# Datenorganisation | DORG

# Datenbank-Entwurf

DORG

## Datenbanksystem DBS

strukturierte Datenorganisation
Trennung der Daten von den Anwendungen

DBS = DB + DBMS

- Datenbank DB:
  Datenbestand, die eigentlichen Daten
- Datenbank-Managementsystem DBMS:
  Datenbankverwaltungssystem

## DBMS

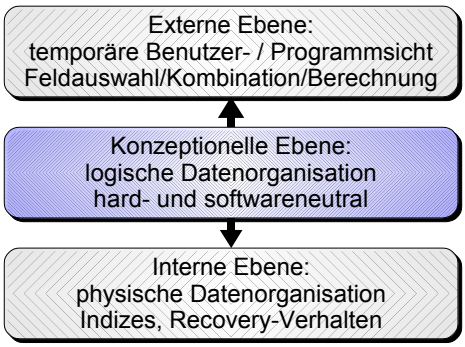
Bestandteile:
- DDL: Data Definition Language (Objektdefinition)
- DML: Data Manipulation Language (Datenfunktionen)
- DCL: Data Control Language (Zugriffsrechte)
- Dienstprogramme

## Funktionen DBMS

- Persistente Datenhaltung
- Speicherverwaltung im Hintergrund
- Datenintegrität / Konsistenz:
  - semantisch: inhaltlich, benutzerdefinierte Bedingungen
  - operational: konsistente Zustandsübergänge
  - logisch: referentielle Integrität
  - physisch: Zugriffspfade, Speicherstrukturen
- Ad hoc-Abfragen
- Mehrbenutzerbetrieb mit unterschiedlichen Zugriffsrechten
- Wiederanlauf

## Architektur DBMS

Modell von ANSI (American National Standardisation Institute)




## DB-Entwurfsphasen

- Vorstudie:
  Realitätsanalyse

- Hauptstudie / Definitionsphase:
  Semantisches Datenmodell,
  konzeptioneller Entwurf (ERM)

- Detailstudie / Entwurfsphase:
  Normalisierung,
  Logisches Datenmodell (DBMS-spezifisch)

- Systembau / Realisierungsphase:
  Physisches Datenmodell

## Realitätsanalyse

Grundlagen für den konzeptionellen Entwurf erarbeiten:
- Anforderungen, Informationserhebung
- Kernentitäten (haben keine Fremdschlüssel)
- Beziehungen mit Kardinalitäten
- Schlüssel-Attribute
- wichtigste Attribute

Verfahren:
- Top down: komplexe, neue Systeme
- Bottom up: kleine Systeme, Redesign
- Inside out: globale Datenmodelle

# Semantisches Datenmodell

DORG

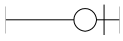
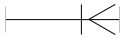
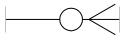
## Zielsetzung Datenmodellierung

- Analysieren des Untersuchungsbereichs aus der datenorientierten Perspektive
- Festlegen von eindeutigen Begriffen für Datenobjekte, Datenfelder und Datenbeziehungen
- Erstellung eines konzeptionellen, redundanzfreien Datenmodells
- Einordnung des projektspezifischen Datenmodells in ein unternehmensweites Datenmodell

## Elemente der Datenmodellierung

- Entität, Objekt, Datensatz, Record, Zeile, Row, Tupel: individuelles und identifizierbares Exemplar einer Sache, einer Person oder eines Begriffs
- Entitätsmenge, Entitätstyp, Tabelle, Relation: eindeutig benannte Kollektion von Entitäten gleichen Typs
- Entitätsattribut, Feld, Spalte: Eigenschaft
- Domäne: Wertebereich, zulässige Eigenschaften, durch Felddefinition oder Plausibilisierung
- Beziehung, Relationship, Assoziation: assoziiert wechselseitig zwei oder mehr Entitäten

## Notation Assoziationstypen

| Assoziationstyp | Kardinalität* | min:max | IEM** | C.A. Zehnder |
|---|---|---|---|---|
| einfach | genau ein | 1:1 |  | 1 |
| konditionell | kein oder ein | 0:1 |  | c |
| komplex | ein oder mehrere | 1:n |  | m |
| kondit.-komplex | kein, ein, mehrere | 0:n |  | mc |

* Mengenverhältnisse
** Information Engineering Method

## Beziehungsrelation

| Assoziation | 1 | c | m | mc |
|---|---|---|---|---|
| 1 | **1 – 1** | **1 – c** | **1 – m** | **1 – mc** |
| c | | c – c | c – m | c – mc |
| m | | | m – m | m – mc |
| mc | | | | mc – mc |

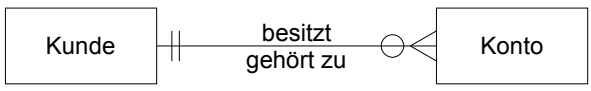
## ERM

Entity Relationship Model:
- Modell zur strukturierten Modellierung von Daten
- zur Erstellung des konzeptionellen Schemas
- unabhängig von der späteren Implementierung

Bestandteile:
- ERD
- Relationenmodell

## ERD

Entity Relationship Diagram:
Grafische Darstellung der Entitätstypen und deren Beziehung zueinander.




Beziehungen werden im Uhrzeigersinn beschrieben

## Relationenmodell

Auflistung der Entitätstypen auf Attributsstufe

Kunde (Kunden#, Name, Vorname, Adresse)
Konto (Konto#, Kunden#, Saldo)

Primärschlüssel, Primary Key, Identifikationsschlüssel:
- eindeutiger Identifikator eines Datensatzes
- kann aus einem oder mehreren Attributen bestehen
- unterstrichen

Fremdschlüssel, Foreign Key:
- verweist auf Primärschlüssel einer anderen Relation
- auf n-Seite einer 1-n-Beziehung
- kursiv oder gestrichelt unterstrichen

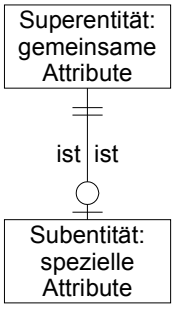
# ERM-Spezialfälle

DORG

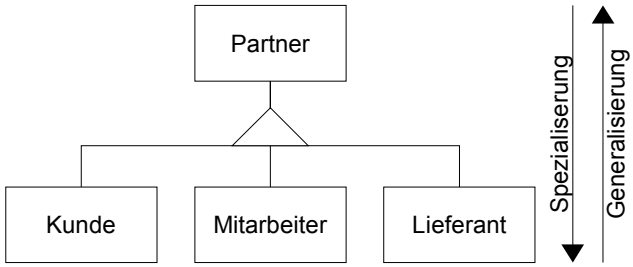
## Generalisierung / Spezialisierung

Charakteristisch für 1–c-Beziehungen
Prinzip der Vererbungshierarchie in OO

Primärschlüssel ist überall gleich →
Realisierung in einer Tabelle möglich
Vorteil: Zugriffsoptimierung
Nachteil: Speicherplatz



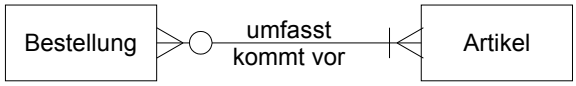

## Alternative Darstellung




braucht weder Kardinalitäten noch Beziehungsbeschreibung

## Komplexe Beziehung

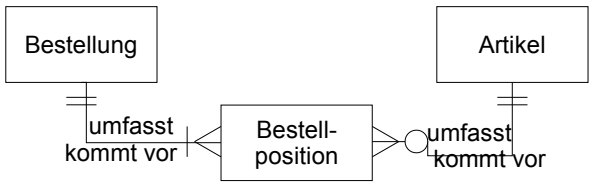
Beziehungen vom Typ m(c)–m(c) müssen aufgelöst werden.




## Aufgelöste Darstellung

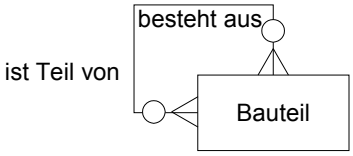
Aggregation: Umwandlung der Beziehung in eine Entität
Benennungsmöglichkeiten der neuen Entität:

- Kombination der beteiligten Entitäten
- Substantivierung der Beziehung (z.B. Teilnahme)



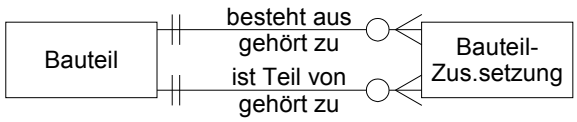

## Rekursive Beziehung

Beziehung eines Entitätstyps mit sich selber in verschiedenen Rollen.




## Aufgelöste Darstellung

Komplexe Beziehungen müssen wiederum aufgelöst werden.




Bauteil (Bauteil#, Bezeichnung, Farbe, Grösse)
Bauteil-Zus.setzung (*Ober#, Unter#*, Anzahl Unterteile)

# Normalisierung

DORG

## Zweck der Normalisierung

Bottom up-Verfahren zur Bereinigung bestehender Systeme.

- Speicheranomalien verhindern
- Redundanzen verhindern (mehrfach gespeicherte Informationen, d.h. Daten im gleichen Zusammenhang)
- Datenintegrität sicherstellen
- Strukturen flexibel halten

Das ERM enthält oft bereits Relationen in 2. oder 3. NF, der Normalisierungsprozess dient dann der Überprüfung.

## Unnormalisierte Form

Auflistung von Attributen zur Abbildung der Realität

**Rechnung:**

| Re# | Kunde | Artikel | Betrag |
|---|---|---|---|
| 01 | Schwarz, Zürich | Prozessor, Speicher, Monitor | 900 |
| 02 | Black, London | Monitor | 300 |
| 03 | Noir, Paris | Speicher, Prozessor | 600 |
| 04 | Black, London | Speicher | 200 |

## 1. Normalform

Alle Attribute atomar

Vorgehen:
- zusammengesetzte Attribute aufteilen (s. Kunde)
- Mehrfachwerte in separaten Zeilen führen (s. Artikel)
- abhängige Summen aufteilen
- Tabellentitel anpassen

## Beispiel 1. NF

**Rechnungsposition:**

| Re# | KuName | KuOrt | Artikel | Preis |
|---|---|---|---|---|
| 01 | Schwarz | Zürich | Prozessor | 400 |
| 01 | Schwarz | Zürich | Speicher | 200 |
| 01 | Schwarz | Zürich | Monitor | 300 |
| 02 | Black | London | Monitor | 300 |
| 03 | Noir | Paris | Speicher | 200 |
| 03 | Noir | Paris | Prozessor | 400 |
| 04 | Black | London | Speicher | 200 |

## 2. Normalform

1. Normalform +

Alle Attribute funktional vom Geamtschlüssel abhängig

Vorgehen:
- Abhängigkeiten analysieren (KuName mit KuOrt von Re#, Preis von Artikel)
- Identifikationsschlüssel bestimmen: alle unabhängigen Attribute (Re#, Artikel)
- nicht vom Gesamtschlüssel abhängige Attribute in separate Tabellen auslagern (KuName und KuOrt, Preis)
- Primärschlüssel ausgelagerter Tabellen ist Fremdschlüssel in ursprünglicher Tabelle

## Beispiel 2. NF

**Rechnungspos.:**

| *Re#* | *Artikel* |
|---|---|
| 01 | Prozessor |
| 01 | Speicher |
| 01 | Monitor |
| 02 | Monitor |
| 03 | Gehäuse |
| 03 | Prozessor |
| 04 | Speicher |

**Rechnung:**

| Re# | KuName | KuOrt |
|---|---|---|
| 01 | Schwarz | Zürich |
| 02 | Black | London |
| 03 | Noir | Paris |
| 04 | Black | London |

**Artikel:**

| Artikel | Preis |
|---|---|
| Prozessor | 400 |
| Speicher | 200 |
| Monitor | 300 |

## 3. Normalform

2. Normalform +

Alle Attribute direkt vom Schlüssel abhängig

Vorgehen:
- transitive (indirekte) Abhängigkeiten bestimmen (KuOrt von KuName)
- transitive Attribute in separate Tabellen auslagern (KuOrt)
- Primärschlüssel ausgelagerter Tabellen ist Fremdschlüssel in ursprünglicher Tabelle

## Beispiel 3. NF

Die Tabellen Rechnungsposition und Artikel bleiben gleich.

**Rechnung:**

| Re# | *KuName* |
|---|---|
| 01 | Schwarz |
| 02 | Black |
| 03 | Noir |
| 04 | Black |

**Kunde:**

| KuName | KuOrt |
|---|---|
| Schwarz | Zürich |
| Black | London |
| Noir | Paris |

# Logisches Datenmodell

DORG

## Zweck

Anpassung des semantischen Datenmodells an das DBMS des Zielsystems.

Denormalisierung:
Verletzung der 3. NF aus Performance-Gründen (weniger Relationen)

Übernormalisierung:
Auslagerung von Attributen in eigene Tabellen zur dynamischen Plausibilisierung von Auswahlwerten

## Umsetzung ERM – RDM

Regeln für die Umsetzung in ein relationales Datenmodell

| Asso-ziation | 1 | c | m | mc |
|---|---|---|---|---|
| 1 | (1) | (2) | (4) | (4) |
| c | | (3) | (5) | (5) |
| m | | | (6) | (6) |
| mc | | | | (6) |

## Regel 1

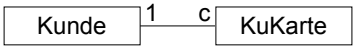
Umsetzung in 1 Relation


Mietwohung:

| MiWo# | Name | Adresse | Ort |
|---|---|---|---|
| 01 | Schwarz | Z-Strasse | Zürich |
| 02 | Black | L-Street | London |
| 03 | Noir | Rue P. | Paris |

## Regel 2

Umsetzung in 1 Relation mit Null-Werten (gute Performance) oder 2 Relationen ohne Null-Werte (weniger Speicherplatz)




Kunde:

| Ku# | Name | Punkte |
|---|---|---|
| 01 | Schwarz | 200 |
| 02 | Black | ---------- |
| 03 | Noir | 0 |

Kunde:

| Ku# | Name |
|---|---|
| 01 | Schwarz |
| 02 | Black |
| 03 | Noir |

Kundenkarte:

| Ku# | Punkte |
|---|---|
| 01 | 200 |
| 03 | 0 |

## Regel 3

Umsetzung in 3 Relationen


Mann:

| Ma# | Name |
|---|---|
| 01 | Schwarz |
| 02 | Black |
| 03 | Noir |

Heirat:

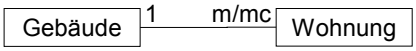
| *Ma#* | *Fr#* |
|---|---|
| 02 | 01 |
| 03 | 02 |

Frau:

| Fr# | Name |
|---|---|
| 01 | Black |
| 02 | Noir |
| 03 | Nera |

## Regel 4

Umsetzung in 2 Relationen




Gebäude:

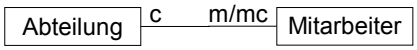
| Ge# | Name |
|---|---|
| 01 | Block X |
| 02 | Block Y |
| 03 | Fabrik |

Wohnung:

| Wo# | Name | *Ge#* |
|---|---|---|
| 01 | Schwarz | 01 |
| 02 | Black | 02 |
| 03 | Noir | 01 |

## Regel 5

Umsetzung in 2 Relationen mit Null-Werten oder 3 Relationen ohne Null-Werte




Abteilung:

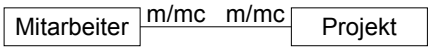
| Ab# | Name |
|---|---|
| 01 | Einkauf |
| 02 | Service |
| 03 | Verkauf |

Mitarbeiter:

| Mi# | Name | *Ab#* |
|---|---|---|
| 01 | Schwarz | 01 |
| 02 | Black | 03 |
| 03 | Noir | ------- |
| 04 | Nero | 01 |

## Regel 6

Umsetzung in 3 Relationen




Mitarbeiter:

| Mi# | Name |
|---|---|
| 01 | Schwarz |
| 02 | Black |
| 03 | Noir |
| 04 | Nero |

Projektarbeit:

| *Mi#* | *Pr#* |
|---|---|
| 02 | 01 |
| 03 | 02 |
| 04 | 01 |

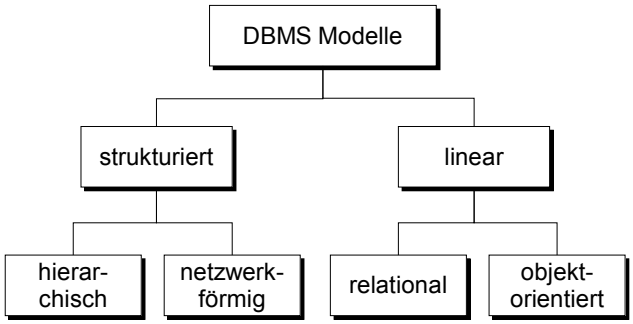
Projekt:

| Pr# | Name |
|---|---|
| 01 | Pluto |
| 02 | Neptun |
| 03 | Saturn |

# DBMS Modelle



## Übersicht



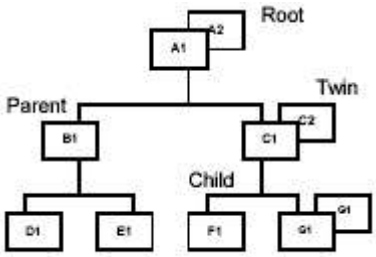

## Hierarchisch

- Baumstruktur aus Segmenten mit klarer Rangordnung
- Beziehungen über zusätzliche Pointer
- Verwendung: für sehr grosse Datenmengen
- Beispiele: Information Management System IMS

Vorteile:
- gute Performance

Nachteile:
- umständliche Navigation
- Änderungen schwierig
- Redundanz bei komplexen Beziehungen

## Grafik hierachisch




## Netzwerkförmig

- Statische Verknüpfung der Datensätze über Pointer
- beliebig viele logische Beziehungen
- beliebige Zugriffspfade, gezielte Zugriffe
- Modell von Codasyl: Owner- / Member-Record, Set (Beziehung), Next-, Prior-, Owner-Pointer

Vorteile:
- gute Performance
- komplexe Beziehungen möglich

Nachteile:
- hohe Komplexität
- Änderungen kompliziert
- heute selten verwendet

## Relational

- Ansammlung von zweidimensionalen über Schlüssel verknüpften Tabellen
- logisches Datenmodell kann beinahe unverändert übernommen werden
- Beispiele: DB2, Oracle, DBase, Sybase, Access

Vorteile:
- Portierbarkeit
- gut erweiterbar
- viele Tools

Nachteile:
- bei grossen Datenmengen schlechte Performance

# Physisches Datenmodell

DORG

## DDL

Data Definition Language

- Datenbankelemente erstellen, löschen, ändern
- Zugriffspfade (Indizes) implementieren
- Mengenverhältnisse berücksichtigen
- Referentielle Integrität implementieren

## Darstellung

Zugriffspfadmatrix:
- geplante Zugriffe auf die Daten

Konzeptionelles Strukturdiagramm:
- logische Datenstrukturen
- Datenmengen
- Integritätsbeziehungen

## Zugriffspfadmatrix

Gilt für bestimmte Verarbeitung der Daten in Prozessen.
Geplante Zugriffe Batch / Online pro definierter Zeiteinheit.

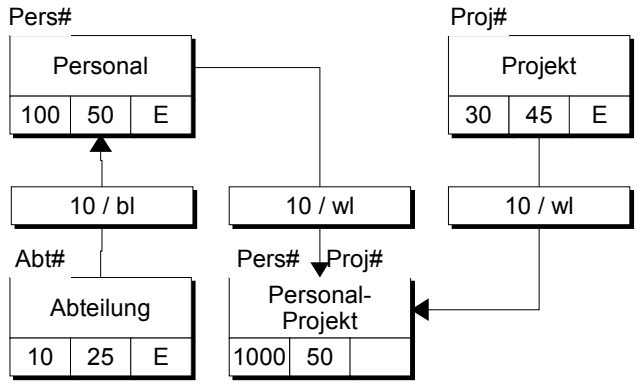
| Relation von nach | Abteilung | Personal | Pers.-Projekt | Projekt |
|---|---|---|---|---|
| Abteilung | | 1 / 10 | | |
| Personal | | | 1 / 10 | |
| Pers.-Projekt | | | | 1 / 10 |
| Projekt | | | | |

## Logische Datenstrukturen

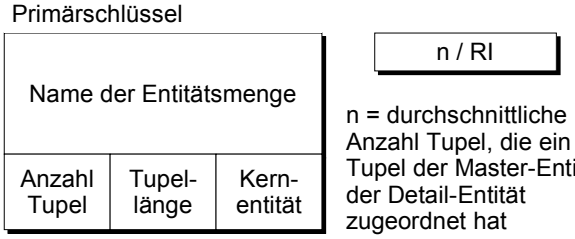
Attribut-Definition:
- Attributeigenschaften: Feldart und Feldlänge
- Attributwerte: Domäne
- Attributanordnung im Datensatz

Datensatz-Kennzahlen:
- Berechnung Gesamtlänge Datensatz (Paddings beachten)
- Schätzung der Anzahl Datensätze pro Tabelle

## Konzeptionelles Strukturdiagramm




## Erklärung Strukturdiagramm




n = durchschnittliche Anzahl Tupel, die ein Tupel der Master-Entität der Detail-Entität zugeordnet hat

## Referentielle Integrität RI

Der Fremdschlüssel darf nur Werte annehmen, welche in der entsprechenden Relation als Identifikationsschlüssel existieren.

Die Regeln der referentiellen Integrität stellen sicher, dass beim Löschen oder Ändern eines Identifikationsschlüssels keine ungültigen Werte in den zugehörigen Fremdschlüsseln entstehen.

## Regeln RI

On delete/update:
- cascade: Weitergabe der Löschung/Modifikation (wl/wm) an Datensätze mit dem zugehörigen Fremdschlüssel
- restrict: Bedingte Löschung/Modifikation (bl/bm) Abbruch falls zugehörige Fremdschlüssel vorhanden sind
- set null: Nullsetzung bei Löschung/Modifikation (nl/nm) (Null bedeutet nicht existent)
- set to default: Setzung eines definierten Defaultwertes
- ignore: Missachtung der referentiellen Integrität (normalerweise nicht erlaubt)