: GALLENBACHER "Abenteuer Informatik – IT zum Anfassen – von Routenplaner bis Online-Banking"; Springer Spektrum

Aufgaben:

1. *Funktioniert der "Zaubertrick" auch, wenn eine Karte aus den Paritäts-Reihen verändert wird?*
2. *Lassen sich mit dem Verfahren auch sicher zwei veränderte Karten erkennen? Begründen Sie Ihre Meinung ausführlich!*
3. *Könnte man den Zaubertrick so verändern, dass man auch zwei veränderte Karten entdeckt?*

Redundanz 1.Ordnung (informations-theoretische Redundanz)



bezieht sich auf die Zeichen einer Sprache

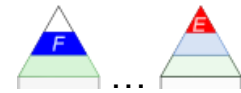
Bei Texten aus lateinischen Buchstaben ist die obere Hälfte informativer als die untere Hälfte. Es gelingt uns deutlich besser den oberen Text zu entziffern, als den unteren.

Texterkennung ist gar nicht schwer

Texterkennung ist gar nicht schwer

Definition(en): Redundanz 1. Ord.

Redundanz 2. Ordnung (sprach-wissenschaftliche Redundanz)



bezieht sich auf die Wörter einer Sprache
gemeint sind hier sinn-behaftete bzw. reguläre Ausdrücke einer Sprache

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

Laut einer Studie der Cambridge University, ist es egal in welcher Reihenfolge die Buchstaben in Wörtern vorkommen. Es ist nur wichtig, dass der erste und letzte Buchstabe an der richtigen Stelle sind. Der Rest kann total falsch sein und man kann es ohne Probleme lesen. Das ist, wie das menschliche Gehirn nicht jeden Buchstaben liest sondern das Wort als Ganzes.

Krasas oder?

originale Quelle dieses Textes nicht mehr identifizierbar

Obwohl er von der Buchstaben-Sequenz her, eher Daten-Müll ist, können wir ihn recht schnell lesen. Mit ein paar Übungen gelingt 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, dass Redundanzen immer System-abhängig sind.

Definition(en): Redundanz 2. Ord.

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 vorkommenden 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 beinhaltet 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 –Verarbeitungs-optimal zur Verfügung stehen.

Definition(en): Redundanz 3. Ord.

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	
1	1	1	1	1	1	1	

b)

1	0	0	1	1	1	0	
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
1	1	1	1	1	1		0

b)

1	0	0		1	1	0	1
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 (den heute technisch überholten) 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?
7. Berechnen Sie die Redundanz einer 1,44 MB-Diskette!

2.0.2.2. Daten-Integrität



lat.: integritas (Unversehrtheit, Reinheit, Unbescholtenheit) meint auch: Makellosigkeit, Unbestechlichkeit, Unverletzlichkeit

beschreibt Korrektheit der Daten und korrekte Funktion des Systems
Schutz vor unberechtigter Löschung und Veränderung bei der Übertragung

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 (z.B. Datum's-Zeit-Format) zur einem Attribut an

neben **Verfügbarkeit** (→) und **Vertraulichkeit** (→) eines der drei Zielrichtungen der Informations-Sicherheit

Überprüfen z.B. mit einer mitgelieferten Prüfsumme
sind z.B. eine eventuell 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.
Unter Integrität versteht man den gültigen semantischen Zustand einer Datenbank, bezogen auf den dargestellten / modellierten Ausschnitt aus der realen Welt.

statische Integritäts-Bedingungen

- Domainen-Integrität (Domain Integrity)
- (Entity Integrity)
- referenzielle Integrität (Referential Integrity)

dynamische Integritäts-Bedingungen

- Einschränkungen bei Zustands-Übergängen (der Datenbank)
- Trigger

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
wenn zwei gleiche Schlüssel in einer Tabelle vorkommen, dann ist nicht mehr erkennbar,
welcher der beiden Datensätze der gültige ist

Gegenstands-Integrität

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

Integritäts-Ebenen

<ul style="list-style-type: none">• Wertebereichs-Integrität Integrität auf Datenfeld-Ebene	alle Attributwerte liegen im gültigen Wertebereich des Attributes (Datenfeldes) Attributwert liegt innerhalb der Domäne
<ul style="list-style-type: none">• Datensatz-Integrität Integrität auf Datensatz-Ebene	hier setzt Normalisierung (→) an
<ul style="list-style-type: none">• referenzielle Integrität Integrität auf der Beziehungs-Ebene	die Beziehungen zwischen Tabellen besteht auch noch nach dem Manipulieren der Tabellen es existieren keine Verweise auf einen schon gelöschten Datensatz in einer anderen Tabelle mögliche Anomalien: <ul style="list-style-type: none">• Einfüge-Anomalie• Änderungs-Anomalie• Löschen-Anomalie

referenzielle Integrität bedeutet, dass ein Fremdschlüssel entweder einen tatsächlich existierenden Schlüssel oder einen Null-Wert enthält

Voraussetzungen für eine Beziehung mit referentieller Integrität:

- die zu verbindenden Tabellen (Master- und Detail-Tabelle) befinden sich in der gleichen Datenbank
- beide Tabellen beinhalten ein Attribut mit gleichem Datentyp, über das 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 (Wertebereiche))
verantwortlich: Anwendungs-Programmierer, Datenbank-Adminsistrator, Hersteller des DBS

referentielle Integrität (Beziehungs-Integrität; RI; R.I.)

meint die Aufrechterhaltung von Datenbeziehungen nach / bei (referentiellen) Löschkaktionen ein Verweis von einer Tabelle (→ Fremdschlüssel) muss immer auf einen existierenden Quelleintrag (→ Primärschlüssel) in einer anderen (Ziel-)Tabelle verweisen
fehlt der Eintrag in der Ziel-Tabelle, dann ist die Integrität der Datenbank verletzt

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

Verfahren:

besteht aus zwei Teilen:

- ein **neuer Datensatz** mit einem integrierten Verweis kann nur dann in die DB eingebaut 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); SONST: muss der referenzierte Eintrag in der (ziel-)Tabelle zuerst erzeugt werden
- 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 Voraussetzungen, 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 Fremdschlüsseln

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 Integrität des (eingegebenen) Datum's zu prüfen!*
- 2. In einer Tabelle sind Name, Vorname und Wohnort (codiert als PLZ) gespeichert. Eine zweite Tabelle enthält die PLZ und die zugehörigen Ortsnamen usw.*
 - a) Stellen Sie die Tabellen und die Beziehungen zwischen diesen schematisch dar!*
 - b) Füllen Sie die Tabellen mit jeweils mindesten 5 Datensätzen! Bei der zweiten Tabelle (Orte) sollten einige PLZ in der ersten Tabelle (Personen) benutzt und andere unbenutzt sein!*
 - c) Prüfen Sie welche Datensätze der Personen-Tabelle über verlässliche Daten verfügen!*
 - d) Prüfen Sie, welche Daten der Orts-Tabelle sicher gelöscht werden könnten!*
 - e) Testen Sie, welche Datensätze der Personen-Tabelle ohne Probleme gelöscht werden könnten!*
 - f) Stellen Sie Regeln für einen (Fremd-)Nutzer Ihrer Tabellen auf, damit Datensätze aus der PLZ-Tabelle dieser sicher neue Datensätze in die Personen aufnehmen kann!*

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 noch 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 den Realitäts-Bezug, die Wahrhaftigkeit der Daten bezüglich den abzubildenden Realwelt- / Miniwelt-Objekten

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.

Konsistenz bedeutet, dass in einem System mit redundanter (auf mehreren Knoten verteilten) Datenspeicherung alle Replika den gleichen Stand / Wert haben.

Sind Daten in einem System nicht konsistent, 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 "Bad Schönquellen" 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 schreibt 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 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.

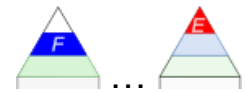
Konsistenz in relationalen Datenbanken

- **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** beinhaltet 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 Konsistenz-Bereiche erfüllt sind. Ist ein Bereich nicht konsistent, dann ist auch die Datenbank inkonsistent.

Aufgaben:

1. ***Wählen Sie einen etwas komplizierteren Ortsnamen aus Ihrer Heimat-Gegend! Überlegen Sie sich verschiedene Schreibweisen / Notierungen / Eingabe-Fehler mit denen der Ort immer noch als "richtig" durchgehen würde! Wieviele Varianten finden Sie?***
2. ***Warum ist die exakte Führung von referenzierten Tabellen eigentlich wichtig? Kann man nicht immer schnell die Daten korrigieren, sie wirken sich doch dann auch gleich auf die Datenbank aus?***



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

<ul style="list-style-type: none"> • Client-bezogene Konsistenz (Client-centric Consistency) 	
<ul style="list-style-type: none"> ○ monotone Lese-Konsistenz (Monotonic Read Consistency) 	aufeinander folgende Lese-Vorgänge müssen sich immer auf die gleiche oder eine neuere Version beziehen
<ul style="list-style-type: none"> ○ monotone Schreib-Konsistenz (Monotonic Write Consistency) 	aufeinander folgende Schreib-Vorgänge müssen immer in der gleichen Reihenfolge durchgeführt werden
<ul style="list-style-type: none"> ○ Konsistenz (Read Your Writes Consistency) 	nach dem Schreiben einer Version beschränkt sich das Lesen auf die gleiche oder eine neuere Version
<ul style="list-style-type: none"> ○ Konsistenz (Write Follows Reads Consistency) 	gelesene und dann geänderte Versionen werden nur auf Replika übertragen, wenn diese mindestens die gleiche Version beinhalten
<ul style="list-style-type: none"> • Daten-bezogene Konsistenz (Data-centric Consistency) 	
<ul style="list-style-type: none"> ○ kausale / logische Konsistenz (Causal Consistency) 	Operations-Folgen auf dem einen Replikat müssen / werden auf den anderen Replikaten in der gleichen Reihenfolge abgearbeitet
<ul style="list-style-type: none"> ○ sequentielle Konsistenz (Sequential Consistency) 	betrachtet die exakte zeitliche Folge von Prozessen
<ul style="list-style-type: none"> ○ (Linearizability) 	beinhaltet sowohl die sequenzielle – als auch die zeitliche – Abfolge bezieht auf nebenläufige Prozesse / System / Daten-Veränderungen

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.

Mittlerweise wird die Authentizität zu den übergeordneten Zielen in der IT-Branche gezählt. Alle anderen Ziele werden sinnlos, wenn nicht die Verlässlichkeit gesichert 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 Schlüsseln** (echte Hardware od. moderne kryptographische) oder die **persönliche Anwesenheit** bzw. Personen-Identität (z.B. biometrische Merkmale) als Grundlage für eine Authentifizierung geeignet.

Aber Achtung! Eine authentifizierte Nachricht muss nicht zwangsläufig wahr und / oder konsistent sein!

Authentifizierungs-Grundlage	Information	Besitz	Gegenwart / Körper-Merkmale
Beispiele	<ul style="list-style-type: none"> • Passwort • PIN • Sicherheitsfrage • Zero-Knowledge-Beweis (Kenntnis-freier Beweis) 	<ul style="list-style-type: none"> • Chip-Karte • Zertifikat • TAN-Liste • USB-Stick (als Passwort-Tresor) • SIM-Karte • RFID-Karte 	<ul style="list-style-type: none"> • Finger-Abdruck • Gesichts-Erkennung • Tipp-Verhalten • Stimm-Erkennung • Iris-Erkennung • Handschrift-Erkennung • Handflächen- und Handlinien-Erkennung • Hardware-Nummer od.ä. • Netzwerk-Adresse (MAC)
Vorteile	<ul style="list-style-type: none"> • einfach zu benutzen • keine technischen Mittel notwendig • leicht transportierbar • 	<ul style="list-style-type: none"> • schwerer zu duplizieren • kann als Objekt geschützt werden • ersetzbar bzw. austauschbar • 	<ul style="list-style-type: none"> • ist immer dabei • nicht übertragbar oder duplizierbar •
Nachteile	<ul style="list-style-type: none"> • kann vergessen werden • kann verteilt oder verraten werden • lässt sich ev. erraten oder ermitteln (Brute-Force-Angriff) • Nutzung schwer zu kontrollieren • 	<ul style="list-style-type: none"> • Verlust, Diebstahl, Weitergabe möglich • Verlust des Objektes bedeutet auch Verlust der Authentizität • Erstellung notwendig und z.T. aufwändig • 	<ul style="list-style-type: none"> • ist öffentliche Information • Hardware und eigentliche Prüfung aufwändig • auch Fälschliche Akzeptanz 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. Üblicherweise kombiniert man unterschiedliche Authentifizierungs-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 Einloggen 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-Verfahren's wird eine bestimmte TAN aus einer Liste angefordert oder eine TAN per SMS an ein Mobil-Telefon gesendet.

Heute typische 2-Faktor-Authorisierungen verwenden z.B. ein Login mit Passwort und dazu einen separat zugesandten oder mit einem Gerät erzeugten PIN (Kombination Wissen und externer Weg bzw. Wissen und Hardware).

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 erforderlich 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! Recherchieren Sie ev. nach!*

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 (Privacy), Nicht-Abstreitbarkeit (Authentizität, non repudation) und Verlässlichkeit (Verbindlichkeit, Reliability) und ev. auch Anonymität (Anonymity).

Daten in seiner Datenbank sollen vor unberechtigten Zugriffen geschützt werden. Niemand möchte sensible Daten frei im Internet verfügbar sehen.

Auch allgemein ist es nicht gewünscht, dass die Daten beim Übertragen durch das Internet von irgendjemand mitgelesen werden.

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 Administration 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 Zugriffskontrollen (über ODBC)
- Daten-Verlust einer Datenbank
- Verlust der Daten-Integrität und -Konsistenz

- **vorsätzliche Handlungen**

- unberechtigte 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



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, das 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 abgelehnt.

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.

Der Anfänger, der glaubt besonders schnell und effektiv zu arbeiten, beginnt gleich mit dem Eingeben der Daten in den Computer.

In der Praxis hat die folgende Sequenz von Praktiken 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 mal andere Implementierungs-Möglichkeiten anzuschauen. Interessant sind dabei vielmehr die Gemeinsamkeiten (→ SQL: [5. SQL – die Datenbank-Sprache](#)) und weniger die Unterschiede. Wer universelle Systeme anstrebt, setzt dann auf die Gemeinsamkeiten. Die Unterschiede sind nur für Spezial-Lösungen interessant und ev. zu Befriedigung des eigenen Ego's ("Ich kann ein ganz seltenes, schwer zu verstehende Datenbanksystem programmieren / administrieren.").

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

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)
• hierarchisches Modell	Datenbank-Systeme mit hierrarchischen oder Baumartigen 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 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 **relationale Modell** dominierend. Die anderen Modelle sind aber für das Verständnis von informatischen Strukturen und den Konzepten moderner Datenbanken ebenfalls interessant und besprechenswert.

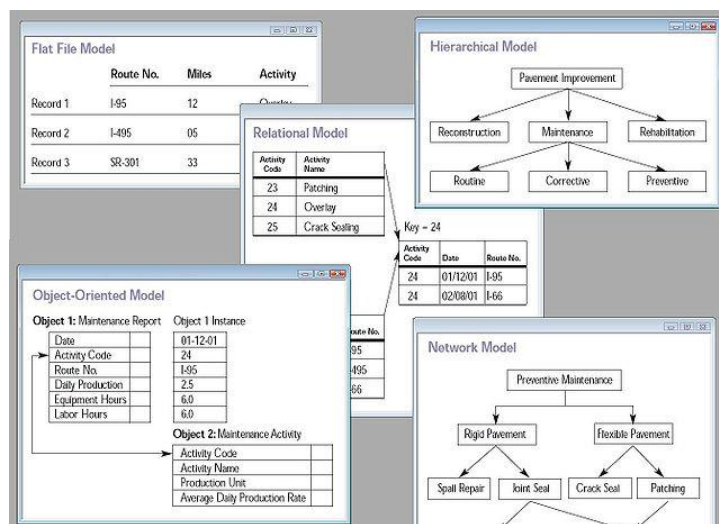
Definition(en): Datenbank-Modell
Ein Datenbank-Modell ist eine Sammlung von Strukturen und deren Beziehungen untereinander zur Charakterisierung eines Datenbank-Systems.
Ein Datenbank-Modell ist die Beschreibung der Daten und ihrer Strukturen in einem Datenbank-System.
Ein Datenbank-Modell ist die Darstellung / 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 Augenmerk 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. Daten-Strukturen – wenn sie denn einmal festgelegt sind – stellen ein sehr stabiles, schwer zu veränderndes Element dar. 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 vorhandenen Daten sind möglich. In der Branche wird das durch den Grundsatz "Data is stable – functions are not." (Daten sind stabil – Funktionen sind es nicht.) ausgedrückt.

Dieses Prinzip wird schon durch die von CODD definierten Eigenschaften(-Ebenen) eines Datenbank-Modell wiedergegeben.

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

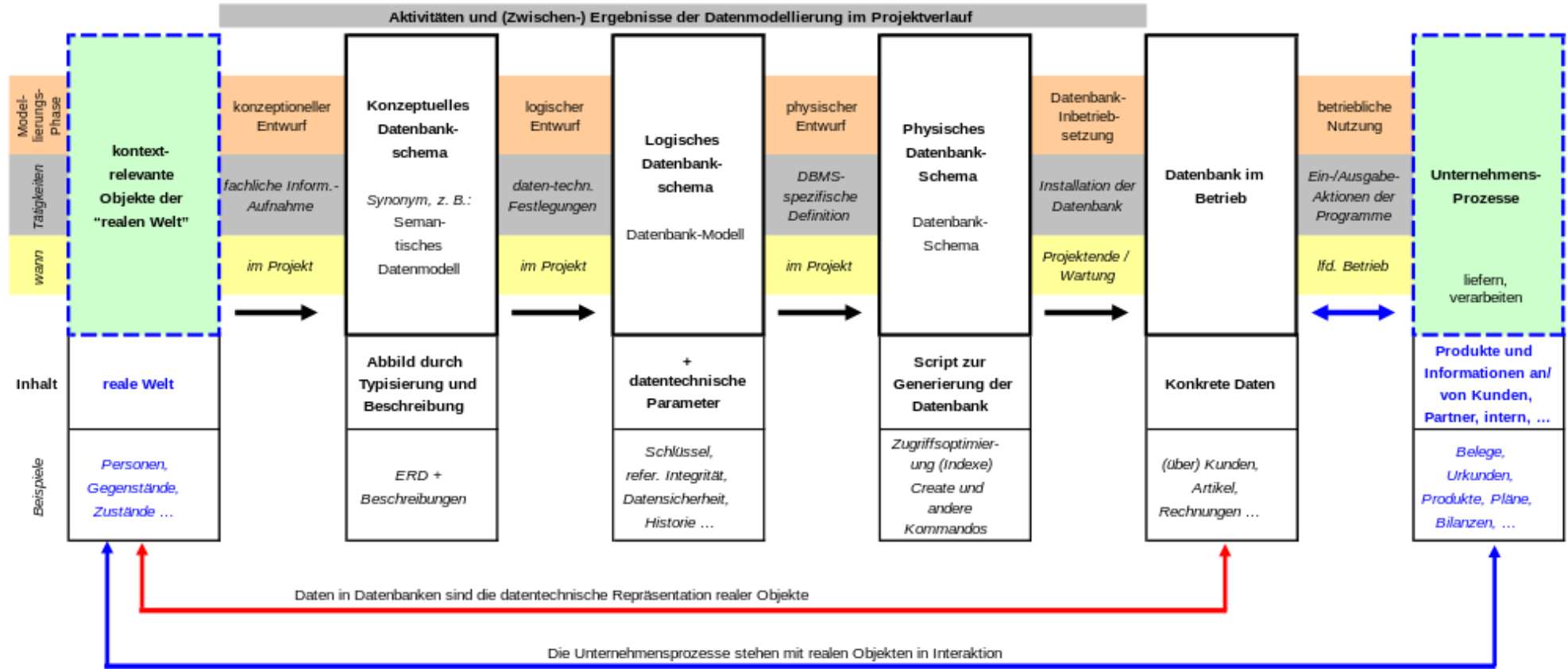
- 1. generische Daten-Struktur** beschreibt die Struktur der Datenbank; generisch, weil die Relationen und Attribute frei wählbar sind
- 2. generische Operatoren** Menge von Operatoren bezüglich der benutzten Daten-Struktur; dienen zum Einfügen, Ändern, Abfragen und Verknüpfen der Daten
- 3. Integritäts-Bedingungen**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

Definition(en): Daten-Modellierung

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

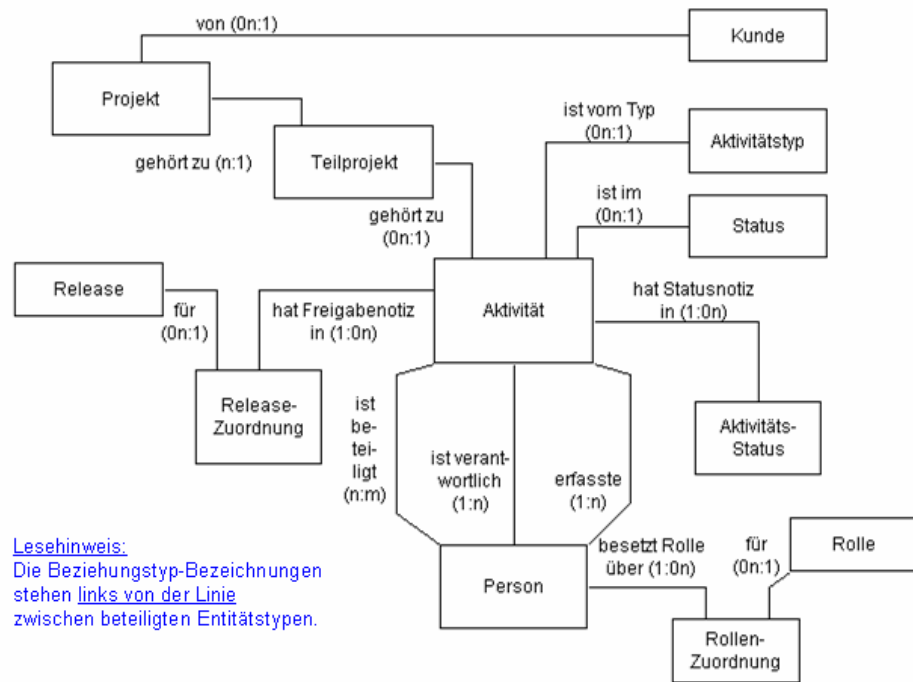
Als Fluß-Diagramm kann man sich die Erstellung einer Datenbank aus einem Entwurf in etwa so vorstellen:

Datenmodellieren: Entwicklung von der fachlichen, implementierungsunabhängigen Konzeption bis zur Datenbank



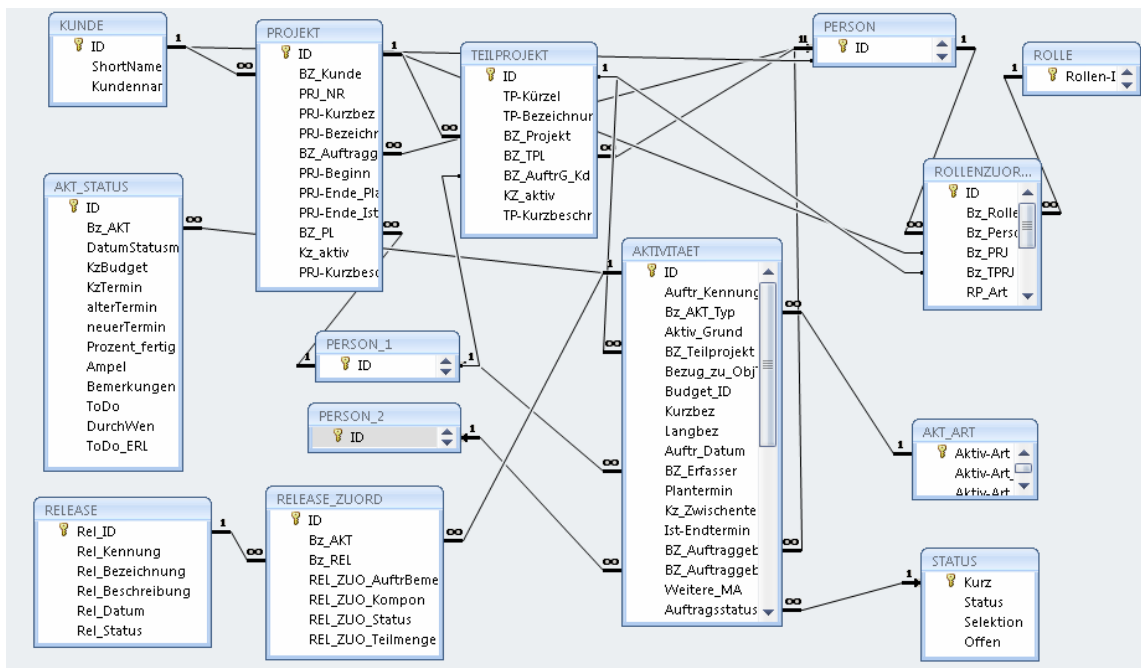
Daten-Modellierung - vom fachlichen Entwurf zur Datenbank
 Q: de.wikipedia.org (VÖRBY), gemeinfrei

Datenmodell für eine Projektmanagement-Auftragsverwaltung



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



Damit die Real-Objekte sinnvoll in einer Datenbank repräsentiert werden, müssen deren Daten und Daten-Verbindungen (Zusammenhänge) analysiert werden. Aus der Daten-Analyse können dann Strukturen abgeleitet werden. Diese sind dann wiederum die Basis für eine passende Datenbank.

Definition(en): Daten-Modell
Das Daten-Modell ist die Definition der zu bearbeitenden / verarbeitenden Daten eines Anwendungsbereiches.
Ein Daten-Modell ist die Beschreibung der Struktur von Daten.
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,

Ausgangspunkt war 4-Ebenen-Modell DIAM von Michael SENKO (1973)
DIAM steht für Data Independent Accessing Model

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

• entity set model (Gegenstands-/Objekt-Satz-Modell)	logische und anwenderneutrale Datenstrukturen (Mengen von Objekten, die vom DBMS verwaltet werden)
• string model (Strang-/Pfad-Modell)	logische Zugriffspfade (z.B. Indexe, Hash-Tabellen, ...)
• encoding model (Kodierungs-Modell)	Speicherungsstrukturen; physischer Zugang zu den Daten und Datenstrukturen (Abbildung der Daten in speziellen Strukturen (Baum, Tabelle, ...))
• physical device model (((physikalisches) Geräte-Modell)	Speicherordnungsstrukturen; Zuordnung der Daten ins das Dateisystem und den Datenträgern

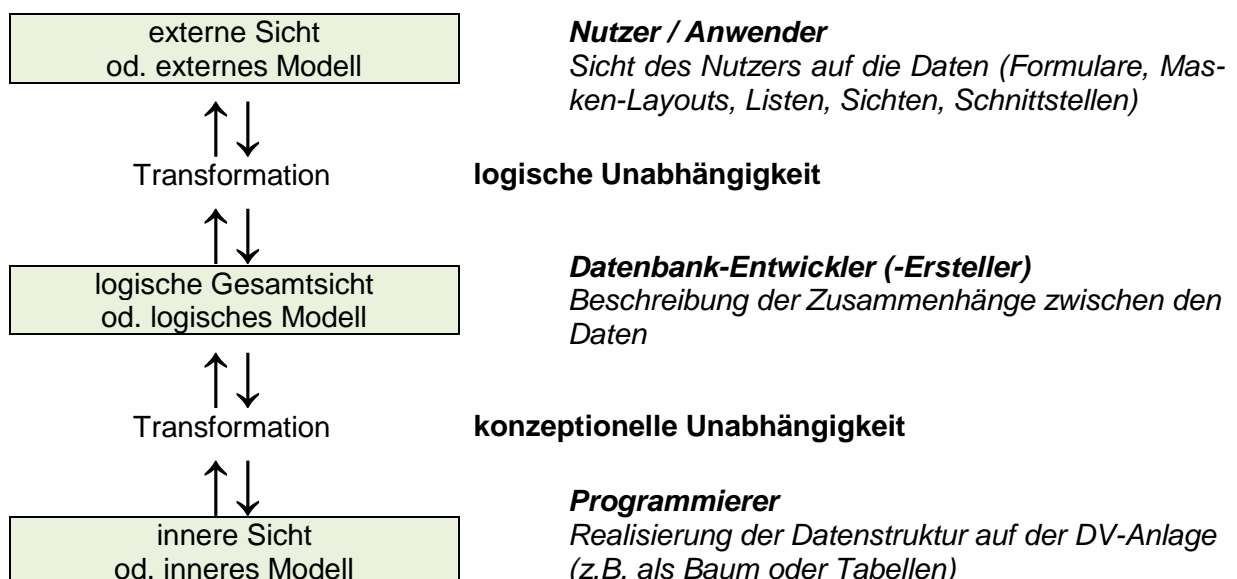
die Begriffe Ebene, Modell, Schicht und Sicht werden praktisch äquivalent benutzt von Ebenen und Schichten spricht man eher, wenn es um das Objekt geht soll mehr die Arbeit von uns Menschen mit der Ebene / Schicht betont werden, dann wird der Begriff Sicht bevorzugt betrachtet man die Sachverhalte eher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

wegen der Verwechslungsgefahr mit dem 3-Ebenen-Modell (zur Beschreibung menschlichen Verhaltens in Mensch-Maschine-Systemen, 1980) von Jens RASMUSSEN wird das hier gemeinte 3-Ebenen-Modell 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



je nach Ebene gibt es unterschiedliche Verantwortliche:

- Anwendungs-Administrator
- Unternehmens-Administrator
- Datenbank-Administrator

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-Zusammenhänge 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 innerhalb 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 (Daten-Typ) und wie die Daten miteinander verbunden sind (Relationship)

Definition von Tabellen, Optimierung der Tabellen (→ Normalisierung)

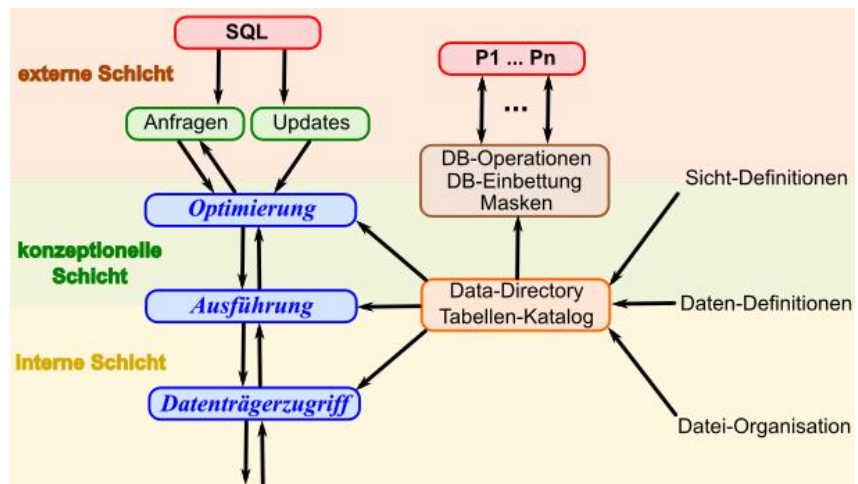
Inhalt und Zweck der Datenbank bestimmen das Arbeiten in diesem Bereich

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 Aufteilung in Schichten 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

irgend wann 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 Anwendungen / Dialoge / Abfragen / ..., die sich eben mit den neu(artig)en Daten beschäftigen müssen dann (unabhängig) angepasst werden

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

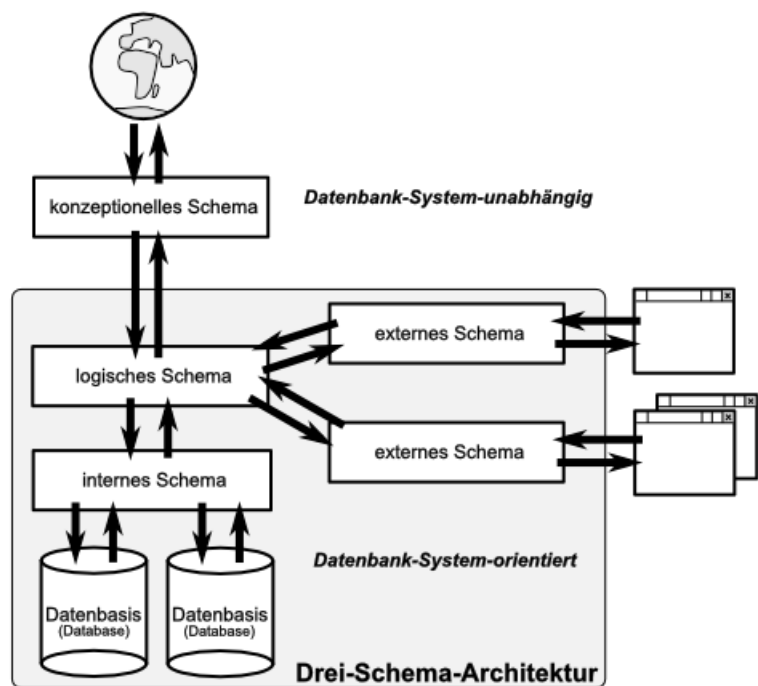
Das **Drei-Schema-Modell** ist die Erweiterung des klassischen Schichten-Modell's. Durch die neue Drei-Schema-Architektur sind zusätzliche Betrachtungen und Sichtweisen auf Datenbank-Systeme möglich.

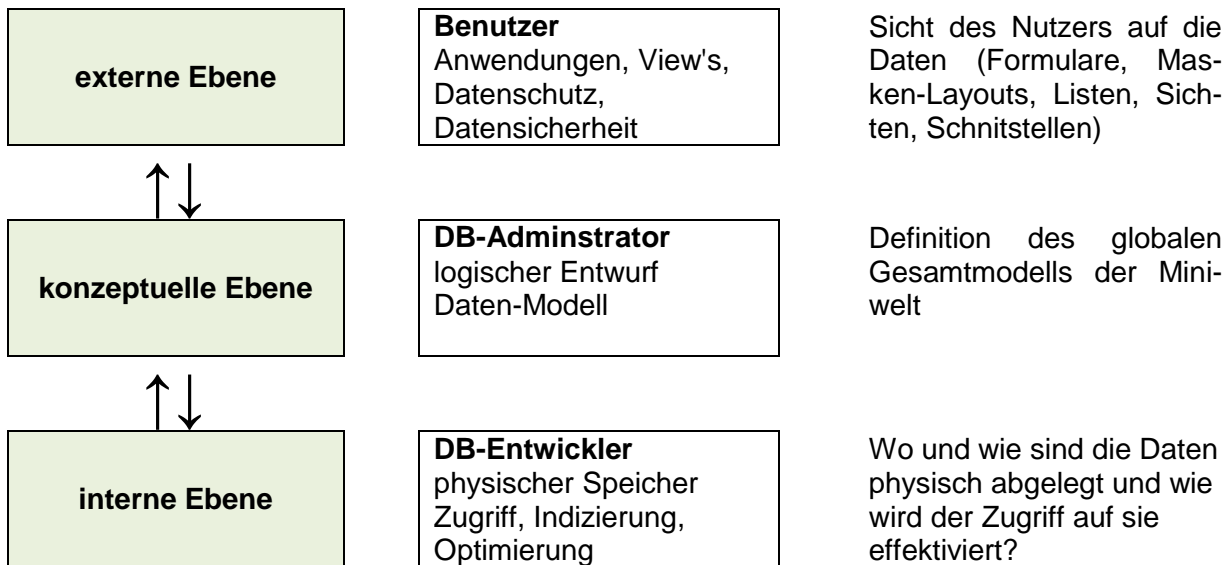
Mit den externen Schemen wird die unterschiedliche Sicht der Anwendungen und Nutzer auf die Daten beschrieben.

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.

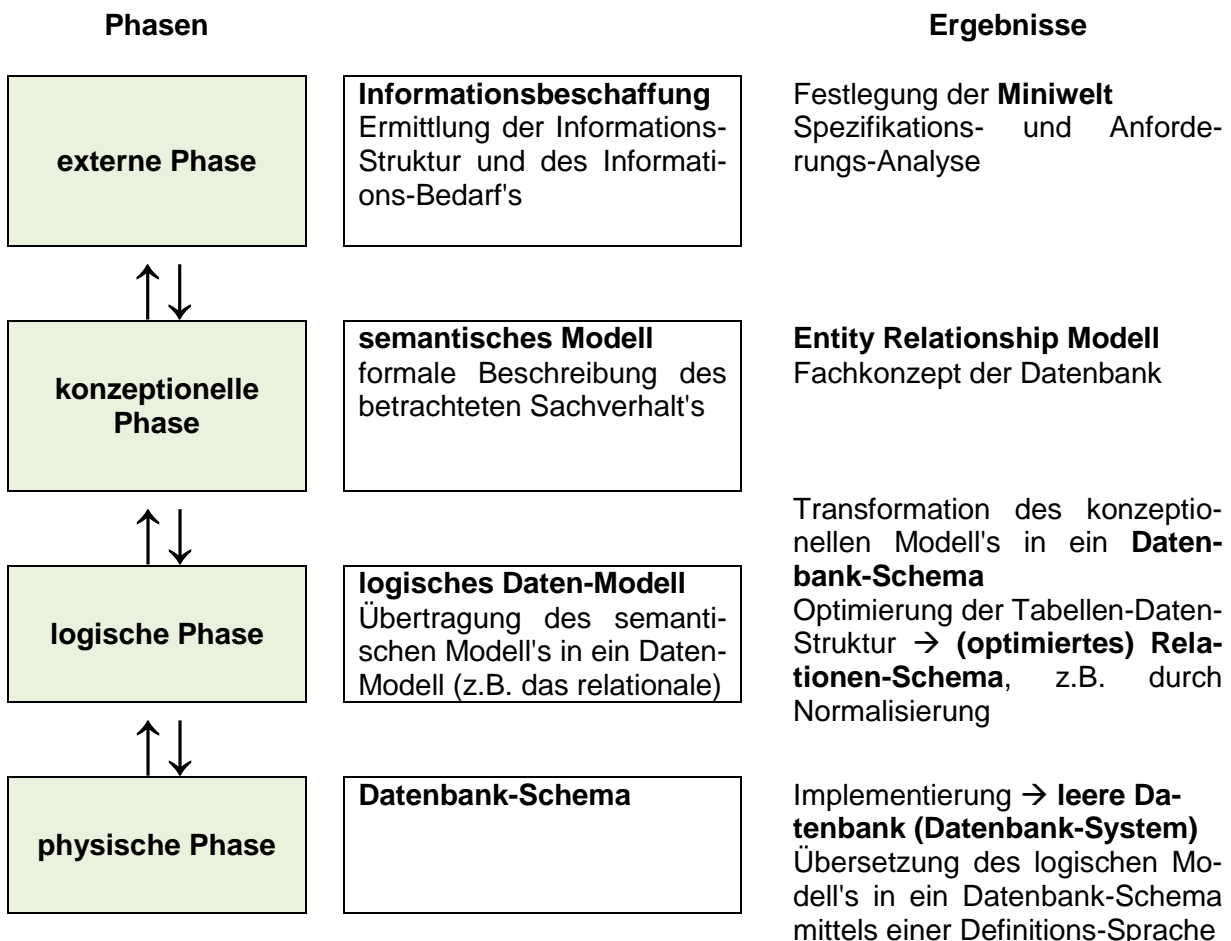
Jede Datenbank hat somit immer ein logisches und ein internes Schema.

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





Phasen der Datenbank-Entwicklung



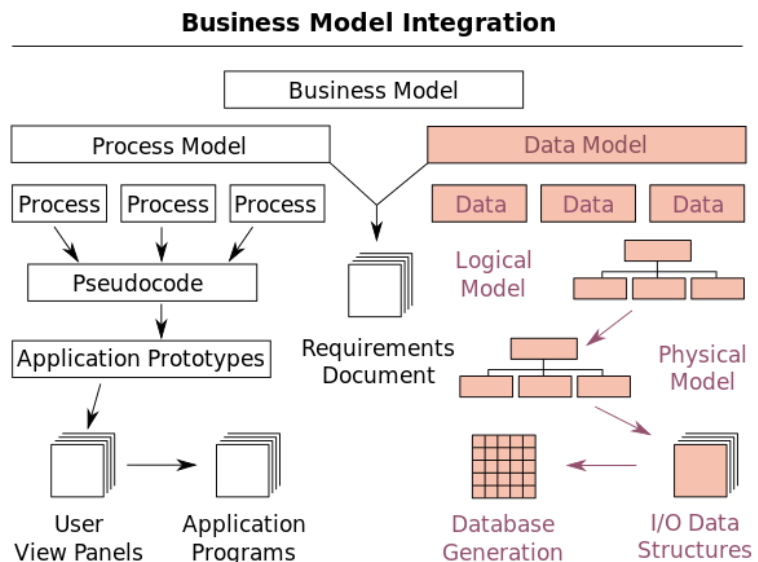
begrifflich und erst recht in der praktischen Umsetzung sind die Ebenen und Schichten nicht sauber getrennt viele Begriffs-Welten existieren parallel Orientierung aber gut an den Adjektiven des Modell-Ebenen möglich, die sind nämlich immer gleich. Es gibt also eine Konzeption, eine Logik und eine Physis.

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. Entsprechend offen sich die Modellierungswerkzeuge, zu denen z.B. das Entity-Relationship-Diagramm (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 herauszuholen. 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 etablierten 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.

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 (zuerst einmal 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 laufender Datenbank-Systeme!

Definition(en): physisches Daten-Schema

Das physische Daten-Schema ist informatisch-technische Umsetzung der Datenbank.

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 des Datenbestandes 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

modellhafte Darstellung der im DBS abgebildeten Mini-Welt 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 Relationship-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-Diagramm (→ [2.1.4. Entity-Relationship-Diagramme \(ERD\)](#)) benutzt.

Vielfach ist das ERM ausschließlich durch das ERD repräsentiert.

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.: Frau Zander 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 (Verbindungs-Typ / - Art)	Typisierung gleichartiger Beziehungen z.B.: Angestellter leitet Projekt; Buch stammt vom Schulbuch-Verlag
• Attribut(s-Typ) (Domäne)	Typisierung gleichartiger Eigenschaften z.B.: Name für Angestellte; Bezeichnung für Projekt Texte, Zahlen, Wahrheitswerte, Bilder, ...

Üblich ist bei einer semantischen Modellierungen der 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 Objekte 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 (unterschiedlichen) 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

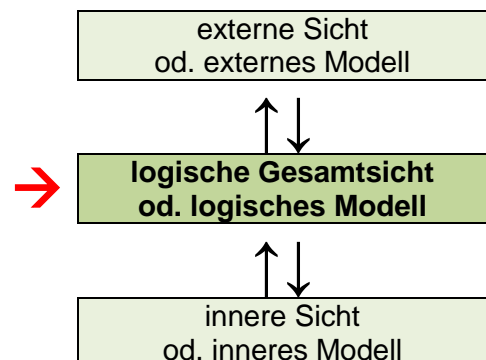
Eine Entität ist ein individuelles und / oder identifizierbares Exemplar eines Dinges, einer Person oder eines Begriffes aus / der realen Welt.

Eine Entität ist ein bestimmtes, wohl unterscheidbares Objekt.

Ein Entity ist ein Informations-Objekt (Individuum, Real-Objekt (Ding), abstraktes 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 oder Kenn-Daten (ID's, Schlüssel) 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.



Entitäts-Typ



Attribute

Kundennummer
Name
Vorname
Straße
Ort
PLZ
Kreditkarten-Nr.
...

Domäne

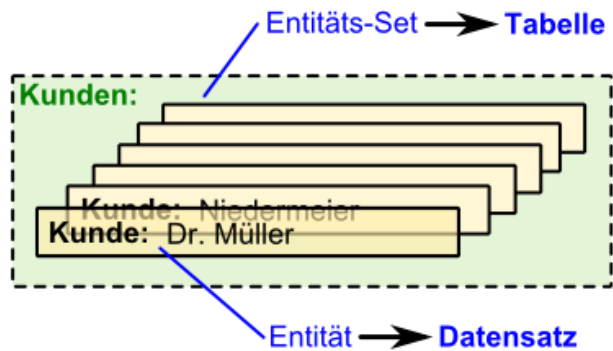
5-stellige Nummer
30 Zeichen
30 Zeichen
25 Zeichen
25 Zeichen
5 Zeichen
19 Zeichen
...



Entität

13865
Dr. Müller
Erwin
E.-Welk-Str. 3
Mustern
12345
0987-9876-1524
...

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

Datensatz → Tupel
Tupel als strukturierte Liste von Merkmalen (Attributen) eines Objekt's

Relation ist Ausprägung eines Relations-Schema's

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 $\text{dom}(a)$.

Eine Domäne (Werte-Bereich) besteht aus einem Namen (/ einer Typ-Bezeichnung) und einer Menge atomarer Werte.
Eine Domäne definiert den Werte-Bereich eines Attribut's.

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 / Relationship's

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.

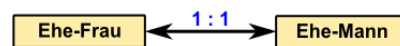
Die Anzahl der beteiligten Entitätstypen an einer Beziehung bestimmt ihren **Grad**.

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

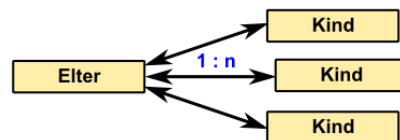
Definition(en): Relation
Eine Relation ist eine Menge / Sammlung von Tupeln / Datensätzen, die gleich-klassige Objekte abbilden.
Eine Relation ist die mathematische Formalisierung der Tabelle.

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

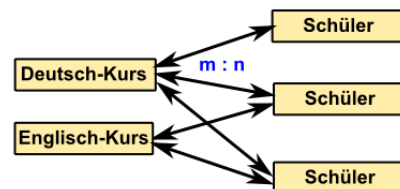
Bei einer "1 zu 1"-Beziehung sind zwei Objekte direkt miteinander verpaart. Eine Ehe besteht z.B. aus zwei Partner, hier durch die klassische Paar-Beziehung dargestellt.



"1 zu n"-Beziehungen beschreiben Verknüpfungen von einem Objekt mit mehreren anderen (meist) eines anderen Typ's. Einem Klassenleiter / Tutor sind z.B. mehrere Schüler zugeordnet. Oder ein Eltern-Teil hat ein bis viele Kinder. Ohne ein Kind wäre derjenige kein Eltern-Teil.



Die "m zu n"-Beziehung beschreibt die freie Kombination von meist zwei verschiedenen Objekt-Typen. Z.B. kann ein Schüler in verschiedenen Kursen sein. Die Kurse bestehen aus mehreren – frei gemischten – Schülern.



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.

Das Entity-Relationship-Modell ist ein informatisches Modell zum Bearbeiten von Entitäten und deren Beziehungen untereinander.

Das Entity-Relationship-Modell ist ein informatisches Modell für Entitäten und deren zugehörige Beziehungen.

Aufgaben:***1. Identifizieren Sie für die nachfolgende Situations-Beschreibung die Elemente für ein Entity-Relationship-Modell!***

Für eine Übersicht über die CD-Sammlung von Karl Brohm sollen die wichtigsten Daten verarbeitet werden.

2. Identifizieren Sie für die nachfolgende Situations-Beschreibung die Elemente für ein Entity-Relationship-Modell!

Lisa Carlsen und Matthes sind ein Paar und haben eine große CD-Sammlung (327 Stück). Damit sie die jeweils selbst-gekauften CD's auch bei einer Trennung auseinander halten können, wollen sie die CD-Sammlung in einer kleinen Datenbank speichern.

3. Identifizieren Sie für die nachfolgende Situations-Beschreibung die Elemente für ein Entity-Relationship-Modell!

In einer Schüler-Bibliothek sind die Bücher (1724 Exemplare) mit einem kleinen Aufkleber versehen, auf dem die Regal-Nummer (von 1 bis 10) und die Ebenen-Nummer (1 – 5) aufgedruckt sind. Die wichtigsten Daten zum Buch und zur Regal-Position sollen später in einer kleinen Datenbank verwaltet werden.

4. Identifizieren Sie für die nachfolgende Situations-Beschreibung die Elemente für ein Entity-Relationship-Modell!

In einer Schule ("Zweistein-Gymnasium Niedermoor") mit gymnasialer Oberstufe sind die 15 Schüler-Gruppen als Kurs organisiert. Zu jedem Kurs gehört ein unterrichtender Lehrer und 12 bis 24 Schüler. Ein Lehrer kann mehrere Kurse unterrichten. Die Schüler sind – entsprechend ihrer Einwahl – verschiedensten Kursen zugeordnet.

(informatische) Begriffs-Welten

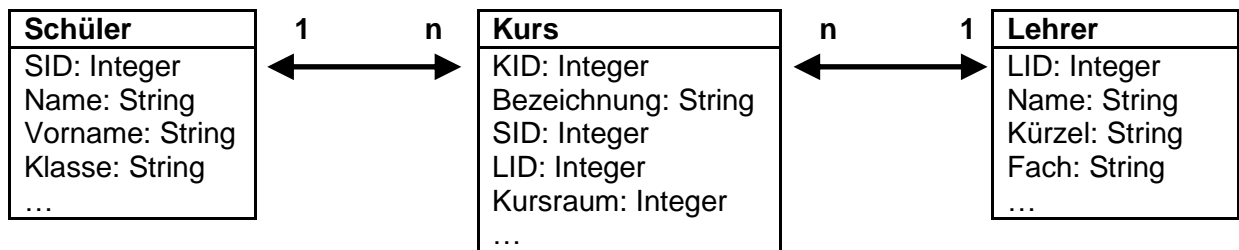
allgemein	Datenbanken (allgemein)	relationale Datenbank	Relationen-Modell	Entity-Relationship-Modell	UML	(Objekt-orientierte) Programmierung	
	Datenbank		Menge an Relationen				
		Tabelle	Relation	Entitäts-Menge Entitäts-Set		Klasse Objekt-Typ	
Merkmals-Bezeichnungen / Eigenschaft(en-Name)		Kopfzeile	Relationen-Typ Relationen-Format	Entitäts-Typ		Klasse Objekt-Typ	
		Spalten-Überschriften	Fremdschlüssel	funktionelle Beziehung (Relationship)	Assoziation		
		Spalte	Attribut				
Gegenstand, Person, Begriff, Ereignis	Entität			Entität		Instanzen, Objekt	
	Datensatz	Zeile	Tupel	Entität	Instanzen, Objekt		
Eigenschaft / Merkmal	Attribut		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!

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		

Eine moderne Form der Darstellung von ERM's ist eine an UML-Diagrammen orientierten Kasten-Form



In einiger Literatur wird diese Form der Darstellung auch als ERD (Entity-Relationship-Diagramm) bezeichnet. In diesem Skript verstehen wir diese Form eher als UML-Diagramm. Die vom Skript bevorzugte Form wird im (folgenden) Abschnitt → [2.1.4. Entity-Relationship-Diagramme \(ERD\)](#) beschrieben

2.1.4. Entity-Relationship-Diagramme (ERD)



Entity-Relationship-Diagramme – kurz ERD – sind eine Form der Darstellung von Daten-Modellen. Meist sind ERD's die einzige Darstellung eines Entity-Relationship-Modell's.

ERD's sind besonders für relationale Modelle geeignet.

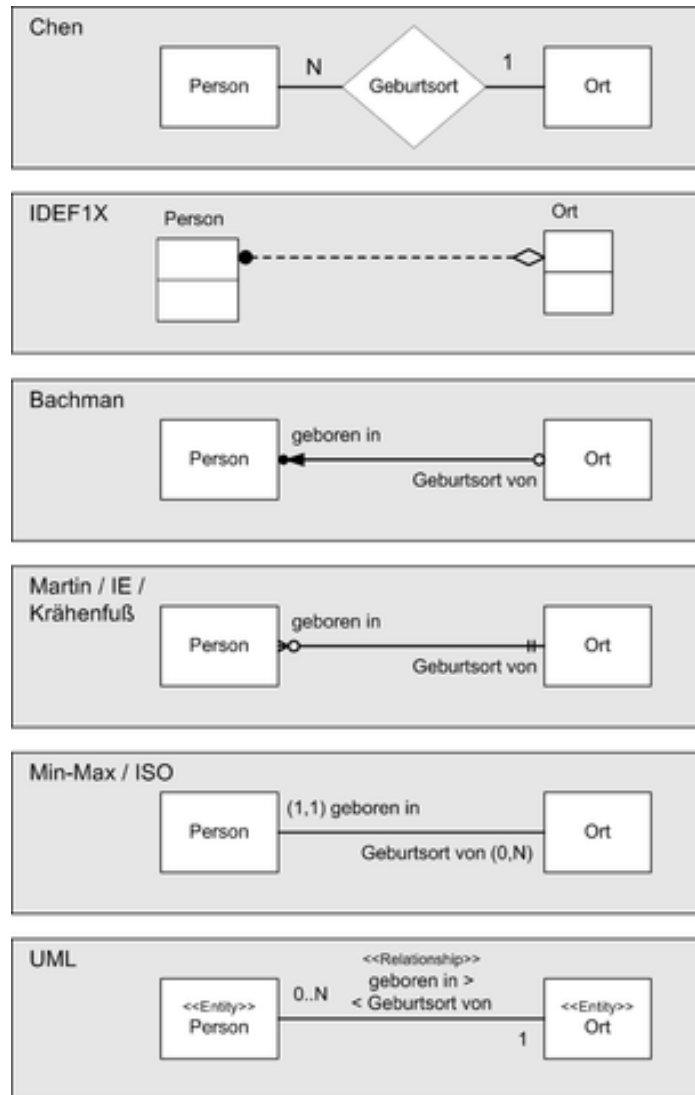
Die Symboliken sind in den letzten Jahrzehnten mehrfach standardisiert und verbessert worden.

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 Projekte 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 objektorientiert programmieren, dann wird neben dem ERD auch UML (Unified Modeling Language (vereinheitlichte Modellierungs-Sprache)) benutzt.

ERD's nach CHEN stellen heute einen Quasi-Standard dar.



ERD (oben) im Vergleich zu anderen Varianten von ER-Modellen
Q: de.wikipedia.org (Frank Roeing)

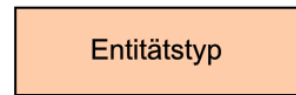
Objekte und Symbole der Entity-Relationship-Diagramme

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 Primärschlüssel-Charakter werden wird der Attributname 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.

In diesem Kurs werden sie als nicht zugelassen (quasi: verbotene) Symbole betrachtet.



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.



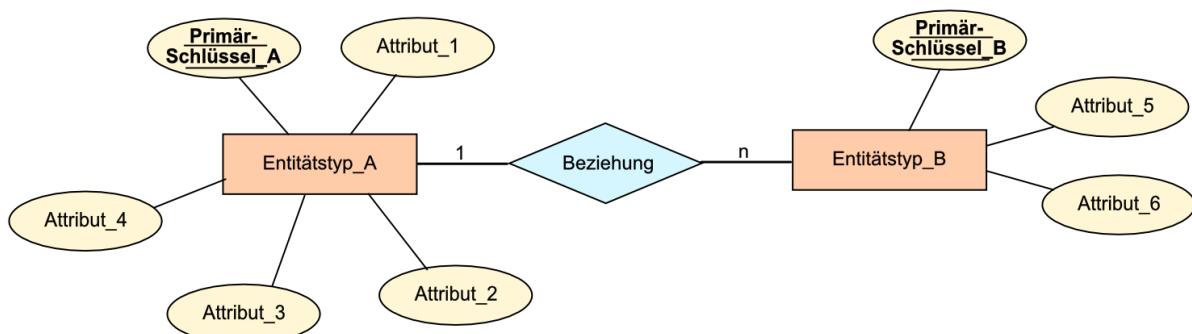
Beziehung, Assoziationen, Kardinalität

Beziehung besteht aus zwei Assoziationen in beide Richtungen mit zugehörigen 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-Typ 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".

Besonders wenn Kardinalitäts-Angaben oder Pfeile angegeben sind, dann gilt nur eine Lesart.

Das Symbol für eine Beziehung ist der Rhombus.

Im nächsten Schritt ermitteln wir die Kardinalität 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).

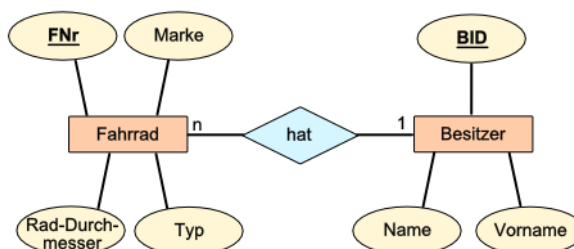
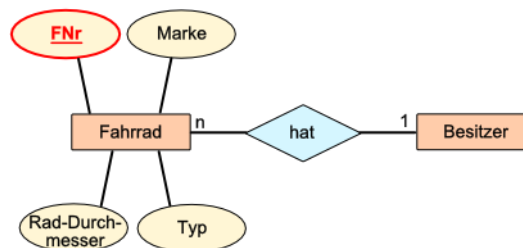
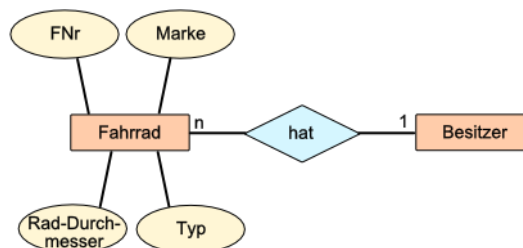
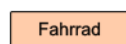
Die Fahrrad-Nummer bietet sich auch als Erkennungs-Merkmal (- oder Schlüssel -) für die Entitäten an.

Das Schlüssel-Attribut kennzeichnen wir durch eine Unterstreichung.

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

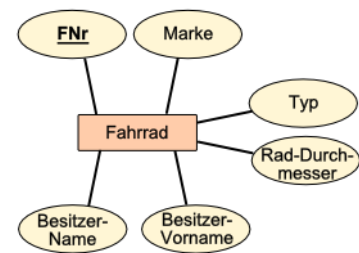
Genau, wie beim Fahrrad – verfahren wir jetzt auch beim zweiten Entitäts-Typ.

Er bekommt seine Attribute 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 (möglicherweise) weggelassen 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.

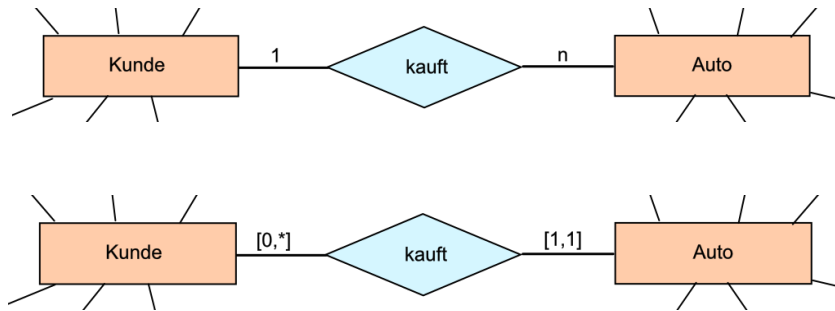
Nicht zwingende Beziehungen sind freigestellt (optional). In beiden Entitäts-Typen können unverbundene Entitäten vorkommen.

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:

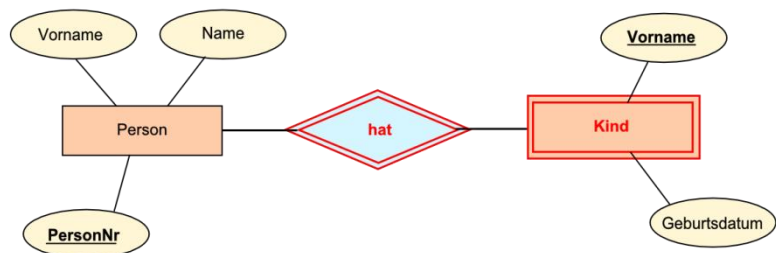


- **[0,*]** keine Einschränkung vorgesehen; ein Kunde kauft (z.B. in diesem Jahr) kein Auto oder beliebig viele
- **[1,*]** der Kunde muss mindestens 1 Auto kaufen / gekauft haben
- **[1,1]** hier ist die Einschränkung so, dass 1 Kunde auch genau nur 1 Auto kauft / gekauft hat

zu den Integritäts-Bedingung (→ [2.0.2.2. Daten-Integrität](#)) zählt, dass von jeder Beziehung mindestens eine Kante (Assoziation) aus- / abgeht
Entitäten müssen nicht zwingend eine Kante (Assoziation) zu einer Beziehung besitzen

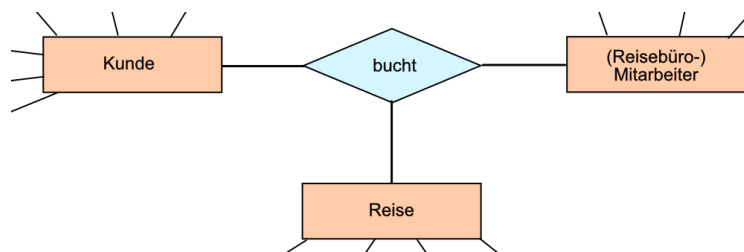
Im ERD werden sowohl die Beziehung und der "schwache" Entitätstyp doppelt umrandet gezeichnet.

schwache Entitäten benötigen immer eine eindeutige Identifikation über eine Beziehung



schwache Entität und schwache Beziehung

mehrdeutige Beziehungen



Schlüssel dienen der eindeutigen Identifizierung von Entitäten

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.

Definition(en): Entitätsschlüssel / Schlüsselfeld / Schlüsselattribut
Ein Entitätsschlüssel 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.

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 abhängig? Erläutern Sie!*

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. Das benötigt aber zusätzlichen Speicherplatz.

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.

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.

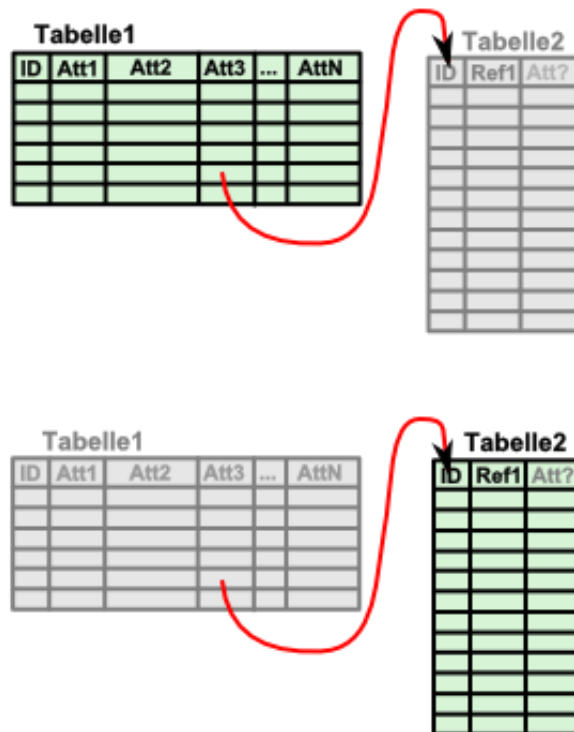
Ein Primärschlüssel ist ein ausgewählter / festgelegter Entitätsschlüssel.
 Dabei sind natürlichen Zahlen (Nummern) vor Texten und vor zusammengesetzten Attributs-Kombinationen der Vorrang zu geben.

Ein Primärschlüssel ist das eindeutige / einmalige, primär-genutzte Zugriffselement auf einen Datensatz (einer Tabelle).

Verweisen wir in einer (Basis- od. Master-)Tabelle auf Datensätze in anderen Tabellen, dann nutzen wir dazu die dort genutzten Schlüssel. Diese Schlüsselnummern sind nun sogenannte Fremdschlüssel. Man spricht auch von Sekundärschlüsseln (secondary keys).

In der Objekt-Tabelle (Master-Tabelle) können sie auch mehrfach vorkommen, weil bestimmte Daten ja vielleicht auch mehrfach vorkamen. (Deshalb werden wir eine sogenannte Normalisierung vornehmen. Dieses Verfahren schauen wir uns später an (→ [2.2.2. Normalisierung](#).)

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.



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

Definition(en): Fremdschlüssel / Foreign key

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.

Ein Fremdschlüssel ist in einer relationalen Datenbank eine Verbindung zu einem Primärschlüssel einer anderen Tabelle.

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

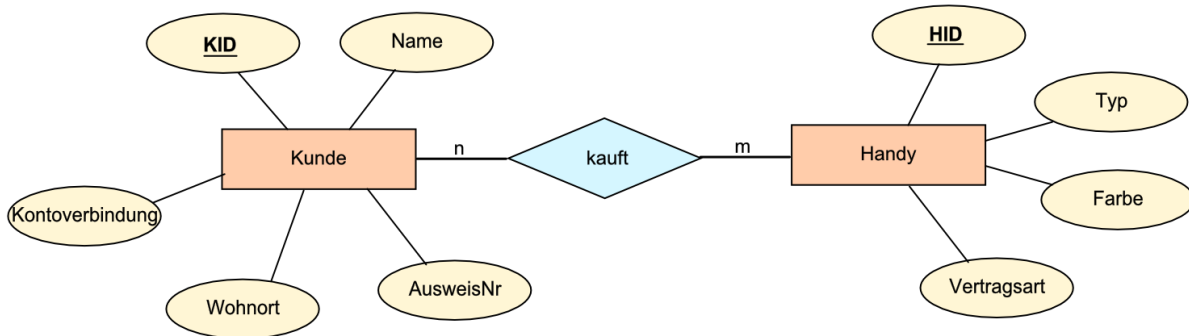
3. Wenn Sie bei der Primärschlüssel-Auswahl nur die vorhandene Tabelle beacht haben, dann prüfen Sie Ihre Wahl noch einmal unter dem Gesichtspunkt, dass die Datenbank für Gesamt-Deutschland genutzt werden soll!

für die gehobene Anspruchsebene:

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

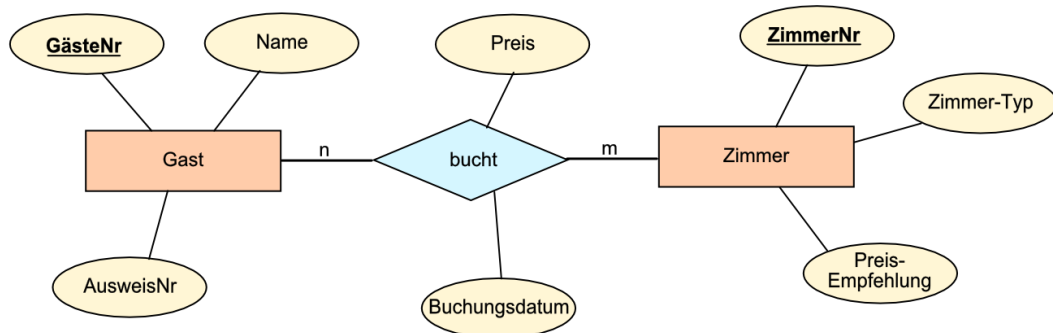
Übungs-Aufgaben:

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!

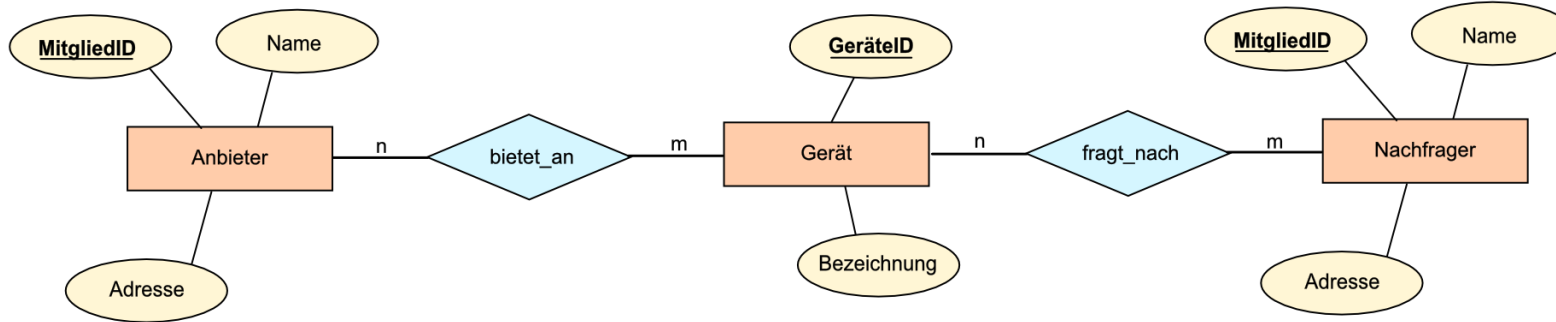
für die gehobene Anspruchsebene:

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

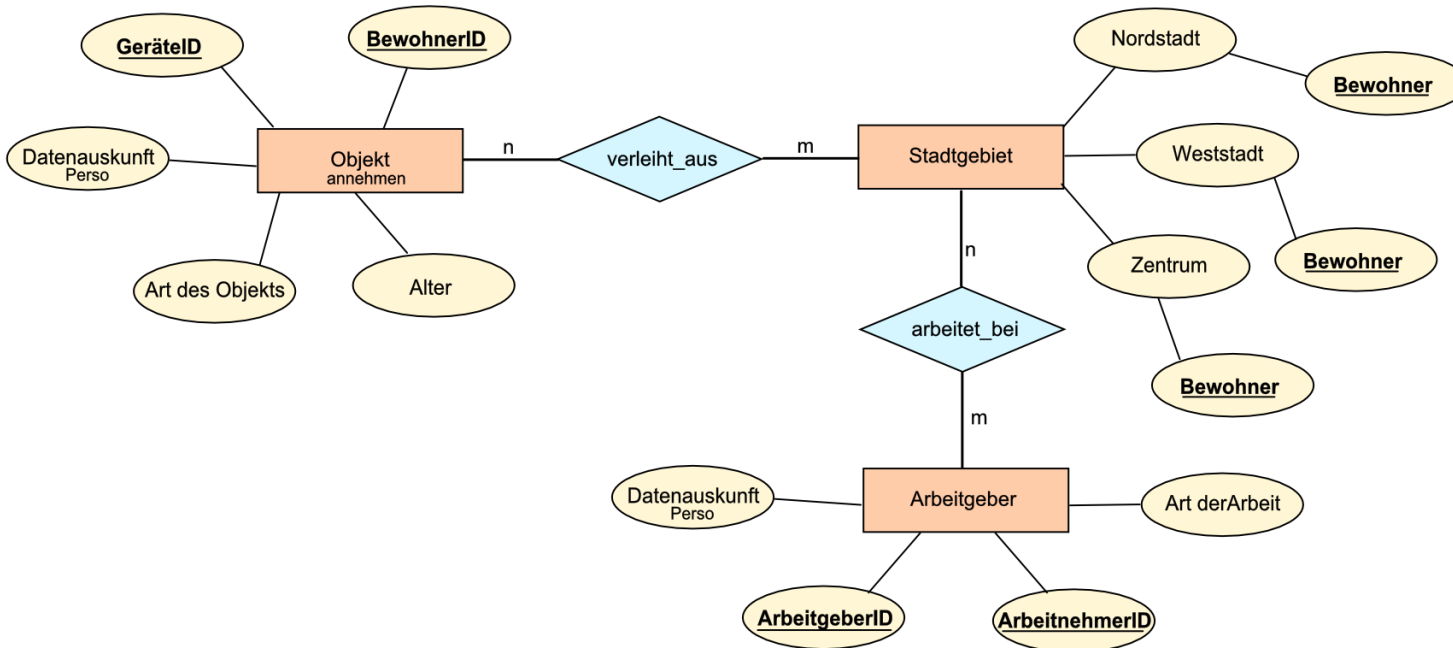
Komplexe Aufgabe zu den Entity-Relationship-Modellen

- 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-Relationship-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!**

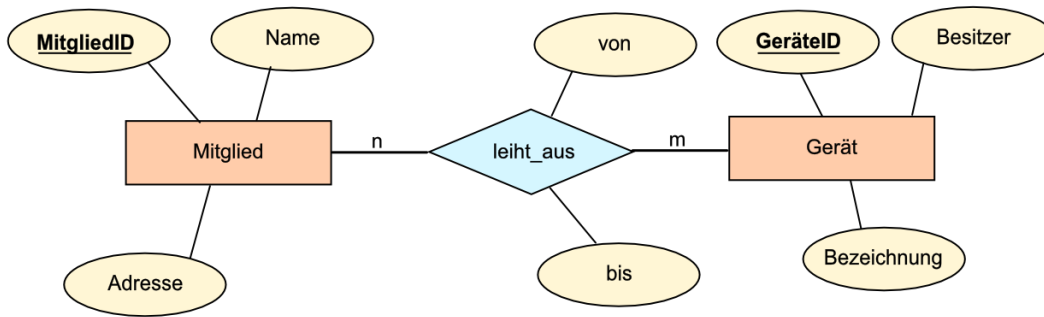
Vorschlag1:



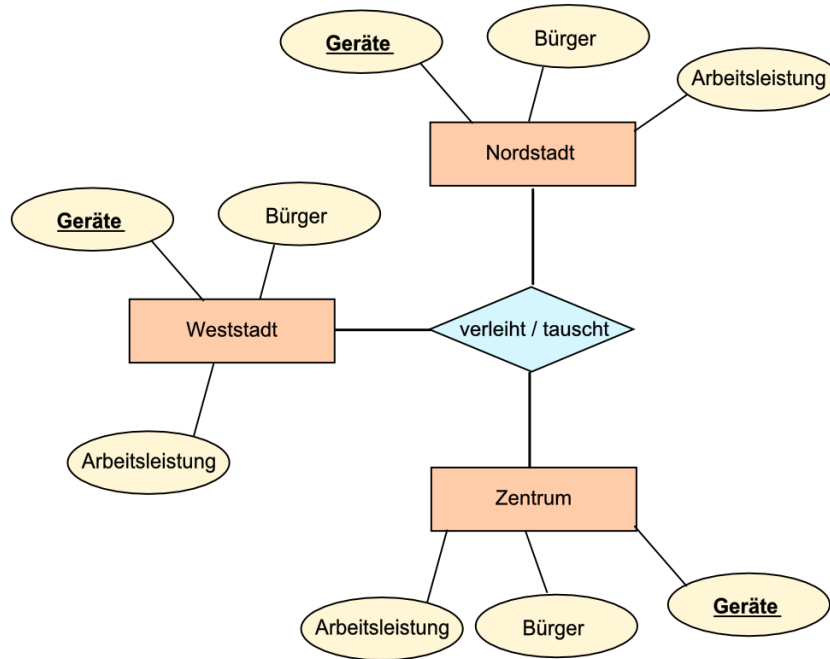
Vorschlag2:



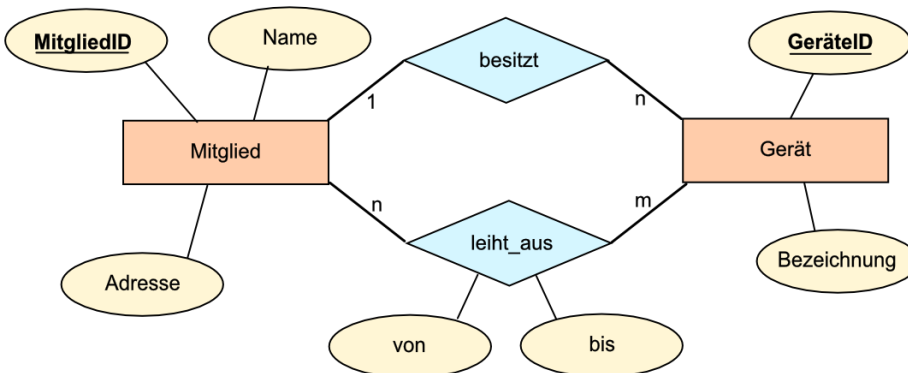
Vorschlag3:



Vorschlag4:



Vorschlag5:



2.1.5. Erweiterungen des Entity-Relationship-Modells

Erweiterungen bei Attribut

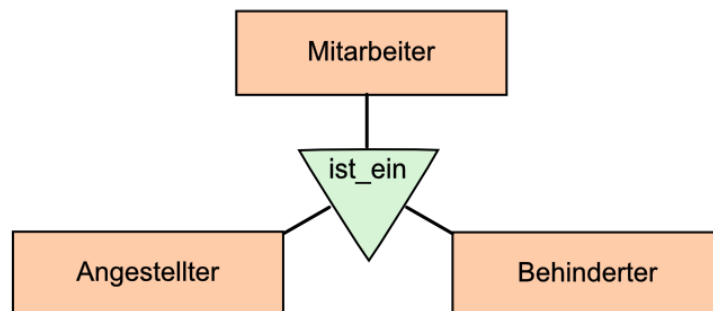
- | | |
|----------------------------------|---|
| • optionale Attribute | sind Attribute, die nicht bei allen Entitäten eines Types einen Wert annehmen müssen
(als graphisches Zeichen in ERD wird ein Kringel in der Verbindungslinie verwendet) |
| • strukturierte Attribute | sind Attribute, die sich 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 Umkehrung der Generalisierung, 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



Eine erste Annäherung an die relationale Datenbank könnte die das Relationen-Schema sein. In diesem Schema wird praktisch der Kopf der Tabellen (Relationen) dargestellt. Weiterhin sind die Verknüpfungen und die Schlüssel-Spalte im Schema sichtbar. Praktisch handelt es sich um eine textuelle Codierung des Entity-Relationship-Diagramm's. Nehmen wir als Beispiel dieses ERD:

verschiedene Syntax-Versionen beschrieben

recht praktisch erscheint mir eine Variante mit Gleichheits-Zeichen zwischen Relationen-Name und der Attributs-Menge – dem Relationen-Schema

Relationen-Name = Relationen-Schema

und

Relationen-Schema = (Attributs-Menge)

damit lassen sich dann später Datenbank-Schemata einfacher notieren

alternativ kann auch definiert werden, dass die weit verbreitete Schreibweise:

Relationen-Name(Attributs-Menge)

wie die obige Schreibung verstanden werden soll

in diesem Skript werden beide Schreibweisen äquivalent verwendet

Das **Vorgehen für die Erstellung des Relationen-Schema** ist recht einfach zu realisieren:

1. alle Entitäts-Typen werden zu jeweils einer Relation

Relationen-Name

2. die Attribute werden innerhalb einer runden Klammer Komma-getrennt aufgezählt

Relationen-Name(Attributs-Menge)

genauer:

Relationen-Name(Attribut1, Attribut2, ..., AttributN)

3. das Schlüssel-Attribut wird unterstrichen (ev. auch fett gedruckt)

Relationen-Name(Schlüssel, Attribut1, ..., AttributN)

4. Datentypen können hinter den einzelnen Attributen nach einem Doppelpunkt festgelegt werden

(übliche Datentypen: integer; decimal; string; boolean)

Relationen-Name(Attribut1: **integer**, Attribut2: **string**, ..., AttributN: **decimal**)

-
5. "1 zu 1"- oder "1 zu n"-Beziehungen werden über ein zusätzliches Attribut in der Relation mit der höheren Kardinalität bereitgestellt, das Attribut bekommt einen Verweis-Pfeil;

Relationen-Name(Attribut1, Attribut2, ..., ↗AttributN+1)

- 5a. sind die Fremdschlüssel-Attribute nicht eindeutig benannt, dann erfolgt eine Notierung über den Relation-Namen und das Primärschlüssel-Attribut in runden Klammern

(alternative Notierungs-Form: Relation-Name_Schlüssel-Attribut)

Relationen-Name(Attribut1, ↗Relation(Schlüssel), ..., AttributN)

oder:

Relationen-Name(Attribut1, Relation(↗Schlüssel), ..., AttributN)

6. "n zu m"-Beziehungen werden in einer extra Relation mit dem Beziehungs-Namen abgebildet; die beiden Verweise werden als Attribute mit Verweis-Pfeil notiert; zusätzliche Attribut der Beziehung werden als Attribute in die Relation aufgenommen

Relationen-Name(Relation1(↗Schlüssel), Relation2(↗Schlüssel),..., AttributN)

7. die Relations-Schemata können zu einem Datenbank-Schemata zusammengefasst werden

Datenbank-Name = (RelationSchema1, RelationSchema2, ..., RelationSchemaN)

die konkrete Datenbank ist dann die Menge aus Relationen

Definition(en): Relationen-Schema

Das Relationen-Schema ist eine textuelle Umschreibung eines Entity-Relationship-Modell's mit Aussagen darüber, welche Tabellen (Relationen) mit welchen Inhalten (Attributen) wie miteinander verknüpft sind.

Ein Relationen-Schema R – Schreibweise $R(A_1, A_2, \dots, A_n)$ – bezeichnet eine Menge von Attributen $\{ A_1, A_2, \dots, A_n \}$.

Relationen-Schema

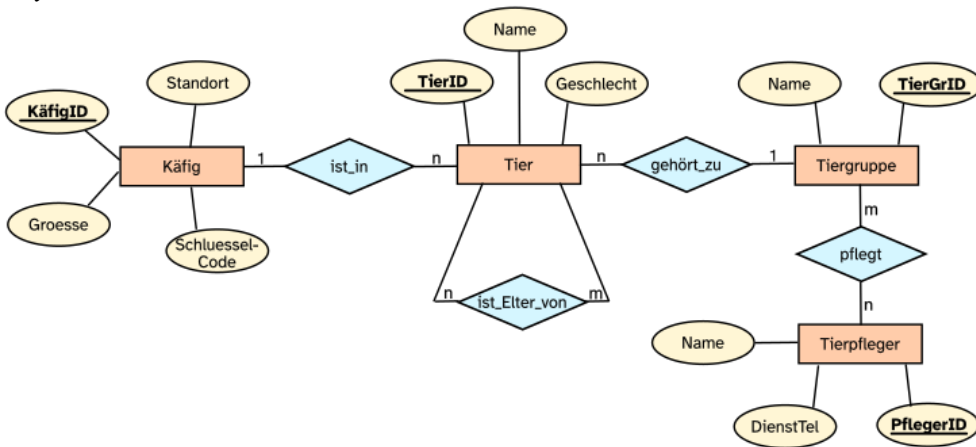
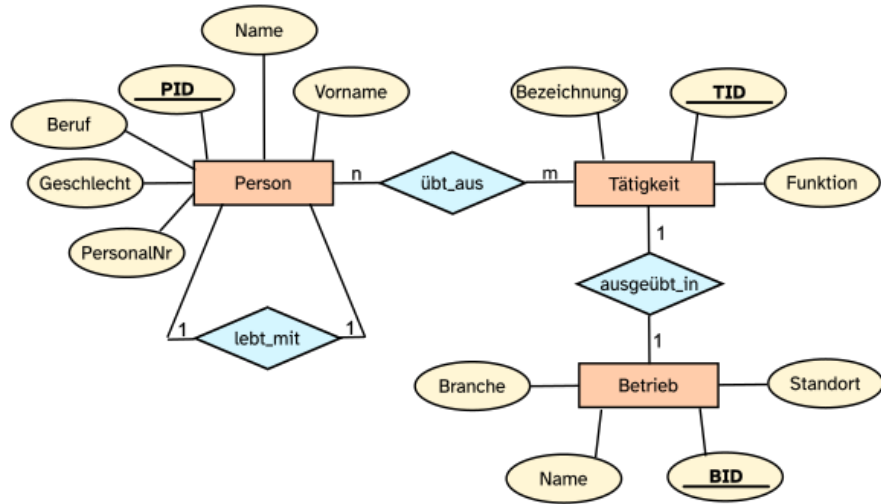
Entitäts-Typ \rightarrow Relation-Name

Attribute (eines Entitäts-Typ's im ERD) \rightarrow Attribute im Relationen-Schema

Aufgaben:

1. Erstellen Sie aus dem nebenstehenden ERD das Datenbank-Schema mit allen Relationen-Schemata!

2. Entwickeln Sie aus dem folgenden ERD die Relationen-Schemata und das zusammenfassende Datenbank-Schema!



3. Entwickeln Sie aus dem folgenden Datenbank-Schema ein ERD!

KaufBelegeDB = (Kunde, Kauf, Artikel)

Kunde(KundenID, Name, PLZ, Ort, Strasse)

Artikel(ArtikelID, Name, Preis)

Kauf(BelegNr, ↗KundenID, ↗ArtikelID, Anzahl, Datum, GesamtPreis)

für die gehobene Anspruchsebene:

4. Ein ausgeschiedener DB-Entwickler hinterließ das folgende Relationen-Schema. Leiten Sie daraus ein Entity-Relationship-Diagramm für eine Präsentation vor weniger informatisch gebildetem Personal ab!

LehramtsStudent(MatrikelNr, Name, Vorname, PLZ, Ort, Strasse, ImmatrikulationsJahr, Semester, eMail)

StudentischeHilfskraft(VertragNr, MatrikelNr, StartDatum)

Fach(FachID, Bezeichnung)

Professor(PersNr, Name, Lehrstuhl, eMail)

Vorlesung(VorlesID, Bezeichnung, ↗Fach, ↗PersID, Raum, Tag, ZeitSlot, StdAnzahl)

Lehrstuhl(LSID, Bezeichnung, ↗Fach)

Übung(ÜbgNr, Leiter(↗Assistent), Raum, Tag, ZeitSlot, StdAnzahl, ↗Vorlesung)

Fächer(FächerID, ↗MatrikelNr, ↗Fach, StartSem, EndeSem)

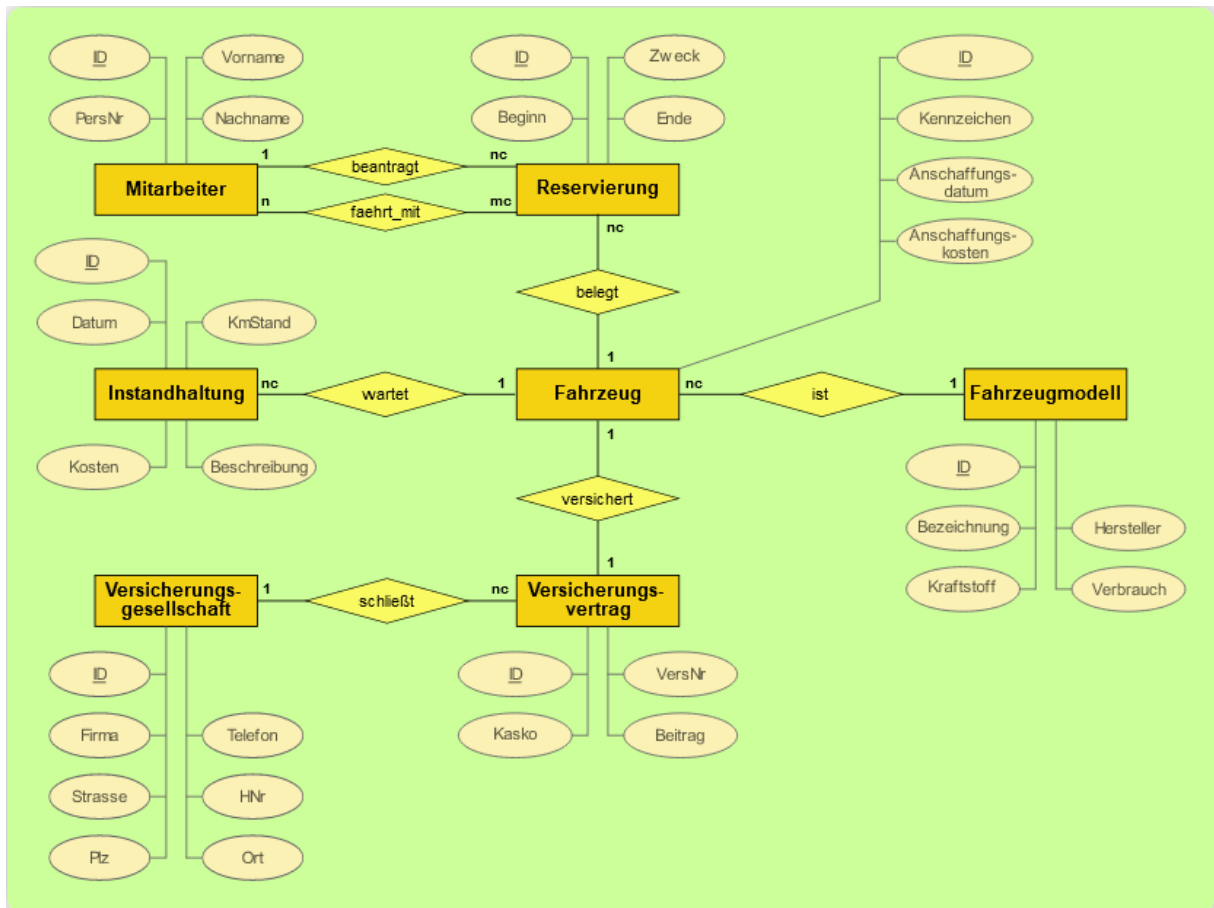
korrigiert(↗MatrikelNr, ↗Übung)

Assistent(AssisID, Name, akadGrad, eMail)

5. Beurteilen Sie das vorliegende Relationen-Schema! Machen Sie ev. Verbesserungsvorschläge

komplexeres Übungs-Beispiel: Fuhrpark

Q: https://www.kstbb.de/informatik/rdb/01/1_2_Entwurf.html

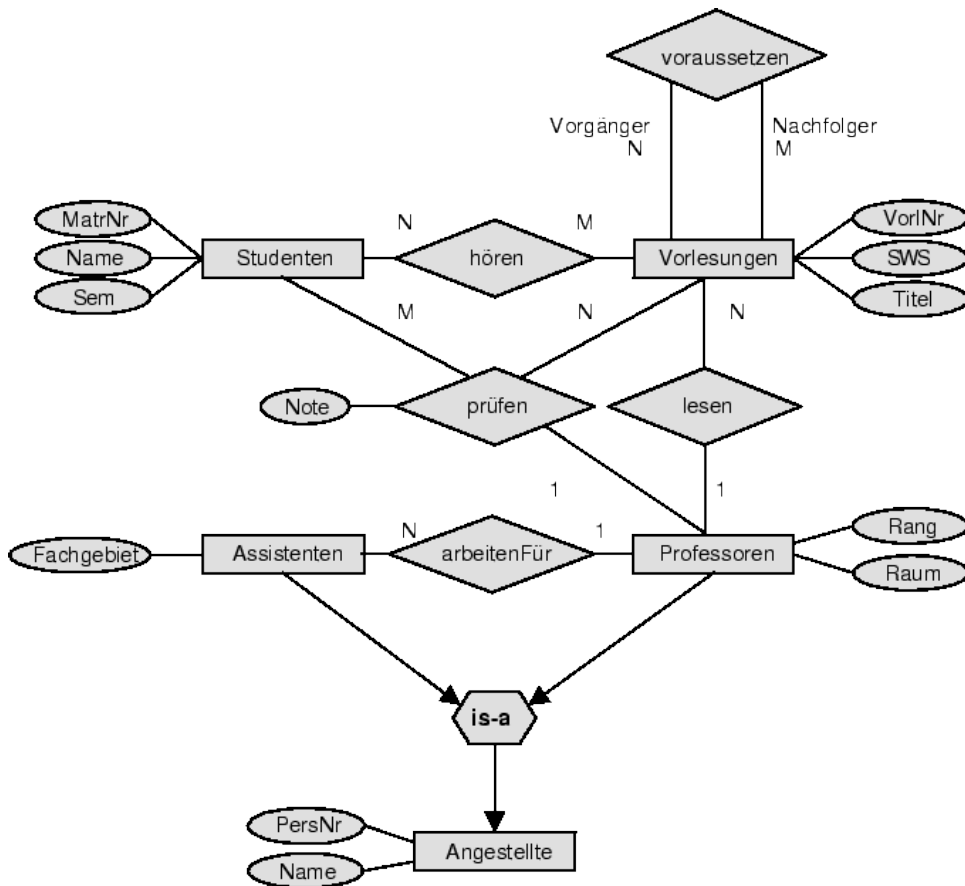


Lösung:

Fahrzeugmodell(ID, Bezeichnung, Hersteller, Kraftstoff, Verbrauch)
Fahrzeug(ID, Kennzeichen, Anschaffungsdatum, Anschaffungskosten,
↑Fahrzeugmodell_ID)
Mitarbeiter(ID, PersNr, Vorname, Nachname)
Reservierung(ID, Zweck, Beginn, Ende, ↑Mitarbeiter_ID, ↑Fahrzeug_ID)
Mitarbeiter_faehrt_mit_Reservierung(↑Mitarbeiter_ID, ↑Reservierung_ID)
Instandhaltung(ID, Datum, KmStand, Kosten, Beschreibung, ↑Fahrzeug_ID)
Versicherungsgesellschaft(ID, Firma, Telefon, Strasse, HNr, Plz, Ort)
Versicherungsvertrag(ID, VersNr, Kasko, Beitrag, ↑Fahrzeug_ID,
↑Versicherungsgesellschaft_ID)

komplexeres Übungs-Beispiel: Universität

Q: <http://www-lehre.informatik.uni-osnabrueck.de/~dbs/2001/skript/node41.html>



Lösung:

Studenten : {[MatrNr : integer, Name : string, Semester : integer] }

Vorlesungen : {[VorlNr : integer, Titel : string, SWS : integer] }

Professoren : {[PersNr : integer, Name : string, Rang : string, Raum : integer] }

Assistenten : {[PersNr : integer, Name : string, Fachgebiet : string] }

hören : {[MatrNr : integer, VorlNr : integer] }

lesen : {[PersNr : integer, VorlNr : integer] }

arbeitenFür : {[AssiPersNr : integer, ProfPersNr : integer] }

voraussetzen : {[Vorgänger : integer, Nachfolger : integer] }

prüfen : {[MatrNr : integer, VorlNr : integer, PersNr : integer, Note : decimal] }

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
 $R = \{a_1, a_2, a_3, \dots, a_n\} \cup \{k\}$
- 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)
 $R = \{a_1, a_2, a_3, \dots, a_n\} \cup \{k\} \cup \{k'\}$
- 3. 1:1-Beziehungen** eine der beiden Relationen (T od. S) wird um den Fremdschlüssel der anderen ergänzt
 $S = \{a_1, a_2, a_3, \dots, a_n\} \cup \{k\} \cup \{k_T\}$ oder
 $T = \{a_1, a_2, a_3, \dots, a_n\} \cup \{k\} \cup \{k_S\}$
- 4. 1:N-Beziehungen** N-kardinale Relation T wird um den Fremdschlüssel der 1-kardinalen Relation S ergänzt
 $T = \{a_1, a_2, a_3, \dots, a_n\} \cup \{k\} \cup \{k_S\}$
- 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
 $R = \{a_1, a_2, a_3, \dots, a_n\} \cup \{k\} \cup \{k_T\} \cup \{k_S\}$
- 6. mehrwertige Attribute** erstellen einer Relation R mit dem mehrwertigen Attribut a_x aus (der Relation) S und dem Fremdschlüssel von S
 $R = \{a_x\} \cup \{k_S\}$
- 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
 $R = \{a_1, a_2, a_3, \dots, a_n\} \cup \{k_1, k_2, k_3, \dots, k_m\}$
für alle anderen Fälle ist der Primärschlüssel die Menge von n-1 Fremdschlüssel (, wobei die mit der Kardinalität >1 immer im Primärschlüssel enthalten sein müssen

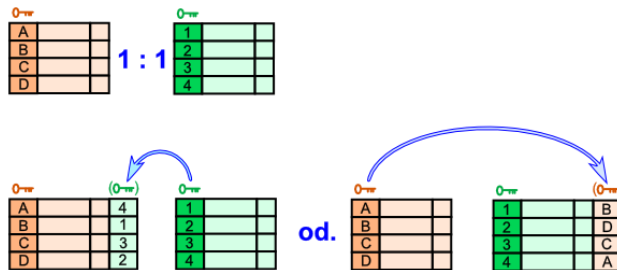
Legende:

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

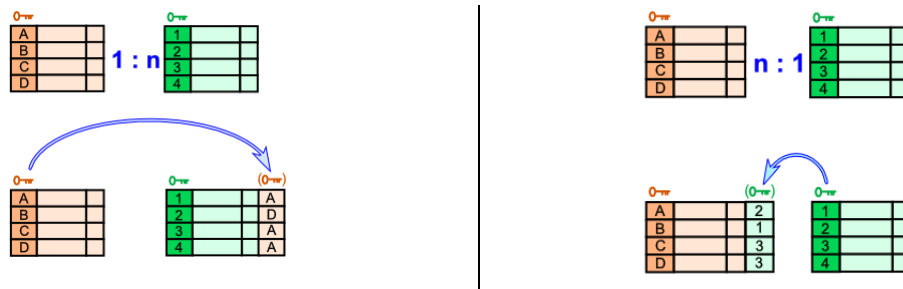
abgeleitet: vereinfachte Transformations-Regeln zur Überführung eines ERM in eine relationale Datenbank

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

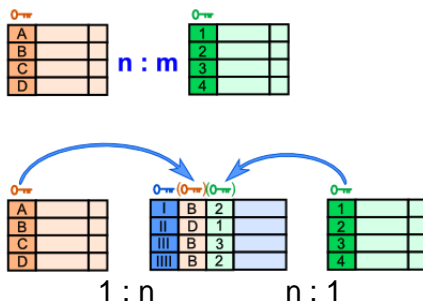
2. Regel Gilt zwischen zwei Entitäten der Beziehungs-Typ / die Kardinalität 1 : 1, dann wird an einer der Tabellen eine Spalte / ein Attribut (mit dem Namen der Beziehung) ergänzt und mit den zugehörigen Primär-Schlüsseln der Datensätze der anderen Entität / Tabelle (vollständig) ausgefüllt.



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).



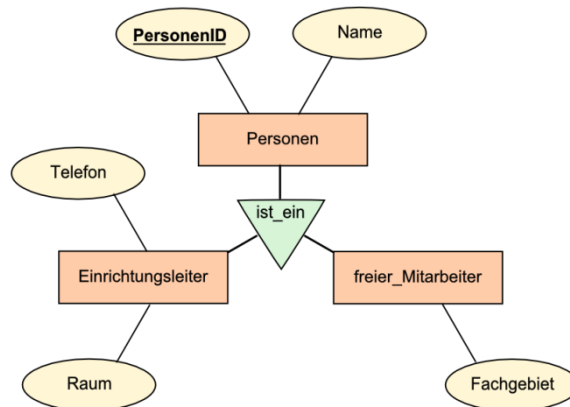
4. Regel Gilt zwischen zwei Entitäten der Beziehungs-Typ / die Kardinalität n : m, dann wird eine neue Tabelle erstellt, die neben einem eigenen Primär-Schlüssel und ev. zusätzlichen Attributen im Relations-Schema auch zwei Spalten (Felder / Attribute) enthält, welche die Namen der Tabellen tragen und 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)

weitere Übungs-Aufgaben:

https://www.kstbb.de/informatik/rdb/01/1_4_Uebungsaufgaben.html

2.1.6.1. Weiterentwicklungen des Entity-Relationship-Modells

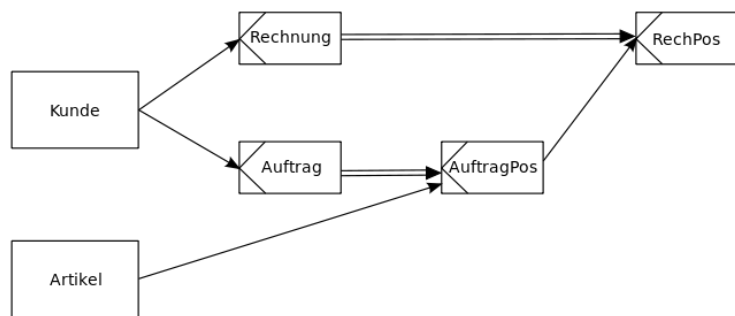
Structured ERM (SERM)

1988 von Elmar SINZ

Ziele

Strukturierung großen Daten-Schemata durch quasi-hierarchische Anordnung
Visualisierung von Existenz-Abhängigkeiten durch Beziehungs-Semantik(en)
Vermeidung von Inkonsistenzen 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.

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

Tabellen liegen hier i.A. denormalisiert vor

Ziel ist eine hohe Performance

Nachteil ist ein erhöhter Speicherbedarf

Fakten-Tabelle besteht vorrangig aus Fremdschlüsseln zu Dimensionstabellen

eigene Attribute sind in der Faktentabelle auch zugelassen

zu den Dimensionstabellen besteht eine 1 : n -Beziehung (Dim-Tab : Fak-Tab)

die Dimensionstabellen beinhalten beschreibende Daten und können somit auch eigene Attribute besitzen

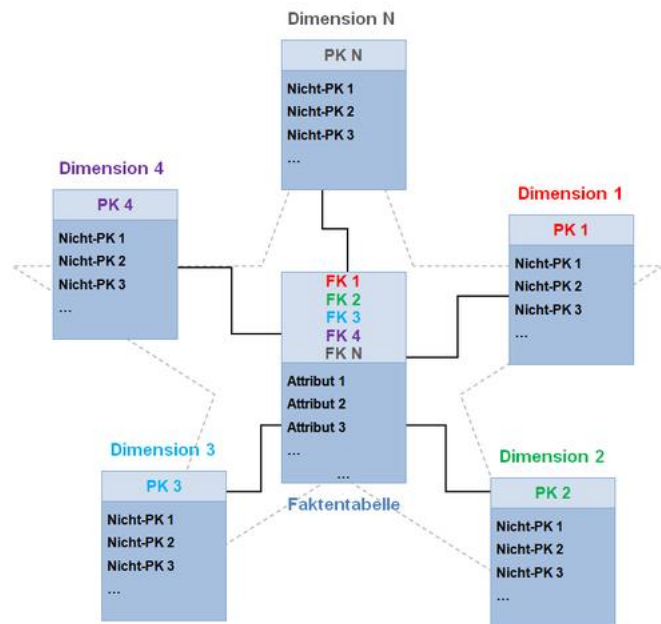
nächste Erweiterung / Verbesserung ist das Schneeflocken-Schema

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, Betrügereien, ...)

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
```



Stern-Schema (allg.)

(Die Fakten-Tabelle hat einen zusammengesetzten Primärschlüssel aus den Primärschlüsseln der Dimensionstabellen)

Q: de.wikipedia.org (Dhbw dbi); cc-by-sa 3.0

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

Normalisierungen 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



2.2.0. Grundbegriffe und Allgemeines

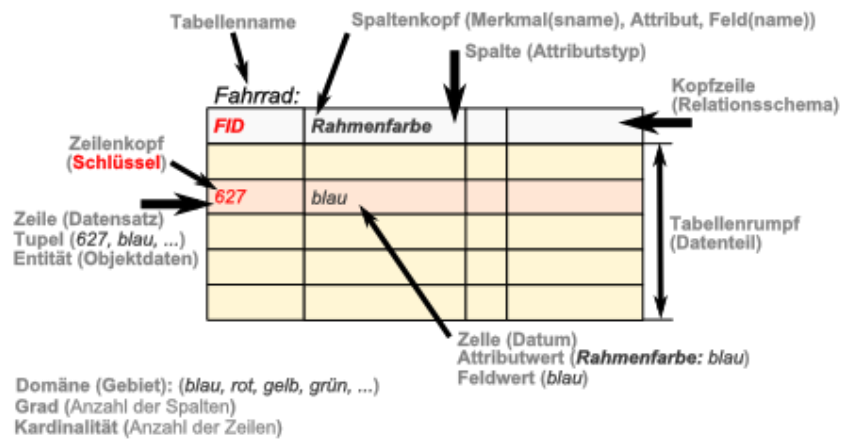
relatio (lt: zurückfragen); Beziehungen
 Daten werden in Tabellen-Form gespeichert, verwaltet

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

Definition(en): relationales Daten-Modell
Das relationale (Daten-)Modell ist das Daten-Modell, dass Tupel bzw. Tabellen als Daten-Struktur verwendet.
Werden Daten und ihre Beziehungen untereinander in Form von Tupeln (praktisch Tabellen), strukturiert und verarbeitet, dann spricht man vom relationalen Daten-Modell.
Das relationale Daten-Modell ist das logische Konzept, Daten und ihrer Beziehungen in Form von Relationen (notiert als Tupel oder Tabellen) zu beschreiben.

Beispiel: Kartei	Beispiel:	Datenbank	allgemein	Äquivalent: Objekt-Orientierung
Aktenschrank		Datenbank	Gesamt-Sammlung	Klasse Main-Objekt
Karteikasten		Tabelle	Teil / Bereich der Sammlung	Unter-Klasse untergeordnetes Objekt
Karteikarte		Datensatz	abgelegtes / gespeichertes / eingelagertes Objekt	Unter-Klasse untergeordnetes Objekt
Überschrift		Primär-Schlüssel	Name / Kennung des Objektes	Objekt-Name
Eintrag		Attribut	Merkmale des Objektes	(Objekt-)Attribut
Reiter od. Lochung		Index	hervorgehobene Such- und Kennungs-Merkmale	Attribut

Tabelle (Entitätstyp)



Definition(en): Relationales Datenbank-Management-System (RDBMS)

Ein relationales DBMS ist ein DBMS, welches das relationale Daten-Modell unterstützt. Charakterisiert wird das relationale Daten-Modell durch eine Daten-Anordnung in Tabellen (Relationen), die einen Namen haben und eine feste Anzahl von Attributen (Spalten) besitzen. Jede Zeile enthält einen Datensatz mit den Ausprägungen der einzelnen Attribute.

2.2.0.1. Sind Tabellen und Relationen ein und das Selbe?

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

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

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

$$-3 \rightarrow 9 \quad -2 \rightarrow 4 \quad \dots \quad 3 \rightarrow 9$$

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 $x \rightarrow y$ eindeutige Werte.

Versucht man $y \rightarrow x$, 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!*

Definition(en): Relation (allg.)

Eine Relation ist eine Menge von Objekten, die durch Paare / Tupel einer eindeutigen und vollständigen Attributs-Kombination gebildet wird.

2.2.0.2. Grund-Begriffe, -Elemente und –Verfahren in relationalen Datenbank-Modellen

Definition(en): flache Tabellen / flache Relationen

Eine flache Relation bildet alle Daten zu einem Objekt in einer Zeile einer (einzig) Tabelle ab, die alle Objekte dieser (Objekt-)Klasse / -Kategorie enthält.

Modellierung z.B. über Entity-Relationship-Modell (ERM) → [2.1.3. das Entity-Relationship-Modell \(ERM\)](#)

Darstellung dann z.B. als Entity-Relationship-Diagramm → [2.1.4. Entity-Relationship-Diagramme \(ERD\)](#)

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 identifizierbar sein

kommen völlig gleichartige Objekte mehrfach vor und gibt es keinen inneren Schlüssel-Kandidaten, 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 zusammengesetzten 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 numerische 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 numerischen 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; Segmente können wieder segmentiert sein
durch (häufige / mehrfache) Indizierung sind Performance-Probleme bei größeren Daten-
Mengen zu erwarten
Manipulation der Daten durch indirekte 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 zusammengehörige Relationen verringern 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 Primärschlüssel
• Querverweis-Tabelle	beinhaltet die Beziehungen zwischen Master-Tabellen; als Schlüssel werden zumeist Kombinationen aus mehreren Spalten genutzt
• Transaktions-Tabelle	speichert die Interaktionen und deren Ergebnisse zwischen Master-Tabellen; als Schlüssel werden häufig automatisch generierte Ganzzahlen verwendet

referenzielle Integrität

Integrität wird hier als Korrektheit der Daten im Zusammenhang mit anderen Daten verstanden
bei der referenziellen Integrität wird gefordert, dass jede Referenz auch mit einem existierenden Eintrag / Objekt / Datum verbunden ist, d.h. es darf keinen Verweis auf ein nicht-existierendes 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 angegebenen Fremd-Schlüssel-Wert, d.h. zu jedem angegebenen Fremd-Schlüssel existiert in der referenzierten / verknüpften Tabelle ein gültiger Primär-Schlüssel-Wert.

Um dem Problem mit (versehentlich) gelöschten referenzierten 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 benutzt 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.

Solche Tabellen sind häufig die Ergebnisse von Normalisierungen (→ [2.2.2. Normalisierung](#)). Darunter verstehen wir grob betrachtet Verbesserungen der Redundanzen.

Die Tabelle2 ist somit eine Referenz-Tabelle.

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.

Tabelle1					
ID	Att1	Att2	Att3	...	AttN

klassische Daten-Tabelle (Rohdaten)

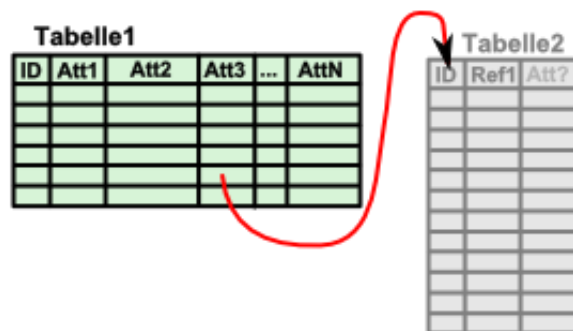
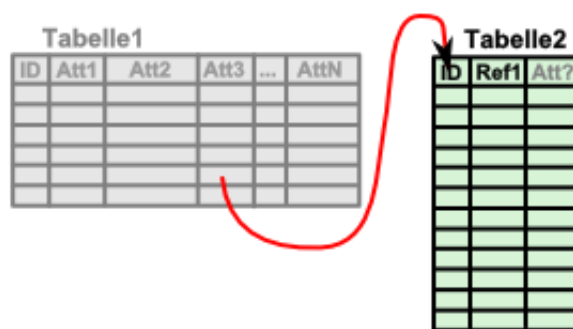
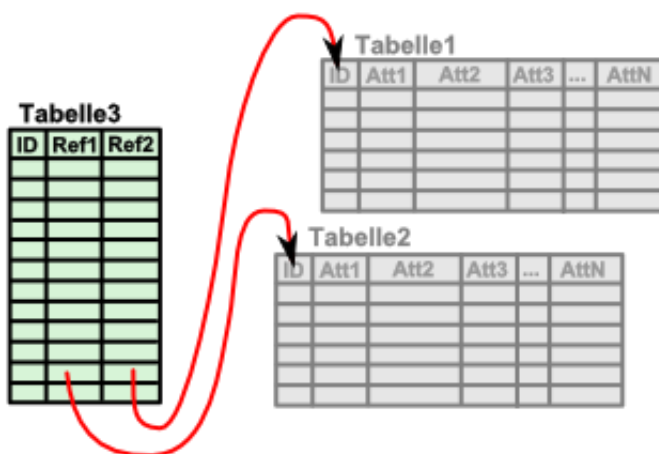


Tabelle mit einer Referenz auf eine andere



Referenz-Tabelle enthält Detail-Informationen



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 einer Datenbank. In relationalen Datenbanken 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-Administrator, 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 zusammengehörnde 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			
Tabellenkörper / 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 Relations-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:

$$\text{RelationsSchema} = (\text{Attribut } \{, \text{Attribut} \})$$

Ein Relationen-Schema ist also praktisch ein Tupel aus den Eigenschaften eines Objektes. Es stellt weiterhin den Werte- oder Eigenschaften- oder Attribute-Teil eines Datensatzes in der entsprechenden Tabelle (Relation) dar.

Die Relation (Tabelle) hat dann die folgende Notierung

$$\text{Relation}(\text{RelationsSchema}) \quad \text{oder} \quad \text{Relation}(\text{Attribut } \{, \text{Attribut} \})$$

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.

$$\text{RelationsSchema} = (\underline{\text{ID}}, \text{Attribut1}, \text{Attribut2}, \dots, \text{AttributN})$$

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.

$$\text{Tabelle} = (\underline{\text{ID}}, \text{Attribut1}, \text{Attribut2}, \nearrow \text{FremdID}, \dots, \text{AttributN})$$

Das folgende Relationen-Schema kann als Beispiel gelten:

$$\text{Schueler} = (\underline{\text{SID}}, \text{Name}, \text{Vorname}, \text{GebDatum}, \text{GebOrt}, \text{Geschlecht})$$
$$\text{Klasse} = (\underline{\text{KID}}, \text{Bezeichnung}, \text{KlassenLeiter}, \text{KlassenRaum})$$
$$\text{IstInKlasse} = (\underline{\text{IKID}}, \nearrow \text{KID}, \nearrow \text{SID})$$

Nach KEMPER und EICK werden Relationenschemata mit Daten-Typen notiert, Sie verwenden folgende Struktur (Typisierung **rot** hervorgehoben):

$$\text{Relationenschema} : \{ [\underline{\text{Attribut1: Typ}}, \text{Attribut2: Typ}], \dots, \text{AttributN: Typ} \}$$

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 bei KEMPER und EICK nicht.

Relationen werden dann in der Form:

$$\text{Relation} : \{ \text{Attribut1}, \text{Attribut 2}, \dots, \text{AttributN} \}$$

notiert.

Für praktische Anwendungen könnte man sicher eine Kombination der oben besprochenen Relationenschema-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 einen Zusatz für die Fremdschlüssel. Ein Relationen-Schema 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-Inkmente 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-Schlüssel 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 Primär-Schlüssel der anderen Relation als Fremd-Schlüssel
- **1 : n-Beziehungen:** die Relation mit der n-Kardinalität wird um den Primär-Schlüssel der 1-Kardinalität ergänzt
- **n : m-Beziehungen:** Anlage einer neuen Relation mit eigenen Attributen (dieser Beziehung) und den Primär-Schlüsseln beider beteiligter Relationen als Fremd-Schlüssel
- **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 CODD's Regel für eine ideale Datenbank

CODD'sche Regel definieren eine ideale Datenbank. Ursprünglich (1960er und 70er Jahre) waren es neun Regeln / Anforderungen. Später (1985) stellte er ein 12-Regel-System auf. 1990 hat CODD noch eine 0. Regel (sowie noch 6 weitere) hinzugefügt:

Regel 0 (Foundation-Regel):

Das System muss sowohl als Datenbank, als auch als Management-System relational strukturiert sein.

Regel 1 (Informations-Regel):

Alle Informationen in der Datenbank müssen auf eine einzige Weise dargestellt werden, d.h. als Werte in Tabellen.

Die Daten dürfen ausschließlich über einen Weg präsentiert werden – als Werte innerhalb von Spalten und innerhalb von Zeilen. Die Tabelle ist logische Gruppierung der zusammengehörigen Daten in Zeilen- und Spalten-Form. Jede Zeile steht für eine Sache / ein Objekt in der Tabelle, Jede Spalte beschreibt eine Eigenschafts-Art eines Objekts. Jeder Wert ist durch Zeile und Spalte adressiert.

Regel 2 (Zugriffs-Garantie-Regel):

Alle Daten sollten logisch durch die Kombinationen aus Tabellen-Name, Primär-Schlüssel und dem Spalten-Namen zugänglich sein. Dadurch ist eine eindeutige Identifikation und Erreichbarkeit gegeben.

Regel 3 (Nullwert-Regel):

Das Datenbank-Management-System muss fehlende und unzutreffende Informationen in systematischer Weise und unabhängig vom Daten-Typ als Null-Wert darstellen.

NULL wird als "unbekannt" interpretiert. NULL bedeutet die Abwesenheit eines Wertes und hat somit keinen Wert. NULL ist nicht identisch mit Null (0) oder einer leeren Zeichenkette. Jeder Wert mit NULL verglichen, ergibt NULL.

Regel 4 (Online-Katalog-Regel):

Die Datenbank muss einen relationalen Online-Katalog unterstützen, auf den autorisierte Benutzer über ihre normale Abfrage-Sprache zugreifen können.

Neben den Benutzer-definierten Tabellen (Daten-Tabellen) enthält ein Datenbank-System auch Daten über sich selbst (System-Tabellen (= System-Katalog, Data Dictionary)). Daten, die Struktur der Datenbank, deren Objekte und deren Beziehungen untereinander beschreiben, nennt man Meta-Daten.

Regel 5 (Subsprachen-Regel):

Die Datenbank muss mindestens eine Zugriff-Sprache / umfassende Kommunikations-Sprache unterstützen, die mittels einer linearen Syntax eine Funktionalität definiert, die z.B. Daten-Definitionen, Daten-Manipulationen, Daten-Integrität und (Datenbank-)Transaktions-Steuerung beinhaltet.

Die Zugriffs-Sprache muss sowohl direkt, als auch in / über Anwendungen nutzbar sein.

Regel 6 (Aktualisierungs-Regel (für Sichten / View's / Abfragen):

Die Darstellung der Daten muss auch über Abfragen (View's, Sichten) in verschiedenen logischen Kombinationen möglich sein. Alle Sichten, die (theoretisch) aktualisierbar sind, müssen vom System auch aktualisiert werden.

Sichten sind virtuelle Tabellen oder Abstraktionen der Quell-Tabellen (Daten- bzw. Objekt-Tabellen). Eine Sicht ist keine Duplikation der Original-Daten, sondern wird immer aktuell zusammengestellt. Werden Daten in einer Sicht geändert, dann sollte diese Änderung in die Quell-Tabelle zurückgeschrieben (- ein Update durchgeführt -) werden. Hierbei kann es u.U. zu Problemen kommen, da die neuen Daten die Integritäts-Regeln verletzen könnten..

Regel 7 (Operationen-Regel):

Das Datenbank-Management-System muss die zu einem Zeitpunkt gesetzten Operationen (Funktionen / Methoden) "Einfügen", "Aktualisieren" und "Löschen" unterstützen.

Zu den grundlegenden Operationen einer Datenbank gehören algebraische Funktionen (Selektion, Projektion, Join's) und Set-Operationen (Union, Intersektion, Division, Differenz). Mit diesen Funktionen in anderen (neuen) Tabellen neue Relationen erzeugt werden.

Regel 8 (physische / physikalische Daten-Unabhängigkeits-Regel):

Änderungen auf der physischen Ebene dürfen keine Auswirkungen auf das Anwendungs-Programm haben oder eine Änderung erfordern.

Die physikalische Schicht der (Gesamt-)Architektur wird auf die logische Schicht gemappt. Damit sind Benutzer und Anwendungen nicht mehr von der physikalischen Struktur abhängig. Die Implementierung der physikalischen Schicht ist Aufgabe des Datenbank (Datenbasis). Wenn sich die physikalische Basis der Datenbank (z.B. die Festplatten) oder die Zugriffs-Methode (z.B. neues RAID-System) ändert, dann hat dies keine Auswirkungen auf die Anwendungen.

Regel 9 (logische Daten-Unabhängigkeits-Regel):

Änderungen in der logischen Ebene dürfen keine Auswirkungen haben und auch keine Änderung im Anwendungs-Programm erfordern.

Regel 10 (Integritäts-(Unabhängigkeits-)Regel):

Es müssen Integritäts-Regeln definiert sowie unabhängig und getrennt vom Anwendungs-Programm sein. Das Changing Constraints muss erlaubt / möglich sein, ohne die Anwendung zu beeinflussen.

Regel 11 (Verteilungs-(Unabhängigkeits-)Regel):

Der Benutzer soll sich nicht über den Speicherort der Datenbank im Klaren sein, d.h. er braucht nicht zu wissen, ob die Daten an einem oder mehreren Standorten gespeichert sind.

Die Anwendungen sind unabhängig davon, ob die Daten durch direkten (lokalen) Zugriff oder über eine Join von verschiedenen Standorten bereitgestellt werden.

Regel 12 (Non-Subversions- / Nicht-Gefährdungs-Regel):

Wenn ein System eine Sprachen mit geringerem Umfang bereitstellt (Low-Level-Sprache), dann darf es keine Möglichkeit geben, die Integritäts-Regeln der umfangreicheren Sprache (High-Level-Sprache) zu untergraben.

Das Datenbank-Management-System muss Zugriffe über Maschinen-Sprache oder untergeordnete Schnittstellen verhindern.

Da die CODDSchen Regeln und auch SQL eine ternäre Logik voraussetzen und nutzen, ist Regel 3 die am stärksten diskutierte. Das liegt eben an dem Null-Wert, der keinen definierten Wahrheits-Wert hat.

1990 erweiterte CODD seine Regeln dann auch auf 18. Die zusätzlichen Regeln beziehen sich auf Kataloge, Daten-Typen (Domänen) und Authorisierungen. Selbst mit diesen lässt sich kein wirklich vollständiges Datenbank-System beschreiben, wie er später selbst einräumte. Besonders die Regeln 6, 9, 10, 11 und 12 (des klassischen Regel-Satzes) sind schwer zu erfüllen.

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

Eliminierung 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 wird hierfür die 0. Zeichen-Position genutzt. In unserem 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 Betriebssystems. 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 Umorganisation / (Neu-)Aufteilung der Daten einer Tabelle in der Form, dass sie keine vermeidbaren Redundanzen mehr enthält.

Normalisierung ist die Neu-Strukturierung der Daten einer Tabelle in mehrere Teil-Tabellen, um Dopplungen von Daten-Gruppen zu vermeiden. Durch Normalisierung soll die Datenbank optimiert werden und weniger Problemen bei Update's von Daten unterliegen.

Normalisierung ist die Aufteilung der Attribute eines Schema's mit dem Ziel Redundanzen zu verringern.

häufig wird der Begriff der Normalisierung 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)
-

Ziel der Normalisierung sind kleine (schmale, (tiefe)) Tabellen
Daten sind mit einer möglichst kleinen Redundanz zusammengestellt

Vorteile von normalisierten (schmalen) Tabellen:

- sehr geringe Redundanz (sehr geringer Speicher-Verbrauch; Daten werden praktisch immer nur einmalig gespeichert)
- verständliche Strukturen (Zusammenstellung von zusammengehörenden Daten in einer Relation)
- Eintragung neuer Daten kann vereinfacht werden
- Eingabe geht meist schneller
- i.A. schneller Zugriff auf relevante Daten
- ...

Nachteile:

- höherer Aufwand bei Zusammenstellung aller Daten zu einem Objekt (z.B. für eine flache Tabelle, Anzeigen, ...)
- höherer Aufwand bei der Analyse
- Fehler in den Daten haben oft weitreichende Wirkung (auch auf andere Objekte)

- fehlende Übersichtlichkeit (für viele Menschen nicht mehr überschaubar, Konsequenzen von Änderungen / Erweiterungen / Fehlern werden unterschätzt)
- ...

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

• Verarbeitungs-Geschwindigkeit zu erhöhen	Mehrfach-Verweise erfordern erneute Suchvorgänge 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
• online-Daten-Analyse	Big Data Data Warehouse

Von einer **funktionalen 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 *ID bestimmt funktional Name*
 ID → Vorname
 ID → {Name, Vorname}

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

Der **Identifikations-Schlüssel** ist ein Attribut, dessen Werte jeweils immer nur einmalig innerhalb des Attributes / der Spalte vorkommen.

Die Spalten "**Name**" oder "**Vorname**" eignen sich nicht als Schlüssel, da hier praktisch Doppelungen auftreten könnten. Dieser Effekt taucht ja schon bei relativ kleinen Datenbanken auf. Gemeint sind aber meist unterschiedliche Personen mit eben zufällig dem gleichen Namen.

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. Bei größeren Datenbanken würde das aber auch wieder nicht mehr funktionieren, da auch die Kombinationen aus häufig vorkommenden Vor- und nachnamen relativ häufig auftreten.

Von einer **vollen funktionalen 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.

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

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 einer Normalisierung bezeichnen wir als eine normalisierte Tabelle. Die Tabelle befindet sich dann in einer Normal-Form. Es werden mehrere Stufen der Normalisierung 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 verdeutlichen. 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 sicher, 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.

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

Für "Claudia Bauer" ergeben sich wirklich zwei Personen, so dass wir hier nicht löschen.

Übrig bleibt eine Tabelle, die wir in der Nullten Normalform sehen.

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

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.

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 Inkonsistenzen 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 bzw. ONF).

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. man sagt, sie habe die Non-First-Normal-Form (NF², Non-First-Form).

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 über 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 eliminieren, 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 wieder 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, das sind Probleme, die beim Löschen bzw. Einfügen von Daten in Tabellen auftreten können (→).

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 Relationenschema befindet sich in der 1. Normalform, 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.

In bestimmten Datenbank-Systemen, die recht ähnlich zu den relationalen Datenbank-Systemen funktionieren, werden auch Tabellen-orientiert Daten verarbeitet, bei denen ein Attribut eines Daten-Satzes mehrere Werte haben kann. Ein Beispiel dafür könnte ein gespeichertes Array von Werten sein. Dies widerspricht der 1. Normalform. Solche Datenbank-Systeme werden deshalb auch NonFirstNormalform-DBS (NF2-Systeme) bzw. MultiValue-DBS bezeichnet

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 irgendwann seine Datenbank um weitere Informationen erweitern, z.B. um eine spezielle Ansprech-Person. Mit solchen (Personen-)Netzwerken lassen sich vertrauensvolle, stabilere und unkomplizierte Geschäfts-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.

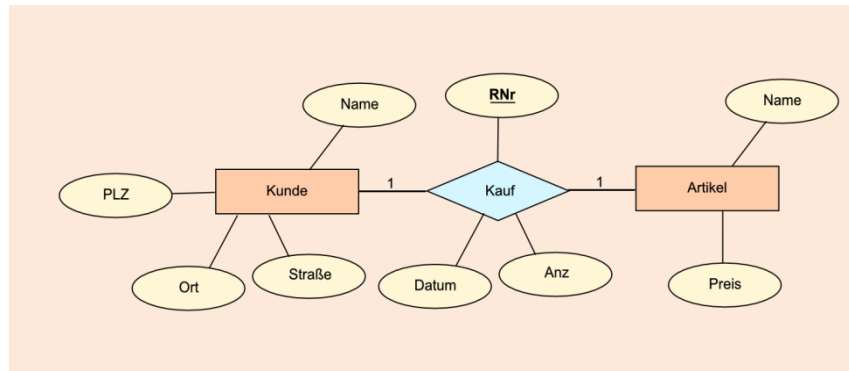
Beispiel:

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 Tabellenspalten 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.

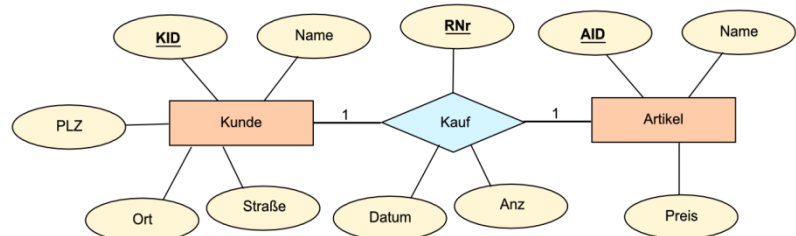
Die Daten zu den Kunden werden nun in einer separaten Tabelle geführt.

Es folgt das gleiche Vorgehen auch für die Artikel. Somit erhalten wir eine einfache, nicht-redundante – leicht zu pflegende – Artikel-Tabelle. Als Schlüssel hätten wir auch die Bezeichnung nutzen können. In Computer-Systemen ist es effektiver reine Zahlen 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 dazugehörige mono-thematische Tabellen, die sich für sich gesehen, in der 1. Normalform sind.

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 enthält 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 immer noch eine Anfälligkeit gegenüber DELETE-Anomalien (hier beim Löschen von referenzierten Datensätzen)

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. Geübte Datenbank-Planer haben da ihre Erfahrungen, was sich für den einen Anwendungsfall anbietet, oder was man lieber unter bestimmten Bedingungen unterläßt.

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 Relationenschema 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 Relationenschema 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 Relationenschema ist in der 3NF, wenn es sich in der 2NF befindet und kein Nicht-Schlüssel-Attribut Determinate ist.

3. Normalform (3NF):

Ein Relationenschema 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

2. Prüfen Sie Ihr Normalisierungs-Können unter der folgenden URL!

<http://edb.gm.fh-koeln.de/nf/start.jsp?action=wl>

für die gehobene Anspruchsebene:

3. Normalisieren Sie die folgende Relation (Zoo-Tiere) bis zur 3NF! Fehlende, korrigierte oder angepasste Daten dürfen Sie logisch passend ersetzen.

Nr.	Tier	Ort	Alter	Gewicht	Höhe	Länge
16	Elephant "Dumbo"; männlich	Gehege "Afrika"	19 a	5,8 t	310 cm	420 cm
17	Tiger "Indi"; Männchen	Raubtier-Gehege	6 a	112 kg	45 cm	136 cm
22	Löwe "Lois"	Raubtier-Gehege	7 Jahre	198 kg	1,18 m	2,0 m
23	Springmaus "Flipsi"; männl.	Wüsten-Haus	3 Mo	93 g	6,3 cm	10,8 cm
38	"Martha"; Löwin	Raubtier-Gehege	5 Jahre	167 kg	96 cm	1,6 m
50	weiblich; Tiger "Ursa"	Raubtier-Gehege	7,5 a	87 kg	43 cm	119 cm
72	Elephant "Elphi"; weiblich	Gehege "Afrika"	22 J	6300 kg	3,2 m	4,2 m
94	Nashorn "Bernd"; männlich	Gehege "Afrika"	14 a	2,49 t	1,65 m	365 cm

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

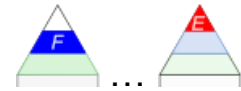
Schüler

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 Relation 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 Schlüsselwert 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

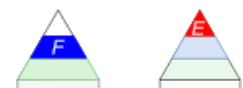
PersID	Lieblingsmusik
45	Hip-Hop
45	House
71	Hip-Hop
88	Schlager
88	Volksmusik
88	Country
94	House

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. Im Beispiel ist auch bei "mag" eine Mehrfach-Nennung zu erwarten, so dass die Tabelle ebenfalls weiter normalisiert werden sollte.

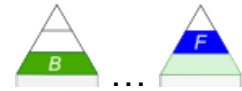
2.2.2.5.3. die 5. Normalform



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 sich die Werte 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 Attribut 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) muss auch ein geeignetes Verwaltungssystem enthalten sein.

Reisekosten

<u>Rechnungs- Nummer</u>	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

<u>Rechnungs- Nummer</u>	Reise-Datum	Kunden- Name	Kunden- Vorname	PLZ	Ort	Straße

Position

<u>Rechnungs- Nummer</u>	Kostenart	Anzahl

Kostenarten

<u>Kostenart</u>	Einzel- Vergütung



**Überführung in die
3. Normalform**

- Auslagern Personal-Daten und Orts-angaben

Reise

<u>Rechnungs- Nummer</u>	Reise- Datum	Personal- Nummer

Personal

<u>Personal- Nummer</u>	Name	Vorname	PLZ	Straße

Position

<u>Rechnungs- Nummer</u>	Kostenart	Anzahl

Kostenarten

<u>Kostenart</u>	Einzel- Vergütung

Orte

<u>PLZ</u>	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

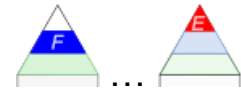
Fachbelegung

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



Relationsschema:

Schüler(SID, Name, Vorname, Geschlecht, Geburtsdatum, Geburtsort, ...)

allg: $R_1(X_1, X_2, \dots, X_n)$

Relation r ist endliche Menge $r \subseteq \text{Tup}(X)$

Menge aller Relationen über X : $\text{Rel}(X)$

Instanz von X : $r \in \text{Rel}(X)$

Relationsbezeichner: R

Relationsschema: $R(X)$ oder als Attributmenge: $R(\{A_1, A_2, \dots, A_k\})$

Stelligkeit der Relationsbezeichner: k

$R(A_1: \text{dom}(A_1), A_2: \text{dom}(A_2), \dots, A_k: \text{dom}(A_k))$

Relationsschema: $R(X)$

Schlüssel K ist $K \subseteq X$

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

Instanz I : $I(R_i) = r_i, 1 \leq i \leq m$

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" sondern **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 Realisierungen für eher kleine oder eben für eher große oder sehr große Datenbanken)
- schlecht verteilbar (benötigt / 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 (→ [2.3.6. Graph-Datenbanken](#))
- Mehrwert-Modell
- Dokumenten-orientiertes Modell (→ [2.3.5. Dokument\(en\)-orientiertes Daten\(bank\)-Modell](#))
- Datei-Systeme (in Betriebssystemen) (→ [2.3.1. hierarchisches Datenbank-Modell](#))
- Baum-Datenbanken (→ [2.3.2. Baum-artiges Daten\(bank\)-Modell](#))

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 realisiert
-

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 realisiert
-

Nachteile

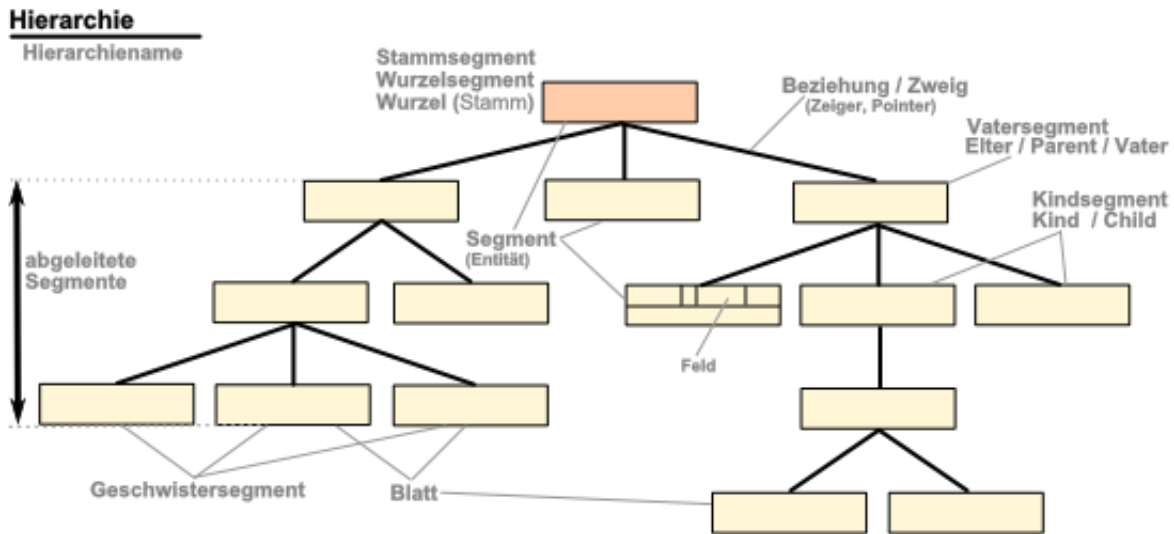
-

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:

schneller Zugriff auf einzelne Elemente, da nur wenige Entscheidungen / Abfragen gemacht werden müssen

Hierarchien (Firmen, Behörden, Organisationen, Komponenten (z.B. von Groß-Maschinen), Projekte) lassen sich praktisch 1 zu 1 umsetzen / abbilden

Nachteile:

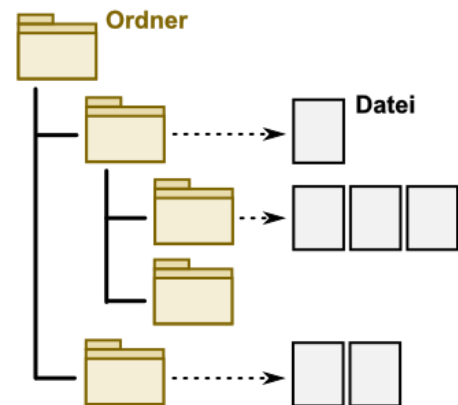
keine Beziehungen zwischen verschiedenen Bäumen

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

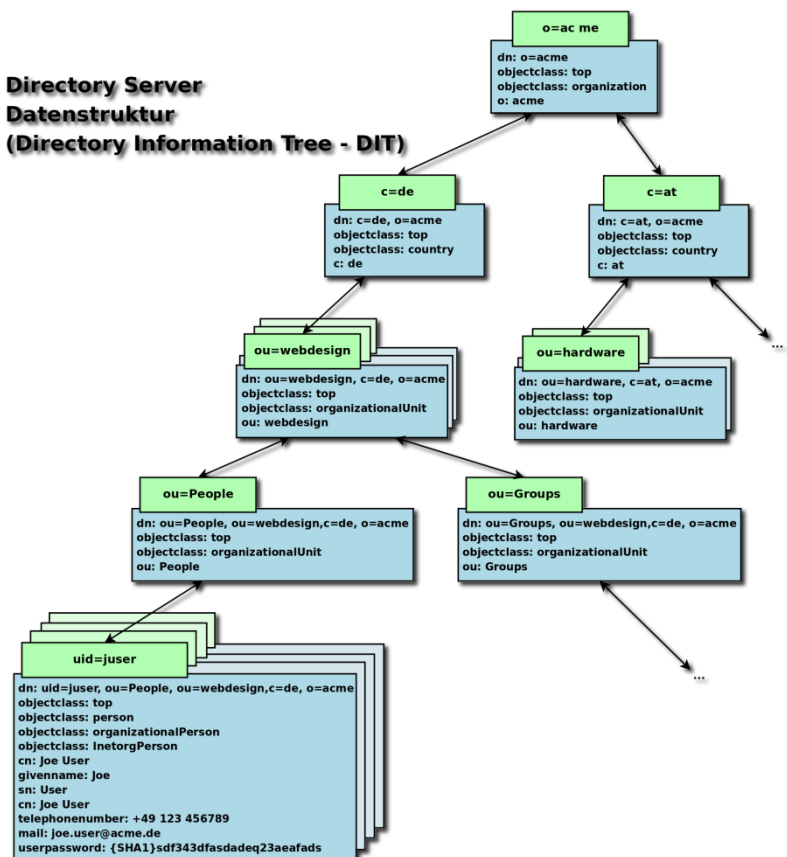
n : m-Beziehungen sind immer mit Redunzen verbunden

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))

Directory Server Datenstruktur (Directory Information Tree - DIT)



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

heute mit XML-Dateien wieder aktuell

XML (Extended 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 wegen 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, je nach dem, wieweit man Informationen braucht.

beliebig 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> <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
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

optimal für hierrarchische Systeme / Daten

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 3 Hierrarchie-Ebenen und mindestens (12 Blatt-)Elemente!)*

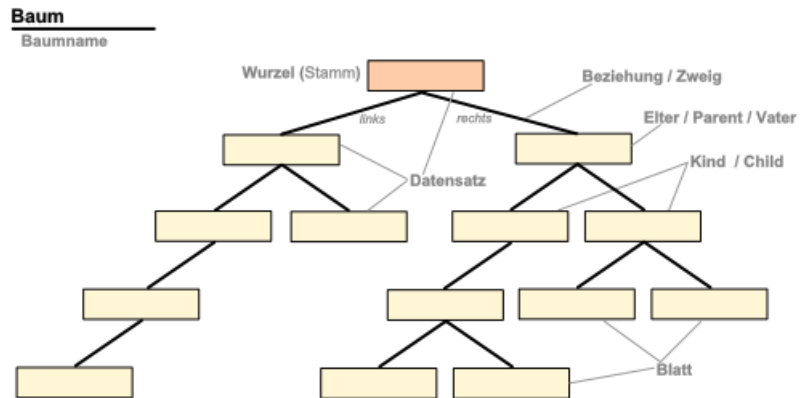
2.3.2. Baum-artige Daten(bank)-Modelle

2.3.2.1. binäre Bäume

Eltern-Element hat max. 2 Kinder

Spezial-Fall der hierarchischen Struktur

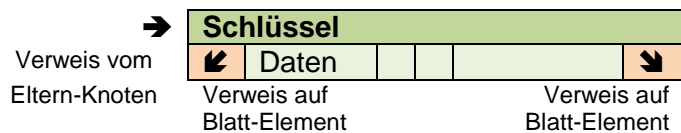
im Allg. gleichartige Datensätze



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

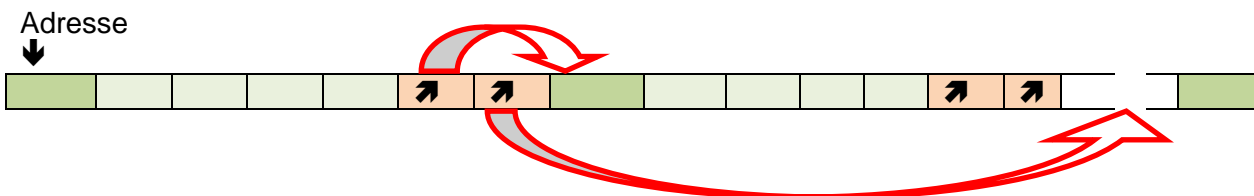
In der Programmierung besteht dann ein Baum-Element immer aus drei Teilen. Das ist (-meist in der Mitte dargestellt -) der Inhalt. Dieser kann für sich weiter strukturiert sein. Hier hinein würde z.B. ein Schlüssel gehören.

Daneben gibt es zwei Verweisfelder (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 des Baumes abgefragt und entsprechend dem Inhalt (ein Verweis oder eben keiner) weiter verfahren.

Anordnung der Struktur-Elemente eines Binär-Baum's im Speicher bzw. in einer Datei



Vorteile:

sehr effektive Algorithmen verfügbar

Nachteile:

Zeiger-Strukturen sind in der Programmierung etwas undurchsichtig

2.3.2.2. weitere Baum-Strukturen

Bäume mit variabler Blatt-Anzahl

B-Bäume

balancierte Bäume

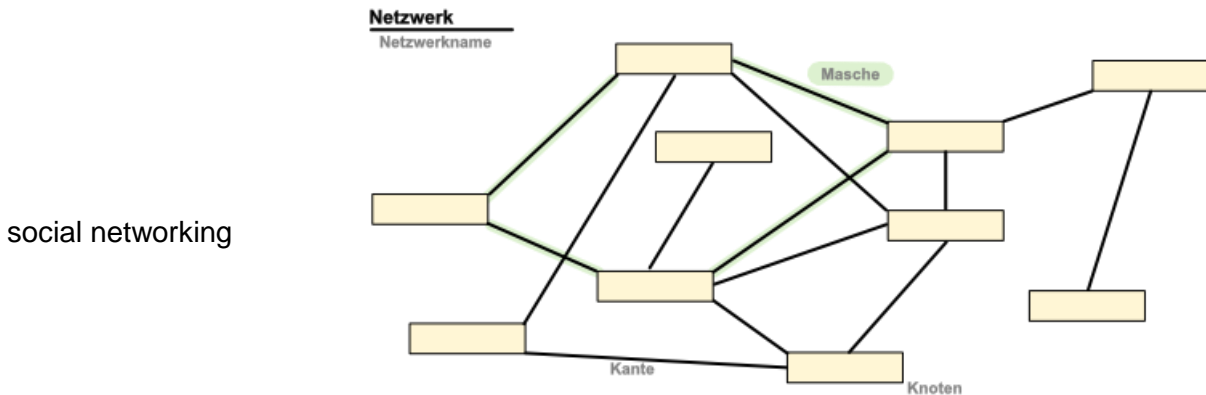
B+-Bäume

2.3.3. netzwerkartiges Daten(bank)-Modell

hohe Ausfall-Sicherheit

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

moderne Stichworte: Cloud's, Blockchain's (z.B.: BitCoin)



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 (unberechtigte 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 Ausfallsicherheit; → erhöhter Kommunikations-Aufwand zwischen Knoten
• keine zentrale Verwaltungs-Instanz	Rechner verwalten sich gegenseitig; → erhöhter Kommunikations-Aufwand zwischen Knoten
• ständige Verfügbarkeit	Gesamtsystem muss auch bei Teil-Ausfällen noch voll verfügbar sein / sollte nicht abschaltbar sein
• 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 Zugriffssprachen (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 Eigenschaftensystem

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

Definition(en): CAP-Theorem

Das CAP-Theorem besagt, dass von den gewünschten (drei) Eigenschaften eines verteilten System's immer nur zwei maximiert werden können.

C ... Konsistenz (Consistency); A ... Verfügbarkeit (Availability); P ... Partions-Toleranz (Partition tolerancy)

Definition(en): ACID

ACID ist die Abkürzung für gewünschte Eigenschaften für zusammengehörende Arbeitsschritte (einer Transaktion) in einem Datenbank-System.

A ... Atomizität (Atomicity); C ... Konsistenz (Consistency); I ... Isolation; D ... Dauerhaftigkeit (Durability)

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 / können dann durch Algorithmen-Teile für die Spezialitäten der untergeordneten Objekte erweitert oder geändert (überschrieben) werden

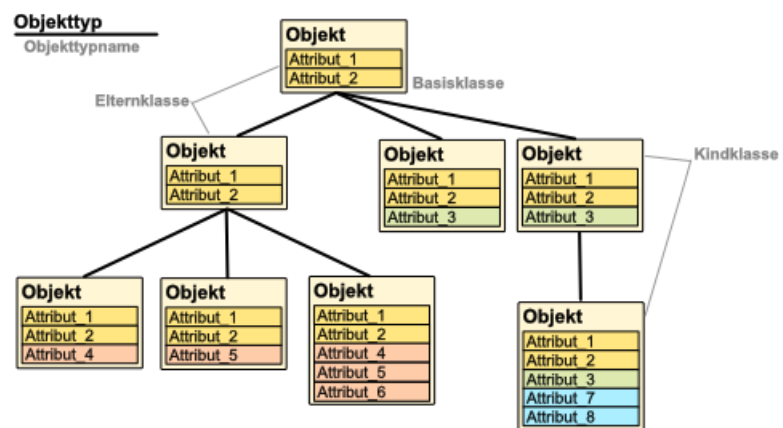
wegen besonderer Struktur wird auch den Datenbank-Management-Systemen (DBMS) auch ein O für Objekt vorgesetzt (ODBMS)

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

reale 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-orientiert, wenn ihr grundlegendes Konzept auf Objekte, 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 Methoden (getter/setter) ermittel- bzw. manipulierbar
• Vererbung	Eigenschaften und Methoden 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

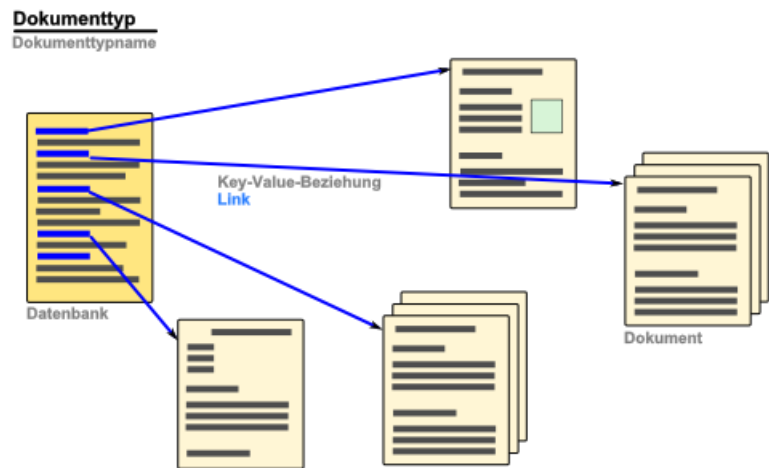
basiert auf Lotus Notes

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

JavaScript Object Notation

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. Hierarchien) 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.

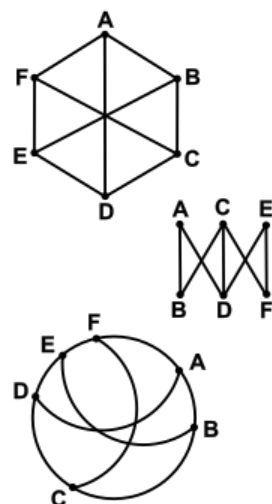
Die Knoten selbst haben i.A. keine Attribute. Es existiert auch kein Index über alle Knoten. Den Kanten (Beziehungen) können Attribute zugeordnet werden. Oft sind sie durch die Länge der Kante repräsentiert. Als Attribute eignen sich z.B.:

Größe
Länge / Abstand
Kosten
Durchfluss-Mengen / -Kapazitäten
Übergangs-Raten
...

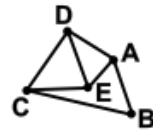
Sind die Kanten-Länge kein Repräsentant für ein Attribut, dann kann ein Graph auf sehr unterschiedliche Art und Weise dargestellt werden. Man nennt den Graph dann **isomorph**.

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.



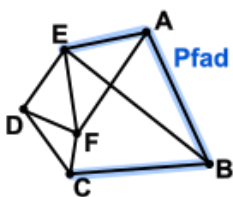
Kreuzen sich die Kanten nur in den Knoten, dann nennt man den Graph auch **planar**. Die Räume einer gewöhnlichen Wohnung könnten durch einen planaren Graphen dargestellt werden. Die Zimmer liegen dann in einer Ebene. Überkreuzen sich Kanten außerhalb der Knoten, dann muss eine zusätzliche Ebene (Etage)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



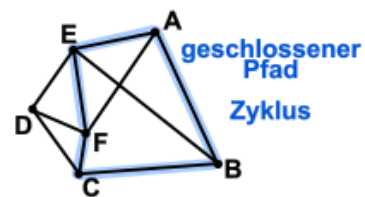
Die Anzahl der Kanten, die sich in einem Knoten treffen steht für den **Grad des Knoten's**. Der Mittelwert aller Knoten-Grade nennt sich Branching-Faktor.

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 war es bis vor wenigen Jahren noch nicht möglich, dass ein Computer einen erfahrenen Go-Spieler schlagen kann.



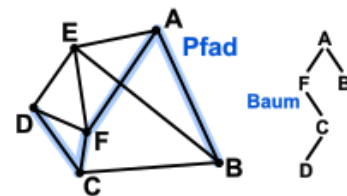
Eine endliche Folge von Knoten und Kanten, die wieder in einem Knoten enden, werden **Pfad** genannt. Bei einem **geschlossenen Pfad** ist der Start- und End-Knoten identisch.



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.



HAMILTON-Kreise sind z.B. bei der Planung von Liefer-Routen bedeutsam.

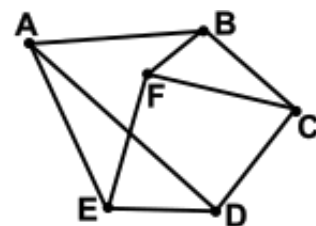
Das HAMILTON-Problem ist NP-vollständig.

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.

Aufgaben:

1. *Geben Sie zum nebenstehenden Graphen drei (einfache) Pfade und einen geschlossenen an!*
2. *Skizzieren Sie sich den Graphen ab! Prüfen Sie, ob ein HAMILTON-Kreis existiert! Geben Sie diesen ev. an!*
3. *Prüfen Sie, ob ein EULER-Kreis existiert! Geben Sie diesen ev. an!*
4. *Ist es möglich, eine Traverse durch den Graphen zu legen? Begründen Sie Ihre Meinung!*



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 um semantic web

Daten werden hier weiterhin 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 sogenannte 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

Definition(en): Graph-Datenbank-Management-System
Ein Graph-DBMS ist ein DBMS, dass sich auf Datenstrukturen / Daten-Modellen stützt, die Sachverhalte als Knoten und Kanten (Verbindungen zwischen den Knoten) darstellen.

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)**

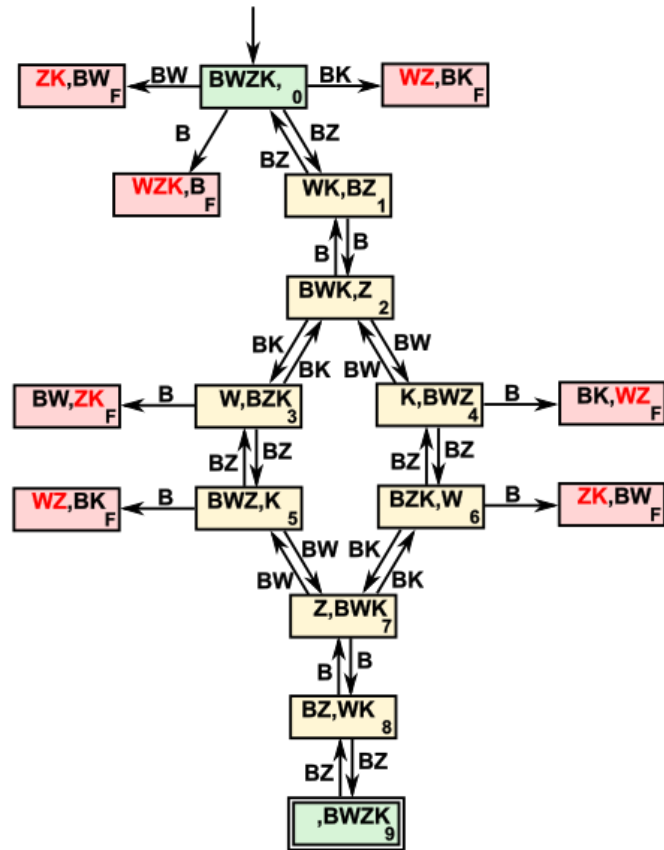
**bei einem Boot-Wechsel von Bauer und Ziege schreiben wir an die Kante:
BZ und dann den neuen Knoten (Situation, Zustand)**

also: BWZK// \xrightarrow{BZ} 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?

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 in der 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.

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:

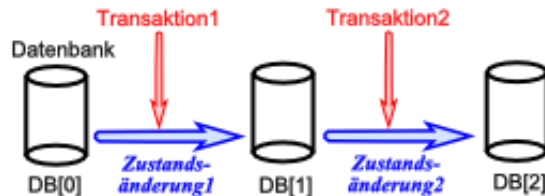
- neo4j
- OpenLink Virtuoso

2.4. Transaktionen und das Transaktions-Modell



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

transformieren eines gegebenen (eindeutigen / definierten) Datenbank-Zustand' in einen neuen – ebenfalls definierten



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 bei Kontrollen ev. 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 erzeugen

bei einem Buchungs-Versuch eines Sitzplatzes in einem Flugzeug muss eine offene Abfrage durch den einen Client (z.B. ein Reisebüro) diesen Platz solange blockieren, bis entweder gebucht oder nicht gebucht wird. Erst danach darf der Sitzplatz für andere Clients (andere Reisebüro's) 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
entweder werden alle Schritte vollzogen oder keiner |
| • Consistency
Konsistenz | jede Transaktion ergibt ausgehend von einem konsistenten Zustand wieder einen konsistenten (Bedingungs-gerechten) Zustandsübergang in der Datenbank
ist ein Teilschritt der Transaktion nicht möglich, dann muss die gesamte Transaktion abgebrochen werden und die vorherigen Schritte zurückgenommen werden
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
ein Zugriff auf die beteiligten Datensätze durch einen weiteren Nutzer |

darf während einer Transaktion nicht möglich sein
das Ergebnis einer Transaktion darf für andere Nutzer erst nach
einem erfolgreichen Abschluss sichtbar sein

- **Durability**
Dauerhaftigkeit nach Vollzug der Transaktion (oder eben keiner) ist der neue Zustand persistent (dauerhaft)
Änderungen nach einer Transaktionen sind gespeichert und müssen auch nach einem Absturz nachvollziehbar / erhalten sein
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.

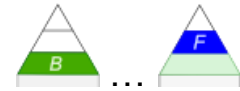
write-ahead-log-Regel:

Die Protokoll-Informationen für eine Transaktion muss physikalisch in die Protokoll-Datei geschrieben werden, bevor die Transaktion physikalisch die Veränderungen vornimmt.

Beispiel für eine Transaktion (Überweisen eines Geld-Betrag's)

1. Lesen des Konto-Stand's des Überweisenden
2. Vermindern des Konto-Stand's um den (Überweisungs)Betrag
3. Schreiben des neuen Konto-Stand's des Überweisenden
4. Lesen des Konto-Stand's des Ziel-Konto's (Empfänger)
5. Erhöhen des Konto-Stand's um den (Überweisungs-)Betrag
6. Schreiben des neuen Konto-Stand's des Empfänger's

2.5. Daten-Verteilung und -Sicherheit in Datenbanksystemen



Funktions-Prinzipien

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

Gründe für die Verteilung von Daten

- **große Datenmengen** z.B. auf mehrere Server, um schneller auf relevante Daten schneller zugreifen zu können
- **Ausfallsicherheit** Schaffung von Redundanzen
parallel arbeitende Server
- **schneller (regionaler) Zugriff**
Last-Balanzierung lokale Server für schnellere Zugriffe
mehrere Server, die Anfragen unter sich verteilen

Typen / Architekturen der Verteilung

- **Shared Memory** gemeinsam genutzter Speicher von mehreren Verarbeitungseinheiten (z.B. CPU's einschließlich eigener Datenträger)
- **Shared Disk** Verarbeitungseinheiten (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

- | | |
|--|--|
| <ul style="list-style-type: none">• Fragmentierung Sharding | verteilen unterschiedlicher Daten oder von Daten-Teilen auf mehrere Server; verbessert Performance und sorgt für gleichmäßige / angepasste Lasten-Verteilung |
| <ul style="list-style-type: none">• Replikation | verteilen der gleichen Daten auf mehrere Server; verbessert die Daten- und Ausfall-Sicherheit |

Verfahren der Daten-Verteilung (Partitionierung)

- | | |
|--|--|
| <ul style="list-style-type: none">• Round Robin Verfahren | <p>Verteilung neuer Datensätze (Tupel) oder Daten-Blöcke immer reihum auf die einzelnen Server</p> <p>Vorteile:</p> <ul style="list-style-type: none">• gute Last-Verteilung <p>Nachteile:</p> <ul style="list-style-type: none">• kein Server hat den gesamten Datenbestand• Ausfallsicherheit nur begrenzt gegeben |
| <ul style="list-style-type: none">• Partitionieren | <p>Aufteilung der Daten nach bestimmten Kriterien</p> <p>Vorteile:</p> <ul style="list-style-type: none">• gute Kombination von Zugriffszeit und Speicherort <p>Nachteile:</p> <ul style="list-style-type: none">• kein Server hat den gesamten Datenbestand• Ausfallsicherheit nur begrenzt gegeben• ev. schlechte Lasten auf den Servern |
| <ul style="list-style-type: none">• Hash-basierte Verfahren | <p>aus den (einzelnen) Daten(sätzen) wird ein Hash berechnet und dieser Hash-Wert wird dann zur Partitionierung / Verteilung der Daten genutzt</p> <p>Vorteile:</p> <ul style="list-style-type: none">• gute Last-Verteilung <p>Nachteile:</p> <ul style="list-style-type: none">• hoher Rechenaufwand• zusätzliche Daten (Hash-Wert) |

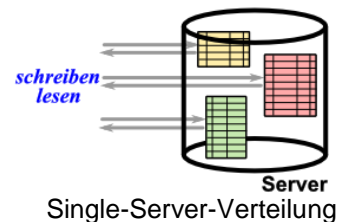
Kombinationen

- 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

Single-Server-Verteilung

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)



Shard-Verteilung / fragmentierte Verteilung

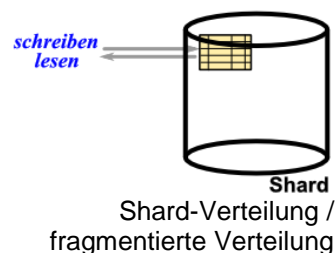
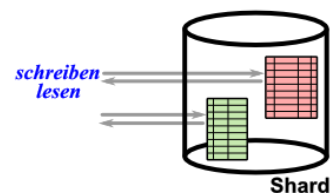
Hier besteht die Datenbank aus mehreren – voneinander unabhängigen Datenbank-Teilen, die quasi als eigenständige Datenbanken fungieren. Auf jeder der Teil-Datenbanken (Fragmente) wird separat zugegriffen. Shard ist die englische Bezeichnung für ein Fragment

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.

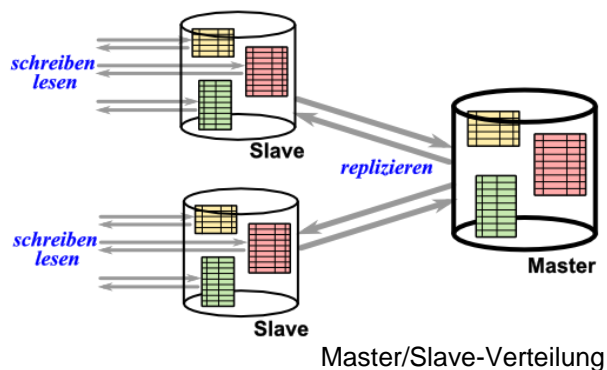


Master/Slave-Verteilung

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.

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 aktuellen Gesamt-Daten-Bestand verfügen.

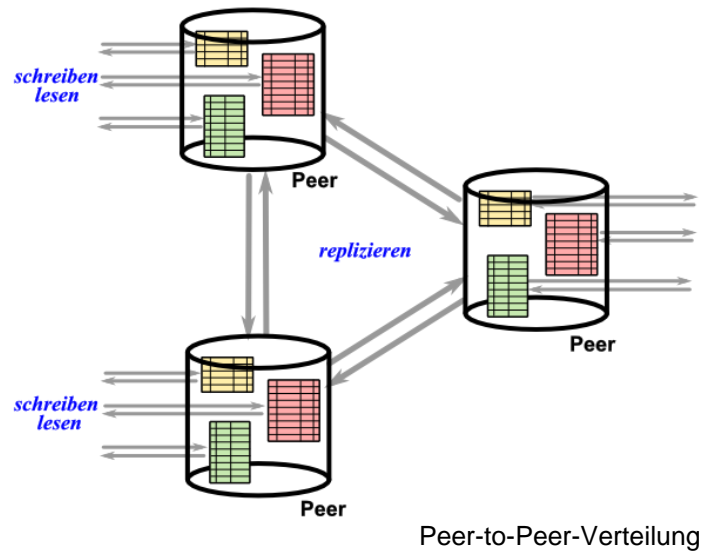


Peer-to-Peer-Verteilung

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



Definition(en): Dauerhaftigkeit / Durability

Dauerhaftigkeit ist eine Eigenschaft eines Datenbank-System's gewünschte Änderungen an den Daten erfolgreich festzuhalten (z.B. auch nach einem System-Absturz).

Definition(en): Replikation

Replikation ist das redundante Abspeichern der Daten eines DBMS auf verschiedene Knoten (eines Netzwerkes).

Links:

<http://www.datenbanken-verstehen.de/>

<http://www.oszhandel.de/gymnasium/faecher/informatik/datenbanken/index.htm>

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:

// lern-soft-projekt: drews (c,p) 1997 - 2023 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 - 2023 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)

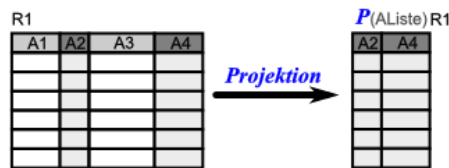
- **Selektion**

Auswahl von Zeilen (Tupeln) aus einer Relation aufgrund einer oder mehrerer Bedingungen



- **Projektion**

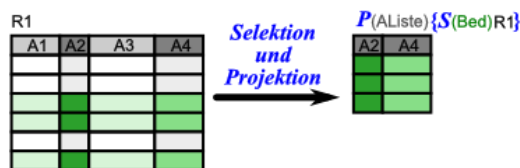
Auswahl (/ Einschränkung) von Spalten (Attribute) aus einer Relation aufgrund einer Auswahlliste



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

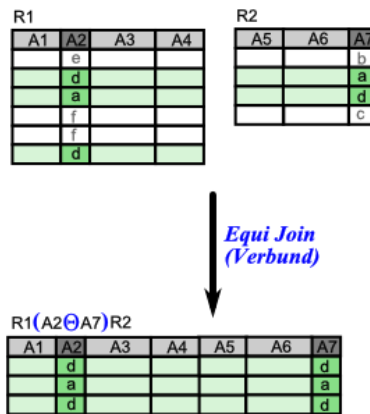


-
-

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