

# **Datenbanken I (T2INF2004)**

## **Foliensatz 1: Einführung**

**Uli Seelbach, DHBW Mannheim, 2023**

**Foliensatz freundlicherweise zur  
Verfügung gestellt von Mirko Schick**



# Datenbanksysteme

## Wozu?

- Die meisten Anwendungen benötigen dauerhaft verfügbare Daten und das auch noch effizient
- Die Anzahl der Daten nimmt immer weiter zu
- Viele Anwendungen brauchen große Mengen von Daten.
- Zum Teil fallen die großen Datenmengen auch einfach an (z.B. bei Verkaufstransaktionen, Interaktion mit Webseiten).
- Datenbanken sind das Speichermedium für strukturierte Informationen (z.B. Tabellen).
- Mit SQL kann man auch relativ komplexe Auswertungen der Daten kompakt aufschreiben.
- Seit Jahrzehnten bewährte Technologie
- SQL ist aber mehr als `SELECT FROM WHERE`. Investieren Sie jetzt Zeit!



# Datenbanksysteme

## Wozu?

- SQL bleibt modern.
- Andere Frameworks und Technologien kommen und gehen, SQL gibt es schon lange, und es ist kein Ende abzusehen. SQL entwickelt sich weiter.
- Da SQL eine deklarative Sprache ist, können DBMS (SQL-Implementierungen) leicht an neue Technologien angepasst werden.
- Einfache und übersichtliche Struktur.
- Transaktionen und die Synchronisation paralleler Zugriffe werden schon bei einfachen Web-Anwendungen wichtig.
- Datenbanken bieten dafür wichtige Unterstützung, aber nicht immer vollautomatisch, so dass Wissen (z.B. aus dieser Vorlesung) wichtig ist



# Datenbanksysteme

## Motivation?

Folgende Probleme/Problemgebiete bereiten oft Kopfzerbrechen:

- Unkontrollierte Datenredundanz
- Daten filtern, sortieren, aggregieren, umformatieren, ...
- Verknüpfung von Daten in verteilten Dateien
- Mehrbenutzerbetrieb
- Datensicherung
- Datensicherheit
- Datenintegrität / Datenkonsistenz
- Entwicklungsaufwand, um diesen Problemen entgegenzuwirken



**Gibt es „kontrollierte“  
Datenredundanz? Ist das sinnvoll?**



**Jedes Backup ist z.B. auch Redundanz. Oder eine  
Berechnung, die gespeichert wird.**



# Datenbanksysteme

## Lösung? Eine Datenbank! Wo genau?

### Einsatzformen:

- In Unternehmen (Produktionsplanung, Verwaltung, Buchungssysteme, ...)
- Wissenschaft
- Privat (Mobiltelefon, moderne Fernseher, PC)

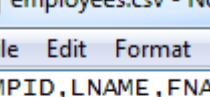
### Architektur:

- Auf privatem Einzel-PC
- spezieller Server
- Clusterverbund
- Verteilt

# Datenbanksysteme

## Wie sieht so etwas aus?

## CSV:



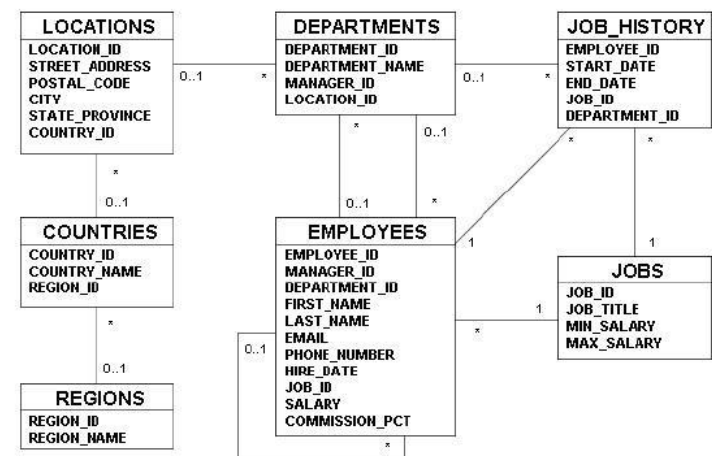
employees.csv - Notepad

File Edit Format View Help

EMPID,LNAME,FNAME,BDATE  
1236,Smith,John,05-01-1962  
2398,Adams,John,03-22-1976  
4534,Johnson,Mary,10-07-1971  
7834,Kirby,Frank,05-27-1978  
9823,Harris,Kathy,11-18-1982  
9875,Jones,Bill,01-25-1953

## Relationales Datenbanksystem:

## Human Resources (HR) Schema





# Datenbanksysteme

## Was macht der DBA?

- Datenstruktur („Schema“) definieren
- Integrität prüfen
  - besser gesagt: Monitoring der Instrumente, die Integrität garantieren
- Systembetrieb überwachen
  - Schnittstellen zur DB monitoren
  - DB selbst monitoren
- Performance optimieren
  - Hardware: vertikale Skalierung
  - Software: Abfrageoptimierung und Systemkonfig
  - HW+SW: horizontale Skalierung
- SW aktualisieren
- Rechteverwaltung umsetzen (nicht definieren!)
- Sich um Datensicherung kümmern



# Datenbanksysteme

## Immer sinnvoll?

- Unnötige Zusatzkosten durch: [Emasri, Navathe]
  - zu hohe Universalität eines Datenbanksystems
  - Mehrkosten für die Umsetzung der Punkte: Sicherheit, Integrität, Wiederherstellung, Hardware, Software, Schulungen, ...
- Datenbankadministratoren / Designer könnten Fehler machen oder Anwendungen auf der DB könnten fehlerhaft implementiert sein.
- Verwendung regulärer Dateien unter folgenden Umständen ggf. wünschenswert:
  - Extrem hohe Echtzeitanforderungen
  - Gleichzeitiger Zugriff mehrerer Benutzer ist nicht erforderlich
  - Datenbank und Anwendung sind einfach, wohl definiert und voraussichtlich frei von zukünftigen Änderungen



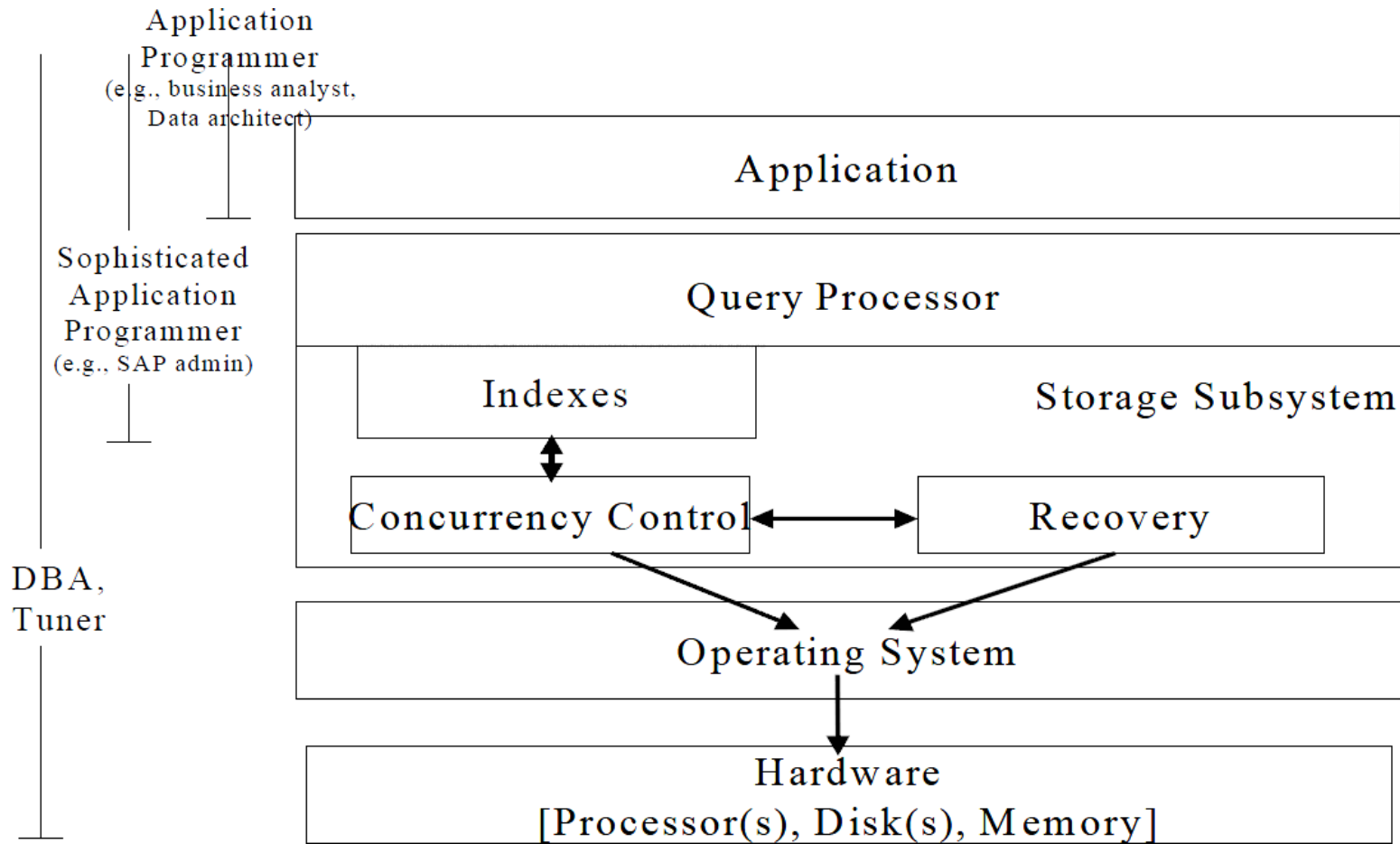
**Ihre Meinung zu den Punkten?**





# Datenbanksysteme

## Wer macht was?





# Datenbanksysteme

„Performance“? Komplexes Thema, eigene Vorlesung nötig!

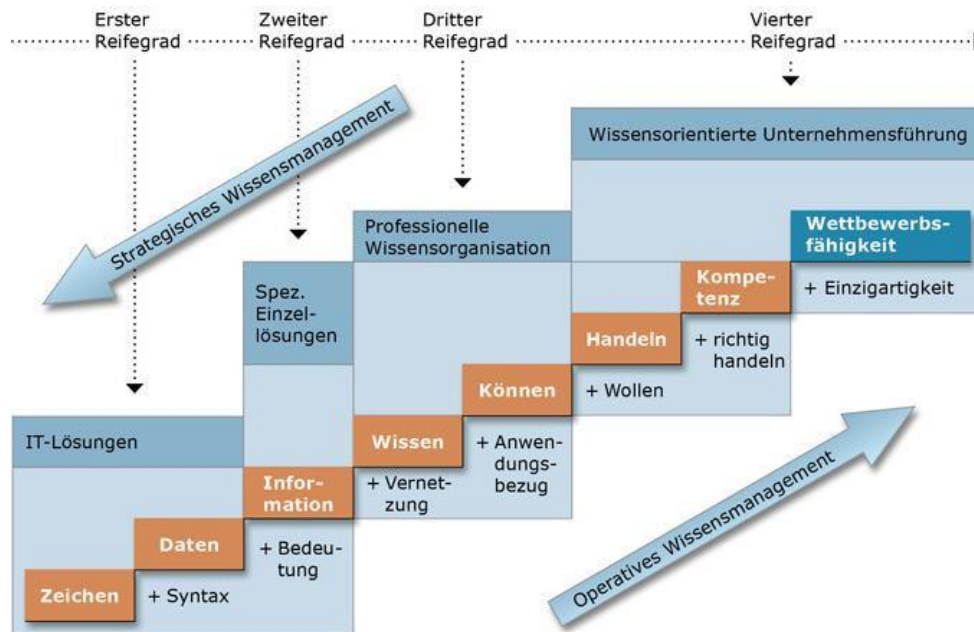


# Begriffsdefinitionen

## Daten, Informationen, Wissen?



Die Wissenstreppe



Quelle: North, 2002

### ■ Zeichen:

- zusammenhangslos und alleinstehend
- Von einem bestimmten Datentypen
- z. B. „fkeüf wpefk“

### ■ Daten:

- Zusammenhängende Daten durch Syntax und zielorientierter Semantik
- z. B. „50 km/h“

### ■ Informationen:

- Wenn Daten eine Bedeutung beigemessen werden kann
- z. B. obige Daten auf die Frage wie schnell man in geschlossenen Ortschaften standardmäßig fahren darf



?

**Schufa-Datenbanken,  
Verkehrszentralregister,  
dualis.dhbw.de**

**Liegen in solchen Datenbanken  
nun Daten oder Informationen?**

- Zeichen:
  - zusammenhangslos und alleinstehend
  - Von einem bestimmten Datentypen
  - z. B. „fkeüf wpefkW“
- Daten:
  - Zusammenhängende Daten durch Syntax und zielorientierter Semantik
  - z. B. „50 km/h“
- Informationen:
  - Wenn Daten eine Bedeutung beigemessen werden kann
  - z. B. obige Daten auf die Frage wie schnell man in geschlossenen Ortschaften standardmäßig fahren darf



# Begriffsdefinitionen

## Informationssystem, Datenbanksystem

- Ein **Informationssystem** ist ein soziotechnisches System, das die Deckung von Informationsnachfrage zur Aufgabe hat. Es handelt sich dabei um ein so genanntes Mensch/Aufgabe/Technik-System, das Daten (bzw. Informationen) produziert, beschafft, verteilt und verarbeitet.  
[\[http://de.wikipedia.org/wiki/Informationssystem\]](http://de.wikipedia.org/wiki/Informationssystem)
- Ein [...] **Datenbanksystem** [...] ist ein System zur elektronischen Datenverwaltung. Die wesentliche Aufgabe eines DBS ist es, große Datenmengen effizient, widerspruchsfrei und dauerhaft zu speichern und benötigte Teilmengen in unterschiedlichen, bedarfsgerechten Darstellungsformen für Benutzer und Anwendungsprogramme bereitzustellen [\[http://de.wikipedia.org/wiki/Datenbanksystem\]](http://de.wikipedia.org/wiki/Datenbanksystem)

?

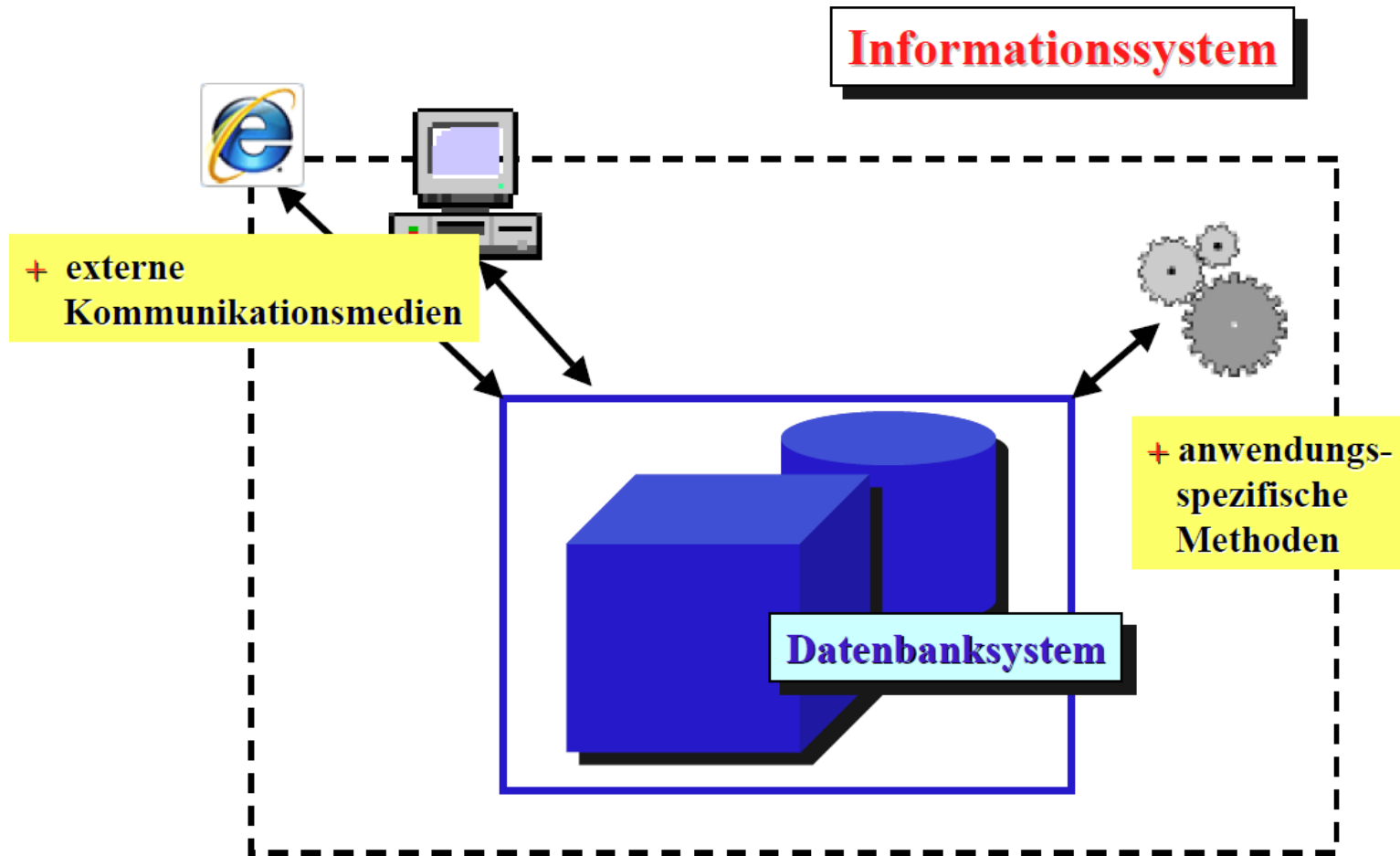
Was ist der Unterschied?

?

Ist die Bing-Suchmaschine ein IS?

# Begriffsdefinitionen

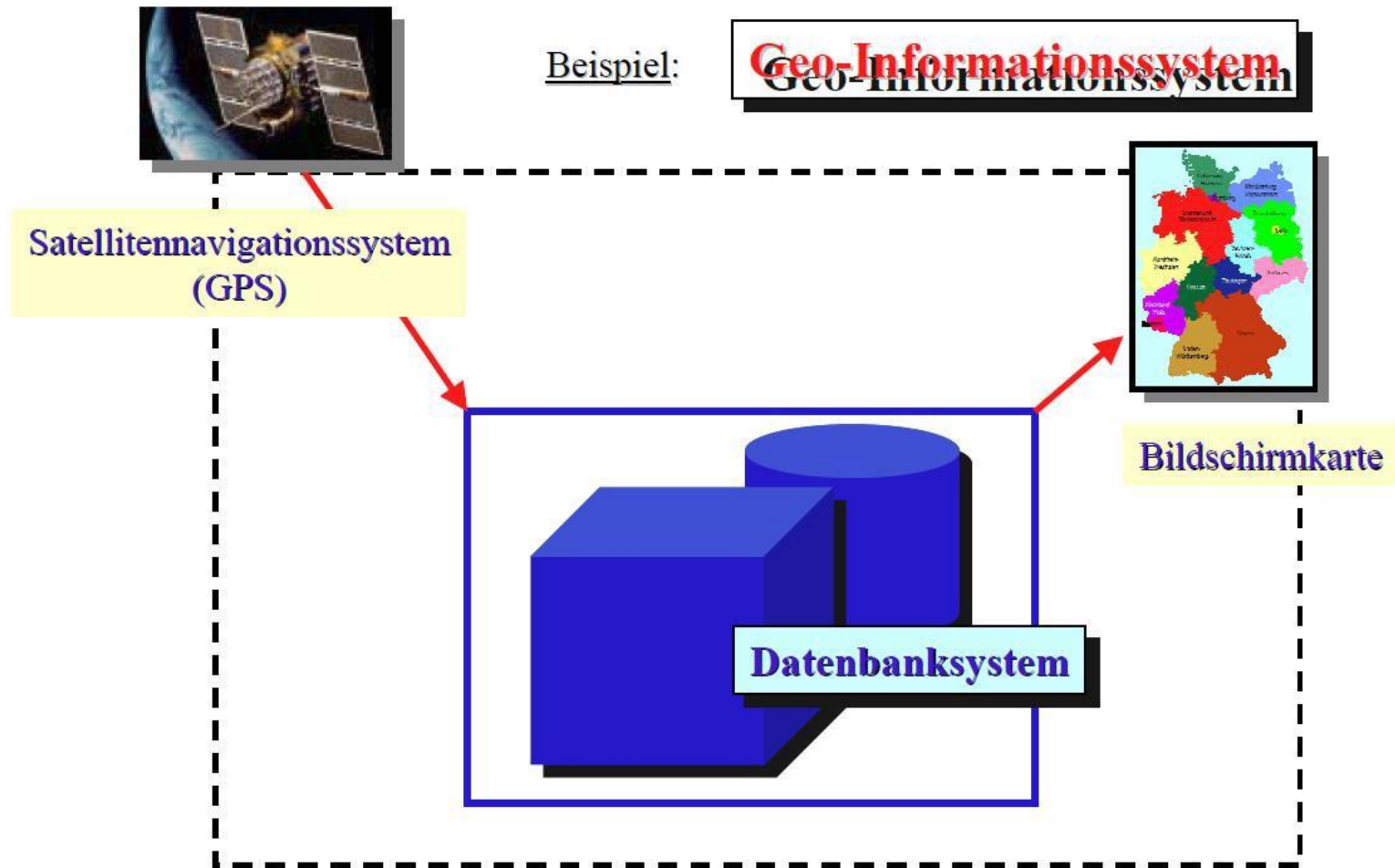
## Informationssystem





# Begriffsdefinitionen

## Informationssystem, Beispiel





# Datenbanksysteme

## Geschichte

- 1968: IMS – erstes kommerzielles hierarchisches DBMS (IBM)
- 1970: E. Codd: relationales Modell
- 1976: ERM von Peter Chen (Modellierung)
- 1979: Oracle Database (erstes kommerz. relationales DBMS)
- 1986: SQL erstmalig standardisiert
- 1995 – 1997: MySQL kostenlos / open source





# Datenbanksysteme

## Geschichte

- 1968: IMS – erstes kommerzielles hierarchisches DBMS (IBM)
- 1970: Edgar F. Codd (IBM): relationales Modell
- 1970er: System R, erste Implementierung des rel. Modells; SEQUEL
- 1976: ERM von Peter Chen (Modellierung)
- 1979: Oracle Database (erstes kommerz. relationales DBMS)
- 1983: IBM DB2, kommerzieller Nachfolger von System R
- 1986: SQL erstmalig standardisiert
- 1995 – 1997: MySQL kostenlos / open source
- 2000er: NoSQL-Datenbanken kommen auf  
SQL-Standard mehrfach erweitert

# Datenbanksysteme

## Aufbau

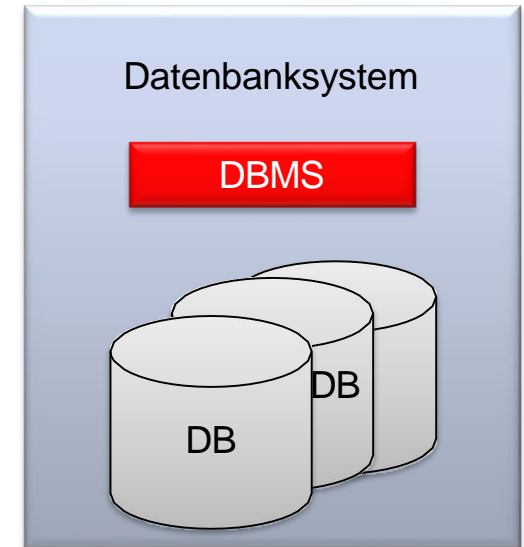
Ein Datenbanksystem (DBS) besteht aus 2 Komponenten

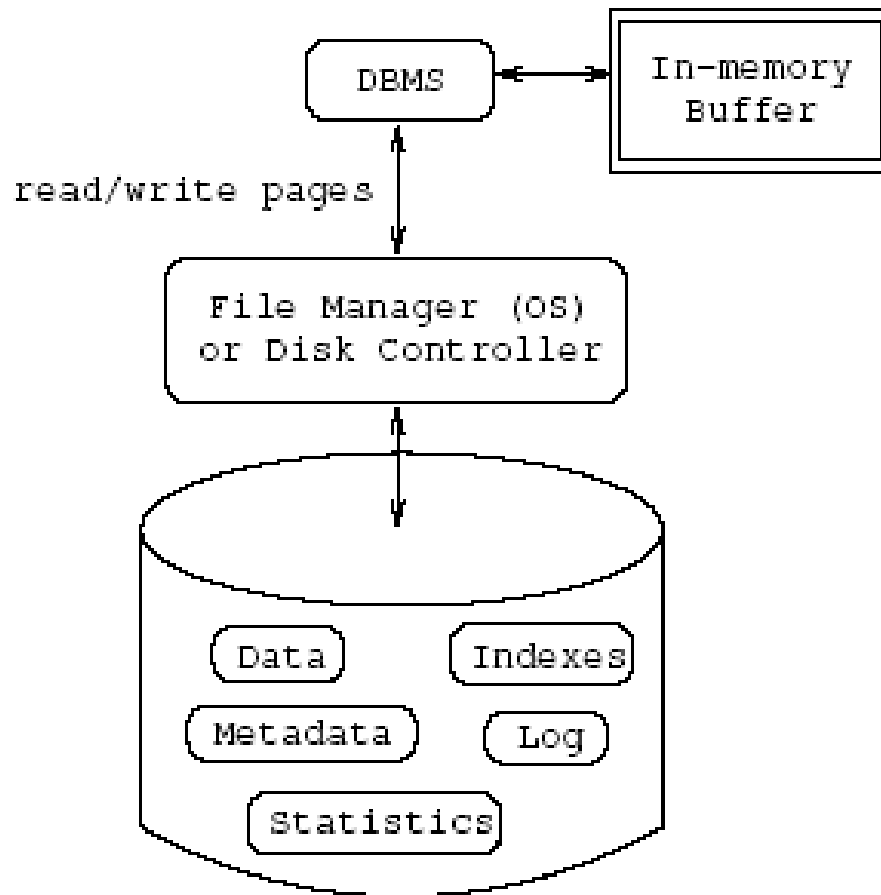
Datenbankmanagementsystem, oder auch Instanz (DBMS)

- anwendungsunabhängige Dienste
- Aufgaben sind unter anderem
  - Verwaltung von Schemata („Struktur“ der Informationen)
  - Bearbeitung von Abfragen (u.a. eigene Abfragesprache)
  - Jobs
  - *Transaktionen (wird später ein Thema)*
  - Sicherheit
  - Speicherverwaltung
  - Bereitstellen von *Werkzeugen (Utilities)*

n Datenbanken

- anwendungsspezifische Daten ohne Funktionalität
- Logische und physische **Realisierung** von Datenbank-Objekten (Umsetzung eines Modells)





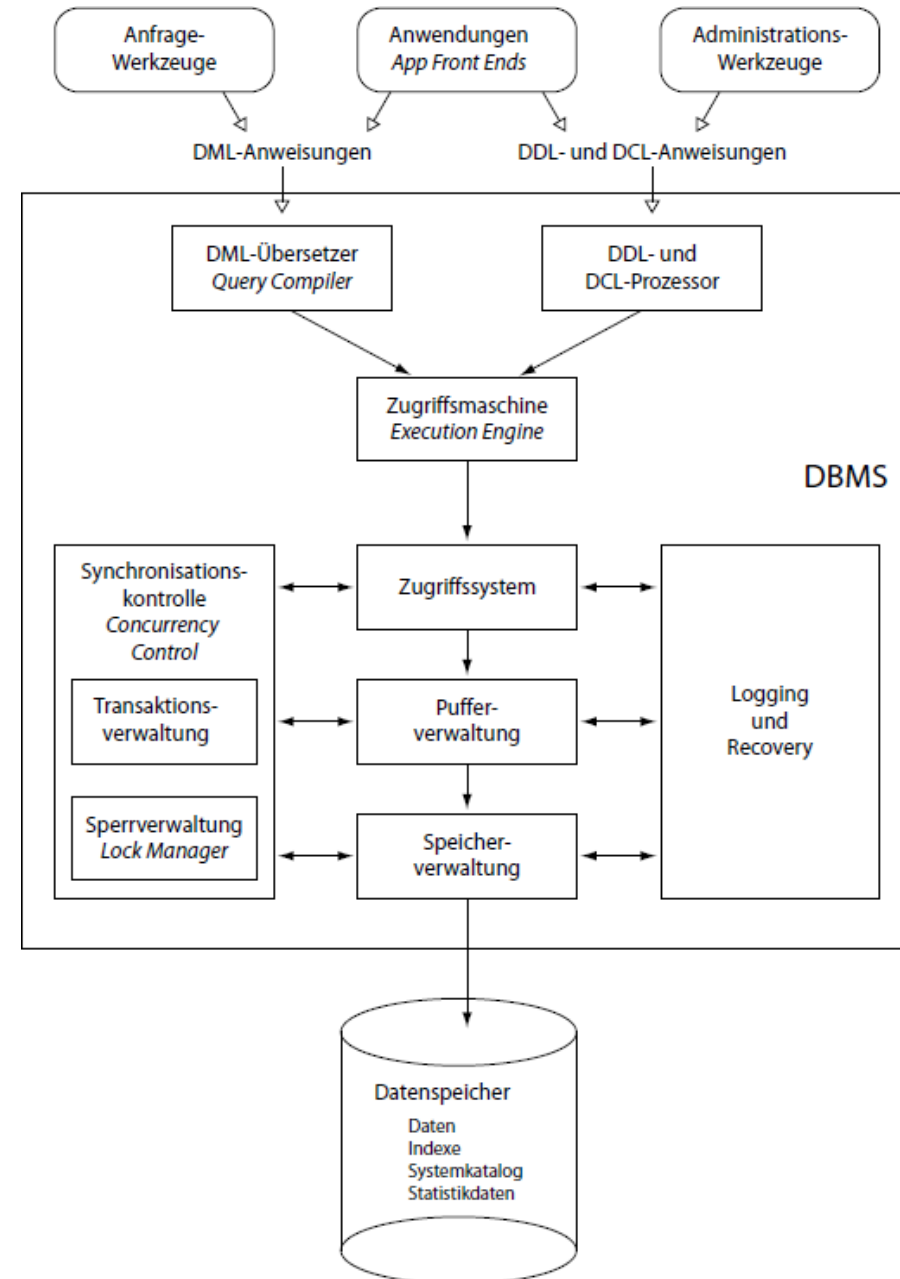


# Datenbanksysteme

## Aufbau eines DBMS

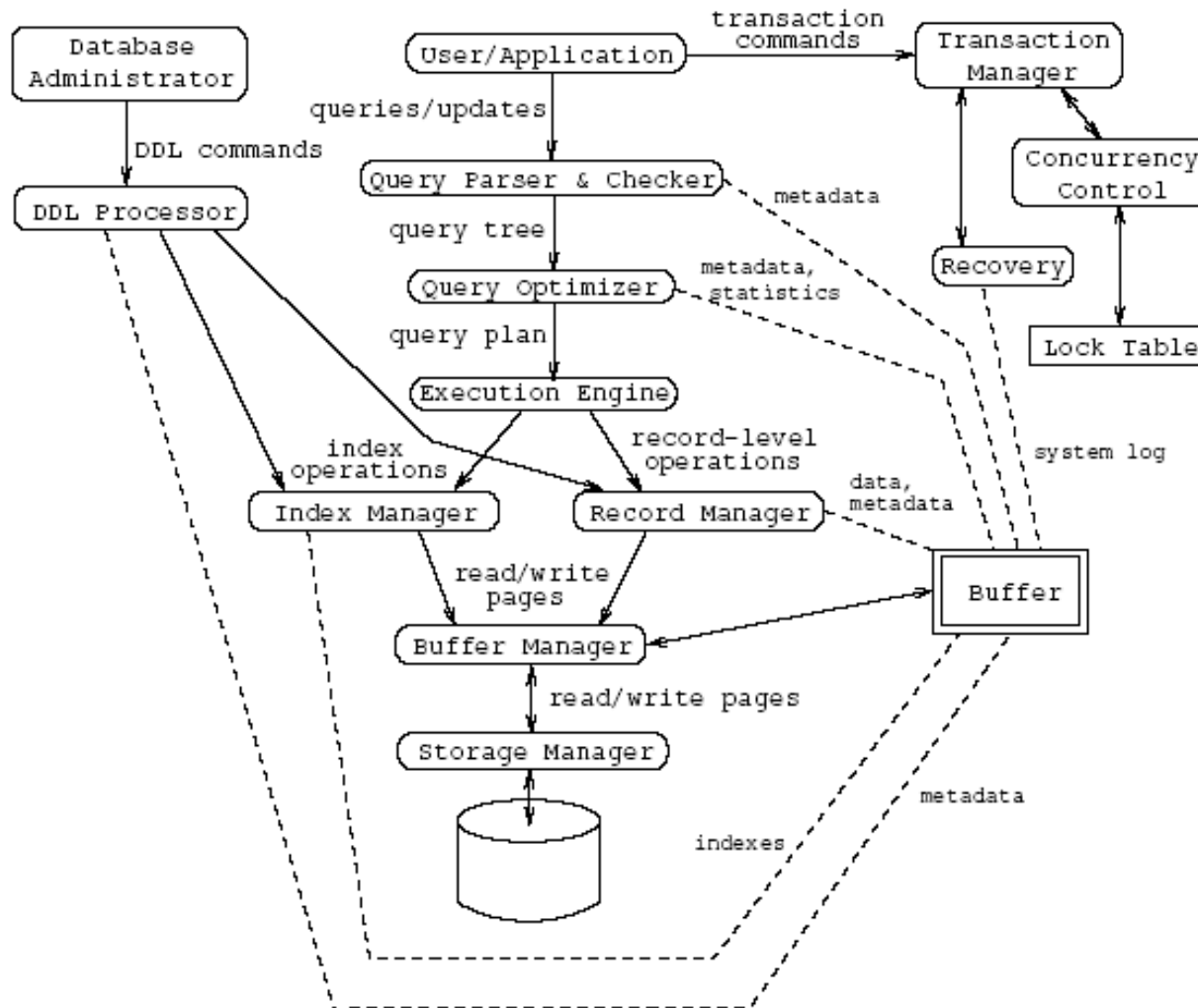
Der Kern des Datenbanksystems besteht konzeptionell aus drei Schichten:

- Zugriffssystem\*
  - Adressierung von Tabellen, Datensätzen und Indizes
- Pufferverwaltung
  - Anfragen nach Daten werden übersetzt in Anfragen nach Pages, die die Pufferverwaltung im Hauptspeicher hält. Alle Zugriffe auf die Daten geschehen durch die Pufferverwaltung.
- Speicherverwaltung
  - macht die eigentlichen Zugriffe auf den sekundären, externen Speicher, heute in der Regel Festplatten; ihre Aufgabe ist der Transfer der Daten von der Platte in die Puffer der Pufferverwaltung und umgekehrt.



# Datenbanksysteme

## Aufbau eines DBMS, Prozesssicht





# Datenbanksysteme

## Zugriffsschichten

### Das 3-Schichten-Modell... aber vorher die Frage:

Wodurch wird ein „gutes“ Schichten-Modell definiert?

- Schichten müssen zielführend sein
  - Jede Schicht wird tatsächlich benötigt
  - Nicht zu komplex, nicht zu fein
- Schichten müssen Ansprüche erfüllen
  - Tiefer liegende Schichten bieten was die höher liegende Schicht benötigt
  - Nicht mehr und nicht weniger
- Schichten müssen implementierungsunabhängig sein
  - anpassungsfähiger, universell einsetzbar
  - Ausnahme: proprietäre Schichten wenn Anforderung zu sonderbar / zu komplex
- Schichten müssen allgemeingültig sein
  - Möglichst wenig Randbedingungen, Einschränkungen, Ausnahmen
  - ➔ Änderungen einer Schicht werden durch mögliche Wechselwirkungen schwierig!

# Datenbanksysteme

## Zugriffsschichten

### Das 3-Schichten-Modell...

#### Externe Ebene

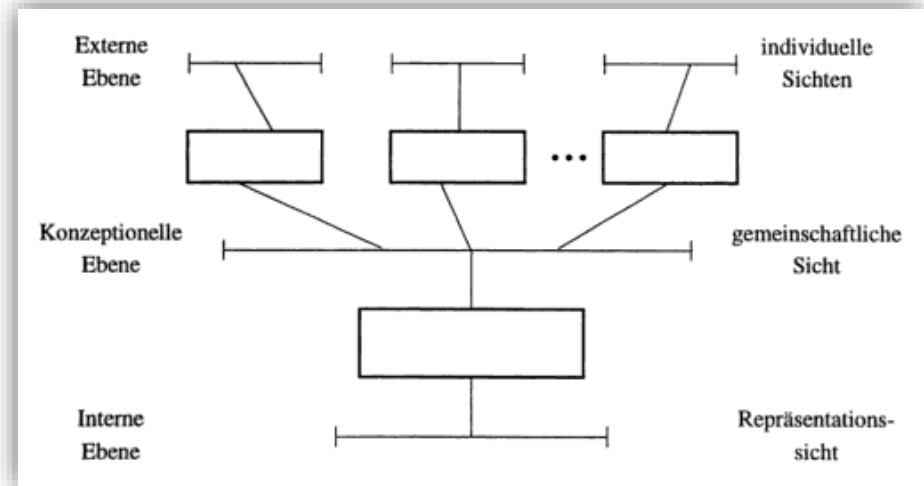
- Sicht auf das System für Endanwender und Anwendungen
- Sicherheit, Datenschutz, Filterfunktion

#### Konzeptuelle Ebene

- Sicht auf das System für Business Analysten etc. (fachübergreifend)
- Das logische Datenmodell (Tabellen, Beziehungen) liegt hier

#### Interne Ebene

- Sicht auf das System für Entwickler / Datenbankadmins / Performanceoptimierer
- Physische Speicherung der Daten
- Indizes, Prozeduren, Jobs, Optimizer





# Datenbanksysteme

## Klassifizierung

Eine mögliche Klassifizierung kann folgendermaßen vorgenommen werden

- Anzahl Nutzer
  - 1 Nutzer
  - n Nutzer
- Verteilung
  - Ein DBMS
  - Mehrere DBMS (Distributed DBMS)
    - Homogen
    - Föderativ
- Kosten kommerzieller Systeme (Sicht des Dozenten)
  - Einbenutzersysteme < 1000 €
  - Standard-Systeme (Oracle, MS SQL Server) < 100.000 €
  - Cluster- / Mainframe-Systeme oder Hochleistungs-DBen > 100.000 €
- Datenspeicherung
  - In memory
  - Persistent
  - Mischformen
- Zugriffspfade
  - Lokal
  - Client-Server
- Hochverfügbarkeit
  - Ausfallwahrscheinlichkeit
  - time to recover
  - ...
- Funktionen
- Usw.





# Datenbanksysteme

## Klassifizierung

Eine mögliche Klassifizierung kann aber auch folgendermaßen vorgenommen werden

Allgemeine / generische DBS

- Können für so gut wie jedes Anwendungsgebiet genutzt werden
- Unterklassifizierung beispielsweise auf Basis des Datenbankmodells möglich
  - Relationale DBS (das relationale Datenmodell ist Schwerpunkt der Vorlesung)
  - Objektrelationale DBS (**wird kurz thematisiert**)
  - Objektorientierte DBS (**Schwerpunkt „Datenbanken 2“**)
  - Hierarch. DBS (**wird nicht weiter thematisiert**)
  - NoSQL-DBS (Key-Value-Stores, Graph-DBS, ...)
  - (**teilweise Thema in „Datenbanken 2“**)
  - ...

(anwendungs)spezifische DBS (**kein Vorlesungsbestandteil**)

- Nur für spezielle Anwendungsgebiete einsetzbar
  - „Geo-Datenbanken“
  - „Temporale Datenbanken“
  - „Bild-Datenbanken“



# Begriffsdefinitionen

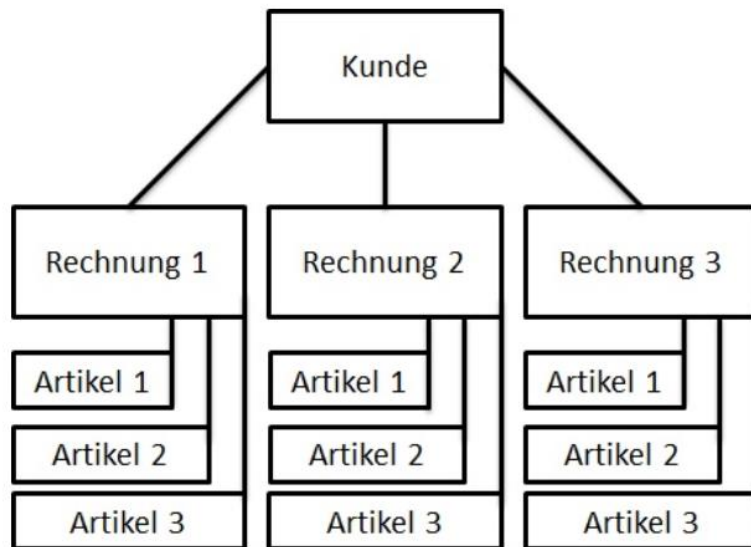
## Daten(bank)modell

- Definition hier schwer, da der Begriff sehr uneinheitlich genutzt wird. Eine Übersicht mit recht hoher Qualität bietet hier Wikipedia:  
<http://de.wikipedia.org/wiki/Datenbankmodell>
- Ist allg. die „Infrastruktur“ für die Modellierung
- Im weiteren Kontext folgende Definition:
  - Ein Datenmodell legt die Modellierungskonstrukte fest, mit denen ein Modell aus der realen Welt (bzw. ein Ausschnitt) in ein für den Computer verarbeitbares Abbild überführt werden kann. Es beinhaltet die Möglichkeit zur
    - Beschreibung der Datenobjekte
    - Festlegung der anwendbaren Operatoren auf Datenobjekte
- Es besteht als aus 2 Teilsprachen (auf Ebene eines DBMS oft mehr als 2)
  - Datendefinition: data definition language, DDL (Strukturbeschreibung = Schema)
  - Datenmanipulation: data manipulation language, DML  
Diese DML wird nochmal untergliedert in
    - Anfragesprache (query language)
    - Datenmanipulationssprache
  - Rechtevergaben (DBMS-seitig und nicht modellseitig): DCL, data control language



# Datenbankmodelle

## Hierarchisches Modell



Daten werden in einer Baumstruktur gespeichert, ähnlich Dateisystem auf einem PC.

### Navigation

- von oben nach unten, vom „Parent“ zu den „Children“
- über sekundäre Indexe

Anwendungsgebiete:  
Sehr hohe Transaktionsraten

Heute weitgehend von relationalen DB abgelöst.

[Quelle: 5 Datenbankmodelle im Vergleich](#)



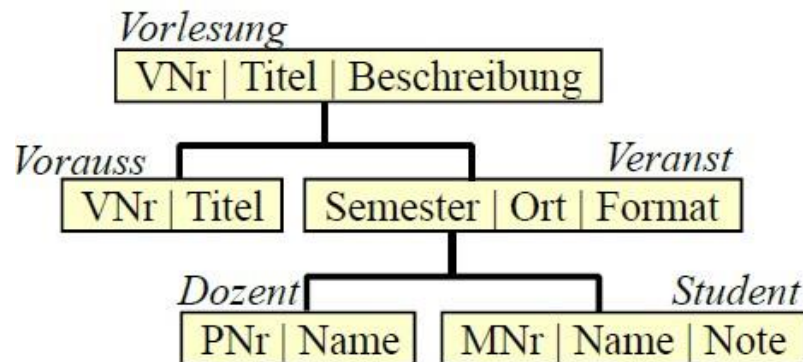
# Datenmodelle

## Hierarchisches Datenbankmodell

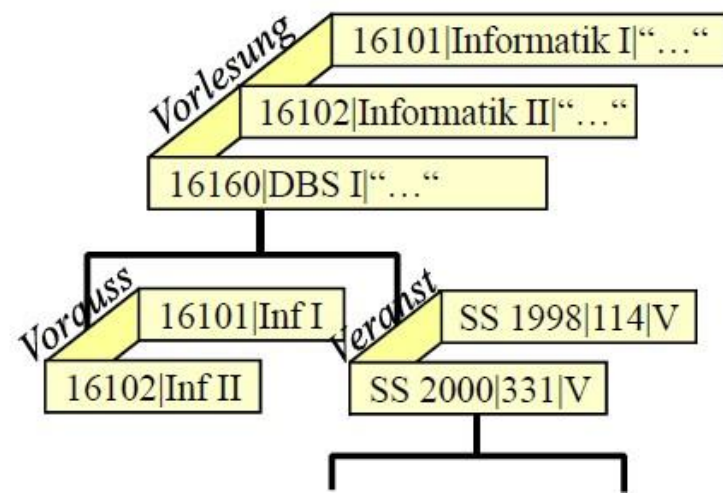
### Aufbau:

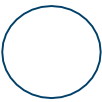
- Schema + Daten werden durch Baum strukturiert
- Der gesamte Datenbestand muss hierarchisch repräsentiert werden (oft schwierig)
- Verzeichnisdienste, XML-Datenbanken

### Schema:



### Inhalt:





# IMS-Beispielprogramm

... bevor es Abfragesprachen gab ...

IDENTIFICATION DIVISION.

PROGRAM-ID. IMSTEST.

DATA DIVISION.

WORKING-STORAGE SECTION.

01 DLI-FUNCTIONS.

```
05 DLI-GU      PIC X(4)  VALUE 'GU  '.
05 DLI-GHU     PIC X(4)  VALUE 'GHU  '.
05 DLI-GN      PIC X(4)  VALUE 'GN   '.
05 DLI-GHN     PIC X(4)  VALUE 'GHN  '.
05 DLI-GNP     PIC X(4)  VALUE 'GNP  '.
05 DLI-GHNP    PIC X(4)  VALUE 'GHNP '.
05 DLI-ISRT    PIC X(4)  VALUE 'ISRT '.
05 DLI-DLET    PIC X(4)  VALUE 'DLET '.
05 DLI-REPL    PIC X(4)  VALUE 'REPL '.
05 DLI-CHKP    PIC X(4)  VALUE 'CHKP '.
05 DLI-XRST    PIC X(4)  VALUE 'XRST '.
05 DLI-PCB     PIC X(4)  VALUE 'PCB  '.
```

01 SEGMENT-I-O-AREA PIC X(150).

LINKAGE SECTION.

01 STUDENT-PCB-MASK.

```
05 STD-DBD-NAME PIC X(8).
05 STD-SEGMENT-LEVEL PIC XX.
05 STD-STATUS-CODE PIC XX.
05 STD-PROC-OPTIONS PIC X(4).
05 FILLER          PIC S9(5) COMP.
05 STD-SEGMENT-NAME PIC X(8).
05 STD-KEY-LENGTH  PIC S9(5) COMP.
05 STD-NUMB-SENS-SEGS PIC S9(5) COMP.
05 STD-KEY          PIC X(11).
```

PROCEDURE DIVISION.

ENTRY 'DLITCBL' USING STUDENT-PCB-MASK.

A000-READ-PARA.

110-GET-INVENTORY-SEGMENT.

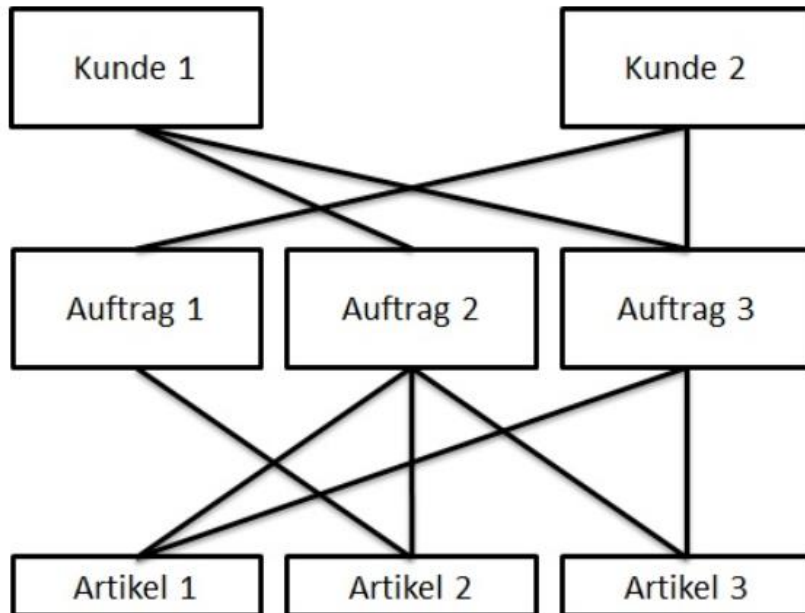
```
CALL 'CBLTDLI' USING DLI-GN
                        STUDENT-PCB-MASK
                        SEGMENT-I-O-AREA.
```

GOBACK.



# Datenbankmodelle

## Netzwerkmodell



Erweiterung des hierarchischen Datenbankmodells

Mehrere Navigationspfade, keine strenge Hierarchie wie beim hierarchischem Datenmodell.

In den 70er und 80er Jahren aufgekommen, aber nicht wirklich erfolgreich

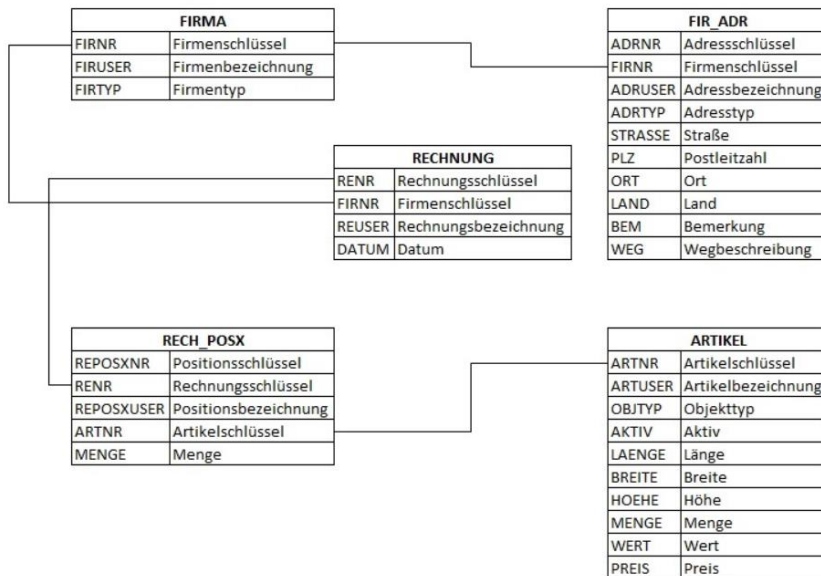
Heute wieder aktueller: Graph-Datenbanken

[Quelle: 5 Datenbankmodelle im Vergleich](#)



# Datenbankmodelle

## Relationales Modell



Basieren auf Relationen in Form zweidimensionaler Tabellen. Jeder Datensatz ist eine Zeile (Tupel) in der Tabelle. Die Attributwerte bilden die Spalten der Tabelle.

Zeilen sind eindeutig über Primärschlüssel identifizierbar.

Tabellen werden über Beziehungen miteinander verknüpft.

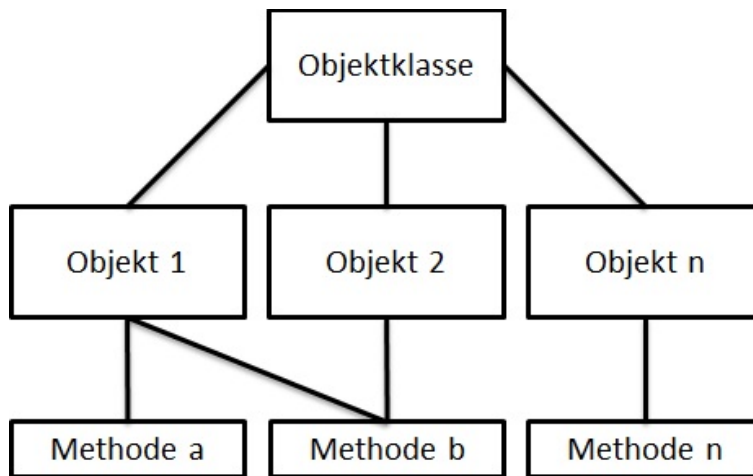
Effizient und flexibel → heute das am weitesten verbreitete Modell

[Quelle: 5 Datenbankmodelle im Vergleich](#)



# Datenbankmodelle

## Objektorientiertes Modell



- Basieren auf dem Objektmodell. Daten werden im Sinne der Objektorientierung verwaltet.
- Rein objektorientierte DB haben keine Verbreitung gefunden.
- Relationale DBMS wurden um OO-Konzepte erweitert („objektrelationale DBMS“)

[Quelle: 5 Datenbankmodelle im Vergleich](#)





# Datenbankmodelle

## Objektorientiertes Datenbankmodell

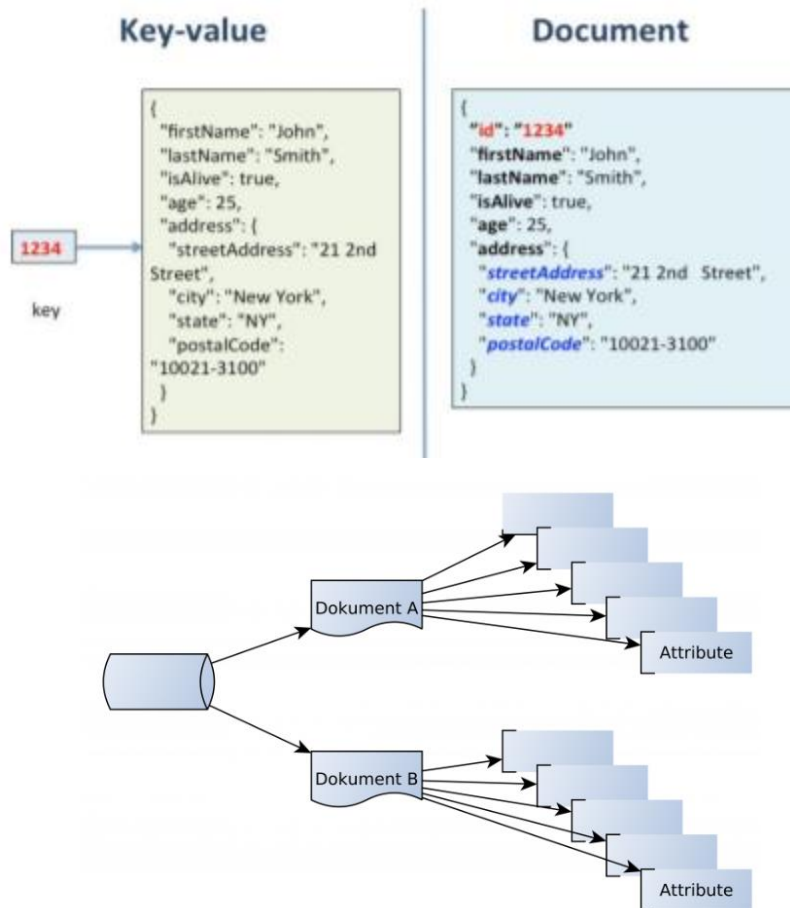
### Aufbau:

- In der Datenbank werden Objekte, d.h. Ausprägungen von Klassen, die zueinander in verschiedenen Beziehungen stehen (z.B. auch Vererbungsbeziehung), persistent gespeichert.
- Rein objektorientierte Datenbanken haben sich kaum durchgesetzt
- Relationale Datenbanken haben die Idee aufgenommen und erlauben jetzt auch Speicherung komplexer Objekte in Relationen
  - Vererbungshierarchien
  - Komplexe Datentypen
    - Records
    - Individuelle Type-Deklarationen mit Methoden
- Thematisierung in Modul Datenbanken II



# Datenbankmodelle

## Dokumentenmodell



Dienen hauptsächlich zum Speichern von semi-strukturierten Daten („Dokumenten“)

Anwendungsgebiete:  
Textdatenbanken,  
Webanwendungen,  
Suchmaschinen



# Begriffsdefinitionen

## Datenbankmodell

- Definition hier schwer, da der Begriff sehr uneinheitlich genutzt wird. Eine Übersicht mit recht hoher Qualität bietet hier Wikipedia:  
<http://de.wikipedia.org/wiki/Datenbankmodell>
- Ist allg. die „Infrastruktur“ für die Modellierung
- Im weiteren Kontext folgende Definition:
  - Ein Datenmodell legt die Modellierungskonstrukte fest, mit denen ein Modell aus der realen Welt (bzw. ein Ausschnitt) in ein für den Computer verarbeitbares Abbild überführt werden kann. Es beinhaltet die Möglichkeit zur
    - Beschreibung der Datenobjekte
    - Festlegung der anwendbaren Operatoren auf Datenobjekte
- Es besteht als aus 2 Teilsprachen (auf Ebene eines DBMS oft mehr als 2)
  - Datendefinition: data definition language, DDL (Strukturbeschreibung = Schema)
  - Datenmanipulation: data manipulation language, DML  
Diese DML wird nochmal untergliedert in
    - Anfragesprache (query language)
    - Datenmanipulationssprache
  - Rechtevergaben (DBMS-seitig und nicht modellseitig): DCL, data control language

# Begriffsdefinitionen

## Datenmodell, Umsetzung wird durch Schema realisiert

### Konzeptuelles Datenmodell

- Form beliebig, E/R-Diagramm oder UML-Diagramm sind die Regel
- Angabe der benötigten Daten, Beziehungen und Kardinalitäten

### Logisches Datenmodell

- „Fachliche“ Sicht auf Daten, basierend auf Regeln des DBMS
- Tabellen mit definierten und standardisierten Datentypen
- „Normalisierung“  
(kommen wir später zu)
- Schlüsseldefinitionen
- z. B. ein Relationenmodell

### Physisches Datenmodell

- „Technische“ Sicht auf Daten, basierend auf Regeln des DBMS
- Tabellen mit definierten spezifischen Datentypen, Indizes, Replikationsmechanismen, Datensicherung, ...
- Synonyme: „Datenschema, Datenbankschema, Schema“



# Datenbankmodelle

## Relationales Datenbankmodell

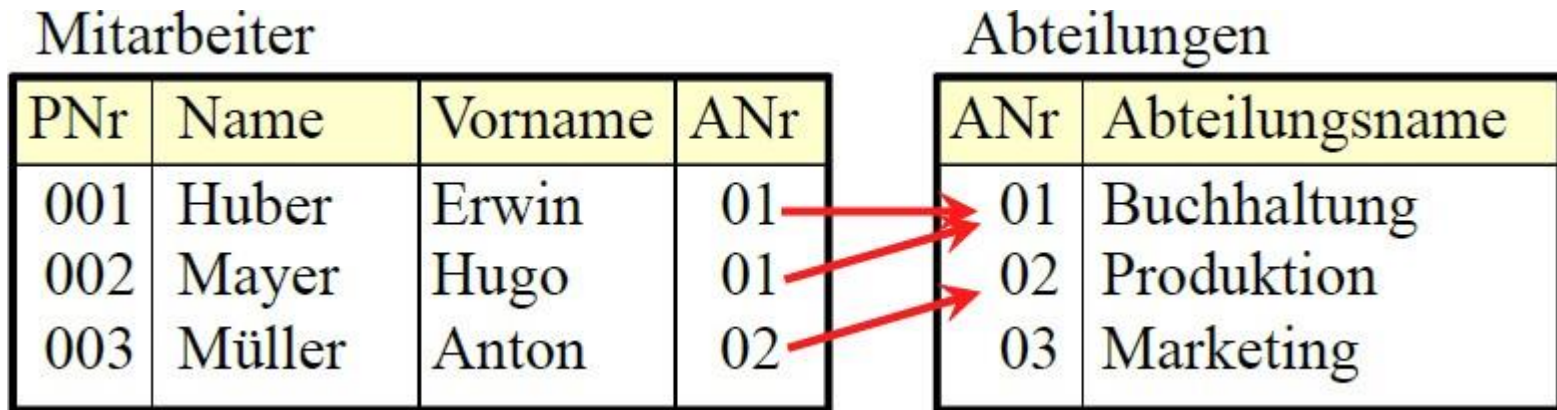
**Aufbau:** (die roten Wörter werden wir in der 2. Vorlesung genauer beleuchten)

- Alle Informationen werden in Tabellenform gespeichert / abrufbar
- Eine Datenbank besteht aus einer Menge von Tabellen (**Relationen**)
  - Mitarbeiter, Abteilung
- Eine Tabelle besteht aus einer Menge von n-spaltiger Zeilen / bzw. Datensätzen mit n Datenfeldern (**n-Tupel**)
  - Mitarbeiter: {\$Vorname, \$Nachname, \$Geburtsdatum, ...}
- Die Spalten (**Attribute**) können strukturell verschieden sein (Datentyp), haben jedoch immer genau einen primitiven Datentypen (**Wertebereich**)
  - Mitarbeiter: Vorname (String), Nachname (String), Geburtsdatum (Date), ...
  - Komplexe Datentypen im relationalen Modell nicht vorgesehen
- Die n-te Spalte, also das n-te Datenfeld (**Attributwert**) eines jeden Datensatzes (**Tupel**) in der Tabelle hat folglich immer den gleichen Datentypen
- Komplexe Sachverhalte werden durch Verknüpfung mehrerer Tabellen dargestellt

# Datenbankmodelle

## Relationales Datenbankmodell

Beispiel:





# Datenbankmodelle

## Beispieldatenbanken

### Relationale Datenbanken:

- MS Access, Firebird, sqlite

### Objektrelationale Datenbanken:

- Db2, Oracle, PostgreSQL

### Netzwerkdatenbanken (modernerweise Graphendatenbanken):

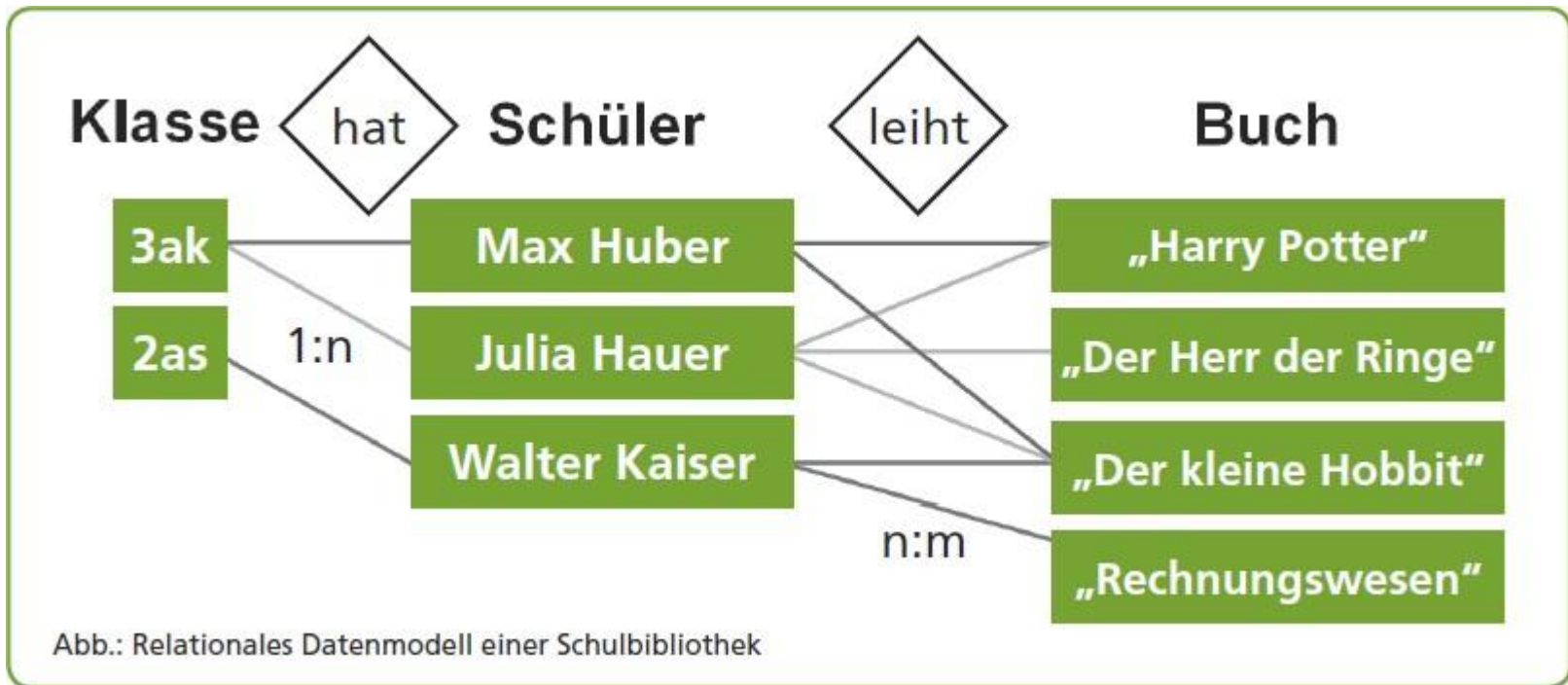
- Neo4j, Oracle Spatial and Graph

### Objektorientierte Datenbanken:

- Db4o, Objectivity/DB

### Hierarchische Datenbanken:

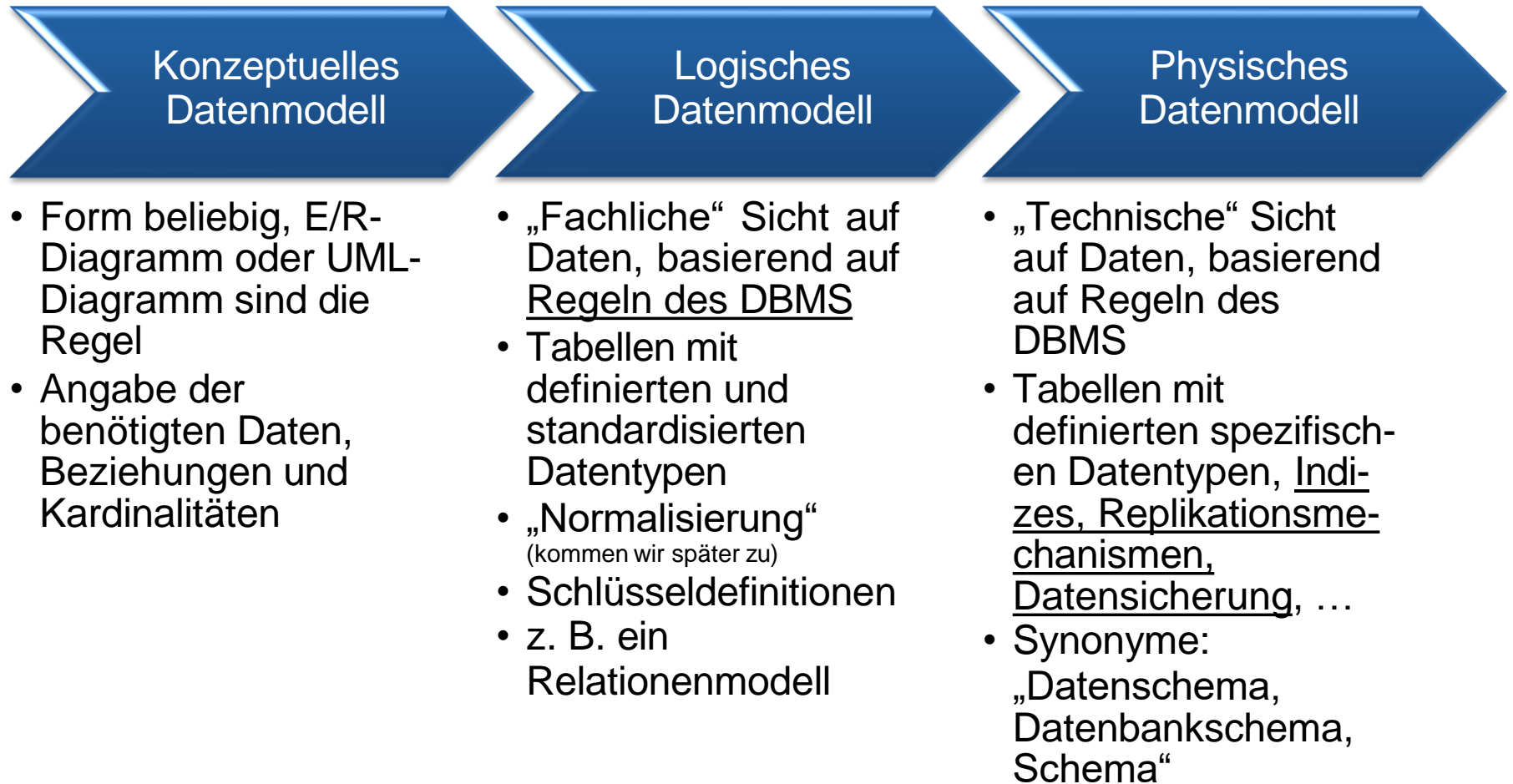
- BaseX, IBM IMS
  - XML-Datenbanken sortiert man zwar in die „**dokumentenorientierten Datenbanken**“ ein. Das Datenmodell ist jedoch trotzdem hierarchisch.





# Begriffsdefinitionen

## Datenmodell, Umsetzung wird durch Schema realisiert



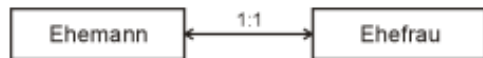


Datenmodellierung	allgemein
<p>„Kardinalitäten sind Mengenangaben, mit der in der Datenmodellierung für Entity-Relationship-Diagramme (ER-Diagramme) für jeden Beziehungstyp festgelegt wird, wie viele Entitäten eines Entitätstyps mit genau einer Entität des anderen am Beziehungstyp beteiligten Entitätstyps (und umgekehrt) in Beziehung stehen können oder müssen.“</p> <p>In UML wäre die Multiplizität ein brauchbares Synonym.</p>	<p>Anzahl der Elemente in einer Menge.</p> <p>Für Tabellen gilt: Anzahl Zeilen</p> <p>Für Spalten gilt: Anzahl verschiedener Attributwerte</p>



### Datenmodellierung (einfache Chen-Notation)

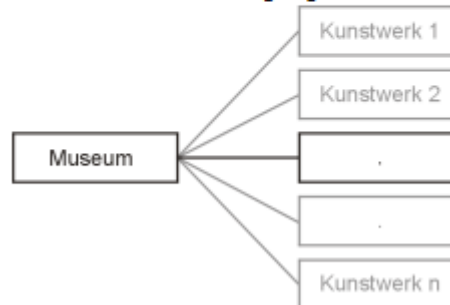
1:1



1:n



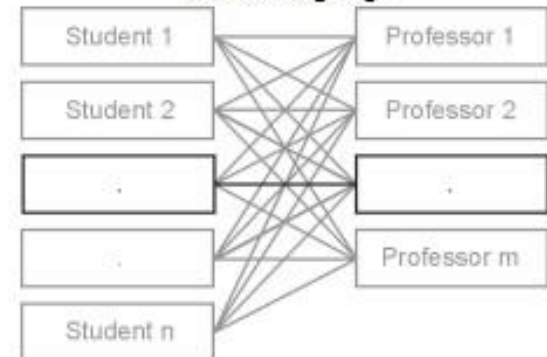
Vorüberlegung:



n:m



Vorüberlegung:



# Datenmodelle

## gängige Arten

?

**Welches Datenmodell ist angebracht?**

- Ein Supermarkt möchte alle Produkte in Gruppen und Untergruppen einteilen. Jedes Produkt gehört einer Untergruppe an, jede Untergruppe gehört einer übergeordneten Gruppe an
- Ein Supermarkt möchte alle Produkte in Gruppen und Untergruppen einteilen, wobei ein Produkt mehreren Gruppen angehören kann und eine Gruppe mehreren übergeordneten Gruppen angehören kann

!

**Hierarchisch oder relational**  
**Relational**

# Datenmodelle

## gängige Arten

?

**Lässt sich mit einem hierarchischen Datenmodell ein relationales Datenmodell implementieren? Wenn ja, was bedeutet dies für die Beziehungskardinalitäten?**

**Lässt sich mit einem relationales Datenmodell ein hierarchisches Datenmodell implementieren? Wenn ja, was bedeutet dies für die Beziehungskardinalitäten?**

!

**Nein, n:m geht nicht  
Ja, 1:n**



# Datenbanksysteme

## Funktionale Anforderungen nach Edgar Frank Codd

Für Datenbanksysteme müssen laut Codd folgende Punkte erfüllt sein:

1. **Integration:** kompletter Datenbestand wird transparent verwaltet, Redundanzfreiheit muss realisierbar sein
2. **Operationen:** SELECT, UPDATE, INSERT, und DELETE Operationen müssen möglich sein
3. **Katalog** (Data Dictionary): Beschreibung aller in der Datenbank existierenden Objekte (Tabellen, Indizes, ...)
4. **Benutzersichten:** unterschiedliche Anwendungen können unterschiedliche Sichten auf Daten bieten, wobei eine Sicht eine Beschränkung der sichtbaren Daten ist
5. **Konsistenzüberwachung:** Änderungen müssen „ankommen“ und stimmig sein → logs! Auch wenn bei Abstürzen Daten verloren gehen, muss die DB wieder konsistent zur Verfügung stehen können
6. **Zugriffskontrolle:** Autorisierungen und Privilegien
7. **Transaktionsverwaltung:** *Atomarität* („ganz oder gar nicht“) muss für mehrere aufeinanderfolgende Anweisungen sichergestellt werden können
8. **Synchronisation:** Problemlösung bei mehrfachen Zugriffen auf ein Datum, welches aktuell verändert wird (siehe auch: Transaktions-Isolationslevel)
9. **Datensicherung:** muss mit Bordmitteln möglich sein



# Relationale Datenbanksysteme

## Anforderungen nach Edgar Frank Codd

Für relationale Datenbanksysteme gelten laut Codd 12+1 Regeln:

1. **Darstellung von Information:** alle Informationen müssen in Tabellen abgelegt sein
2. **Zugriff auf Daten:** Jeder Wert einer relationalen Datenbank muss durch Kombination von Tabellennamen, Primärschlüssel und Spaltenname auffindbar sein.
3. **Systematische Behandlung von Nullwerten:** Das DBMS behandelt Nullwerte durchgängig gleich als unbekannte oder fehlende Daten und unterscheidet diese von Standardwerten.
4. ...
5. **Abfragesprache:** Zu einem relationalen System gehört mindestens eine Abfragesprache mit einem vollständigen Befehlssatz für Datendefinition, Manipulation, Integritätsregeln, Autorisierung und Transaktionen.

Usw.



# 1: Darstellung von Information

Alle Informationen in relationalen Datenbanken müssen logisch in Tabellen dargestellt sein, insbesondere

- Daten
- Definitionen von Tabellen und Attributen
- Integritätsbedingungen und die Aktion bei deren Verletzung (siehe Regel 10)
- Sicherheitsinformationen (z.B. Zugangsberechtigungen)





## 2: Zugriff auf Daten

Jeder Wert einer relationalen Datenbank muss logisch durch eine Kombination von Tabellennamen, Primärschlüssel und Attributnamen (Spaltennamen) auffindbar sein.

Dies bedeutet, dass in einer Tabelle an jedem Schnittpunkt einer Zeile mit einer Spalte nur ein Wert stehen darf.



### 3: Systematische Behandlung von Nullwerten

Fehlende oder unbekannte Informationen werden als Nullwerte („NULL“) dargestellt bzw. gespeichert.

Nullwerte werden durchgängig gleich, insbesondere unabhängig vom Datentyp des Attributes, behandelt.

*(Man kann also in numerischen Feldern bei fehlenden Daten das Feld nicht leer lassen, und man sollte Textfelder nicht leer lassen oder ein besonderes Zeichen, z.B. —, einfügen.)*



### 3: Systematische Behandlung von Nullwerten

Fehlende, unbekannte oder nicht anwendbare Daten werden als Nullwerte („NULL“) dargestellt bzw. gespeichert.

Nullwerte werden durchgängig gleich, insbesondere unabhängig vom Datentyp des Attributes, behandelt.

*(Man kann also in numerischen Feldern bei fehlenden Daten das Feld nicht „leer“ lassen, und man sollte Textfelder nicht leer lassen oder ein besonderes Zeichen, z.B. —, abspeichern.)*



## 4: Der Katalog

Die Datenbankstruktur (Metadaten) wird in derselben logischen Struktur wie die Daten gespeichert, also in Tabellen.

Dazu muss die Struktur aller Tabellen, die zu einer Datenbank gehören, in Systemtabellen (dem Katalog) zugänglich sein.

Der Katalog lässt sich mit den gleichen Mitteln abfragen wie jede andere Tabelle auch.

Änderungen an der Datenbankstruktur wirken sich sofort im Katalog aus.



## 5: Abfragesprache

Ein relationales System enthält mindestens eine befehlsgesteuerte Abfragesprache, die mindestens die folgenden Funktionen unterstützt:

- Datendefinition
- Definition von Views (logische Sichten der Datenbank, die der Benutzer aus den Attributen der Basistabellen erstellt und mit den gewohnten Operatoren manipulieren kann)
- Definition von Integritätsbedingungen
- Definition von Transaktionen (Eine Transaktion ist eine Folge von Befehlen, die eine Datenbank von einem konsistenten Zustand in einen anderen überführen. Eine Transaktion muss entweder vollständig durchgeführt oder, bei einem Abbruch, vollständig zurückgesetzt werden.)
- Definition von Berechtigungen

Eine weit verbreitete Abfragesprache hierfür ist SQL.



## 6: Aktualisieren von Views

Alle Views, die theoretisch aktualisiert werden können, können auch vom System aktualisiert werden.

*(Beispiel: Hat man zwei Spalten A und B mit Zahlen, so kann man sich eine weitere Spalte definieren, die  $A*B$  enthält. Ändert man einen Wert in dieser Spalte, so kann daraus nicht der Wert der Spalten A und B bestimmt werden, da im allgemeinen aus dem Produkt zweier Zahlen diese zwei Zahlen nicht bestimmt werden können. Dieses View kann also theoretisch nicht aktualisiert werden.)*



## 7: Abfragen und Editieren ganzer Tabellen

Ein Datenbanksystem muss das Einfügen, Aktualisieren und Löschen von Datensätzen als mengenorientierte Operation unterstützen.

Es muss auch Vereinigungs-, Schnitt- und Differenzoperationen unterstützen, um Sätze von Datensätzen zu erhalten.



## 8: Physikalische Unabhängigkeit

Der Zugriff auf die Daten durch den Benutzer muss unabhängig davon sein, wie die Daten gespeichert werden oder wie physikalisch auf sie zugegriffen wird.

Dies bedeutet, dass Anwendungen nur auf die logische Struktur des Systems zugreifen dürfen.

Ändert der Datenbankverwalter die physikalische Struktur, darf der Anwender davon nichts mitbekommen.

*Beispiel: verschiedene Zeichensätze, verschiedene binäre Zahlenformate, Komprimierung, ...*





## 9: Logische Unabhängigkeit der Daten

Anwendungen und Zugriffe dürfen sich nicht ändern, wenn Tabellen so geändert werden, dass alle Informationen erhalten bleiben.

Wenn zum Beispiel zwei Tabellen zusammengelegt oder eine Tabelle in zwei verschiedene Tabellen aufgeteilt wird, darf dies keine Auswirkungen auf die Benutzeranwendung haben.

Dies ist eine der am schwierigsten anzuwendenden Regeln.



# 10: Unabhängigkeit der Datenbank von Anwendungen

Eine Datenbank muss unabhängig von der Anwendung sein, die sie nutzt.

Integritätsbedingungen können geändert werden, ohne dass eine Änderung in der Anwendung erforderlich ist. Eine Datenbank ist also unabhängig von der Front-End-Anwendung und ihrer Schnittstelle.

Alle Integritätsbedingungen müssen in der Abfragesprache definierbar sein und in Tabellen dargestellt werden.

Das System muss mindestens die folgenden Integritätsbedingungen prüfen:

- Vollständigkeitsintegrität (Entity Integrity, Existential Integrity), d.h. ein Primärschlüssel muss eindeutig sein.
- Beziehungsintegrität (Referentielle Integrität, Referential Integrity), d.h. zu jedem Fremdschlüsselwert existiert ein Primärschlüsselwert.



# 11: Verteilung der Daten im Netzwerk

Anwendungen für eine nicht-verteilte Datenbank dürfen sich beim Übergang zu einer verteilten Datenbank logisch nicht ändern. Der Benutzer sollte immer den Eindruck haben, dass sich die Daten nur an einem Standort befinden.



## 12: Kein Unterlaufen der Abfragesprache

Unterstützt ein relationales Datenbanksystem neben der High-Level-Abfragesprache eine Low-Level-Abfragesprache oder Programmierschnittstelle (API), so darf diese die Integritätsbedingungen der High-Level-Sprache nicht unterlaufen.

Beispiel: Die Low-Level Abfragesprache darf nicht direkt auf die physisch gespeicherten Daten zugreifen.



?

**Entscheiden Sie, ob es sich bei folgenden Begriffen um Datenbanksysteme handelt und begründen Sie Ihre Entscheidung:**

- Eine CSV-Datei
- MS Excel in Verbindung mit der Datei „bla.xlsm“

**Bearbeitungszeit: 5 Minuten**



!

**CSV-Datei: nein, es gibt nicht mal ein DBMS!**

**MS Excel+Bla.xlsm: prinzipiell nein, aber schwierig zu beantworten!  
Alle Anforderungen könnten doch irgendwie mit VBA umgesetzt werden...**



?

**Die DHBW Mannheim möchte sämtliche Informationen in Form einer Datenbank vorhalten. Überlegen Sie welche Informationen dort alles grob hinterlegt sein könnten...**

**Bearbeitungszeit: 2 Minuten**

?

**Wie wären die Daten ohne ein Datenbanksystem vorzuhalten?  
Ist das umsetzbar?**

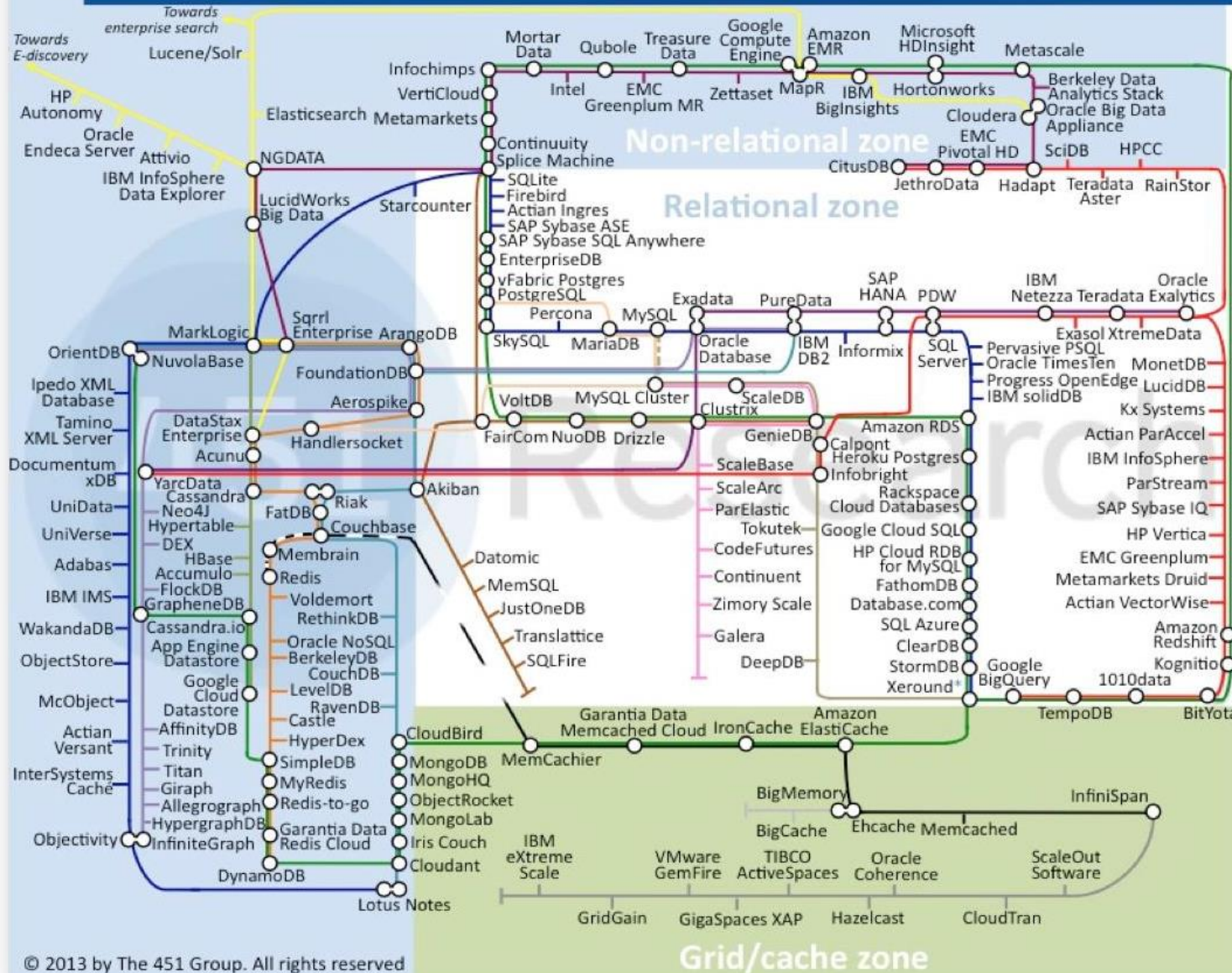
# Database Landscape Map – June 2013

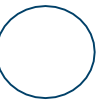
451 Research

## Key:

- General purpose
- Specialist analytic
- as-a-Service
- - - NoSQL extension
- Big Tables
- Graph
- Document
- Key value stores
- Key value direct access
- Hadoop
- - - NewSQL extension
- MySQL storage engines
- Advanced clustering/sharding
- New SQL databases
- - - Data caching extension
- Data caching
- Data grid
- Search
- Appliances
- Off-heap memory

\*Xeround closed May 2013 until further notice





# Vorlesung

## Software

### Apache Derby

- Setzt keine Installationen voraus, OS-unabhängig (Java), sehr einfach zu handhaben
- Gute Dokumentation, Sourcecode verfügbar
- Eingeschränkte Funktionalität, aber für DB-Grundlagen völlig ausreichend

### MySQL, PostgreSQL

- Sehr umfangreich, OS-unabhängig
- PostgreSQL: Oft sehr nah am SQL-Standard
- MySQL: sehr einsteigerfreundlich und im Webumfeld oft verwendet
- Durch verschiedenste Versionen und Betriebssysteme jedoch problematisch
- Komplexität führt teilweise zu Erklärungsmehraufwand (viel Wahlfreiheit = viele Fragen...), MySQL nicht immer standardkonform und teilweise sogar fehlerhaft!
- Hat in vielen Konzernen weniger Bedeutung als die „big player“

### DB2

- Sehr umfangreich, OS-unabhängig, Bedeutung in größeren Unternehmen



# Vorlesung

## Software

### MS SQL

- Sehr umfangreich, hohe Bedeutung in größeren Unternehmen
- Oft nicht standardkonform (Oracle und DB2 zwar auch nicht, aber MS hat sehr viele „Sonderlocken“)

### Oracle

- Sehr umfangreich, hohe Bedeutung in größeren Unternehmen
- Gibt's als vorinstalliertes VM-Image

### MS Access

- Sehr umfangreich, hohe Bedeutung in Unternehmen jeder Größe
- Sehr einfach verständlich, da stark visuelle und weniger technische Orientierung
- Nicht OS-unabhängig, kostenlos maximal via MSDNAA o.ä. beziehbar
- Sehr weit weg vom SQL-Standard, die Umsetzung komplexer Abfragen ist nicht trivial

Wahl für die Vorlesung fällt auf:

ORACLE

Oracle  
SQL Developer



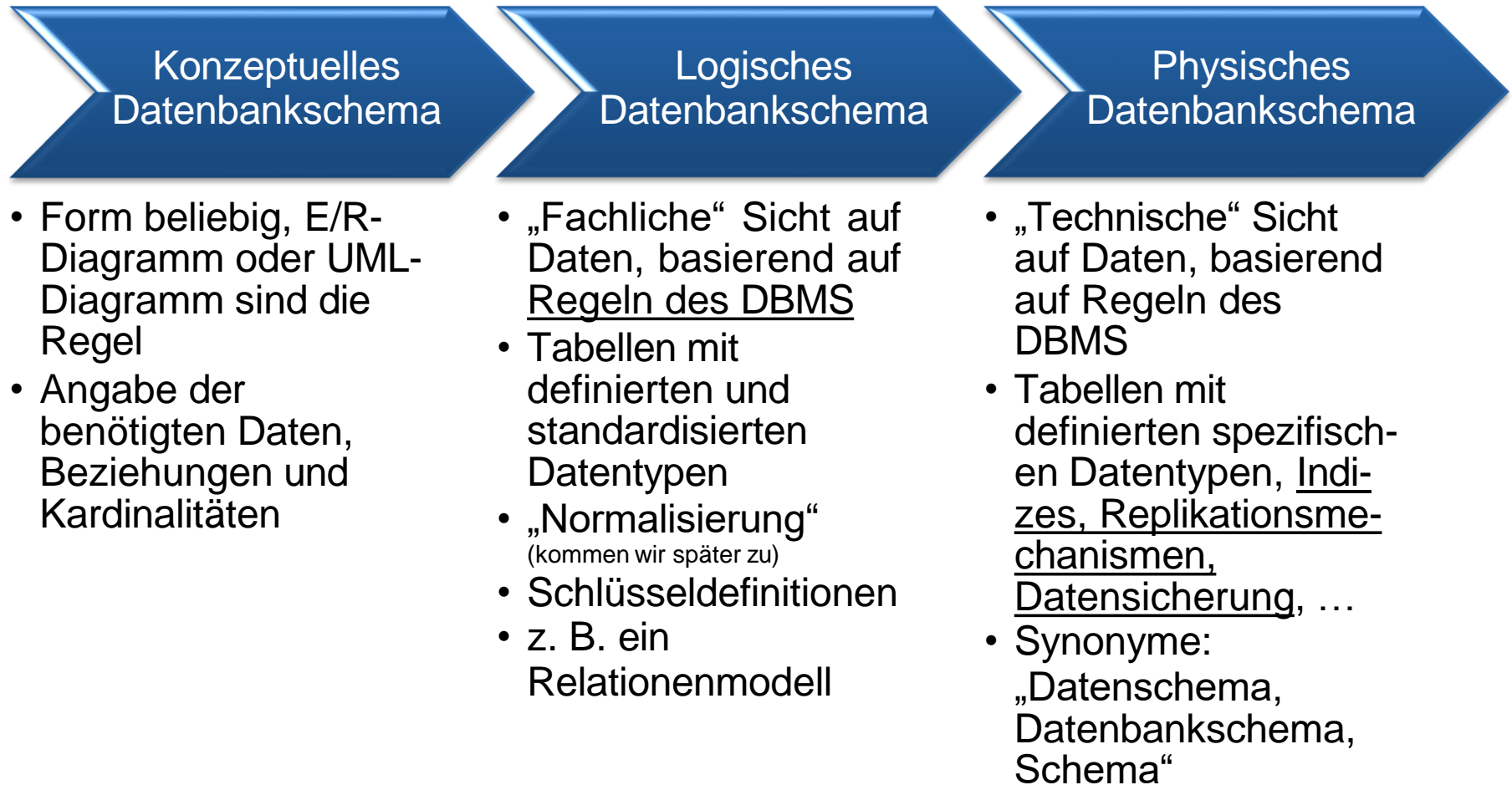
Fallbackvorschlag für Studenten, die  
keine Software installieren können /  
dürfen:

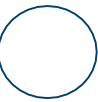
SQLite

SQLite Studio

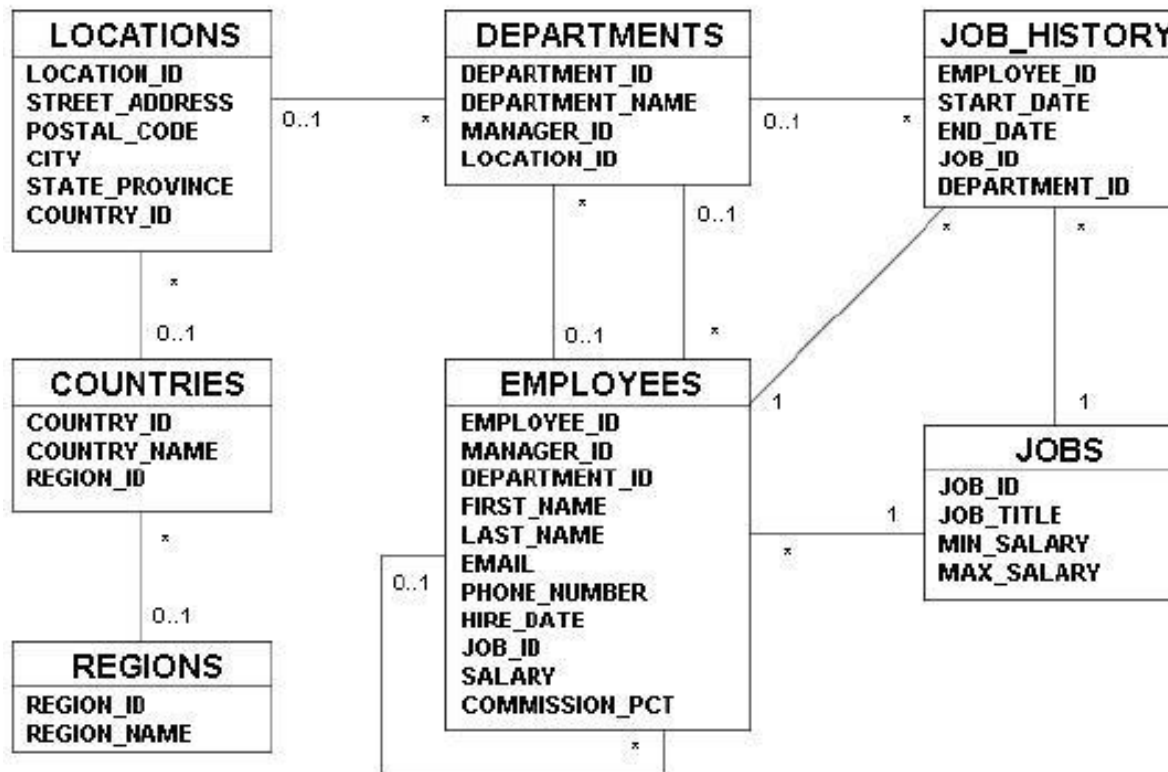
# Wir erinnern uns: Begriffsdefinitionen

## Datenmodell, Umsetzung wird durch Schema realisiert



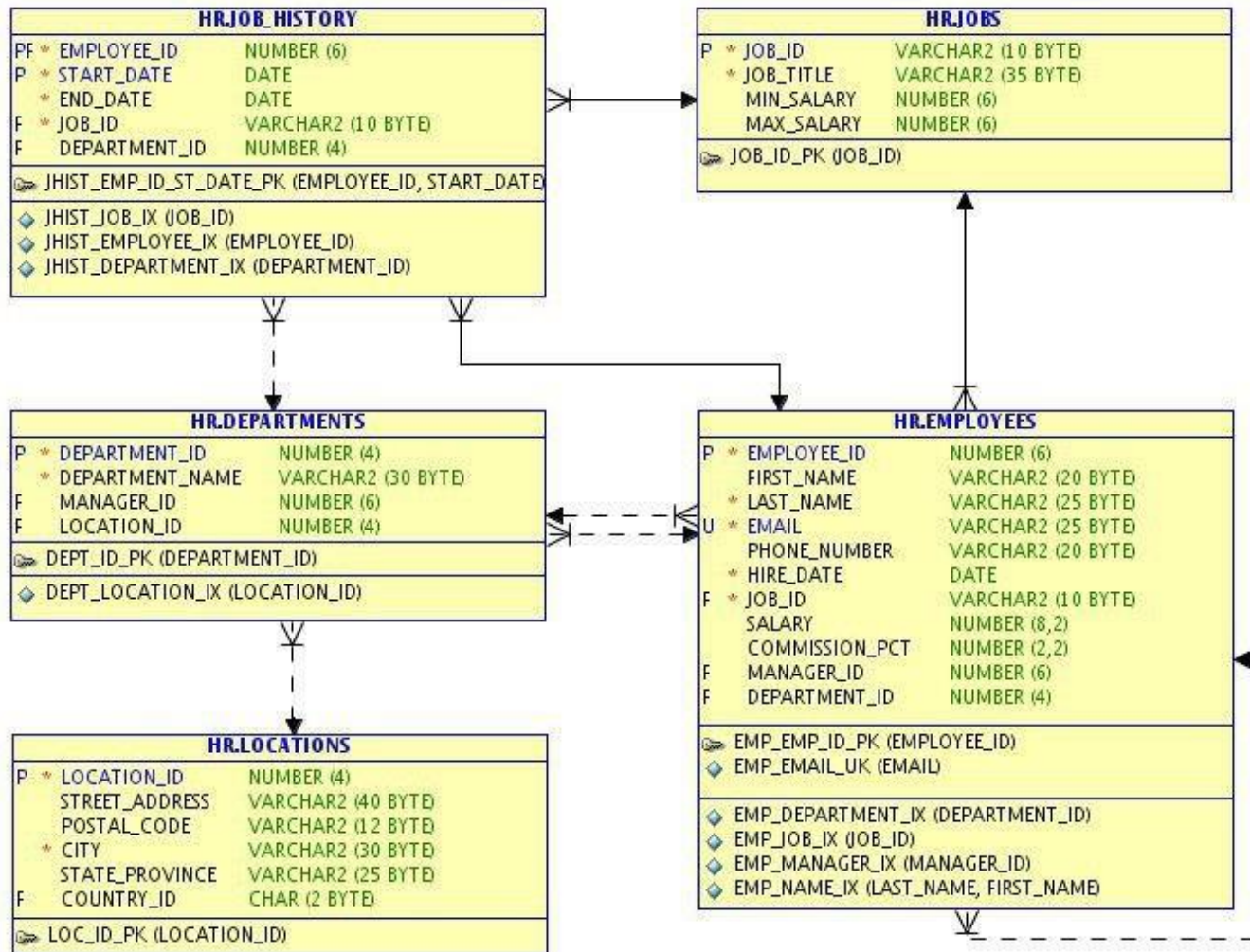


## Human Resources (HR) Schema



# Oracle

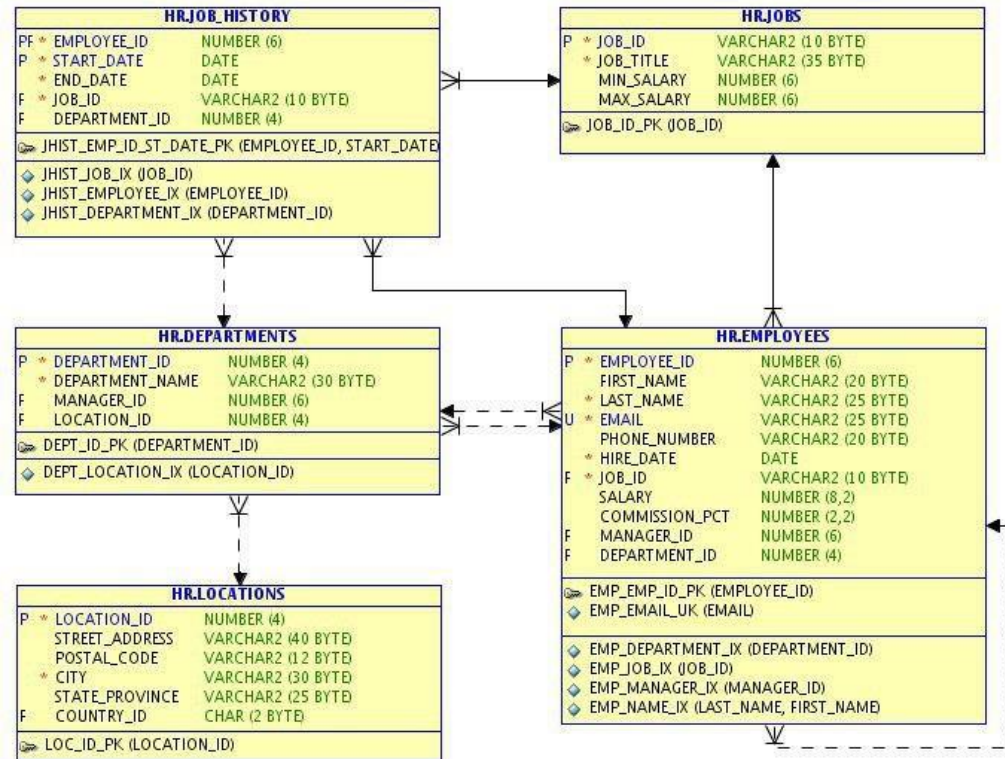
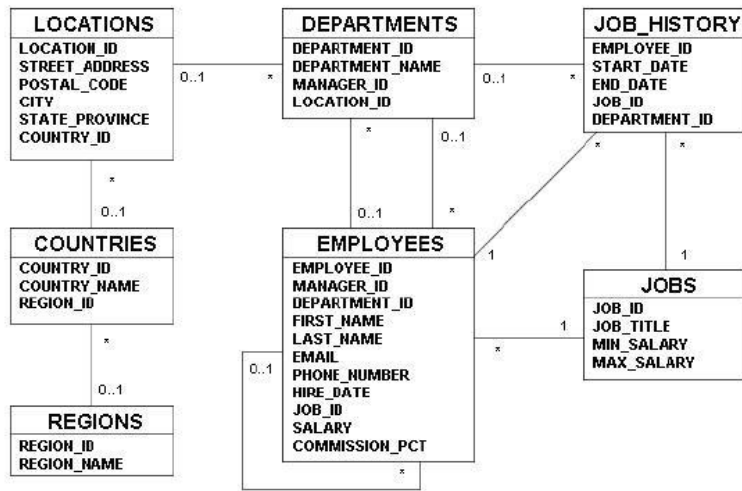
## HR-Schema



# Oracle

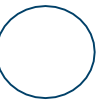
## HR-Schema

### Human Resources (HR) Schema



Worum handelt es sich hier?  
Konzeptuelles, logisches oder physisches  
Datenbankschema?

Bearbeitungszeit: 3  
Minuten



# Hausaufgaben

bis zur übernächsten Vorlesung

@

Installieren Sie auf Ihrem Rechner die Eclipse IDE mit Dbeaver Plugin und importieren Sie das bereitgestellte Derby-Projekt.

---

@

Starten Sie Apache Derby gemäß der Anleitung im Moodle und machen Sie sich mit Eclipse und DBeaver vertraut. Verschaffen Sie sich einen Überblick über die mitgelieferte Dokumentation (wenigstens Inhaltsverzeichnisse lesen)