**Stand:**  
12.11.2017 (Revision 1)  
Bis Folie 265

Zusammenfassung

DBI 3. Klasse

Michael Briedl

Inhalt

[1. Datenbanken 4](#_Toc498242712)

[1.1 Grundlegendes zu Datenbanken 4](#_Toc498242713)

[1.1.1 Arten von Datenbanken 4](#_Toc498242714)

[1.1.2 DB-Geschichte 4](#_Toc498242715)

[1.1.3 NoSQL 4](#_Toc498242716)

[1.1.4 CAP-Theorem 5](#_Toc498242717)

[1.1.5 Spaltenorientierte DB 5](#_Toc498242718)

[1.1.6 Hadoop 5](#_Toc498242719)

[1.1.7 SPRAIN 5](#_Toc498242720)

[1.2 Datenbankmanagementsysteme 5](#_Toc498242721)

[1.2.1 Filesystem (DB ohne DBMS) 5](#_Toc498242722)

[1.2.2 Was ist ein DBMS? 5](#_Toc498242723)

[1.2.3 Vorteile eines DBMS 6](#_Toc498242724)

[1.3 Geschichte 6](#_Toc498242725)

[1.3.1 Oracle 6](#_Toc498242726)

[1.3.2 Festplattengeschichte 6](#_Toc498242727)

[2 Datenbankmodellierung 7](#_Toc498242728)

[2.1 3 Sichten auf Daten 7](#_Toc498242729)

[2.1.1 Konzeptionelles Schema 7](#_Toc498242730)

[2.1.2 Internes Schema 7](#_Toc498242731)

[2.1.3 Externes Schema 7](#_Toc498242732)

[2.1.4 Implementierung eines 3-Schema-Systems 7](#_Toc498242733)

[2.2 Entity Relationship Diagrams 8](#_Toc498242734)

[2.2.1 Entities 8](#_Toc498242735)

[2.2.2 Entity-Sets 8](#_Toc498242736)

[2.2.3 Attribute 9](#_Toc498242737)

[2.2.4 Naming Conventions 9](#_Toc498242738)

[2.2.5 Relationships 9](#_Toc498242739)

[2.3 Von den Anforderungen zum ERD 10](#_Toc498242740)

[2.3.1 Heuristiken zu Entities 10](#_Toc498242741)

[2.3.2 Heuristiken zu Attributen 10](#_Toc498242742)

[2.3.3 Heuristiken zu Relationships 11](#_Toc498242743)

[2.4 Vom ERD zur Tabelle 11](#_Toc498242744)

[2.4.1 Relationales Modell 11](#_Toc498242745)

[2.4.2 Primary Keys 12](#_Toc498242746)

[2.4.3 Foreign Keys 12](#_Toc498242747)

[2.4.4 Vorgehen 12](#_Toc498242748)

# Datenbanken

## Grundlegendes zu Datenbanken

### Arten von Datenbanken

* Nach Struktur
  + Hierarchisch
  + Netzartig
  + Relational
  + Objektrelational
  + Objektorientiert
  + XML
* Nach Nutzeranzahl
  + Single-User
  + Multi-User
  + Rechnernetzwerk
* Nach Nutzungsform
  + Abfrage-DB
  + Transaktions-DB
  + Analyse-DB
    - ETL
      * Extract, Transform, Load
      * Daten laden, verändern und in DB schreiben
* Nach Abfragesprache
  + SQL
  + NoSQL
* Nach Speichernutzung
  + Massenspeicher (Standard)
  + In-Memory
* Die BESTE Datenbank
  + Anforderungsbedingt
  + Relationale/objektrelationale am häufigsten verwendet

### DB-Geschichte

* Edgar Frank Codd
  + "A Relational Model of Data for Large Shared Databanks"
  + Abfragesprachen
    - Alpha
    - Query by Example
    - SEQUEL (Structured English Query Language)

### NoSQL

* In manchen Fällen schneller
* Ansonsten unpraktisch wegen fehlender Konsistenz

### CAP-Theorem

* Consistancy
* Availability
* Partition Tolerance (Verteilung ohne Ausfall)

### Spaltenorientierte DB

* Zugriff über Spalten möglich
* Nicht an Zeilen gebunden

### Hadoop

* Daten auf 64MB Blöcke aufgeteilt
* Verteilt auf verschiedene Server über Kontinente verteilt
* Mehrfachsicherung möglich
* Kein DBMS

### SPRAIN

* Scalability
* Performance
* Relaxed consistency (siehe CAP-Theorem)
* Agility („Mut“ zu neuen Systemen)
* Intriacy (Datenkapazität)
* Necessity (Gebundenheit an Hersteller, Sicherheit dass Hersteller bestehen bleibt)

## Datenbankmanagementsysteme

### Filesystem (DB ohne DBMS)

* Redundant
* Schwierige Aktualisierung
* Sperre auf Dateiebene
* Bei Änderungen muss jedes Programm geändert werden
* Routinen (Suchen, Filtern) sind in jedem Programm zu implementieren
* Datenschutz schwierig

### Was ist ein DBMS?

User

DB

DBMS

Datenbasis

DBMS

Data  
Dictionary

Optimizer

SQL Interpreter

### Vorteile eines DBMS

* Wenig Redundanz
* Integrität (der Daten)
* Datensicherheit, -unabhängigkeit (von DBMS verwaltet, nicht Programm)
* Datensichten möglich (Views; nur bestimmte Teile von Daten sichtbar für bestimmte User)
* Effizient

## Geschichte

### Oracle

* 1979: Queries, Joins
* 1983: Transaktionen
* 1985: Client-Server Architektur
* 1992: Foreign Key Beziehungen, Stored Procedures
* 1997: objektorientierte Daten, BLOB
* 2003: Regex, Recycle-Bin

### Festplattengeschichte

* „Geburtstag“: 4.11.1959
* Kapazität erster Festplatte: 5MB
* GB-Grenze geknackt: 1997
* TB-Grenze: 2007
* Erste SSD: 1978
* HDD-Höchstgrenze 2017: 12TB
* SSD-Höchstgrenze: 15TB
* 60TB SSD bis Ende 2017 angekündigt (von Seagate)

# Datenbankmodellierung

## 3 Sichten auf Daten

### Konzeptionelles Schema

* Logisches Schema (Modell)
* Unabhängig von anderen Schichten
* Aufgaben des konzeptionellen Schemas
  + Beschreibung der logischen Dateien und des Satzaufbaus
  + Beschreibungen der Beziehungen (**Relationships**) zwischen Tabellen
    - **Relationen** sind Referenzen in Datensätzen
  + Beschreibung der Felder
  + Beschreibung der Gültigkeitsbereiche
    - Zum Beispiel: „Alter“ gilt nur zwischen 0 und 150

### Internes Schema

* Dazu gehören: DBMS, Datenbasis
* Aufgaben
  + Repräsentation von Zahlen (binär, dezimal, …)
  + Repräsentation von Zeichen (Unicode, ASCII, EBCDIC, …)
  + Speicherung von Datensätzen und Zugriff auf Datensätze
    - Hashes, sequentielle Indizes, Pfade, Verzweigungen, VSAM, Clustering, …

### Externes Schema

* UI, Druckvorlagen
* Nur Teilsichten
* Auch Teil d. externen Schemas: Views
  + Vorgefilterte Tabellen

### Implementierung eines 3-Schema-Systems

* Ziel: gutes konzeptionelles Schema
  + Andere können fast automatisch generiert werden
* Vorgehensweise
  + Anforderungsanalyse
  + Ergebnis: unstrukturierte Daten
    - Hauptsächlich Anforderungen für externe Sichten
  + Ordnen
  + Synonyme, Homonyme beseitigen
    - Wörter mit gleicher Bedeutung
    - Dabei hilft CASE-Software (Computer Aided Software Engineering)

## Entity Relationship Diagrams

* Kurz ERD
* Von **Peter Chen** (chinesischer Name: Chen Pin-Shan) und **Matt Flavin** entwickelt

### Entities

* „Ding“/Objekt
  + Begriff „Daten“ nicht passend: gehört zu internem Schema

DEFINITION

Individuelles und identifizierbares Exemplar von Dingen,  
Personen oder Begriffen der realen oder der Vorstellungswelt

* Können sein:
  + Dinge
  + Organisationen
  + Personen
  + Ereignisse
  + Grundsätze
  + Selten: wichtige Beziehungen
* Stärke eines Entities
  + **Fundamental**
    - Auch Kernentität oder Regular Entity genannt
    - Können alleine bestehen
  + **Abhängig** bzw. **Weak**
    - Brauchen Fundamentalentität um bestehen zu können
* Grafische Darstellung verhindert Missverständnisse

### Entity-Sets

* Enthalten Entities mit gleichen oder ähnlichen Merkmalen
* Werden oft (unsauber) mit Entities gleichgesetzt
* Können disjunkt oder überlappend sein:

A

B

**DISJUNKT**

Zum Beispiel: KFZ und PERSONEN

A

B

**ÜBERLAPPEND**

Zum Beispiel: SCHÜLER\_DER\_HTL und BAHNNUTZER

C

### Attribute

* Beschreiben Eigenschaft/Wert eines Entities
* Kriterien für Attribute:
  + Attribut zu Entity = 1:1
  + Attribut besitzt keine Attribute
  + Attribut ist nur Attribut des betroffenen Entities
* Wenn Kriterien nicht erfüllt -> Attribut wird zu Entitiy
* Haben Wertebereiche (z.B.: ASCII-Zeichen, alle Wochentage, 0 bis 100, …)

### Naming Conventions

* Singular
* Abkürzungen vermeiden
* Möglichst wenig Fachsprache
* Einheitlich bleiben
* Synonyme/Homonyme vermeiden
* Beschreibungen ins **Data Dictionary** (Metadatenkatalog) eintragen
* Keine…
  + Leerzeichen
  + Nicht-englischen Schriftzeichen (ß, Umlaute usw.)
* Hungarian Notation ist üblich

### Relationships

* Verbindungen zwischen Entities
* Legen fest, wieviele Entities aus Menge A Entities aus Menge B zugeordnet sind

E

E

E

Menge A

E

E

E

Menge B

E

E

E

Menge A

E

E

E

E

E

Menge B

E

E

E

Menge A

E

E

E

E

E

Menge B

**1 : 1**

Beziehung

**1 : N**

Beziehung

**N : M**

Beziehung

* Bestehen aus jeweils 2 **Assoziationen:**

|  |  |  |
| --- | --- | --- |
| Bezeichnung | | Anzahl der Entities |
| 1 | Einfache Assoziation | 1 |
| C | Konditionelle Assoziation | 0 oder 1 |
| m | Multiple Assoziation | ≥ 1 |
| MC | Multiple konditionelle Assoz. | ≥ 0 |

FAHRER

FAHRZEUG

fährt

wird gefahren von

KUNDE

AUFTRAG

erteilt

wurde erteilt von

ARZT

PATIENT

wird behandelt von

behandelt

FAHRER

FAHRZEUG

fährt

wird gefahren von

**1**

**1**

KUNDE

AUFTRAG

erteilt

wird erteilt von

**M**

**1**

ARZT

PATIENT

behandelt

wird behandelt von

**MC**

**M**

**Chen-Notation**

Benannt nach Peter Chen  
Eher selten

PROJEKT

MITARBEITER

**M**

**11**

Projektleitung

**Barker**

**Zehnder**

Notationen

## Von den Anforderungen zum ERD

* 2 Möglichkeiten:
  + Entities finden, ins Modell eintragen -> Attribute hinzufügen -> auf Normalform prüfen
  + Attr. existieren bereits (durch externe Schemen) -> Entities bilden -> auf NF prüfen
* Wichtigste Normalform: 3NF

### Heuristiken zu Entities

* Beschrieben mit Hauptwort
* Man soll etwas „im Auge behalten“
* Wenn unklar ob Attribut, Entity oder Relationship -> Entity
* Wenn Attribut mit -name, -nummer oder -code endet -> Schlüsselattribut eines Entities
* Wenn zu Attribut Definition besteht, die auf Entity hinweist

### Heuristiken zu Attributen

* Wenn etwas Wert annehmen kann
* Wenn Attributsdefinition sich auf Entity bezieht -> Entity zuordnen
* Wenn Def. sich auf mehrere Entity-Sets bezieht -> **Relationship-Set** erstellen, um Entity-Sets zu verbinden -> Attribut dem Relationship-Set zuordnen
  + Relationship-Sets werden wie Entity-Sets behandelt
* Wenn Attribut nicht zu allen Instanzen eines Entity-Sets passt -> Entity-Set unterteilen

### Heuristiken zu Relationships

* Beschrieben mit Verb
* Alleinstehendes Entity-Set -> Relationships zu anderen Entity-Sets suchen
* 2 Attribute von verschiedenen Entities sind nahe beisammen
* Eine Definition enthält mehr als ein Entity
* Mehrere eindeutige Schlüssel in Datensatz

## Vom ERD zur Tabelle

### Relationales Modell

* Datenbanknahe
* Von Edgar F. Codd entwickelt

AirlineRoute

Aus

wird:

|  |  |  |  |
| --- | --- | --- | --- |
| AirlineRoute | | | |
| **Flugnr** | **Airport** | **Startzeit** | **Gate** |
| LH4711 | LOWS | 14:32 | 1 |
| PA333 | LOWS | 16:20 | 1 |
| AN2345 | LOWW | 12:30 | 3 |
| SU6789 | LOWG | 14:29 | 1 |

Attribute

Relationsname

Tupel

Relation

* Begriffe:
  + **Tupel:** Tabellenzeile/Datensatz
  + **Tabelle:** Menge von Tupeln
  + **Kardinalität:** Anzahl der Tupel
  + **Grad:** Anzahl der Attribute
* Tabelle (Relation)
  + Menge von Tupeln
  + Eindeutiger Name
  + Normalerweise >1 Zeile
  + Ordnung der Zeilen egal (über Werte angesprochen)
  + Spalten (Attribute) auch >1, Ordnung egal
* Mathematisch

DEFINITION

Eine Relation ist eine Teilmenge eines kartesischen Produktes über nicht disjunkte Wertebereiche

Eine Relation ist eine Menge von Tupeln der Form mit

* + Begriffe:
    - Kartesisches Produkt: Menge aller Wertekombinationen von bis

### Primary Keys

* **Candidate Keys (CK)**
  + „Kandidaten“ für Primary Key
  + Hat jede Tabelle min. einmal
  + Können aus mehreren (nach wie vor getrennten) Spalten bestehen
* **Primary Key (PK)**
  + Einer pro Tabelle
  + Aus CK ausgewählt
  + Sollte leicht schreibbar und sprechend sein
* **Alternate Keys (AK)**
  + Alle CKs, die nicht PK wurden

### Foreign Keys

* Gewöhnliche Attribute
* Zeiger auf Zeile/Datensatz in anderer Tabelle
  + Dort ist er PK
* Können (natürlich) Teil des PK sein
* Können auch auf Zeilen in eigener Tabelle zeigen -> Hierarchie
* Kann NULL sein (außer wenn in PK)

### Vorgehen

* Aus jeder Entitätsmenge wird Tabelle (braucht PK)
* 1:N Relationships werden zu FK in N-Tabelle
* Aus M:N Relationships werden assoziative Tabellen

ANGESTELLTER

PROJEKT

wird bearbeitet von

arbeitet an

ANGESTELLTER

PROJEKT\_MITARBEIT

PROJEKT

Wird zu

Verhältnis Bestellung zu Artikel = M : N

BEST

BEST\_ART

ART

**Primary Key:**

Nicht ideal: Datensatz muss gelöscht werden, um Bestellung zu bearbeiten

BEST\_ART(BestNr, ArtNr, …) ->

Besser: ArtNr kann geändert werden

BEST\_ART(BestNr, LfdNr, ArtNr, …) ->

Nicht ideal: Bestellung 119 könnte LfdNr 236, 354 und 512 haben (also völlig zusammenhangslos)

BEST\_ART(LfdNr, BestNr, ArtNr, …) ->

Beispiel Bestellung

* Attribute werden zu Spalten
* Relationships zu Foreign Keys