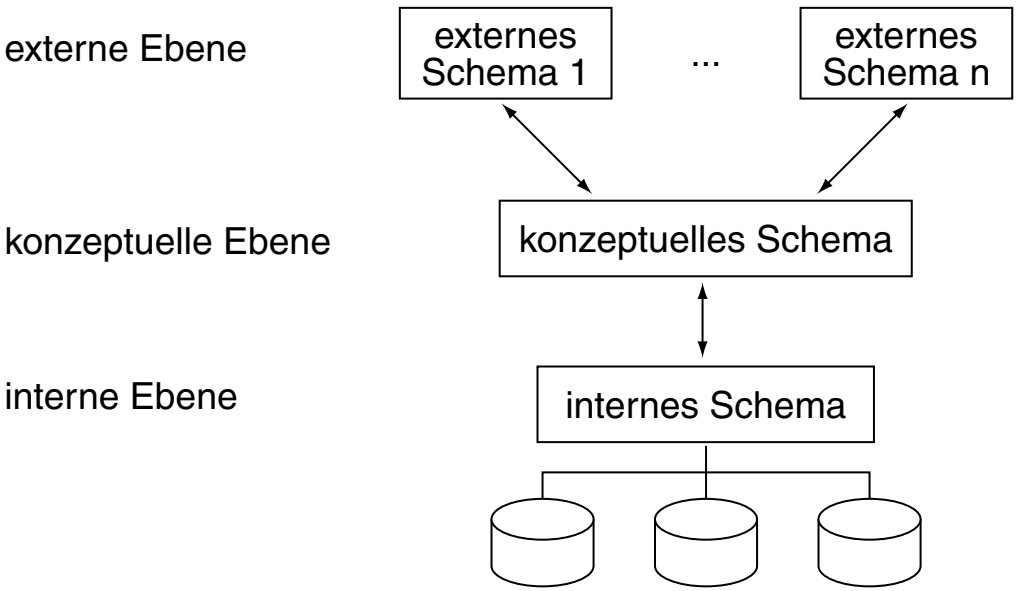
Kapitel DB:II

II. Datenbankentwurf und Datenbankmodelle

- Entwurfsprozess
- Datenbankmodelle

Entwurfsprozess

ANSI/SPARC-Schema-Architektur



"Usually, a representational [= implementational, logical] *data model is used to describe the conceptual schema when a database system is implemented. This implementation conceptual schema is often based on a conceptual schema design in a high-level data model."*

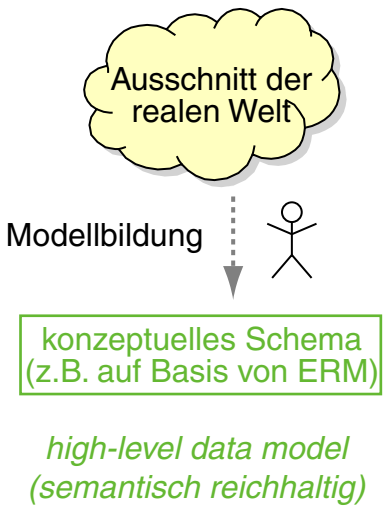
[p.30 Elmasri/Navathe 2010]

Bemerkungen zu [p.30 Elmasri/Navathe 2010]:

- In einem Datenbanksystem ist das konzeptuelle Schema oft durch ein implementierungsnahes Datenmodell wie dem relationalen Datenmodell definiert.
- Im Entwurfsprozess entsteht dieses implementierungsnahe Datenmodell auf Grundlage eines semantisch reicheren Modells wie dem ER-Modell oder UML.
- Idealerweise sollte eine Datenbankbeschreibung direkt auf dem semantisch reicheren Modell beruhen, ohne dass eine (manuelle) Transformation in ein implementierungsnahes Datenmodell erfolgen muss.

Entwurfsprozess

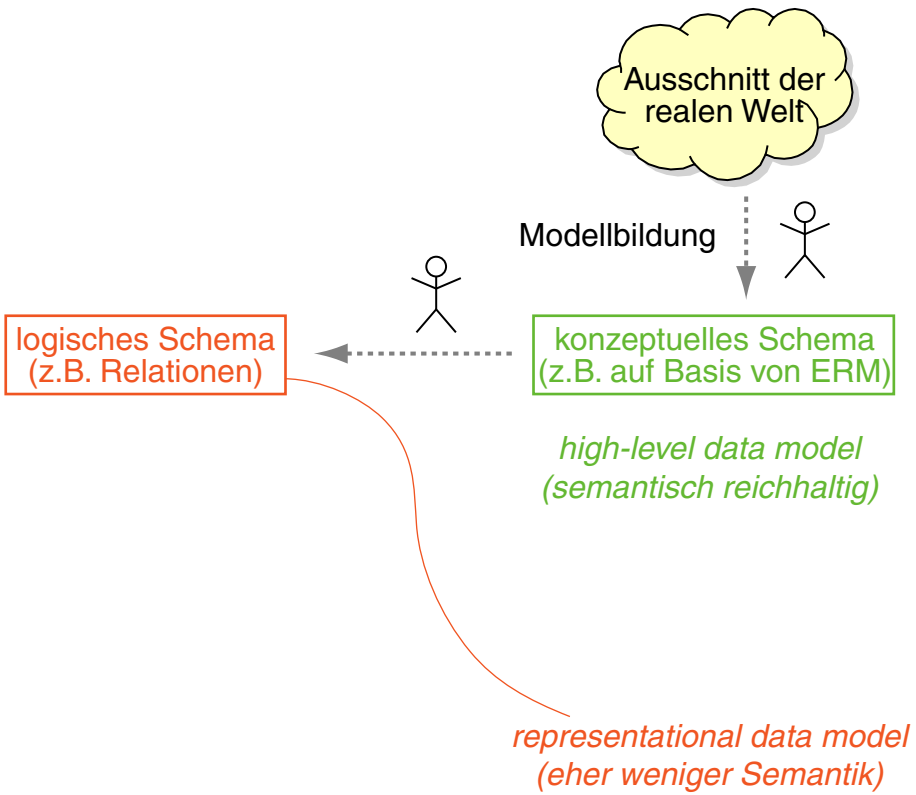
ANSI/SPARC-Schema-Architektur: Entwurfspraxis



high-level data model (semantisch reichhaltig)

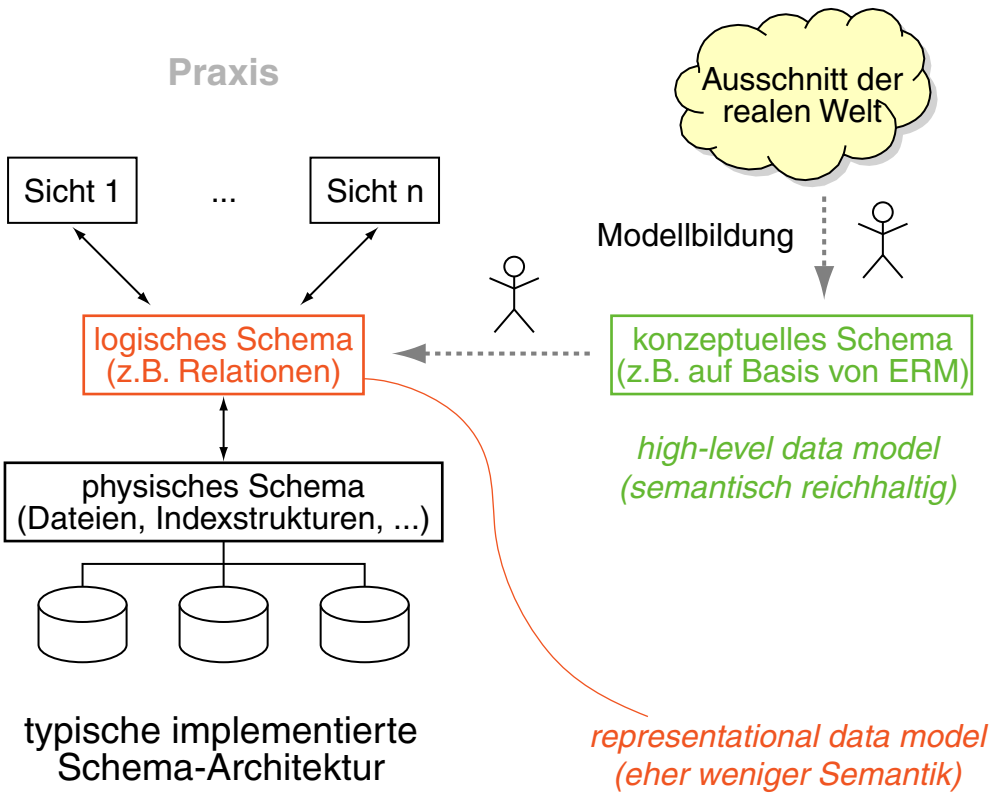
Entwurfsprozess

ANSI/SPARC-Schema-Architektur: Entwurfspraxis



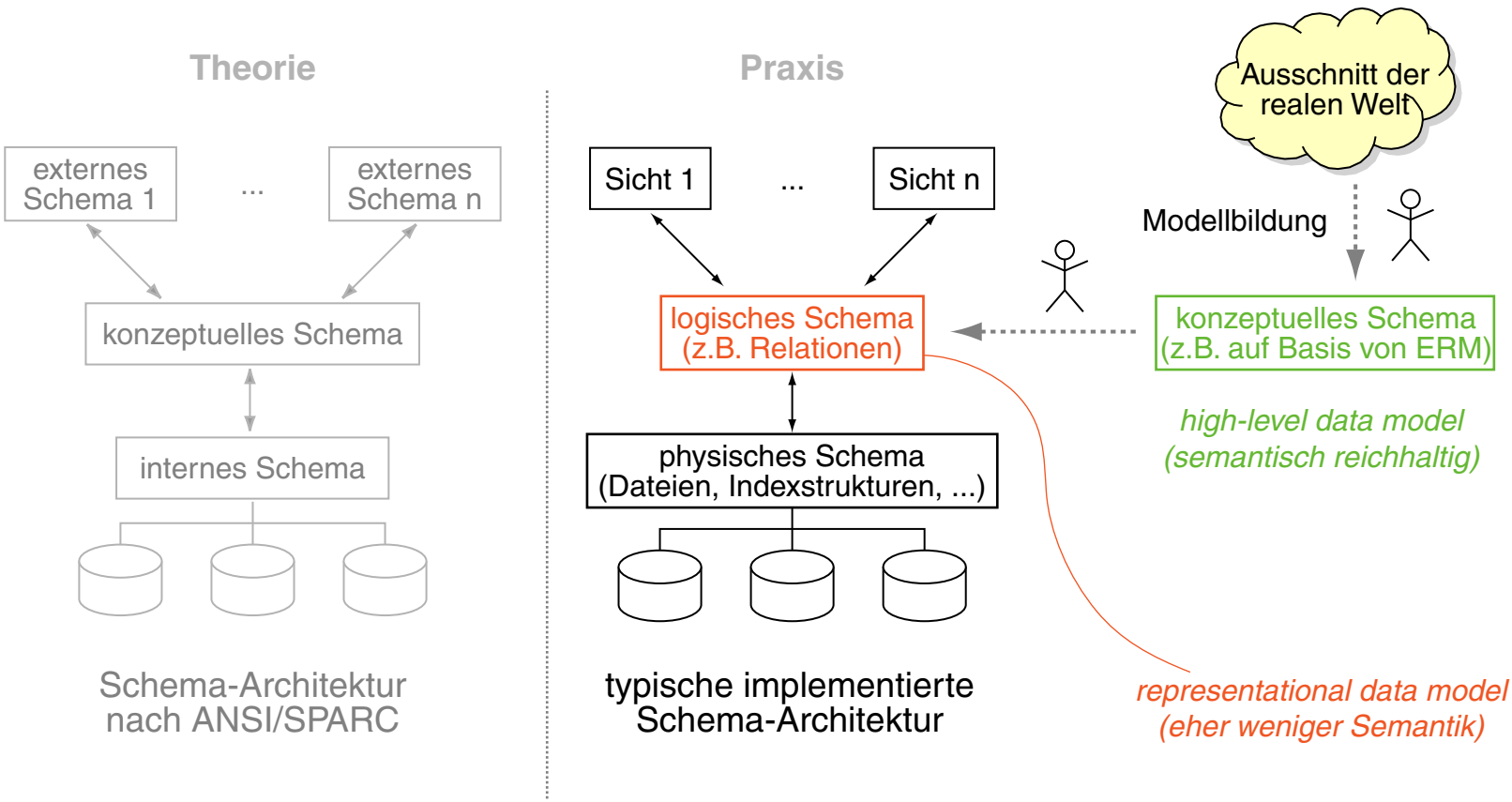
Entwurfsprozess

ANSI/SPARC-Schema-Architektur: Entwurfspraxis



Entwurfsprozess

ANSI/SPARC-Schema-Architektur: Entwurfspraxis



Entwurfsprozess

Die zwei zentralen Anforderungen an den Entwurfsprozess sind:

1. Informationserhaltung
2. Konsistenzerhaltung

Weitere, zum Teil informelle Gütekriterien:

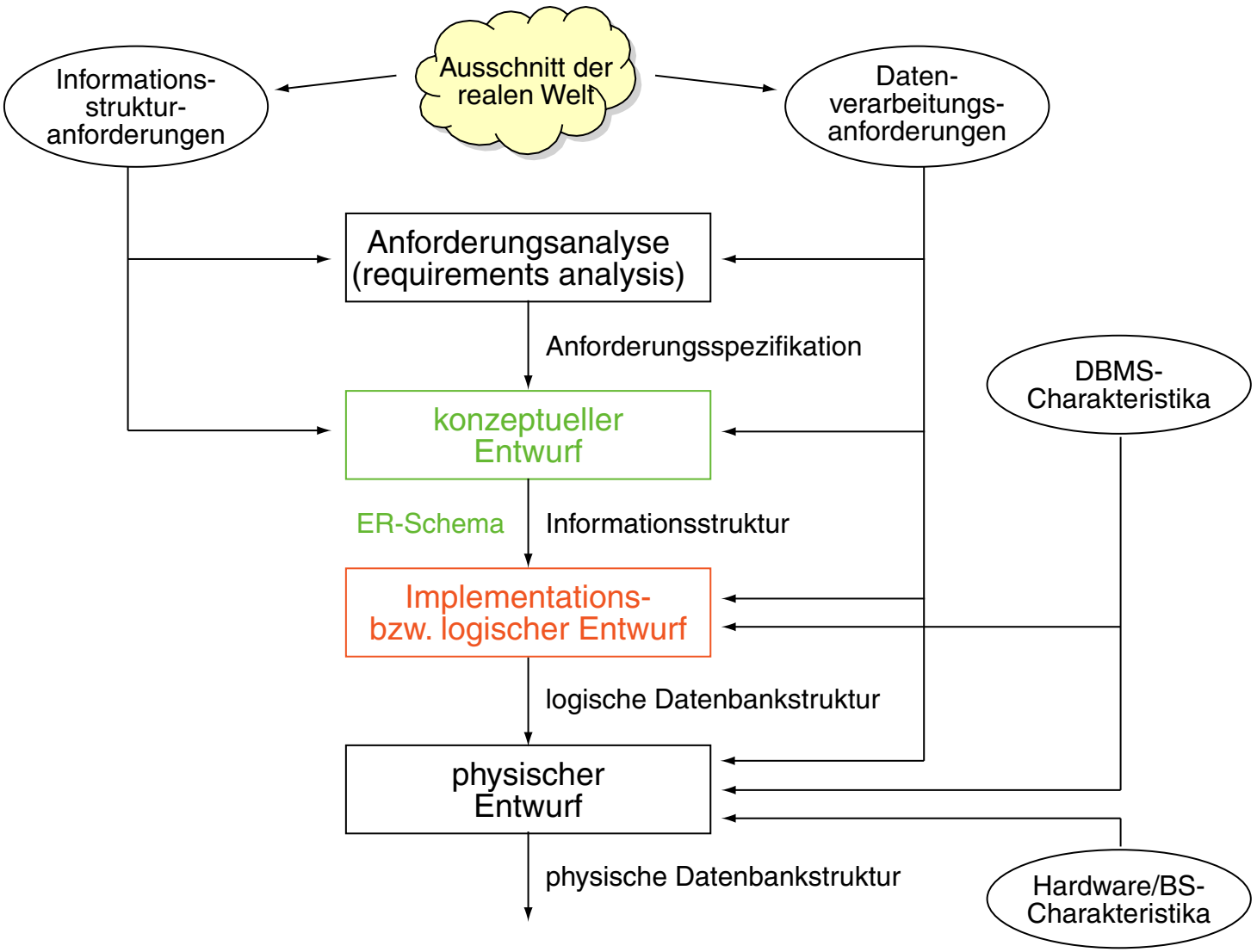
- Redundanzfreiheit
- Vollständigkeit bzgl. der Anforderungsanalyse
- Konsistenz der Beschreibung
- Ausdrucksstärke und Verständlichkeit des benutzten Formalismus
- formale Semantik
- Lesbarkeit der Dokumente
- Unterstützung von Erweiterbarkeit, Modularisierung, Wiederverwendbarkeit, Werkzeugeinsatz

Bemerkungen:

- Informations- und Konsistenzerhaltung beziehen sich auf die Transformation ausgehend von der Anforderungsanalyse hin zur Implementierung:
  1. Informationserhaltung fordert, dass die gewünschten Informationen des Weltausschnitts auf jeder Abstraktionsstufe darstellbar sind.
  2. Konsistenzerhaltung fordert, dass die gewünschten Regeln und Einschränkungen *(Constraints)* des Weltausschnitts auf jeder Abstraktionsstufe eingehalten werden.

Entwurfsprozess

Phasenmodell des Datenbankentwurfs



Bemerkungen:

- Verschiedene Phasen dieses Modells lassen sich weiter aufschlüsseln. Beispielsweise gliedern [Heuer/Saake 2013] den konzeptuellen Entwurf noch in einen Sichtenentwurf, eine Sichtenanalyse und eine Sichtenintegration.
- Sollen die Daten auf mehreren Rechnern *verteilt* vorliegen, muss Art und Weise der verteilten Speicherung festgelegt werden. Dies geschieht im sogenannten Verteilungsentwurf, einer Entwurfsphase zwischen konzeptuellem und logischen Entwurf.

Entwurfsprozess

Phasenmodell: Anforderungsanalyse

Ziel:

Sammlung des Informationsbedarfs in den verschiedenen Abteilungen bzw. Benutzergruppen.

Ergebnis:

- informelle Beschreibung des Problems bzw. der Aufgabenstellung (Use-Cases, Texte, tabellarische Aufstellungen, Formblätter, Maskenentwürfe, etc.)
- Trennen der Informationen über Daten (Datenanalyse) von den Informationen über Funktionen (Funktionsanalyse)

Bemerkungen:

- Im klassischen Datenbankentwurf wird nur die Datenanalyse einschließlich ihrer Folgeschritte behandelt; die Funktionsanalyse wird weitgehend ignoriert. Mittelfristig wird sich eine integrierte objektorientierte Betrachtung von Daten und Funktionen durchsetzen.

Entwurfsprozess

Phasenmodell: konzeptueller Entwurf

Ziel:

Formale Beschreibung der Aufgabenstellung (der zu speichernden Daten) in einem abstrakten (semantisch reichhaltigen, „high-level“) Datenmodell. Hierfür wird das Entity-Relationship-Modell am häufigsten eingesetzt.

Vorgehensweise:

1. Modellierung von Sichten (z.B. einer Abteilung oder Benutzergruppe)
2. Analyse der vorliegenden Sichten hinsichtlich von Konflikten:
   - Namenskonflikte: Homonyme, Synonyme
   - Typkonflikte: verschiedene Strukturen für das gleiche Konzept
   - Wertebereichskonflikte: keine Vereinheitlichung möglich
   - Bedingungskonflikte: verschiedene Sichten fordern eigene Integritätsbedingungen
   - Modellierungskonflikte: gleicher Sachverhalt ist unterschiedlich modelliert
3. Integration der Sichten in ein Gesamtschema bzw. ER-Diagramm

Bemerkungen:

- Ein Homonym ist ein Begriff, der für mehrere Konzepte steht. Beispiel: Jaguar
- Synonyme sind verschiedene Begriffe für dasselbe Konzept. Beispiel: {Haus, Gebäude}
- Beispiel für einen Typkonflikt: ein Patient hat aus Sicht der Krankenkasse andere Eigenschaften als aus Sicht des Arztes.
- Beispiele für einen Bedingungskonflikt: verschiedene Schlüssel wurden (von verschiedenen Benutzergruppen) für dieselbe Menge von Objekten vorgesehen.

Entwurfsprozess

Phasenmodell: logischer Entwurf

Ziel:

Umsetzung des konzeptuellen Entwurfs in das Datenmodell des Realisierungs-DBMS, zur Zeit meist das relationale Modell.

Vorgehensweise:

1. teilweise automatische Transformation des konzeptuellen Schemas, z.B. des Entity-Relationship-Modells in das relationale Modell
2. Verbesserung des relationalen Schemas anhand von Gütekriterien durch entsprechende Normalisierungsalgorithmen. Stichwort: Normalformen

Ergebnis:

Ein logisches (relationales) Datenbankschema, das Datenredundanzen „weitgehend“ vermeidet und die Konsistenzbedingungen des Entity-Relationship-Modells „weitgehend“ erhält.

Entwurfsprozess

Phasenmodell: Verteilungsentwurf

Ziel:

Festlegung von Art und Weise einer verteilten Speicherung.

Beispiel:

`Kunde(KdNr, Name, Adresse, PLZ, Kontostand)`

- horizontale Verteilung:

  `Kunde_1(KdNr, Name, Adresse, PLZ, Kontostand)`

  mit PLZ $\leq$ 50 000

  `Kunde_2(KdNr, Name, Adresse, PLZ, Kontostand)`

  mit PLZ $>$ 50 000

- vertikale Verteilung:

  `Kunde_adr(KdNr, Name, Adresse, PLZ)`

  `Kunde_kto(KdNr, Kontostand)`

  Zusammenhang kann über das Attribut `KdNr` hergestellt werden.

Entwurfsprozess

Phasenmodell: physischer Entwurf

Ziel:

Effizienzsteigerung ohne die logische Struktur der Daten zu verändern.

Konzepte:

„Tuning“ der Abbildung der Relationen auf den Sekundärspeicher:

- Definition von Indexstrukturen, die direkten (assoziativen) Zugriff auf alle Tupel einer Relation mit bestimmten Attributwerten erlauben.
- Clusteranalyse zur Gruppierung von Daten im Sekundärspeicher, so dass zusammen benötigte Daten auf denselben Seiten liegen. Typisch insbesondere bei objektorientierten DBMS.

Bemerkungen [Schuerr 2001]:

- Kritik an dieser Vorgehensweise aus Sicht der Softwaretechnik:
  - es handelt sich um das überholte Wasserfallmodell der Softwaretechnik
  - Rückgriffe von einer Phase zu vorgehenden Phasen fehlen
  - Testaktivitäten sind nicht explizit aufgeführt
  - Wartung mit Aktivitäten aus allen Phasen ist unstrukturiert
  - inkrementelle bzw. schrittweise Realisierung eines DBS wird nicht unterstützt
  - Modellierung der Funktionen bleibt nahezu unberücksichtigt
  - Verzahnung mit Entwicklung sonstiger Teile eines Informationssystems fehlen
- Bessere Vorgehensweise:
  - objektorientierter Entwurf für das gesamte Informationssystem
  - Abbildung von Klassendiagrammen (statt ER-Diagrammen) auf Relationen
  - Verwendung moderner(er) Vorgehensmodelle der Softwaretechnik

Datenbankmodelle

Definition 1 (Datenmodell)

Datenmodelle dienen zur Erfassung und Darstellung der Informationsstruktur für eine Anwendung oder einen Anwendungsbereich.

Datenbankmodelle

Definition 1 (Datenmodell)

Datenmodelle dienen zur Erfassung und Darstellung der Informationsstruktur für eine Anwendung oder einen Anwendungsbereich.

Beispiele:

- Typsysteme in Programmiersprachen
- Formalismen zur Wissensrepräsentation in Expertensystemen (Frames, Regeln, logische Formeln)
- Repräsentationsmodelle in Graphiksystemen (BRep, CSG)
- SPO-Tripel im Resource Description Framework des Semantic Web
- Datenbankmodelle in Datenbanksystemen (hierarchisches Modell, Relationenmodell, Entity-Relationship-Modell)

Datenmodell für Datenbanksysteme = Datenbankmodell

Datenbankmodelle

Definition 2 (Datenbankmodell, Datenbankschema [Heuer/Saake 2013]**)**

Ein Datenbankmodell ist ein System von Konzepten zur Beschreibung von Datenbanken. Es legt Syntax und Semantik von Datenbankbeschreibungen für ein Datenbanksystem fest.

Eine konkrete Datenbankbeschreibung wird Datenbankschema genannt.

Datenbankmodelle

Definition 2 (Datenbankmodell, Datenbankschema [Heuer/Saake 2013]**)**

Ein Datenbankmodell ist ein System von Konzepten zur Beschreibung von Datenbanken. Es legt Syntax und Semantik von Datenbankbeschreibungen für ein Datenbanksystem fest.

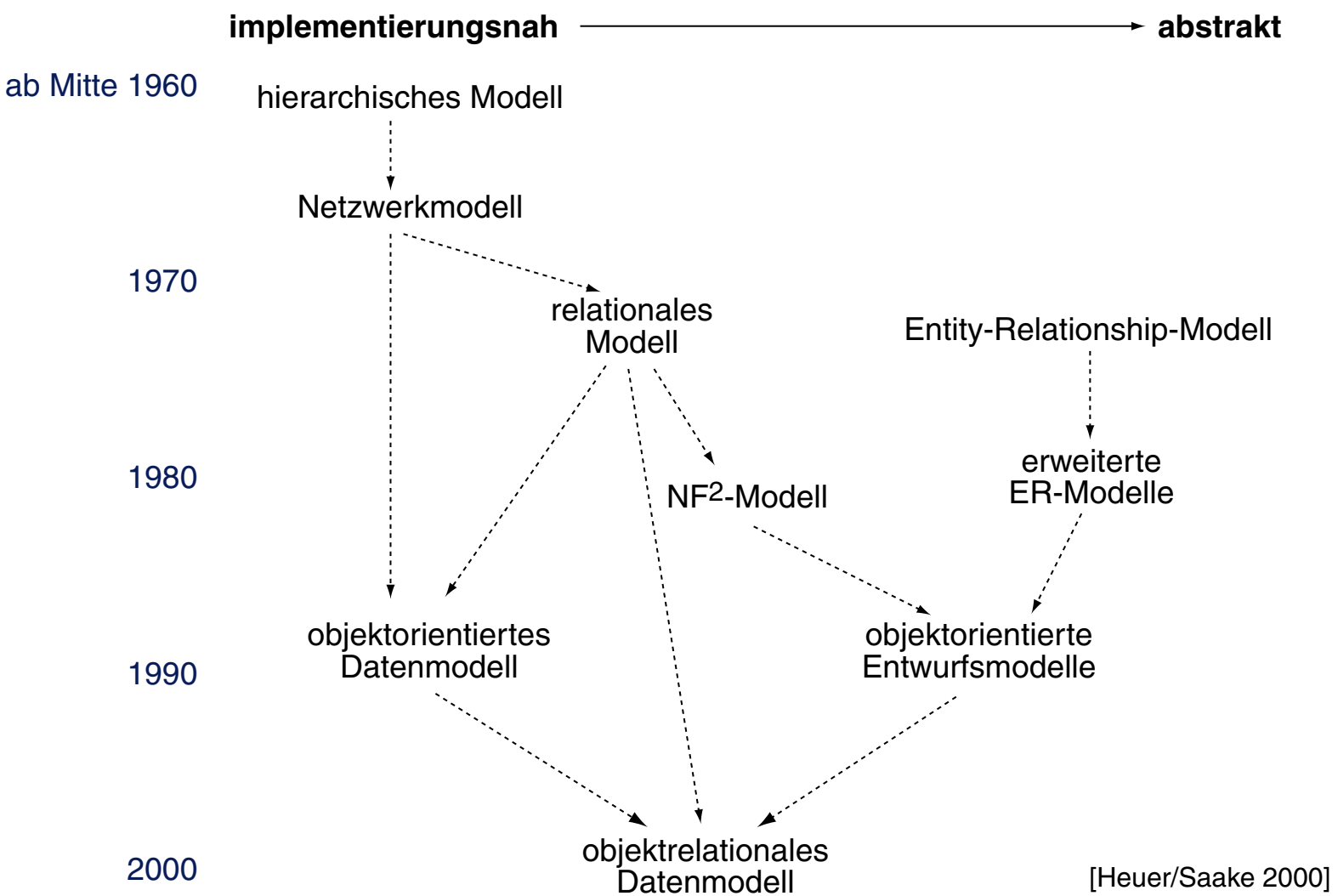
Eine konkrete Datenbankbeschreibung wird Datenbankschema genannt.

Gegenüberstellung zu Programmiersprachen:

Datenbank	**Programmiersprache**
Datenbankmodell *Relation, Attribut, ...*	Typsystem *class, int, String, ...*
Datenbankschema `Kunde(KdNr, Name, ...)`	Variablendeklaration `class Kunde{int KdNr; String Name;}`
Datenbank, DB `(2305, "Meier", ...)`	Werte `2305, "Meier"`
Datenbankmanagementsystem DBMS	Entwicklungs- und Laufzeitumgebung
Datenbanksystem DBS = DB + DBMS	Programm zur Laufzeit

Datenbankmodelle

Historie



[Heuer/Saake 2000]

Datenbankmodelle

Datenbankschema

Das Datenbankschema einer Datenbank definiert für eine Anwendung:

1. statische Eigenschaften
   (a) identifizierbare Objekte (Basisdatentypen)
   (b) Beziehungen zwischen Objekten
   (c) Attribute von Objekten und Beziehungen

2. dynamische Eigenschaften
   (a) Operationen auf Daten
   (b) Abfolge und Koordination von Operationen

3. Integritätsbedingungen
   (a) an Objekte
   (b) an Operationen

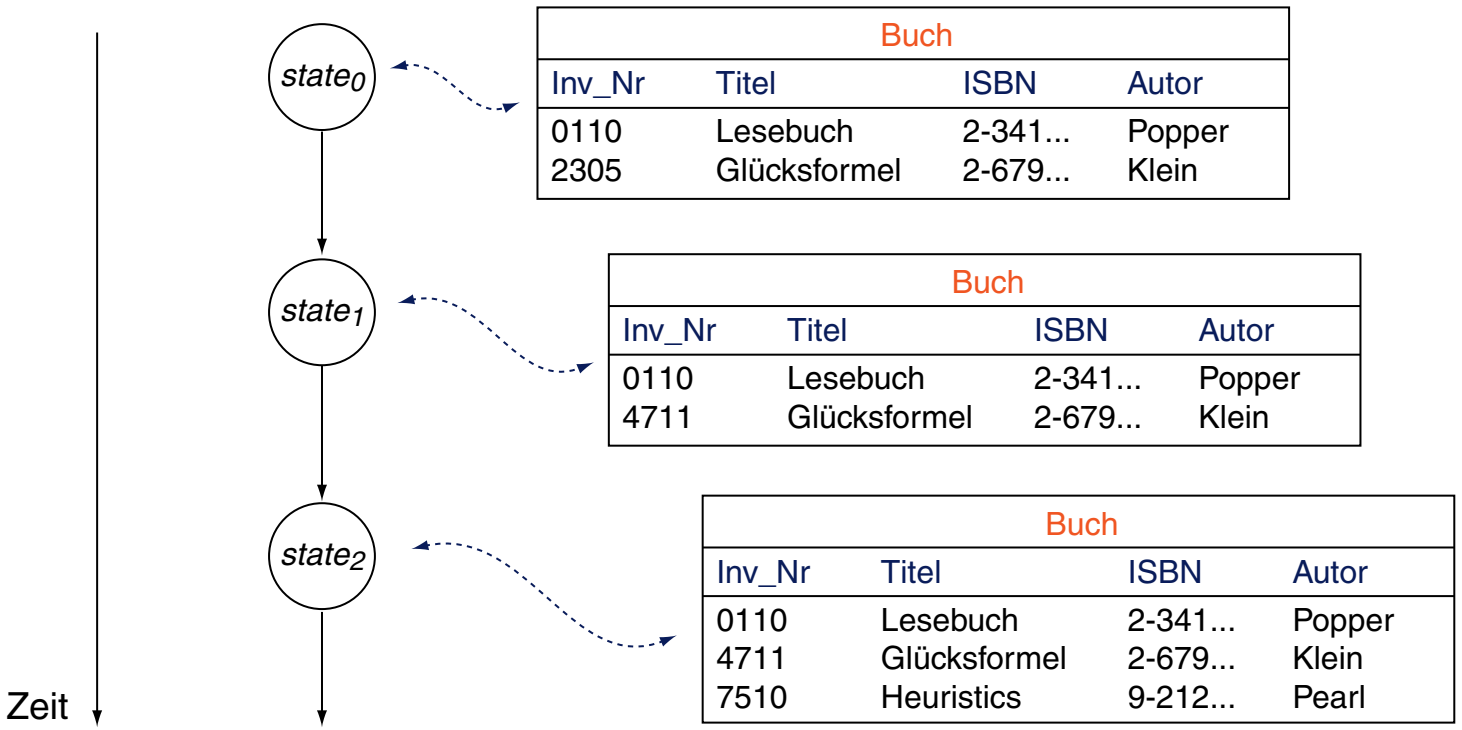
Bemerkungen:

- Beispiel für eine statische Eigenschaft: Sportler haben eine Nationalität.
- Beispiel für eine dynamische Eigenschaft: Ein Auto muss zugelassen werden, bevor man es fahren darf.
- Beispiel für eine Integritätsbedingung für Objekte: Studierende haben unterschiedliche Matrikelnummern. (Schlüsselbedingung)
- Beispiel für eine Integritätsbedingung für Operationen: Bei einer Gehaltsänderung darf das Gehalt nur steigen. (Übergangsbedingung)

Datenbankmodelle

Datenbankzustand

Ein Datenbankzustand ist der zu einem Zeitpunkt t gültige bzw. gespeicherte Zustand aller Objekte und ihren Beziehungen und muss den im Datenbankschema festgelegten Strukturbeschreibungen gehorchen.



Buch

Inv_Nr	Titel	ISBN	Autor
0110	Lesebuch	2-341...	Popper
2305	Glücksformel	2-679...	Klein

Buch

Inv_Nr	Titel	ISBN	Autor
0110	Lesebuch	2-341...	Popper
4711	Glücksformel	2-679...	Klein

Buch

Inv_Nr	Titel	ISBN	Autor
0110	Lesebuch	2-341...	Popper
4711	Glücksformel	2-679...	Klein
7510	Heuristics	9-212...	Pearl

Bemerkungen:

- Typischerweise ändert sich ein Datenbank*schema* selten; der Datenbank*zustand* ist Gegenstand laufender Modifikationen. Beispiel Flugbuchungssystem: Jede Reservierung entspricht einer Änderung des Datenbankzustands.
- Ein Datenbanksystem kann als kontinuierlich laufender Prozess aufgefasst werden, dessen jeweils aktueller Zustand *state* den Inhalt der Datenbank (Datenbasis) festlegt. Eine formale Definition der Semantik dieses Prozesses lässt sich durch eine lineare Folge von Zuständen modellieren, wobei die Zustandsübergänge den Änderungen der Datenbankinhalte entsprechen.

Kapitel DB:III

III. Konzeptueller Datenbankentwurf

- Einführung in das Entity-Relationship-Modell
- ER-Konzepte und ihre Semantik
- Charakterisierung von Beziehungstypen
- Existenzabhängige Entity-Typen
- Abstraktionskonzepte
- Konsolidierung, Sichtenintegration
- Konzeptuelle Modellierung mit UML

Einführung in das Entity-Relationship-Modell

Historie:

- Entity-Relationship-Modell kurz: ER-Modell bzw. ERM
- 1976 von Peter Chen vorgeschlagen
- Standardmodell für frühe Entwurfsphasen in der Datenbankentwicklung
- mittlerweile existieren viele Varianten und Erweiterungen
- eingesetzt auch in anderen Bereichen der Informatik

Eigenschaften:

- unabhängig von einem bestimmten Datenbanksystem
- Grundkonzepte „Entity“ und „Relationship“ werden als natürlich und – trotz ihrer Einfachheit – als ausreichend für viele Situationen empfunden
- unterstützt die Abstraktionsmechanismen der *Klassifikation*, der *Aggregation* und der *Verallgemeinerung*
- starre Informationsstruktur, d. h., Ausrichtung auf große Datenmengen
- Definition von statischen Eigenschaften und Integritätsbedingungen

Einführung in das Entity-Relationship-Modell

Elemente

1. Entity(-Instanz)
   Ein Objekt oder Ding (*Entity*) der realen oder einer virtuellen Welt, für das Informationen zu speichern sind. Jedes Entity ist von einem bestimmten Entity-Typ.

2. Beziehung(sinstanz)
   Ein Tupel von Entities. Jede Beziehung (*Relationship*) ist von einem bestimmten Beziehungstyp. Beziehungen zwischen Entities derselben Typen gehören zu demselben Beziehungstyp.

3. Attribut
   Definiert eine Eigenschaft von Entity-Typen oder Beziehungstypen.

4. Rolle
   Dokumentiert die Rolle eines Entities (Objektes) in einer Beziehung.

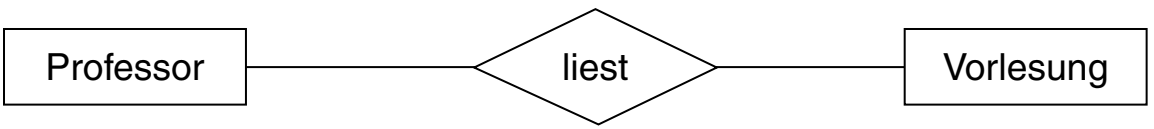
Einführung in das Entity-Relationship-Modell

Die Beschreibung der Informationsstruktur eines Anwendungsbereichs im Entity-Relationship-Modell heißt Entity-Relationship-Diagramm (ER-Diagramm) oder Entity-Relationship-Schema (ER-Schema).

" Der Professor liest eine Vorlesung "

Einführung in das Entity-Relationship-Modell

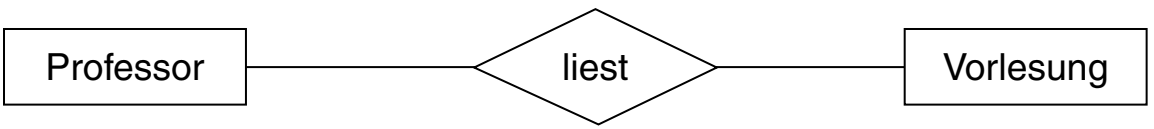
Die Beschreibung der Informationsstruktur eines Anwendungsbereichs im Entity-Relationship-Modell heißt Entity-Relationship-Diagramm (ER-Diagramm) oder Entity-Relationship-Schema (ER-Schema).



" Der Professor liest eine Vorlesung "

Einführung in das Entity-Relationship-Modell

Die Beschreibung der Informationsstruktur eines Anwendungsbereichs im Entity-Relationship-Modell heißt Entity-Relationship-Diagramm (ER-Diagramm) oder Entity-Relationship-Schema (ER-Schema).

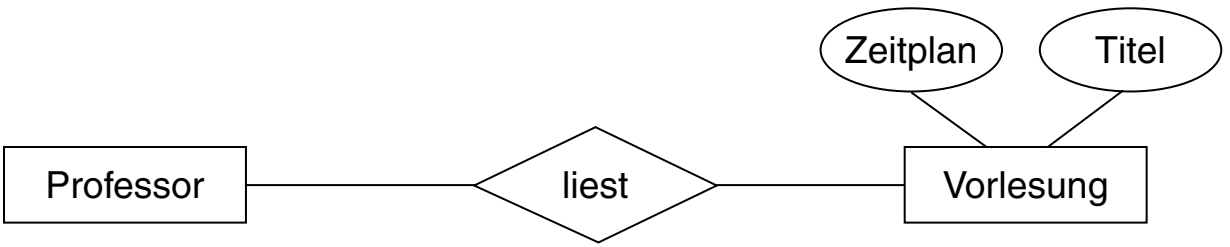


" Der Professor liest eine Vorlesung "

" Eine Vorlesung hat einen Zeitplan und einen Titel "

Einführung in das Entity-Relationship-Modell

Die Beschreibung der Informationsstruktur eines Anwendungsbereichs im Entity-Relationship-Modell heißt Entity-Relationship-Diagramm (ER-Diagramm) oder Entity-Relationship-Schema (ER-Schema).

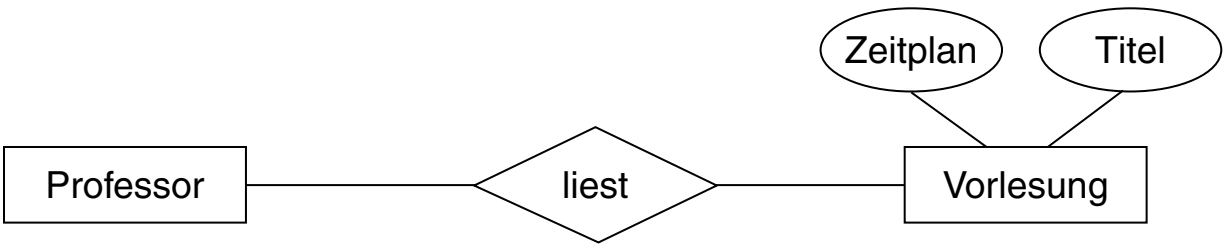


" Der Professor liest eine Vorlesung "

" Eine Vorlesung hat einen Zeitplan und einen Titel "

Einführung in das Entity-Relationship-Modell

Die Beschreibung der Informationsstruktur eines Anwendungsbereichs im Entity-Relationship-Modell heißt Entity-Relationship-Diagramm (ER-Diagramm) oder Entity-Relationship-Schema (ER-Schema).

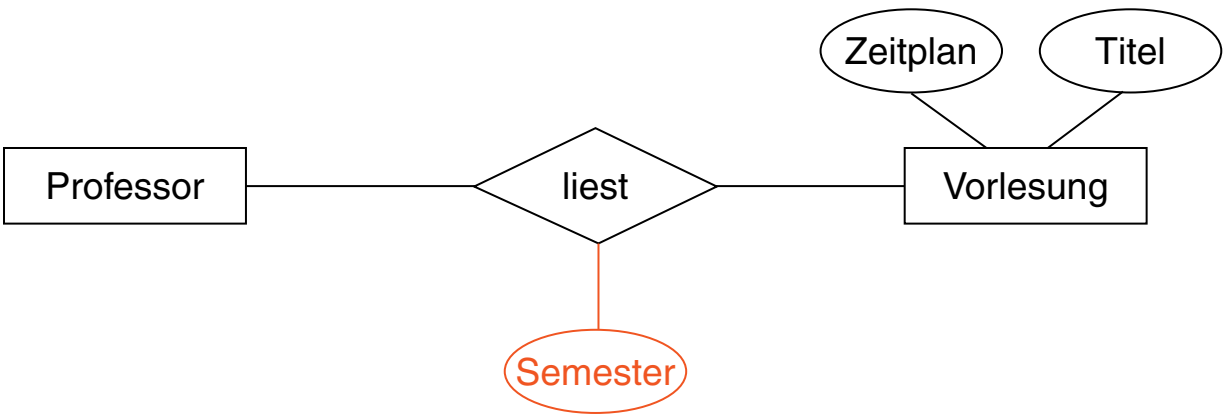


" Der Professor liest eine Vorlesung im zweiten Semester "

" Eine Vorlesung hat einen Zeitplan und einen Titel "

Einführung in das Entity-Relationship-Modell

Die Beschreibung der Informationsstruktur eines Anwendungsbereichs im Entity-Relationship-Modell heißt Entity-Relationship-Diagramm (ER-Diagramm) oder Entity-Relationship-Schema (ER-Schema).

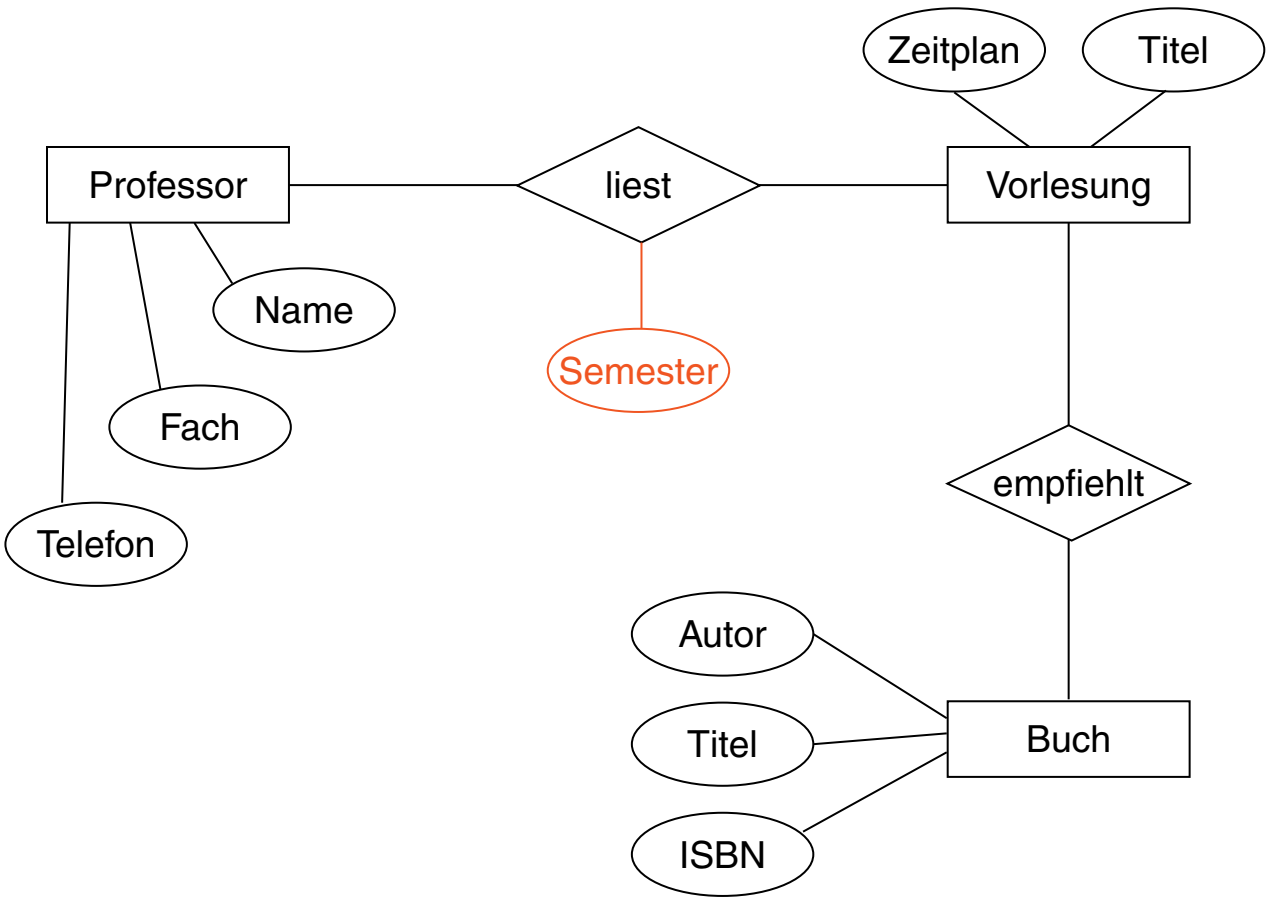


" Der Professor liest eine Vorlesung im zweiten Semester "

" Eine Vorlesung hat einen Zeitplan und einen Titel "

Einführung in das Entity-Relationship-Modell

Die Beschreibung der Informationsstruktur eines Anwendungsbereichs im Entity-Relationship-Modell heißt Entity-Relationship-Diagramm (ER-Diagramm) oder Entity-Relationship-Schema (ER-Schema).



ER-Konzepte und ihre Semantik

Datentypen

Datentypen, hier insbesondere Entity-, Beziehungs- und Attributtypen, dienen zur Definition von Wertebereichen. Ein Datentyp X ist gekennzeichnet durch:

1. *dom*(X) Domain, Definitionsbereich.
   Die möglichen Werte, die ein Objekt vom Typ X annehmen kann.

2. *state*(X) Zustand.
   Die zu einem Zeitpunkt angenommenen Werte eines Objektes vom Typ X.

3. Operationen, Funktionen, Prädikate, die für die Bearbeitung von Werten und für Aussagen über Werte zur Verfügung stehen.

ER-Konzepte und ihre Semantik

Datentypen

Datentypen, hier insbesondere Entity-, Beziehungs- und Attributtypen, dienen zur Definition von Wertebereichen. Ein Datentyp X ist gekennzeichnet durch:

1. *dom*(X) Domain, Definitionsbereich.
   Die möglichen Werte, die ein Objekt vom Typ X annehmen kann.

2. *state*(X) Zustand.
   Die zu einem Zeitpunkt angenommenen Werte eines Objektes vom Typ X.

3. Operationen, Funktionen, Prädikate, die für die Bearbeitung von Werten und für Aussagen über Werte zur Verfügung stehen.

Beispiele:

- abstrakter Datentyp `Integer`.
  dom$(\texttt{Integer}) = \mathbf{Z}$ mit Operationen $+ \quad - \quad * \quad / \quad < \quad > \quad =$
- abstrakter Datentyp `String`.
  dom$(\texttt{String}) = \Sigma^*$ für ein Alphabet Σ mit Operationen $+$ length

ER-Konzepte und ihre Semantik

Entity-Typen

Ein Entity-Typ deklariert eine Menge von Objekten mit bestimmten Eigenschaften, die (in der Datenbank) repräsentiert werden sollen. Graphische Darstellung:

Bezeichner für Entity-Typen sind $E, E_1, E_2, ...$ oder wie im Beispiel *Professor, Vorlesung* und *Buch*.

dom(E):

Menge der möglichen Entities (Objekte) vom Typ E. Diese Menge wird meist nicht explizit festgelegt, sondern als genügend groß, z.B. isomorph zu $\mathbf{N}$ angenommen. Man kann *dom*(E) als die Menge der Namen der Entities auffassen.

ER-Konzepte und ihre Semantik

Entity-Typen (Fortsetzung)

state(E):

Menge der aktuellen Entities vom Typ E im Zustand *state* der Datenbank. Durch Operationen auf der Datenbank kann sich *state*(E) verändern. Insbesondere gilt:

- *state*$(E) \subseteq$ *dom*(E)
- *state*(E) ist endlich
  Die Anzahl der zum Typ E gespeicherten Entities kann einen beliebigen Wert annehmen, ist aber in jedem Zustand *state* endlich.

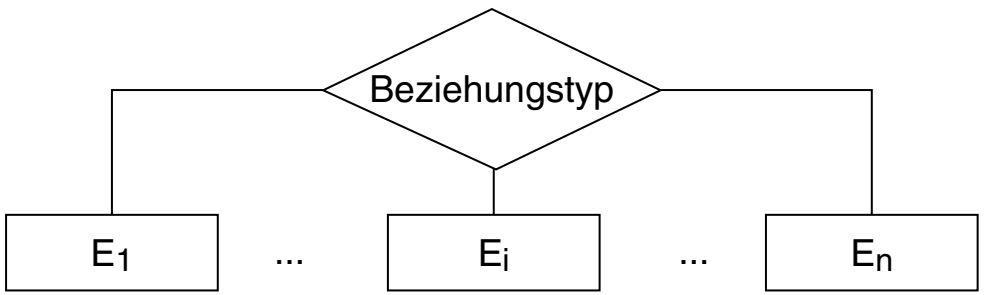
Beispiel:

- Entity-Typ *Professor*
- *dom*(*Professor*) $= \Sigma^*$ mit $\Sigma = \{A, \ldots, Z, a, \ldots, z\}$
- *state*(*Professor*) $= \{$Froehlich, Lucks, Wuethrich$\}$

ER-Konzepte und ihre Semantik

Beziehungstypen

Ein Beziehungstyp deklariert eine Beziehung zwischen Entity-Typen. Es kann eine beliebige Anzahl $n \geq 2$ von Entity-Typen an einem Beziehungstyp teilhaben. Graphische Darstellung:



Bezeichner für Beziehungstypen sind $R, R_1, R_2, \ldots$, oder wie im Beispiel *empfiehlt* und *liest*.

dom(R):

Menge der möglichen Beziehungen (Entity-Tupel) vom Typ R zwischen Entities der Typen $E_1, \ldots, E_n$.

$$\mathit{dom}(R) = \mathit{dom}(E_1) \times \ldots \times \mathit{dom}(E_n)$$

ER-Konzepte und ihre Semantik

Beziehungstypen (Fortsetzung)

state(R):

Menge der aktuellen Beziehungen vom Typ R im Zustand *state* der Datenbank. Durch Operationen auf der Datenbank kann sich *state*(R) verändern.

$$\textit{state}(R) \subseteq \textit{state}(E_1) \times \ldots \times \textit{state}(E_n)$$

Schreibweise:

$$R(E_1, \ldots, E_n)$$

Beispiel:

- Beziehungstyp *liest*
- *state*(*Professor*) $=$ {Froehlich, Lucks, Wuethrich}
  state(*Vorlesung*) $=$ {Algorithmen, Computergrafik, Kryptographie, HCI, DB}
- *state*(*liest*) $=$ {(Wuethrich, Algorithmen), (Lucks, Kryptographie)}

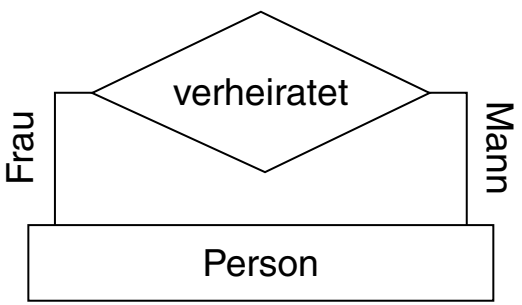
ER-Konzepte und ihre Semantik

Beziehungstypen (Fortsetzung)

Sind einzelne Entity-Typen mehrfach in einem Beziehungstyp eingebunden, legen Rollenbezeichner die Rollen der beteiligten Objekte in der Beziehung fest.

Beispiel:

$$verheiratet(Frau : Person, Mann : Person)$$



Bemerkungen:

- In der textuellen Notation sind Rollenbezeichner nicht erforderlich: die Rolle kann über die Position im Argumentetupel (Signatur) festgelegt werden: $\mathit{verheiratet}(\mathit{Person}, \mathit{Person})$
- In der graphischen Darstellung sind Rollenbezeichner erforderlich.

ER-Konzepte und ihre Semantik

Attribute und Attributtypen

Ein Attribut definiert eine Eigenschaft für alle Instanzen eines Entity-Typs oder Beziehungstyps. Ein Attribut kann deshalb als *Funktion* verstanden werden, die jeder Instanz eine Eigenschaftsausprägung zuordnet. Graphische Darstellung:

Bezeichner für Attribute sind $A, A_1, A_2, \ldots$ oder wie im Beispiel *Name*, *ISBN*, *Titel* oder *Semester*.

Der Attributtyp T bezeichnet den Typ der Eigenschaft und ist üblicherweise ein Standard-Datentyp wie `Integer` oder `String`.

dom(T):

Menge der möglichen Werte des Attributtyps T.

ER-Konzepte und ihre Semantik

Attribute von Entity-Typen

dom(A):

Menge der möglichen Abbildungen von *dom*(E) nach *dom*(T).

state(A):

Eine Abbildung *state*$(E) \rightarrow$ *dom*(T) im Zustand *state* der Datenbank, die jedem aktuellen Entity aus *state*(E) einen Wert aus *dom*(T) zuordnet.

Schreibweise (Entity-Typ mit Attributen $A_1, \ldots, A_n$):

$$E(A_1 : T_1, \ldots, A_n : T_n) \quad \text{bzw.} \quad E(A_1, \ldots, A_n)$$

Beispiel: $\leadsto \mathcal{TAFEL}$

Bemerkungen:

- Modellierungstechnisch gibt es zwei Blickwinkel, unter denen Attribute betrachtet werden. *dom*(A) und *dom*(T) spiegeln diese Unterscheidung wider:

  1. Aus Sicht des ER-Modells.
     Hier wird ein Attribut als eine Abbildung über dem Wertebereich eines Entity-Typs bzw. Beziehungstyps aufgefasst. *dom*(A) ist die Menge der möglichen Abbildungen. Diese Sichtweise ist durch die Operationen auf der Datenbank motiviert: Zu jedem Zeitpunkt t ist der Zustand der Datenbank durch die Attribute (= den entsprechenden Abbildungen) festgelegt. [DB:II Datenbankmodelle » Datenbankzustand]

     Beispiel: Das Attribut *ISBN* ordnet jedem Buch eine ganze Zahl zu.

  2. Aus Sicht des relationalen Modells.
     Hier wird ein Attribut lediglich als Bezeichner des Datentyps einer Eigenschaft verstanden. *dom*(A) im relationalen Modell ist die Menge der möglichen Attributwerte und entspricht der Menge *dom*(T) im ER-Modell. [DB:IV Das Relationale Modell]

     Beispiel: Das Attribut *ISBN* ist vom Typ „ganze Zahl“.

ER-Konzepte und ihre Semantik

Attribute von Beziehungstypen

dom(A):

Menge der möglichen Abbildungen von *dom*(R) nach *dom*(T).

state(A):

Eine Abbildung *state*$(R) \rightarrow$ *dom*(T) im Zustand *state* der Datenbank, die jeder aktuellen Beziehung aus *state*(R) einen Wert des Datentyps T aus *dom*(T) zuordnet.

Schreibweise:

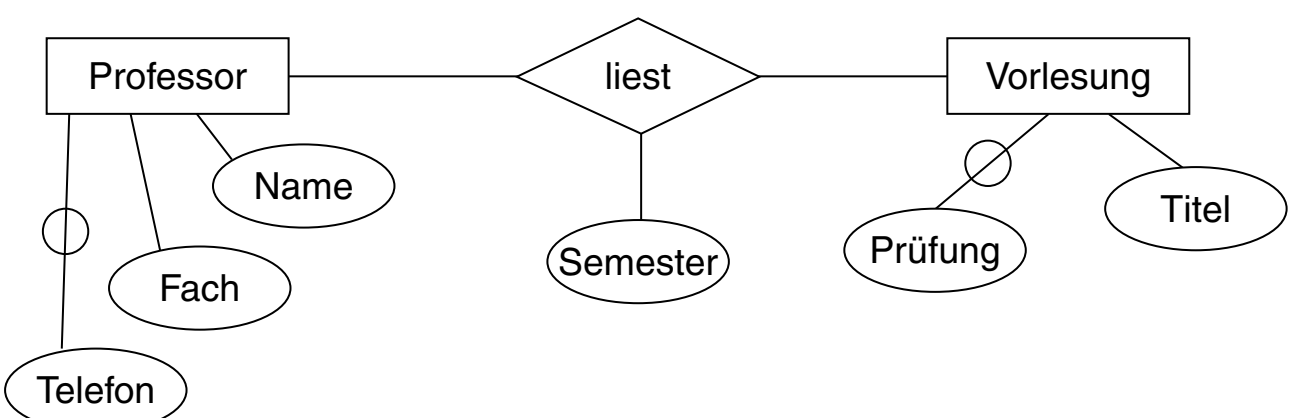
$$R(E_1, \ldots, E_m;\ A_1 : T_1,\ \ldots,\ A_n : T_n) \quad \text{bzw.} \quad R(E_1, \ldots, E_m;\ A_1, \ldots, A_n)$$

ER-Konzepte und ihre Semantik

Optionale Attribute

Optionale Attribute müssen nicht für alle Entities einen definierten Wert annehmen. Graphische Darstellung:

Beispiel:



Bemerkungen:

- Eine Kennzeichnung von A als optionales Attribut zeigt also an, dass es sich bei der Funktion *state*(A) um eine *partielle* Funktion handelt.

ER-Konzepte und ihre Semantik

Schlüssel

Definition 3 (Schlüssel im ER-Modell, Schlüsselattribut)

Sei $\{A_1, \ldots, A_n\}$ die Menge der Attribute eines Entity-Typs E. Eine nicht verkleinerbare Menge von Attributen $\{A_{i_1}, \ldots, A_{i_k}\} \subseteq \{A_1, \ldots, A_n\}$ nennt man Schlüssel, gdw. ($\leftrightarrow$) deren Werte das zugeordnete Entity eindeutig innerhalb aller Entities seines Typs identifiziert. Die $\{A_{i_1}, \ldots, A_{i_k}\}$ heißen Schlüsselattribute.

Beispiele:

- Ein/e Student/in wird eindeutig durch eine Matrikelnummer identifiziert.
- Ein Buch wird eindeutig durch eine ISBN identifiziert.

ER-Konzepte und ihre Semantik

Schlüssel

Definition 3 (Schlüssel im ER-Modell, Schlüsselattribut)

Sei $\{A_1, \ldots, A_n\}$ die Menge der Attribute eines Entity-Typs E. Eine nicht verkleinerbare Menge von Attributen $\{A_{i_1}, \ldots, A_{i_k}\} \subseteq \{A_1, \ldots, A_n\}$ nennt man Schlüssel, gdw. ($\leftrightarrow$) deren Werte das zugeordnete Entity eindeutig innerhalb aller Entities seines Typs identifiziert. Die $\{A_{i_1}, \ldots, A_{i_k}\}$ heißen Schlüsselattribute.

Beispiele:

- Ein/e Student/in wird eindeutig durch eine Matrikelnummer identifiziert.
- Ein Buch wird eindeutig durch eine ISBN identifiziert.

Folgerung auf Basis von Definition 3:

$\{A_2\}$, $\{A_2\} \subseteq \{A_1, A_2, A_3\}$, sind die Schlüsselattribute von $E(A_1, A_2, A_3)$.

$\Rightarrow$

$\forall e_1, e_2 \in \textit{state}(E)$:
Gleiche Werte für A_2 implizieren die die Gleichheit von e_1 und e_2.

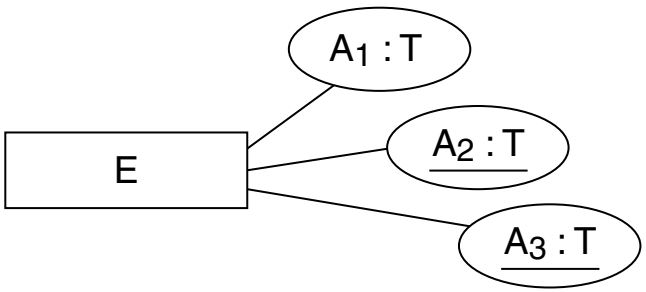
Bemerkungen:

- Es handelt sich um eine Folgerung und keine Äquivalenzbeziehung, da aus der eindeutigen Identifizierbarkeit in der Menge *state*(E) nicht auf die eindeutige Identifizierbarkeit in der Menge *dom*(E) geschlossen werden kann: *state*$(E) \subseteq$ *dom*(E).
  Die eindeutige Identifizierbarkeit innerhalb der Menge *aller* Entities eines Typs (= *dom*(E)) ist aber in der Definition gefordert.
- Es kann mehrere *Schlüsselkandidaten* geben. Von ihnen ist einer auszuwählen und wird als *Primärschlüssel* bezeichnet.
- Verschiedene Schlüsselkandidaten können eine unterschiedliche Anzahl von Attributen besitzen.

ER-Konzepte und ihre Semantik

Schlüssel (Fortsetzung)

Graphische Darstellung:



Schreibweise:

$$E(A_1, \underline{A_2}, \underline{A_3})$$

ER-Konzepte und ihre Semantik

Datenbankzustand

Für jeden Zustand *state* einer Datenbank besitzt ein ER-Schema folgende Zuordnungen:

$$
\begin{aligned}
E &\mapsto \mathit{state}(E) \subseteq \mathit{dom}(E) \\
R(E_1, \ldots, E_n) &\mapsto \mathit{state}(R) \subseteq \mathit{state}(E_1) \times \ldots \times \mathit{state}(E_n) \\
E(\ldots, A_i : T, \ldots) &\mapsto \mathit{state}(A_i) : \mathit{state}(E) \to \mathit{dom}(T) \\
R(\ldots; \ldots, A_j : T, \ldots) &\mapsto \mathit{state}(A_j) : \mathit{state}(R) \to \mathit{dom}(T)
\end{aligned}
$$

Vorausgesetzt sind passende Interpretationen für $\mathit{dom}(E)$ und $\mathit{dom}(T)$: Mengen möglicher Entities für Entity-Typen und Wertebereiche für Datentypen.

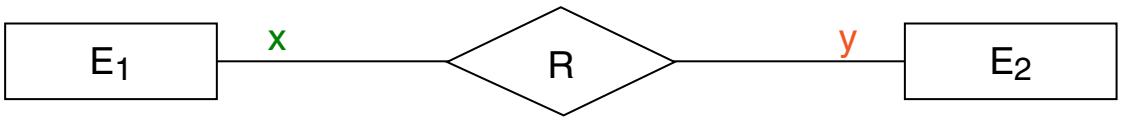
Kapitel DB:III (Fortsetzung)

III. Konzeptueller Datenbankentwurf

- Einführung in das Entity-Relationship-Modell
- ER-Konzepte und ihre Semantik
- Charakterisierung von Beziehungstypen
- Existenzabhängige Entity-Typen
- Abstraktionskonzepte
- Konsolidierung, Sichtenintegration
- Konzeptuelle Modellierung mit UML

Charakterisierung von Beziehungstypen

Funktionalitäten bei zweistelligen Beziehungen (Formalismus I für Kardinalitäten)

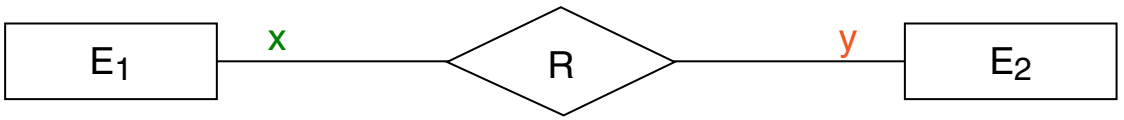


Zweistellige Beziehungstypen und ihre Semantik:

- 1:1-Beziehung
- 1:n-Beziehung
- n:m-Beziehung

Charakterisierung von Beziehungstypen

Funktionalitäten bei zweistelligen Beziehungen (Formalismus I für Kardinalitäten)



Zweistellige Beziehungstypen und ihre Semantik:

- 1:1-Beziehung

  Jedem Entity $e_1 \in \mathit{state}(E_1)$ ist höchstens ein Entity $e_2 \in \mathit{state}(E_2)$ zugeordnet und umgekehrt.

- 1:n-Beziehung

  Jedem Entity $e_1 \in \mathit{state}(E_1)$ sind beliebig viele (auch gar keine) Entities aus $\mathit{state}(E_2)$ zugeordnet, und jedem Entity $e_2 \in \mathit{state}(E_2)$ ist höchstens ein Entity $e_1 \in \mathit{state}(E_1)$ zugeordnet.

- n:m-Beziehung

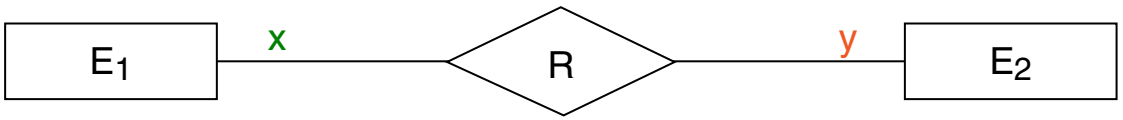
  Keine Restriktionen gelten; jedes Entity $e_1 \in \mathit{state}(E_1)$ kann mit beliebig vielen Entities $e_2 \in \mathit{state}(E_2)$ in Beziehung stehen und umgekehrt.

Bemerkungen:

- Mit dem Wort „Funktionalität“ wird hier eine Abhängigkeitsbeziehung zwischen Entity-Typen bzw. zwischen Entity-Typen und einem Beziehungstyp bezeichnet. Eine andere Bezeichnung für Funktionalität ist *Constraint*.
- Die Semantik von n:1-Beziehungen ist analog zu der von 1:n-Beziehungen.
- Bei jeder Art von zweistelligen Beziehungen kann es Entities aus *state*(E_1) (bzw. *state*(E_2)) geben, denen kein Element aus *state*(E_2) (bzw. *state*(E_1)) zugeordnet ist.

Charakterisierung von Beziehungstypen

Funktionalitäten bei zweistelligen Beziehungen (Formalismus I für Kardinalitäten)



Beispiele:

- 1:1-Beziehung. „verheiratet“-Relation zwischen Männern und Frauen nach europäischem Recht.
- 1:n-Beziehung. „beschäftigen“-Relation zwischen Firmen und Personen.

Bemerkungen:

- Funktionalitäten stellen *Integritätsbedingungen* dar, die in der zu modellierenden Welt immer gelten müssen.

- Die binären 1:1, 1:n und n:1-Beziehungen stellen partielle Funktionen dar. Insbesondere ist jede 1:1-Beziehung umkehrbar, und es existieren folgende Funktionen:
  $R : E_1 \rightarrow E_2$
  $R^{-1} : E_2 \rightarrow E_1$

  Beispiel:
  Ehemann: Frauen $\rightarrow$ Männer
  Ehefrau: Männer $\rightarrow$ Frauen

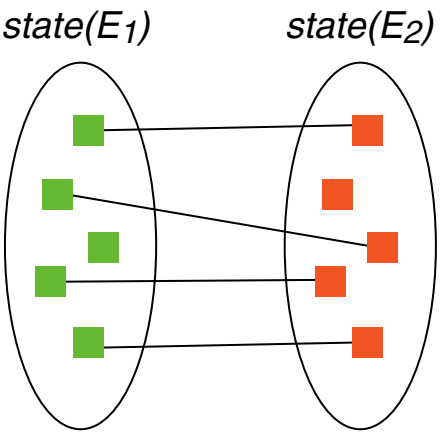
- Bei einer 1:n-Beziehung ist die Richtung der Funktion zwingend, vom „n-Entity-Typ“ zum „1-Entity-Typ“.

  Beispiel:
  beschäftigen: Personen $\rightarrow$ Firmen

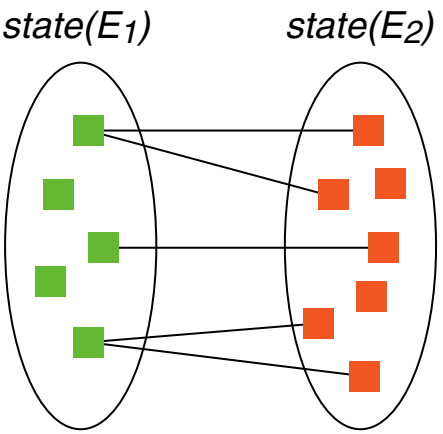
Charakterisierung von Beziehungstypen

Funktionalitäten bei zweistelligen Beziehungen (Formalismus I für Kardinalitäten)

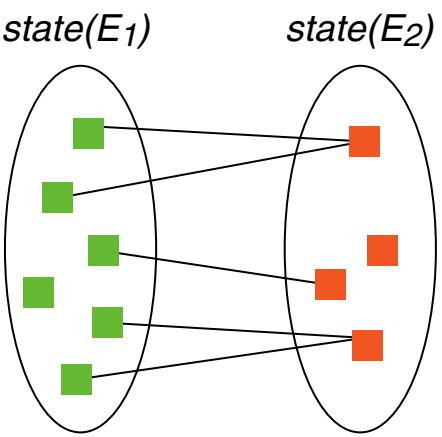
1:1



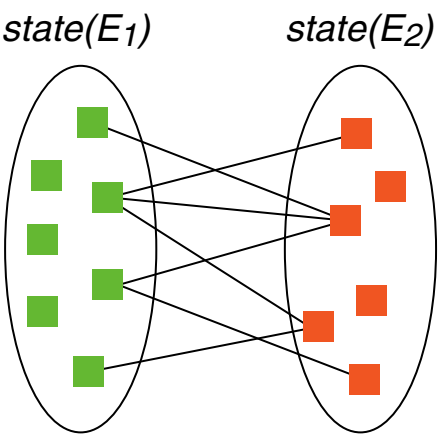
1:n



n:1



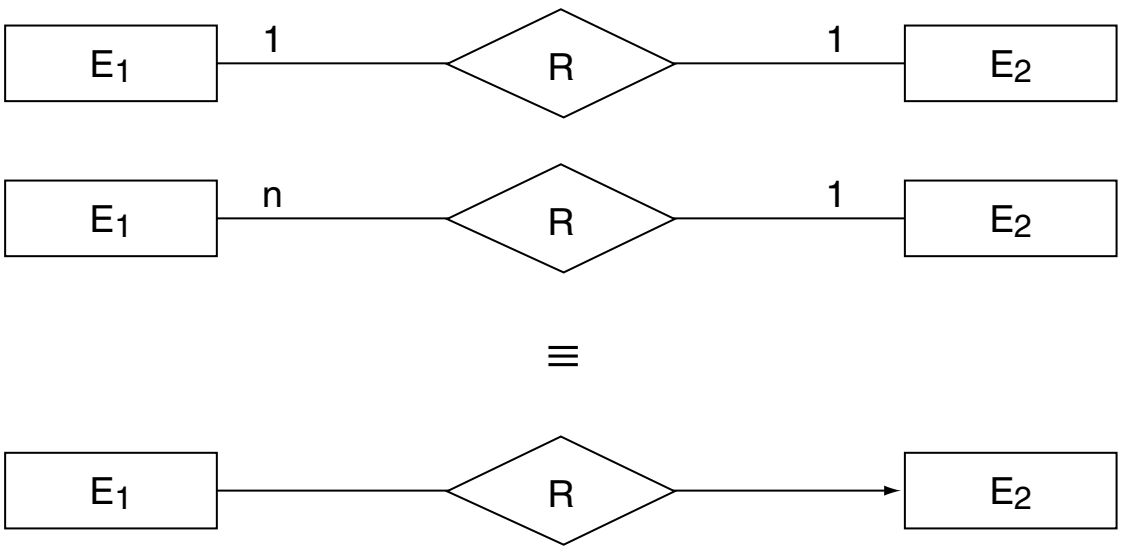
n:m



Charakterisierung von Beziehungstypen

Funktionalitäten bei zweistelligen Beziehungen (Formalismus I für Kardinalitäten)

Alternative graphische Darstellungen für zweistellige funktionale Beziehungen:



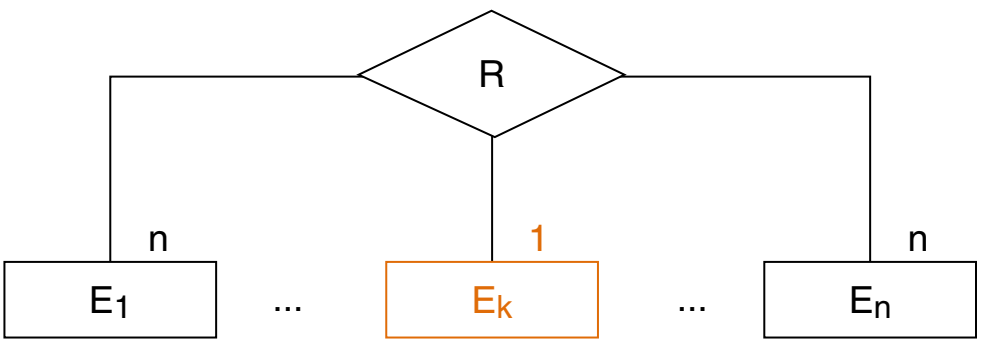
Charakterisierung von Beziehungstypen

Funktionalitäten bei n-stelligen Beziehungen (Formalismus I für Kardinalitäten)

Sei R eine Beziehung zwischen den Entity-Typen $E_1, \ldots, E_n$, wobei die Funktionalität bei Entity-Typ E_k, $1 \leq k \leq n$, mit 1 spezifiziert ist. Dann wird durch R folgende partielle Funktion vorgegeben:

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

[Kemper/Eickler 2011]



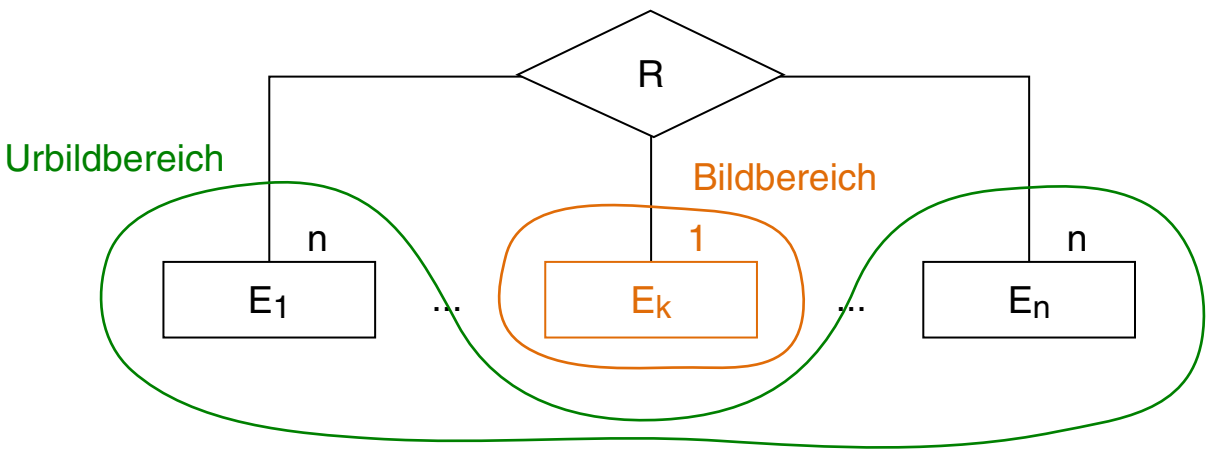
Charakterisierung von Beziehungstypen

Funktionalitäten bei n-stelligen Beziehungen (Formalismus I für Kardinalitäten)

Sei R eine Beziehung zwischen den Entity-Typen $E_1, \ldots, E_n$, wobei die Funktionalität bei Entity-Typ E_k, $1 \leq k \leq n$, mit 1 spezifiziert ist. Dann wird durch R folgende partielle Funktion vorgegeben:

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

[Kemper/Eickler 2011]



Bemerkungen:

- n-stellige (n-äre) Beziehungstypen spezifizieren genau *die* Menge der Funktionen, für deren Signatur gilt: die Bildmenge besteht aus einem Entity-Typ mit der Kardinalität 1; die Urbildmenge ist das kartesische Produkt der restlichen $n-1$ Entity-Typen.
- Diese Definition von n-stelligen Beziehungstypen erweitert die binären n:1-Beziehungen in kanonischer Weise. Dabei ist der 1-Entity-Typ der rechten Seite in der Rolle des rechten 1-Entity-Typs der binären Funktionalität; die restlichen $n-1$ Entity-Typen sind zusammen in der Rolle des linken n-Entity-Typs der binären Funktionalität.
- n-stellige Beziehungstypen sind in der Regel *keine* Kurzschreibweise von $\binom{n}{2}$ binären Beziehungstypen. Selbst eine ternäre 1:1:1-Relation R lässt sich nicht zwangsläufig verlustlos bzw. verbundtreu in binäre Beziehungstypen zerlegen.

Charakterisierung von Beziehungstypen

Funktionalitäten bei n-stelligen Beziehungen (Formalismus I für Kardinalitäten)

Beispiel Prüfungsordnung (= Ausschnitt der realen Welt) :

(a) Ein Student darf bei demselben Professor nur ein Seminarthema machen.

(b) Ein Student darf dasselbe Thema nur einmal bearbeiten und nicht zu einem anderen Professor damit gehen.

(c) Ein Professor kann dasselbe Thema an verschiedene Studenten vergeben.

(d) Dasselbe Thema kann von verschiedenen Professoren vergeben werden.

Charakterisierung von Beziehungstypen

Funktionalitäten bei n-stelligen Beziehungen (Formalismus I für Kardinalitäten)

Beispiel Prüfungsordnung (= Ausschnitt der realen Welt) :

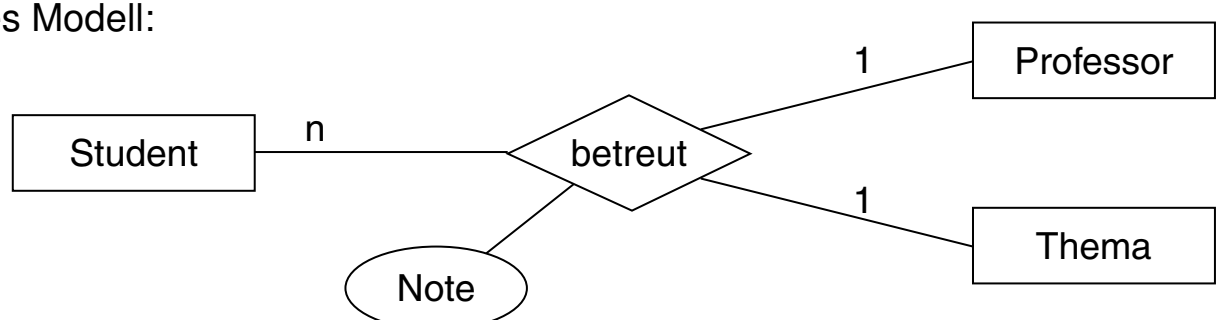
(a) Ein Student darf bei demselben Professor nur ein Seminarthema machen.

(b) Ein Student darf dasselbe Thema nur einmal bearbeiten und nicht zu einem anderen Professor damit gehen.

(c) Ein Professor kann dasselbe Thema an verschiedene Studenten vergeben.

(d) Dasselbe Thema kann von verschiedenen Professoren vergeben werden.

Konzeptuelles Modell:



Charakterisierung von Beziehungstypen

Funktionalitäten bei n-stelligen Beziehungen (Formalismus I für Kardinalitäten)

Beispiel Prüfungsordnung (= Ausschnitt der realen Welt) :

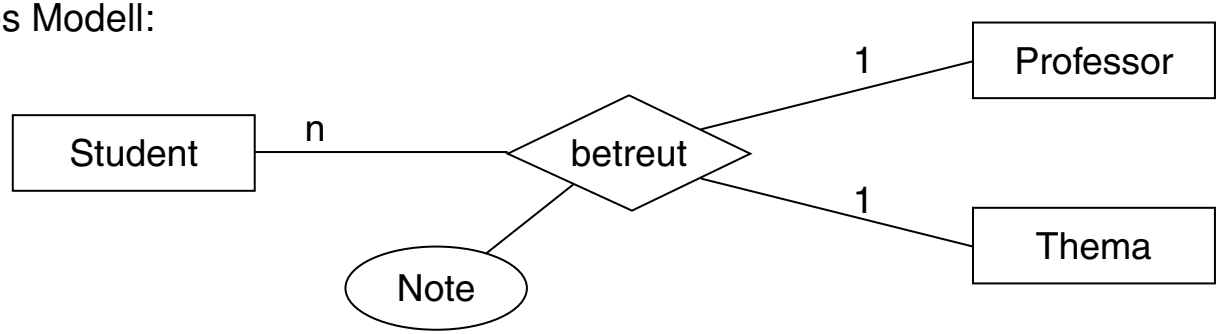
(a) Ein Student darf bei demselben Professor nur ein Seminarthema machen.

(b) Ein Student darf dasselbe Thema nur einmal bearbeiten und nicht zu einem anderen Professor damit gehen.

(c) Ein Professor kann dasselbe Thema an verschiedene Studenten vergeben.

(d) Dasselbe Thema kann von verschiedenen Professoren vergeben werden.

Konzeptuelles Modell:



Analyse des konzeptuellen Modells:

zu (a)
zu (b)
zu (c)
zu (d)

Charakterisierung von Beziehungstypen

Funktionalitäten bei n-stelligen Beziehungen (Formalismus I für Kardinalitäten)

Beispiel Prüfungsordnung (= Ausschnitt der realen Welt) :

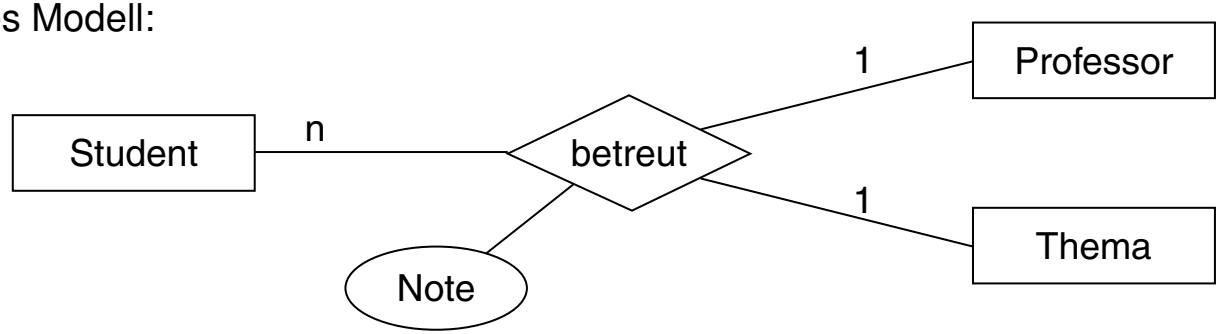
(a) Ein Student darf bei demselben Professor nur ein Seminarthema machen.

(b) Ein Student darf dasselbe Thema nur einmal bearbeiten und nicht zu einem anderen Professor damit gehen.

(c) Ein Professor kann dasselbe Thema an verschiedene Studenten vergeben.

(d) Dasselbe Thema kann von verschiedenen Professoren vergeben werden.

Konzeptuelles Modell:



Analyse des konzeptuellen Modells:

zu (a) Abgebildet durch $betreut_{f_1} : Professor \times Student \rightarrow Thema$

zu (b)

zu (c)

zu (d)

Charakterisierung von Beziehungstypen

Funktionalitäten bei n-stelligen Beziehungen (Formalismus I für Kardinalitäten)

Beispiel Prüfungsordnung (= Ausschnitt der realen Welt) :

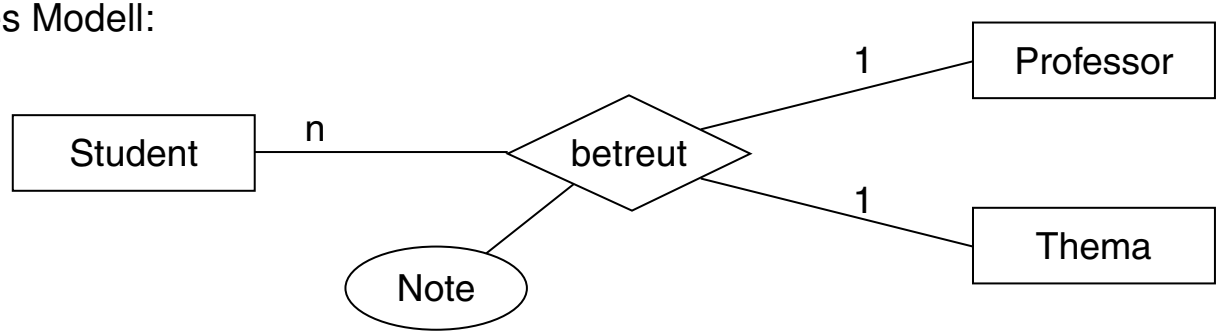
(a) Ein Student darf bei demselben Professor nur ein Seminarthema machen.

(b) Ein Student darf dasselbe Thema nur einmal bearbeiten und nicht zu einem anderen Professor damit gehen.

(c) Ein Professor kann dasselbe Thema an verschiedene Studenten vergeben.

(d) Dasselbe Thema kann von verschiedenen Professoren vergeben werden.

Konzeptuelles Modell:



Analyse des konzeptuellen Modells:

zu (a) Abgebildet durch $\mathit{betreut}_{f_1} : \text{Professor} \times \text{Student} \rightarrow \text{Thema}$

zu (b) Abgebildet durch $\mathit{betreut}_{f_2} : \text{Thema} \times \text{Student} \rightarrow \text{Professor}$

zu (c)

zu (d)

Charakterisierung von Beziehungstypen

Funktionalitäten bei n-stelligen Beziehungen (Formalismus I für Kardinalitäten)

Beispiel Prüfungsordnung (= Ausschnitt der realen Welt) :

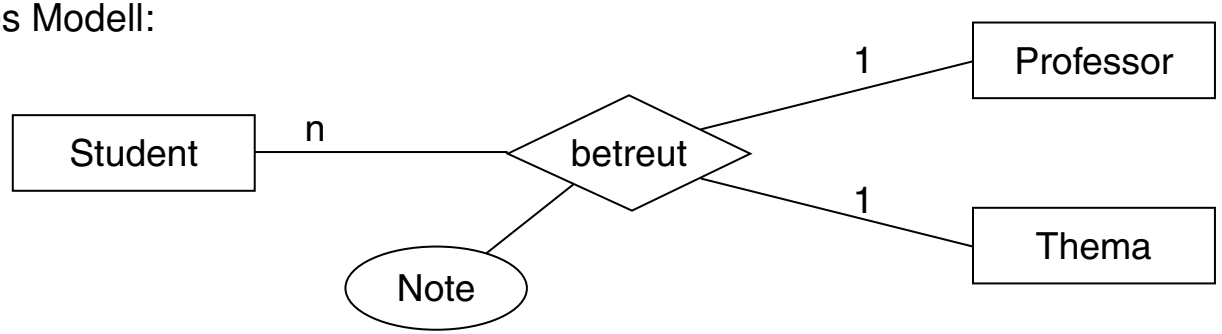
(a) Ein Student darf bei demselben Professor nur ein Seminarthema machen.

(b) Ein Student darf dasselbe Thema nur einmal bearbeiten und nicht zu einem anderen Professor damit gehen.

(c) Ein Professor kann dasselbe Thema an verschiedene Studenten vergeben.

(d) Dasselbe Thema kann von verschiedenen Professoren vergeben werden.

Konzeptuelles Modell:



Analyse des konzeptuellen Modells:

zu (a) Abgebildet durch $\mathit{betreut}_{f_1}$: Professor $\times$ Student $\rightarrow$ Thema

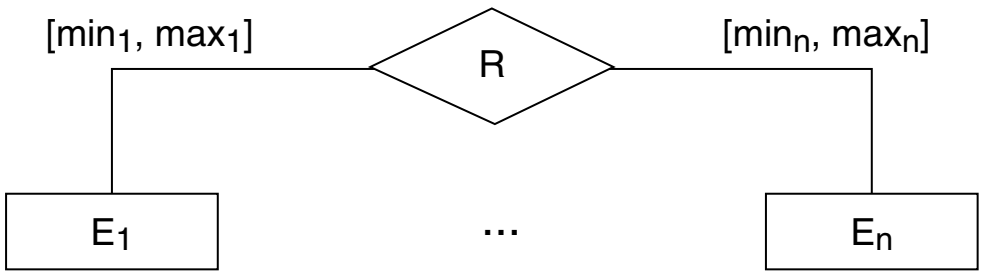
zu (b) Abgebildet durch $\mathit{betreut}_{f_2}$: Thema $\times$ Student $\rightarrow$ Professor

zu (c) Zugelassen durch $\mathit{betreut}_{f_1}$, $\mathit{betreut}_{f_2}$.

zu (d) Zugelassen durch $\mathit{betreut}_{f_1}$, $\mathit{betreut}_{f_2}$.

Charakterisierung von Beziehungstypen

[*min*, *max*]-Notation (Formalismus II für Kardinalitäten)

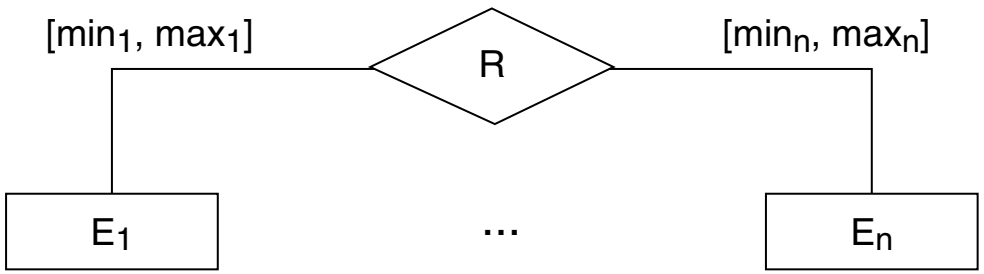


Schreibweise:

$$R(E_1[min_1, max_1], \ldots, E_i[min_i, max_i], \ldots, E_n[min_n, max_n])$$

Charakterisierung von Beziehungstypen

$[min, max]$-Notation (Formalismus II für Kardinalitäten)



Schreibweise:

$$R(E_1[\mathit{min}_1, \mathit{max}_1], \ldots, E_i[\mathit{min}_i, \mathit{max}_i], \ldots, E_n[\mathit{min}_n, \mathit{max}_n])$$

Semantik der $[min, max]$-Notation:

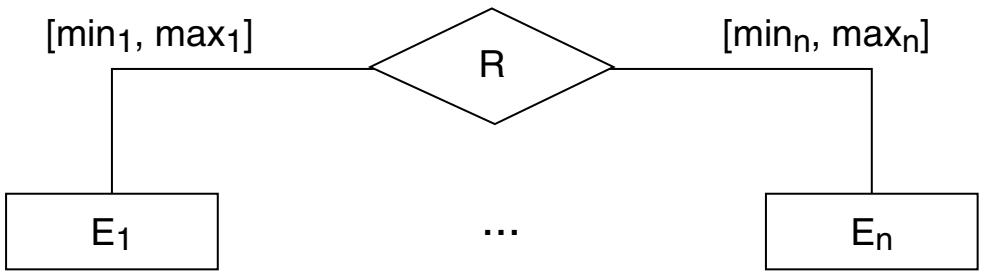
Die Anzahl der Beziehungsinstanzen vom Typ R mit Beteiligung der Instanz e_i vom Typ E_i ist zwischen min_i und max_i:

$$\forall i \in \{1, \ldots, n\} \;\; \forall e_i \in \mathit{state}(E_i) : \;\; \mathit{min}_i \leq |\{r \in \mathit{state}(R) \mid r.E_i = e_i\}| \leq \mathit{max}_i$$

$r.E_i$ bezeichne die Projektion der Beziehungsinstanz r auf den Entity-Typ E_i.

Charakterisierung von Beziehungstypen

$[min, max]$-Notation (Formalismus II für Kardinalitäten)



Schreibweise:

$$R(E_1[min_1, max_1], \ldots, E_i[min_i, max_i], \ldots, E_n[min_n, max_n])$$

Semantik der $[min, max]$-Notation:

Die Anzahl der Beziehungsinstanzen vom Typ R mit Beteiligung der Instanz e_i vom Typ E_i ist zwischen min_i und max_i:

$$\forall i \in \{1, \ldots, n\} \;\; \forall e_i \in state(E_i) : \;\; min_i \leq |\{r \in state(R) \mid r.E_i = e_i\}| \leq max_i$$

$r.E_i$ bezeichne die Projektion der Beziehungsinstanz r auf den Entity-Typ E_i.

Charakterisierung von Beziehungstypen

[*min*, *max*]-Notation (Formalismus II für Kardinalitäten)

Beispiele:

- *arbeitet_in*(Mitarbeiter[0,1], Raum[0,3])

- *verantwortlich_für*(Mitarbeiter[0,*], Computer[1,1])

Charakterisierung von Beziehungstypen

[*min*, *max*]-Notation (Formalismus II für Kardinalitäten)

Beispiele:

- *arbeitet_in*(Mitarbeiter[0,1], Raum[0,3])

  „Mitarbeitern ist ein oder kein Raum zugeordnet. Pro Raum gibt es höchstens drei Mitarbeiter.“

- *verantwortlich_für*(Mitarbeiter[0,*], Computer[1,1])

  „Es gibt Mitarbeiter, die für keinen Computer verantwortlich sind – aber auch Mitarbeiter, die für mehrere Computer verantwortlich sind. Jedem Computer ist genau ein Mitarbeiter zugeordnet, der verantwortlich für diesen Computer ist.“

Bemerkungen:

- Eine spezielle Wertangabe für max_i ist $*$. Sie dient zum Anzeigen einer unbegrenzten Anzahl.
- $[0, *]$ bedeutet keine Einschränkung und ist die Standardannahme, wenn keine Kardinalitäten angegeben sind.
- $R(E_1[0, 1], E_2)$ entspricht einer *partiellen funktionalen* Beziehung $R : E_1 \to E_2$.
- $R(E_1[1, 1], E_2)$ entspricht einer *totalen funktionalen* Beziehung $R : E_1 \to E_2$.
- Die Kardinalitätsangabe auf der rechten Seite verrät auch etwas über die Surjektivität der Abbildung: $[0, *]$ bedeutet, dass nicht jedes Element in einer Beziehung stehen muss; $[1, *]$ bedeutet, dass jedes Element erreicht wird, die Funktion also surjektiv ist.

Charakterisierung von Beziehungstypen

Formalismus I versus Formalismus II

Unterschied in der Semantik:

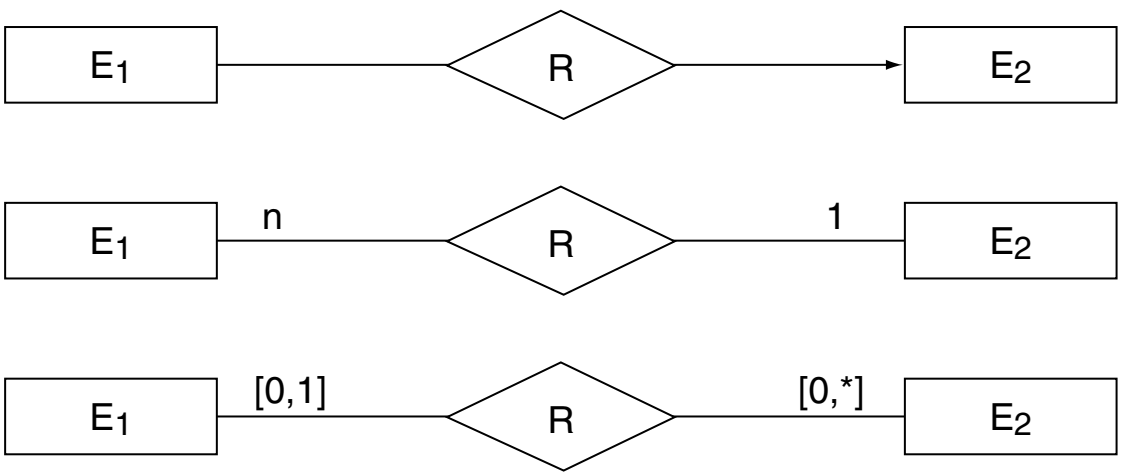
- Die Semantik der Kardinalitäten bei Formalismus I bezieht sich auf Instanzen von Entity-Typen.
- Die Semantik der Kardinalitäten bei Formalismus II ($[min, max]$-Notation) bezieht sich auf Instanzen von Beziehungstypen.

Charakterisierung von Beziehungstypen

Formalismus I versus Formalismus II

Unterschied in der Semantik:

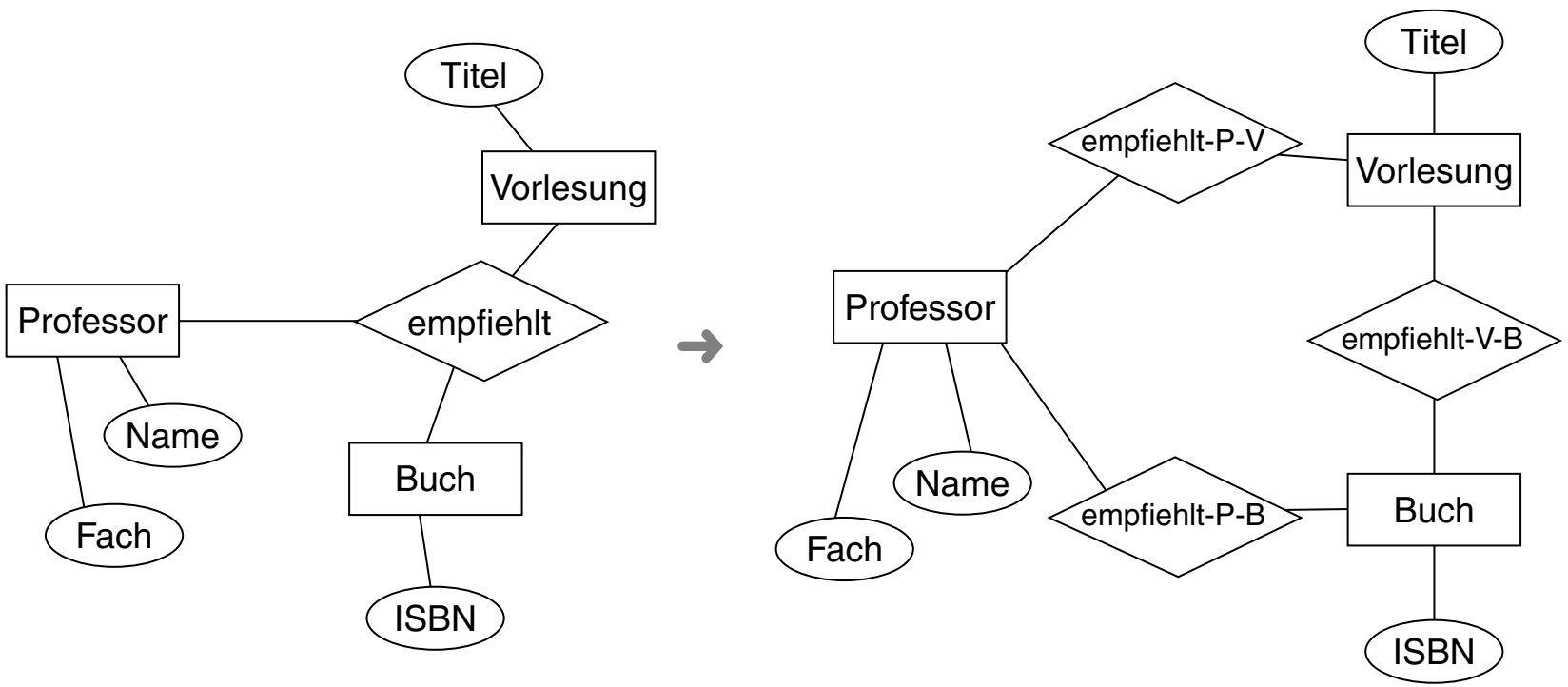
- Die Semantik der Kardinalitäten bei Formalismus I bezieht sich auf Instanzen von Entity-Typen.
- Die Semantik der Kardinalitäten bei Formalismus II ($[min, max]$-Notation) bezieht sich auf Instanzen von Beziehungstypen.
- Somit kann eine partielle Funktion $R : E_1 \rightarrow E_2$ graphisch u.a. auf folgende Arten dargestellt werden:



Charakterisierung von Beziehungstypen

Umwandlung n-stelliger Beziehungen

Beispiel für die Umwandlung einer dreistelligen Beziehung in drei zweistellige Beziehungen:



Charakterisierung von Beziehungstypen

Umwandlung n-stelliger Beziehungen (Fortsetzung)

ursprünglicher Zustand:

empfiehlt		
Professor	Vorlesung	Buch (ISBN)
Pearl	Datenbanken	0-341...
Pearl	IT-Systeme	2-305...
Graham	Datenbanken	2-305...
Graham	IT-Systeme	2-305...

Zustände der drei zweistelligen Beziehungen:

empfiehlt-P-V	
Professor	Vorlesung
Pearl	Datenbanken
Pearl	IT-Systeme
Graham	Datenbanken
Graham	IT-Systeme

empfiehlt-P-B	
Professor	Buch (ISBN)
Pearl	0-341...
Pearl	2-305...
Graham	2-305...

empfiehlt-V-B	
Vorlesung	Buch (ISBN)
Datenbanken	0-341...
IT-Systeme	2-305...
Datenbanken	2-305...

Charakterisierung von Beziehungstypen

Umwandlung n-stelliger Beziehungen (Fortsetzung)

ursprünglicher Zustand:

empfiehlt		
Professor	Vorlesung	Buch (ISBN)
Pearl	Datenbanken	0-341...
Pearl	IT-Systeme	2-305...
Graham	Datenbanken	2-305...
Graham	IT-Systeme	2-305...

Zustände der drei zweistelligen Beziehungen:

empfiehlt-P-V	
Professor	Vorlesung
Pearl	Datenbanken
Pearl	IT-Systeme
Graham	Datenbanken
Graham	IT-Systeme

empfiehlt-P-B	
Professor	Buch (ISBN)
Pearl	0-341...
Pearl	2-305...
Graham	2-305...

empfiehlt-V-B	
Vorlesung	Buch (ISBN)
Datenbanken	0-341...
IT-Systeme	2-305...
Datenbanken	2-305...

Charakterisierung von Beziehungstypen

Umwandlung n-stelliger Beziehungen (Fortsetzung)

ursprünglicher Zustand:

empfiehlt		
Professor	Vorlesung	Buch (ISBN)
Pearl	Datenbanken	0-341...
Pearl	IT-Systeme	2-305...
Graham	Datenbanken	2-305...
Graham	IT-Systeme	2-305...
Pearl	**Datenbanken**	**2-305...**

Problem 1

Zustände der drei zweistelligen Beziehungen:

empfiehlt-P-V	
Professor	Vorlesung
Pearl	Datenbanken
Pearl	IT-Systeme
Graham	Datenbanken
Graham	IT-Systeme

empfiehlt-P-B	
Professor	Buch (ISBN)
Pearl	0-341...
Pearl	2-305...
Graham	2-305...

empfiehlt-V-B	
Vorlesung	Buch (ISBN)
Datenbanken	0-341...
IT-Systeme	2-305...
Datenbanken	2-305...

Charakterisierung von Beziehungstypen

Umwandlung n-stelliger Beziehungen (Fortsetzung)

ursprünglicher Zustand:

empfiehlt		
Professor	Vorlesung	Buch (ISBN)
	?	

Problem 2

Zustände der drei zweistelligen Beziehungen:

empfiehlt-P-V	
Professor	Vorlesung
Pearl	Datenbanken
Pearl	IT-Systeme
Graham	Datenbanken
Graham	IT-Systeme

empfiehlt-P-B	
Professor	Buch (ISBN)
Pearl	0-341...
Pearl	2-305...
Graham	2-305...

empfiehlt-V-B	
Vorlesung	Buch (ISBN)
Datenbanken	0-341...
IT-Systeme	2-305...
Datenbanken	2-305...
Web-Services	**1-100**

Bemerkungen:

- Verschiedene Zustände der dreistelligen Relation (oben) bilden auf dieselben zweistelligen Relationen (unten) ab. Bei der Umkehrung dieser Zerlegung (von unten nach oben) können zusätzliche Tupel enstehen.
- Die illustrierte Problematik wird als die Eigenschaft der *Verlustlosigkeit* bzw. *Verbundtreue* in der Entwurfstheorie relationaler Datenbanken behandelt.
- Die im Beispiel durchgeführte Zerlegung ist nicht verlustlos: verloren gegangen ist die Information, dass Pearl das Buch 2-305 für IT-Systeme, aber nicht für Datenbanken empfiehlt.
- Mit den drei zweistelligen Beziehungstypen können auch Zusammenhänge abgebildet werden, die vorher in der dreistelligen Relation nicht möglich waren: ein Buch (1-100) wird für eine Vorlesung (Web-Services) empfohlen, für die keine Professorin angegeben ist.
- Manche Datenbanksysteme bieten für die Modellierung nur binäre Beziehungstypen an.

Existenzabhängige Entity-Typen

Bisher vorausgesetzt: Entities existieren autonom und sind innerhalb ihres Typs über die Schlüsselattribute eindeutig identifizierbar.

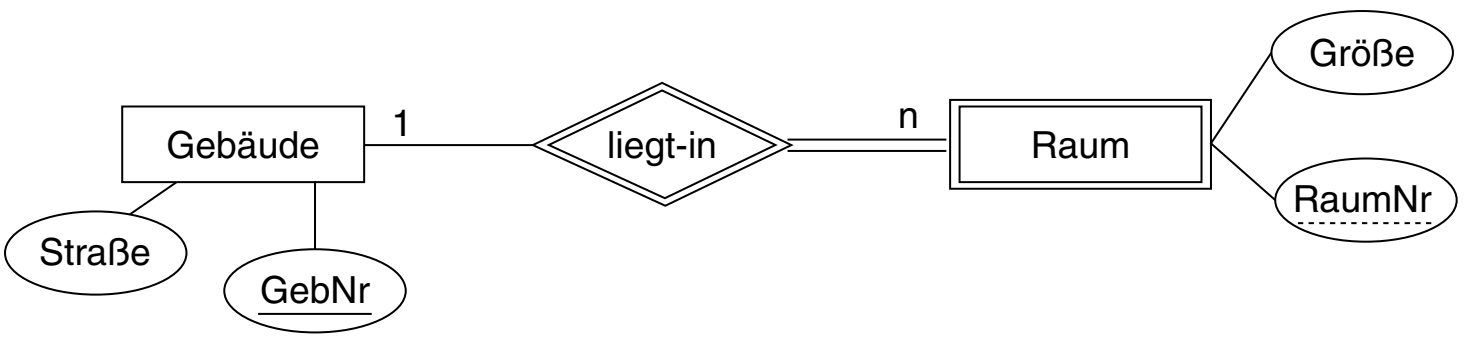
Existenzabhängige Entity-Typen

Bisher vorausgesetzt: Entities existieren autonom und sind innerhalb ihres Typs über die Schlüsselattribute eindeutig identifizierbar.

Aus Modellierungssicht sind auch Entity-Typen sinnvoll, die

1. von einem anderen, übergeordneten Entity-Typ abhängig sind,
2. nur in Kombination mit dem Schlüssel des übergeordneten Entity-Typs eindeutig identifizierbar sind.

Beispiel:



Bemerkungen:

- Existenzabhängige Entity-Typen werden auch als *schwache* Entity-Typen bezeichnet.
- Existenzabhängige Entity-Typen werden durch doppelt umrandete Rechtecke, die Beziehung zu dem übergeordneten Entity-Typ durch eine doppelt umrandete Raute und die zugehörige Kante durch eine Doppellinie repräsentiert.
- Die Beziehung eines existenzabhängigen Entity-Typs zu dem übergeordneten Entity-Typ ist meist n:1, manchmal 1:1, aber nie n:m.
- Existenzabhängige Entity-Typen haben im Allgemeinen keinen Schlüssel, der alle Entities eindeutig identifiziert. Statt dessen gibt es ein Attribut (bzw. eine Menge von Attributen), das alle existenzabhängigen Entities, die *einem* übergeordneten Entity zugeordnet sind, voneinander unterscheidet. Diese Attribute werden im ER-Diagramm gestrichelt unterstrichen.

Abstraktionskonzepte

Generalisierung / Spezialisierung

Die Generalisierung wird im konzeptuellen Entwurf eingesetzt, um eine bessere – im Sinne von „natürlichere“ – Strukturierung der Entity-Typen zu erzielen.

Abstraktionskonzepte

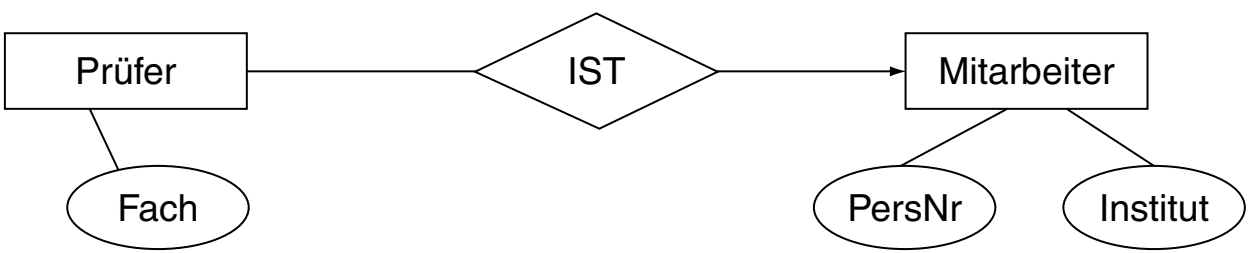
Generalisierung / Spezialisierung

Die Generalisierung wird im konzeptuellen Entwurf eingesetzt, um eine bessere – im Sinne von „natürlichere“ – Strukturierung der Entity-Typen zu erzielen.

Vorgehensweise:

- Die Eigenschaften ähnlicher Entity-Typen werden „faktorisiert“ und einem gemeinsamen *Obertyp* zugeordnet.
- Eigenschaften, die nicht faktorisierbar sind, verbleiben beim jeweiligen *Untertyp*, der somit eine Spezialisierung des Obertyps darstellt.

Beispiel:



Prüfer(*Institut*, *PersNr* , *Fach*)

geerbt von Mitarbeiter

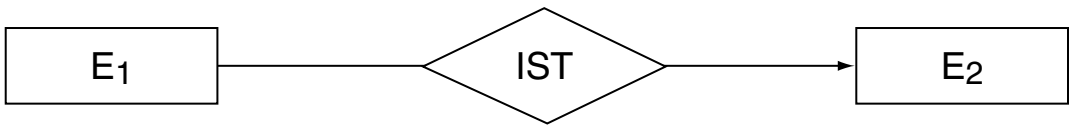
Bemerkungen:

- Im Beispiel ist „Mitarbeiter“ der generellere Obertyp und „Prüfer“ der speziellere Untertyp.
- Ein wichtiges Prinzip der Generalisierung ist die Vererbung: ein Untertyp erbt alle Eigenschaften des Obertyps.
- Bei der Übersetzung in das Relationenmodell werden Generalisierungsbeziehungen besonders behandelt.

Abstraktionskonzepte

Generalisierung / Spezialisierung (Fortsetzung)

Die Entities eines Untertyps sind gleichzeitig Entities des Obertyps; es handelt sich also um eine Teilmengenbeziehung: $\mathit{state}(E_1) \subseteq \mathit{state}(E_2)$



Schreibweise: E_1 IST E_2

Hinsichtlich der Teilmengensicht sind zwei Fälle von besonderem Interesse:

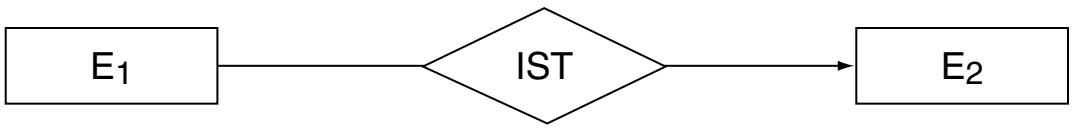
1. Disjunkte Spezialisierung

2. Vollständige Spezialisierung

Abstraktionskonzepte

Generalisierung / Spezialisierung (Fortsetzung)

Die Entities eines Untertyps sind gleichzeitig Entities des Obertyps; es handelt sich also um eine Teilmengenbeziehung: $\mathit{state}(E_1) \subseteq \mathit{state}(E_2)$



Schreibweise: E_1 IST E_2

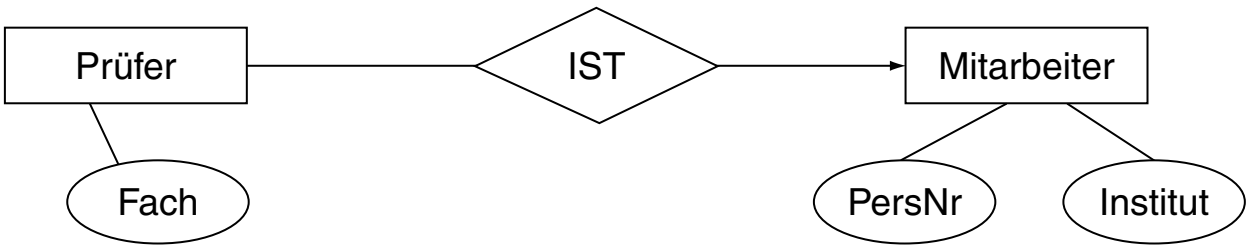
Hinsichtlich der Teilmengensicht sind zwei Fälle von besonderem Interesse:

1. Disjunkte Spezialisierung
   Die Mengen der Entity-Untertypen eines Entity-Obertyps sind paarweise disjunkt.
2. Vollständige Spezialisierung
   Die Menge des Entity-Obertyps enthält keine unspezialisierten Elemente sondern ergibt sich vollständig durch die Vereinigung der Mengen der Entity-Untertypen.

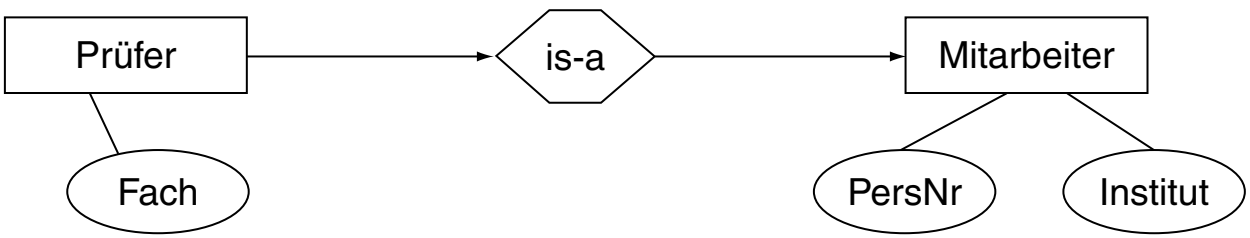
Abstraktionskonzepte

Generalisierung / Spezialisierung (Fortsetzung)

Graphische Notation als Standard-Beziehungstyp:

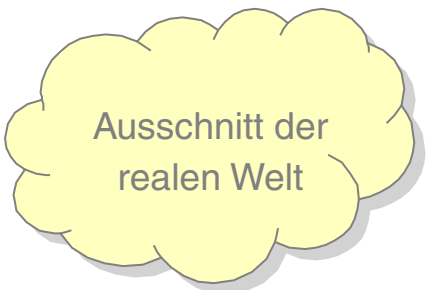


Alternative graphische Notation:



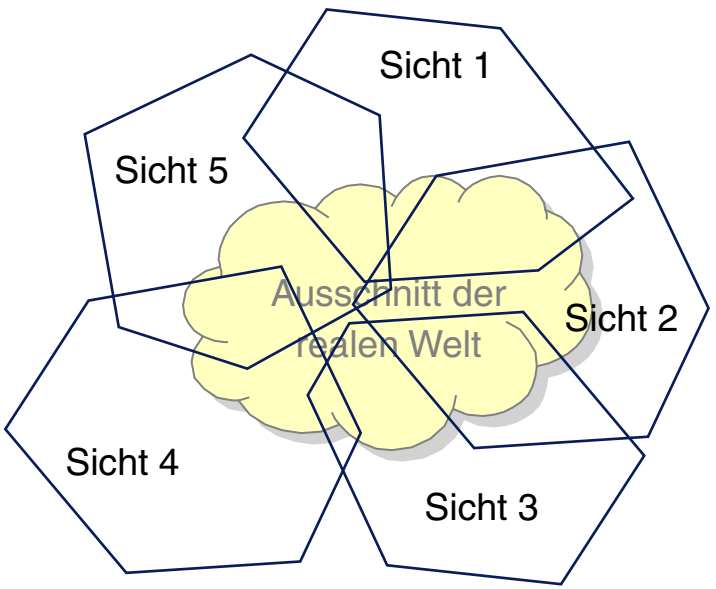
Konsolidierung, Sichtenintegration

Modellierungsproblematik



Konsolidierung, Sichtenintegration

Modellierungsproblematik



Konsolidierung

globales Schema

- redundanzfrei
- widerspruchsfrei
- Synonyme bereinigt
- Homonyme bereinigt
- ...

Bemerkungen: [DB:II Entwurfsprozess > Konzeptueller Entwurf]

- Bei größeren Anwendungen ist es nicht praktikabel, den konzeptuellen Entwurf in einem Guss durchzuführen; sinnvoll ist die Aufteilung in verschiedene Anwendersichten.
- Konsolidierung bedeutet die Zusammenfassung einzelner Sichten zu einem globalen Schema, das u.a. redundanz- und widerspruchsfrei ist.
- Bei der Konsolidierung einer größeren Anwendung sollte man schrittweise vorgehen, so dass man jeweils nur zwei Teilschemata gleichzeitig betrachtet.
- Arten von Widersprüchen, die bei einer Konsolidierung aufzulösen sind:
  - unterschiedliche Benennung gleicher Sachverhalte (Synonyme)
  - gleiche Benennung unterschiedlicher Sachverhalte (Homonyme)
  - Sachverhalt einmal als Entity-Typ und ein andermal als Beziehungstyp modelliert (struktureller Widerspruch)
  - widersprüchliche Funktionalitätsangaben
  - widersprüchliche Datentypen, widersprüchliche Schlüsselattribute

Konsolidierung, Sichtenintegration

Beispiel: drei Sichten einer Universitätsdatenbank

Erstellung von
Dokumenten

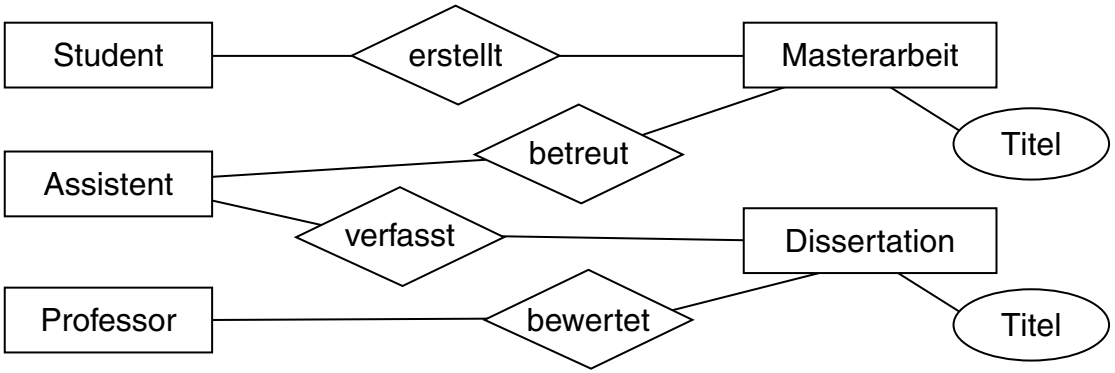
Bibliotheks
-verwaltung

Buchempfehlungen

Konsolidierung, Sichtenintegration

Beispiel: drei Sichten einer Universitätsdatenbank

Erstellung von Dokumenten



Bibliotheks -verwaltung

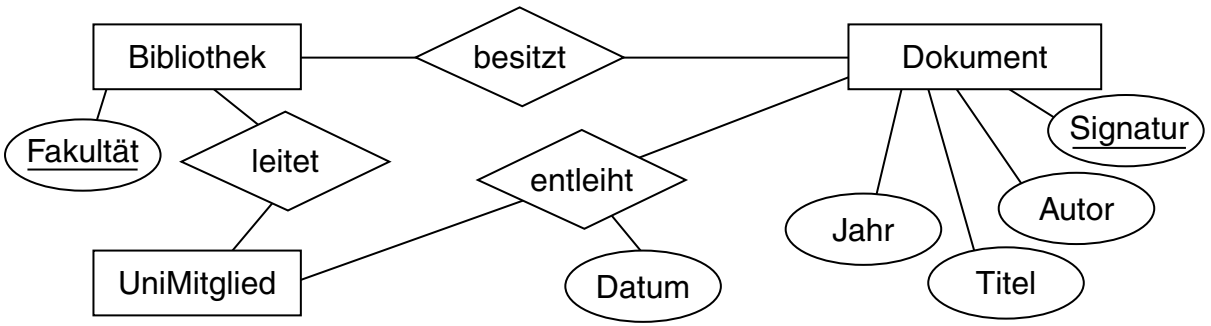
Buchempfehlungen

Konsolidierung, Sichtenintegration

Beispiel: drei Sichten einer Universitätsdatenbank

Erstellung von Dokumenten

Bibliotheks -verwaltung



Buchempfehlungen

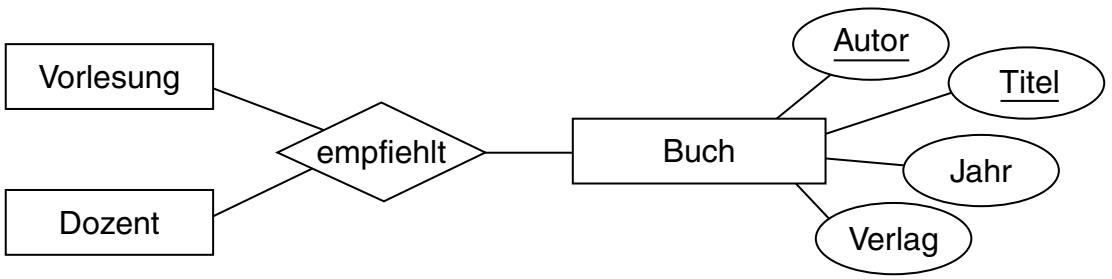
Konsolidierung, Sichtenintegration

Beispiel: drei Sichten einer Universitätsdatenbank

Erstellung von
Dokumenten

Bibliotheks
-verwaltung

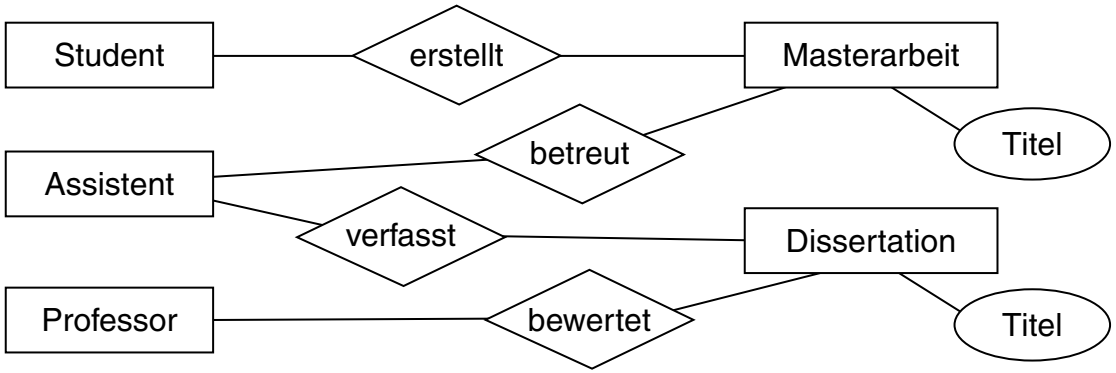
Buchempfehlungen



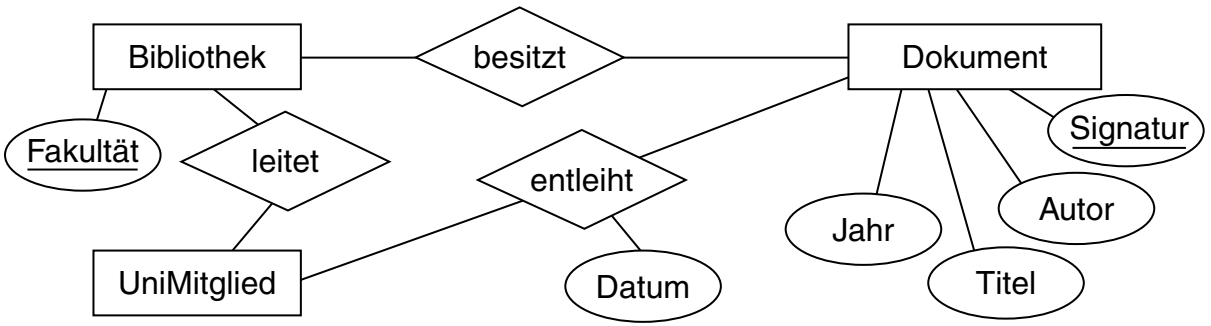
Konsolidierung, Sichtenintegration

Beispiel: drei Sichten einer Universitätsdatenbank

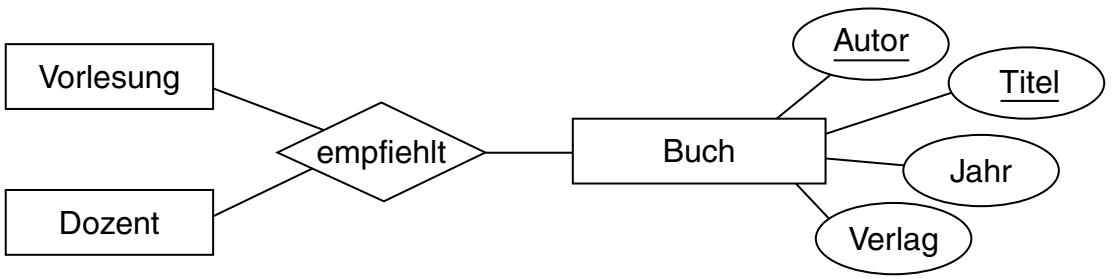
Erstellung von Dokumenten



Bibliotheks -verwaltung



Buchempfehlungen



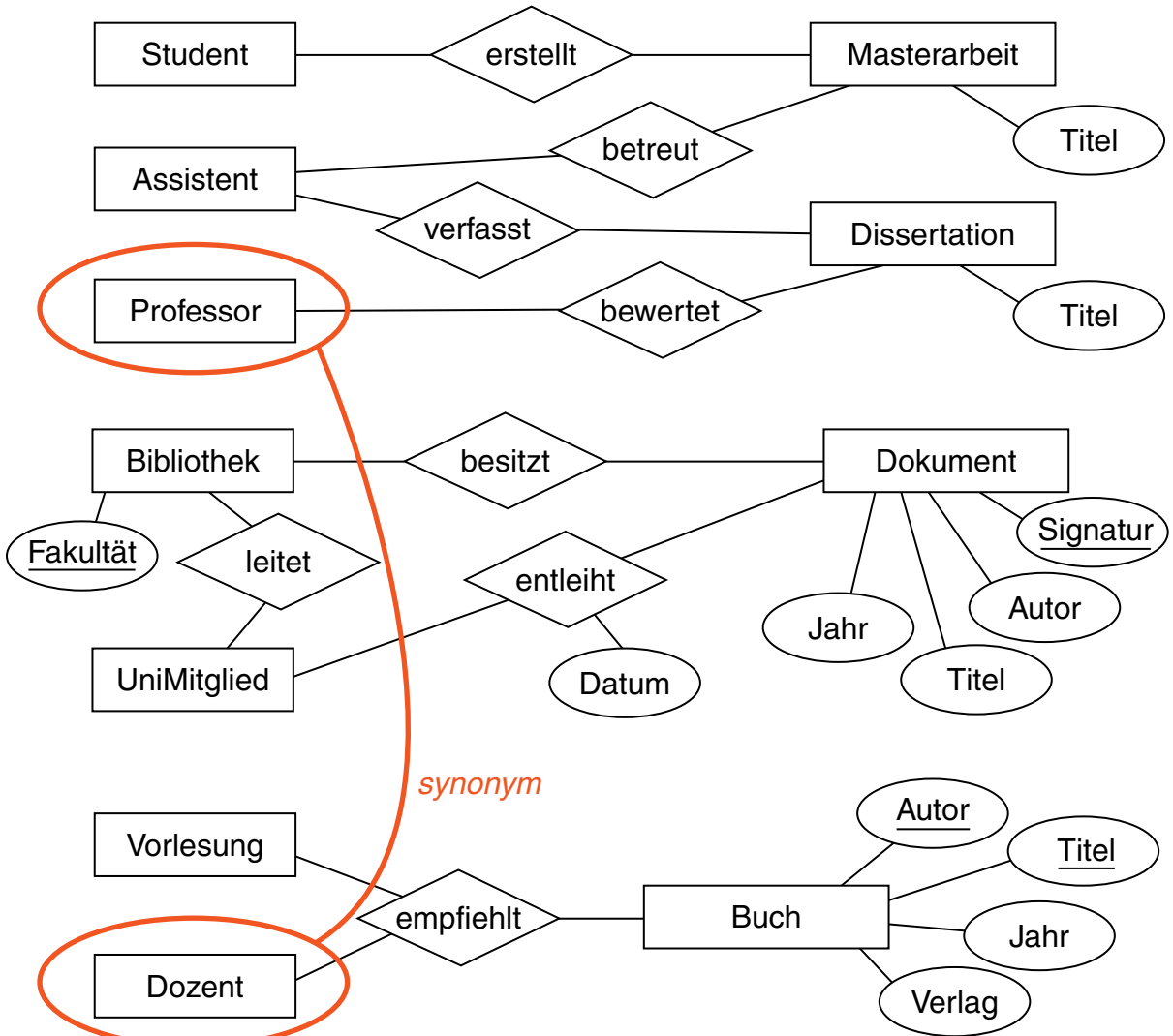
Konsolidierung, Sichtenintegration

Beispiel: drei Sichten einer Universitätsdatenbank

Erstellung von Dokumenten

Bibliotheks -verwaltung

Buchempfehlungen



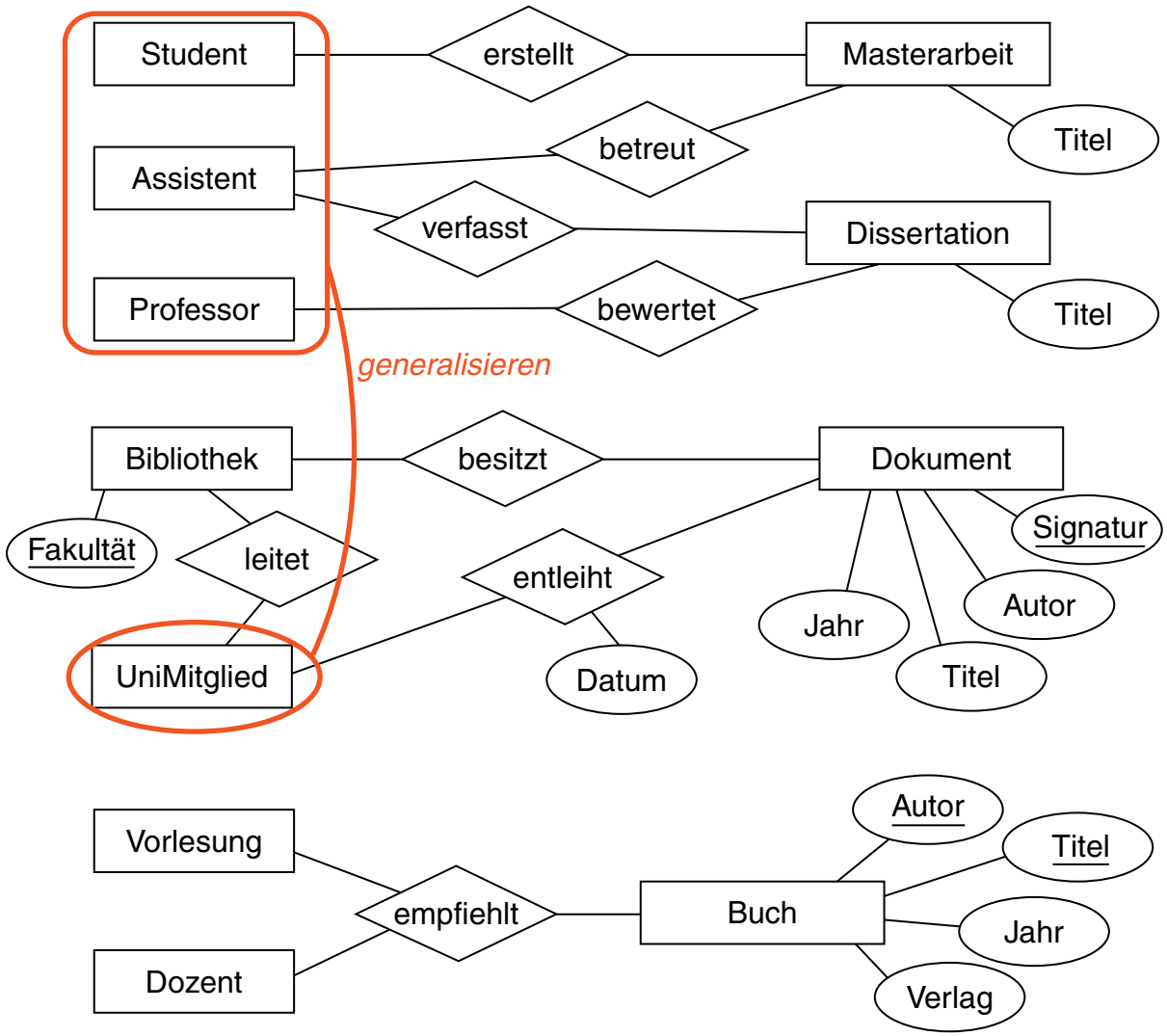
Konsolidierung, Sichtenintegration

Beispiel: drei Sichten einer Universitätsdatenbank

Erstellung von Dokumenten

Bibliotheks -verwaltung

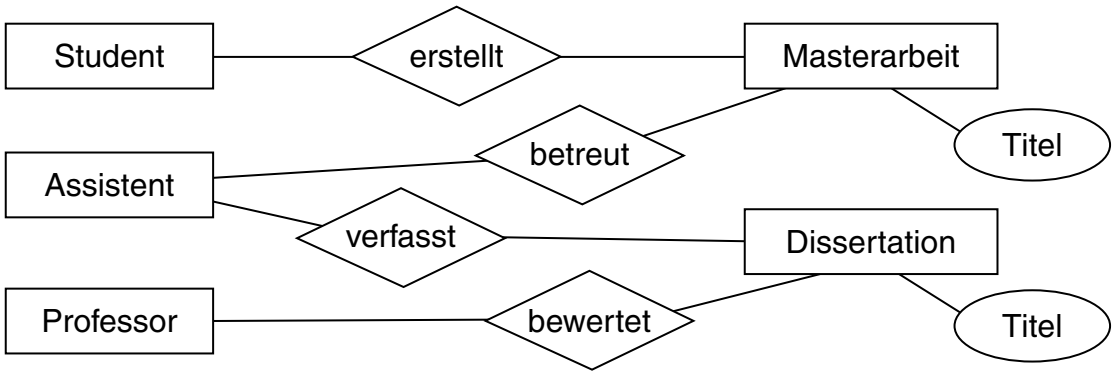
Buchempfehlungen



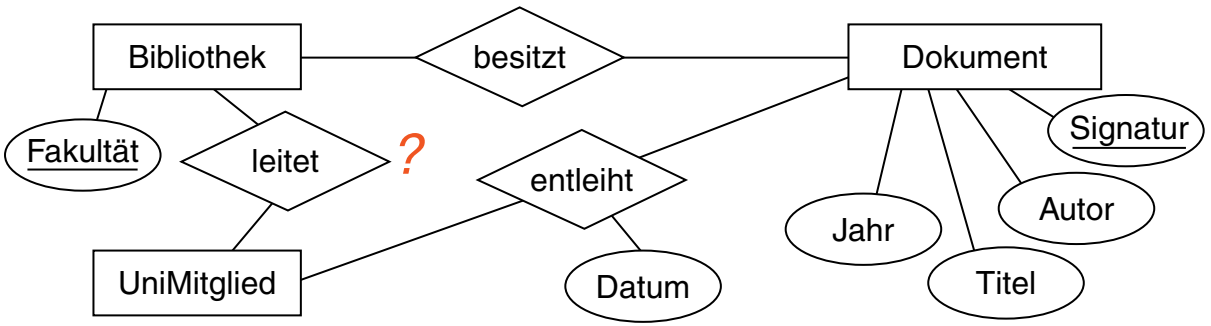
Konsolidierung, Sichtenintegration

Beispiel: drei Sichten einer Universitätsdatenbank

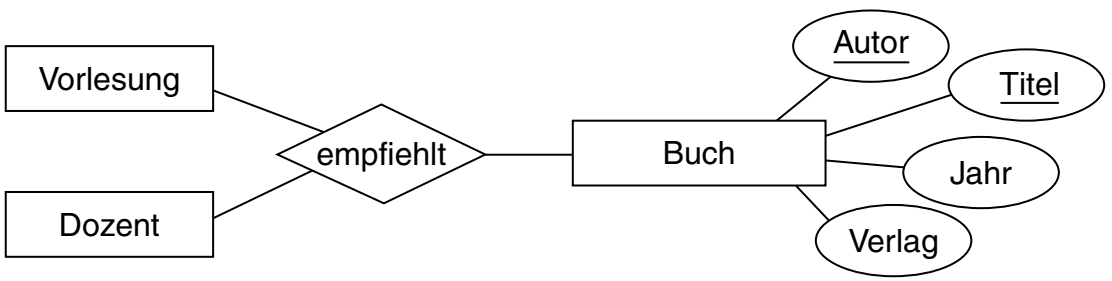
Erstellung von Dokumenten



Bibliotheks -verwaltung



Buchempfehlungen



Bemerkungen:

- Aufgabe ist die Konsolidierung dieser Sichten in *ein* Schema.
- Die Entity-Typen „Dozent“ und „Professor“ sind synonym verwendet worden.
- Der Entity-Typ „UniMitglied“ ist eine Generalisierung der Entity-Typen „Student“, „Professor“ und „Assistent“.
- Fakultätsbibliotheken werden von Angestellten und nicht von Studierenden geleitet: der Beziehungstyp „leitet“ in Sicht 2 sollte revidiert werden.
- Die Entity-Typen „Dissertation“, „Masterarbeit“ und „Buch“ sind Spezialisierungen des Entity-Typs „Dokument“.
- Alle an der Universität erstellten Diplomarbeiten und Dissertationen werden in Bibliotheken verwaltet.
- Die Beziehungtypen „erstellt“ und „verfasst“ in Sicht 1 modellieren denselben Sachverhalt wie das Attribut „Autor“ vom Entity-Typ „Dokument“ in Sicht 2.
- Alle in einer Bibliothek verwalteten Dokumente werden durch die Signatur identifiziert.