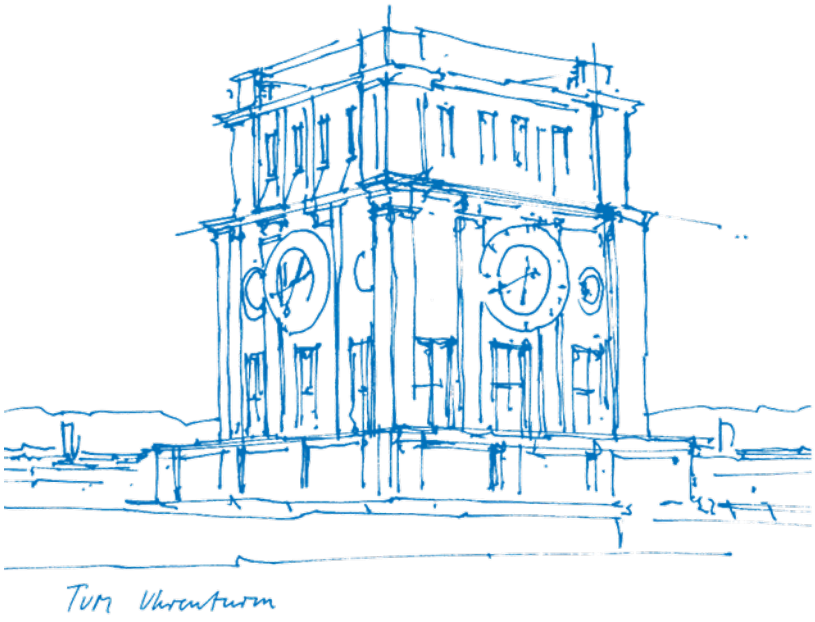
Grundlagen Datenbanken

Benjamin Wagner

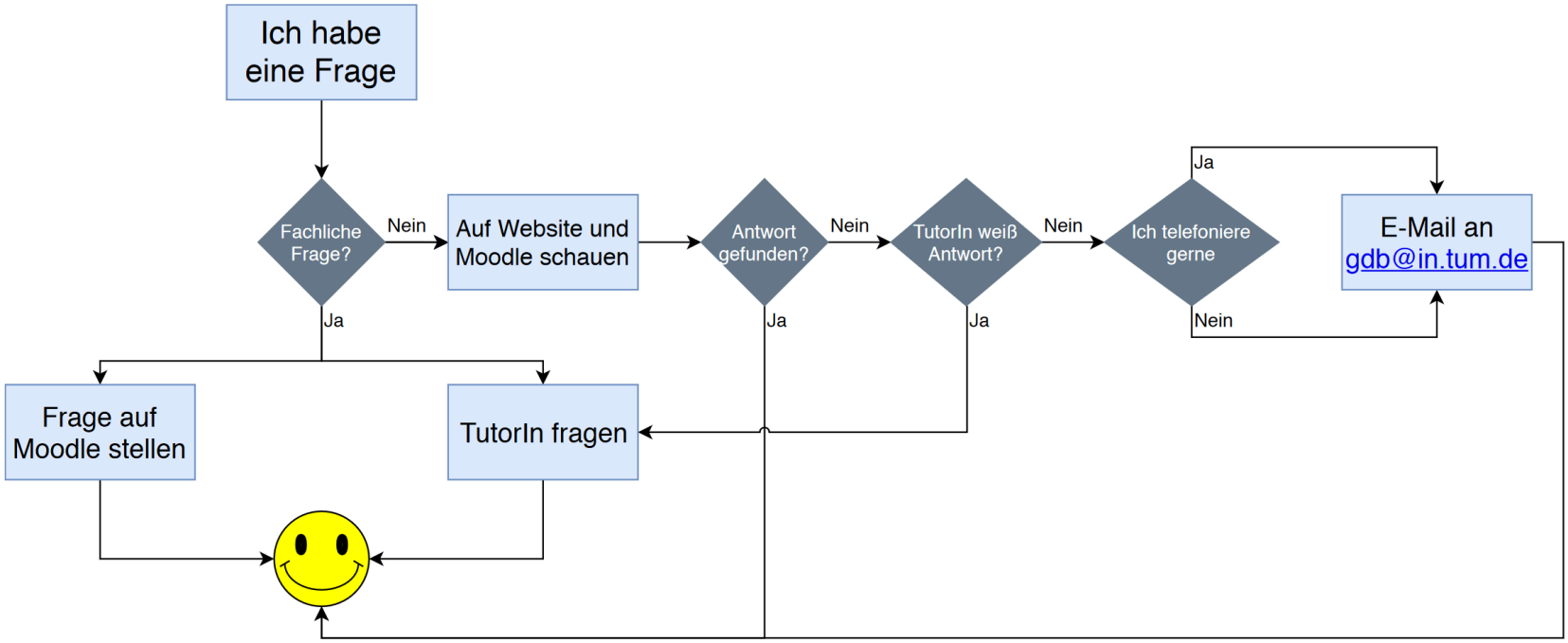
7. November 2018



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/wagjamin/GDB2018`

Kontakt zur Übungsgsleitung



Bonusverfahren

- Keine Hausaufgaben im klassischen Sinne
- Bonusverfahren im Rahmen der Tutorübungen
- Aufgaben werden von Teilnehmern vorgerechnet
- +1 für Anwesenheit in der Tutorgruppe
- +1 für sinnvolles Vorstellen von Hausaufgaben
- Bonus: Mit mind. 13 + 2 (Donnerstags) Punkten am Ende des Semesters
- Punkte gibt es nur in einer Übung pro Woche

$\Rightarrow$ Aktive und regelmäßige Teilnahme ist wichtig

Vorlesungsinhalt

- Nutzung relationaler Datenbanksysteme
  - Relationale Entwurfstheorie
  - Anfragesprachen (SQL)
- Innere Funktionsweise einer Datenbank
  - Physische Datenorganisation
  - Anfragebearbeitung
  - Transaktionsverwaltung
  - Fehlerbehandlung

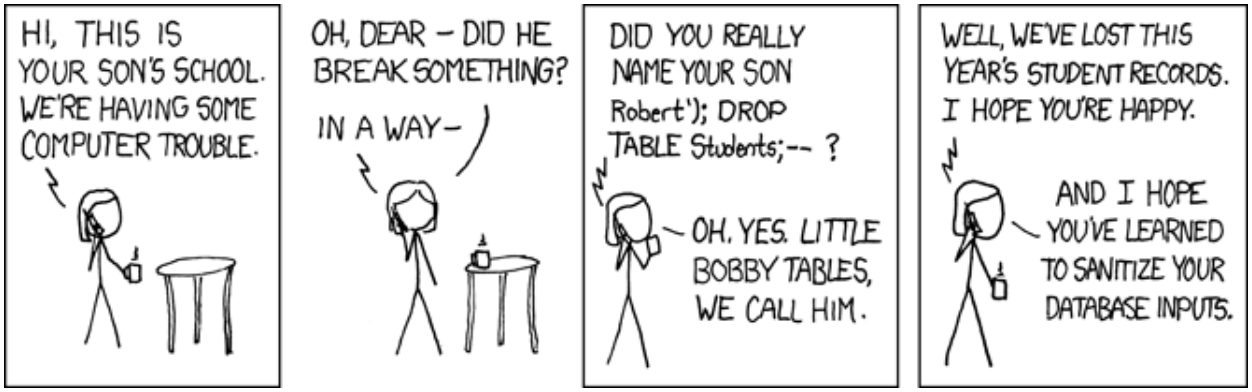
$\Rightarrow$ Verständnis von Nutzung und Funktionsweise moderner Datenbanksysteme

Fragen ?

Datenbanken sind cool

- Hochperformante Softwaresysteme
- Hohe Komplexität durch Anforderungen an Daten
  - **ACID**: Atomicity, Consistency, Isolation, Durability
  - Mehr dazu im Verlauf des Semesters
- Zentrales Element in modernen Anwendungen
- Trotz hohem Anwendungsbezug theoretischere Betrachtung möglich

$\Rightarrow$ Wissen zu Datenbanken ist nützlich **und** interessant



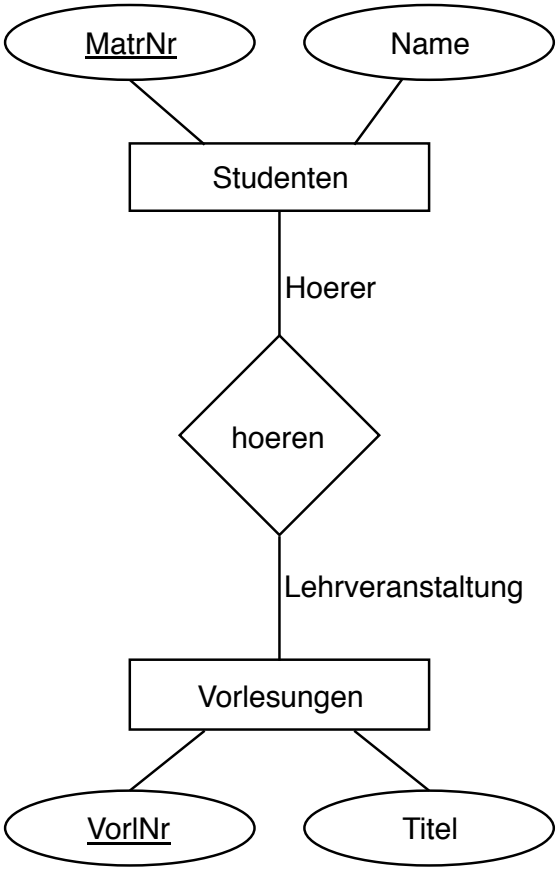
Quelle: https://xkcd.com/327/

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

$\Rightarrow$ Lässt sich als Graph darstellen, siehe Universitätsschema

Beispiel: Schema

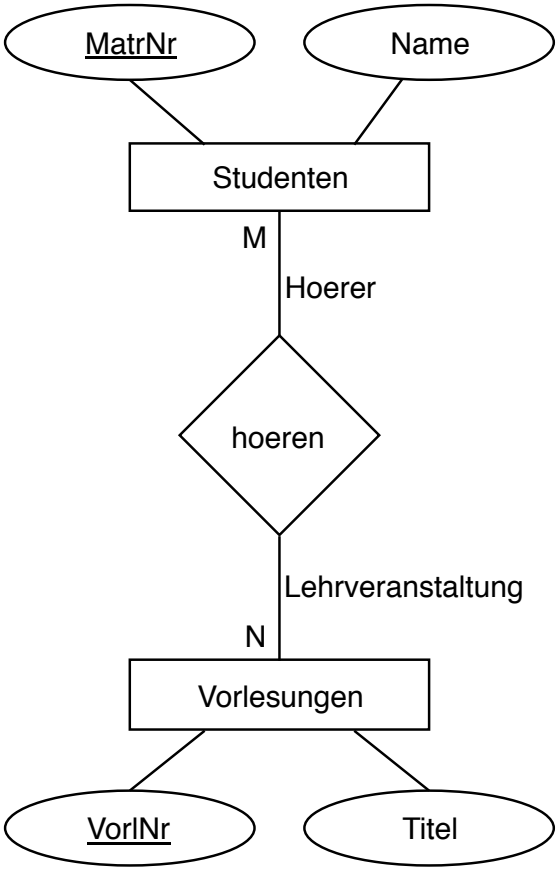


- Repräsentiert Studenten, die bestimmte Vorlesungen hören
- Schlüssel sind unterstrichen, ein Student ist eindeutig durch seine MatrNr bestimmt
- Hören modelliert eine Relationship zwischen Studenten und Vorlesungen
- Studenten treten hier in Rolle ”Hörer” auf

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

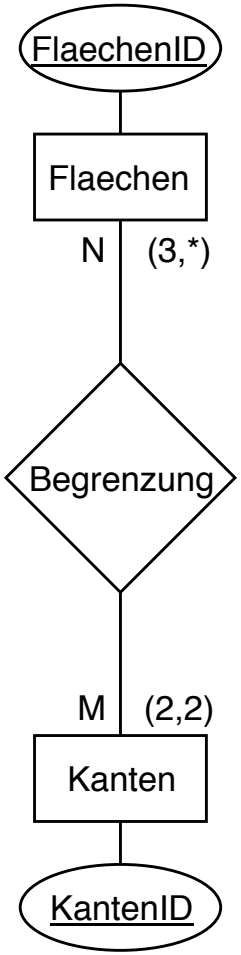
(min, max)-Notation

- 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 $(e, ...) \in R$

  Für alle $e \in E_1$: maximal max_1 Tupel $(e, ...) \in R$

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

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

Das relationale Modell

- Es gibt Domänen $D_1, D_2, \ldots, D_n$, das entspricht Wertebereichen z.B. Integer, Strings, Chars, Booleans
- Für eine Relation R gilt: $R \subset D_1 \times D_2 \times \ldots \times D_n$
- 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
$\sigma_{Kondition}$	Selektion
$\prod_{Attribute}$	Projektion
$\times$	Kreuzprodukt
$\rho_{neu \leftarrow alt}$	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

$\div$

a3
1
2
3

$=$

a1	a2
3	5
5	5

Kalküle

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