Hierarchisches Datenmodell / IMS

Inhalt:

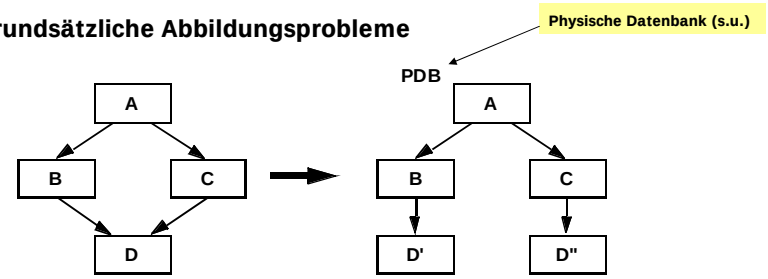
Überblick
*Schemaarchitektur von **IMS***
Aufbau physischer und logischer Datenbanken
Die Anfragesprache DL/I
Realisierung von (n:m)-Abbildungen

Übersicht (1)

- **Ursprünglicher Ansatz (bereits ~ 1968)**
  - kein theoretisches Datenmodell
  - pragmatische Lösung praktischer Datenhaltungsprobleme
  - gemeinsame Entwicklung durch Hersteller (IBM) und Anwender
- **Datenstrukturen**
  - hierarchisch organisierter Record-Typ
    - zur Abbildung von hierarchisch abhängigen Entity-Mengen
    - eine physische DB pro Record-Typ
  - mehrere Datenbanken pro Anwendung
  - Informationsdarstellung nicht nur durch Werte
    - implizite Darstellung hierarchischer Beziehungen
    - Reihenfolge
- **Operatoren**
  - satzorientierte Operationen
  - Navigation längs hierarchischer Pfade
  - keine Operationen über Datenbanken (hierarchische Sätze) hinweg

Übersicht (2)

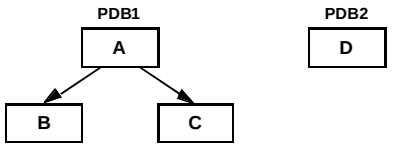
- **Grundsätzliche Abbildungsprobleme**

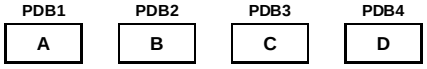


  - Redundanz bei n:m
  - Einweg-Beziehung: von oben nach unten! (unsymmetrisches DM)
  - Verlust an Information: unten → oben?
  - Einstieg: immer die Wurzel
  - Implizite Darstellung der 1:n-Beziehung (kein Set-Name)

Set: CODASYL-Konstrukt, explizit

Übersicht (3)

- **Andere** (vgl. Folie 3) **Abbildungsmöglichkeiten**
  - 

  - oder 

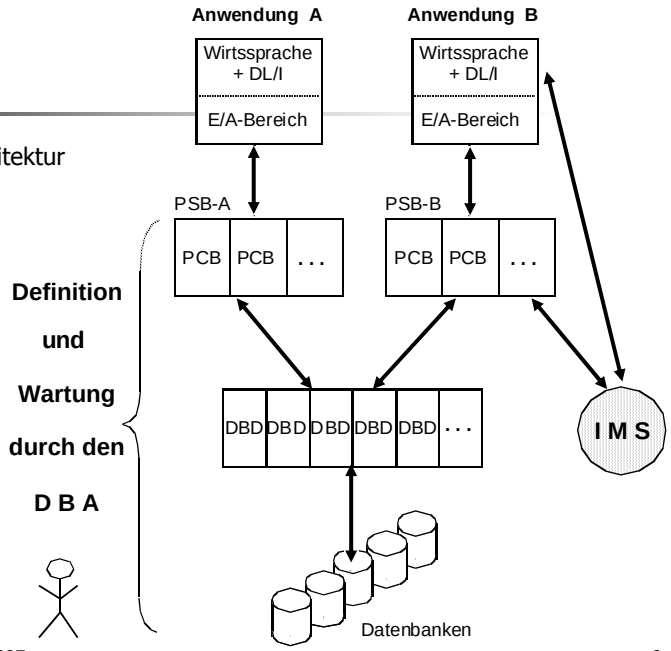
Degeneration zum Datei-Konzept
  - etc.
- keine DB-übergreifenden Operationen
- Verknüpfung durch Benutzer!

Hierarchiemodell am Beispiel IMS

- **I M S**: Information Management System
- Datenorganisation → mehrere physische DBs (PDB)
  - eine PDB: geordnete Menge von hierarchischen PDBR (Physical DB Record)
  - Sammlung von PDBs: Konzeptuelles Datenmodell
- konzeptuelles/internes Schema
  - PDB wird beschrieben durch DBD (Data Base Description)
- externes Schema (logische Sicht)
  - Sammlung von logischen DBs (LDB)
  - LDB: Untermenge einer oder mehrerer PDBs
  - LDB wird beschrieben durch PCB (Program Communication Block)
- Sicht eines Programms
  - Zugriff auf mehrere LDBs
  - Menge aller PCBs wird zusammengefasst im PSB (Program Specification Block)
- DB-Programmier-Schnittstelle (CALL-Schnittstelle): PSB + DL/I (DML)

IMS

- Schemaarchitektur



Datenmodell

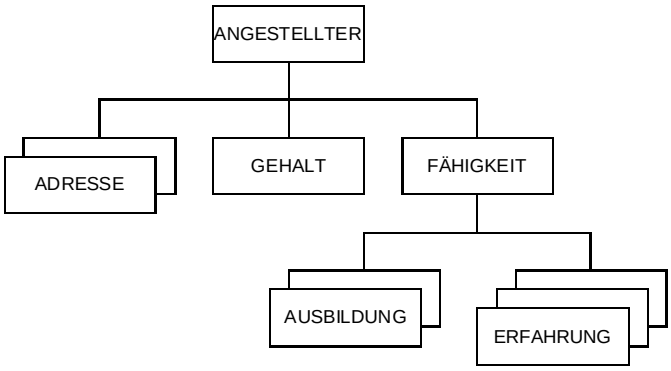
- Administrator-Sicht: DB(-Gruppe), INDEX-DB, PDB, LDB, log. Beziehungen, RECORD, SEGMENT, FIELD
- Aufbau eines PDB
  - geordnete Menge von PDBRs eines Typs
  - PDBR besteht aus hierarchisch angeordneten Segmentausprägungen verschiedener Typen
  - Segment setzt sich zusammen aus Menge von Feldausprägungen
- Terminologie
  - ROOT-Segment
  - PARENT-Segment
  - CHILD-Segment
  - TWIN-Segment (alle Ausprägungen eines Segmenttyps, die zu einem PARENT gehören)

Regeln für die Hierarchie-Bildung

1. Eine Datenbank besteht aus **1** bis **N** Datenbanksätzen.
2. Ein Datenbanksatz besteht aus **1** bis **K** Segmenten.
3. Die maximale Anzahl an Segment-Typen beträgt **255**.
4. Maximal sind **15** Segment-Ebenen zulässig.
5. Es gibt nur **1 Root-Segment** pro Datenbanksatz, andere Segmente können **0** bis **M** mal pro **Parent** vorkommen.
6. Jedes Segment hat nur ein Schlüsselfeld.
7. Hierarchische Integrität: Definiertheit des Vaters.

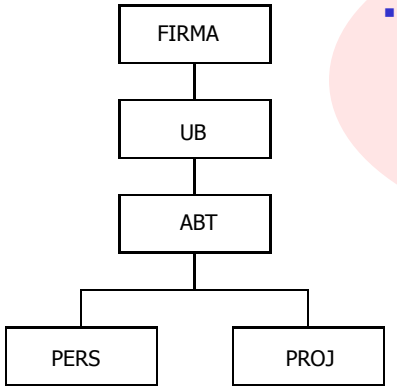
PDBR-Beispiel ...

- ... mit 6 Segment-Typen und 3 Segment-Ebenen



PDBR-Beispiel

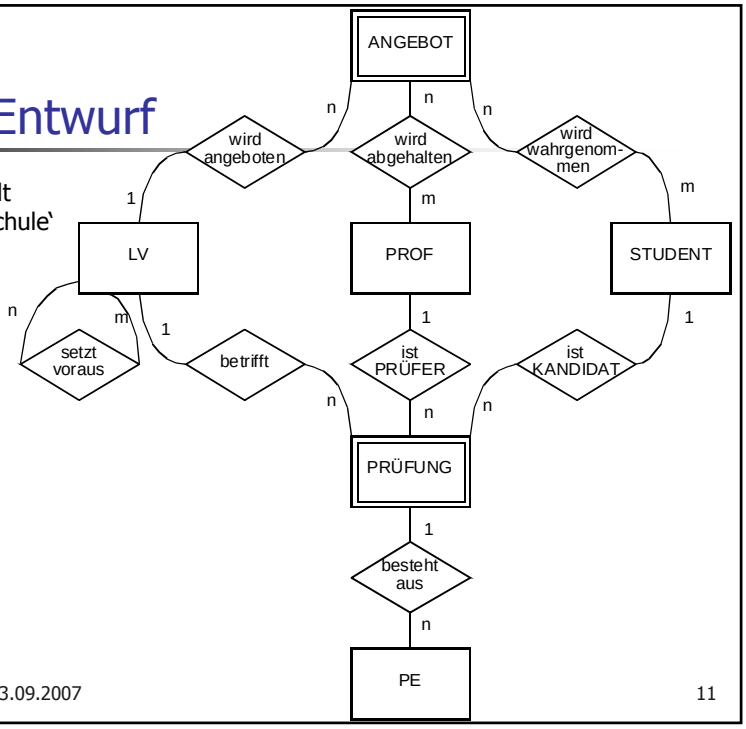
- Echte Hierarchie



- Relationendarstellung
  - FIRMA (FI-NR, ORT, . . .)
  - UB (UB-NR, FI-NR, NAME, UMSATZ, . . .)
  - ABT (ANR, UB-NR, . . .)
  - PERS (PNR, ANR, . . .)
  - PROJ (PRONR, ANR, . . .)

DB-Entwurf

- Miniwelt ‚Hochschule'

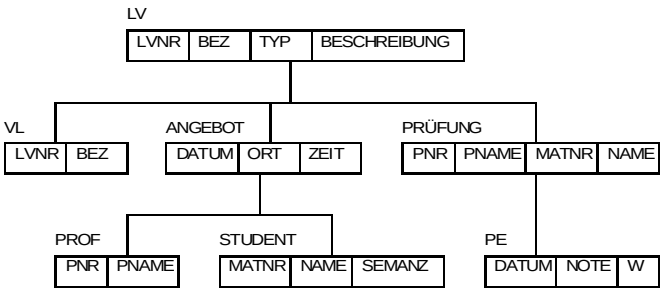


DB-Entwurf (Forts.)

- Erinnerung: Relationendarstellung
  - LV (LVNR, . . .)
  - PE (LVNR, PNR, MATNR, DATUM, . . .)
  - VL (LVNR, VLVNR, . . .)
  - ANGEBOT (LVNR, DATUM, . . .)
  - PRÜF (LVNR, PNR, MATNR, . . .)
  - REL1 (LVNR, DATUM, PNR, . . .)
  - REL2 (LVNR, DATUM, MATNR, . . .)
  - PROF (PNR, . . .)
  - STUDENT (MATNR, . . .)

DB-Entwurf (Forts.)

- Hierarchische Datenbank ‚LV-DB'



DB-Entwurf (Forts.)

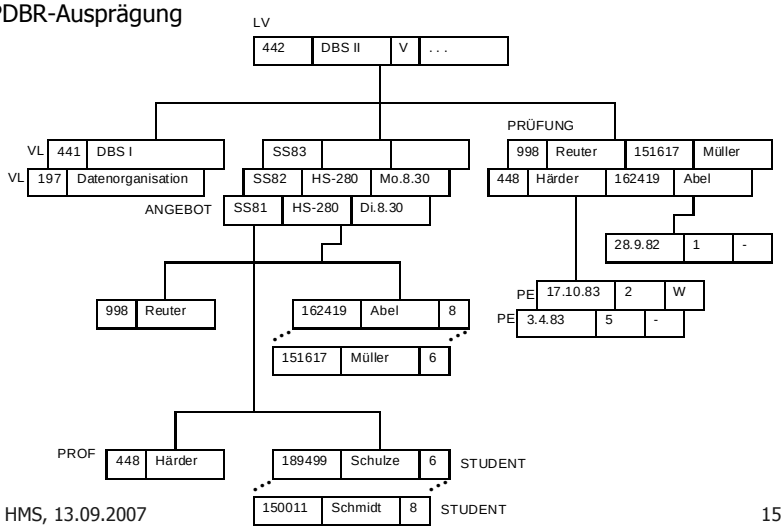
- DBD der physischen Datenbank (PDB):

```
1   DBD    NAME = LV-DB
2   SEGM   NAME = LV, BYTES = 200
3   FIELD  NAME = (LVNR, SEQ), BYTES = 3, START = 1
4   FIELD  NAME = BEZ, BYTES = 40, START = 4
5   FIELD  NAME = TYP, BYTES =   1, START = 44
6   FIELD  NAME = BESCHREIBUNG, BYTES = 156, START = 45
7   SEGM   NAME = VL, PARENT = LV, BYTES = 43
8   FIELD  NAME = (LVNR, SEQ), BYTES = 3, START = 1
9   FIELD  NAME = BEZ, BYTES = 40, START = 4
10  SEGM   NAME = ANGEBOT, PARENT = LV, BYTES = 16
11  FIELD  NAME = (DATUM, SEQ, M), BYTES = 4, START = 1
12  FIELD  NAME = ORT, BYTES = 6, START = 5
13  FIELD  NAME = ZEIT, BYTES = 6, START = 11
14  SEGM   NAME = PROF, PARENT = ANGEBOT, BYTES = . .
…
```

- Beschreibung von oben nach unten, von links nach rechts

DB-Entwurf (Forts.)

- PDBR-Ausprägung

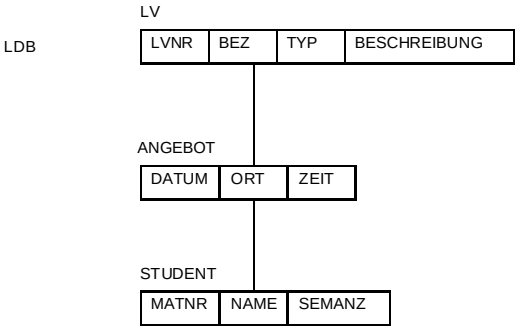


Hierarchische Reihenfolge

- Die Reihenfolge der Segmenttypen und der Segmentausprägungen sind durch das logische Datenmodell definiert.
  - Reihenfolge der SEGM-Anweisungen definiert die hierarchische Struktur.
  - Reihenfolge der Segment-Ausprägungen definiert durch SEQ-Option der ersten FIELD-Anweisung.
  - Die Reihenfolge aller Segmentausprägungen eines hierarchischen Satzes ist damit eindeutig definiert.
- Typ-Code des Segmentes (im Beispiel oben)
  - **1** LV, **2** VL, **3** ANGEBOT, **4** PROF, **5** STUDENT, **6** PRÜFUNG, **7** PE
- Hierarchischer Reihenfolge-Schlüssel
  - Wert des SEQ-Feldes einer Segmentausprägung
  - Typ-Code des Segmentes als Präfix
  - hierarchischer Reihenfolge-Schlüssel des Vaters als Präfix
- Beispiel für STUDENT ‚Schmidt': **1 4 4 2 3 S S 8 1 5 1 5 0 0 1 1**

Logische DB auf einer physischen DB

- **Subschema**

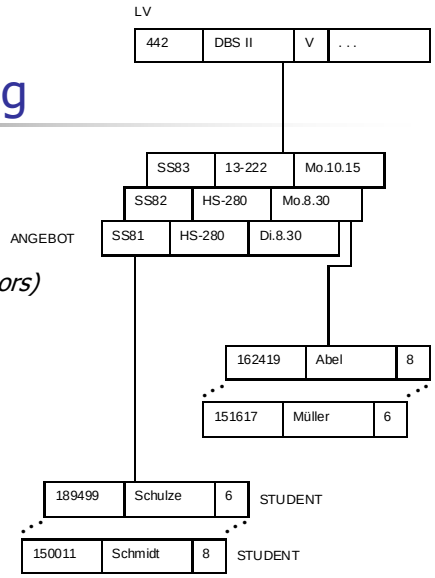


Logische DB auf einer physischen DB

- Möglicher logischer Datenbankrecord-Typ (LDBR):
  - Kriterium für die Definition: jede Subhierarchie mit der Wurzel als Ausgangspunkt
  - Hinzufügen von neuen Segment-Typen im PDBR nur, wenn keine existierende Vater-Kind-Beziehung gestört wird.
- PCB (Program Communication Block) für die LDB
  -
```
1 PCB     TYPE = DB, DBDNAME = LV-DB, KEYLEN = 13
2 SENSEG  NAME = LV, PROCOPT = G
3 SENSEG  NAME = ANGEBOT, PARENT = LV, PROCOPT = G
4 SENSEG  NAME = STUDENT, PARENT = ANGEBOT, PROCOPT = G
```
- PCB hat eine Key Feedback Area

LDBR-Ausprägung

- Key Feedback Area enthält jeweils den voll konkatenierten Schlüssel des zuletzt angeforderten Segmentes *(Funktion eines Cursors)*
  - Beispiel: **442 SS91 150011**

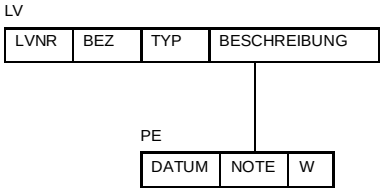


Zugriffskontrolle

- Im LDBR können nur ‚sensitive' Segmente gesehen werden.
  - Isolationsfunktion des Subschemas
  - wertunabhängige Zugriffskontrolle durch PROCOPTS
    - G = GET
      I = INSERT
      R = REPLACE
      D = DELETE
      L = LOAD
      . . .
- Problem für die Zugriffskontrolle:
  - Prinzipiell müssen immer alle Segmente in einem hierarchischen Pfad sensitiv sein.
  - Behelfslösung: PROCOPT = K ("key sensitivity")

Zugriffskontrolle (Forts.)

- Beispielanwendung: Statistik-Sicht
  - 1 PCB TYPE=DB, DBNAME=LV-DB, KEYLEN=15
    2 SENSEG NAME=LV, PROCOPT=G
    3 SENSEG NAME=PRÜFUNG, PROCOPT=K
    4 SENSEG NAME=PE, PROCPT=G



  - ACHTUNG: Es wird immer der voll konkatenierte Schlüssel geliefert !

Kommunikationsbereich

- Kommunikationsbereich von DBS und AP für eine LDB
- Beispiel in PL/I

```
DLITPLI:    PROCEDURE(LVPCB)OPTIONS(MAIN);
            ...
            DCL     1       LVPCB,
                    2       DBDNAME CHAR(8),
                    3       SEGLEVEL        CHAR(2),
                    4       STATUS          CHAR(2),
                    5       PROCOPT CHAR(4),
                    6       RESERVED        FIXED BINARY(31),
                    7       SEGNAME CHAR(8),
                    8       KEYFBLEN        FIXED BINARY(31),
                    9       #SENSEGS        FIXED BINARY(31),
                    10      #KEYFBAREA      CHAR(13);
```

Kommunikationsbereich (Forts.)

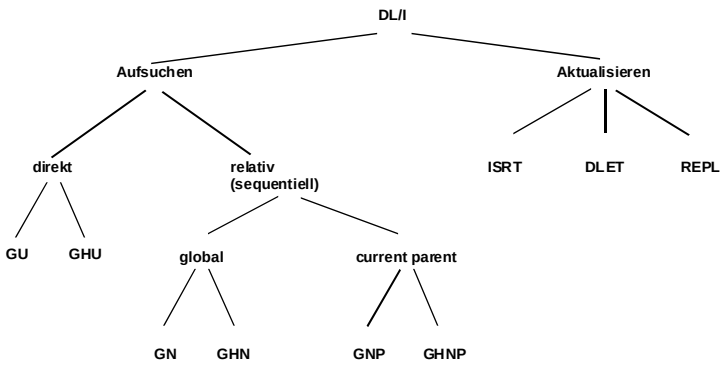
- Nach Ausführung des Retrieval-Statements
  - GU LV (BEZ = DBSII)
    ANGEBOT (DATUM = SS81 OR DATUM = SS89)
    STUDENT (SEMANZ = 8)

auf der PDB LV enthält LVPCB:

LV-DB	03	03	G	.....	Student	13	3	442SS81150011

Operationen auf hierarchischen Sätzen

- IMS Data Sublanguage



Operationen (Forts.)

- Wie wird in einer hierarchischen DB lokalisiert?
  - SSA : Segment Search Argument spezifiziert hierarchischen Pfad
    - ***Op*** *S1 [SSA]*
      S2 [SSA]
      ...
      Sn [SSA]
    - mit SSA: *(Feldname Θ Wert) [{AND/OR} (Feldname Θ Wert) . . .]*
- Welches Segment wird aufgesucht?
- Operationen "von außen" verlangen SSAs für den gesamten hierarchischen Pfad → GU + ISRT
- relative Operationen können SSAs besitzen, müssen aber nicht (hierarchischer Pfad kann irgendwo beginnen) → GN + GNP
- DLET + REPL haben keine SSAs

Aufsuchoperationen

1. Direktes Aufsuchen
   - Finde das erste Angebot einer LV in HS-280.
     - `GU  LV`
       `ANGEBOT (ORT = HS-280)`
   - GU ist eigentlich GET FIRST
2. Relatives Aufsuchen mit SSA
   a. Finde alle Studenten in LV-DB vom gefundenen LV-Angebot an
     - `GU  LV`
       `ANGEBOT (ORT = HS-280)`
       `STUDENT`
       `OUTPUT`
       `X:  GN  STUDENT`
       `OUTPUT`
       `GOTO X`

Aufsuchoperationen (Forts.)

2. Relatives Aufsuchen mit SSA (Forts.)
   b. wie a), aber nur Studenten des 7. Semesters

```
GU    LV
      ANGEBOT (ORT = HS-280)
      STUDENT (SEMANZ = 7)
      ...
X:    GN      STUDENT (SEMANZ = 7)
              ...
      GOTO X
```

   c. Suche beginnt am Anfang der DB

```
GU    LV
      ...
X:    GN      STUDENT (SEMANZ = 7)
              ...
      GOTO X
```


Aufsuchoperationen (Forts.)

3. Pfadsuche
   a. Alle Studenten, die im 7. Semester DBS I in HS-280 gehört haben

```
GU    LV       (BEZ = DBS I)
      ANGEBOT  (ORT = HS-280)
               STUDENT  (SEMANZ = 7)
               ...
X:    GN       LV       (BEZ = DBS I)
               ANGEBOT  (ORT = HS-280)
               STUDENT  (SEMANZ = 7)
               ...
      GOTO X
```

   b. Lesen aller Segmente

```
GU    LV
      ...
X:    GN
      ...
      GOTO  X
```

Aufsuchoperationen (Forts.)

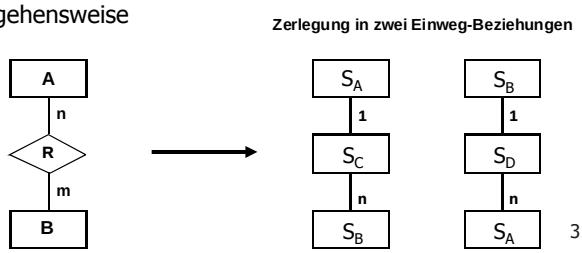
4. Einschränkung der Suche durch GNP
   a.
```
GU      LV      (BEZ = DBS II)
        ANGEBOT (DATUM = SS81)
        ...
X:      GNP     STUDENT
                ...
        GOTO X
```


Änderungsoperationen

- Einfügen
  - Speichere ein neues Angebot von DBSII:
    - Aufbau des Angebot-Segmentes im Arbeitsbereich
```
ISRT    LV      (LVNR=130)
        ANGEBOT
```
- Ändern
  - Im Angebots-Segment für WS90 soll der Ort geändert werden:
```
GHU     LV      (LVNR=130)
        ANGEBOT (DATUM = WS90)
```
    Durchführung der Änderung im Arbeitsbereich
```
REPL
```
- Löschen
  - Das Angebots-Segment für WS89 soll gelöscht werden:
```
GHU     LV      (LVNR=130)
        ANGEBOT (DATUM=WS89)
DLET
```

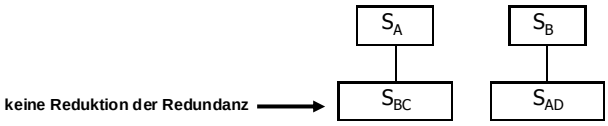
Darstellung von (n:m)-Beziehungen

- bisher: eine LDB über eine PDB
- Erweiterung
  - eine LDB über mehrere PDPs
  - Verwendung bei Abbildung von binären (n:m)-Beziehungen im hierarchischen Modell
- Eigenschaften
  - Reduzierung von Redundanz
  - Benutzung logischer Beziehungen (logical relationships)
  - Abbildungstechnik nicht verallgemeinerungsfähig
- grundsätzliche Vorgehensweise



Darstellung von (n:m)-Beziehungen

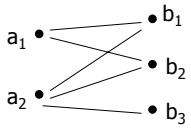
- Zwischensegment jeweils mit <u>einem</u> Blattsegment verknüpft



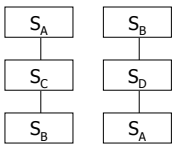
Darstellung von (n:m)-Beziehungen

- (n:m)-Abbildung durch hierarchische Segmente

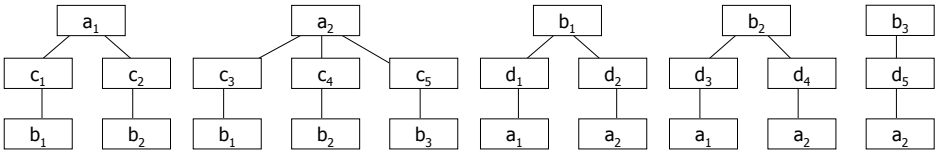
Strukturelle Beziehung:



Hierarchische Schemas:

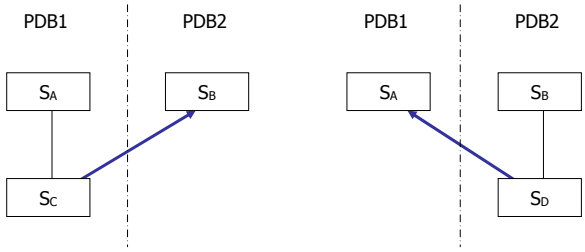


Ausprägungen:



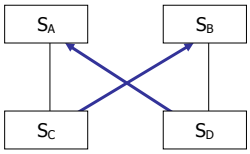
Nutzung von zwei Einweg-Beziehungen

- Einführung eines symbolischen Pointer-Typs (durch konkatenierten Schlüssel)
- Darstellung von logischen Beziehungen über PDB-Grenzen hinweg
- hierarchische Pfade



Physical Pairing

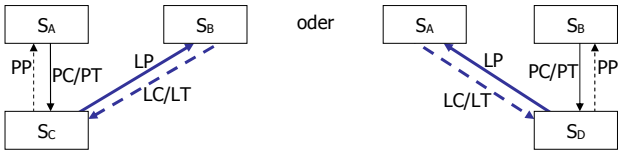
- Überlagerung der Einweg-Beziehungen: *physical pairing*



- SC und SD heißen "paired"
- "paired"-Segmente speichern konkatenierten Schlüssel ihres jeweiligen Logical Parent und jeweils einen identischen Rest

Virtual Pairing

- Redundanzverminderung durch Einsparung eines Zwischensegmentes

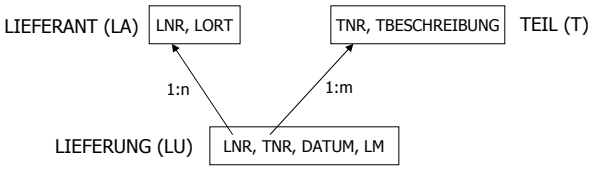


- Verknüpfungsarten
  - PP: Physical Parent
  - LP: Logical Parent

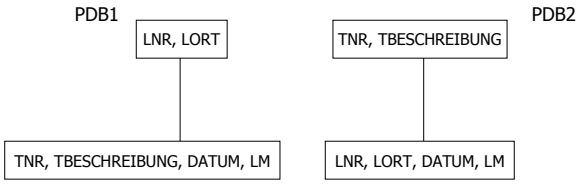
  } jeweils ein Zeiger
- n-Child-Pointer pro Parent schwierig zu implementieren, deshalb Implementierung durch
  - PC/PT: Physical Child / Physical Twin
  - LC/LT: Logical Child / Logical Twin

Beispiel

a. Relationales Modell

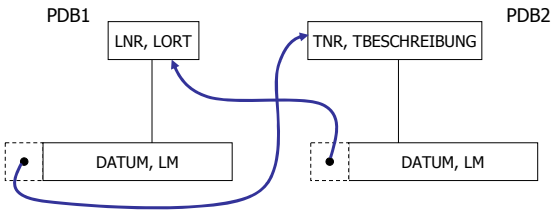


b. Realisierung durch volle Redundanz



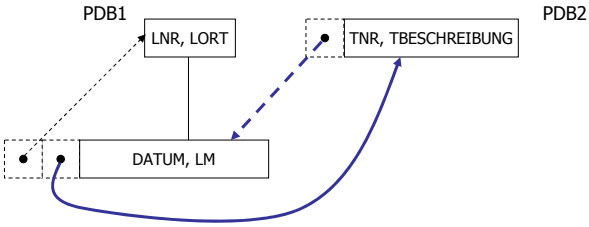
Beispiel

c. Physical Pairing



- zweiseitige hierarchische Sicht (2 LDB)

d. Virtual Pairing



Ausprägungen

a. Relationales Modell

LA

(LNR,	LORT)
K51,	K'lautern
K55,	Frankfurt
K56,	Darmstadt

L

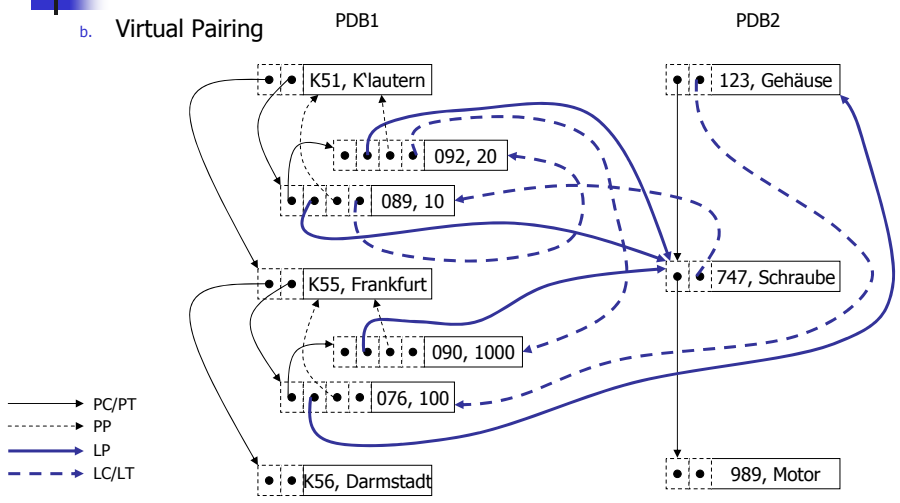
(TNR,	TBESCHREIBUNG)
123,	Gehäuse
747,	Schraube
989,	Motor

LU

(LNR,	TNR,	DATUM,	LM)
K51,	747,	089,	10
K51,	747,	092,	20
K55,	123,	076,	100
K55,	747,	090,	1000

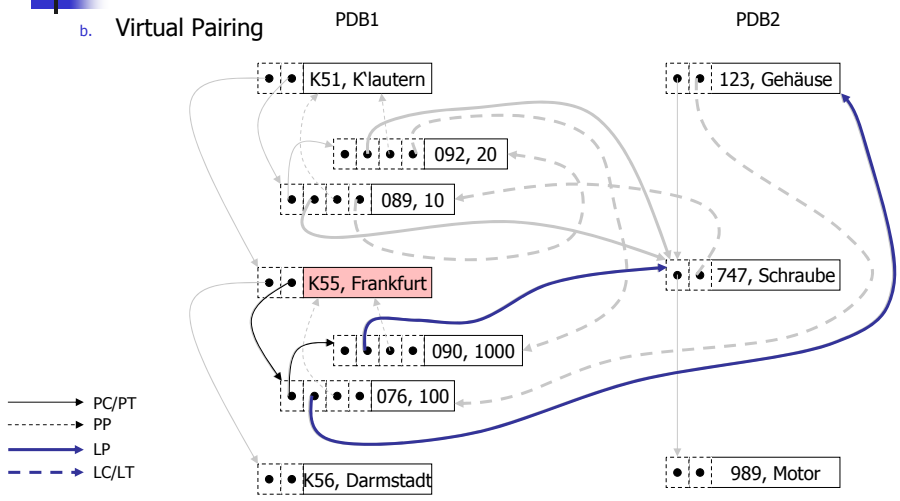
Ausprägungen

b. Virtual Pairing



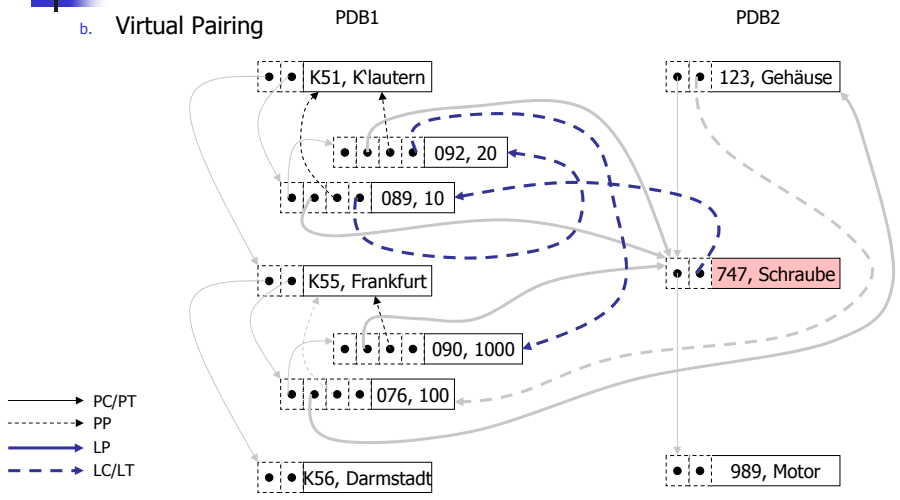
Ausprägungen

b. Virtual Pairing



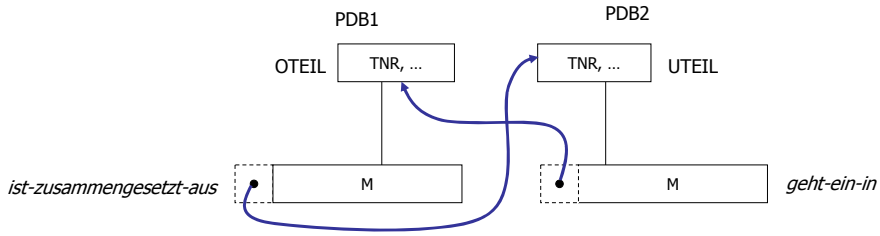
Ausprägungen

b. Virtual Pairing

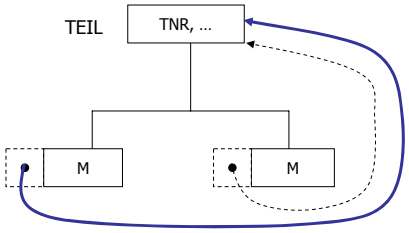


Stückliste

a. Physical Pairing (2 PDB, 2 LDB)



b. Physical Pairing (1 PDB, 2 LDB)



Stückliste

c. Virtual Pairing (1 PDB, 2 LDB)