

Datenbanken

`github/toiletcoders`

Inhaltsverzeichnis

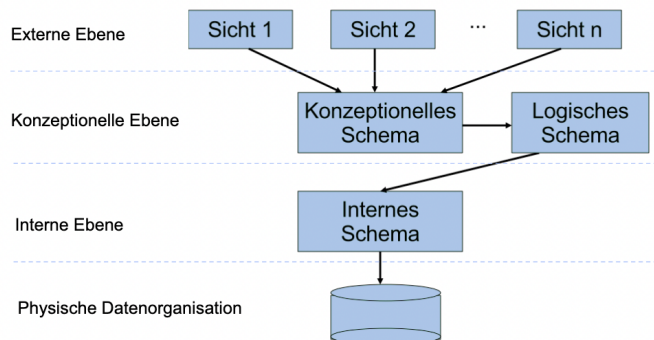
	Seite
1 Einführung	1
1.1 Grundlagen	1
2 Konzeptionelles Modell	5
2.1 Entitäten und Entitätstypen	5
2.2 Beziehungen	7
2.3 Beziehungen (weitere Konzepte)	8
3 Datenbankmodelle	11
3.1 Datenbankmodell	11
3.2 Relationales Datenmodell	11
3.3 NoSQL (andere Datenmodelle)	12
3.4 Datentypen	12
4 Logisches (relationales) Modell	14
4.1 Überblick Transformation	14
4.2 Duale und rekursive Beziehungstypen	14
5 Normalformen	15
5.1 Überblick Normalisierung	15
5.2 Erste Normalform	15
5.3 Zweite Normalform	16
5.4 Dritte Normalform	17
5.5 Weitere Normalformen	18
6 Structured Query Language (SQL)	19
6.1 Einführung in SQL	19
6.2 Datendefinition (DDL)	20
6.3 Datenmodifikation (DML)	21
6.4 Aggregatfunktionen	22
7 Erweiterte Abfragen mit SQL	23
7.1 Unterabfragen	23
7.2 Verbundabfrage (JOINS)	23
7.3 Outer JOINS	24
7.4 Mengenoperationen	25

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 Sichten

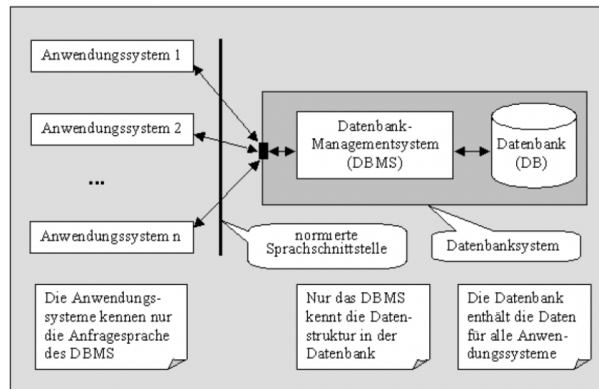
1.1.3 Datenbank- vs. dateibasierte Anwendungssysteme

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
- Sachlogische Zusammenhänge von Entitäten

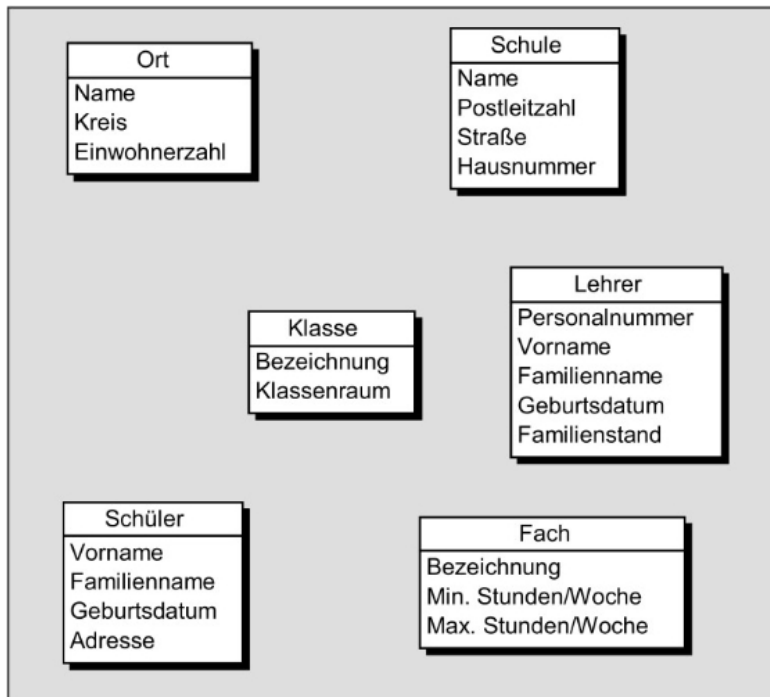
2.1.3 Attribut

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

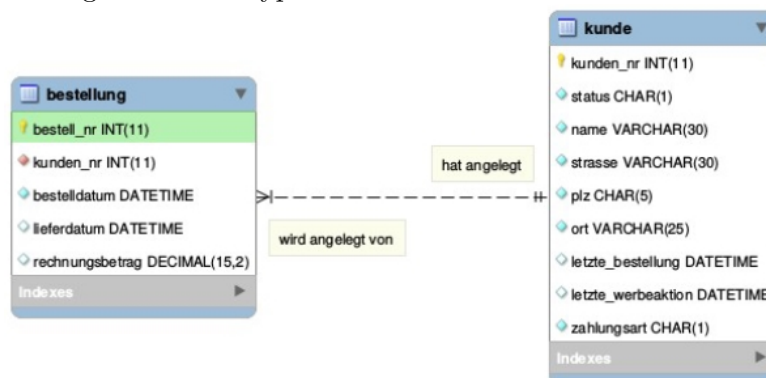
2.2 Beziehungen

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 ein 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 Komma

Optionalität

Muss jedes A mit einem B in Beziehung stehen?

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

Kardinalität

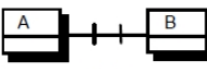
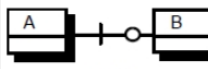
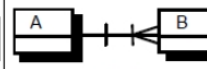
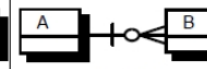
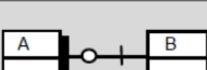
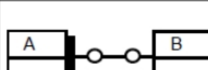
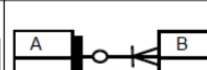
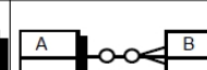
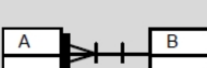
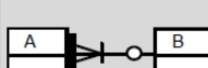
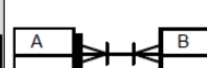





Kann ein A mit mehreren Bs in Beziehung stehen?

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 → B		Kardinalität 1		Kardinalität N	
B ↓ 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

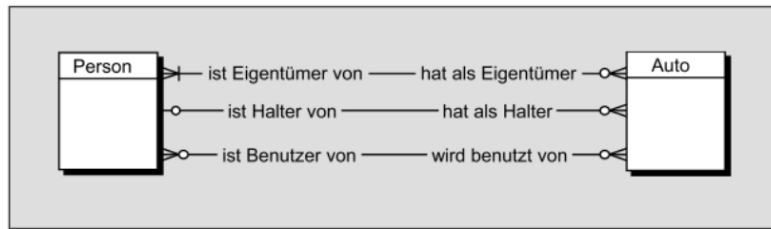
2.3 Beziehungen (weitere Konzepte)

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

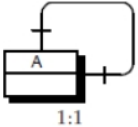
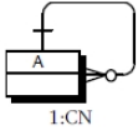
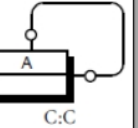
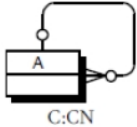
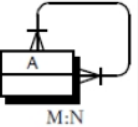
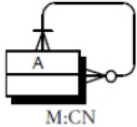
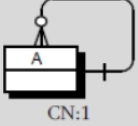
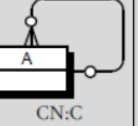
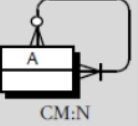
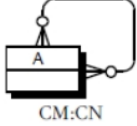
Primärschlüssel Ein eindeutiger Schlüssel (Personalnummer)

Zusammengesetzte Schlüssel mehrere teil-identifizierende Elemente (Attribut und Beziehungstyprichtung)

Teilschlüssel echte Teilmenge der teil-identifizierenden Elemente

2.3.4 Rekursiv-Beziehungstypen

Entitäten gehören dem selben Entitätstypen an

1. Beziehungstyp-Richtung →		Kardinalität 1		Kardinalität N	
2. Beziehungstyp-Richtung ↓		nicht-optional	optional	nicht-optional	optional
Kardinalität 1	nicht-optional	 1:1			 1:CN
	optional		 C:C		 C:CN
Kardinalität N	nicht-optional			 M:N	 M:CN
	optional	 CN:1	 CN:C	 CM:N	 CM:CN

2.3.5 Eigenschaften von Beziehungstypen

Generelle Regel:

Eigenschaften des Beziehungstyps werden mithilfe eines zusätzlichen Entitätstyps modelliert

3 Datenbankmodelle

3.1 Datenbankmodell

logisches Beziehungsgebilde,
beschreibt die Art und Weise der Verbindung von Datensätzen

Traditionelle Datenbankmodelle

- Hierarchisch: nur einen Elternknoten
- Netzwerk
- Relational (RDBM)
- Objektorientiert und objektrational

”Neue” Datenbankmodelle (seit 2005):
Spaltenbasiert, Dokumentorientiert

3.2 Relationales Datenmodell

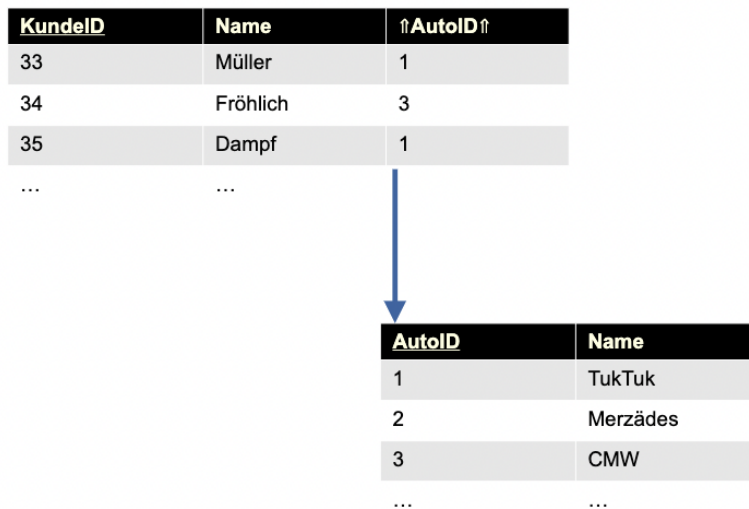
Theorie:

- streng mathematisch
- Entitätstypen werden als Relationen gespeichert
- Beziehungen zwischen Entitäten werden über Referenzattribute in Tabellenspalten gespeichert
- **Relation:** Teilmenge der Produktmenge der Wertebereiche der Attribute

3.2.1 Schlüssel im RDMBS

Ein Schlüssel ist jede identifizierende Attributmenge, die minimal ist.
Primärschlüssel ist ein Schlüssel zur Identifizierung einzelner Tupel

Fremdschlüssel ist eine Attributmenge die einen anderen Primärschlüssel referenziert



3.3 NoSQL (andere Datenmodelle)

NoSQL Eigenschaften

- Datenmodell ist nicht-Relational
- keine Relationen
- Sammelbegriff für nicht-relationale Datenmodelle
- NoSQL Modell ist oft frei von einem mathematischen Schema

3.4 Datentypen

Jedem Attribut wird ein Datentyp zugewiesen

3.4.1 Numerische Datentypen

Ablage von numerischen Werten (Vorzeichenbehaftet - Signed, sonst Unsigned)

3.4.2 Ganzzahlige Datentypen

BOOLEAN (1 Byte) true/false

TINYINT (1 Byte) sehr kleine Werte (-128 bis 127) (Unsigned 0 bis 255)

SMALLINT (2 Byte)

MEDIUMINT (3 Byte)

INT (4 Byte)

BIGINT (8 Byte)

3.4.3 Kommazahlen

Festkommazahl hat immer den gleichen Exponenten (Preise, Ausmaße)

Fließkommazahl hat wechselnde Exponenten

3.4.4 Fließkommazahlen

FLOAT (4 Byte) einfache Genauigkeit

DOUBLE (8 Byte) doppelte Genauigkeit

3.4.5 Festpunktzahlen

DECIMAL Angabe von exakten Zahlen, Festlegung der Länge und Nachkommastellen

3.4.6 Datums-Datentyp

Speichern Datums- und Zeitwerte, (DATETIME, DATE, TIMESTAMP, YEAR, TIME)

3.4.7 String Datentypen

CHAR(N) Ablage von Zeichenketten der festen Länge N

VARCHAR(N) Ablage von Zeichenketten mit variabler Länge bis zu N

CHAR benötigt 1 Byte pro Zeichen

VARCHAR benötigt 1 Byte zusätzlich pro Spalte

4 Logisches (relationales) Modell

4.1 Überblick Transformation

4.1.1 T1: Regel für Entitätstypen

Jeder Entitätstyp wird im RDBM zu einer Tabelle
Attribute werden zu Spalten

4.1.2 T1: regel für Beziehungstypen

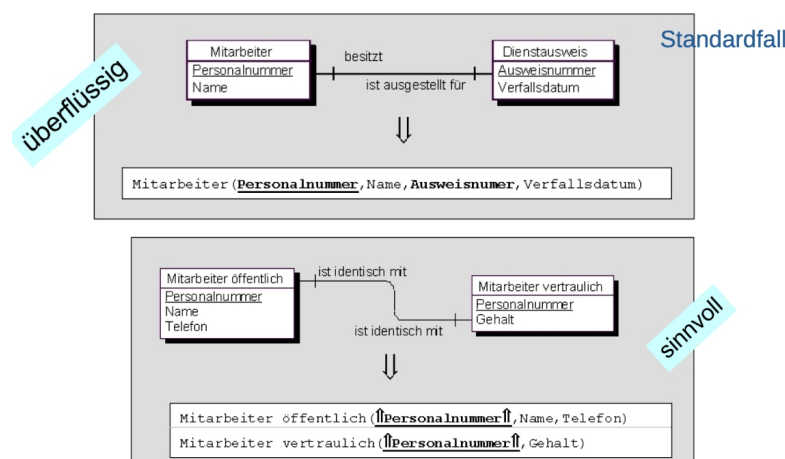
Beziehungstypen werden über **Fremdschlüssel** modelliert
Fremdschlüssel werden mit Pfeilen dargestellt.
Zusammengesetzte Schlüssel werden mit "+" dargestellt.
Primärschlüssel werden unterstrichen

4.1.3 Regeln

UNIQUE: Attributwert darf nur einmal in der Tabelle vorkommen
NOT NULL: Attributwert darf nicht ausgelassen werden

4.2 Duale und rekursive Beziehungstypen

4.2.1 1:1 Beziehungstypen

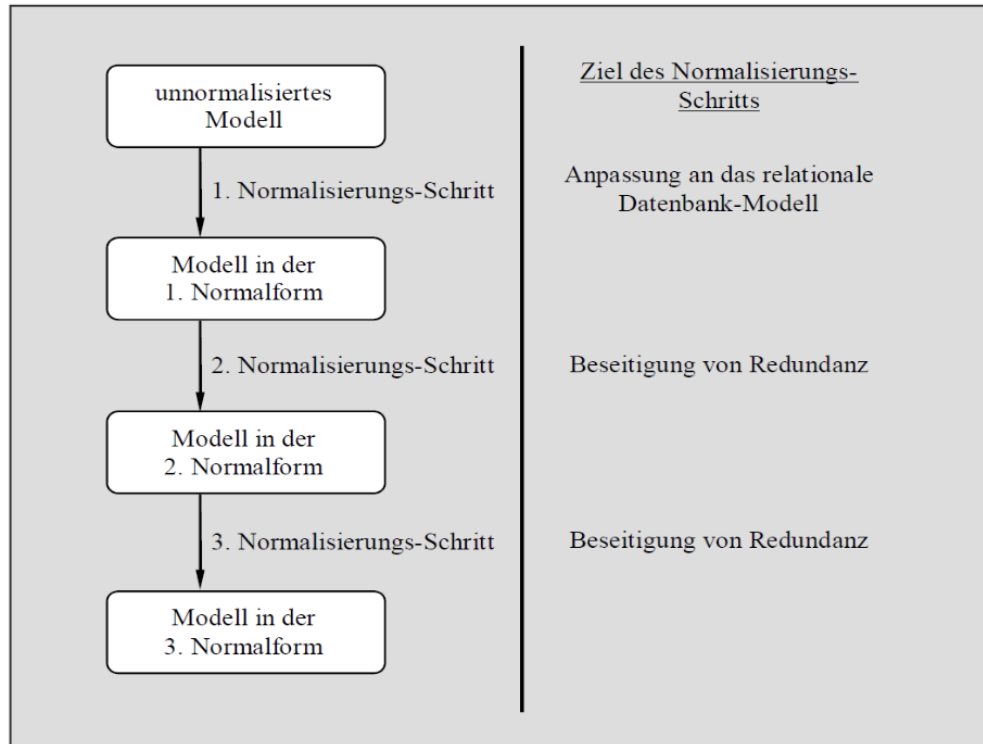


... (Please read this in Script)

5 Normalformen

5.1 Überblick Normalisierung

Gesucht sind objektive Qualitätskriterien zur Erstellung von Modellen



5.2 Erste Normalform

5.2.1 Komplexe Eigenschaften

Mehrere Werte gleichzeitig für ein Attribut einer Entität
Mehrwertiges Attribut (Telefonnummer)
Zusammengesetztes Attribut

5.2.2 1. Normalform

Überführen der Unnormalisierten Form in die 1NF
Bei mehrwertigen Attribut → Einführung von eigenen Attributen

Bei multiplen Eigenschaften → Einführung von "stark redundanten" Tupeln
Anomalien durch Redundanz

- Einfügeanomalie: Neuer Lieferant nicht ohne mind. einen Artikel einfügbar
- Modifikationsanomalie: Änderung der Artikelbezeichnung an mehreren Stellen
- Löschanomalie: Löschen von Artikeln führt zum Löschen der Lieferanten

5.3 Zweite Normalform

5.3.1 Funktionale Abhängigkeit

$A \rightarrow B$ ist funktional abhängig von A, A bestimmt B
Kundennummer → Vorname, Nachname

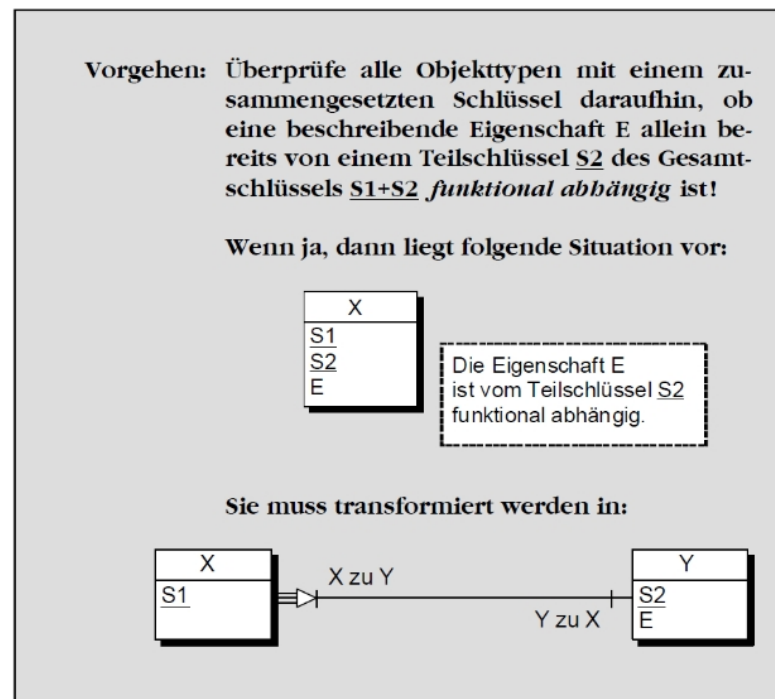
5.3.2 2NF

Eine Relation ist in zweiter Normalform wenn sie in 1NF ist
und jedes nicht-Schlüsselattribut vom Schlüssel voll funktional abhängig ist.

Merkregel:

Hat ein Entitätstyp einen nicht-zusammengesetzten Schlüssel (nur ein Attribut als PK),
liegt 2NF automatisch vor.

5.3.3 Transformierung



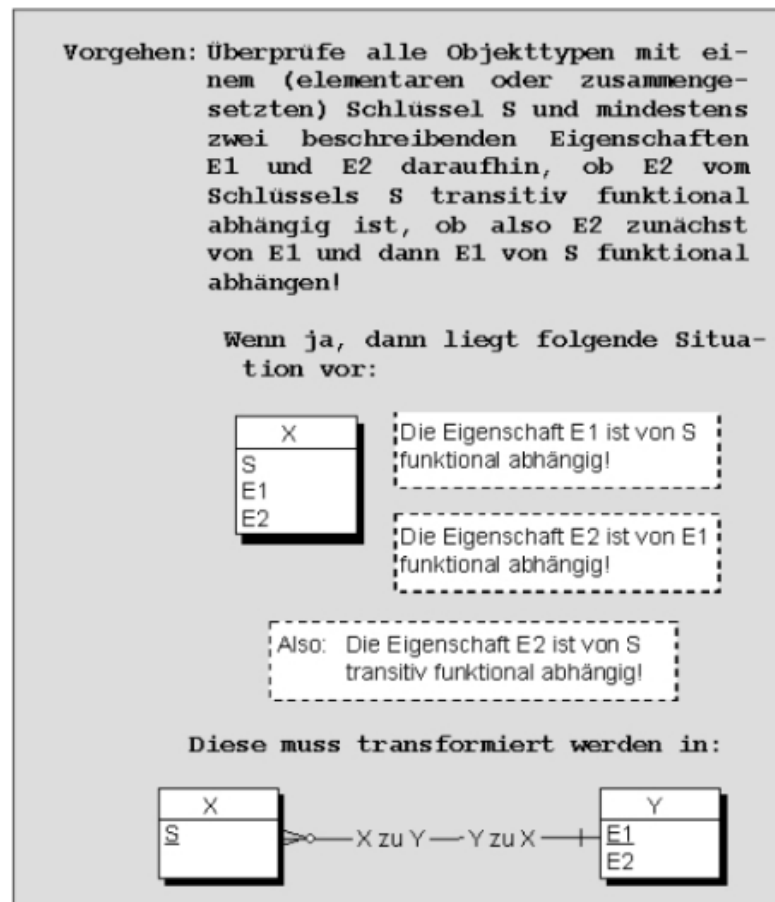
5.4 Dritte Normalform

5.4.1 Transitive Abhängigkeit

C ist transitiv abhängig von A ($A \rightarrow B, B \rightarrow C$), wenn

- Wert von A ist bekannt
- Daraus lässt sich Wert von B bestimmen
- Aus Wert B lässt sich C bestimmen

5.4.2 3NF



5.5 Weitere Normalformen

Boyce-Codd Normalform (BCNF)

- Relation in 3NF und
Jeder Determinant ist ein Schlüsselkandidat. Ein Determinant ist eine Attributmenge, von der ein anderes Attribut vollständig funktional abhängig ist.
- Verletzung BCNF selten; setzt ebenso compound PK voraus
- „strenger“ als 3NF

4. Normalform

- In BCNF und
- Keine „mehrwertigen“ nicht-trivialen Abhängigkeiten im PK

5. Normalform

- In 4NF und gar keine „mehrwertigen“ Abhängigkeiten

6 Structured Query Language (SQL)

6.1 Einführung in SQL

6.1.1 Eigenschaften von SQL

- Kann von verschiedenen Benutzern verwendet werden
- SQL ist intuitiv
- SQL ist eine deklarative Sprache
- EBNF-Notation

6.2 Datendefinition (DDL)

6.2.1 Anlegen von Tabellen

Syntax CREATE TABLE

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name (  
  col_name data_type [NOT NULL | NULL]  
  [DEFAULT default_value] [AUTO_INCREMENT], ...,  
  [PRIMARY KEY (col_name)]  
);
```

- Pflicht-Elemente
 - Tabellenname, Spaltenname und Datentyp
- Optionale Angabe
 - NOT NULL → Eingabepflicht
 - DEFAULT → Standardwert falls ohne Wert beim Einfügen
 - AUTO_INCREMENT → Autom. Vergabe von PK-Werten
 - PRIMARY KEY → Primärschlüssel Spalte(n)

Syntax CREATE TABLE mit Fremdschlüssel

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name (  
  col_name data_type [NOT NULL | NULL]  
  [DEFAULT default_value] [AUTO_INCREMENT], ...,  
  [PRIMARY KEY (col_name)],  
  [CONSTRAINT [fk_name]] FOREIGN KEY (col_name)  
    REFERENCES src_tbl_name(src_col_name)  
    ON DELETE reference option  
    ON UPDATE reference option  
);
```

- FOREIGN KEY → Spalte mit Fremdschlüsselwert
- REFERENCES → Tabelle (Spalte) des referenzierten Primärschlüssels
- ON DELETE → Maßnahmen bei Löschen des ref. Primärschlüssels
- ON UPDATE → Maßnahmen bei Ändern des ref. Primärschlüssels

Reference Option

RESTRICT | CASCADE | SET NULL | NO ACTION

- RESTRICT → Operation verbieten
- CASCADE → Operation für Fremdschlüssel auch durchführen
- SET NULL → Fremdschlüsselwert Löschen
- NO ACTION → Keine Anpassung

CREATE LIKE → erstellt die neue Tabelle und kopiert die Datenstruktur

6.2.2 Ändern von Tabellen

- CHANGE → CHANGE < Spaltenname > < Spaltenname_{neu} >
- ADD → ADD < Spaltenname > < Datentyp > [< Eigenschaft >]
- DROP → DROP COLUMN < Spaltenname >
- ADD PRIMARY KEY → ADD PRIMARY KEY < Keyname > (< Spaltenname >)
- ADD FOREIGN KEY
- DROP FOREIGN KEY
- RENAME TABLE → < Tabellenname > TO < Tabellenname_{neu} >

Tipps:

Bei CHANGE and Datentypen können Daten verändert werden!

6.3 Datenmodifikation (DML)

6.3.1 Datenmanipulation DDL

Dient zum Schreiben, Lesen, Ändern und Löschen von Daten
Daten lesen mit SELECT und Form

6.3.2 Einfügen mit INSERT

INSERT INTO < Tabellenname > [(Spaltenliste)] **VALUES**(Werteliste);

- Beispiel

- IdArtikel ist AUTO_INCREMENT



- **INSERT INTO** Artikel2NF (Bezeichnung , Sparte)
VALUES ('Klingenfix Rasierer', 'Kosmetik');

- **INSERT INTO** Artikel2NF **VALUES**
(NULL , 'Wattebäuschchen', 'Kosmetik ');

- Mehrere Zeilen auf einmal auch möglich

- **INSERT INTO** SchuelerIn **VALUES**
(1, 'Oster ', 'Hase ', 'm'),
(2, 'Harry ', 'Hurtig ', 'm'),
(3, 'Horst ', 'Schlaemmer ', 'm'),
(4, 'Evje ', 'van Dampen ', 'w');

6.3.3 Lesen mit SELECT

SELECT[*DISTINCT*|*ALL*]
*|[*SpaltenAusdruck*[*ASneuerName*]][, ...]
*FROM**Tabelle*[*ALIAS*][, ...]
[*WHERE**Bedingung*]
[*GROUPBY**SpaltenListe*]
[*HAVING**Bedingung*]
[*ORDERBY**SpaltenListe*]
DISTINCT eliminiert Duplikate

6.3.4 Lesen mit SELECT und DISTINCT

DISTINCT eliminiert Duplikate
SELECT *DISTINCT* *Beispielspalte* FROM *Beispieltabelle*;

6.3.5 Lesen mit SELECT und AS

AS benennt die neuberechnete Spalte Umprogrammieren

6.3.6 Lesen mit SELECT und WHERE

WHERE filtert Zeilen
BETWEEN lässt Werte zwischen zwei Werten finden

6.4 Aggregatfunktionen

6.4.1 COUNT

Anzahl der Einträge zählen
*SELECT**COUNT*(*)*ASExpensive*
*FROM**Kassenbon1NF*
*WHERE**Preis* > 100;

6.4.2 COUNT DISTINCT

DISTINCT eliminiert wieder die Duplikate

6.4.3 SUM

SUM(*Preis*) zählt alle Einträge in *Preis* zusammen

7 Erweiterte Abfragen mit SQL

7.1 Unterabfragen

SQL Statements können eingebettete SELECTs enthalten

```
SELECT *  
FROM Mitarbeiter  
WHERE idBuero = ( SELECT idBuero  
                  FROM Buero  
                  WHERE Ort = 'London');
```

7.1.1 Regeln für Unterabfragen

- ORDER BY Klauseln
 nicht in Unterabfragen
 nur in äußerstem SELECT-Statement
- Spaltenliste der Unterabfrage
 nur **ein einziger** Spaltenname als Ergebnis
 Ausnahme: EXISTS
- Bei Vergleichen: Unterabfrage steht **rechts**

7.2 Verbundabfrage (JOINS)

7.2.1 Einfacher JOIN

- Kunden und Mitarbeiter
 SELECT Name, Nachname, Vorname
 FROM Kunde, Mitarbeiter
 WHERE Kunde.idMitarbeiter = Mitarbeiter.idMitarbeiter;
- Vorgehen
 Nur Kunden mit zugeordnetem Mitarbeiter erscheinen im Ergebnis
 Kunden ohne Mitarbeiter (NULL) erscheinen nicht
 Verbund hier über Fremdschlüssel
- Alternativen

- Verwendung von Tabellen-Alias
SELECT k.Name, m.Nachname, m.Vorname
FROM Kunde k, Mitarbeiter m
WHERE k.idMitarbeiter = m.idMitarbeiter;
- Alternative mit Schlüsselwörtern JOIN und ON
SELECT k.Name, m.Nachname, m.Vorname
FROM Kunde k **JOIN** Mitarbeiter m
ON k.idMitarbeiter = m.idMitarbeiter;

7.2.2 NATURAL JOIN

”Natürlicher” JOIN

- Verwendung aller gemeinsamen Attribute
- Vergleich auf Gleichheit
- Zumeist: Abgleich von Fremd- und Primärschlüsselattributen

wegen Häufigkeit eigenes Schlüsselwort: NATURAL JOIN

```
SELECT k.Name,b.Bestelldatum
FROM Kunde k NATURAL JOIN Bestellung b;
```

7.2.3 VIEWS: Virtuelle Tabellen

Virtuelle Tabellen

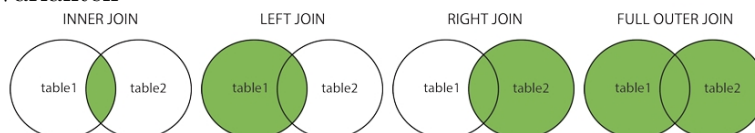
Anlegen:

```
CREATE VIEW Viewname AS (SELECT Statement);
```

Views können wie normale Tabellen verwendet werden

7.3 Outer JOINS

- Innerer JOIN:
 Zeile ohne Entsprechung im ON fehlt komplett im Ergebnis
 Anstatt JOIN kann auch INNER JOIN geschrieben werden
- Äußere Joins:
 Zeile mit Entsprechung im ON erscheint
 Zeile ohne Entsprechung in der anderen Tabelle erscheint auch
 Fehlende Werte sind NULL
 Nicht mit CROSS JOIN verwechseln
- Varianten



7.3.1 Existenzabfrage: EXISTS

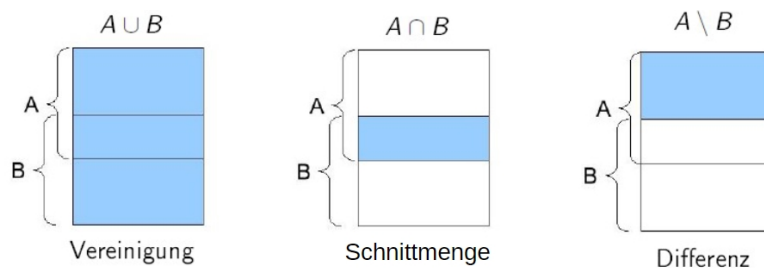
wahr, wenn Unterabfrage nicht leeres Ergebnis Lieferant

falsch, wenn kein Treffer in Unterabfrage

Beispiel:

```
SELECT * FROM Mitarbeiter m
WHERE EXISTS (
    SELECT * FROM Kunde k
    WHERE m.idMitarbeiter = k.idMitarbeiter
    AND Ort = 'Boston');
```

7.4 Mengenoperationen



- **UNION**
Vereinigungsmenge $A \cup B$
bei Tabellen: alle Datensätze aus A und B aneinanderhängen
- **INTERSECT**
Schnittmenge $A \cap B$
bei Tabellen: alle Datensätze, die in A und B vorkommen
- **EXCEPT**
Schnittmenge $A \setminus B$
bei Tabellen: alle Datensätze aus A, die aber nicht in B vorkommen