3. Datenbankmodelle

Datenmodelle allgemein

- Konzepte zur Erfassung der Informtionsstruktur einer Anwendung
- Informationsstruktur:
  1. statische Eigenschaften
     (a) Objekte
     (b) Beziehungen
  2. dynamische Eigenschaften
     (a) Operationen
     (b) Beziehungen
  3. Integritätsbedingungen
     (a) an Objekte
     (b) an Operationen
- Beispiele von Datenmodellen:
  – Programmiersprachen: Typsysteme
  – Graphiksysteme: Repräsentationmodelle
  – Expertensysteme: Wissensrepräsentationsmethoden
  – Datenbanksysteme: Datenbankmodelle

Datenbankmodelle

- enthalten/beschreiben 1.a, 1.b, 3.a; 2.a (nur implizite Standardoperationen)

- Modelle:

	Daten			Parallele: Algorithm.	
↑abstrakter	ER-Modell			Flußdiagr.	
↓konkreter	hierarch. Modell	Netzwerk-Modell	Relat. Modell	Modula-Progr.	C-Progr.

objektorientierte DBM; nur eingeschränkt aus ER-Modell ableitbar

3.1 Entity-Relationship-Modell

- nach P.P. Chen, 1976

- Grundkonzepte:

  - Entity: zu modellierende Informationseinheit, 'Gegenstand' der realen Welt; Bsp.: ein Professor, ein Student, eine Prüfung

  - Relationship: Beziehung zwischen Entities; Bsp.: *wird gehalten von* ist Beziehung zwischen Vorlesung und Professor

  - Attribut: Eigenschaft von Entities oder Beziehungen; Bsp.: Matrikelnr. bei Student, Name bei Professor

- weitere Konzepte:

  - Entity-Mengen: Zusammenfassung ähnlicher 'Gegenstände' der realen Welt; Bsp.: alle Professoren

  - Entity-Typ: Strukturbeschreibung einer Entity-Menge

  - Domäne: Wertebereich von Attributen; Bsp.: Note hat Domäne 1.0 ... 6.0, Basistypen wie `INTEGER`, `STRING`,...

  - Schlüssel: Menge von Attributen, die ein Entity eindeutig identifizieren; Bsp.: {Matrikelnr.} ist Schlüssel für Studenten, aber kein Schlüssel für Prüfungen, {Matrikelnr.,Fach,Datum} Schlüssel für Prüfungen

Entity-Relationship-Modell

- weitere Konzepte:
  - is-a-Relationship: Spezialisierungs-/Generalisierungsbeziehung zwischen Entities, Attribute des allgem. Typs gelten auch für spezielleren; Bsp.: Professoren is-a Angestelle
  - funktionale/nichtfunktionale Relationships: Klassifizierung danach, wieviel Entities der einen Menge mit Entities der anderen in Beziehung stehen können:
    one-to-one: pro Entity existiert höchstens ein zugeordnetes in der anderen Entity-Menge (Bijektion); Bsp.: Professoren, Lehrstühle
    many-to-one: ein Entity der Menge M2 steht in Beziehung mit einer (evtl. leeren) Menge von Entities der Entity-Menge M1, jedes Entity aus M1 aber nur mit höchstens einem Entity aus M2 (Funktion); Bsp.: Vorlesungen, Professoren
    many-to-many: bel. viele Entities aus M1 stehen mit bel. vielen Entities aus M2 in Beziehung und umgekehr (Relation); Bsp.: Vorlesung, Studenten
  - mehrstellige Relationships: mehr als zwei Entity-Mengen stehen in Beziehung zueinander; Bsp.: Professor hält Vorlesung und empfiehlt dazu Bücher

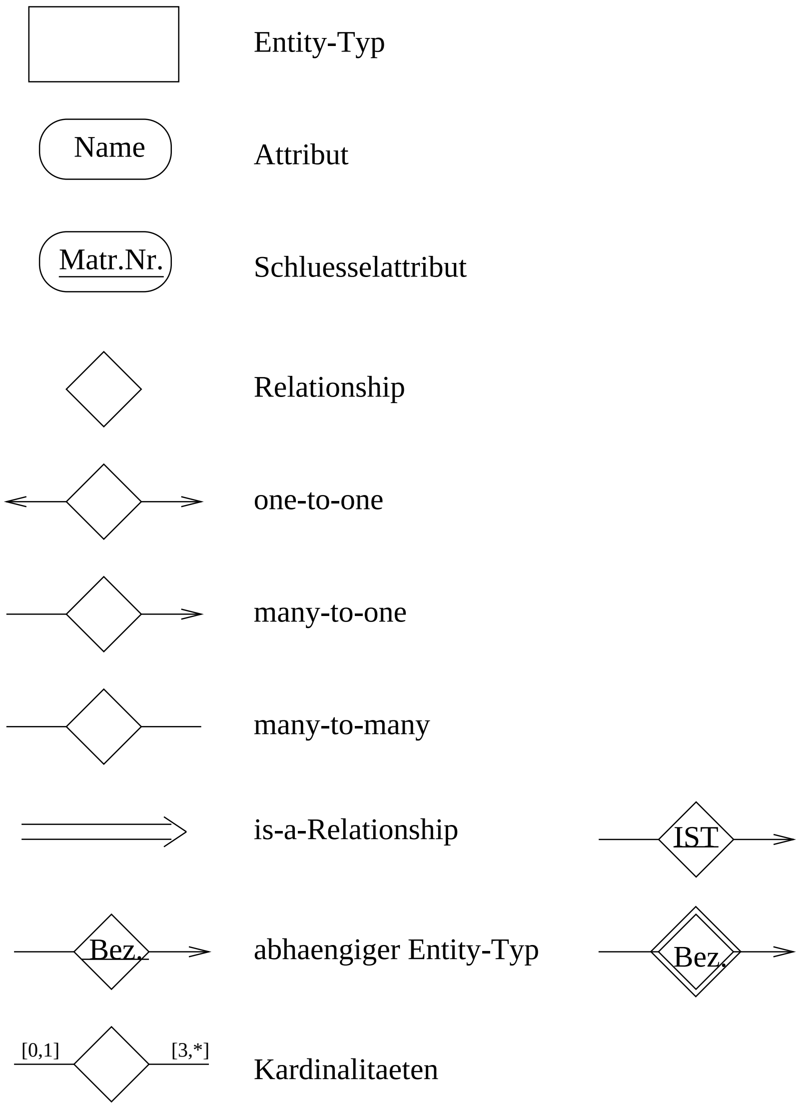
Entity-Relationship-Modell

- weitere Konzepte:
  - Kardinalitäten: genauere Angabe zu beteiligten Entities bei Beziehungen, Angabe jeweils eines minimalen und maximalen Wertes, * steht für unbegrenzte Anzahl; Bsp.: Studenten hören keine/eine/bel. viele Vorlesungen, jede Vorlesung wird von 3 oder mehr Studenten gehört (*hört(Studenten[0,*], Vorlesung[3,*])*)

    Quicky: Welche Kardinalitäten gehören zu one-to-one-Beziehung, many-to-one-Beziehung?
  - abhängige Entity-Typen (weak entity): Instanzen eines Entity-Typs hängen von Existenz eines anderen Entity-Typs ab, funktionale Beziehung zwischen den Entity-Mengen stellt Schlüssel des abhängigen Typs; Bsp.: Entity-Typ BuchExemplar (mit Nummer, Ausleiher, Rückgabe) über Beziehung *gehört zu* abhängig von Entity-Typ Buch (mit ISBN, Titel,..)

Entity-Relationship-Modell

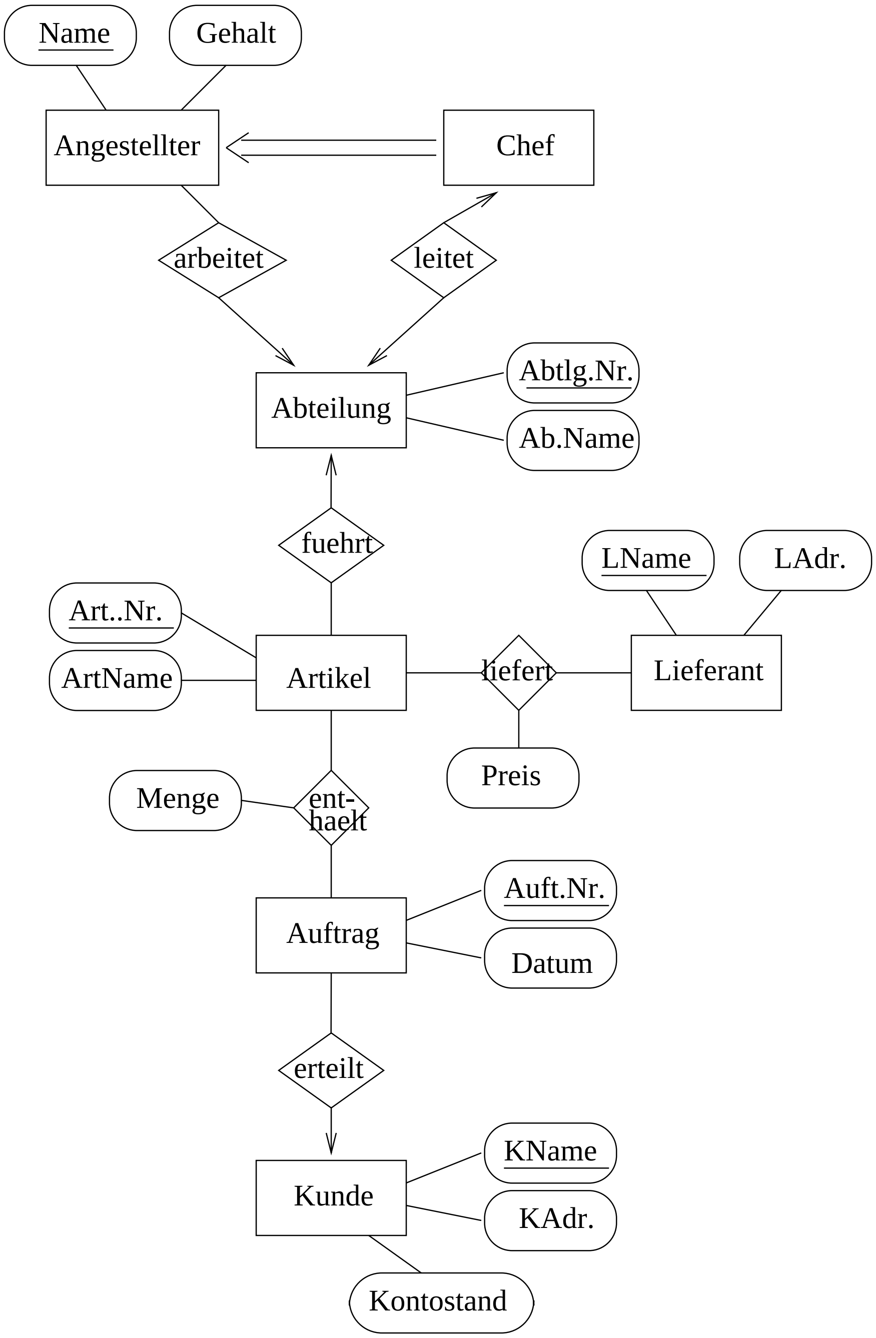
- graphische Notation:



Entity-Typ
Name
Attribut
Matr.Nr.
Schluesselattribut
Relationship
one-to-one
many-to-one
many-to-many
is-a-Relationship
IST
Bez.
abhaengiger Entity-Typ
Bez.
[0,1]
[3,*]
Kardinalitaeten

Entity-Relationship-Modell

- Beispiel: Warenhaus



Entity-Relationship-Modell

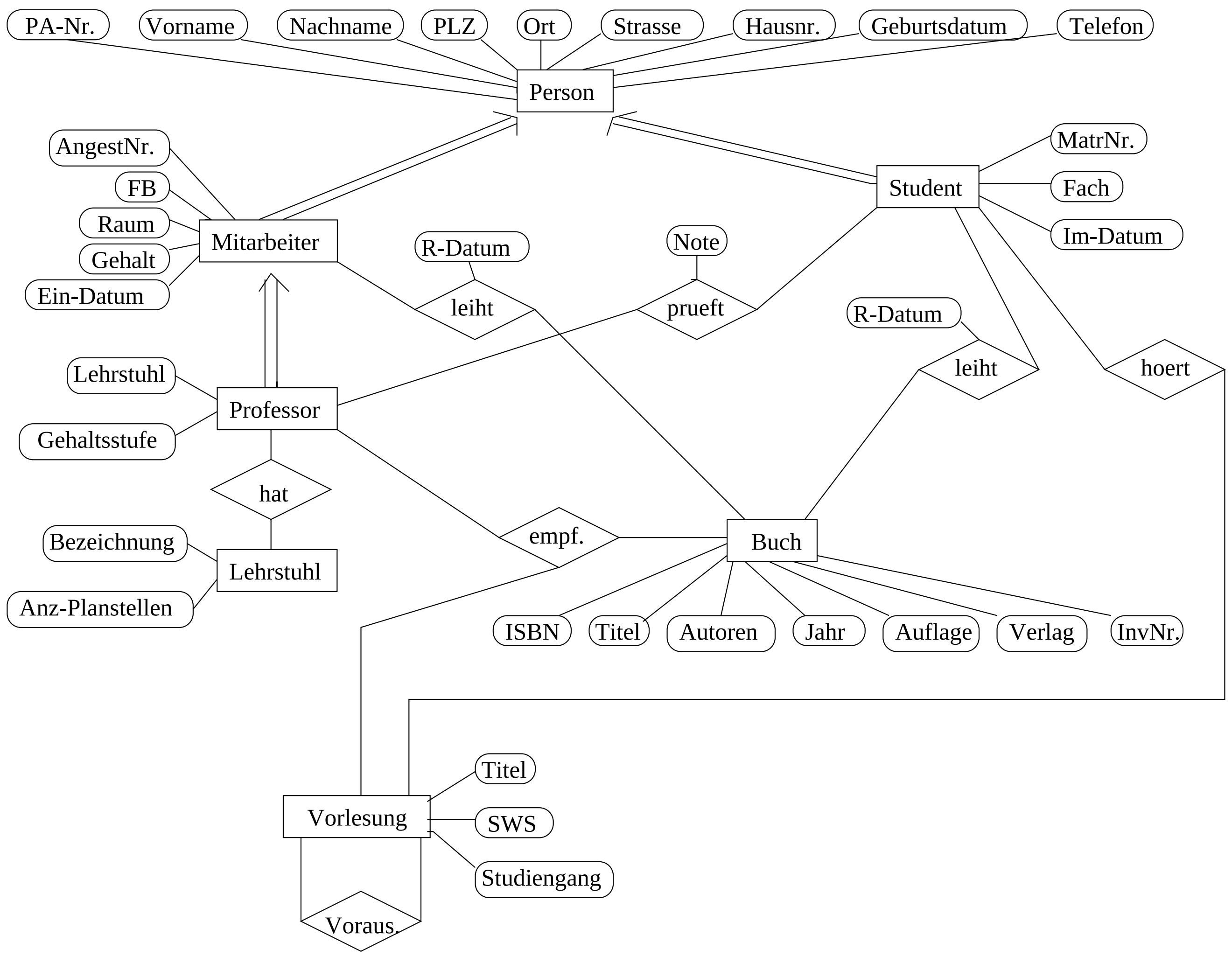
Quicky: Modellierung folgender Universitätsanwendung im ER-Modell?

- folgende Personen spielen eine Rolle: Mitarbeiter, Professoren, Studenten
- Personen haben als Eigenschaften eine PA-Nummer, Vor- und Nachname, Adresse bestehend aus PLZ, Ort, Straße und Hausnummer, ein Geburtsdatum und eine Reihe von Telefonnummern
- Mitarbeiter besitzen eine Angestelltennummer, einen Fachbereich, Raum, ein Einstellungsdatum und ein Gehalt
- Professoren sind Mitarbeiter, die einen Lehrstuhl innehaben und eine Gehaltsstufenangabe (C4, C3,..)
- Studenten haben eine Matrikelnr., ein Studienfach und ein Immatrikulationsdatum
- Lehrstühle kennzeichnen vertretene Fachrichtungen; notwendige Informationen dazu sind die Lehrstuhlbezeichnung und die Anzahl an Planstellen
- Vorlesungen haben einen Titel, eine Wochenstundenzahl, einen zugeordneten Studiengang und andere Vorlesungen als Voraussetzungen
- Bücher haben eine ISBN, einen Titel, eine Menge von Autoren, ein Erscheinungsjahr, eine Auflagennr., eine Verlagsangabe und eine Inventarnummer
- sowohl Mitarbeiter als auch Studenten dürfen Bücher ausleihen; zu jeder Ausleihe gibt es ein Rückgabedatum
- Professoren empfehlen bestimmte Bücher in ihren Vorlesungen

- einige Professoren sind Inhaber von Lehrstühlen
- Professoren lesen in bestimmten Semestern bestimmte Vorlesungen
- Studenten hören in bestimmten Semestern einige Vorlesungen
- einige Vorlesungen haben andere als Voraussetzung
- Professoren prüfen ihre Studenten in bestimmten Vorlesungen; für jede Prüfung wird dabei eine Note vergeben

Entity-Relationship-Modell

Musterlösung:

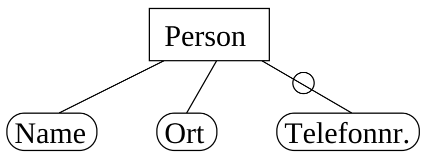


3.2.Erweiterungen

- es ex. div. erweiterte ER-Modelle (EER-Modelle)

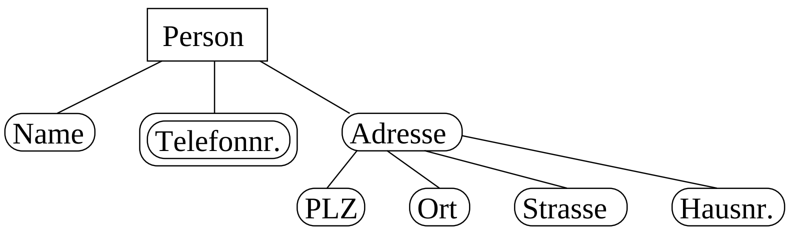
- mögliche Erweiterungen:

  – Optionalität von Attributen/Beziehungen: Angabe, ob Entity an Beziehung teilnehmen *muß* bzw. Attribut einen definierten Wert annehmen *muß*; Schreibweise (z.B.) `1c:m`, ohne `c`: Entity muß in Beziehung auftauchen; graphisch (z.B.) mittels transparentem Kreis an Verbindungslinie:



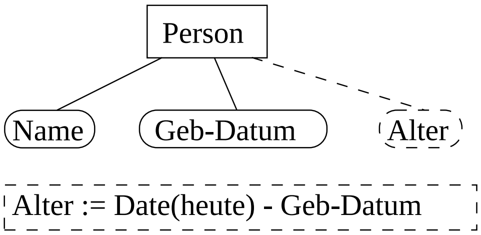
  – neben Standardtypen (`integer, string,...`) für Domänen auch benutzerdefinierte Typen, z.B. `date, time, image,..`

  – Typkonstruktoren bei Attributen, z.B. `TUPLE, LIST, SET, BAG`

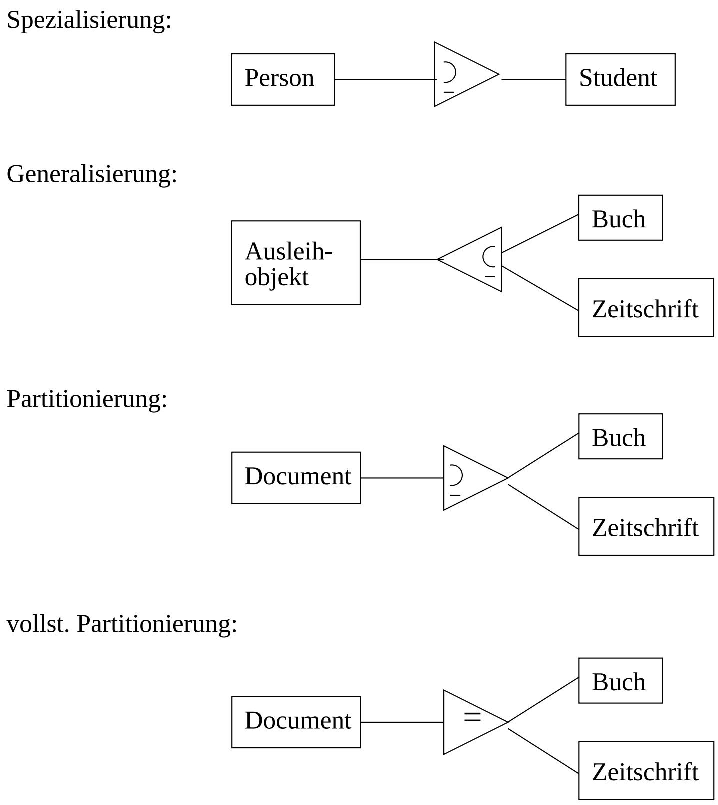


Erweiterungen

- abgeleitete (berechnete; derived) Attribute: Wert wird nicht abgespeichert, sondern mittels Anfrage berechnet



- is-a-Beziehung: Unterscheidung nach Spezialisierung, Generalisierung, Partitionierung
  Spezialisierung: speziellerer Kontext
  Generalisierung: Vereinigungsmenge mehrerer (spezieller) Entity-Mengen zu allgemeinerem Kontext
  Partitionierung: Entity-Typ wird in mehrere *disjunkte* Entity-Typen spezialisiert
  vollständige Partitionierung (Überdeckung): Vereinigung der disjunkten, speziellen Entity-Mengen ergibt genau Instanzen der allgemeineren Entity-Menge



- Assoziationen (Mengenbildung) und Aggregationen (Tupelbildung) zwischen Entities:

  Tupelbildung: Entity setzt sich aus anderem Entities unterschiedlicher Typen zusammen; Bsp.: Auto besteht aus Karosserie, Motor, Kabelbaum, etc.

  Mengenbildung: Entity wird durch Menge von Entities eines anderen Typs gebildet; Bsp.: Menge von Flächen bilden einen Würfel

  zu beachten: Existenzabhängigkeiten

Erweiterungen

- Beziehungen höheren Typs (selten!): Spez./Gen. bei Beziehungen; Bsp.: Semesterausleihe, Kurzausleihe sind Spezialisierungen von Ausleihe

Quicky: was wäre bei Universitätsanwendung in EER-Modell anders?