4. Datenbankentwurf, Datendefinition

Entwurfsaufgabe

1. Anwendungsdaten sollen (effizient) aus gespeicherten Daten abgeleitet werden können

2. nur vernünftige Anwendungsdaten sollen gespeichert werden können

3. Anwendungsdaten sollen möglichst nicht redundant dargestellt werden

Grob: drei Entwurfsschritte

- konzeptuelle Modellierung der Datenbank

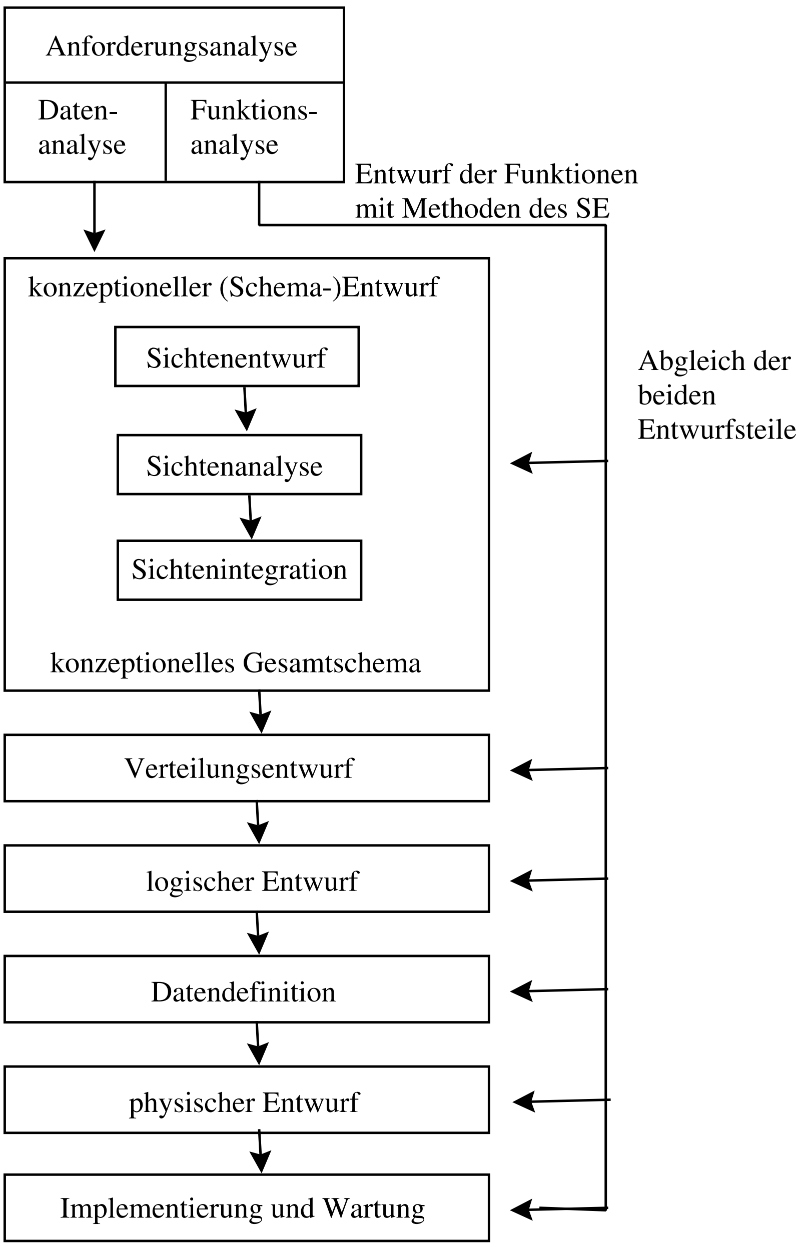
- Abbildung des konzeptuellen Modells auf kommerzielles DB-Modell

- Definition der DB in DDL

Genauer: Phasenmodell

Datenbankentwurf, Datendefinition

Phasenmodell



Datenbankentwurf, Datendefinition

Konzeptueller Entwurf, Sichtintegration

- erste 'formale' Beschreibung in semantischem Datenbankmodell (z.B. ER-Modell)
- Vorgehensweise:
  - Modellierung von Sichten in Fachabteilungen
  - Analyse der Sichten in Bezug auf
    - Namenskonflikte
    - Typ- und Wertebereichskonflikte
    - Bedingungskonflikte
  - Integration der Sichten in ein Gesamtschema
- Ergebnis:
  - Entity-Typen, Relationship-Typen
  - Attribute und Domänen
  - Integritätsbedingungen
  - Beschreibung des dynamischen Verhaltens

Datenbankentwurf, Datendefinition

Abbildung des ER-Modells auf andere DB-Modelle

- unterschiedliche Vorgehensweisen
  - manuelle Transformation der ER-Konzepte in Konzepte des konkreten DB-Modells nach Faustregeln
  - automatische Transformation der ER-Konzepte
- Ziele
  - Darstellung aller Informationen des ER-Diagramms im resultierenden DB-Schema
  - Erhaltung der Informationskapazität (im resultierenden DB-Schema genauso viele Instanzen darstellbar wie in ER-Diagramm - nicht mehr und nicht weniger!)

4.1 Abbildung ER-Modell → Relationenmodell

- Entity-Typ → Relationenschema mit allen Attributen des Entity-Typs, Schlüssel wird übernommen

- Relationship-Typ → Relationenschema mit allen Attributen des Relationship-Typs, zusätzlich alle Primärschlüssel der beteiligten Entity-Typen

  1:n : Primärschlüssel der n-Seite wird Schlüssel
  1:1 : beide Primärschlüssel werden je ein Schlüssel
  n:m : beide Primärschlüssel zusammen werden Schlüssel

- Is-a-Relationship → Relationenschema mit allen Attributen des spezielleren Entity-Typs, zusätzlich den Primärschlüssel des allgemeineren Entity-Typs

- bei *muß*-Beziehungen ([1,*])

  1:n : Entity-Relationenschema der n-Seite wird in Relationenschema der Relationship integriert
  1:1 : beide Entity-Relationenschemata verschmelzen

Abbildung ER-Modell → Relationenmodell

Beispiel: ER-Diagramm Warenhaus im Relationenmodell

Angestellter = { $\underline{\text{Name}}$, Gehalt}

Abteilung = { $\underline{\text{Abtlgsnr}}$, Abtlgsname}

Artikel = { $\underline{\text{Artnr}}$, Artname}

Lieferant = { $\underline{\text{LName}}$, LAdresse}

Auftrag = { $\underline{\text{Auftragsnr}}$, Datum}

Kunde = { $\underline{\text{KName}}$, KAdresse, Kontostand}

Chef = { $\underline{\text{Name}}$}

leitet = { $\underline{\text{Name}}$, $\underline{\text{Abtlgsnr}}$}

arbeitet = { $\underline{\text{Name}}$, Abtlgsnr}

führt = { $\underline{\text{Artnr}}$, Abtlgsnr}

liefert = { $\underline{\text{LName, Artnr}}$, Preis}

enthält = { $\underline{\text{Artnr, Auftragsnr}}$, Menge}

bestellt = { $\underline{\text{Auftragsnr}}$, KName}

Abbildung ER-Modell → Relationenmodell

Problemfälle ER-Modell → Relationenmodell

- gleichnamige, gleichbedeutende Relationships (leiht in Uni-Anwendung)
- Relationship eines Entity-Typs mit sich selbst (Voraus in Uni-Anwendung)
- mehrwertige Beziehungen (empfiehlt in Uni-Anwendung)

nicht abbildbar

- komplexe Attribute wie Stichwort, Autoren
- Generalisierung (Disjunktheitsbedingung)

Abbildung ER-Modell → Relationenmodell

Datendefinition im Relationenmodell:
mittels SQL-DDL (Teil von SQL; Normen: SQL-89, SQL-92, SQL3)

SQL als DDL:

- externe Ebene: `create view, drop view`
- konzeptuelle Ebene: `create table, alter table, drop table`
- interne Ebene: `create index, alter index, drop index`

SQL-89 mit IEF:
+ Definition von Schlüsseln
+ Definition von Fremdschlüsseln

SQL-92:
+ `default`-Klausel
+ `create domain`-Anweisung für benutzerdefinierte Wertebereiche
+ `check`-Klausel

Bsp.: `domain` Fachgebiet `char(20) default` 'Informatik'

```
create table Vorlesung
( Titel char(30) not null, primary key,
SWS integer check(SWS > 0);
Studiengang Fachgebiet )
```

Abbildung ER-Modell → Relationenmodell

Beispiel: Datendefinition Warenhaus mittels SQL-DDL von Oracle

```
create table artikel (
artnr number(4) constraint art_pk primary key,
artname varchar2(40) );


create table kunde (
kname varchar2(20) constraint kunde_pk primary key,
kadresse varchar2(30),
kontostand number(6,2) );

create table auftrag (
auftragnr number(6) constraint auf_pk primary key,
datum date );

create table arbeitet (
angname varchar2(40),
abtlgnr number(2),
constraint arbeitet_fk1 foreign key (angname) references ange-
stellter(angname),
constraint arbeitet_fk2 foreign key (abtlgnr)
references abteilung(abtlgnr) );

alter table chef
add constraint chef_primk primary key (angname);
```

4.2 Abbildung ER $\rightarrow$ Netzwerkmodell

- Entity-Typ $\rightarrow$ logical record type mit allen Attributen als fields

- Relationship-Typ $\rightarrow$ link (Set-Typ) im Fall von binärer 1:n Relationship ohne Attributen, andernfalls z.B. Ausweg über künstl. record type ('Kett-Record-Typ')

  1:n (binär, ohne Attribute) : link
  1:n mit Attributen : Attribute der Beziehung in Record-Typ der Member-Seite aufnehmen
  n:m : über Kett-Record-Typ (mit allen Attributen)
  1:1 : wie 1:n Relationship, Zusatzbedingung im Programm überwachen

- Is-a-Relationship $\rightarrow$ wie 1:n Relationship, zwei Zusatzbedingungen (1:1, gleiches Entity) im Programm überwachen

Abbildung ER → Netzwerkmodell

Beispiel: Warenhaus in Netzwerkmodell

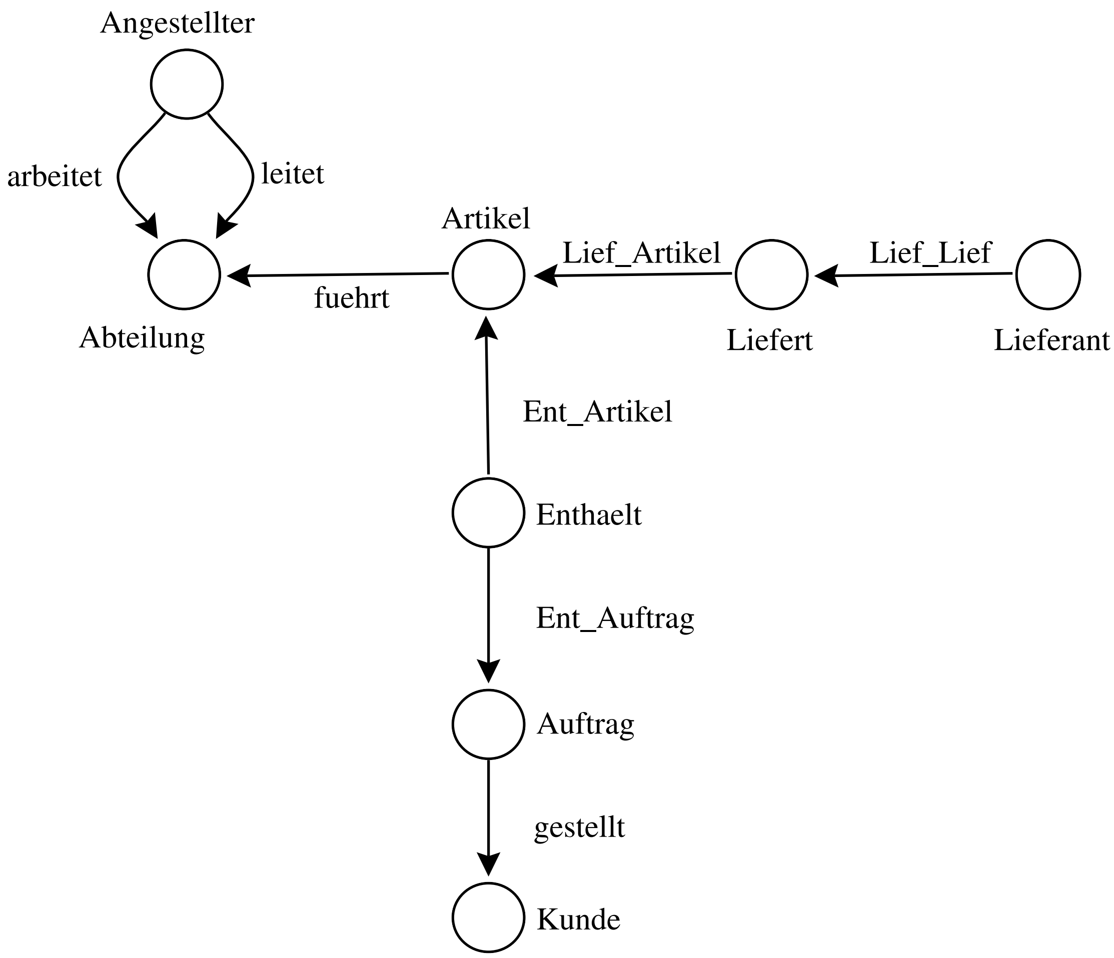


Abbildung ER → Netzwerkmodell

Datendefinition im Netzwerkmodell:
z.B. in DBTG DDL

Record-Typen allgemein:

```
RECORD r
   1 Feldname1 Typ1
   1 Feldname2 Typ2
   1 Feldname3
        2   Feldname31 Typ31
        2   Feldname32 Typ32
   1 Feldname4 Typ4
   KEY   Feldname_i
```

Set-Typen allgemein:

```
DBTG SET s
        OWNER IS o
        MEMBER IS m
```

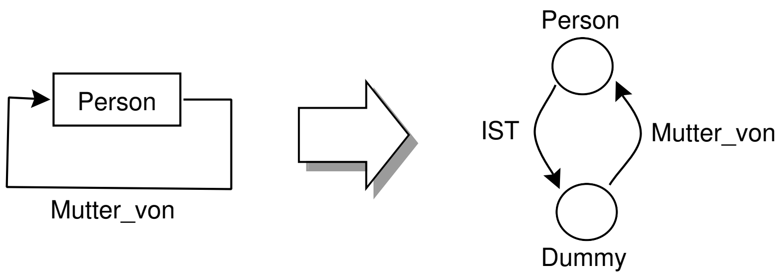
Abbildung ER → Netzwerkmodell

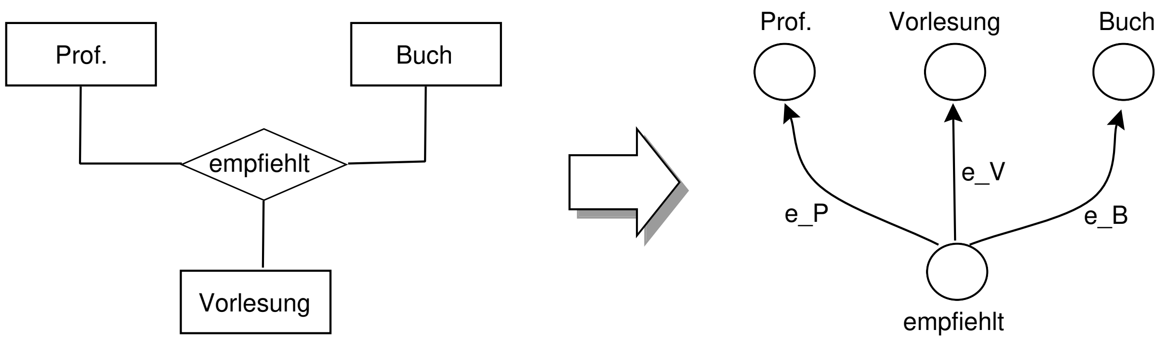
Datendefinition DBTG DDL Warenhaus:

```
RECORD Angestellter
   1 Name CHAR(40)
   1 Gehalt REAL
   KEY   Name

RECORD Abteilung
   1 Abtlgsnr INTEGER
   1 Abtlgsname CHAR(25)
   KEY    Abtlgsnr

...

RECORD Liefert
   1 Preis REAL

RECORD Enthaelt
   1 Menge INTEGER

DBTG SET arbeitet
        OWNER IS Abteilung
        MEMBER IS Angestellter

DBTG SET Leitet
        OWNER IS Abteilung
        MEMBER IS Angestellter

DBTG SET Lief_Artikel
        OWNER IS Artikel
        MEMBER IS Lieferant

DBTG SET Lief_Lief
        OWNER IS Lieferant
        MEMBER IS Liefert

...
```

Abbildung ER → Netzwerkmodell

Problemfälle ER → Netzwerkmodell:





4.3 Abbildung ER → Hierarchisches Modell

- da Hierarchisches Modell Einschränkung des Netzwerk-Modells: Hierarchisches Schema aus Netzwerk ableiten:
- aus Netzwerkschema Menge von Bäumen erstellen
- Merkregel: Links zeigen vom Nachfolger zum Vorgänger
- Ansatzpunkt: Ecken ohne auslaufende Kanten werden Wurzel von Bäumen
- Ecken mit mehr als einer auslaufenden Kante: Auftrennen mit virtual record types
- gerade n:m Beziehungen nachträglich durch wechselseitige virtual record types nachbessern

Beispiel Warenhaus: Wurzeln werdenAbteilung, Lieferant, Kunde
Hierarchisches Schema:

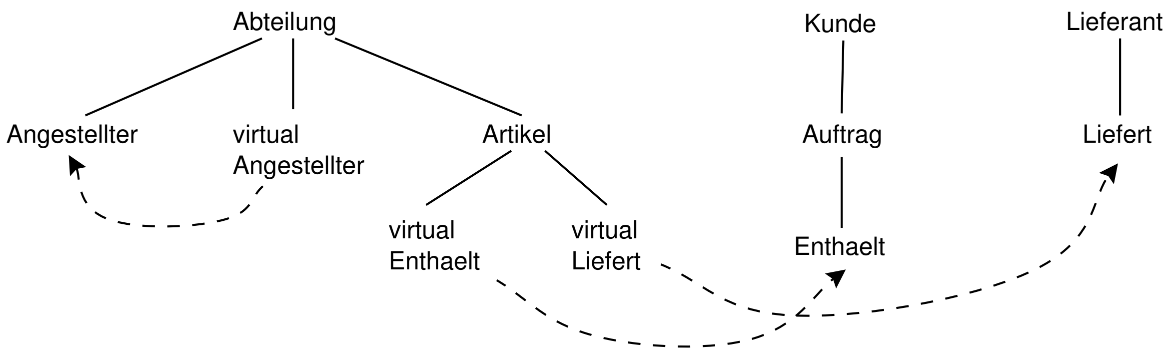


Abbildung ER → Hierarchisches Modell

Datendefinition im Hierarchischen Modell:
z.B. in IMS-DDL (hier nur angelehnt an IMS-DDL)

Baum mit Record-Typen allgemein:

```
TREE Baumname
RECORD r1 Liste_von_Record-Informationen
RECORD r2 Liste_von_Record-Informationen
...
```

wobei Record-Informationen sind

- Feldangabe (wie bei DBTG-DDL)
- Position im Baum:
  - ROOT
  - PARENT = Name_des_Vorfahren
- virtuelle Felder mit
  VIRTUAL Record-Typ IN Name_des_Baumes

Abbildung ER → Hierarchisches Modell

Datendefinition IMS-DDL Warenhaus:

```
TREE Abteilungsbaum
        RECORD Abteilung ROOT
                1 Abtlgsnr INTEGER
                1 Abtlgsname CHAR(25)
        RECORD Angestellter PARENT=Abteilung
                1  Name CHAR(40)
                1 Gehalt REAL
        RECORD Chef PARENT=Abteilung
                VIRTUAL Angestellter IN Abteilungsbaum
        RECORD Artikel PARENT = Abteilung
                ...
        ...
        RECORD Virt_Enthaelt PARENT=Artikel
                VIRTUAL Enthaelt IN Kundenbaum
        RECORD Virt_Lief PARENT=Artikel
                VIRTUAL Liefert IN Lieferantenbaum
...
```