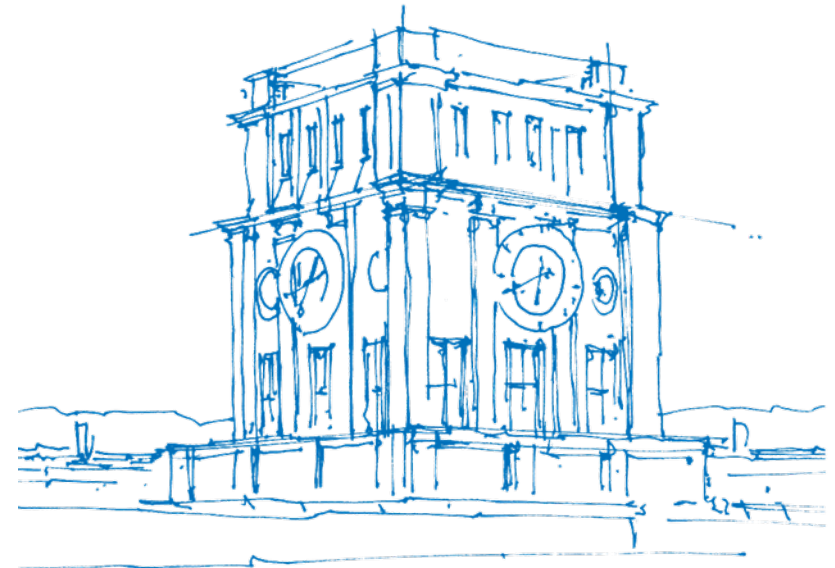


Grundlagen Datenbanken

Benjamin Wagner

22. November 2018



TUM Uhrenturm

Allgemeines

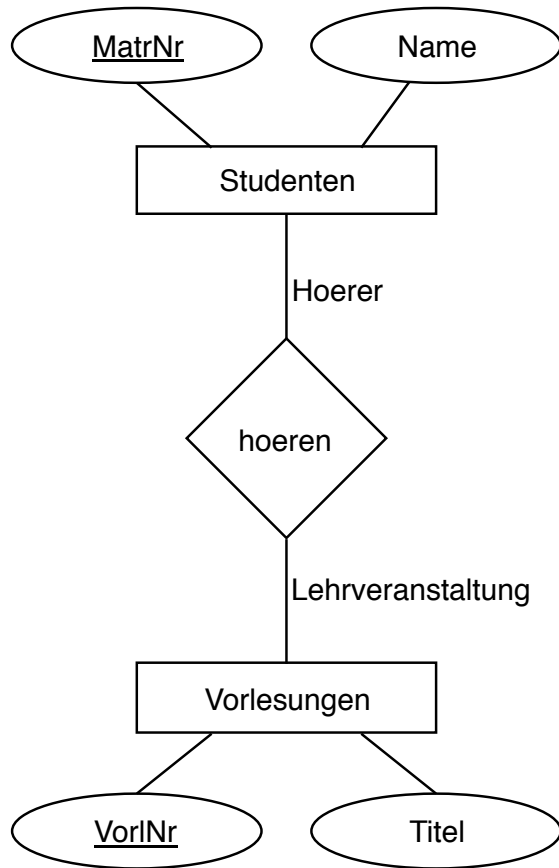
- Folien von mir sollen unterstützend dienen. Sie sind nicht von der Übungsleitung abgesegnet und haben keinen Anspruch auf Vollständigkeit (oder Richtigkeit).
- Bei Fragen: wagnerbe@in.tum.de
- Vorlesungsbegleitendes Buch von Professor Kemper (Chemiebib)
- Mein Foliensatz ist online: <https://github.com/wagjain/GDB2018>

Entity/Relationship-Modellierung

- **Entity:** Gegenstandstyp, welcher mit anderen Gegenständen in Beziehung steht
- **Relationship:** Modelliert die Beziehung zwischen Entities
- **Attribut:** Eine Eigenschaft einer Entity
- **Schlüssel:** Identifiziert eindeutig einen Datensatz
- **Rolle:** Welche Rolle nimmt eine Entity in einer Beziehung ein

⇒ Lässt sich als Graph darstellen, siehe Universitätsschema

Beispiel: Schema

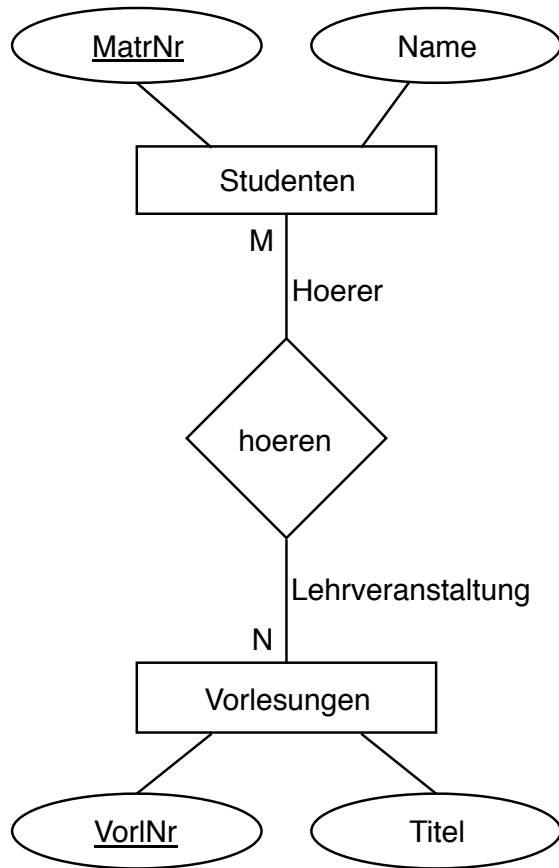


- Repräsentiert Studenten, die bestimmte Vorlesungen hören
- Schlüssel sind unterstrichen, ein Student ist eindeutig durch seine MatrNr bestimmt
- Hören modelliert eine Relationship zwischen Studenten und Vorlesungen
- Studenten treten hier in Rolle "Hörer" auf

Funktionalitäten

- Für eine Relationship R zwischen zwei Entities E_1 und E_2 gilt:
$$R \subset E_1 \times E_2$$
- Funktionalitäten charakterisieren die Relationship
- Mögliche Funktionalitäten: 1:1, 1:N, N:1, N:M
- Das kann auf Relationships mit vielen Entities ausgedehnt werden
- **Beispiel?**

Beispiel: Funktionalitäten

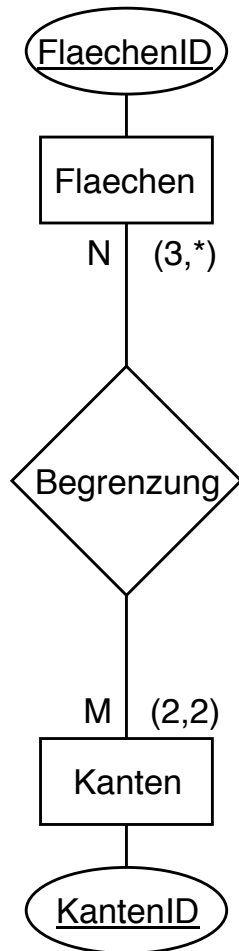


- Nun mit Funktionalitätsangaben
- Ein Student kann N Vorlesungen hören
- Eine Vorlesung kann von M Studenten gehört werden

(min, max)-Notation

- Ergänzt Funktionalitätsangaben
- **Achtung:** Eines ersetzt nicht das Andere!
- Betrachte Relationship $R \subset E_1 \times E_2$
- (min_1, max_1) bei E_1 bedeutet:
Für alle $e \in E_1$: mindestens min_1 Tupel $(e, \dots) \in R$
Für alle $e \in E_1$: maximal max_1 Tupel $(e, \dots) \in R$

Beispiel: (min, max)-Notation



- **Funktionalitäten sagen aus:**
 Eine Fläche kann M Kanten haben
 Eine Kante kann N Flächen begrenzen
- **(min, max) sagt aus:**
 Eine Fläche muss von mehr als drei Kanten begrenzt werden
 Eine Kante begrenzt genau zwei Flächen
- Volles Beispiel in den Folien

Sonstige Konzepte

- Existenzabhängige Entities: Funktionalität immer 1:N oder 1:1
- Generalisierung: "is-a"-Relationship
- Aggregation: "teil-von"-Relationship
- Das kann alles mit UML modelliert werden

Das relationale Modell

- Es gibt Domänen D_1, D_2, \dots, D_n , das entspricht Wertebereichen
z.B. Integer, Strings, Chars, Booleans
- Für eine Relation R gilt: $R \subset D_1 \times D_2 \times \dots \times D_n$
- Ein Tupel ist ein Element einer Relation
- Das Schema gibt die Struktur der Relationen vor

Das relationale Modell

- Es gibt Domänen D_1, D_2, \dots, D_n , das entspricht Wertebereichen
z.B. Integer, Strings, Chars, Booleans
- Für eine Relation R gilt: $R \subset D_1 \times D_2 \times \dots \times D_n$
- Ein Tupel ist ein Element einer Relation
- Das Schema gibt die Struktur der Relationen vor
- Sonstige Begriffe:

Ausprägung: der aktuelle Zustand einer Relation

Schlüssel: minimale Teilmenge von Attributen, welche Tupel eindeutig identifiziert

Primärschlüssel: Einer der Schlüsselkandidaten

Relationale Modellierung

- Wir können eine Relation nun aufschreiben:
User: {[Cust_Id, Name, Bday, Credit_Card]}
- Es können Datentypen ergänzt werden:
User: {[Cust_Id: Integer, Name: String, Bday: Date ...]}
- Falls partielle Funktionen gelten kann das Schema verfeinert werden
- Das darf aber nur bei gleichem Schlüssel passieren
- **Achtung:** NULL-Werte sind zu vermeiden

Relationale Algebra

- Beschreibt auf abstrakte Art und Weise Anfragen an die Datenbank
- Trotzdem in der Realität wichtig (\rightarrow später)
- Beachte: Es gibt eine ganze Reihe verschiedener Joins

Symbol	Bedeutung
$\sigma_{\text{Kondition}}$	Selektion
$\Pi_{\text{Attribute}}$	Projektion
\times	Kreuzprodukt
$\rho_{\text{neu} \leftarrow \text{alt}}$	Umbenennung
\bowtie	Join
$-, +, \div, \cup, \cap$	Mengenoperationen

Wichtigste Operatoren, **nicht vollständig**

Relationale Division

- Divisionsoperator sorgt oft für Verwirrung
- Kann bei Aussagen mit Allquantoren verwendet werden
- Bei $R \div S$ muss immer gelten: $Schema(S) \subset Schema(R)$
- Das Schema des Ergebnisses ist dann: $Schema(R) / Schema(S)$
- Unpräzise: es werden Tupel in R gesucht, welche für **jedes** Tupel in S einen Match haben

Relationale Division - $R \div S$

a1	a2	a3
1	2	1
1	2	2
2	1	5
3	5	1
3	5	2
3	5	3
4	8	1
4	8	2
4	6	3
5	5	1
5	5	2
5	5	3
5	5	4

÷

a3
1
2
3

=

a1	a2
3	5
5	5

Kalküle

- **Tupelkalkül:** Schreibweise (hoffentlich) aus Mathe-Vorlesungen bekannt: $\{t \mid P(t)\}$, mit $P(t)$ aussagenlogischer Formel
- **Domänenkalkül:** Domänenvariablen: $\{[v_1, \dots, v_n] \mid P(v_1, \dots, v_n)\}$
- **Achtung:** "Sicherheit" muss in Tupel- und Domänenkalkül sichergestellt sein. D.h. keine unendlichen Ergebnisse.
- **Mächtigkeit:** Relationale Algebra, Tupel- und Domänenkalkül gleich mächtig

Wiederholung: Relationen

Professoren:

PersNr	Name	Rang	Raum
2125	Sokrates	C4	226
2126	Russel	C4	232
2127	Kopernikus	C3	310
2133	Popper	C3	52
...

- Jede Tabelle hat Spalten: **Attribute**
- Die einzelnen Zeilen nennt man: **Tupel**
- Jede Spalte hat einen: **Typ**
- **Schlüssel** markieren ein Tupel eindeutig

SQL

- Standard Anfragesprache für relationale Datenbanken
- Web-Interface: <http://hyper-db.de/interface.html>
- Möglichkeit, Anfragen auf Uni-Schema zu realisieren
- Läuft auf Hyper (Datenbank des Lehrstuhls)
- Grundstruktur einer SQL-Anfrage:

1 **SELECT** . . .

2 **FROM** . . .

3 **WHERE** . . .

SQL - Datentypen

- Es gibt eine Reihe von Datentypen in SQL
- Z.B: char(n), varchar(n), integer, blob, date ...
- Damit können Tabellen erstellt werden:

```
1      CREATE TABLE Customers (  
2          CustId      integer not null ,  
3          Name        varchar(30) not null ,  
4          Birthday    date  
5      );
```

SQL - Einfache Anfrage

- Suche Namen aller Professor*innen, deren Rang C4 ist

SQL - Einfache Anfrage

- Suche Namen aller Professor*innen, deren Rang C4 ist

```
1      SELECT Name
2      FROM Professoren
3      WHERE Rang = 'C4'
```

- Suche alle Studierenden, die seit mehr als vier Semestern studieren

SQL - Einfache Anfrage

- Suche Namen aller Professor*innen, deren Rang C4 ist

```
1      SELECT Name
2      FROM Professoren
3      WHERE Rang = 'C4'
```

- Suche alle Studierenden, die seit mehr als vier Semestern studieren

```
1      SELECT *
2      FROM Studenten
3      WHERE Semester > 4
```

SQL - Sprachkonstrukte

- **Kreuzprodukt von Relationen** - *from R1, R2*
- **Aufgabe:** Was ist der Name, des Professors, der 'Ethik' liest

SQL - Sprachkonstrukte

- **Kreuzprodukt von Relationen** - *from R1, R2*
- **Aufgabe:** Was ist der Name, des Professors, der 'Ethik' liest

```
1      SELECT Professoren.Name
2      FROM Professoren, Vorlesungen
3      WHERE Professoren.persNr
4              = Vorlesungen.gelesenVon
5      AND Vorlesungen.Titel = 'Ethik'
```


SQL - Sprachkonstrukte

- **Duplikateliminierung** - *select distinct*
- **Aufgabe:** Suche das Semester aller Studierenden, die Logik hören

SQL - Sprachkonstrukte

- **Duplikateliminierung** - *select distinct*
- **Aufgabe:** Suche das Semester aller Studierenden, die Logik hören

```
1      SELECT DISTINCT studenten.semester
2      FROM studenten, hoeren, voerlesungen
3      WHERE studenten.matrnr = hoeren.matrnr
4      AND hoeren.vorlNr = vorlesungen.vorlNr
5      AND vorlesungen.titel = 'Logik'
```

SQL - Sprachkonstrukte

- **Relation benennen** - *from Professoren p1, Professoren p2*
- **Mengenoperationen** - *union, intersects, minus*
- **Quantor** - *exists*

```
1      (SELECT p.Name
2         FROM Professoren p
3         WHERE NOT EXISTS (
4             SELECT *
5             FROM Vorlesungen v
6             WHERE v.gelesenVon = p.persNr);)
7
8      INTERSECT
9
10     (...)
```

SQL - Sprachkonstrukte

- **Gruppierung** - *group by*
- Bildet Gruppen von Tupeln mit den selben Werten in den Attributen der "group by" Klausel
- Auf den anderen Attributen können dann Aggregatsfunktionen aufgerufen werden
- **Aufgabe:** Wie viele Studenten studieren in welchem Semester?

SQL - Sprachkonstrukte

- **Gruppierung** - *group by*
- Bildet Gruppen von Tupeln mit den selben Werten in den Attributen der "group by" Klausel
- Auf den anderen Attributen können dann Aggregatsfunktionen aufgerufen werden
- **Aufgabe:** Wie viele Studenten studieren in welchem Semester?

```
1      SELECT semester , count (*)  
2      FROM Studenten  
3      GROUP BY semester
```

SQL - Sprachkonstrukte

- **Gruppierung** - *group by*
- Es gibt viele Aggregatsfunktionen: avg, max, min, count, sum
- Für Selektion auf Aggregaten: *having*
- **Aufgabe:** Welche Professoren halten mehr als 2 Vorlesungen?

SQL - Sprachkonstrukte

- **Gruppierung** - *group by*
- Es gibt viele Aggregatsfunktionen: avg, max, min, count, sum
- Für Selektion auf Aggregaten: *having*
- **Aufgabe:** Welche Professoren halten mehr als 2 Vorlesungen?

```
1      SELECT p.Name , count (*)
2      FROM Professoren p, Vorlesungen v
3      WHERE p.persNr = v.gelesenVon
4      GROUP BY v.gelesenVon
5      HAVING count (*) > 2
```

SQL - Sprachkonstrukte

- **Temporäre Relation** - *with ... as()*
- Komplexe Anfragen können u.U. modularisiert werden

```
1      WITH h AS (SELECT VorlNr ,
2                      count(*) AS AnzProVorl
3                      FROM hoeren
4                      GROUP BY VorlNr) ,
5
6      . . .
```


SQL - Sprachkonstrukte

- **String Vergleiche** - *like ...*
- `'_'` dient als Placeholder für ein Zeichen
- `'%'` dient als Placeholder für beliebig viele Zeichen

```
1      SELECT *  
2      FROM Studenten  
3      WHERE name like 'T%eophrastos';
```

SQL - Sprachkonstrukte

- **Fallunterscheidung** - *case when ...*
- Die erste passende Bedingung wird ausgewertet

```
1      SELECT MatrNr , (CASE
2          WHEN Note < 1.5 THEN 'sehr_gut'
3          WHEN Note < 2.5 THEN 'gut'
4          WHEN Note < 3.5 THEN 'befriedigend'
5          WHEN Note < 4.0 THEN 'ausreichend'
6          ELSE 'nicht_bestanden'
7          END)
8      FROM prüfen;
```