Grundlagen Datenbanken: Übung 01

Tanmay Deshpande

Gruppe 20 & 21

E-Mail: ge94vem@mytum.de

Organisatorisches

- Abstimmung für die Startzeit
- **Kommunikation:**
  - Allgemeine Fragen zum Vorlesungsstoff, Übungsblätter, usw. => Tutoren fragen (E- Mail: ge94vem@mytum.de oder per Zulip-DM)
  - Fragen zum Vorlesungsbetrieb, die von öffentlichem Interesse sind => entsprechender Zulip-Channel
  - sonstige Fragen, die nicht von öffentlichem Interesse sind => E-Mail an gdb@in.tum.de
- Tutoriumsfolien jede Woche auf GitHub hochgeladen: https://github.com/tanmaydeshp/GDB2425
- **Anmerkung:** Vorlesung richtet sich nach dem Buch von Prof. Kemper *„Datenbank-Systeme: Eine Einführung“*. In großen Stückzahlen in Bibliothek ausleihbar!
- Vorlesungsseite: https://db.in.tum.de/teaching/ws2425/grundlagen/
- Übersicht aller Tutorien: https://erdbtutor.db.in.tum.de/schedule?course_id=16

QR-Code für die Folien

Bonusverfahren

- Punktesystem:
  - +1 für aktive Anwesenheit (max. einmal pro Woche, nicht notwendigerweise in einer bestimmten Gruppe)
  - Daher Anwesenheitskontrolle bei jedem Tutorium
  - Falls ihr nicht in dieser Gruppe angemeldet seid, meldet euch damit ich eure Punkte temporär eintragen kann
  - Falls ihr permanent eure Gruppe ändern möchtet, mir informieren
  - +1 für Vorrechnen der (Teil-)Aufgaben (müssen nicht 100% korrekt sein, aber Eigenarbeit und Beschäftigung mit dem Stoff sollte ersichtlich sein)
  - Wenn ihr (*Anzahl Übungswochen* + 2) Punkte erreicht, bekommt ihr einen Notenbonus von 0,3 auf die Klausur (gilt für Endterm und Retake)
  - Bonus gilt nur, wenn ihr die Klausur besteht

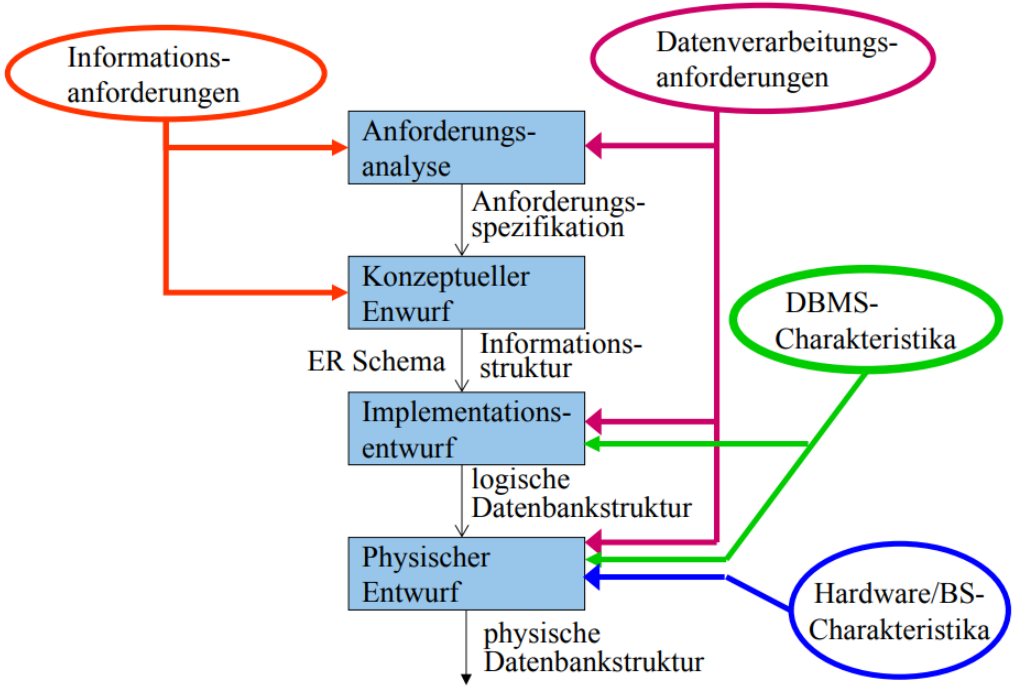
Wiederholung

Woche 01

Warum Datenbanken?

- 2,5 Trillionen Bytes Daten pro Tag erzeugt (2018) (Speicherkapazität von 36 Millionen iPads)
- Um derartige Datenmengen effizient und organiziert zu verwalten brauchen wir Datenbanken bzw. DBMS
- Vorteile von DBMS:
  - Vermeiden Integritätsverletzung, Redundanz, Datenverlust
  - Mehrbenutzersynchronisation
  - Einfache Einbindung in andere Programme

Phasen des Datenbankentwurfs



Anforderungsanalyse:
siehe Kapitel 2 des Buchs

Relationale Datenbank

- Besteht aus eine Menge von Relationen
- Eine Relation kann als eine Tabelle dargestellt werden. Es besteht aus:
  - Zeilen ↔ Tupel, Spalten ↔ Attribute/Felder
  - Domäne D ↔ Wertebereich (Bsp: integer)
- Beispielschema: $\text{studenten}(\text{matrnr}: \text{integer},\ \text{name}: \text{string},\ \text{semester}: \text{integer})$

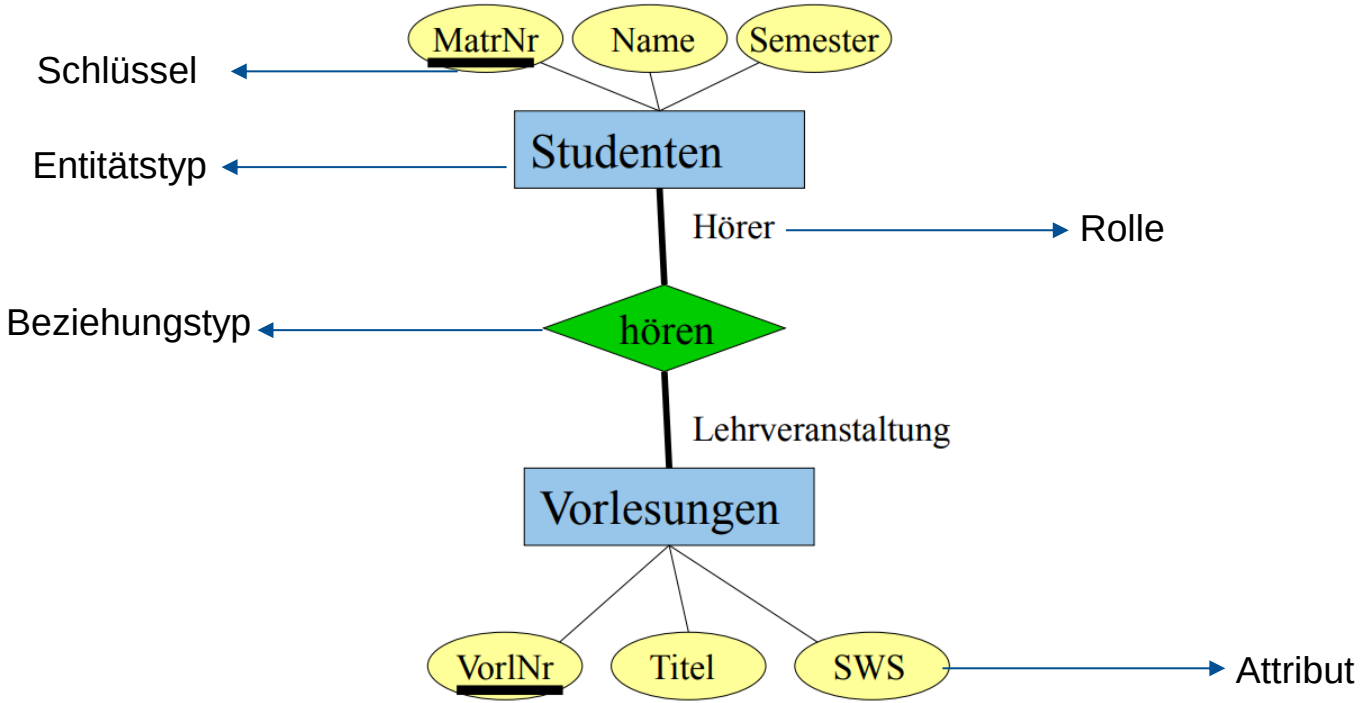
$$studenten \subseteq D_{matrnr} \times D_{name} \times D_{semester}$$

matrnr	name	semester
24002	Xenokrates	18
25403	Jonas	12
26120	Fichte	10
26830	Aristoxenos	8
27550	Schopenhauer	6
28106	Carnap	3
29120	Theophrastos	2
29555	Feuerbach	2

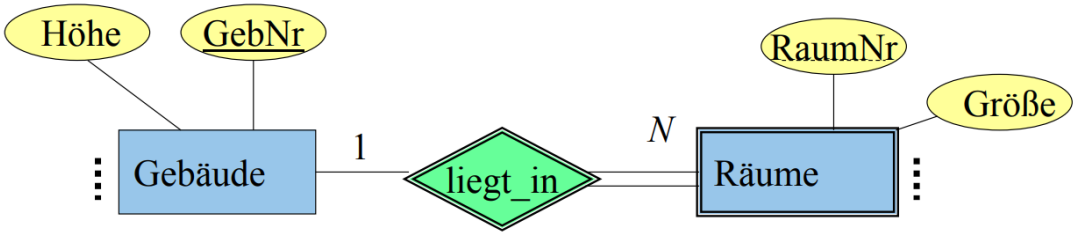
Schlüssel

- **Primärschlüssel:** Minimale Menge von Attributen, die es ermöglicht, ein Tupel in einer Relation eindeutig zu identifizieren
- Bsp: MatrNr bei Studenten
- In Schema mit Unterstrich gezeigt. Bsp: *Studenten:* {MatrNr: integer, Name: string, Semester: integer}
- **Fremdschlüssel:** Verweisen auf Primärschlüssel einer anderen Relation
- Bsp: Boss bei Assistenten (Primärschlüssel von Professoren)

Entity-Relationship Modell (ERM)



Existenzabhängige/Schwache Entitätstypen



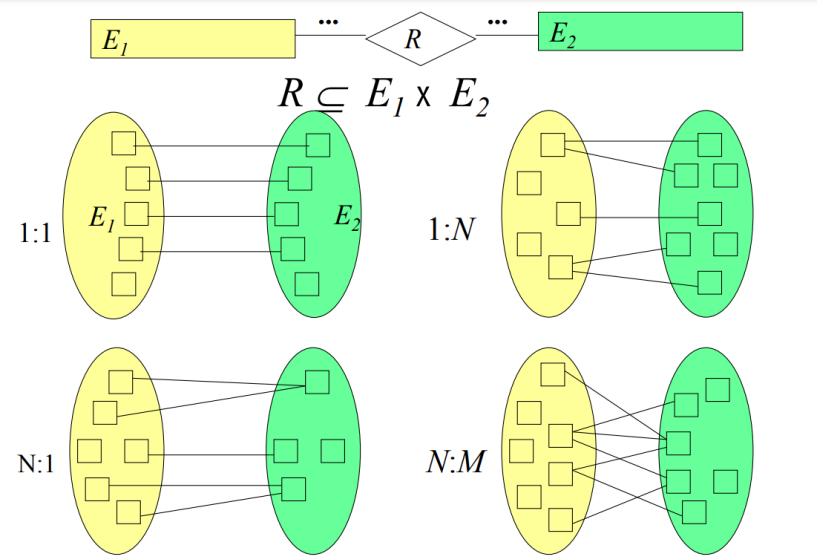
- Immer in 1:N (oder seltener, 1:1) Beziehungen zum übergeordneten Entitätstypen
- Schlüsselattribut durch gepunkteten Unterstrich gezeigt
- Schlüssel des schwachen Entitätstyps immer in Kombination mit dem Schlüssel des übergeordneten Entitätstypen gebildet. In diesem Fall: {GebNr, RaumNr}

Beziehungen

- Die Ausprägung einer Beziehung ist eine Teilmenge des kartesischen Produkts alle an der Beziehung beteiligten Entitätstypen
- $R \subseteq E_1 \times E_2 \times \cdots \times E_n$, wobei *n* der Grad der Beziehung gennant wird
- Bsp: hoeren $\subseteq$ Studenten $\times$ Vorlesungen

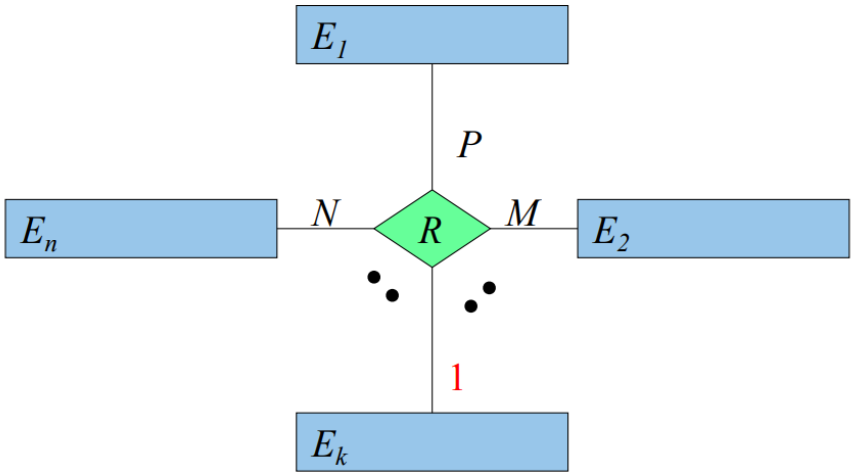
Funktionalitäten

- Beziehungstypen lassen sich nach Funktionalitäten charakterisieren
- Geben an, mit wie vielen anderen Entitäten eine Entität eine Beziehung eingeht
- Funktionalitäten stellen Integritätsbedingungen dar, die immer gelten müssen



Funktionalitäten als partielle Funktionen

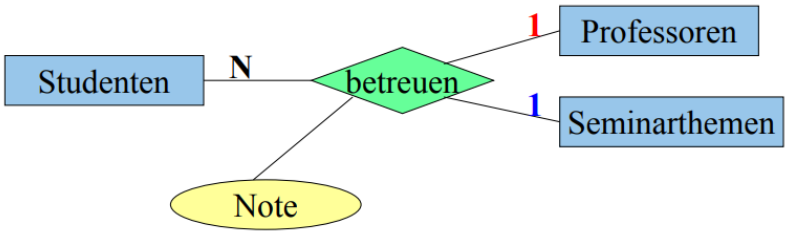
Funktionalitäten bei *n*-stelligen Beziehungen



$$R : E_1 \times \ldots \times E_{k-1} \times E_{k+1} \times \ldots \times E_n \rightarrow E_k$$

- Sei R eine Beziehung zwischen den Entitymengen E1,…,En, wobei die Funktionalität bei der Entitymenge Ek (1<= k <= n) mit einer “1” spezifiert sein soll und die anderen Funktionalitäten seien beliebig, dann gilt, dass durch R die folgende partielle Funktion vorgegeben wird:
  R : E1 x … x Ek-1 x Ek+1…X En → Ek
- Intuitiv: Von der rechten Seite der partiellen Funktion steht immer der Entitätstyp mit Funktionalität “1”, und links steht das Kreuzprodkt der restlichen Entitätstypen

Beispiel-Beziehung: *betreuen*

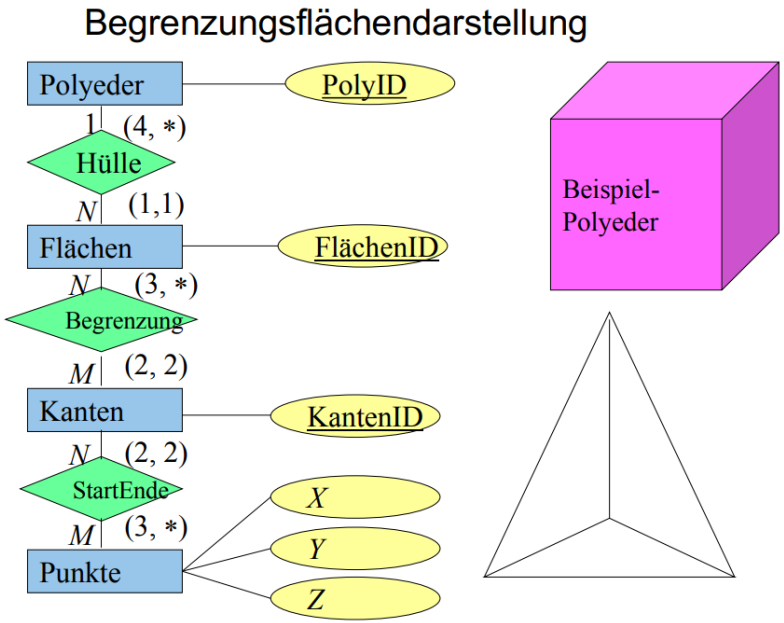


betreuen : Professoren x Studenten → Seminarthemen

betreuen : Seminarthemen x Studenten → Professoren

- Durch *betreuen* erzwungene Konsistenzbedingung: Ein Student darf bei dem selben Professoren nur ein Seminarthema ableisten!
- Funktionalitätsangabe in diesem Fall durch Kardinalitätsangaben
- Andere Möglichkeit : (min,max)-Notation

(min-max)-Notation



- Notation der Form (x, y), wobei
  x = minimale Anzahl an Tupel des Entitätstypen, die an der Beziehung beteiligt sein müssen
  y = maximale Anzahl an Tupel des Entitätstypen, die an der Beziehung beteiligt sein dürfen
- “*” bedeutet beliebig viele
- Im ersten Blick kann es verwirrend aussehen, da es “andersrum” aussieht. Immer an die Definition denken!

Aufgaben

Woche 01

Aufgabe 1

- Schätzen Sie die Größe der Datenbank des Amazon-Shops ab. Berücksichtigen Sie dabei Produkte, Kunden und Bestellungen. Schätzen Sie außerdem ab, wie viele Transaktionen pro Sekunde von dieser Datenbank abgewickelt werden und leiten Sie daraus ab, wie schnell die Datenbank wächst (Bytes pro Sekunde).

- a) Ermitteln Sie möglichst aktuelle statistische Werte für die Anzahl der Produkte, Kunden und Bestellungen sowie die durchschnittliche Anzahl neuer Bestellungen pro Sekunde.
- b) Berechnen Sie anhand der in a) geschätzten Werte die Größe der Datenbank und den durch neuen Bestellungen verursachten Durchsatz (Bytes pro Sekunde).

Lösungsvorschlag 1

a)

- Anzahl Produkte = 350 Millionen
- Anzahl Kunden = 600 Millionen
- Anzahl Bestellungen = 1,5 Milliarden
- Bestellungen/Sekunde = 20

b)

- **Größe eines Datensatzes eines Produkts:** – Beschreibung + Name (1 KB) – Produktbilder (1 MB) – Produktvideos (1 MB durchschnittlich, nicht alle Produkte haben ein Video) – Preis(e), Varianten und Lieferbedingungen (Etwa 1 KB) – Rezensionen potentiell mit Bildern und Videos (5 MB) Insgesamt also **7,002 MB pro Produkt**.

Lösungsvorschlag 1 (contd.)

- **Größe eines Datensatzes eines Kunden:** – Stammdaten (Name, E-Mail-Adresse, etc.) (1 KB) – Zahlungs- und Adressinformationen (1 KB) Insgesamt also **2 KB pro Kunde**.
- **Größe einer Bestellung:** – Referenz auf den Kunden (8 B) – Pro bestelltem Artikel: ∗ Referenz auf Artikel (8 B) ∗ Anzahl (8 B) ∗ Referenz auf Lieferant (8 B) ∗ Lieferdatum (8 B) ∗ Lieferhinweise oder -kommentare (50 B) ∗ Möglicher Rabatt (8 B) ∗ Steuer (8 B) Pro Artikel also 98 B ≈ 100 B. Bei durchschnittlich 2 Artikeln pro Bestellung also **200 B**.
- Mit den Größen aus a), kommt man auf die folgenden Zahlen:
  - Produkte: $350 \cdot 10^6 \cdot 7,002\ \text{MB} = 2,4507\ \text{PB}$
  - Kunden: $600 \cdot 10^6 \cdot 2\ \text{KB} = 1,2\ \text{TB}$
  - Bestellungen: $1,5 \cdot 10^9 \cdot 200\ \text{B} = 300\ \text{GB}$

  Der Durchsatz der Bestellungen ist $20 \cdot 200\ \text{B/s} = 4\ \text{KB/s}$

Aufgabe 2

- Sie designen eine Webanwendung zur Univerwaltung. Früh entschließen Sie sich zum Einsatz eines Datenbanksystems als Backend für Ihre Daten. Ihr Kollege ist skeptisch und würde die Datenverwaltung lieber selbst implementieren. Überzeugen Sie ihn von Ihrem Entschluss. Finden Sie stichhaltige Antworten auf die folgenden von Ihrem Kollegen in den Raum gestellten Äußerungen:

- a) Die Installation und Wartung eines Datenbanksystems ist aufwendig, die Erstellung eines eigenen Datenformats ist straight-forward und flexibler.
- b) Mehrbenutzersynchronisation wird in diesem Fall nicht benötigt.
- c) Es ist unsinnig, das jeder Entwickler zunächst eine eigene Anfragesprache (SQL) lernen muss, nur um Daten aus der Datenbank zu extrahieren.
- d) Redundanz ist hilfreich, wieso sollte man auf sie verzichten?

Lösungsvorschlag 2

a)

- Einige Datenformate inherent unflexibel (Bsp: Maximale Filegröße 2GB bei FAT32 Dateien)
- Durch manuelle Erstellung von Dateiformaten werden nicht-standardisierte Datentypen verwendet
- Uneinheitliche Datenspeicherung
- Im Vergleich dazu erfördert Erstinstallation wenig Aufwand heutzutage (Bsp: eingebettete Datenbanken wie sqlite)

b)

- Mehrbenutzersynchronisation immer erforderlich, wenn mehr als ein Nutzer auf dem Datenbank zugreift
- Kann nicht “nachgepatched” werden
- ACID-Properties von Datenbanken gewährleistet

Lösungsvorschlag 2 (contd.)

c)

- Ein eigenes Datenformat und dessen API muss auch gelernt werden
- SQL ist standardisiert und kann beim Wechsel des DBMS weiterverwendet werden

d)

- Redundanz sorgt auch für Anomalien (Bsp: Update, Delete)

Aufgabe 3

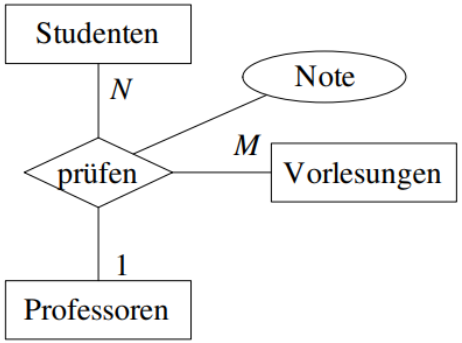
- Beim konzeptuellen Entwurf hat man gewisse Freiheitsgrade hinsichtlich der Modellierung der realen Welt. Unter anderem hat man folgende Alternativen, die Sie an unserem Universitätsschema beispielhaft illustrieren sollten:

a) Man kann ternäre Beziehungen in binäre Beziehungen transformieren. Betrachten Sie dazu die Beziehung prüfen und erläutern Sie die Vor- und Nachteile einer solchen Transformation.

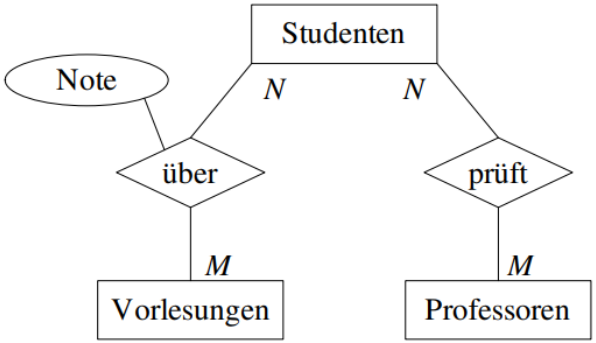
b) Man hat manchmal die Wahl, ein Konzept der realen Welt als Beziehung oder als Entitytyp zu modellieren. Erörtern Sie dies wiederum am Beispiel der Beziehung prüfen im Gegensatz zu einem eigenständigen Entitytyp Prüfungen.

c) Ein Konzept der realen Welt kann manchmal als Entitytyp mit zugehörigem Beziehungstyp und manchmal als Attribut dargestellt werden. Ein Beispiel hierfür ist das Attribut Raum des Entitytyps Professoren im bekannten Uni Schema. Diskutieren Sie die Alternativen.

Lösungsvorschlag 3a



Studenten × Vorlesungen → Professoren

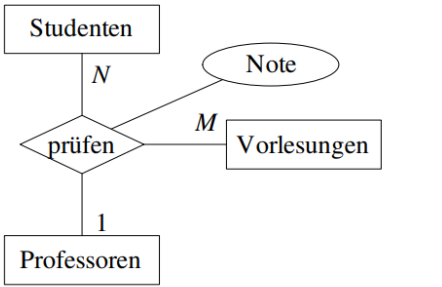


Keine Einschränkungen

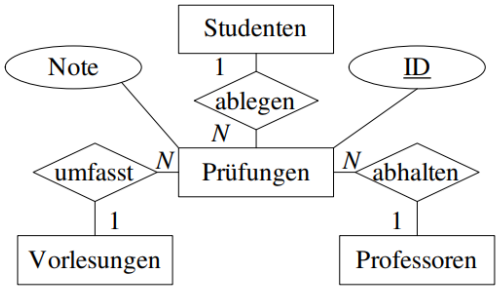
Nachteile:

- Semantikverlust (Partielle Funktion gilt nicht mehr)
- Inkonsistenz (z.B. wenn wir ein Attribut wie *Prüfungszeit* hinzufügen, um die Semantik zu behalten)
- Reale Welt unzureichend wiedergegeben

Lösungsvorschlag 3b



Studenten × Vorlesungen → Professoren



Keine Einschränkungen

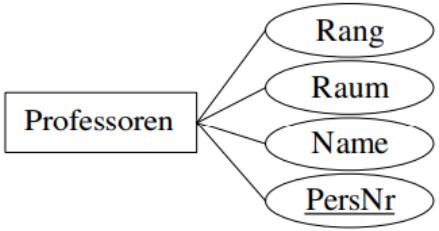
Nachteile:

- Semantikverlust (es ist möglich, dass es kein Prüfer zu einer Prüfung gibt)
- Konsistenzverletzung (ein Student kann bei einer Vorlesung mehr als eine Prüfung schreiben und bei verschiedenen Profs. prüfen lassen)

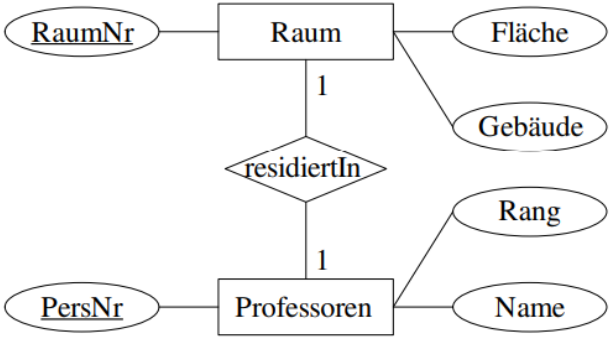
Vorteil:

- Spezifiziert, dass jede Prüfung nur eine Vorlesung umfasst

Lösungsvorschlag 3c

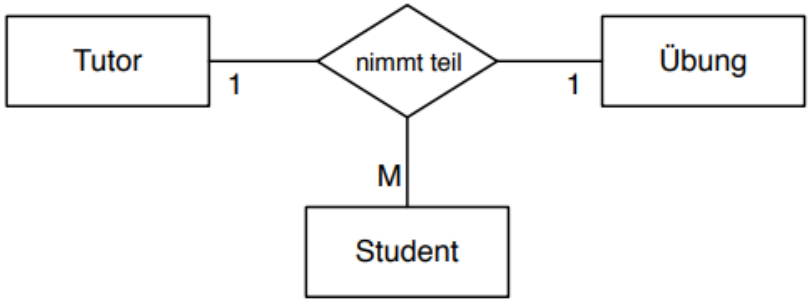


Raum als Attribut



Raum als Entitätstyp

Aufgabe 4



Ignorieren Sie die Funktionalitätsangaben und beantworten Sie:

a) Wie viele partielle Funktionen der Form $A \times B \rightarrow C$ können in einer ternären Beziehung auftreten (Ignorieren Sie beim Zählen die Reihenfolge auf der linken Seite der Beziehung).

b) Nennen Sie alle möglichen partiellen Funktionen in der hier gezeigten Beziehung „nimmt teil“.

c) Nennen Sie für jede Funktion in Prosa, welche Einschränkung diese darstellt, falls sie gilt.
Unter Berücksichtigung der Funktionalitätsangaben:

d) Welche partiellen Funktionen gelten hier?

Lösungsvorschlag 4

a) Drei mögliche partielle Funktionen

b) Tutor × Übung → Student (1)
Tutor × Student → Übung (2)
Übung × Student → Tutor (3)

c) (1): ein Tutor darf pro Übung nur einen Studenten haben
(2): ein Student darf bei einem Tutor nur eine Übung besuchen
(3): Es gibt nur einen Tutor für einen konkreten Student pro Übung

d) (1) gilt nicht
(2) und (3) gelten, auch bei uns an der Uni so