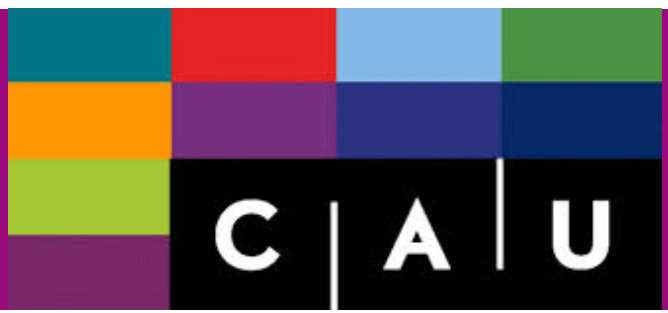
Informationssyteme[1]

Kapitel 5: Daten-Modellierung mit dem E/R-Modell

Wintersemester 2020/21

Prof. Dr. Peer Kröger
Institut für Informatik
Arbeitsgruppe Informationssysteme und Data Mining



[1] Die folgenden Folien basieren in großen Teilen auf Material zur Vorlesung "Datenbanksysteme", die ich in meiner Zeit an der LMU München mehrmals gehalten habe. Das Material ist von den damaligen Kollegen maßgeblich (mit-)gestaltet worden, insbes. von Prof. Dr. C. Böhm.

Übersicht

1. Das Entity/Relationship-Modell

2. Vom E/R-Modell zur Relation

Agenda

1. Das Entity/Relationship-Modell
2. Vom E/R-Modell zur Relation

Schema-Entwurf

- Generelle Aufgabe:

  Finde eine formale Beschreibung (Modell) für einen zu modellierenden Teil der realen Welt
- Zwischenstufen:
  - Beschreibung durch natürliche Sprache (Pflichtenheft)

    Beispiel: *In der Datenbank sollen alle Studierenden mit den durch sie belegten Lehrveranstaltungen gespeichert sein*
  - Beschreibung durch abstrakte grafische Darstellungen

  - Beschreibung im relationalen Modell

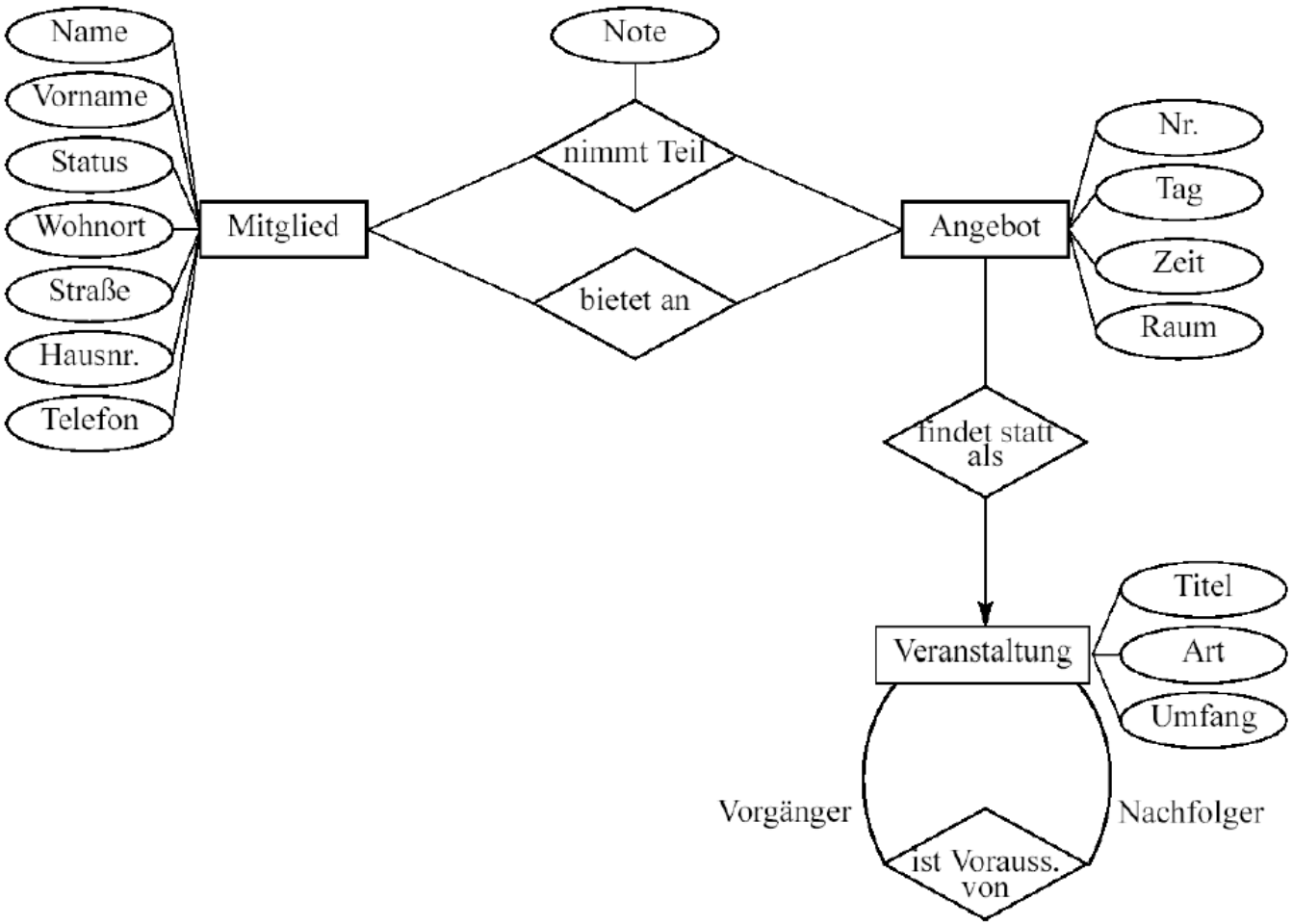
```
CREATE TABLE student (...);
CREATE TABLE vorlesung(...);
```

Das Entity/Relationship (E/R) Modell

- Dient dazu, für einen Ausschnitt der realen Welt ein konzeptionelles Schema zu erstellen
- Grafische Darstellung: E/R-Diagramm
- Maschinenfernes Datenmodell / hohes Abstraktionsniveau
- Überlegungen zur Effizienz spielen keine Rolle
- Das E/R-Modell muss in ein relationales Schema überführt werden
  - Einfache Grundregeln zur Transformation
  - Gewinnung eines effizienten Schemas erfordert tiefes Verständnis vom Zielmodell

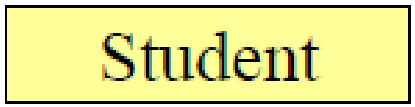
Das Entity/Relationship (E/R) Modell

Beispiel: Lehrveranstaltungen

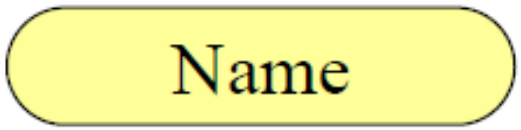


Elemente des E/R-Modells

- Entities (eigentlich: Entity Sets): Objekttypen



- Attribute: Eigenschaften

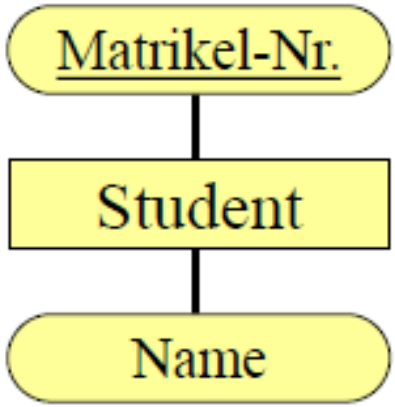


- Relationships: Beziehungen zw. Entities

Entscheidende Aufgabe des Schema-Entwurfs: Finde geeignete Entities, Attribute und Relationships

Entities und Attribute

- Entities
  - Objekte, Typen, "Seiendes"
  - Objekte der realen Welt, unterscheidbar
  - Bsp: Mensch, Haus, Vorlesung, Bestellung, ...

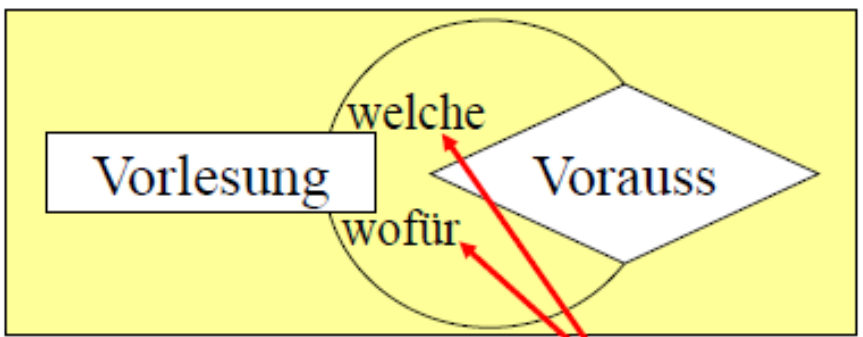


- Attribute
  - Entities werden durch charakterisierende Eigenschaften beschrieben
  - Einfache Datentypen wie INT, STRING usw.
  - Bsp: Farbe, Gewicht, Name, Titel, ...
  - Häufig beschränkt man sich auf die wichtigsten Attribute
- Schlüssel
  - Ähnlich definiert wie im relationalen Modell
  - Primärschlüssel-Attribute werden unterstrichen

Relationships (Beziehungen)

- Stellen Zusammenhänge zwischen Entities dar
- Beispiele:

Ausprägung: belegt (Anton, Informatik 1)
belegt (Berta, Informatik 1)
belegt (Caesar, Wissensrepräsentation)
belegt (Anton, Datenbanksysteme 1)

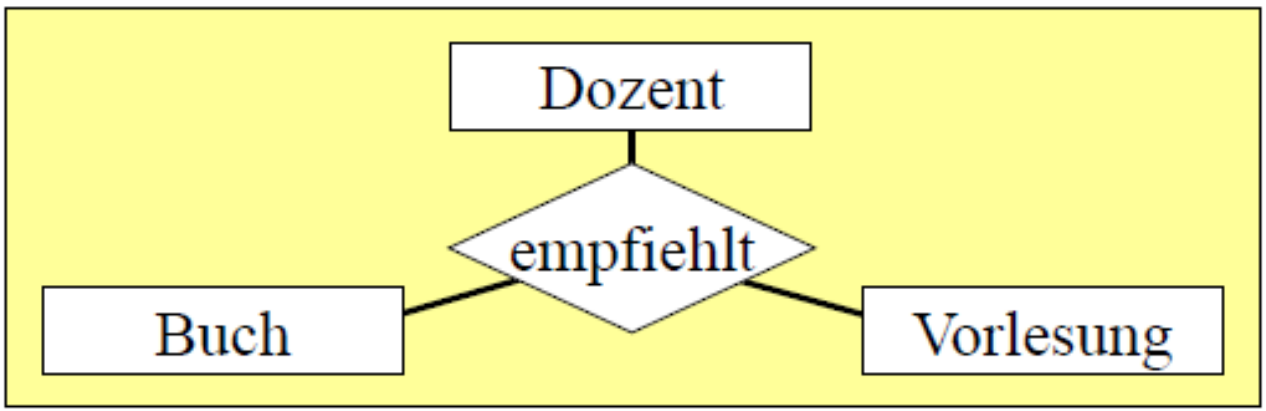


Ausprägung:
Vorauss (I 1, DBS 1)
Vorauss (I 1, Software-Eng.)
Vorauss (DBS 1, DBS 2)
Vorauss (I 1, Wissensrepr.)

unterschiedliche Rollen

Relationships (Beziehungen)

- Relationships können eigene Attribute haben (z.B. "belegt"-Beziehung von oben hat noch Attribut "Note", ...)
- Mehrstellige (hier: ternäre) Relationships:



Ausprägung: empfiehlt (Böhm, Heuer&Saake, DBS 1)
empfiehlt (Böhm, Kemper&Eickler, DBS 1)
empfiehlt (Böhm, Bishop, Informatik 1)
empfiehlt (Böhm, Bishop, Programmierkurs)

Funktionalität von Relationships

1:1-Beziehung (one-to-one-relationship)

- Charakteristik: Jedes Objekt aus dem linken Entity steht in Beziehung zu höchstens einem Objekt aus dem rechten Entity und umgekehrt.
- Grafische Notation: Pfeile auf jeder Seite

Beispiel: 1:1-Beziehung

Jede Abteilung hat maximal einen Leiter und kein Angestellter leitet mehrere Abteilungen

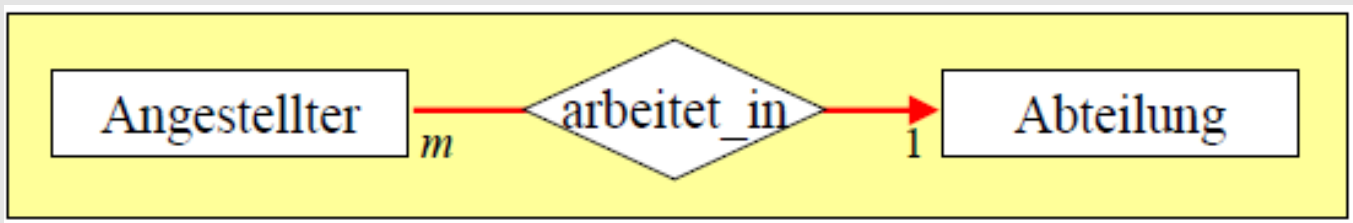
Funktionalität von Relationships

m:1-Beziehung (many-to-one-relationship)

- Charakteristik: Jedes Objekt auf der *m*-Seite steht in Beziehung zu höchstens einem Objekt auf der 1-Seite (i. A. nicht umgekehrt)
- Grafische Notation: Pfeil zur 1-Seite

Beispiel: m:1-Beziehung

Jeder Angestellter arbeitet in höchsten einer Abteilung aber eine Abteilung hat i.A. mehrere Angestellte



Funktionalität von Relationships

m:n-Beziehung (many-to-many-relationship)

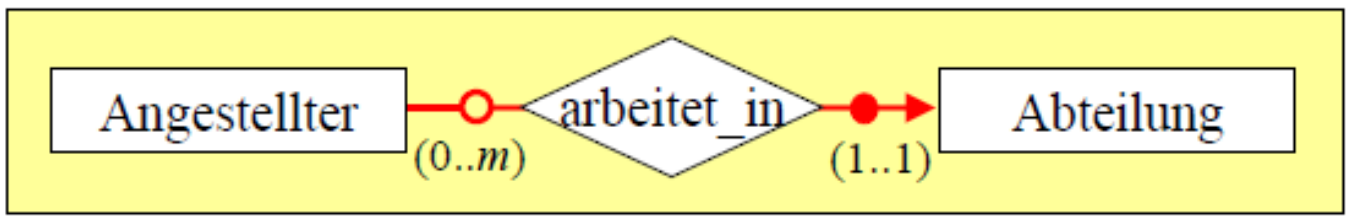
- Charakteristik: Jedes Objekt auf der linken Seite kann zu mehreren Objekten auf der rechten-Seite in Beziehung stehen (d.h. keine Einschränkung)
- Grafische Notation: kein Pfeil

Beispiel: m:1-Beziehung

Mehrere Studenten belegen i.d.R. mehrere Vorlesung

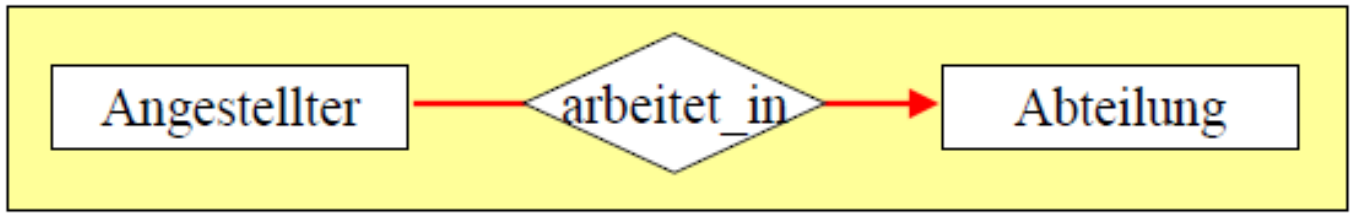
Optionalität

- Das E/R-Modell trifft lediglich Aussagen darüber, ob ein Objekt zu mehreren Objekten in Beziehung stehen darf oder zu max. einem
- Darüber, ob es zu mindestens einem Objekt in Beziehung stehen muss, keine Aussage
- Daher gibt es in der Literatur die Erweiterung der Notation mit:
  - Geschlossener Kreis: Verpflichtend mindestens eins
  - Offener Kreis: Optional
  - Gilt auch für Attribute (z.B. vgl. `NOT NULL`-Constraint)

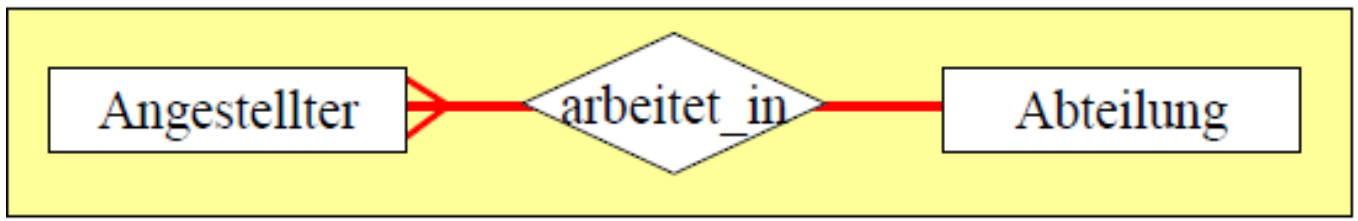


Verschiedene Notationen in der Literatur

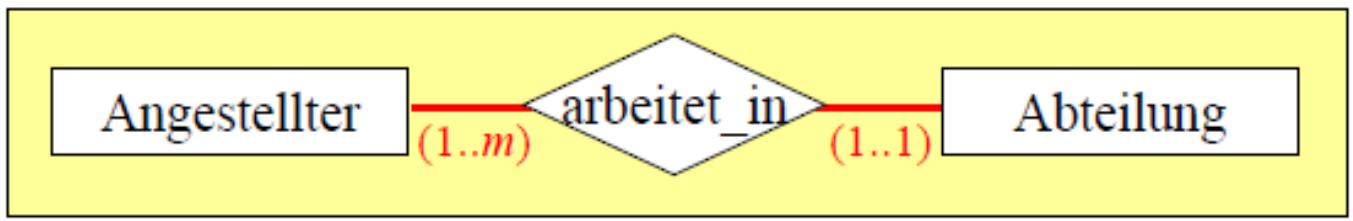
- Pfeil-Notation der Funktionalität



- Krähenfuß-Notation:



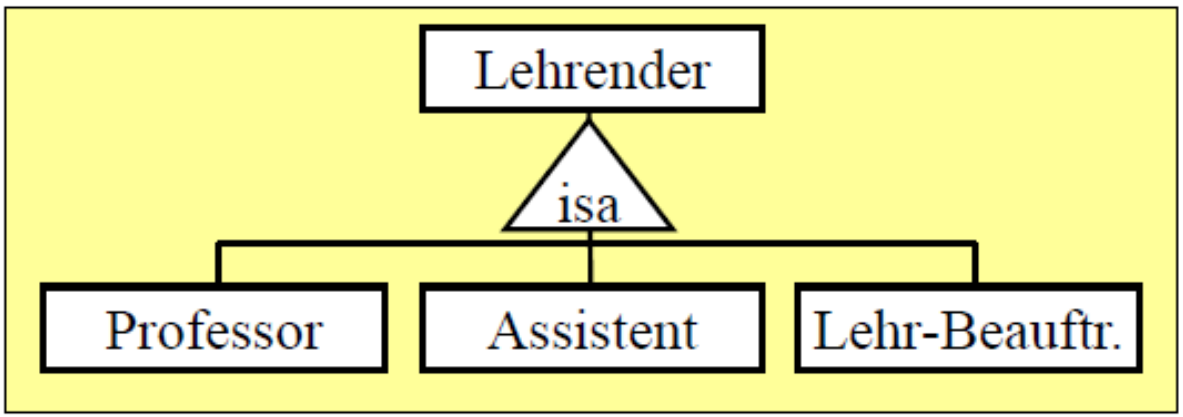
- Kardinalitäts-Notation



... also nicht verwirren lassen ...

Vererbungs-Beziehung

- Das Erweiterte E/R-Modell kennt Vererbungs-Beziehungen (sog. “is-a”-Relationships) für Entities
- Beispiel:



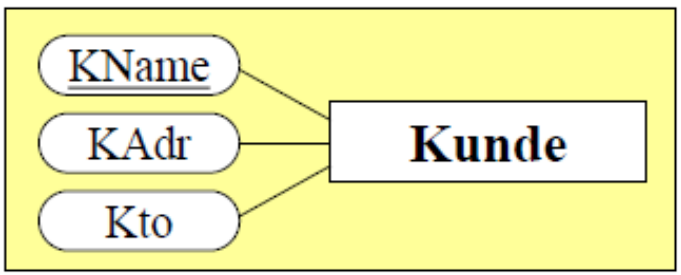
- Charakteristik und Bedeutung
  - Assistent ist ein Lehrender (engl.: “is a”)
  - Vererbung aller Attribute und Beziehungen

Agenda

1. Das Entity/Relationship-Modell
2. Vom E/R-Modell zur Relation

Umsetzung von Entities

- Es gibt relativ einfache (und v.a. klare) Umsetzungsregeln, um einen E/R-Entwurf in ein entsprechendes relationales Schema zu transformieren
- Entities und Attribute:
  - Jedem Entity-Typ wird eine Relation zugeordnet
  - Jedes Attribut des Entity wird ein Attribut der Relation (zusätzliche Attribute können im weiteren Verlauf dazukommen)
  - Der Primärschlüssel des Entity wird Primärschlüssel der Relation



wird zu:

Kunde(**KName**, KAdr, Kto)

Umsetzung von Relationships: Übersicht

- Bei Relationships ist die Umsetzung abhängig von der Funktionalität / Kardinalität:

1:1	Zusätzliche Attribute in bestehenden Relationen (für die beteiligten Entities)
m:1	
m:n	Erzeugung einer zusätzlichen (eigenen) Relation für die Relationship

- Bemerkungen:
  - Die ersten beiden Funktionalitäten sind Spezialfälle der dritten
  - Deshalb ist es immer auch möglich, zusätzliche Relationen einzuführen (wie bei n:m), es ist jedoch nicht erforderlich

Umsetzung von m:1-Relationships

- Eine zusätzliche Tabelle wird nicht angelegt
- Der Primärschlüssel der Relation auf der one-Seite der Relationship kommt in die Relation der many-Seite (Umbenennung bei Namenskonflikten)
- Die neu eingeführten Attribute werden Fremdschlüssel
- Die Primärschlüssel der Relationen ändern sich nicht
- Attribute der Relationship werden ebenfalls in die Relation der many-Seite genommen (kein Fremdschlüssel!)

Umsetzung von m:1-Relationships

Beispiel

Umsetzung einer m:1-Beziehung:

Abteilung (ANr, Bezeichnung, ...)

Mitarbeiter (PNr, Name, Vorname, ..., ANr)

```
create table Mitarbeiter (
        PNr char(3) primary key,
        ...
        ANr char(3) references Abteilung (ANr) ) ;
```

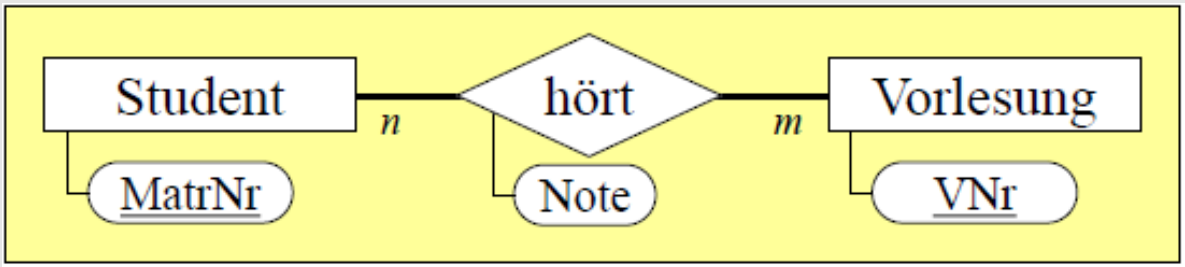
Umsetzung von m:n-Relationships

- Einführung einer zusätzlichen Relation mit dem Namen der Relationship
- Attribute: Die Primärschlüssel-Attribute der Relationen, die den Entities beider Seiten zugeordnet sind
- Diese Attribute sind jeweils Fremdschlüssel
- Zusammen sind diese Attribute Primärschlüssel der neuen Relation
- Attribute der Relationship werden ebenfalls in die neue Relation als "normale" (keine Schlüssel'-)Attribute aufgenommen

Umsetzung von m:n-Relationships

Beispiel

Umsetzung einer m:n-Beziehung:



Student (MatrNr, ...) Vorlesung (VNr, ...)

Hoert (MatrNr, VNr, Note)

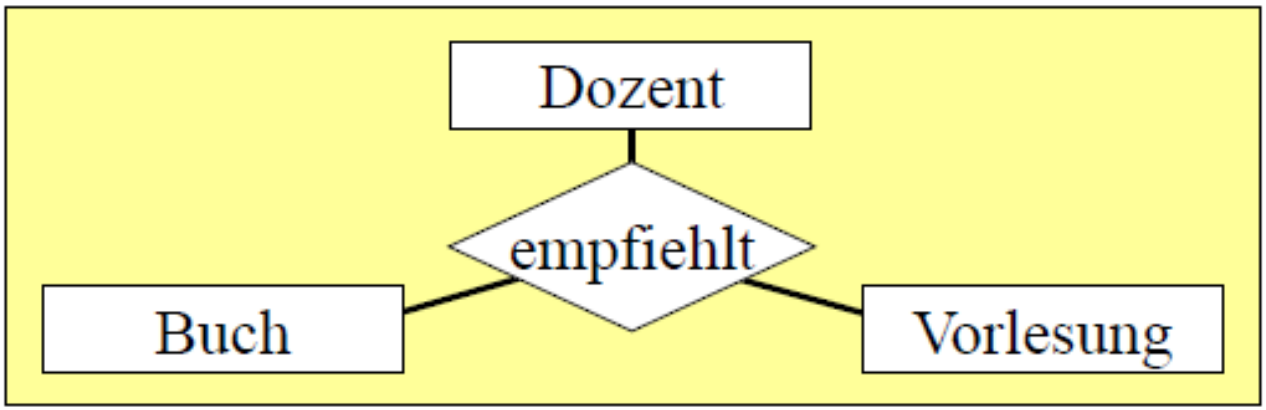
...
primary key (MatrNr, VNr),
foreign key MatrNr **references** Student,
foreign key VNr **references** Vorlesung...

Umsetzung von 1:1-Relationships

- Die beiden Entities werden zu einer Relation zusammengefasst
- Einer der Primärschlüssel der Entities wird Primärschlüssel der Relation (der andere bleibt Schlüsselkandidat)
- Häufig auch Umsetzung wie bei 1:n-Beziehung (insbes. wenn eine Seite optional ist), wobei die Rollen der beteiligten Relationen austauschbar sind
- Attribute der Relationship werden natürlich auch in die gemeinsame Relation übergenommen
- Nochmal der Hinweis: sowohl 1:1- als auch m:1-Beziehungen können natürlich auch mit einer eigenen Relation (wie m:n-Beziehungen) gelöst werden!

Mehrstellige Relationships

- Beispiel:



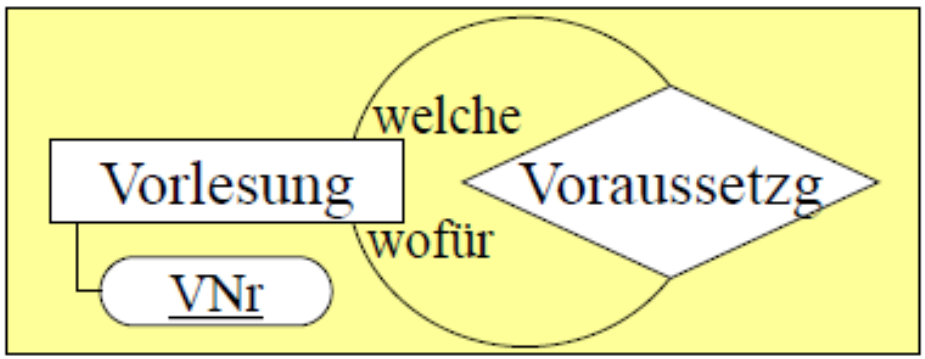
Ausprägung: empfiehlt (Böhm, Heuer&Saake, DBS 1)
empfiehlt (Böhm, Kemper&Eickler, DBS 1)
empfiehlt (Böhm, Bishop, Informatik 1)
empfiehlt (Böhm, Bishop, Programmierkurs)

- Eigene Relation für empfiehlt, falls mehr als eine Funktionalität many ist:
  Dozent(**PNr**, ...) Buch(**ISBN**, ...) Vorlesung(**VNr**, ...)

  empfiehlt(**PNr**, **ISBN**, **VNr**)

Relationships mit Selbsbezug

- Keine Besonderheiten: Vorgehen je nach Funktionalität



Vorlesung(**VNr**, ...) Voraussetzg(**Welche**, **Wofuer**,)

```
...
FOREIGN KEY Welche REFERENCES Vorlesung(VNr),
FOREIGN KEY Wofuer REFERENCES Vorlesung(VNr)
```

Vererbungs-Beziehungen

- Umsetzung der ISA-Beziehung meist wie bei m:1-Beziehungen (Eltern-Entity ist 1-Seite)
- Dadurch werden die Attribute der Eltern-Entities nicht redundant in den Kind-Entities aufgezählt
- Aber auch keine implizite Vererbung von Attributen (sind aber über Fremdschlüsselbeziehung "abrufbar"), d.h. die Semantik der Vererbung geht verloren
- Alternative: Attribute und Relationships vom Eltern-Entity explizit in Kind-Entity übernehmen (aber auch hier geht die Semantik natürlich verloren)

E/R versus UML

- Wegen Unübersichtlichkeit und Einschränkungen Verdrängung der E/R-Diagramme durch Klassendiagramme von UML (Unified Modelling Language — Klassendiagramme z.B. für statischen Entwurf beim oo SW-Engineering)
- Unterschiede in der Notation (z.B. Attribute werden direkt im Entity-Kasten notiert, ...)
- Feinere Semantik der Beziehungen in UML geht bei der Übertragung ins relationale Modell auch verloren
- Methoden (in der oo Programmierung wesentlich) werden im DB-Kontext nicht benötigt