

LE 1: IBIS - Datenbanken

Einführung, ANSI/X3/SPARC, Anwendungsfälle, Transaktionen

Prof. Dr. Markus Grüne, FB03, Wirtschaftsinformatik

Inhalt

- Grundlagen – Datenbank, DBMS und DBS
- Überblick – 3-Schicht- / ANSI-SPARC-Architektur
- Transaktionen
- Markt für Datenbanken
- Datenmodelle

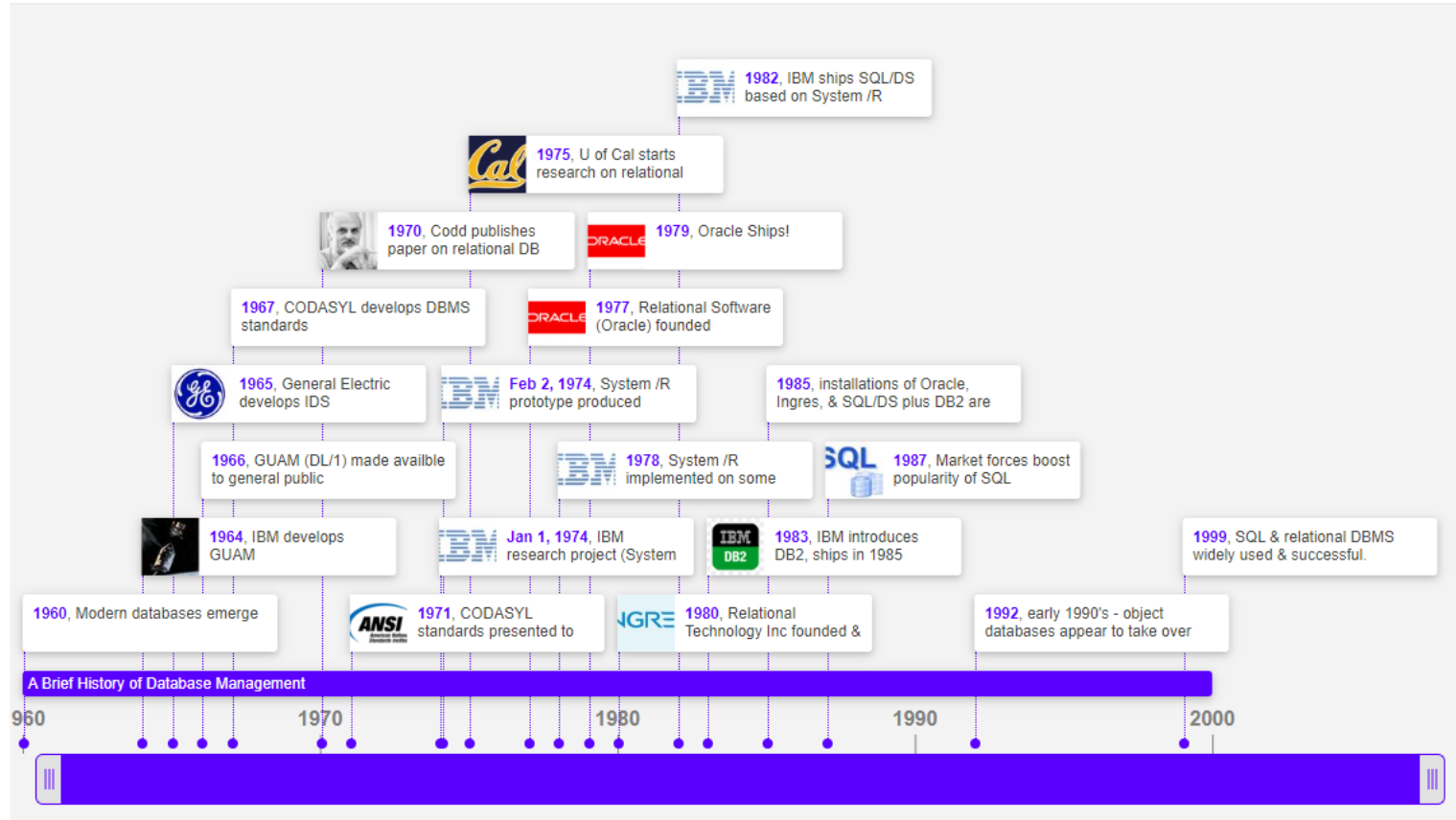
Lernziel / Fragen

- Wie sind Datenbanken geschichtlich entstanden?
- Wozu dienen Datenbanken? Welchen wirtschaftlichen Zweck verfolge ich damit?
- Warum sind Datenbank nicht immer die Lösung?
- Wie sieht die grobe Struktur einer Datenbank aus?

Begriffsklärung und Motivation. Sie wissen, was wir unter einer Datenbank verstehen und wo sie eingesetzt werden kann.

Sie kennen die grobe Struktur und bekommen einen Überblick der Architektur eines DBMS.

Geschichte von Datenbanken – Zeitleiste



- Dateiverwaltungssysteme (SAM, ISAM) unterstützen den index-/ sequenziellen Zugriff auf Dateien
- Entstehung erster Systeme, die Programmlogik und Datenhaltung trennen; Konsistenzsicherung
- Im NASA Apollo Moon Project wird 1964 die erste hierarchische Datenbank (GUAM) von IBM veröffentlicht.
- General Electric veröffentlicht 1965 mit dem System IDS die erste Netzwerkdatenbank.
- Die Database Task Group (DBTG) liefert Ende der 1960er erste DBMS-Standards
- Codd skizziert 1969 / 1970 die Grundlagen relationaler Datenbanken als Gegenentwurf zu Netzwerk- und hierarchischen Datenbanken → Trennung von logischem und physischem Modell sowie Definition der Relationenalgebra

1970er: Der Funktionsumfang wächst und DBn werden für unterschiedliche Betriebssysteme verfügbar.

- 1971: erste Spezifikation einer DML und DDL für Netzwerkdatenbanken durch die DBTG
- 1974: erste Abfragesprachen wie SEQUEL (später SQL) für System /R (IBM) und QUEL für Ingres (Univ. Berkeley)
- 1976: Chen spezifiziert das ER-Modell

- 1979 Relational Software (später Oracle) entwickelt auf Basis der IBM-Spezifikationen „Oracle“, das 1979 ausgeliefert wird

In den 1980ern Verdrängung anderer Ansätze durch relationale Systeme.

- IBM führt 1982 SQL/DS und 1983 DB2 ein
- Anbieterblüte: RIM, RBASE 5000, PARADOX, OS/2 Datab. Mgr., Dbase III, IV
- NIST veröffentlicht einen Standard und ein Testing Programm, für RDBMS der US-Regierung → De facto-Standard

1990er: Marktbereinigung und Datenbankanbieter reichern ihre Produkte mit Funktionen für das Management und Entwicklung an (SQL*Plus, Oracle Developer)

- Datenbanksysteme bilden im WWW den Backbone für E-Commerce-Anwendungen.
- Programmierschnittstellen für Java und C++
- Erste Open Source-Datenbanken (MySQL)
- SQL-Standard nimmt an Umfang zu (Call Level Interface, Object Language Binding, Java Routines and Types, XML)

2003: Weiterentwicklung SQL-Standard + „ISO Reference Model of Data Management“
→ Framework für Entwicklung von Standards für das Management persistenter Daten und Datenbanken

2000er und 2010er: neue Konzepte der Datenhaltung. Hadoop und Map Reduce → NoSQL entsteht; Big Data Frameworks

Mitte 2010er: Cloud-Datenbanken

Verfügbarkeit immer höherer Rechenleistung führt dazu, dass Verfahren zur Datenanalyse, z.B. das Data Mining möglich werden.

Welche Themen sind heute im Bereich
Datenbanken aktuell?

Was hat sich seit den 2010ern im Umfeld
von „Datenbanken“ geändert?

Datenbank und DBMS (1. Definition)

I) Database:

1. A set of data that is required for a specific purpose or is fundamental to a system, project, enterprise, or business.
Note: A database may consist of one or more **data banks** and be geographically distributed among several repositories.
2. A formally structured collection of data.

Note: In automated information systems, the database is manipulated using a database management system.

II) Database management system (DBMS): A software system that facilitates

- the creation and maintenance of a database or databases,
- the execution of computer programs using the database or databases.

<http://www.its.bldrdoc.gov/fs-1037/> ; Institute für Telecommunication Sciences (IST), Colorado; Nationale Forschungseinrichtung des U.S. Department of Commerce (DOC).

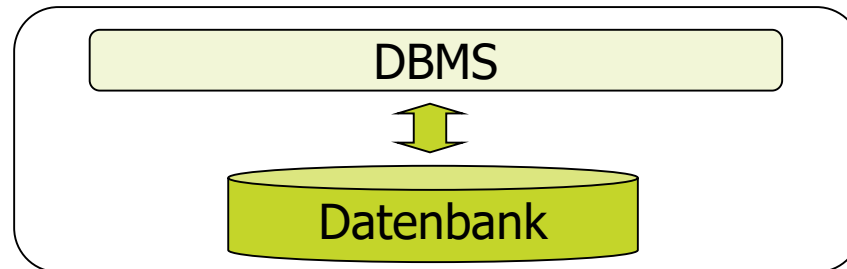
Datenbank und DBMS (2. Definition)

Eine **Datenbank** ist eine Sammlung logisch zusammenhängender Daten aus einem Sachgebiet.

Die Daten der DB beschreiben einen Ausschnitt der realen Welt.

Die Daten werden **persistent** für eine flexible Nutzung gespeichert.

Die Verwaltung der gespeicherten Daten und die Realisierung der Zugriffe auf die Daten obliegt dem **Datenbankmanagement-System**.



Datenbank - Eigenschaften

- Daten persistent speichern (=dauerhaft)
- Zugriff durch standardisierte Sprachen (SQL, XQuery, ...) oder Schnittstellen (JDBC, ODBC, Web Services ...)
- Daten können parallel von vielen Anwendern genutzt werden.
- Der parallele Zugriff wird über so genannte „Transaktionen“ koordiniert.
- Daten in der Datenbank folgen einer vorgegebenen Struktur
- Sicherheit (Zugriffsschutz, Backup und Recovery)

Vorteile des Datenbankeinsatzes

- Standards wie SQL stellen sicher, dass eine breite Community der Nutzerinnen und Nutzer besteht.
- Kürzere Entwicklungszeiten entstehen z.B. durch die Wiederverwendung derselben Datenbank in unterschiedlichen Programmen.
- Hohe Flexibilität: Änderungen an Daten führen nicht (mehr) zwangsläufig zu Änderungen in Programmen.
- Hohe Verfügbarkeit: DBn werden durch Spezialistinnen und Spezialisten betreut und architektonisch „ausfallsicher“ aufgesetzt.
- Backup und Recovery: Durch Sicherungen und Backups kann ein Totalverlust von Unternehmensdaten sichergestellt werden.
- Konsistenz: DBn definieren starre Regeln, an „die sich die Daten zu halten haben“
- Zugriffsschutz - Berechtigungen

Markt für Datenbanken

- Vielzahl von Produkten für unterschiedliche Daten-Arten (relationale, hierarchische, Netzwerk...) und Verwendungszwecke (Unternehmensweit, Client-Datenbanken...)
- Relationale Datenbank-Management-Systeme (RDBMS) speichern Daten in Tabellen / Relationen

Relationale Datenbanken - Überblick

- Sehr robust und seit Jahrzehnten im Einsatz
- Relationales Modell: Schemata, Relation, Attribute, Domain, Keys, referenzielle Integrität
- Transaktionskonzept
- Mengenorientierte Abfragesprachen: SQL und relationale Algebra
- Views und Rollen ermöglichen Anwendungsintegration und Compliance
- Komplexität steckt "unter der Haube"
 - Indizierung, QEP, Page-Design, Tablespaces, ...
 - Im Unternehmen Spezialisten benötigt



„Probleme“ Relationaler Datenbanken

- Eignen sich nicht für unstrukturierte und teilstrukturierte Daten oder Datenströme
- Anfragen müssen von SQL in eine DB-verständliche Form übersetzt werden → langsam (jedoch Optimierung möglich)
- Eine sauber aufgesetzte Datenbank geht mit Aufwänden für deren Entwurf einher: Datenmodell, Verteilungsmodell, Sichten ...
- Im Unternehmenseinsatz stoßen DBn häufig an Performance-Grenzen und müssen laufend "optimiert" werden
- Aber: weitgehende Unabhängigkeit der Daten von der zu programmierenden Applikationen / Reuse

Verteilte Datenbank

In **verteilten Datenbanken** (distributed databases) werden die Daten in mehreren Teildatenbanken (auf mehreren Rechnern) verteilt abgespeichert. Hierbei kann mit echt verteilten oder redundanten Daten (Replikaten) gearbeitet werden.

Es gibt Mechanismen für die Koordination der Zugriffe auf die verschiedenen Knoten. Sonst sprechen wir von einer zentralen Datenbank.

Homogen: Auf jedem Knoten läuft das gleiche DBMS

Heterogen: Einsatz verschiedener DBMS

Datenmodelle

Datenbanken werden mittels Datenmodellen entworfen

- **Logisches** bzw. **konzeptuelles Modell**: Konzepte dienen häufig zur Beschreibung der wahrgenommenen realen (oder gedachten) Welt
- **Physisches Datenmodell**: Konzepte zur Beschreibung von Details der Datenspeicherung auf dem Rechner. [NICHT TEIL DER VORLESUNG]

Konzeptuelle Modelle verwenden Konstrukte zur Darstellung des „Realitätsausschnitts“

- Objekte / Entitäten der realen Welt und deren Eigenschaften / Attribute
- Beziehung zwischen Entitäten und deren Eigenschaften / Attribute

Datenmodelle

- Die Strukturelle Beschreibung der Datenbank (ohne Daten) heißt **Datenbankschema**.
- Die Gesamtheit der gespeicherten Daten heißt **Datenbankzustand**.

Das Datenbankschema ist (über einen längeren Zeitraum) konstant und ändert sich in der Regel durch neue Anforderungen.

Der Datenbankzustand ändert sich in der Regel ständig durch Geschäftstransaktionen (Hinzufügen oder Löschen von Daten).

Drei-Schicht-Architektur / ANSI/SPARC

Drei-Schichten-Architektur nach ANSI/SPARC

Externes Schema bzw. Benutzersicht (View)

- Für Benutzer können individuelle Sichten auf die Daten definiert werden

Konzeptuelle Ebene bzw. konzeptuelles Schema

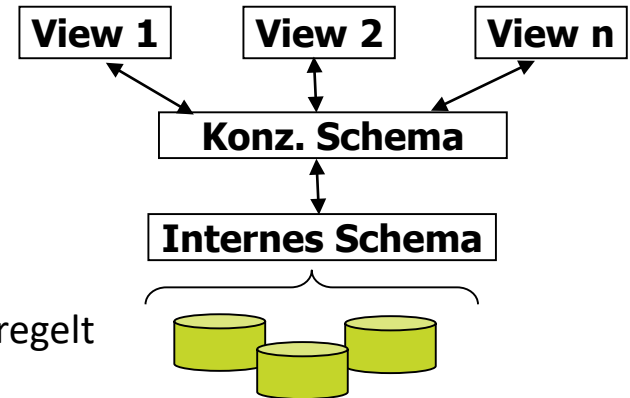
- Ebene der Entitäten und Beziehungen
- Unternehmensdatenmodell
- Integritätsregeln
- Oft auch „logisch“ genannt

Interne Ebene bzw. internes Schema

- Physikalische Speicherstrukturen

Der Transfer zwischen den Schemata wird durch Schnittstellen geregelt

Oft werden die obigen Schemata noch weiter differenziert.



Datenbank-Schemata im ANSI-Modell

- Zwischen den Schemata bestehen natürlich inhaltliche, sachlogische Abhängigkeiten
- Transfer zwischen den verschiedenen Schemata wird durch Zugriffsfunktionen (Schnittstellen) geregelt, die im DBS implementiert sind.
- Die technische Frage des Datenzugriffs – beispielsweise wie aus Datenfragmenten, die auf der Festplatte wild verstreut sind, eine Tabelle zusammengesetzt wird – spielt daher bei der Nutzung von Datenbanken so gut wie keine Rolle.
- Die drei Schemata oder Ebenen der Datenbankarchitektur können daher in gewissen Grenzen unabhängig voneinander gestaltet und betrachtet werden. Das bezeichnet man als **Datenunabhängigkeit**.

Unterstein, Michael; Matthiessen, Günter (2012): Relationale Datenbanken und SQL in Theorie und Praxis. Berlin, Heidelberg: Springer Berlin Heidelberg.

Datenunabhängigkeit (vgl. Datenmodelle)

Physische Datenunabhängigkeit

- Programmierer / Benutzer braucht für die Verarbeitung der Daten keine Kenntnis der Datenspeicherung (Blöcke, physische Schlüssel, ...)
- Spätere Änderungen des physischen Datenbankaufbaus in den Programmen müssen nicht berücksichtigt werden

Logische Datenunabhängigkeit

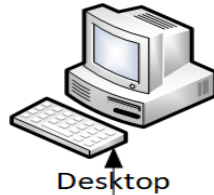
- Programmierer / Benutzer muss nur Beziehungen zwischen den Daten kennen, die auch tatsächlich für dieses Programm benötigt werden
- Spätere Änderungen im logischen Aufbau (neue Felder, Löschen von Feldern, kompatible Änderungen von Datentypen), die das jeweilige Programm nicht unmittelbar betreffen, müssen nicht berücksichtigt werden

Applikationen

- Datenbanksysteme sind zentraler Bestandteil von Applikationen
- Beispiele für Applikationen mit Datenbanken und vielen Anwendern:
 - Online-Shops /-Buchungssysteme
 - Geldausgabeautomaten
 - Marktplätze (ebay ...)
 - CRM- / Supply-Chain-Anwendungen
 - ...

Paralleler Zugriff

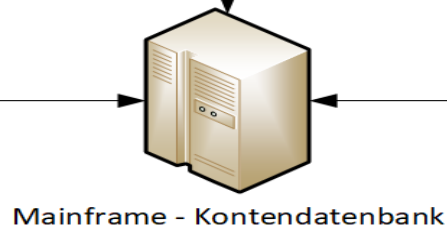
Abfrage nach Kontostand



Überweisung



Abbuchung



System zur Führung von Kontokorrentkonten einer Bank.

Paralleler Zugriff – Problem 1

Vorbedingung: Kontostand 3000 Euro

A liest Kontostand

A zieht 100 Euro ab

A schreibt neuen Kontostand zurück

B liest Kontostand

B zieht 300 Euro ab

B schreibt neuen Kontostand zurück

**Lost Update:
die Aktion von A geht verloren!**

Resultat: 2700 Euro – statt der korrekten 2600 Euro

Paralleler Zugriff – Problem 2

Vorbedingung: Kontostand 3000 Euro

- A liest Kontostand
- A zieht 100 Euro ab
- A schreibt neuen Kontostand zurück

Phantom:
B sieht eine verworfene Aktion

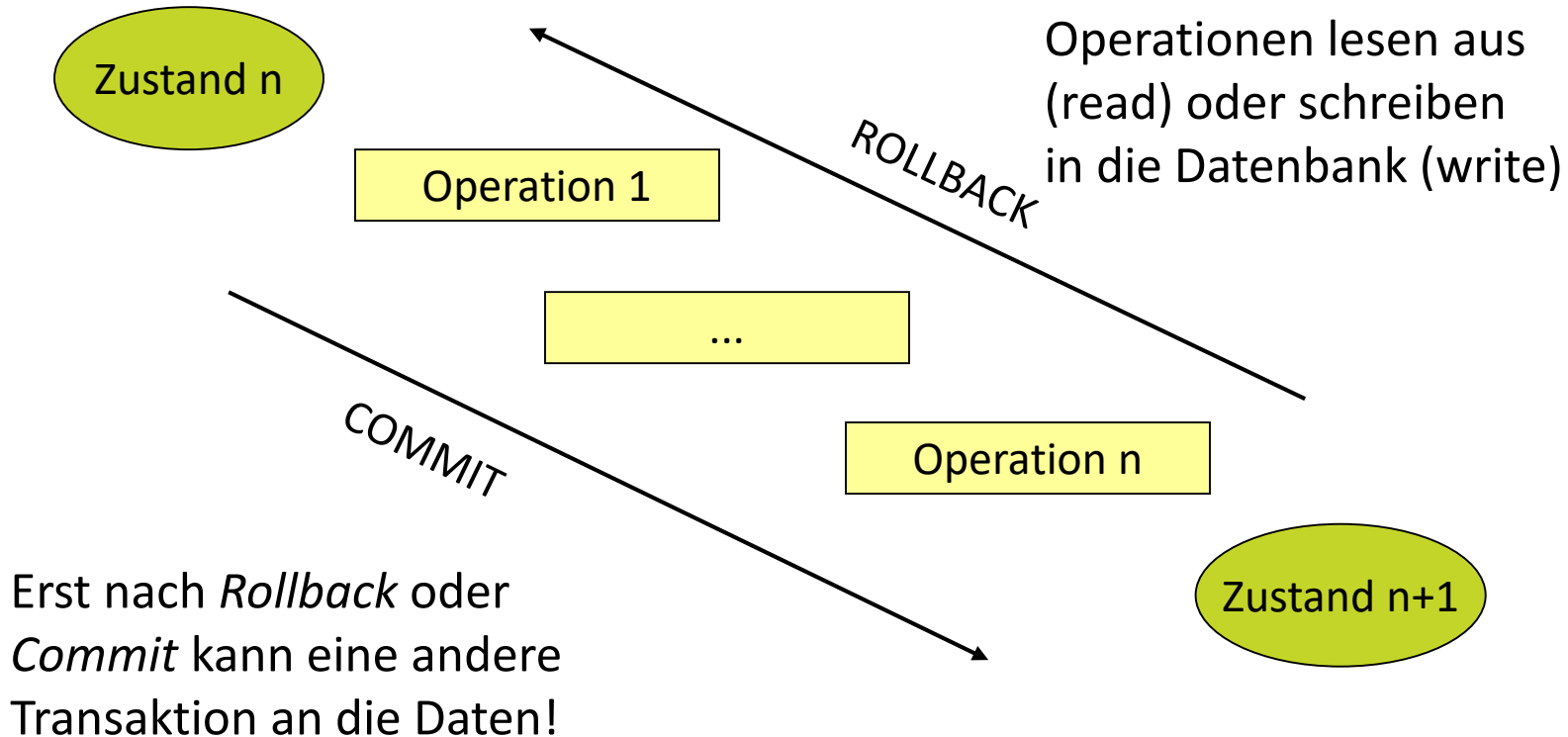
- B liest Kontostand
- B addiert 300 Euro
- B schreibt neuen Kontostand zurück

Resultat: 3300 Euro – statt der korrekten 3200 Euro

Transaktionen

- **Transaktionen** überführen die DB von einem konsistenten **Datenbankzustand** in einen weiteren konsistenten Zustand.
- Alle Änderungen werden aufgezeichnet (Log Mechanismus) und können rückgängig gemacht werden.
- *COMMIT* geht zum neuen Zustand
- *ROLLBACK* geht zurück zum Ausgangszustand
- Locking: Trennung nebenläufiger Transaktionen durch Sperren
 - Wenn A ein Konto bearbeitet kann B das Konto nicht bearbeiten
 - Unterschiedliche Arten von Locking

Transaktionen



Transaktionen – ACID

- **A** = atomic
Eine Transaktion ist atomar = unteilbar, sie wird entweder ganz oder gar nicht durchgeführt
- **C** = consistent
Eine Transaktion erzeugt einen konsistenten Zustand
- **I** = isolated
Keine Beeinflussung durch nebenläufige Transaktionen
- **D** = durable
Die Auswirkungen einer Transaktion sind dauerhaft

Lock Konflikte

- Die Sicherung der Konsistenz hat ihren Preis!
- Absicherung von Transaktionen mit Commit und Rollback führt zu anderen Problemen:

Engpässe durch Sperren (Locks).

Wenn wir jede Zeile sperren, die von einer Transaktion gelesen wird, so kann es zu langen Wartezeiten für die übrigen Transaktionen kommen.

Ein weiteres Risiko ist das Auftreten von Deadlocks.

Relationale Datenbanken: Tabellenstrukturen

Relationen / Tabellen

Eine relationale Datenbank ist eine Sammlung von **Relationen**, die als **Tabellen** dargestellt werden.

Die Spalten (columns) der Tabelle bezeichnen die Attribute der Relation. Die Zeilen (rows) heißen auch Datensätze oder Einträge.

Stellen Sie sich eine EXCEL-Tabelle vor!

🔑 ...	Titel	Nachname	Vorname	Gebdatum	SchulEintritt	Stufe
1	Dr.	Schmidt	Erika	1949-07-01	1974-08-01	StudienDirektorin
2	NULL	Schön	Helmut	1944-01-07	1971-03-01	StudienDirektor
3	NULL	Gliensmann	Jürgen	1964-06-06	1989-05-01	Studienrat
4	NULL	Derwall	Jupp	1954-03-02	1980-06-01	OberStudienrat
5	NULL	Nerz	Otilie	1958-05-06	1986-05-06	OberStudienrat
6	NULL	Lukas	Laura	1975-01-09	2000-08-17	Studienrätin
7	NULL	Meier	Horst	1955-04-04	1985-07-04	Oberstudienrat
8	NULL	Müller	Gerd	1971-02-02	2000-09-14	Studienrat
9	Dr.	Bauer	Renate	1961-05-01	1987-04-02	Oberstudienrätin
10	NULL	Hummel	Heinz	1965-02-02	1990-01-01	Studienrat

Relationen / Tabellen

Eine Relation besitzt


- einen Namen
- Zeilen (auch n-Tupel genannt) (0 bis viele)
- Attribute mit Domänen (Wertebereiche) und Namen (eins bis viele)

Die Wertebereiche können den Wert NULL enthalten.

Column Name	Datatype	NOT NULL	AUTO INC	Default Value	Flags	Commer
PNr	INTEGER	<input checked="" type="checkbox"/>		0	<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	
Titel	CHAR(5)			NULL	<input type="checkbox"/> BINARY <input type="checkbox"/> ASCII <input type="checkbox"/> UNICODE	
Nachname	CHAR(20)	<input checked="" type="checkbox"/>			<input type="checkbox"/> BINARY <input type="checkbox"/> ASCII <input type="checkbox"/> UNICODE	
Vorname	CHAR(20)	<input checked="" type="checkbox"/>			<input type="checkbox"/> BINARY <input type="checkbox"/> ASCII <input type="checkbox"/> UNICODE	
Gebdatum	DATE	<input checked="" type="checkbox"/>		0000-00-00		
SchulEintritt	DATE	<input checked="" type="checkbox"/>		0000-00-00		
Stufe	CHAR(20)	<input checked="" type="checkbox"/>			<input type="checkbox"/> BINARY <input type="checkbox"/> ASCII <input type="checkbox"/> UNICODE	

Relationen / Tabellen

- Ein Attribut oder ein Kombination aus Attributen, welche ein Tupel einer Relation eindeutig identifiziert heißt **Schlüsselkandidat**.
- Ein Schlüsselkandidat erhält die Rolle des Primärschlüssels.
- Der Primärschlüssel einer Relation ist grundsätzlich unveränderlich, da er die Identität des Tupels bestimmt (vgl. Objekt-ID).

Column Name	Datatype	NOT NULL	AUTO INC	Default Value	Flags	Commer
 PNr	INTEGER	<input checked="" type="checkbox"/>		0	<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	
Titel	CHAR(5)			NULL	<input type="checkbox"/> BINARY <input type="checkbox"/> ASCII <input type="checkbox"/> UNICODE	
Nachname	CHAR(20)	<input checked="" type="checkbox"/>			<input type="checkbox"/> BINARY <input type="checkbox"/> ASCII <input type="checkbox"/> UNICODE	
Vorname	CHAR(20)	<input checked="" type="checkbox"/>			<input type="checkbox"/> BINARY <input type="checkbox"/> ASCII <input type="checkbox"/> UNICODE	
Gebdatum	DATE	<input checked="" type="checkbox"/>		0000-00-00		
SchulEintritt	DATE	<input checked="" type="checkbox"/>		0000-00-00		
Stufe	CHAR(20)	<input checked="" type="checkbox"/>			<input type="checkbox"/> BINARY <input type="checkbox"/> ASCII <input type="checkbox"/> UNICODE	

Wertebereiche

Lernziel / Fragen

- Wie sind Datenbanken geschichtlich entstanden?
- Wozu dienen Datenbanken? Welchen wirtschaftlichen Zweck verfolge ich damit?
- Warum sind Datenbank nicht immer die Lösung?
- Wie sieht die grobe Struktur einer Datenbank aus?

Begriffsklärung und Motivation. Sie wissen, was wir unter einer Datenbank verstehen und wo sie eingesetzt werden kann.
Sie kennen die grobe Struktur und bekommen einen Überblick der Architektur eines DBMS.

Übungsaufgaben

Übung 1: Gruppenarbeit (4er Gruppen, pro Gruppe Präsentation)

1. Beschreiben Sie das ANSI-SPARC-Schema für Datenbanken.
2. Benennen Sie dazu die Ebenen und deren Funktion.

Recherchieren Sie im Internet / in Büchern:

1. Was verstehen Sie unter den Begriffen logische und physische Datenunabhängigkeit.
2. Welche anderen Datenformate werden heutzutage durch RDBMS unterstützt?
3. Wozu dienen konzeptuelle Modelle?
4. Was ist eine NoSQL-Datenbank?
 - Inwieweit unterscheiden sich NoSQL-Datenbanken von RDBMS?
 - Gibt es auch Gemeinsamkeiten?
5. Welche Arten von Daten würden Sie in RDBMS speichern, welche eher nicht?