

Datenbanken

Grundlagen

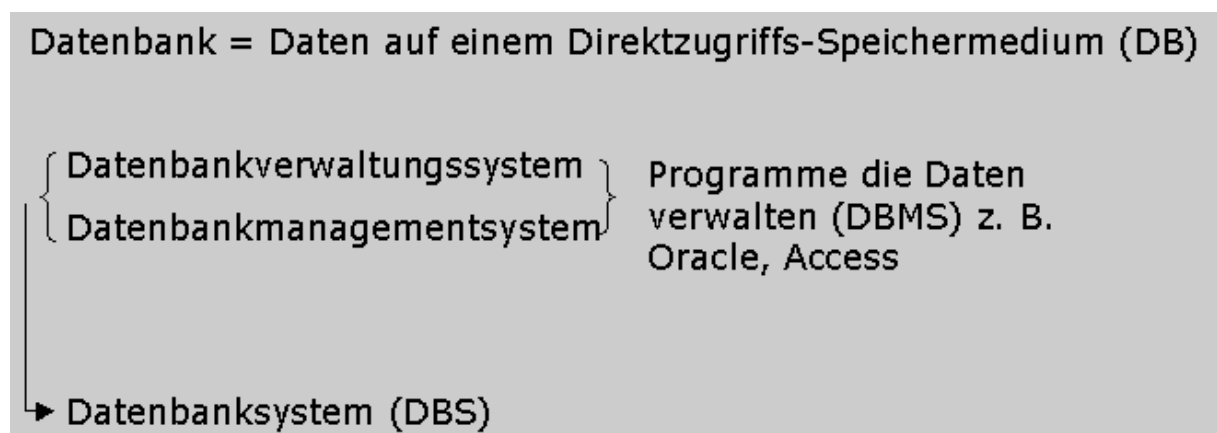
Dateiorganisation

- Sequentiell: ein Datensatz nach dem anderen
Zugriff: sequentiell
- Gestreut: Datensätze werden über einen Rechenalgorithmus auf bestimmte Adressen auf der Platte "verstreut"
Zugriff: direkt, Adresse auf der Platte kann berechnet werden und es kann direkt auf den Datensatz zugegriffen werden!
- Index- sequentiell
In sequentieller oder
mit einer Verwaltung des Schlüsselbegriffes in einer separaten Indextabelle

Nachteile herkömmlicher Dateien:

- Abhängigkeit zwischen Programm und Datei - für jedes Programm eigene Datei(en)
- Redundanz: Daten kommen mehrfach vor
- Speicherplatzverschwendung
- Inkonsistenzen = falsche Daten

Begriffe



- Standard - Anwendungsprogramm
- Anwendungsprogramme - betriebsspezifisch

Begründung für Datenbanken

- Erhöhter Datenanfall
- Daten wurden als zentrales Betriebsmittel (Ressource) erkannt
- Daten zentral halten und verwalten
- Daten den Mitarbeitern zur Erfüllung des betrieblichen Zwecks zur Verfügung stellen → jeder bekommt nur einen Ausschnitt der Daten zu sehen (d.h. einzelne Datenfelder)!

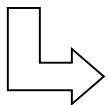
Schichten eines Datenbanksystems:

Datenbank = modellhafte Abbildung des Unternehmens in Form von Daten und ihren Beziehungen (vollständig)

Bei der Entwicklung eines Datenbanksystems müssen folgende Sichtweisen beachtet werden:

- Logische Gesamtsicht → Datenbankdesigner
Beschreibung der relevanten Daten und Beziehungen (unabhängig von EDV-Gesichtspunkten)
- Interne Sicht → Systemadministrator
- Externe Sicht → Sichtweisen der einzelnen Benutzer

1. Schritt	→ 2. Schritt Modellbildung	3. Schritt Schemata
Logische Gesamtsicht	→ Konzeptionelles Modell Konzeptuelles Modell	Konzeptionelles Schema
Interne Sicht	→ Internes Modell	internes Schema
Externe Sicht	→ Externes Modell	externe Schemata



Umsetzung der Modelle in die "Sprache" des DB-Systems.

Konzeptionelles / Konzeptuelles Modell

bildet folgendes ab:

- Daten
- Beziehungen

Darstellungsmethoden

ER - Modell = Entity - Relationship - Modell

- **Entity** (Entität)
Ein eindeutig identifizierbares Objekt der "realen Welt"

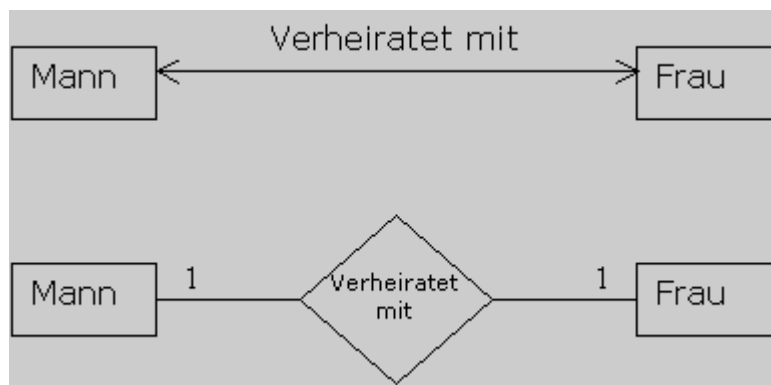
- **Eigenschaften** = Attribute
z. B. KdNr, KdName, KStr, KdPlz.
- **Entities** mit den gleichen Eigenschaften (Attribute) können zu Entity - Typen (Entity - Sets) zusammengefaßt werden.
z. B. Kunde
= alle Entities die die Attribute KdNr, KdName, KStr, KdPlz.
- Beispiele:
 - Grundstück = Typ
 - Herr Meier = Entity
 - Eiffelturm = Entity
 - Brücke = Typ

Relations = Beziehungen

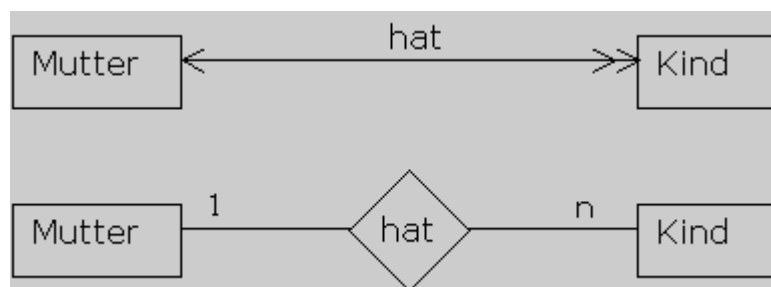
Entity - Typen und deren Entities stehen zueinander in Beziehung.
Beziehungen werden durch ihre Komplexität unterschieden!

Zwei verschiedene Darstellungsarten:

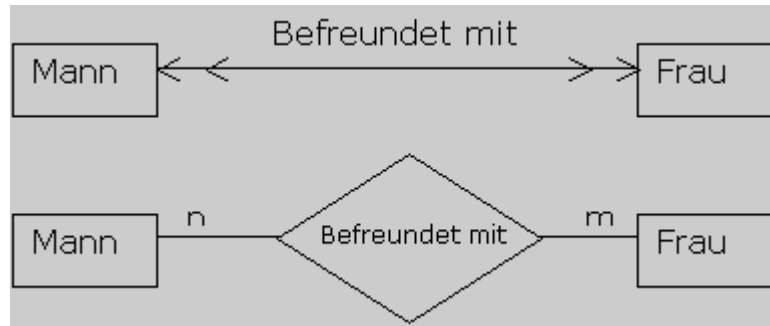
1:1 Beziehung



1 : n Beziehung

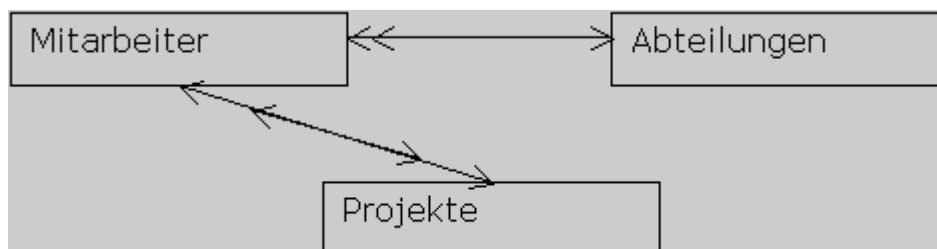


n : m Beziehung



Beispiele:

In einem Unternehmen gibt es Mitarbeiter und Abteilungen. Jeder Mitarbeiter gehört zu einer Abteilung, jede Abteilung hat mehrere Mitarbeiter. Es werden Projekte durchgeführt, wobei an jedem Projekt mehrere Mitarbeiter arbeiten können. Jeder Mitarbeiter kann auch mehreren Projekten zugeordnet sein.



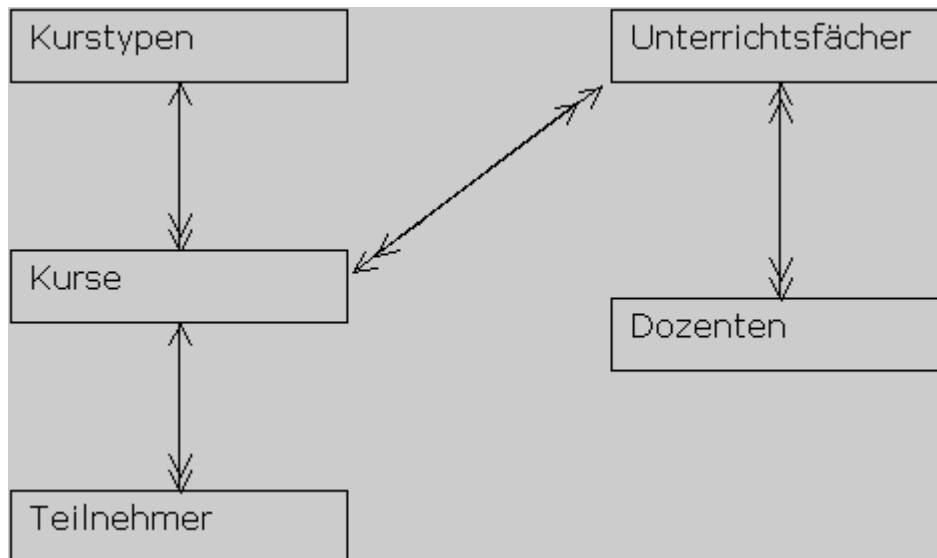
In einem Schulungszentrum gibt es mehrere Kurstypen, zu jedem dieser Typen gibt es mehrere Kurse.

Jeder Kurs ist genau einem Kurstyp zugeordnet.

Jeder Kurs hat mehrere Teilnehmer, wobei jeder Teilnehmer einem Kurs zugeordnet ist.

Außerdem gibt es Unterrichtsfächer, wobei ein Fach in mehreren Kursen unterrichtet wird, jeder Kurs mehrere Fächer beinhaltet.

Das Unternehmen beschäftigt Dozenten, wobei jeder Dozent mehrere Fächer unterrichtet, für jedes Fach gibt es mehrere Dozenten. [Wer's glaubt ;-)]



Exkurs Primärschlüssel
 Datenfeld(er) das (die) einen Datensatz
eindeutig identifizieren.

Sprachtypen bei Datenbanksystemen

DDL - Data Definition Language
 Sprache zur Definition von Daten

DML - Data Manipulation Language
 Daten verändern

Query Languages
 Abfragesprachen → Informationsgewinnung

Historische Entwicklung / Datenmodelle

Überblick:

- Hierarchisches Datenmodell
- Netzwerk-Datenmodell
- Relationales Datenmodell
- objektorientiertes Datenmodell

Hierarchisches Datenmodell (1968 - 1975)

- Zerlegung von Sätzen in hierarchisch voneinander abhängiger Segmente (Dateien). Die Beziehungen wurden über Adresszeiger realisiert.
- Einstieg ist immer nur über die höchste Hierarchiestufe (Root-Segment) möglich
- Werden Daten in Richtung der Hierarchie gesucht, so ist der Zugriff sehr schnell
- Werden Daten gegen die Richtung der Hierarchie gesucht, muss ggf. die gesamte Datenbank durchsucht werden. In diesem Falle sehr langsam.
- $n : m$ Beziehungen können nicht dargestellt werden!

Netzwerk-Datenmodell (1975 - 1980)

- mehrere Hierarchien
- durch Adresszeiger verbunden aber zusätzlicher Rückverweis zum Vorgänger
- Einstieg an beliebiger Stelle möglich
- Darstellung von $n : m$ - Beziehungen über Zerlegung in einen neuen Satztypen und zwei $1 : n$ Beziehungen möglich
- Nachteile:
 - Komplexität
 - Navigation
 - Verwaltungsaufwand
 - Keine einheitliche Sprache

Relationales Datenmodell

- Erste theoretische Grundlagen 1970 von E. F. Codd
- Erste lauffähige Systeme ca. 1980
- Fast alle heutigen Datenbanksysteme beruhen auf diesem Modell (z. B. ORACLE, ACCESS)
- Grundlage: Tabelle
 - Zeilen = Tupel
 - Spalten und Spaltenüberschriften
- Beziehungen werden durch die Verwendung gleicher Attribute in mehreren Tabellen dargestellt!
- Taucht ein Primärschlüssel einer Tabelle in einer anderen wieder auf, so wird er dort Fremdschlüssel genannt, da er die Beziehung zwischen den beiden Tabellen darstellt.
- Informationsgewinnung durch relationale Grundoperationen
- Normierte Sprache = SQL → Structured Query Language

Gegenüberstellung der Begriffe

ER - Modell	Relationales Modell	Herrkömmliche Dateien
Entity	Zeile / Tupel	Datensätze
Attribut	Spalten / Spaltenüberschriften	Datenfelder
Entity - Typ	Tabelle	Datei

Konzeptionelles Datendesign

Ausgehend von einer betrieblichen Problemstellung / Situation soll ein Datenmodell erstellt werden.

Abbildung der betrieblichen Situation in Form von Daten und Beziehungen

- vollständige Abbildung
- auch für zukünftige Anwendungen nutzbar
- Das Modell soll optimal sein (möglichst redundanzfrei)

Objektorientierter Ansatz à Datenobjekte à Entity - Typen

Begriffe	
Entity	= Entität
Entity - Typen	= Entitätsmenge
Attribut	
Domäne	= Wertebereich
Primärschlüssel	
Beziehungen	

Analytischer Zugang

- Im Unternehmen existieren Datenstrukturen
 - Funktionsabhängig (abteilungsbezogen)
 - sie enthalten Fachwissen
- Keine unternehmensübergreifende Informationen
- Strukturen analysieren
- zu Datenobjekten (Entitätsmengen) umwandeln
- Beziehungen ermitteln
- Optimieren

Relevante Attribute ermitteln

- Aus den vorhandenen Datenstrukturen die Datenelemente ermitteln (nicht weiterunterteilte Datenfelder)
- Entfernen Sie alle Datenelemente die
 - aus den übrig bleibenden Feldern errechnete oder errechenbare Werte enthalten
 - Texte oder Konstanten enthalten
- Es sind jetzt nur noch relevante Attribute übrig

Relationen ableiten

- Fassen Sie die gefundenen Attribute gemäss fachlicher Überlegungen zu Relationen / Entitätsmengen zusammen.
- Prüfen Sie, ob es in den Relationen Attribute gibt, bezogen auf eine Entität (Bestellung), die mehrere Werte annehmen können (z. B. mehrere Teile pro Bestellung)
Gibt es solche Attribute, dann bilden sie aus ihnen eine eigene Relation!
Der Primärschlüssel der Ursprungsrelation muss mit übernommen werden, um die Beziehung darzustellen.

Überschneidungen beseitigen

- Redundanzen zwischen Relationen beseitigen

Relationen normalisieren

Mathematischer Prozess zur Beseitigung von Redundanzen innerhalb einer Relation / Tabelle (max. 5 Schritte).

Aus den abhängig erkannten Attributen werden neue Relationen gebildet, Primärschlüssel ist das Attribut, von dem alle anderen abhängig sind!

Normalform:

Abhängigkeiten zwischen "einfachen" Attributen prüfen.

- Aus als abhängig erkannten einfachen Attributen eine neue Relation bilden
- Deren Primärschlüssel festlegen
- Diesen zur Beibehaltung der Beziehungen in der Ursprungsrelation belassen

Quellen:

Inbit, Paderborn