Datenbanken

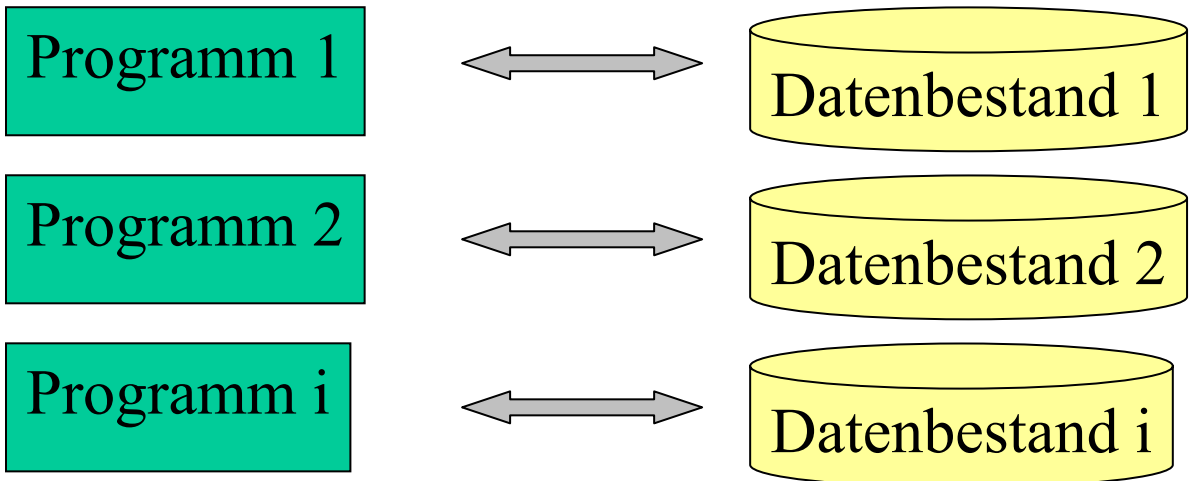
Das Ziel

- Klare Verwendung von Begriffen
- Verständnis für die Problemstellungen rund um Datenbanken
- Funktionsweise eines Datenbanksystems
- Theorie relationaler Datenbanken
- SQL

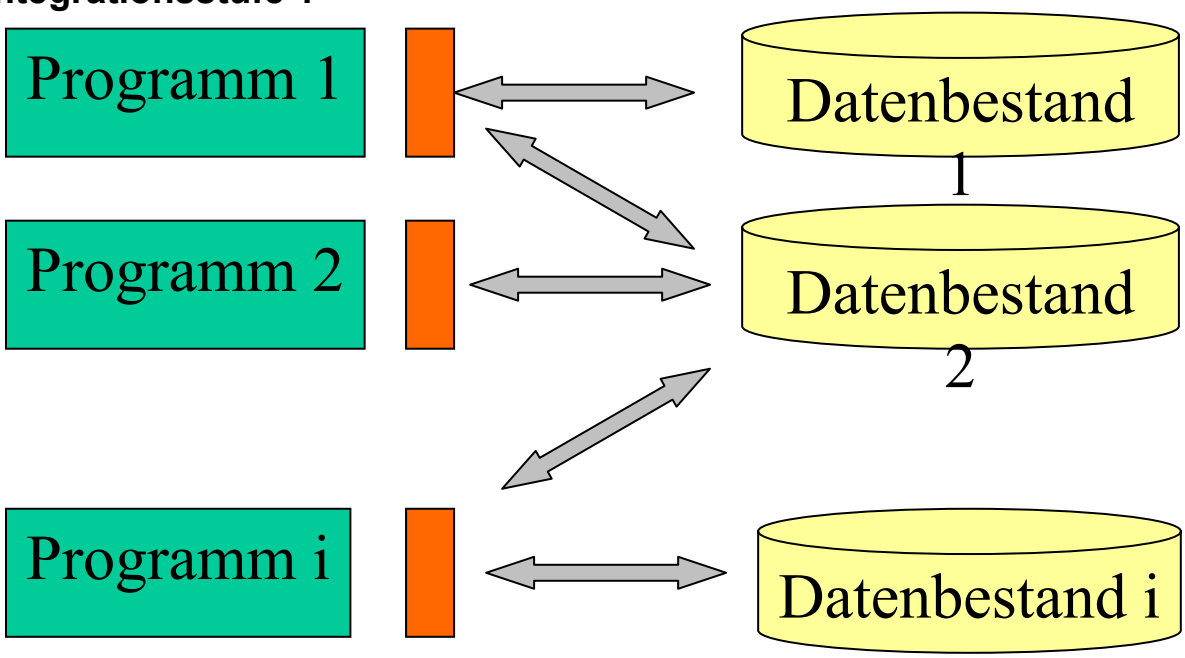
1. Teil

- Zugriffspfadstrukturen
- Zusammenhang zwischen Integrationsstufen und Entwicklungsmethodologie
- Definitionen
- Struktur und Funktionsweise von DBBS

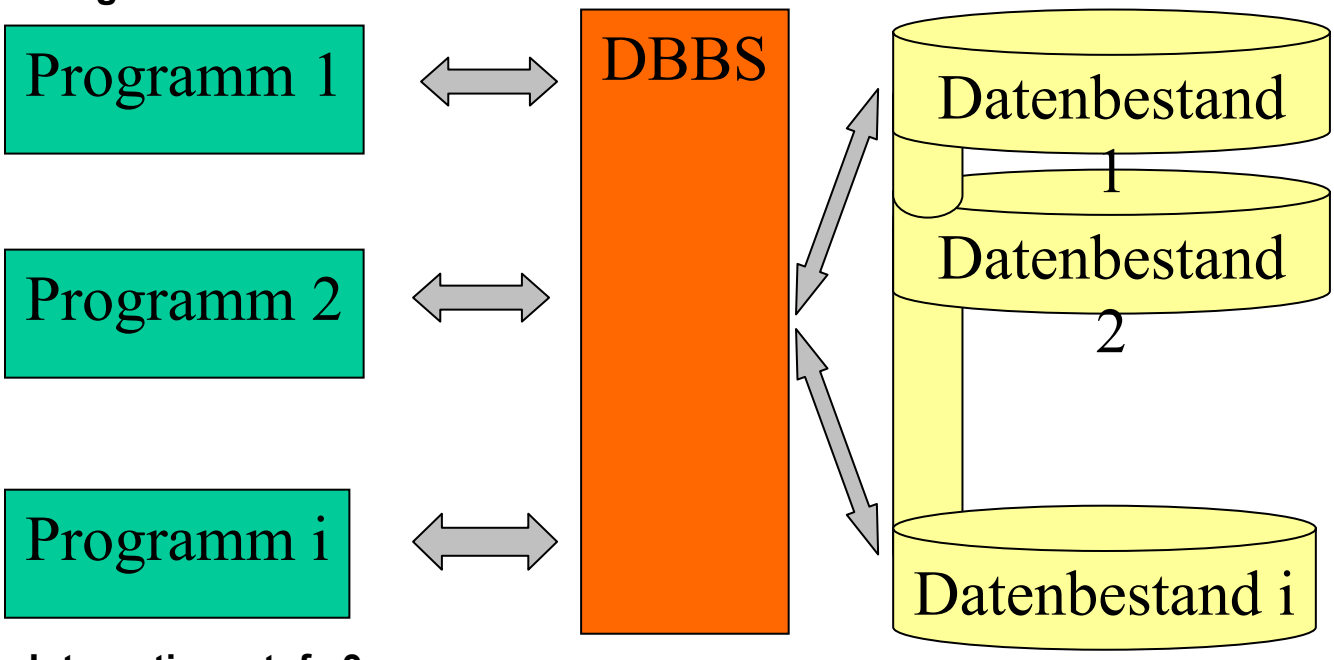
Integrationsstufe 0



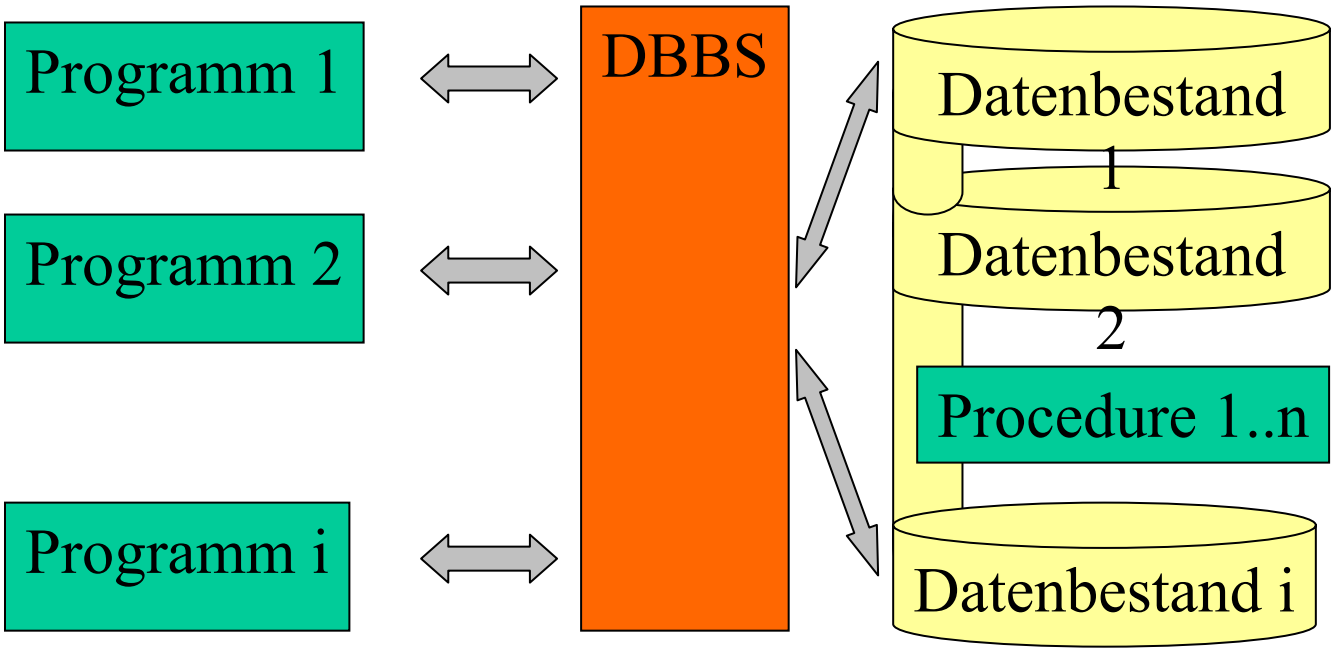
Integrationsstufe 1



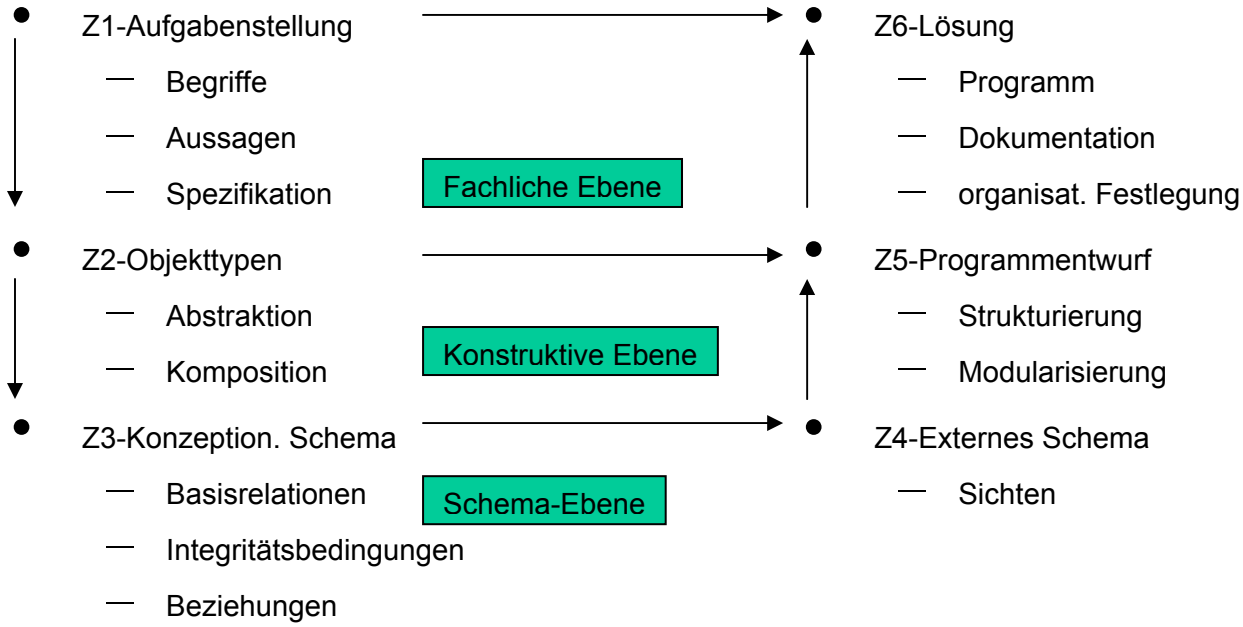
Integrationsstufe 2



Integrationsstufe 3



Entwicklungsmethodologie



Zusammenhang zwischen Integrationsstufen und Entwicklungsmethodologie

- Integrationsstufe 0
  - –Z1, Z6
  - –Katalogauswahl

- Integrationsstufe 1
  - –Z1, Z2, Z5, Z6
  - –konventionelle Entwicklung

- Integrationsstufe 2
  - –Z1, Z2, Z3, Z4, Z5, Z6
  - –Entwicklung mit DBS

Ziele der Datenbanktechnologie

- Redundanzfreiheit
- Konsistenz
- Mehrfachnutzung von Programmen und Daten
- Datensicherheit
- Datenunabhängigkeit

Definitionen

- Ein Datenbestand ist *redundant*, wenn ein Datum mehrfach gespeichert ist.
- Ein Datenbestand heißt *konsistent*, wenn es keine logischen Widersprüche zwischen den Daten gibt.
- Physische Datenunabhängigkeit liegt vor, wenn ein Programm invariant gegen Änderungen der physischen Speicherstruktur ist.

Datenbank

•Eine *Datenbank* ist eine Menge von Daten, die einen Ausschnitt der realen Welt unter Nutzung eines Datenmodells beschreibt.

DBBS

•Ein Softwaresystem, mit dessen Hilfe Daten einer Datenbank verwaltet, manipuliert und recherchiert werden können heißt *Datenbankbetriebssystem* (Database Management System).

Datenbanksystem

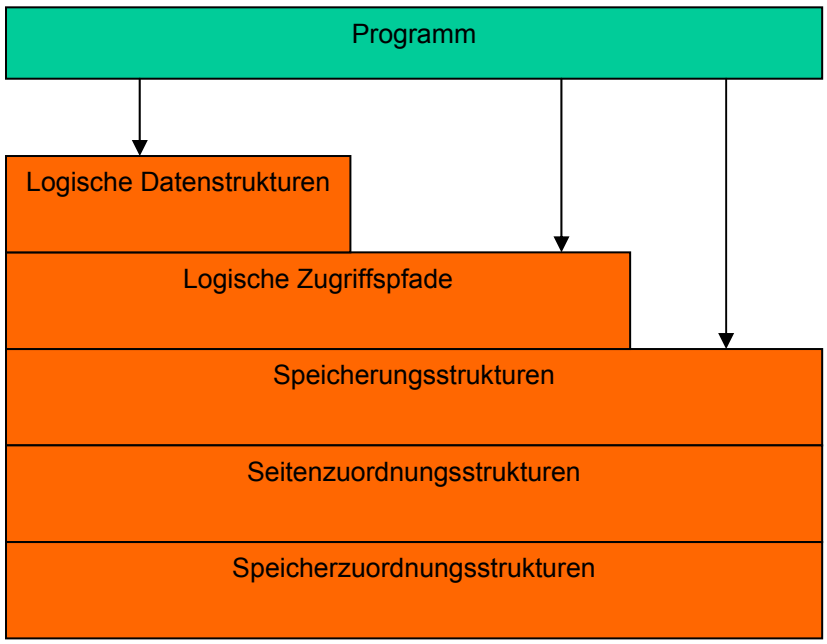
•Datenbank, Datenbankbetriebssystem und Anwendungsprogramme bilden gemeinsam ein *Datenbanksystem*.

•DBS = DB + DBBS + AP

Aufgaben eines DBBS

•Datendefinition
–Strukturen, Zugriffspfade, Sichten, Integritätsbedingungen
•Datenmanipulation
–Einfügen, Löschen, Ändern, Strukturmanipulation
•Datenverwaltung
–Wiederauffindbarkeit, Reorganisation, Datenaustausch
•Datensicherheit und -schutz
–Integrität, Zugriffsschutz, Recovery
•Synchronisation der Benutzeraufträge

5-Schichtenmodell



Logische Datenstrukturen

- Schnittstelle
  - mengenorientiert
  - Relationen, Sichten, Tupel
  - Sprachen SQL, QBE
- Abbildungshierarchie
  - Übersetzung
  - Zugriffspfadoptimierung
- Datensicherungshierarchie
  - Zugriffskontrolle
  - Integritätskontrolle

Logische Zugriffspfade

- Schnittstelle
  - satzorientiert
  - externe Sätze
  - Index- und Set-Strukturen
  - FIND NEXT satz, STORE satz
- Abbildungshierarchie
  - Data Dictionary
  - Currency-Konzept
  - Sortierkomponente
- Datensicherungshierarchie
  - Transaktionsverwaltung

Speicherungsstrukturen

- Schnittstelle
  - interne Satzschnittstelle
  - interne Sätze
  - B*-Bäume
  - Hash-Strukturen
  - Adreßketten
  - STORE satz
  - Füge Eintrag in B*-Baum ein
- Abbildungshierarchie
  - Record Manager
  - Zugriffspfadverwaltung
- Datensicherungshierarchie
  - Sperrverwaltung
  - Log- / Recoverykomponente

Seitenzuordnungsstrukturen

- Schnittstelle
  - Systempufferschnittstelle
  - Segmente, Seiten
  - Bereitstellen Seite j
  - Freigeben Seite j

•Abbildungshierarchie
–Systempufferverwaltung
–Einbringestrategie

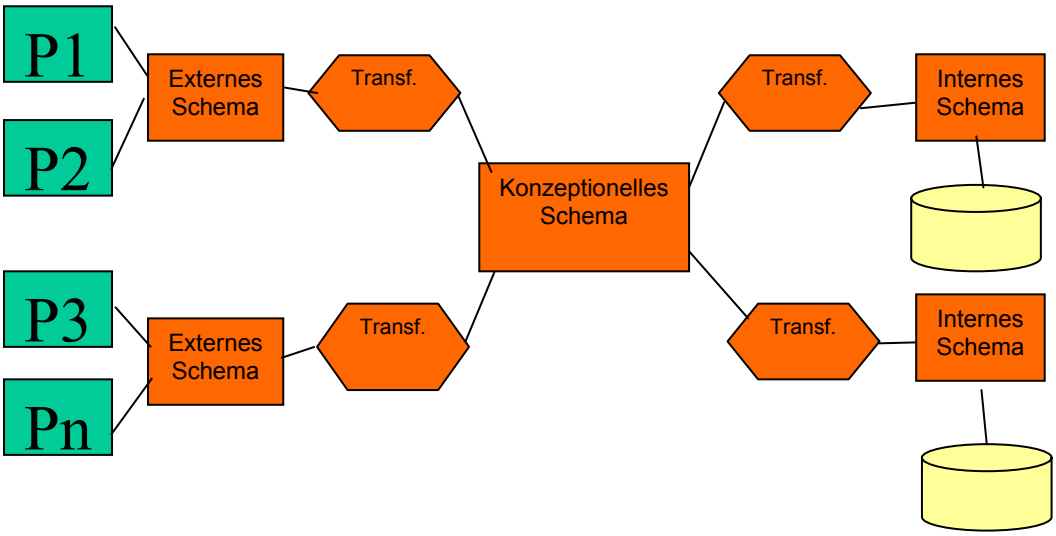
Speicherzuordnungsstrukturen

•Schnittstelle
–Dateischnittstelle
–Dateien, Blöcke
–Lies Block k, Schreibe Block k
•Abbildungshierarchie
–Externspeicherverwaltung

Programmierschnittstellen

•mengenorientierte DB-Schnittstelle
•satzorientierte (navigierende) DB-Schnittstelle
•interne Satzschnittstelle

ANSI/SPARC



Externes Schema

- Individuelle Subschemata (Sichten) der Benutzer
- Datenschutz (Daten verbergen)
- Einschränkung der Datenvielfalt

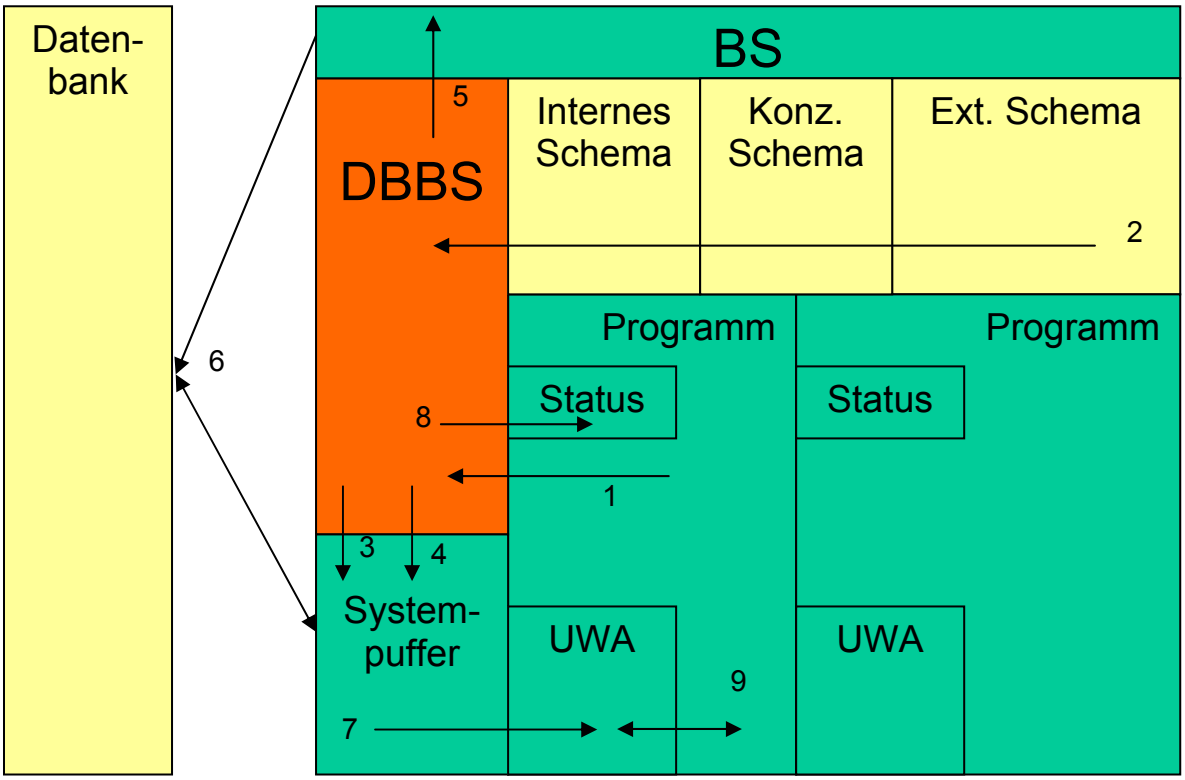
Konzeptionelles Schema

- Logische Gesamtsicht aller Daten
- Datenmodell (Abstraktion vom internen Schema)
- keine Zugriffspfade

Internes Schema

- Nicht physikalische Speicherung, Betrachtung als "Sätze"
- Zugriffspfadstrukturen
- Abbildung auf physische Speicherebene (OS-Funktionen)
- Auswahl spezieller Speicherstrukturen
- Data Storage Definition Language

Funktion eines DBBS



Verarbeitungsablauf

1. Statement an DBBS
   SELECT *
   FROM MITARBEITER
   WHERE MITARBEITER_NR = 12345
2. Vervollständigen des Statements mit Schema- und Subschemainformationen
   Speicherungsstruktur und Zugriffspfade aus internem Schema
3. Überprüfung Seite im Systempuffer, wenn gefunden dann weiter mit 7.
4. Auswahl einer Seite im Systempuffer zum Ersetzen
5. Vorbereitung/Anforderung physischer Ein-/Ausgabe
   Schreiben der zu ersetzenden Seite
6. Physischer I/O durch BS Abspeicherung
   Abspeicherung der Seite im Systempuffer
7. DBBS greift auf Satz im Systempuffer zu und stellt ihn entsprechend dem externen Schema zur Verfügung
8. Statusinformationen / Fehlermeldungen
   Typ des Satzes, Anzahl der Sätze, Fehler-Codes etc.
9. Zugriff auf Daten durch Programm und Änderung der Daten mit der Wirtssprache

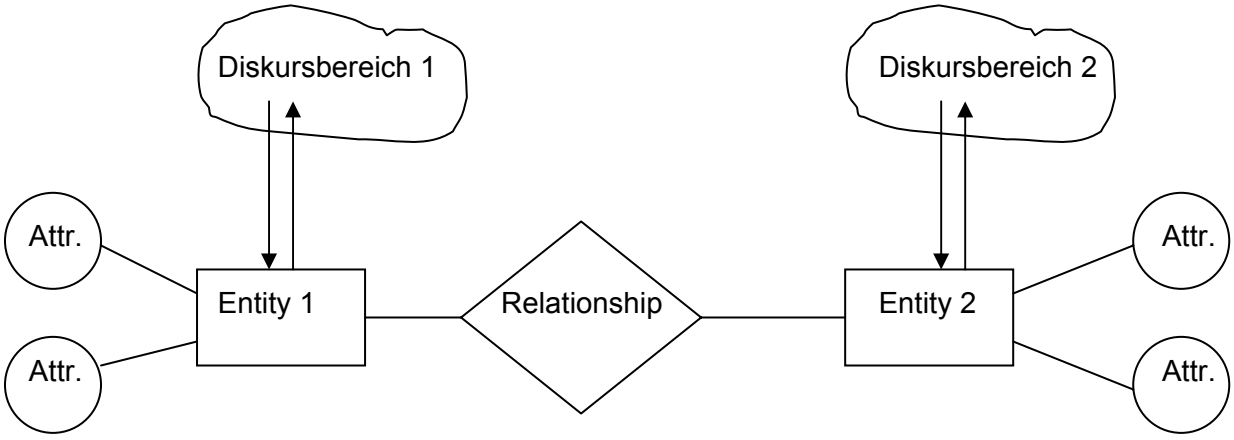
Zugriffspfadtechniken

- B-Bäume
- B*-Bäume
- Hash-Tabellen
- Grid-Files
- Ziel: Performancesteigerung

Datenmodellierung

- Entity-Relationship-Modell
- Entity-Relationship-Diagramme

Definition: Ein Entity-Relationship-Modell ist ein Ausschnitt aus realen Welt. Es beschreibt mit den Mitteln eines konkreten Datenmodells den Diskursbereich.



Objekttypen: Entity = Objekt
Relationship = Beziehungen

Ein Entity stellt ein individuelles Objekt der realen oder der Vorstellungswelt dar.

Eine Entity- Deklaration hat die Form

E = (attr (E), K)

Sie besteht aus einem Namen **E**, einem Format **attr (E)** und einem Primärschlüssel **K**, welcher aus Elementen von attr (E) zusammengesetzt ist.

Der Primärschlüssel muss eindeutig sein, am besten Einführung einer Nummer (Kundennummer, Artikelnummer ...)

KUNDE=((KUNDENUMMER,FIRMA,STRAßE,ORT...),KUNDENNUMMER)

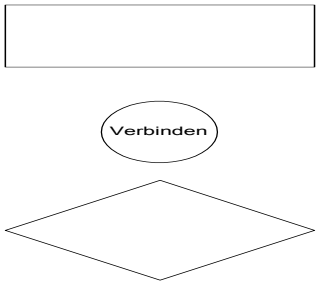
E attr (E) K

Entity-Deklaration

Definition Entity, Entity- Set

Sei E = (attr (E), K) eine Entity- Deklaration mit attr (E) = (A1) ... (Am)dom (A_i) sei der Wertebereich zu A_i (i=1...m).
K sei eine Teilfolge von attr (E).
Ein *Entity e* ist ein Element des kartesischen Produktes der Wertbereiche e € dom (A_1) x...x dom (A_m)
Ein *Entity-Set* zum Zeitpunkt t ist eine Menge von Entities, welche K erfüllt, d.h. E_t < dom (A_1) x...x dom (A_m)
E_t heißt aktueller Wert (Instanz) von E zum Zeitpunkt t.
Bsp. AUTO $_t$ = {(NJ77-07, 353W, 73, STANJA)}

Entity – Relationship - Diagramme



Entities

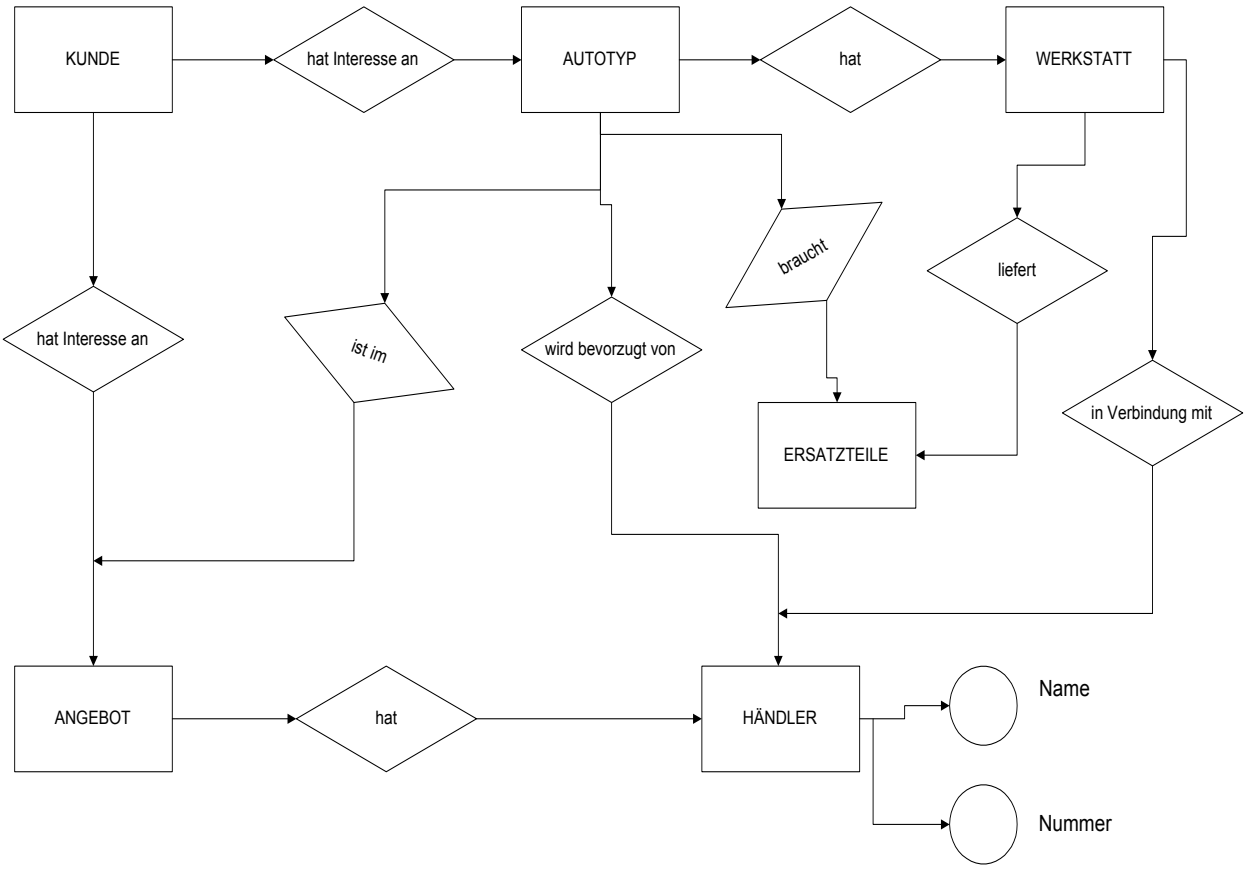
Attribute

Relationship

Beispiel ERM

E1: AUTOTYP=((NAME, NEUWERT, LEISTUNG), (NAME)
E2: WERKSTATT=((WNR, ORT, STRASSE), (WNR)
R: AUTO_WERKSTATT=((AUTOTYP; WERKSTATT), 0)
$E1_t$: AUTOTYP t ={ (WARTBURG, 10000, 46)
(OPEL, 18600, 80)
(FORD, 25000, 70)}
$E2_t$: WERKSTATT t = { (113, LEIPZIG, Kleinweg 13),
(119, DRESDEN, Markt 15),
(116, JENA, HAUPTSTR. 29)}
R_t: AUTO_WERKSTATT t ={ (WARTBURG, LEIPZIG),
(OPEL, JENA) }

Entity-Relationship-Diagramm



Assoziationstypen

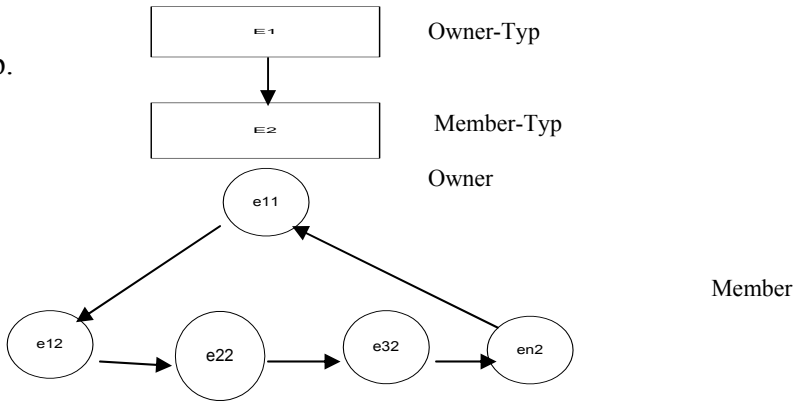
- 1 : 1 injektiv
- 1 : m multipel
- 1 : c conditional
- 1 : mc multipel conditional
- m : n

Vergleich ERM und Netzwerkmodell

Definition:

Ein *Bachmann-Diagramm* ist ein Graph B = (V,E), dessen
Eckenmenge V Entitydeklarationen repräsentiert, und dessen Kantenmengen E c VxV zweistellige Beziehungen (ohne Attribute) zwischen diesen darstellt. Ecken und Kanten sind bekannt.

Bsp.



Datenmodelle

- hierarchisches
- netzwerkartiges
- relationales

Hierarchisches Datenmodell

- Baumstruktur
- keine Vernetzung
- 1:n-Beziehungen

Das Netzwerkmodell

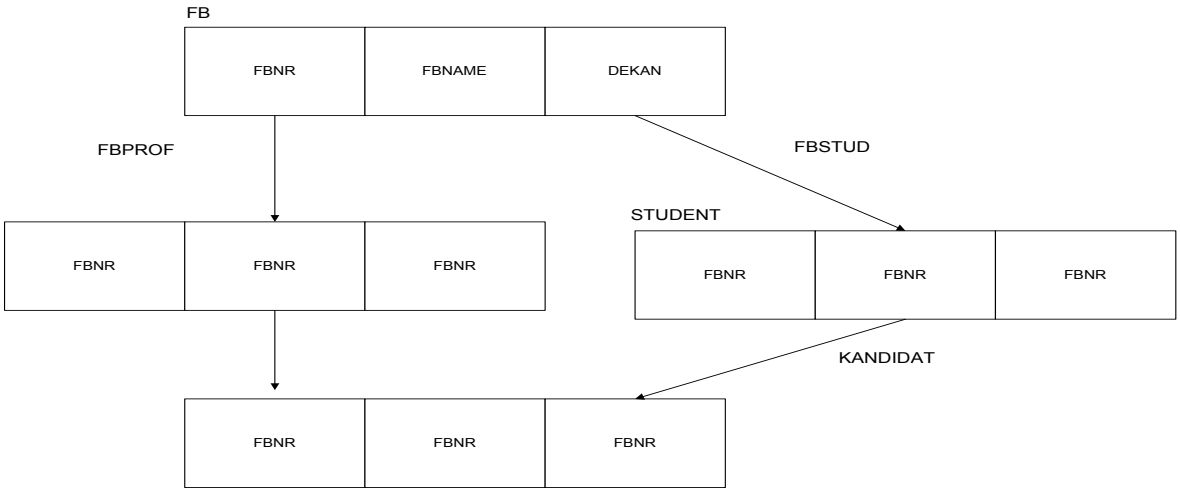
- CODASYL-Komitees (Data Base Task Group) => DTBG-Konzept
- unsymetrisch

Objekttypen
Objekte (Record-Typ)
Beziehungen (Set-Typ)

Nachteile:
- 1:n Beziehungen
- navigierende Programmierung
- record at a time
- Designer-Systeme

Vorteile:
- Darstellung komplexer Zusammenhänge im Modell
- Effizienz

Beispiel Netzwerkdatenmodell



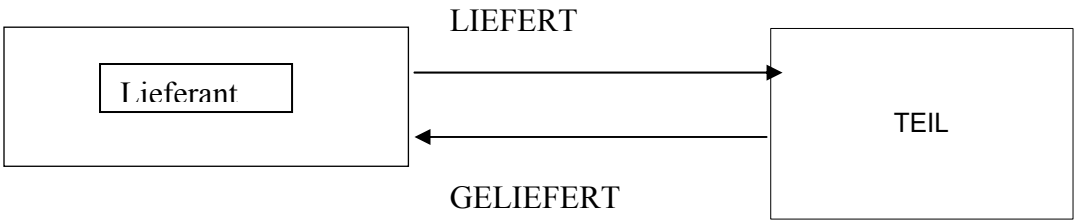
Datenstrukturdiagramm für die STUDENT_DB

Beispiel navigierende Pogrammmierung in Netzwerkdatenmodell

TA 2 Finde alle Studenten des Fachbereiches 20, die im Fach Datenverwaltung eine Note 2 oder besser erhalten haben.

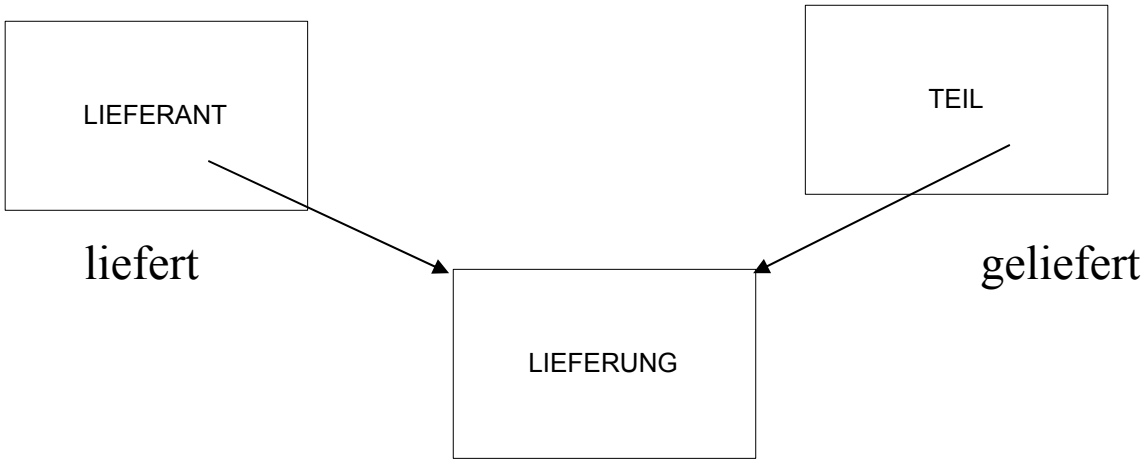
```
MOVE `FB20` TO FBNR OF FB ;
FIND FB USING FBNR ;
NEXT_STUD:     FETCH NEXT PRUEFUNG WITHIN FB-STUD SET ;
NEXT_PRUEF:    FETCH NEXT PRUEFUNG WITHIN KANDIDAT SET ;
IF `END_OF_SET` THEN GOTO NEXT_STUD ;
IF FACH = `DV` AND NOTE <= `2` THEN DO :
PRINT STUDENT RECORD ;
GOTO NEXT_STUD ;
END ;
GOTO NEXT_PRUEF
```

Darstellung von n:m - Beziehungen



Darstellung im NDM nicht möglich, da ein Member-Record nur einen Owner-Record besitzen darf.

IDEE:



Darstellung der n:m – Beziehung als eigenständiger Objekttyp (Relrec)

Das relationale Modell (CODD)

• IDEE:
- Speicherung der Daten in Tabellen
- Ordnung der Zeilen ohne Bedeutung
- Ordnung der Spalten ohne Bedeutung
- Verbindung der Daten über Inhalte nicht über Verzweigung

• BSP:

INVNR	TITEL	AUTOR	ISBN
1	Princ. DB	Ullmann	1234
2	DB	Date	2498
3	VDB	Oßwald	357

Das relationale Datenmodell

• **Definition:**
Ist (A_1,...,A_m) eine Folge von Attributen mit den Wertebereichen dom (A_i), i = 1(1)m, so ist eine Relation r eine Teilmenge des kartesischen Produkts dieser Domains.
r Ì dom (A_1) x...x dom (A_m)
Sei X = {A_1,...,A_m}, so bezeichnet Tup (x) die Menge aller Tupel. Eine Relation r über x ist eine endliche Menge von Tupeln über x, d.h. r Ì Tup (x) Rel(x) bezeichnet die Menge der Relationen über x. r Ì Rel(x)

Relationales Datenmodell und ERM

ERM	Relationales Modell
Entity, Relationship-Deklaration	Relationenschema
Entity, Relationship-Set	Relation
Entity, Relationship	Tupel

Nullwerte

• unbekannt
• existiert, aber unbekannt
• existiert nicht
• dom (A) = {1,2,9,8}
NULL wird zu „jedem“ Wertebereich hinzugenommen
dom (A) = {1,2,9,8} È NULL
ÞDreiwertige Logik
- wahr
- falsch
- NULL

Relationenschema

• Definition:

Ein *Relationenschema* hat die Form R = (x Sx) ;
Es umfasst einen Namen (R), eine Attributmenge x und eine Menge Sx intrarelationaler Integritätsbedingungen.

Zeitinvariante Beschreibung

• Definition:
Die Menge aller Relationen R e Rel(x) die Sx erfüllen bezeichnen wir mit
Sat (x, Sx):= {r e Rel(x) Sx (r) = 1}

Integritätsbedingungen

Klassifikation nach:
– Wirkungsbereich
– Modellabhängigkeit
– Reaktion auf Verletzung
– Reichweite
– Überprüfungszeitpunkt
– Wirkungsbereich
– intrarelational
– interrelational

Überprüfungszeitpunkt
– Sofort
– verzögert

Reaktion auf Verletzung
– Abbruch (zurücksetzen der TA)
– Kompensations –TA (Trigger)
– Warnung

Reichweite
– Attribut
– Tupel
– Menge von Tupeln einer Relation
– Relation
– Menge von Tupel verschiedener Relationen
– Verschiedene Relationen

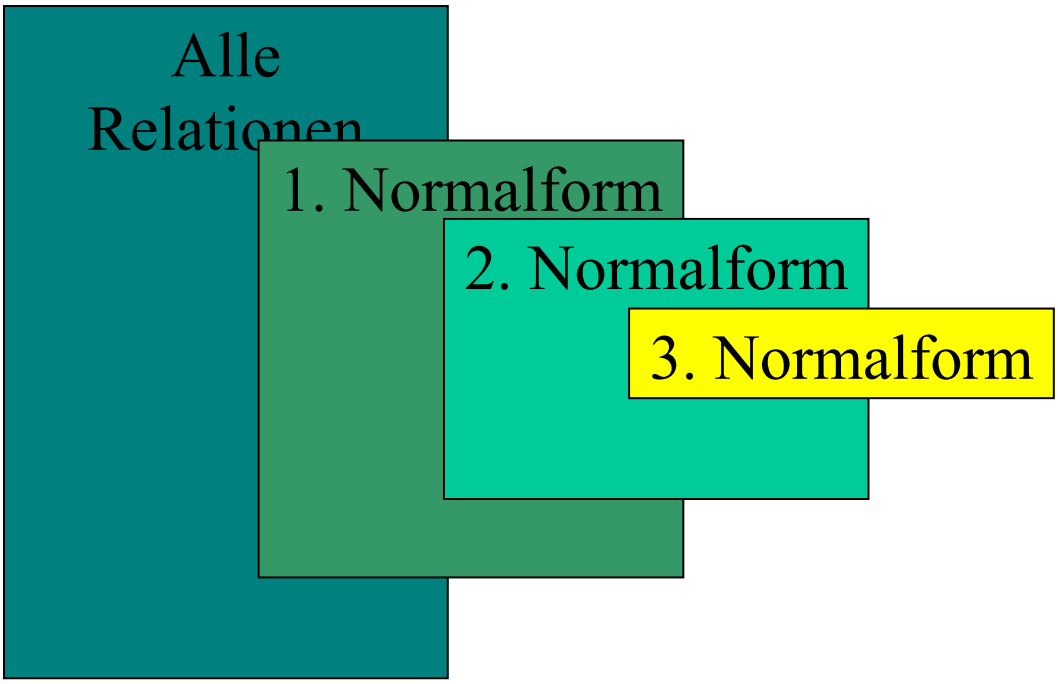
Schlüssel

Definition:
Attributkombinationen heißen *Schlüsselkandidat*, wenn sie die Schlüsseleigenschaft besitzen.

Definition:
Primärschlüssel heiß ein ausgezeichneter Schlüsselkandidat.

Definition:
Fremd- oder *Sekundärschlüssel* heißt eine Attributkombination, die in einer anderen Relation Primärschlüssel ist.
Attribute, die zu einem Schlüssel gehören, heißen *Schlüsselattribute*.

Normalformenlehre



1. Normalform

Definition:

Eine Relation ist in 1. Normalform, wenn alle Attribute atomar sind.

Bsp.:

PERSON (NAME, VORNAME, PKZ, ...)
PKZ 170362 4 210 28
GEBDAT GESCHL Wohnort

Volle funktionale Abhängigkeit

Definition:

Sei v eine Attributmenge und x, y Í v.
y heißt voll funktional abhängig von x, wenn kein x ´Ì x existiert,
so dass x´ -> y gilt.

Bsp.:

v = ABCD r e Rel(v)

	A	B	C	D
	1	2	9	0
r:	2	3	6	4
	3	2	9	1
	4	5	7	3

F= {AB -> C, AB -> D, B-> C}
- ⇨ C nicht voll funktional abhängig von AB, aber von B

2. Normalform

Definition:
Eine Relation befindet sich in 2. Normalform, wenn sie sich in 1. Normalform befindet und alle Attribute voll funktional vom Primärschlüssel abhängig ist.

Transitiv abhängig

• Definition: Eine Menge von Attributen $Z=c_1,c_2...c_n$ in R ist transitiv abhängig von einer Menge von Attributen $X=a_1, a_2,...a_m$ wenn
• X und Z disjunkt sind
• Eine Menge von Attributen $Y=b_1, b_2, ...b_l$ in R existiert, die disjunkt zu X und Z ist so dass gilt
• R.X ® R.Y und R.Y ®Ø R.X
• R.Y ® R.Z

3. Normalform

Definition:
Eine Relation befindet sich in 3. Normalform, wenn sie sich in 2. Normalform befindet und jedes Nicht-Primärattribut von R nicht transitiv abhängig von jedem Schlüsselkandidaten von R ist.

Normalisierung

• Starte mit der Vaterrelation
• Nimm den Primärschlüssel der Vaterrelation und bilde mit dem Primärschlüssel und jeder unmittelbar untergeordneten Relation eine neue selbständige Relation
• Streiche alle nicht einfachen Attribute aus der Vaterrelation
• Wiederhole den Prozess in jeder Sohnrelation

Anomalien in relationalen DB

INSERT - Anomalie
DELETE - Anomalie
MODIFY - Anomalie

R =(U,F) mit
U =L_ieferant, T_eil, A_rtikel, O_rt, P_ostleitzahl
F = {LT->A, L->O, =-> P}

L	T	A	O	P
L1	T1	300	Jena	07743
L1	T2	200	Jena	07743
L1	T3	400	Jena	07743
L1	T4	200	Jena	07743
L1	T5	100	Jena	07743
L1	T6	100	Jena	07743
L2	T1	300	Gera	07546
L2	T2	400	Gera	07546
L3	T2	200	Weimar	99423
L4	T2	200	Kahla	07768
L4	T4	300	Kahla	07768
L4	T5	400	Kahla	07768

SQL

Structured
Query
Language

Bestandteile:

– Data Rerieval Language

– Data Manipulation Language
– Data Description Language
– Data Storage Description
– Data Access Language

[ ] = optional
{} = beliebig oft
| = alternativ

Erzeugen einer Relation
CREATE TABLE

Folgendes muss spezifiziert werden:
• Name der Tabelle
• Namen der Spalten
• Datentyp jeder Spalte
• Breite der Spalten
andere optionale Informationen

CREATE TABLE tablename (columnname datatype)
{, columnname datatype})

Bsp. CREATE TABLE

Eingabe des Tabellenschemas „adresse“
ADR = (ADRID, PLZ; STADT, STR, HNR), (ADRID))

CREATE TABLE adresse (adrid integer 2, plz integer 2
stadt vchar (10), str vchar (20)
hnr ingeger 2);
Ergebnis:
table: adress

adrid	plz	stadt	str	hnr

Konsistente Relationale DB

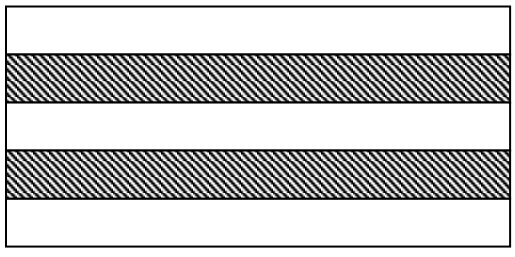
Definition:
Es sei R eine Menge von Relationenschemata, S?R eine Menge interrelationaler Abhängigkeiten.
(i) Ein DB-Schema im relationalen Modell ist ein benanntes Paar
D = (R, S?R)
Relationale DB-Schema dienen der zeitinvarianten Beschreibung der Menge
Sat (D):=Sat(R, S?R):= Sat(R) Sat (S?R)
(ii) Eine DB d heißt konsistent, galls de € Sat (D) heißt aktueller Zustand oder Beispiel von D.

Relationale Algebra

- Operatoren
- Selektion
- Projektion
- Join

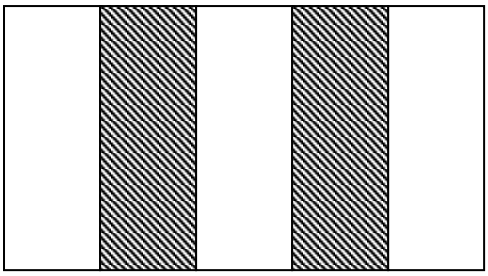
Selektion

- Auswahl von Tupel aus der Relation
- Vergleichsoperatoren <, =, >,<>, ...
- Logische Operatoren und ^, oder v, nicht –



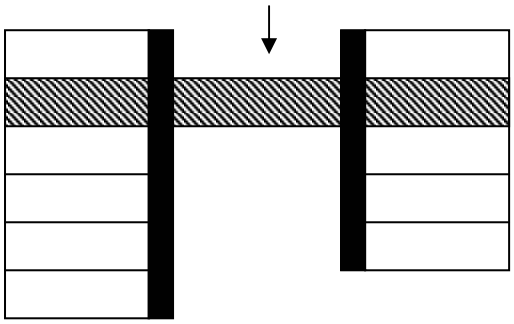
Projektion

- Auswahl von Attributen



Join

- Join (Verbund) von 2 Relationen über ein Attribut mit gleichem Wertebereich
- Ergebnis sind alle Tupel der „linken“ Relation verbunden mit allen Tupeln der „rechten“ Relation, die im Join-Attribut den gleichen Wert besitzen



Outer – Join

IDEE:
– Kein Informationsverlust durch Join
– Auffüllen mit Nullwerten
TEIL (TNR, STÜCK, LAGERORT)
PREIS (TNR, PREIS)
SELECT Teil.*., PREIS.*
FROM TEIL, PREIS
WHERE TEIL.TNR = PREIS.TNR
Alle Teile ohne Preis verschwinden
Alle Teile ohne Lagerbestand fehlen

Left Outer Join

alle Tupel der „linken“ Relation werden mitgeliefert, auch wenn das Join-Attribut in der „rechten“ Relation keinen Partner hat.

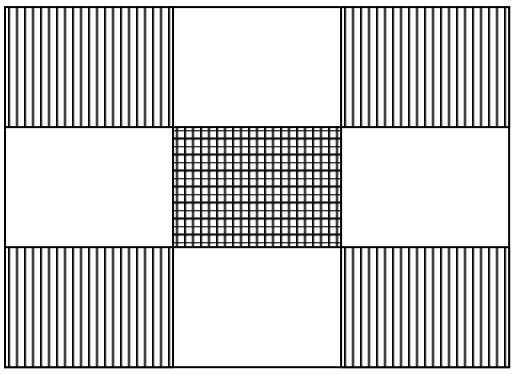
Right Outer Join

wie Left, nur umgekehrt

Full Outer Join (selten implentiert)

alle Tupel der “linken” und der “rechten” Relation werdengeliefert, auch wenn sie keinen Join-Parameter haben.

Mehrfachanwendung



Bsp. Tabelle anlegen:

CREATE TABLE tabelle
(spalte1,
spalte2,
…,
PRIMARY KEY (spalte1))
CREATE UNIQUE INDEX indexname **ON** tabelle (primari key);

Wichtig, wenn ein Primary Key festgelegt wurde !

CREATE TABLE rezept
(rezept_nr **INTEGER NOT NULL**,
Name **VARCHAR**(20),
beschreibung **VARCHAR**(250),
PRIMARY KEY (rezept_nr)),
CREATE UNIQUE INDEX rezept_nr **ON** rezept (rezept_nr);

Bsp. Selektion:

SELECT *
FROM tabelle
WHERE bedingung;

SELECT *
FROM rezept
WHERE rezept_nr > 20;

Bsp. Join:

SELECT *
FROM tabelle1,tabelle2
WHERE bedingung;

SELECT *
FROM rezept,zutaten
WHERE rezept_nr = zut_nr;

Bsp. Werte einfügen

INSERT INTO tabelle (spalte1,spalte2,...) **VALUES** (Wert1,Wert2,...);

INSERT INTO rezept (rezept_nr,name,beschreibung) **VALUES** (1,Wasser,?);

Achtung, bei Spalten mit dem Wert NOT NULL muß unbedingt ein Wert eingegeben werden !

Bsp. Werte löschen

DELETE FROM tabelle **WHERE** bedingung;

DELETE FROM rezept **WHERE** rezept_nr = 10;