

Datenbanken

github/toiletcoders

Inhaltsverzeichnis

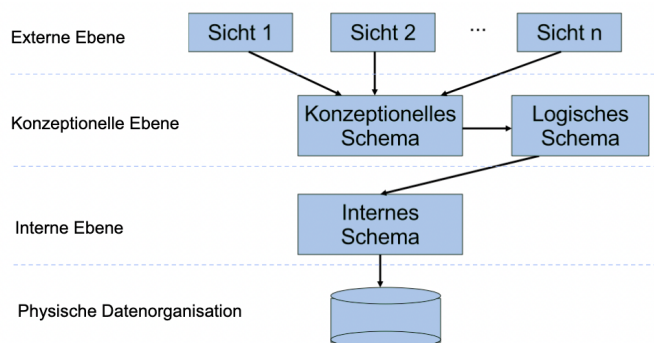
	Seite
1 Einführung	1
1.1 Grundlagen	1
2 Konzeptionelles Modell	5
2.1 Entitäten und Entitätstypen	5
2.2 Beziehungen	7
2.3 Beziehungen (weitere Konzepte)	8

1 Einführung

1.1 Grundlagen

1.1.1 ANSI-SPARC

- Ebene 1: Externe Ebene
Nutzer benötigen nur Teilausschnitt der Daten
Spezifikation der notwendigen Datensicht
- Ebene 2: Konzeptionelle Ebene
Vereinheitlichung der Sichten der Externen Ebene
Vollständige Beschreibung der für alle Anwendungen relevanten Objekte und deren Beziehungen
- Ebene 3: Interne Ebene
physikalische Darstellung der Datenbank im Computer
Speicherstrukturen zur Ablage der Daten



Eigenschaften:

- Gleiche Daten für alle Nutzer
- Änderungen in Nutzersichten sind lokal für die Anwendung
- Datenspeicherung (wie und wo) für Nutzer unwichtig
- Anwendungsstruktur für Datenbankaufbau unerheblich
- Änderung im Datenbankaufbau ohne Wirkung auf Nutzersicht

Beispiel:

Logische Datenunabhängigkeit:

- Anwendungen werden nicht beeinträchtigt, wenn Änderungen am Schema vorgenommen werden
- Bei logischer Datenunabhängigkeit:
Keine Änderung an Spezifikationen durch Hinzufügen, Ändern, Löschen von Objekten
- Nur teilweise gegeben in praktischen Datenbanksystemen

Physische Datenunabhängigkeit: Aus physischer Datenunabhängigkeit folgt:
Keine Änderung am konzeptionellen/externen Schema durch Umstellung der Dateistruktur, Speicherstruktur, Speichermedien, Anzahl der DB Server
Weitestgehend gegeben in praktischen Datenbanksystemen

1.1.2 CASE Werkzeuge

CASE = Computer Aided Software Engineering

Entwicklung von Software nach ingenieur-wissenschaftl. Methoden unter Verwendung eines Computers

Ziel: Erstellung von Software möglichst automatisch aus dem Fachkonzept CASE:

- Oftmals graphische Notation des Fachkonzepts
- CASE-Werkzeuge (Tools)
 - Planung
 - Entwurf
 - Dokumentation
- können in die IDE integriert werden

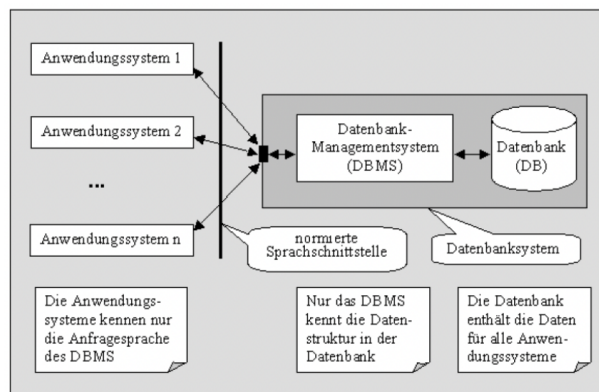
1.1.3 Datenbank- vs. dateibasierte Anwendungssysteme

Architektur Datei-basierter Anwendungen:

Zahlreiche Nachteile

- Redundanz
- Gemeinsamer Zugriff mittels Konvertieren
- Datenstrukturänderung bedingt Umprogrammieren
- Keine parallelen Zugriffe möglich
- Keine Sicherungsmechanismen

Architektur Datenbank-basierter Anwendungen



DBMS vs. DB

Datenbank

- Sammlung strukturierter Daten
- sachlogische Zusammenhänge untereinander

Datenbankmanagementsystem

- Programmsystem
- Systemsoftware für alle Aspekte der Datenverwaltung
- Beinhaltet eine oder mehrere DBen

Architektur DBMS

Zahlreiche Vorteile:

- Redundanzfrei
- Logische Datenunabhängigkeit
- Physische Datenunabhängigkeit
- Mehrnutzerbetrieb mit Rechteverwaltung
- Normierte Schnittstelle mit Effizienter Verwaltung

1.1.4 Phasenmodell für Datenbankentwurf

1. Anforderungsanalyse:

Anforderungen der potentiellen Benutzer werden erfasst

- Informelle Beschreibung
- Unterscheidung in Informations- und Bearbeitungsanforderungen

- Funktionenmodell: **Create, Read, Update, Delete** (CRUD)

2. Konzeptioneller Entwurf

- Erste formale Darstellung erstellen: konzeptionelles und externe Schemata
- Datenmodelle sind abstrakte Darstellungen der Wirklichkeit
- Ansätze zur Erstellung eines konzeptionellen Entwurfs:
 - * **Top-Down-Ansatz**
Modellierung des konzeptionellen Schemas und Ableitung der externen Schemata
 - * **Bottom-Up-Ansatz**
Modellierung der externen Schemata und anschließende Integration der externen Schemata zu einem konzeptionellen Schema.
Hierbei müssen i.d.R. Widersprüche und Konflikte zwischen einzelnen externen Schichten aufgelöst werden.

3. Logischer Entwurf

Das logische Schema beschreibt die Datenstrukturen des konzeptionellen Modells

- Entscheidung für verwendetes DBMS oder mind. für ein Datenbankmodell
- Transformation des konzept. Modells in Abhängigkeit der Anforderungen des Datenbankmodells
- Optimierung des Modells durch Vermeidung von Redundanzen im Rahmen der Normalisierung

4. Datendefinition

- Logisches Modell wird mit **Data Definition Language (DDL)** definiert
- Externe Schemata werden mit **View Definition Language (VDL)** definiert

Umsetzung des log. Schemas und der externen Schemata mit Hilfe der Datenbanksprache.

Diese Datenbanksprache ist i.d.R. SQL

Structured Query Language

SQL ist standardisiert, allerdings gibt es Dialekte je nach DBMS

2 Konzeptionelles Modell

2.1 Entitäten und Entitätstypen

2.1.1 Entity-Relationship Modell

Vereinfachte graphische Darstellung von Entitäten (Objekten), Beziehungen zw. den Objekten: Notwendigkeit, Anzahl der beteiligten Entitäten

Ziele:

- Bessere Kommunikation zwischen den Beteiligten
Experten (Fachabteilung), ANwendungsentwickler, ...
Beurteilung der Qualität des Modells
Grundlage zur Erstellung der Datenbank

2.1.2 Entitäten und Entitätstypen

Entität (Objekt)

- Ein Exemplar von
 - Konkreten (Studierender; Gebäude)
 - Abstrakten/ nicht-materiellen (Zugehörigkeit; Betreuungsverhältnis)
- Dient der Informationsspeicherung

Entitätstyp

- Eindeutig benannt
- "Gruppe" von Entitäten (Objekten)
- Speicherung gleichartiger Informationen
- Gleichartige Verarbeitungsverzögerung
- "Klasse" in der OOP

2.1.3 Attribut

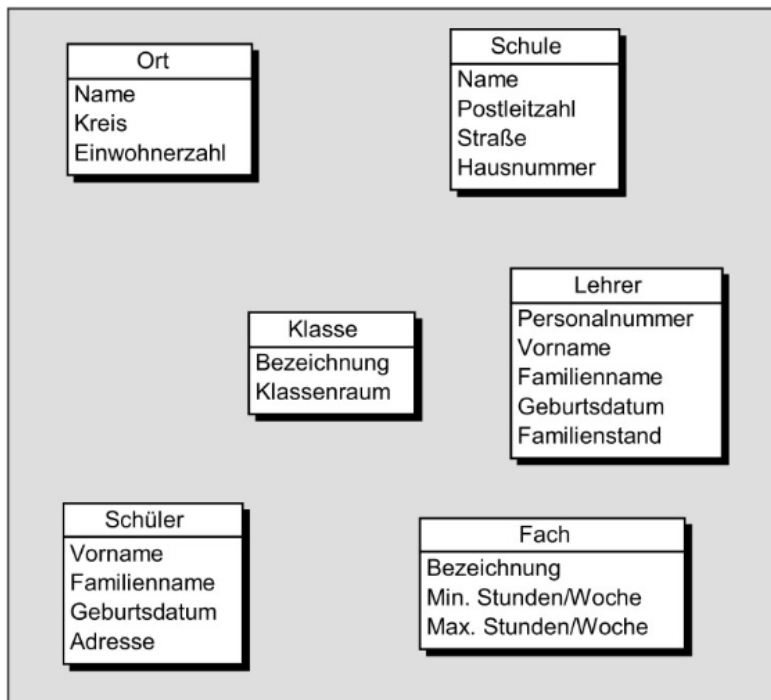
Attribut

- Eigenschaft

- Benennung eines Merkmals
- Ein relevantes Merkmal von Entitäten eines Entitätstyp

Attributwert

- Eigenschaftswert
- Spezielle Ausprägung eines Attributs für ein Objekt



2.1.4 Identifizierungsmöglichkeiten

Möglichkeiten zur Identifizierung einer Entität

- Ein einziges Attribut - Bezeichnung eines Fachs
- Kombination von Attributen - Name und Kreis eines Ortes
- Organisatorisches Attribut - Personalnummer

2.1.5 Hauptattribut

Ein Hauptattribut hat eine identifizierende Eigenschaft oder eine teil-identifizierende Eigenschaft

Es leistet einen Beitrag zur Identifizierung einer Entität und innerhalb eines Entitätstyps

2.1.6 Nebenattribut

Ein **Nebenattribut** hat eine beschreibende Eigenschaft und ist nicht notwendig zur Identifizierung

Es **leistet keinen Beitrag** zur Identifizierung einer Entität und innerhalb eines Entitätstyps

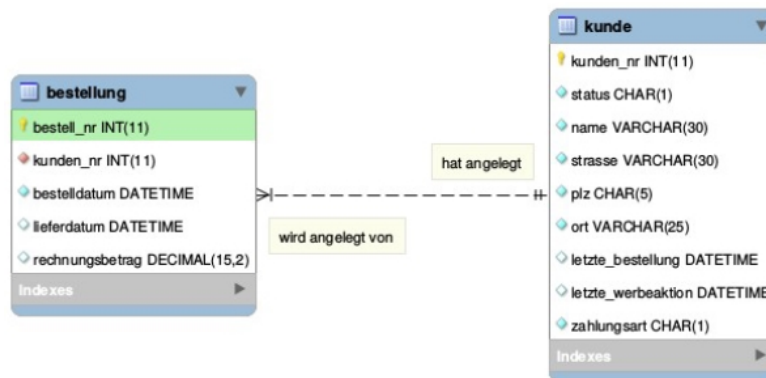
2.2 Beziehungen

Eine Beziehung ist ein konkreter Zusammenhang zwischen realen Entitäten.

An einer Beziehung können auch mehr als zwei Entitäten beteiligt sein. (Duale/ binäre Beziehung)

Ein **Beziehungstypen** ist eine sachlogischer Zusammenhang zwischen Entitäten verschiedener Entitätstypen

Der **Grad eines Beziehungstypen** ist die Anzahl der an einem Beziehungstypen beteiligten Entitätstypen



2.2.1 Multiplizität

Multiplizität $\langle A, B \rangle$ besteht aus zwei Aspekten

Optionalität: Zeichen **vor** dem Komma

Kardinalität: Zeichen **hinter** dem Komm

Optionalität

Muss jedes A mit einem B in Beziehung stehen?

Ja => **1**,? Nein => **0**,?

Kardinalität

Kann ein A mit mehreren Bs in Beziehung stehen?

















Ja => ?,**N** Nein => ?,**1**

2.2.2 Beziehungstyp Regeln

- Ein Beziehungstyp wird durch eine Linie dargestellt
- Die Benennung der Richtung von A nach B steht in der Nähe von A

- Optionalität wird durch einen Kreis gekennzeichnet
- Eine verpflichtende Verbindung wird durch einen Strich gekennzeichnet
- Kardinalität 1 wird durch einen Strich dargestellt, N durch "Krähenfüße"
- Bei > 1 können auch Minimum und Maximum angegeben werden [min,max]

2.2.3 Klassen von Beziehungstypen

$A \rightarrow B$		Kardinalität 1		Kardinalität N	
$B \downarrow A$		nicht-optional	optional	nicht-optional	optional
1	n	 1:1	 1:C	 1:N	 1:CN
	o	 C:1	 C:C	 C:N	 C:CN
N	n	 N:1	 N:C	 M:N	 M:CN
	o	 CN:1	 CN:C	 CM:N	 CM:CN

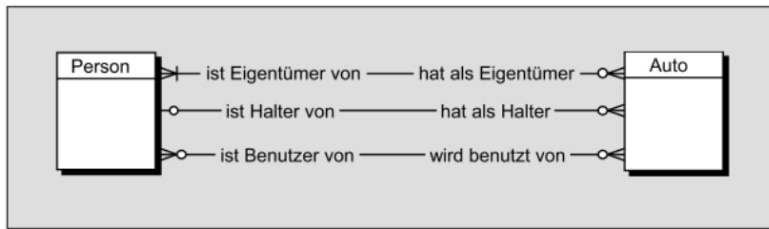
2.3 Beziehungen (weitere Konzepte)

2.3.1 Redundanz

Gründe für die Vermeidung von Redundanz:

- Mehraufwand in der Datenpflege
- Widersprüchlichkeit in den Daten möglich
- Speicherplatzverschwendung

2.3.2 Parallele Beziehungstypen



Ein **Schwacher Entitätstyp** ist:

Eigenschaften nicht ausreichend für eindeutige Identifizierung

Eine oder mehrere Beziehungstyp-Richtungen zur Identifizierung nötig

Identifizierung mittels Attribut, Beziehungstyp-Richtungen und deren Kombination

2.3.3 Schlüssel

Ein Schlüssel ist ein Attribut oder eine Beziehungstyprichtung zur eindeutigen Identifizierung

Primärschlüssel Ein eindeutiger Schlüssel (Personalnummer)