Entwurfsaufgabe

- Anforderungen an Entwurfsprozeß
  - *Informationserhalt*
  - *Konsistenzerhaltung*
  - *Redundanzfreiheit*
  - *Vollständigkeit bezüglich Anforderungsanalyse*
  - *Konsistenz des Beschreibungsdokuments*
  - *Ausdrucksstärke, Verständlichkeit des benutzten Formalismus*
  - *Formale Semantik der Beschreibungskonstrukte*
  - *Lesbarkeit der Dokumente*
  - Weitere Qualitätseigenschaften: Unterstützung von Erweiterbarkeit, Modularisierung, Wiederverwendbarkeit sowie Werkzeugunterstützung etc.

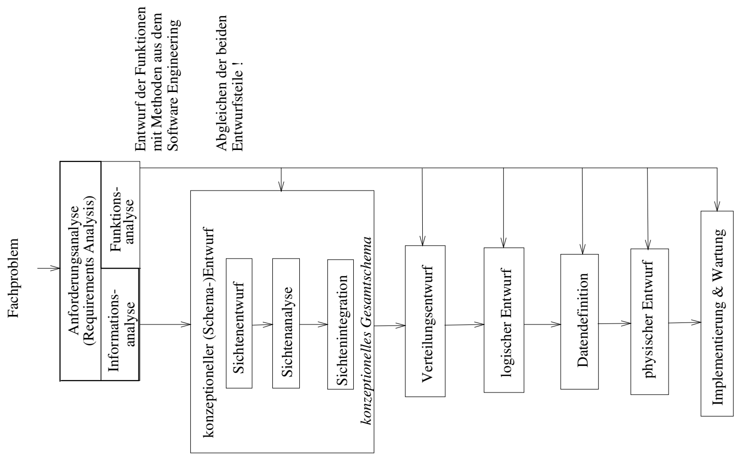
Phasen des Datenbankentwurfs

- *Anforderungsanalyse*
- *Konzeptioneller Entwurf*
- *Verteilungsentwurf*
- *Logischer Entwurf*
- *Datendefinition*
- *Physischer Entwurf*
- *Implementierung und Wartung*

5. Datenbankentwurf

- ➠ Entwurfsaufgabe
- ➠ Phasenmodell
- ➠ Konzeptioneller Entwurf
- ➠ ER-Abbildung auf andere Datenbankmodelle

Phasenmodell



Anforderungsanalyse

- **Vorgehensweise:** Sammlung des Informationsbedarfs in den Fachabteilungen
- **Ergebnis:**
  - informale Beschreibung (Texte, tabellarische Aufstellungen, Formblätter, usw.) des Fachproblems
  - Trennen der Information über Daten (Datenanalyse) von den Information über Funktionen (Funktionsanalyse)
- **„Klassischer“ DB-Entwurf:**
  nur Datenanalyse und Folgeschritte.
- **Funktionsentwurf:**
  siehe Methoden des Software Engineering

Konzeptioneller Entwurf

- erste formale Beschreibung des Fachproblems, *Sprachmittel:* semantisches Datenmodell, z.B. erweitertes ER-Modell
- **Vorgehensweise:**
  - Modellierung von Sichten z.B. für verschiedene Fachabteilungen
  - Analyse der vorliegenden Sichten in Bezug auf Konflikte
  - Integration der Sichten in ein Gesamtschema
- **Ergebnis:** konzeptionelles Gesamtschema, z.B. EER-Diagramm

Konflikte

- **Namenskonflikte**: Homonyme / Synonyme
  - Homonyme: Schloß; Kunde
  - Synonyme: Auto, KFZ, Fahrzeug
- **Typkonflikte**: verschiedene Strukturen für das gleiche Element
- **Wertebereichskonflikte**: verschiedene Wertebereiche für ein Element
- **Bedingungskonflikte**: z.B. verschiedene Schlüssel für ein Element
- **Strukturkonflikte**: gleicher Sachverhalt durch unterschiedliche Konstrukte ausgedrückt

Verteilungsentwurf

Sollen die Daten auf mehreren Rechnern verteilt vorliegen, muß Art und Weise der *verteilten Speicherung* festgelegt werden.

z.B. bei einer Relation `KUNDE (KNr, Name, Adresse, PLZ, Kontostand)`

- **horizontale** Verteilung :
  `KUNDE_1 (KNr, Name, Adresse, PLZ, Kontostand)` **where** `PLZ < 50.000`
  und
  `KUNDE_2 (KNr, Name, Adresse, PLZ, Kontostand)` **where** `PLZ >= 50.000`
- **vertikale** Verteilung :
  `KUNDE_Adr (KNr, Name, Adresse, PLZ)`
  und
  `KUNDE_Konto (KNr, Kontostand)`
  (Verbindung über `KNr` Attribut)

Datendefinition

Umsetzung des logischen Schemas in ein konkretes Schema

Sprachmittel: DDL und DML eines DBMS z.B. Ingres, Oracle

- Datenbankdeklaration in der DDL des DBMS
- Realisierung der Integritätssicherung
- *Definition der Benutzersichten*

Implementierung und Wartung

Phasen

- der Wartung,
- der weiteren Optimierung der physischen Ebene,
- der Anpassung an neue Anforderungen und Systemplattformen,
- der Portierung auf neue Datenbank-Management-Systeme
- etc.

Logischer Entwurf

- *Sprachmittel:* Datenmodell des ausgewählten „Realisierungs"-DBMS z.B. relationales Modell
- *Vorgehensweise:*
  1. (automatische) Transformation des konzeptionellen Schemas z.B. ER $\rightarrow$ relationales Modell
  2. Verbesserung des relationalen Schemas anhand von Gütekriterien (Normalisierung, siehe Kapitel 6):
     Entwurfsziele: Redundanzvermeidung, ...
- *Ergebnis:* logisches Schema, z.B. Sammlung von Relationenschemata

Physischer Entwurf

Ergänzen des physischen Entwurfs um Zugriffsunterstützung bzgl. Effizienzverbesserung, z.B. Definition von Indexen (Indizes)

Sprachmittel: Speicherstruktursprache SSL

Phasenbegleitende Methoden

Validationsmethoden:

Verifikation: Der formale Beweis etwa von Schemaeigenschaften

Prototyping: beispielhaftes Arbeiten mit der Datenbank vor der endgültigen Implementierung

Validation mit Testdaten: Überprüfung der Richtigkeit des Entwurfs anhand von realen oder künstlichen Testdaten

Konzeptioneller Entwurf II

$$S = (O, D, E, A)$$

E *Entwicklungsschicht*: Bedingungen über die zeitliche Entwicklung des Datenbestands.

zeitliche Entwicklungen formalisiert durch Zustandsfolgen:

$$\hat{\sigma} =< \sigma_0, \ldots , \sigma_i, \ldots >$$

A *Aktionsschicht*: implementierungsunabhängige Beschreibungen von Aktionen

Zustandsübergänge $\sigma_i \mapsto \sigma_{i+1}$

Objektorientierte Entwurfsmethoden

Integration von Funktions- und Strukturbeschreibung in Objektbeschreibungen

- Strukturbeschreibung analog OODM
- abstrakte *Ereignisse* / *Methoden* zur Funktions- / Verhaltensmodellierung

Konzeptioneller Entwurf

$$S = (O, D, E, A)$$

O *Objektschicht*: Objekte, Objekttypen und ihre Beziehungen, Integritätsbedingungen.

D *Datenschicht*: Wertebereiche von Attributen

O und D beschreiben *statische Struktur* der Datenbank, d.h. die möglichen Zustände σ der Anwendung.

Datenschicht und Objektschicht legen ein konzeptionelles *Datenbankschema* fest (bzw. den *statischen* Anteil davon).

Konzeptioneller Entwurf III

E und A beschreiben das *dynamische Verhalten*, d.h. die zulässigen (erlaubten und realisierbaren) Zustandsfolgen, mit

$$< \sigma_0, \ldots, \sigma_i, \ldots, \sigma_n > \text{ ist erlaubt}$$

und

$$\sigma_i \mapsto \sigma_{i+1}, \quad i = 0, \ldots, n-1 \text{ ist realisierbar.}$$

Zusätzlich möglich: *Prozeßschicht* P für langfristige Abläufe

Schichten des konzeptionellen Schemas

<table>
<tr><td>Entwicklungsschicht
Temporale Logik</td><td>Aktionsschicht
Vor- und Nachbedingungen</td></tr>
<tr><td colspan="2">Objektschicht
Semantische Datenmodelle</td></tr>
<tr><td colspan="2">Datenschicht
Abstrakte Datentypen</td></tr>
</table>

Objektschicht

Beschreibung etwa im

EER-Modell

...

Abstraktionskonzepte in semantischen Datenmodellen

1. *Typisierung* / *Klassifizierung*:
   Objekte mit gleichen Eigenschaften werden in Objekttypen zusammengefaßt
2. *Generalisierung* bzw. *Spezialisierung*:
   Objekttypen in einer Vererbungshierarchie
3. *Sammlung*:
   Objekte als Menge von anderen Objekten, etwa `Team` als Sammlung von `Mitarbeiter`n
4. *Aggregierung*:
   komplexe Objekte zusammengesetzt aus anderen Teilobjekten

Werte versus Objekte

- Eigenschaften von Werten sind nicht änderbar
- Werte identifizieren sich selber
- Werte sind nur als Eigenschaften von Objekten in einer Datenbank gespeichert
- Werte können nicht isoliert (d.h., ohne Eigenschaft eines Objekts zu sein) eingefügt oder gelöscht werden
- Eine Operation auf Werten ändert nicht den Zustand eines Wertes (wie es bei Objekten der Fall sein kann), sondern liefert einen neuen Wert

Beispiel für temporale Integritätsbedingungen

`DP(MatrNr, Fach, Semester, Note, Versuch)`

- Es gibt maximal zwei Versuche pro Student und Fach:

$$\mathbf{always}(\mathtt{DP}(m,f,s,n,v) \implies (v \in \{1,2\}))$$

- Prüfungsergebnisse sind endgültig:

$$\mathbf{always}((\mathtt{DP}(m,f,s,n,v)) \implies \mathbf{always}(\mathtt{DP}(m,f,s,n,v)))$$

Datenschicht

```
datatype point based on real;
sorts point;
operations distance :        (point × point):  real;
           xcoord, ycoord :  (point):  real;
           createpoint :     (real × real):  point;
           add :             (point × point):  point;
            ...
variables p,q :  point;
          x,y,x1,y1 :  real;
equations
   x = xcoord(createpoint(x,y));
   y = ycoord(createpoint(x,y));
   distance(createpoint(x,y),createpoint(x1,y1))
     = sqrt((x-x1)*(x-x1) + (y-y1)*(y-y1));
   add(p,q)
     = createpoint(xcoord(p)+xcoord(q),ycoord(p)+ycoord(q));
```


Entwicklungsschicht

Angabe von *transitionalen* und *temporalen* Integritätsbedingungen

Oberbegriff: *dynamische* Integritätsbedingungen

$$\forall(a : \mathtt{Angestellte})\ \ a.\mathtt{Gehalt} > 2000$$

$$\forall(s : \mathtt{Sparbuch})\ \ \mathbf{new}(s).\mathtt{Betrag} \geq \mathbf{old}(s).\mathtt{Betrag} - 2000$$

$$\forall(A : \mathtt{Ang})\forall(s : \mathbf{int})\ \ \mathbf{always}((A.\mathtt{Gehalt} = s) \implies \mathbf{always}\neg(A.\mathtt{Gehalt} < s))$$

Aktionsschicht

```
action Exmatrikulation (StudentName:  string,
                        MatrikelNr:  integer):
  pre
    (exists s:  Student)
      s.Name=StudentName and s.MatrNr=MatrikelNr and
      not (exists b:  Buch) Ausleihe(s,b);
  post
    not (exists s:  Student)
      s.Name=StudentName and s.MatrNr=MatrikelNr;
end action Exmatrikulation.
```


ER-Abbildung auf andere Datenbankmodelle

- Erster Teilschritt des logischen Datenbankentwurfs
- Abbildung von ER-Modell auf
  - Relationenmodell
  - Netzwerkmodell
  - hierarchisches Modell
- Vorgehensweisen:
  - Transformation nach Faustregeln manuell
  - automatische Transformation

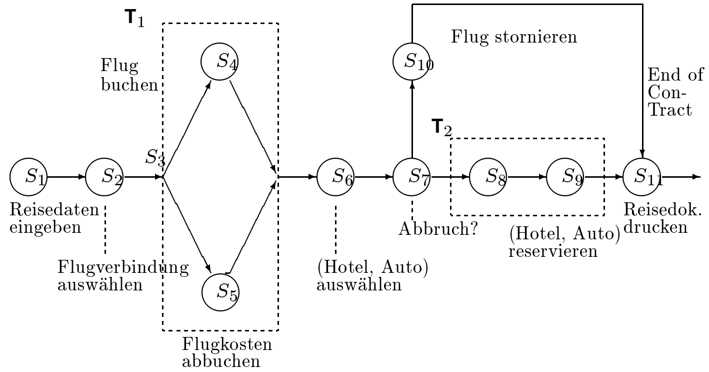
Ziel: kapazitätserhaltende Abbildung

Beispiel für temporale Integritätsbedingungen II

- Ein zweiter Versuch kann nur durchgeführt werden, wenn ein erster Versuch erfolglos vor dem achten Semester durchgeführt wurde (Freischuß):

$$\mathbf{always}(\mathtt{DP}(m,f,s,n,2) \implies \mathbf{sometimeafter}(\mathtt{DP}(m,f,s',5,1) \wedge s' < 8))$$

Modellierung von Anwendungsprozessen



Beispiel einer Ablaufspezifikation im ConTract-Modell

Kapazitätsvermindernde Abbildung von ER-Diagrammen auf relationale Schemata

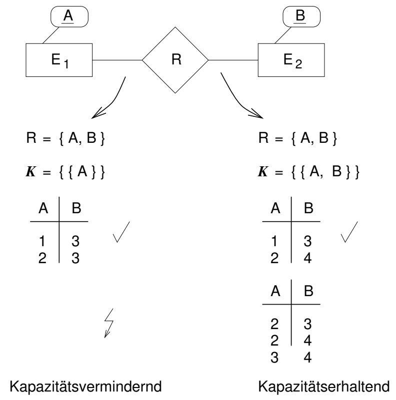


Abbildung ER-Schema nach RDM

ER-Konzept	**wird abgebildet auf relationales Konzept**
Entity-Typ E_i	Relationenschema R_i
Attribute von E_i	Attribute von R_i
Primärschlüssel P_i	Primärschlüssel P_i
Beziehungstyp	Relationenschema
	Attribute: P_1, P_2
dessen Attribute	weitere Attribute
$1 : n$	P_2 wird Primärschlüssel der Beziehung
$1 : 1$	P_1 und P_2 werden Schlüssel der Beziehung
$m : n$	$P_1 \cup P_2$ wird Primärschlüssel der Beziehung
IST-Beziehung	R_1 erhält zusätzlichen Schlüssel P_2

E_1, E_2: an Beziehung beteiligte Entity-Typen, P_1, P_2: deren Primärschlüssel,
$1 : n$-Beziehung: E_2 ist n-Seite, IST-Beziehung: E_1 ist speziellerer Entity-Typ

Kapazitätserhöhende Abbildung von ER-Diagrammen auf relationale Schemata

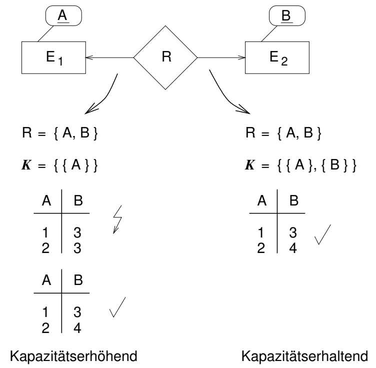


Abbildung auf das relationale Modell

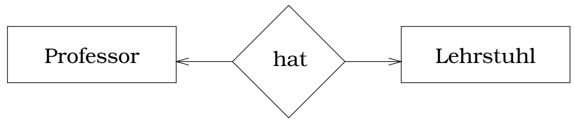
- Entity-Typen und Beziehungstypen $\rightarrow$ Relationenschemata
  - Attribute $\rightarrow$ Attribute des Relationenschemas
  - Schlüssel werden übernommen
- Kardinalitäten der Beziehungen $\rightarrow$ Wahl der Schlüssel
- Relationenschemata von Entity- und Beziehungstypen können eventuell miteinander verschmolzen werden
- Einführung diverser Fremdschlüsselbedingungen

Abbildung von Beziehungstypen

- Beziehungstyp → Relationenschema mit allen Attributen des Beziehungstyps + Primärschlüssel der beteiligten Entity-Typen
- Auswahl der Schlüssel (hier für binäre Beziehungen)
  - m:n-Beziehung: Beide Primärschlüssel werden Schlüssel
  - 1:n-Beziehung: Der Primärschlüssel der n-Seite (bei der funktionalen Notation die Seite ohne Pfeilspitze) wird Schlüssel
  - 1:1-Beziehung: Beide Primärschlüssel werden je ein Schlüssel, einer wird Primärschlüssel

Dies gilt bei optionalen Beziehungen ([0,_])

1:1-Beziehung



- `Professoren` mit den Attributen `PANr` und `Stufe`,
- `Lehrstühle` mit den beiden Attributen `Lehrstuhlbezeichnung` und `Anzahl_Planstellen` und
- `Hat_Lehrstuhl` mit den Primärschlüsseln der beiden beteiligten Entity-Typen jeweils als Schlüssel dieses Schemas, also `PANr` und `Lehrstuhlbezeichnung`.

Abbildung von Entity-Typen

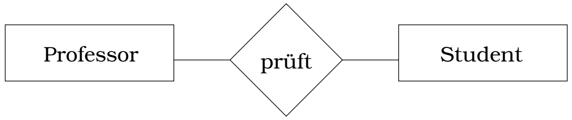
- Entity-Typ → Relationenschema mit allen Attributen des Entity-Typs
- mehrere Schlüssel vorhanden → Auswahl eines Primärschlüssels

Verschmelzen von Relationenschemata

Bei zwingenden Beziehungen ([1,_])

- 1:n-Beziehung: das Entity-Relationenschema der n-Seite kann in das Relationenschema der Beziehung integriert werden
- 1:1-Beziehung: beide Entity-Relationenschemata können in das Relationenschema der Beziehung integriert werden

n:m-Beziehung



- Professoren mit den Attributen PANr und Stufe
- Studenten unter anderem mit den Attributen Matrikelnummer und Studienfach
- Prüft mit den Primärschlüsseln der beteiligten Entity-Typen zusammen als Primärschlüssel dieses Schemas, also {PANr, Matrikelnummer}

Fremdschlüssel?

IST-Beziehung II

- Mitarbeiter mit AngNr als Schlüssel. Zusätzlich Primärschlüssel PANr von Personen geerbt. Entscheidung für PANr als Primärschlüssel
- Professoren: PANr wird von Mitarbeiter vererbt
- Studenten mit Attribut Matrikelnummer (Schlüssel). Auswahl zwischen "lokalem" Schlüssel und geerbtem Schlüssel PANr

1:1-Beziehung: Auswirkung von [1,1] - Kardinalitäten

[1,1]:[1,1]-Beziehung

Professoren

PANr	Lehrstuhlbezeichnung	Stufe	Anzahl_Planstellen
4711	Datenbank- und Informationssysteme	C4	4
5588	Datenbanken und Informationssysteme	C4	5

[0,1]:[1,1]-Beziehung: Lehrstühle können unbesetzt bleiben

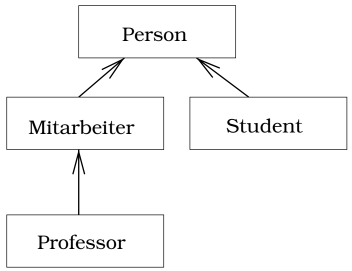
Professoren

PANr	Lehrstuhlbezeichnung	Stufe	Anzahl_Planstellen
4711	Datenbank- und Informationssysteme	C4	4
5588	Datenbanken und Informationssysteme	C4	5
⊥	Rechnernetze	⊥	2

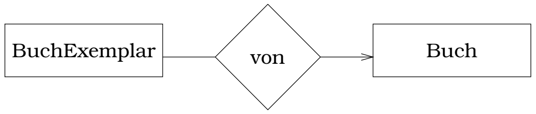
dann besser zwei Relationenschemata

IST-Beziehung

- kein eigenes Relationenschema
- im Relationenschema des spezielleren Entity-Typs zusätzlich der Primärschlüssel des allgemeineren Entity-Typs



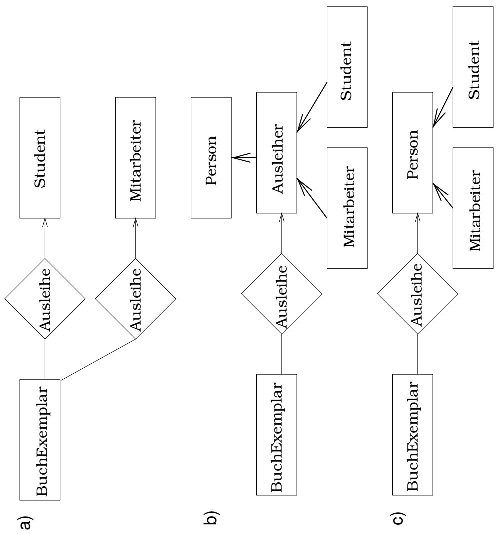
Komplexere Beispiele: 1:n-Beziehung



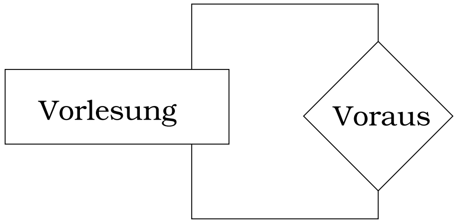
- `Buch_Exemplare` mit dem Attribut `Inventarnr`
- `Bücher` unter anderem mit den Attributen `ISBN` und `Titel`
- `von` mit dem Primärschlüssel der n-Seite `Buch_Exemplare` als Primärschlüssel dieses Schemas

Relationenschema `Buch_Exemplare` kann mit dem Relationenschema `von` verschmolzen werden (zwingende Beziehung)

Auswirkungen der fehlenden Generalisierung im klassischen ER-Modell



Komplexere Beispiele: Rekursive Beziehungen



Umbenennung der übernommenen Primärschlüssel

Komplexere Beispiele: Mehrstellige Beziehungen

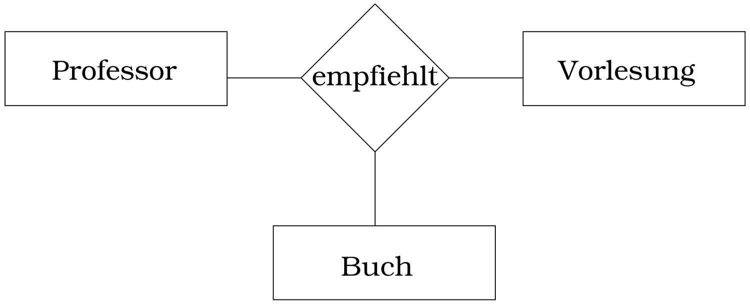


Abbildung auf das Netzwerkmodell II

- 1:1-Beziehungen wie 1:n-Beziehung
- IST-Beziehungen wie 1:n-Beziehung
- Beziehungsattribute →
  - Attribute des "Kett-Record-Typen"
  - Attribut beim Record-Typ der Member-Seite (für 1:n-Beziehungen)

Die `Empfiehlt`-Beziehung als Netzwerkschema

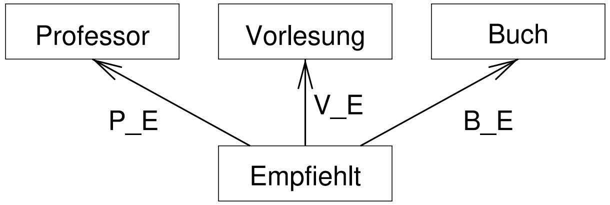


Abbildung auf das Netzwerkmodell

- Entity-Typ → Record-Typ mit allen Attributen als Feldern
- Beziehungstyp → Set-Typ *nur wenn*
  - Beziehung ist 1:n-Beziehung
  - Beziehung ist binär
  - Beziehung hat keine eigenen Attribute
- allgemeine Beziehungen → "Kett-Record-Typen"

Abbildung eines ER-Schemas auf ein Netzwerkschema

ER-Konzept	**wird abgebildet auf Netzwerk-Konzept**
Entity-Typ E_i	Record-Typ R_i
Attribute von E_i	Felder von R_i
Beziehungstyp	Set-Typ
dessen Attribute	—; evtl. Kett-Record-Typ oder bei Original-Record-Typ
$1:n$	Standard-Set-Typ
$1:1$	Standard-Set-Typ; Zusatzbedingung nicht darstellbar
$m:n$	Kett-Record-Typ
IST-Beziehung	Standard-Set-Typ; Zusatzbedingung nicht darstellbar

Bezeichnungen:
E_1, E_2: an Beziehung beteiligte Entity-Typen

Hierarchisches Schema mit virtuellen Record-Typen

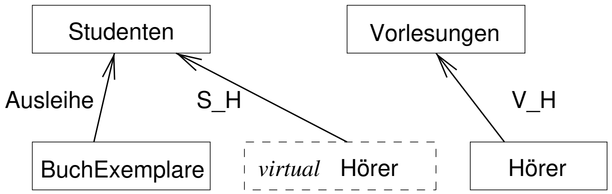
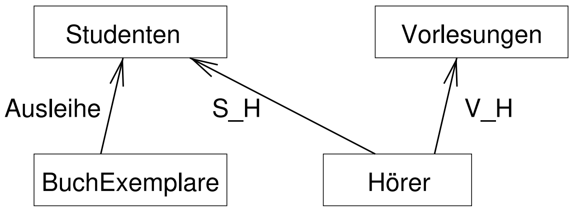


Abbildung auf das hierarchische Modell



Vorläufiges Schema für die Transformation in das hierarchische Modell

Für m:n-Beziehungen optimiertes hierarchisches Schema