Datenbanken Zusammenfassung

Peter Minor
Sommersemester 2025

16. Juli 2025

Inhaltsverzeichnis

1 Kapitel 1: Einführung

Sehr viel Geyappe über Datenbanken und Entwurfsmodelle, später.

2 Kapitel 2: Datenbank-Modellierung

2.1 Modell

Ein Modell ist ein abstrahiertes Abbild der Realität. Es hilft beim Verständnis, bei der Kommunikation und Simulation komplexer Sachverhalte. In der Datenbankmodellierung wird zwischen konzeptuellen, logischen und physischen Modellen unterschieden.

Entity-Relationship-Modell (ER)

2.2 Entitätstyp

Ein Entitätstyp (auch Objekttyp) ist eine Klasse gleichartiger Objekte. Darstellung im ER-Diagramm: Rechteck.

2.3 Attribut

Ein Attribut beschreibt eine Eigenschaft eines Entitätstyps. Darstellung: Ellipse. Attribute können einfach, zusammengesetzt, mehrwertig oder berechnet sein.

2.4 Beziehungstyp (Relationship)

Ein Beziehungstyp stellt eine Relation zwischen Entitäten dar. Darstellung: Raute. Die Kardinalität (1:1, 1:n, m:n) beschreibt die Anzahl möglicher Zuordnungen.

2.5 Schlüssel

Ein Schlüssel ist ein Attribut (oder eine Attributkombination), das jede Entität eindeutig identifiziert. Starke Entitäten haben eigene Schlüssel; schwache Entitäten benötigen eine identifizierende Beziehung zu einer starken Entität.

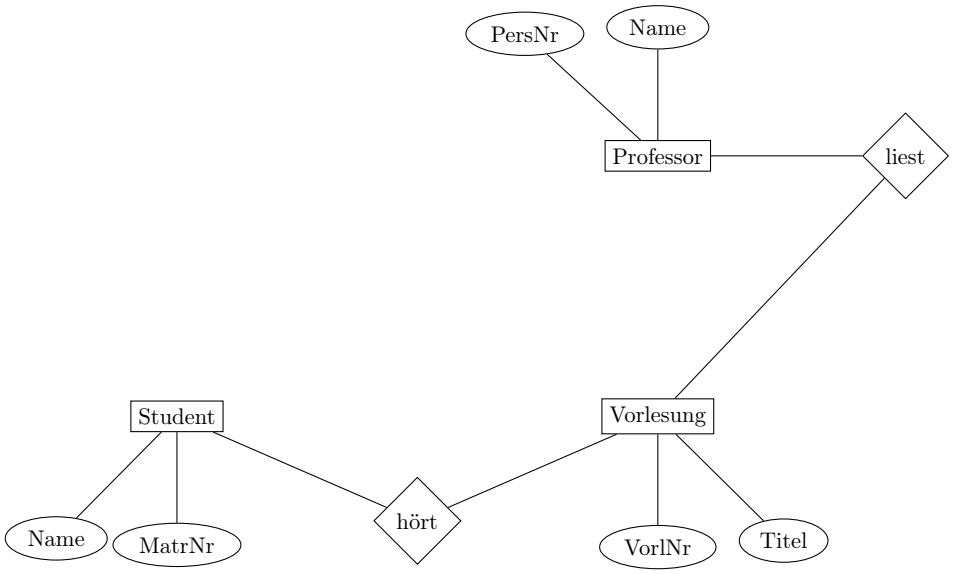
2.6 Partizipation

Beschreibt, ob eine Entität zwingend an einer Beziehung teilnehmen muss:

- **totale Partizipation**: jede Entität muss beteiligt sein
- **partielle Partizipation**: Beteiligung ist optional

2.7 Spezialisierung & Generalisierung (EER)

- **Spezialisierung**: Zerlegung eines Supertyps in Subtypen
- **Generalisierung**: Vereinigung ähnlicher Entitätstypen zu einem Supertyp



3 Kapitel 3: Das relationale Datenmodell

3.1 Relation

Eine Relation ist eine Tabelle mit Attributen (Spalten) und Tupeln (Zeilen). Sie basiert auf dem mathematischen Konzept einer Menge von Tupeln.

3.2 Primärschlüssel

Ein Attribut oder Attributkombination, die ein Tupel eindeutig identifiziert.

3.3 Fremdschlüssel

Ein Attribut, das auf den Primärschlüssel einer anderen Relation verweist und referentielle Integrität sicherstellt.

Relationale Algebra

3.4 Selektion (σ)

Filtert Tupel, die eine bestimmte Bedingung erfüllen. Beispiel:

$$\sigma_{Note \geq 4}(\text{Pruefungen})$$

3.5 Projektion (π)

Reduziert die Anzahl der Attribute. Beispiel:

$$\pi_{Name, MatrNr}(\text{Studierende})$$

3.6 Vereinigung $\cup$ Schnitt $\cap$ Differenz $-$

Klassische Mengenoperationen für Relationen mit gleichem Schema.

3.7 Kartesisches Produkt ($\times$)

Kombiniert zwei Relationen durch paarweise Tupelkombination.

3.8 Join ($\bowtie$)

Verknüpft zwei Relationen über gemeinsame Attribute. Spezialformen:

- natürlicher Join
- theta-Join
- equi-Join

3.9 Umbenennung (ρ)

Benennung einer Relation oder ihrer Attribute neu, z.B. zur besseren Lesbarkeit von Ausdrücken.

Beispielhafte Relationen:

- **Student**(MatrNr, Name)
- **Professor**(PersNr, Name)
- **Vorlesung**(VorlNr, Titel)
- **hört**(MatrNr, VorlNr)
  Fremdschlüssel: MatrNr → Student, VorlNr → Vorlesung
- **liest**(PersNr, VorlNr)
  Fremdschlüssel: PersNr → Professor, VorlNr → Vorlesung

4 Kapitel 4: Relationale Entwurfstheorie

4.1 Funktionale Abhängigkeit

Eine Attributmenge α bestimmt eine andere Attributmenge β, geschrieben als:

$$\alpha \to \beta$$

gilt genau dann, wenn für alle Tupel t_1, t_2 gilt: $t_1[\alpha] = t_2[\alpha] \Rightarrow t_1[\beta] = t_2[\beta]$

4.2 Schlüssel und Superschlüssel

- **Superschlüssel:** α ist Superschlüssel, wenn $\alpha \to R$
- **Kandidatenschlüssel:** Minimaler Superschlüssel

4.3 Ziel der Normalisierung

Die Normalisierung dient dazu, Redundanzen zu vermeiden und Anomalien (Einfüge-, Update-, Löschanomalien) zu verhindern. Dazu wird ein Relationenschema anhand funktionaler Abhängigkeiten in wohldefinierte Formen überführt.

4.4 Überblick über die Normalformen

- **1NF (erste Normalform):** Alle Attributwerte sind atomar (nicht weiter teilbar).
- **2NF (zweite Normalform):** 1NF erfüllt + jedes Nicht-Schlüsselattribut ist voll funktional abhängig vom gesamten Primärschlüssel.
- **3NF (dritte Normalform):** 2NF erfüllt + keine transitiven Abhängigkeiten von Nicht-Schlüsselattributen.
- **BCNF (Boyce-Codd Normalform):** Für jede nicht-triviale funktionale Abhängigkeit $\alpha \to \beta$ gilt: α ist ein Superschlüssel.

4.5 Vorgehen zur Normalisierung

1. Ermittele alle funktionalen Abhängigkeiten (FDs).
2. Bestimme alle Schlüsselkandidaten.
3. Prüfe die aktuelle Normalform.
4. Zerlege die Relation bei Verstoß in mehrere Relationen:
    - Zerlege so, dass jede FD in einer Relation vollständig erfüllt wird.
    - Erhalte dabei die Verlustfreiheit und Abhängigkeitserhaltung.

4.6 Beispiel: Normalisierung auf 3NF

Gegeben sei folgende Relation:

$$R(\underline{MatrNr}, Name, Studiengang, Fakultaet)$$

mit den funktionalen Abhängigkeiten:

F1: MatrNr → Name, Studiengang, Fakultaet
F2: Studiengang → Fakultaet

Analyse:

- F1: MatrNr ist ein Schlüsselkandidat.
- F2: transitive Abhängigkeit: MatrNr → Studiengang → Fakultaet
- ⇒ Verstoß gegen 3NF.

Zerlegung in 3NF:

- $R_1(\underline{MatrNr}, Name, Studiengang)$
- $R_2(\underline{Studiengang}, Fakultaet)$

Ergebnis: Beide Relationen sind in 3NF, keine Redundanz, keine Anomalien.

4.7 Anomalien

Anomalien treten auf, wenn Relationen schlecht strukturiert sind – meist durch Redundanz und fehlende Trennung von unabhängigen Daten. Es gibt drei Hauptarten:

- **Einfügeanomalie:** Daten können nicht eingefügt werden, ohne andere zu erzeugen
- **Updateanomalie:** Inkonsistenz bei mehrfacher Speicherung derselben Information
- **Löschanomalie:** Verlust nützlicher Informationen durch Löschung eines Tupels

5 Kapitel 5: SQL - Structured Query Language

Wir befinden uns in der Datenbank-Installation, also im Physischen Schemaentwurf.

5.1 Historie

- 1974: SEQUEL von IBM, Implementierung für System R
- 1983: SQL ist der Standard geworden
- 1986: SQL-86, bzw. SQL 1 $\Rightarrow$ erster ANSI und ISO-Standard
- 1992: SQL 2, deutliche Erweiterungen im Standard
- Weitere Revisionen:2000(SQL 3), 2003, 2006, 2008, 2011, 2016, 2023

SQL dient als verschiedene Sprachen:

- VDL, DDL, SDL zur Definition von Datenbanken
- DML(Datenmanipulationssprache), DCL(Datenkontrollsprache) zum Zugriff auf Datenbanken

SQL-Befehle:

Befehl	**Beschreibung**
SQL als DDL(Datendefinition)	
`CREATE SCHEMA`	Erstellt ein neues Schema in der Datenbank.
Beispiel	`create schema Unternehmen authorization JSmith create table Projekt;`
Einfacher:	`PID int not null primary key,` geht aber leider nicht mit zusammengesetzten Schlüsseln.
`CREATE Table`	Erstellt eine neue Tabelle im Schema.
Beispiel	`create table Projekt ( PID int not null, Name varchar(50) not null, primary key(PID));`
`ALTER Table`	ändert die angegebene Tabelle.
Es gibt noch andere Verwendungen für `alter:`	
`alter database`	ändert Eigenschaften der Datenbank.
`alter view`	ändert die Definition einer Sicht
`alter index`	modifiziert einen Index
`alter user/role`	ändert die Rollen eines Benutzers
`Add`	Fügt eine Spalte zu einer Tabelle hinzu
Beispiel	`alter table Angestellte add foreign key (Abt) references Abteilung(Nummer);`
`drop`	Löscht das angegebene Objekt. Kann auf Schemen, Tabellen, Sichten, Constraints und Spalten angewendet werden.
Beispiel	`drop table Arbeitszeiten;`
`rename`	Ändert den Namen einer Tabelle
SQL als DML(Datenmanipulation und -abfrage)	
`select [..] from`	Wählt die gegebenen Spalten aus der Tabelle aus und gibt sie zurück

SQL-Keywords:

Keyword	Beschreibung
SQL als DDL(Datendefinition)	
`not null`	Attribut darf nicht leer sein.
`primary key`	Attribut ist Primärschlüssel der Tabelle.
`unique`	Attributwerte müssen eindeutig sein.
`check`	Ermöglicht komplexere Einschränkungen
`cascade`	?
`set null`	Setzt die Referenz auf null
`set default`	Setzt die Referenz auf den Default-Wert
`No Action/Restrict`	?
Beispiel	Constraints beispiel?
`foreign key`	Attribut verweist auf Primärschlüssel einer anderen Tabelle.
`references`	Definiert die referenzierte Tabelle und Spalte für den Fremdschlüssel.
Beispiel:	`foreign key (PID) references Projekt(PID)`
`to_number` oder `to_char`	Konvertiert Datentypen, z.B. von String zu Zahl oder umgekehrt.
Date, Time, Datetimeoffset, interval, year, day, second?	
`where`	filtert nach Bedingungen
Beispiel	`select * from Klausur where Note <= 4;`
`and`	Verknüpft Bedingungen, alle müssen erfüllt sein
`or`	Verknüpft Bedingungen, mindestens eine muss erfüllt sein
`order by`	Sortiert die Ergebnisse nach den angegebenen Spalten
`Asc` bzw. `desc`	Sortiert aufsteigend bzw. absteigend, Asc ist der Standardwert
Beispiel	`select * from Klausur order by Note desc;`
`group by`	Gruppiert die Ergebnisse nach den angegebenen Spalten
Beispiel	`select * from Belegung group by KursID;`
`distinct`	Entfernt doppelte Einträge aus dem Ergebnis Aber ist teuer und braucht man nicht unbedingt.
Beispiel	`select distinct Alter from Student;`
`as` Beispiel	Benennt die Spalte um `select Name as StudentName from Student;`
`count`	Zählt die Anzahl der Tupel
`sum`	Summe der Werte der Tupelattribute
`min`	kleinstes Tupelattribut
`max`	größtes Tupelattribut
`avg`	durschschnittlicher Wert der Tupelattribute
Beispiel	`select max(Gehalt) from Angestellte;`
Wie interagieren die mit group by?	

5.2 SQL als DDL

- Schema, Tabellen, Datentypen, Constraints definieren
- Strukturelle Änderungen mittels `drop`, `alter`
- SCHEMA:
  - Namensraum in DB
  - Hat eindeutigen Namen
  - Hat Autorisierungsbezeichner
  - Beschreibt jedes im Schema enthaltene Objekt
    - Relationen
    - Wertebereiche
    - Restriktionen
    - Sichten
    - Zugriffsrechte
- `information_schema` enthält Metadaten über die Datenbank

5.3 Übergang von relationelem Schema zu SQL Schema

- Name der Relation wird zum Tabellennamen
- Attribute werten untereinander geschrieben(Datentypen angeben)
- Bei einem Schlüssel `primary key` hinterschreiben
- Bei zusammengesetzten Schlüsseln `primary key (A, B)` angeben
- Für IDs ist `serial` als Datentyp sinnvoll
- Fremdschlüssel werden mit `foreign key` gekennzeichnet

Beispiel:

- **Student**(Matrikelnummer, Name, Studiengang)
- **Kurs**(KursID, Titel, Dozent)
- **Belegung**(Matrikelnummer, KursID, Note)

Wird folgendes SQL-Schema:

```
-- Tabelle: Student
CREATE TABLE Student (
    Matrikelnummer INT PRIMARY KEY,
    Name VARCHAR(100),
    Studiengang VARCHAR(100)
);
```

```
-- Tabelle: Kurs
CREATE TABLE Kurs (
    KursID SERIAL PRIMARY KEY,
    Titel VARCHAR(100),
    Dozent VARCHAR(100)
);

-- Tabelle: Belegung
CREATE TABLE Belegung (
    Matrikelnummer INT,
    KursID INT,
    Note DECIMAL(3,1),
    PRIMARY KEY (Matrikelnummer, KursID),
    FOREIGN KEY (Matrikelnummer) REFERENCES Student(Matrikelnummer),
    FOREIGN KEY (KursID) REFERENCES Kurs(KursID)
);
```

5.4 SQL als DML

- Daten manipulieren und abfragen
- Es können Duplikate auftreten, falls nicht gewünscht `distinct` nutzen

5.5 Umsetzung der Operationen der relationalen Algebra in SQL

Operation	SQL-Äquivalent
Kartesisches Produkt	`select * from A, B;`
Join	`select * from A inner join b on <Bedingung>;`
Natürlicher Join	`select * from A natural join B;`
Outer Join	`select * from A left outer join B on <Bedingung>;` man kann auch `right` oder `full` nutzen.
Join mit sich selber mit Alias	`select * from Angestellte A, Angestellte B where A.ID = B.Vorgesetzte;`
Hinweis	wenn zweimal ein gleichnamiges Attribut existiert, kann man mit z.B. A.ID und B.ID darauf zugreifen
Vereinigung	`select * from A union select * from B;`
Schnitt	`select * from A intersect select * from B;`
Differenz	`select * from A minus select * from B;`
Bei Vereinigung, Schnitt und Differenz werden Duplikate entfernt	

5.6 Datentypen in SQL

Datentyp	Beschreibung
`Integer/int, smallint`	Ganze Zahlen, smallint kleinere Zahlen (⇒ kleinerer Speicherbedarf)
`Float, Real, Double precision`	Gleitkommazahlen, Approximativ. Double precision für mehr Genauigkeit
`Decimal(i, j), Numeric(i, j)`	Feste Dezimalzahlen, i: Stellen insgesamt, j: Stellen nach dem Komma
`Serial`	Automatisch inkrementierende Ganzzahl, oft für Primärschlüssel
`Char(n), Varchar(n)`	Text, bei Char wird bei kürzerer Eingabe mit ' ' aufgefüllt, bei Varchar nicht
`create domain`	Definiert einen benutzerdefinierten Datentyp
Beispiel	Gibs nicht, ist mir zu blöd gerade?