Folien zur Vorlesung
Datenbanken

Kapitel 6

Relationale Datenbanksysteme

Fachhochschule Wedel

Prof. Dr. Ulrich Hoffmann

basierend auf den Lehrmaterialien von
Prof. Dr. Hans–Detlef Gerhardt

6.1 Das 3 - Ebenen - Architekturkonzept

Modellierung erfordert **formale Sprache.**

Bei Datenbanken stehen **statische Gegebenheiten** im Vordergrund:

- Welche Objekte gibt es?
- Welche Eigenschaften haben diese Objekte?
- Welche Beziehungen bestehen zwischen den Objekten?

Bei Kenntnis des Gegenstandsbereiches können diese Fragen beantwortet werden unabhängig von der internen Abspeicherung der Daten im Rechner.

Man spricht vom **logischen Datenbankentwurf oder logischer Datenorganisation**, diese ist getrennt von den Implementierungsaspekten.

Datenunabhängigkeit ist zentrales Entwurfsprinzip für Datenbanken.

Nutzer sehen nur die logische Struktur der Daten.

Logische Datenorganisation:

- **Objekte:** viele besitzen die gleiche Struktur **einheitliche Beschreibung**
- **Beziehungen** zwischen Objekten

Operationen auf Daten:	klassisch	-	**prozedural**
	neu	-	**deskriptiv**

Wesentliches Ziel:

Datenunabhängigkeit zwischen Daten und Programm/Benutzer:

- **physische Datenunabhängigkeit**
- **logische Datenunabhängigkeit**

physische Datenunabhängigkeit:

für Programme oder Dialoge mit der Datenbank soll deren **physische Organisation** unsichtbar sein, d.h. ein Programm oder eine Dialogoperation soll nicht auf die konkrete Wahl der Datenstruktur oder die daraus resultierenden Zugriffspfade Rücksicht nehmen müssen.

logische Datenunabhängigkeit:

im Hinblick auf leistungsfähige Benutzerschnittstellen wünschenswert, d.h. man möchte unterscheiden zwischen

- spezifischer, anwendungsbezogener Sicht auf die Datenbank
- logischer Gesamtstruktur.

Diese log. Struktur soll veränderbar sein, ohne dass alle lokalen Anwendungen entsprechend geändert werden müssen.

⇒ Sicht der Daten auf **drei Ebenen** nahe liegend:

- physische Dateiorganisation
- logische Gesamtsicht der Daten
- Benutzersicht.

Beispiel:

Buchbestand wird verwaltet durch Kartei mit folgenden Aufbau:
1. Zeile: Inventarnummer
2. Zeile: ISBN
3. Zeile: Autor
4. Zeile: Titel
5. Zeile: Fachgebiet
6. Zeile: Verlag
7. Zeile: Ort, Jahr
8. Zeile: Auflage
9. Zeile: Preis

physische Organisation:
Kartei mit Karten aller Bücher, sortiert nach Autor

logische Sicht:
beschreibt die Angaben, welche die Zeilen einer Karteikarte enthalten

Benutzersicht:
z.B. Autor und Titel aller Bücher des Gebietes Datenbanken.

Aspekte heißen **Datenunabhängigkeit:**
Betrachtung erfolgt auf 3 **Abstraktionsebenen** (ANSI|X3|SPARC **-** Architekturkonzept für Datenbanksysteme)

Abstraktionsebenen:

- **Interne Ebene**
  nahe physikalischen Speicher, physisch gespeicherte Blöcke werden nicht als Pages oder Blöcke, sondern als „interne Records“ (Datensätze) betrachtet.

Die interne Sicht der Daten („interne Records“, (Datensätze))
wird im **internen Schema** festgelegt:

- Informationen über Art und Aufbau der verwendeten Datenstrukturen
- Informationen über die Organisation der Sätze im logischen Adressraum
- spezielle Zugriffsmechanismen auf die Daten

Schnittstelle zur eigentlichen Datenbank hat damit z.B. die
Abbildung des logischen in den physischen Adressraum
(meist unter Rückgriff auf vorhandene Betriebssystem – Funktionen)
zu gewährleisten.

Konzeptionelle Ebene:

- **logische Gesamtsicht** der Daten in der Datenbank im konzeptionellen Schema dargestellt (Datenmodell)

- Schema ist frei von Datenstruktur- und Zugriffsaspekten

Das konzeptionelle Schema beinhaltet ausschließlich eine Definition des Informationsgehaltes der Datenbank.

Zur Entwicklung eines konzeptionelle Schemas verwendet man **Datenmodelle**.

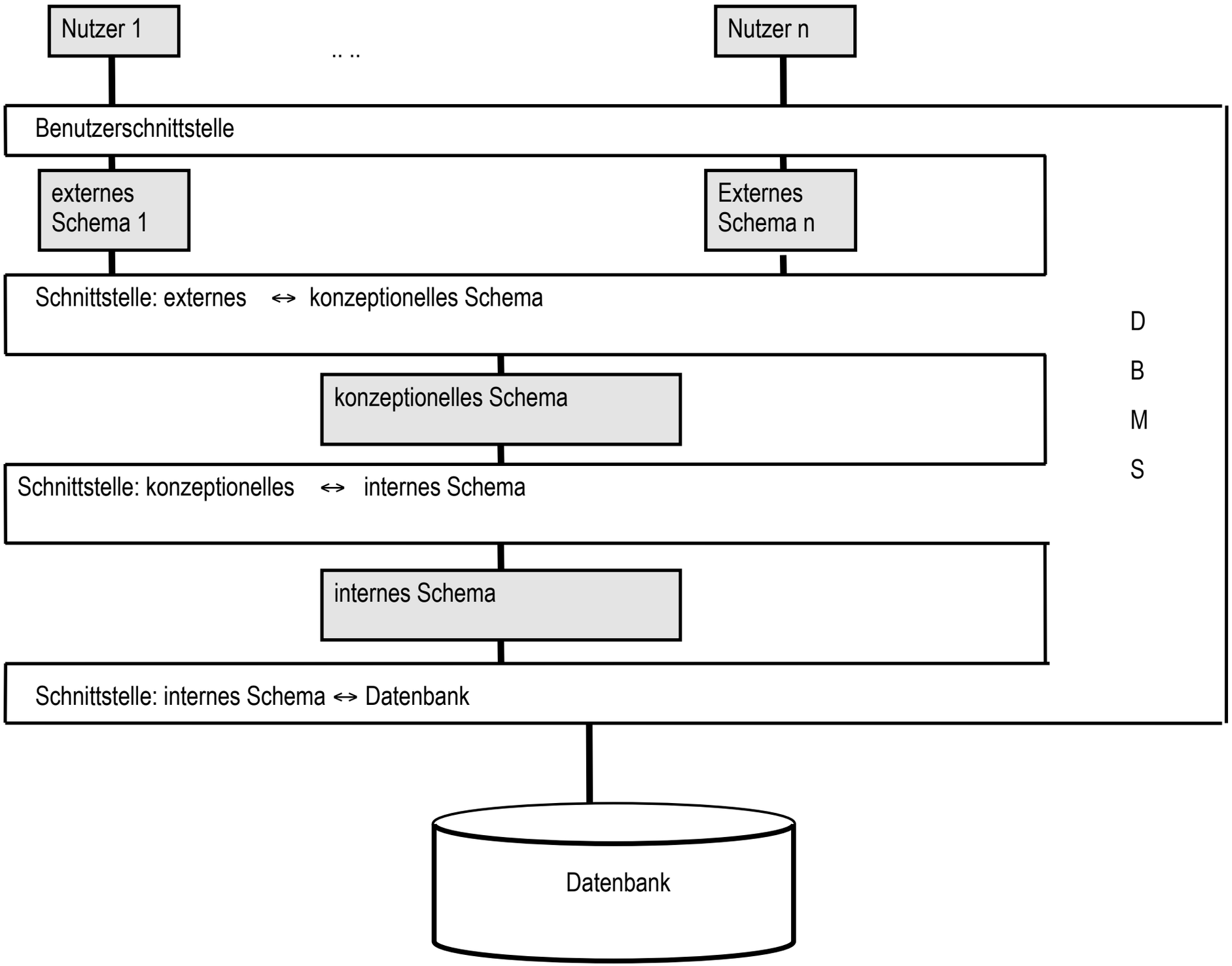
Zur **Implementation** steht die „**Sprache**“ des gewählten Datenmodells zur Verfügung (DDL des konkreten DBMSs)

Externe Ebene:

- umfasst alle **individuellen Sichten** (Views) der Nutzer des Datenbanksystems.

- Sichten werden in einem eigenen externen Schema beschrieben, enthält genau den Ausschnitt der konzeptionellen Sicht, den der Nutzer sehen möchte /darf.

3 - Ebenen Architektur



Die einzelnen Ebenen **kommunizieren** untereinander bzw. mit ihrer jeweiligen Außenwelt **über Schnittstellen**, welche vom DBMS zur Verfügung gestellt werden.

Dabei handelt es sich um **Transformationen,** in welchen einerseits die **„Entsprechungen“** zwischen dem externen Schema und dem konzeptionellen Schema, zwischen dem konzeptionellen Schema und dem internen Schema, bzw. zwischen dem internen Schema und der Datenbank festgelegt sind.

Andererseits stellt ein DBMS stets eine **bestimmte Schnittstelle zum Nutzer** bereit, welche zwischen der externen Ebene und den darauf arbeitenden Nutzern anzusiedeln ist, diese **Schnittstelle umfasst eine Sprache,** über welche die Nutzer mit der Datenbank bzw. den DBMS kommunizieren.

Beispiel: „Implementation“ der Buchkartei

Datenstruktur: lineare Liste,

Bücher nach Autoren sortiert.
Listenelement vom record - Datentyp.
Zeiger auf Kopf der Liste und Hilfsvariable zum sequentiellen Durchlaufen der Liste.

Physisch besteht die Datenbank aus einer **einfach verketteten Liste,** welche sequentiell verarbeitet werden kann.

Die **Bearbeitung** einer Anfrage erfolgt **satzorientiert.**

Einfügen eines neuen Datensatzes erfordert:

1. Stelle lokalisieren, an der der Datensatz eingefügt werden soll und
2. Ändern der Verkettung zu den Nachbarrecords.

Konzeptionelle Ebene:

Im Mittelpunkt steht die **Information** über die Objekte.

Buchbestand ergibt sich als **Set von Records.**

Die Verarbeitung von Anfragen erfolgt **mengenorientiert.**

Externe Ebene:

Externe Sicht „Fachgebiet Datenbank“ in der Sprache des DBMS:

```
type s1 =      if bb.fachgebiet =  'DB'
                  then record  bb.autor,
                               bb.titel
               end;

sicht1 = set of s1;
```

Vor der Arbeit mit einer Datenbank steht das **Einrichten** der Datenbank:

- Definition aller logischen und physischen Strukturen (DBMS stellt DDL und DSDL bereit)
- Definition des externen Schemas.

Der Anwender einer bereits vorhandenen Datenbank arbeitet auf der **externen Ebene:**

- Anfragen (queries) stellen
- Veränderungen (updates) (Einfügen, Ändern, Löschen) vornehmen.

Dazu **DML**:

- eigenständige Dialogsprache
- in Wirtssprache eingebettet, d.h. in einer höheren Programmiersprache geschriebene Programme dürfen DML – Kommandos benutzen, diese werden mittels **Precompiler** in ausführbare Zugriffsmodule übersetzt.

DDL + DML + DCL:

klare Trennung oder de facto nur eine Sprache.

6.2 Transaktionskonzept

Konsistenz
logisch getreue, widerspruchsfreie Modellierung der betroffenen Miniwelt in der Datenbank

Miniwelt:	**Zustands-** bedingungen	**Übergangs-** bedingungen
↓	↓	↓
Datenmodell:	**strukturelle Eigenschaften im Schema**	**Operator-semantik**

dazu: explizite **Konsistenzbedingungen**

Transaktion

Folge von Datenbank - und anderen Operationen (Arbeitseinheiten) mit folgenden Eigenschaften:

- **Atomarität**
- **Konsistenz**
- **Isolation**
- **Dauerhaftigkeit**

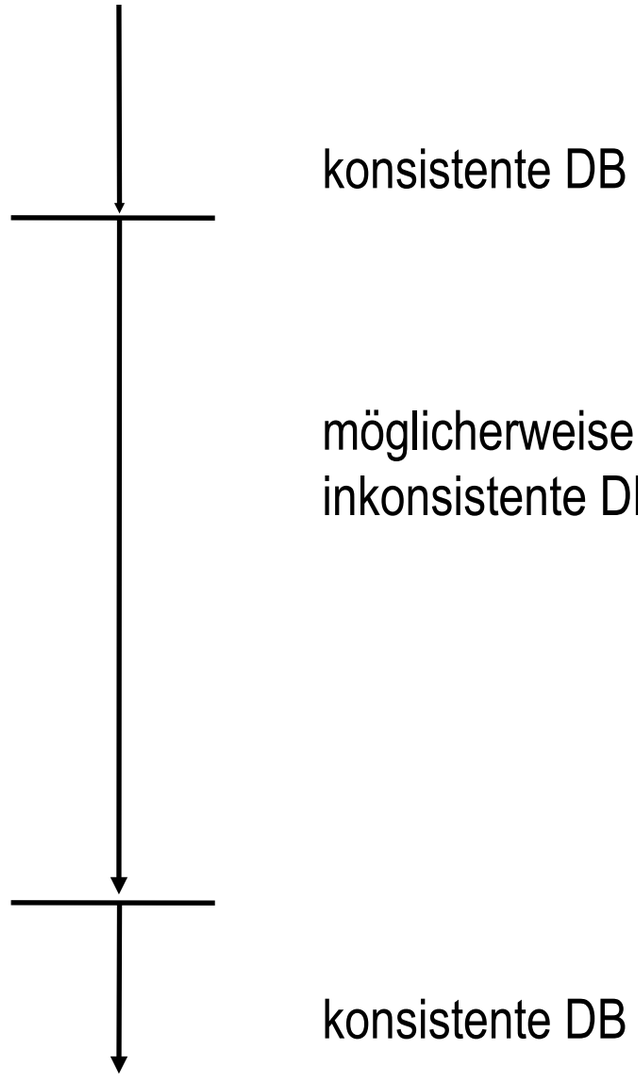
ACID

Ablauf einer Transaktion:

BEGIN TRANSACTION

op1
-
-
-
op n

END TRANSACTION



Transaktionsverwaltung

- Konsistenzsicherung
- Synchronisation des Mehrbenutzerbetriebes
- Datensicherung

6.3 Grundlagen der Datenmanipulation

- Grundlegende Operationen auf den Relationen einer relationalen Datenbank
- Relationale Operationen erzeugen aus Tupelmengen (Relationen) neue abgeleitete Tupelmengen (Relationen)
- Aus Basisrelationen werden durch Operationen abgeleitete Relationen erzeugt.
- Basis für Operationen: Schemainformationen

Anfrage

Formaler Ausdruck, der dem System übermittelt wird.
Wird vom System ausgewertet unter Ersetzung aller Namen durch den aktuellen Inhalt.

6.3.1 Relationale Operationen

1. Einstellige Operationen

Definition:

Sei R = (X, .) ein Relationenschema, r ∈ Rel(X) und Y ⊆ X.

a) π_Y **(r) := {** μ**[Y] :** $\mu \in$**r }** heißt **Projektion** von r auf Y.

$\sigma_{A\theta\, a}$(r) := { $\mu \in$ r : μ[A] θ a} heißt **Selektion** von r bezüglich A θ a,
wobei A ∈ X, a ∈ dom(A) und $\theta \in \{<, \leq, >, \geq, =, \neq\}$ ist.

Beispiel: r ∈ Rel(ABC):

r:

A	B	C
1	2	1
1	1	1
2	2	2

Beachte bei Projektionen:

Das Ergebnis enthält als Menge keine doppelten Tupel.

Beachte bei Selektion:

Selektionsbedingungen dürfen durch $\wedge$, $\vee$, $\neg$ zu einer **komplexen Bedingung C** zusammengesetzt werden, die darin vorkommenden Attribute werden mit attr(C) bezeichnet.

Rechenregeln:

Es sei $r \in Rel(X)$, C_i für i = 1,2 komplexe Bedingungen:

1.	$Z \subseteq Y \subseteq X$	$\Rightarrow$	$\pi_Z(\pi_Y(r))$	$= \pi_Z(r)$
2.	$Z, Y \subseteq X$	$\Rightarrow$	$\pi_Z(\pi_Y(r))$	$= \pi_{Z \cap Y}(r)$
3.			$\sigma_{C1}(\sigma_{C2}(r))$	$= \sigma_{C2}(\sigma_{C1}(r))$
4.	$A \in Y \subseteq X$	$\Rightarrow$	$\pi_Y(\sigma_{A\theta a}(r))$	$= \sigma_{A\theta a}(\pi_Y(r))$
5.	$attr(C) \subseteq Y \subseteq X$	$\Rightarrow$	$\pi_Y(\sigma_C(r))$	$= \sigma_C(\pi_Y(r))$.

(attr(C) : Attribute der komplexen Bedingung C)

2. Zweistellige Operationen

Definition:

Seien r, s ∈ Rel(X):

a) Durchschnitt:
r ∩ s := { μ ∈ Tup(X) : μ ∈ r ∧ μ ∈ s}

b) Vereinigung:
r ∪ s := { μ ∈ Tup(X) : μ ∈ r ∨ μ ∈ s}

c) Differenz:
r – s := { μ ∈ Tup(X) : μ ∈ r ∧ μ ∉ s}

Beispiele:

r:

A	B	C
1	2	1
1	1	1
2	2	2

s:

A	B	C
1	2	1
2	1	2
3	3	3

Definition:

Seien $X_1, ..., X_n$ Attributmengen und $r_i \in Rel(X_i)$ für i=1 (1)n:

$$\overset{n}{\underset{i=1}{\bowtie}} (r_i) := \{ \mu \in Tup (\overset{n}{\underset{i=1}{\cup}} X_i) : \mu[X_i] \in r_i \quad \forall i = 1 (1) n \}$$

heißt der **natürliche Verbund** (natural join) von $r_1, ..., r_n$.

$\Rightarrow$ für n=2

$$r_1 \bowtie r_2 = \{\mu \in Tup (X_1 X_2) : \mu[X_1] \in r_1 \wedge \mu[X_2] \in r_2\}$$

Beispiel:

r_1

A	B	C
1	2	1
1	2	2
2	0	2

r_2

A	D
1	1
0	1
2	0

r_3

C	D	E
1	1	0
0	1	1
2	1	0
2	2	1

Rechenregeln:

Es seien $r_i \in Rel(X_i)$ für i = 1(1)3:

1.	$r_1 \bowtie r_1$	=	r_1
2.	$r_1 \bowtie r_2$	=	$r_2 \bowtie r_1$
3.	$(r_1 \bowtie r_2) \bowtie r_3$	=	$r_1 \bowtie (r_2 \bowtie r3)$
4.	$X_1 \cap X_2 = 0$	$\Rightarrow$	$r_1 \bowtie r_2 = r_1 * r_2$
5.	$X_1 = X_2$	$\Rightarrow$	$r_1 \bowtie r_2 = r_1 \cap r_2$

Definition:

Seien r ∈ Rel(X), s ∈ Rel(Y):

r ⋉ s := πx (r ▷◁ s) heißt **Semijoin** von r mit s.

Definition:

Seien r ∈ Rel(X), s ∈ Rel(Y):

$$r * s := \{\mu v : \mu \in r \ \wedge v \in s\}$$

heißt die **Konkatenation** von r und s.

Gilt $X \cap Y = \varnothing$, dann ist r * s eine **Relation** und r * s ∈ Rel(XY).

Beispiele:

r:

A	B
0	0
1	1

s:

C	D
1	0
0	1

r:

A	B
0	0
1	1

t:

B	D
1	0
0	1

Definition:

Seien $r \in Rel(X)$, $s \in Rel(Y)$, $A \in X$, $B \in Y$, $\theta \in \{<, \leq, >, \geq, =, \neq\}$:

$$\mathbf{r[A\ \theta\ B]s := \{\mu\nu \in r * s : \mu[A]\ \theta\ \nu[B]\}}$$

heißt **Theta-Join** (θ-Join) von r und s.

Ist θ speziell „=“, so spricht man vom **Equijoin** von r und s.

Gilt $X \cap Y = \emptyset$, so ist $r * s \in Rel(XY)$ und es gilt

$$\mathbf{r[A\ \theta\ B]s = \sigma A\ \theta\ B\ (r*s).}$$

Dabei ist definiert für $r \in Rel(X)$ und $A, B \in X$

$$\mathbf{\sigma A\ \theta\ B\ (r) = \{\mu \in r : \mu[A]\ \theta\ \mu[B]\}.}$$

6.3.2 Relationenalgebra

Definition:

Sei **D = (R, $\sum$ R)** ein Datenbankschema mit R ={R1, ..., RK).

Die Menge **RA_D** (kurz RA) der Ausdrücke der **Relationenalgebra** (über D) wird rekursiv wie folgt definiert:

1. $R_i \in RA$ für alle $R_i \in R$.
2. Sind $E_1 \in RA$, $E_2 \in RA$ und C eine Selektionsbedingung,

   $X \subseteq \bigcup_{i=1}^{K} X_i$, so ist

$\sigma_c(E_1)$	$\in$	**RA**	**Selektion**
$\pi_x(E_1)$	$\in$	**RA**	**Projektion**
$E_1 \rhd\lhd E_2$	$\in$	**RA**	**Join**
$E_1 \cup E_2$	$\in$	**RA**	**Vereinigung**
$E_1 - E_2$	$\in$	**RA**	**Differenz**

3. Nur solche Ausdrücke gehören zu RA, welche durch wiederholte Anwendung von 1. und 2. entstehen.

Definition:

Seien $D = (R, \sum_R)$ und $E \in RA$.

Die Auswertung $aw_E(d)$ von E bezüglich $d \in Dat(R)$ wird wie folgt definiert:

$$aw_E(d) := \begin{cases} r_i & \text{falls } E = R_i \\ \sigma_c(aw_{E1}(d)) & \text{falls } E = \sigma_c(E_1) \\ \pi_x(aw_{E1}(d)) & \text{falls } E = \pi_x(E_1) \\ awE_1(d) \rhd\lhd awE_2(d) & \text{falls } E = E_1 \rhd\lhd E_2 \\ awE_1(d) \cup awE_2(d) & \text{falls } E = E_1 \cup E_2 \\ awE_1(d) - awE_2(d) & \text{falls } E = E_1 - E_2 \end{cases}$$

Beispiel:

Bibliotheksdatenbank mit d = {b, l, elhn}

1. Zeige alle Bücher vom Autor ‘Meier’:

$$E_1 = \sigma_{AUTOR = 'Meier'}(B)$$

$$\Rightarrow$$

$$aw_{E1}(d) = \sigma_{AUTOR = 'Meier'}(b)$$

2. Zeige alle Verlage, von denen Bücher vorhanden sind

$$E_2 = \pi_{VERLAG}(B)$$

$$\Rightarrow$$

$$aw_{E2}(d) = \pi_{VERLAG}(b)$$

3. Zeige die Rückgabedaten der Bücher mit den Inventarnummern 1 bzw. 2:

$$E_3 = \pi_{RDAT}(\sigma_{INVNR=1 \vee INVNR=2}(ELHN))$$

$$\Rightarrow$$

$$aw_{E3}(d) = \pi_{RDAT}(\sigma_{INVNR=1 \vee INVNR=2}(elhn))$$

4. Zeige die Adressen der Leser, die zur Zeit ein Buch entliehen haben:

$$E_4 = \pi_{ADRESSE}\ (L \rhd\lhd ELHN)$$

$$\Rightarrow$$

$$aw_{E4}(d) = \pi_{ADRESSE}\ (l \rhd\lhd elhn)$$

5. Zeige Buchtitel und Lesernamen jedes aktuellen Entleihvorganges:

$$E_5 = \pi_{TITEL,\ NAME}\ (B \rhd\lhd L \rhd\lhd ELHN)$$

$$\Rightarrow$$

$$aw_{E5}(d) = \pi_{TITEL,\ NAME}\ (b \rhd\lhd l \rhd\lhd elhn)$$

6. Zeige die Buch-Relation:

$$E_6 = B$$

$$\Rightarrow$$

$$aw_{E6}(d) = b$$

Unterscheidung zwischen

- Anfrage - Ausdruck auf konzeptueller Ebene
- Auswertung auf interner Ebene

Ermöglicht

- eine **formale Ebenentrennung**
- ist Grundlage einer **implementierungsunabhängigen Optimierung** relationaler Ausdrücke

Definition: Sei $D = (R, \sum_R)$

E und E' seien Ausdrücke aus RA.

E und E' heißen äquivalent,
kurz $E \approx E'$, falls gilt

$$(\forall\ \mathbf{d} \in \mathbf{Dat}\ (R))\ \mathbf{aw_E(d) = aw_{E'}(d)}$$

Umformungen auf Basis der Kenntnis von

- **algebraischen Gesetzen**
- **Schemainformationen**,

ohne Änderung des Abfrageergebnisses.

Beispiele für E ≈ E':

a) Seien E = R1 ⋈ R2 und E' = R2 ⋈ R1 .

(∀ d ∈ Dat(R)) gilt

$aw_E(d) = r_1 \bowtie r_2 = r_2 \bowtie r_1 = aw_{E'}(d)$.

b) Seien E = R1 ⋈ R2 und E' = R1 ⋈ (R1 ⋈ R2)

$aw_E(d) = r_1 \bowtie r_2 = (r_1 \bowtie r_1) \bowtie r_2 = r_1 \bowtie (r_1 \bowtie r_2) = aw_{E'}(d)$.

Seien R1 = (AB, .), R2 = (BC, .),

d) E = $\sigma_{A=1}(R_1 \bowtie R_2)$

E' = $\sigma_{A=1}(R_1) \bowtie R_2$

$awE(d) = \sigma_{A=1}(r_1 \bowtie r_2) = \sigma_{A=1}(r_1) \bowtie r_2 = aw_{E'}(d)$

Seien R1 = (AB, .), R2 = (CD, .),

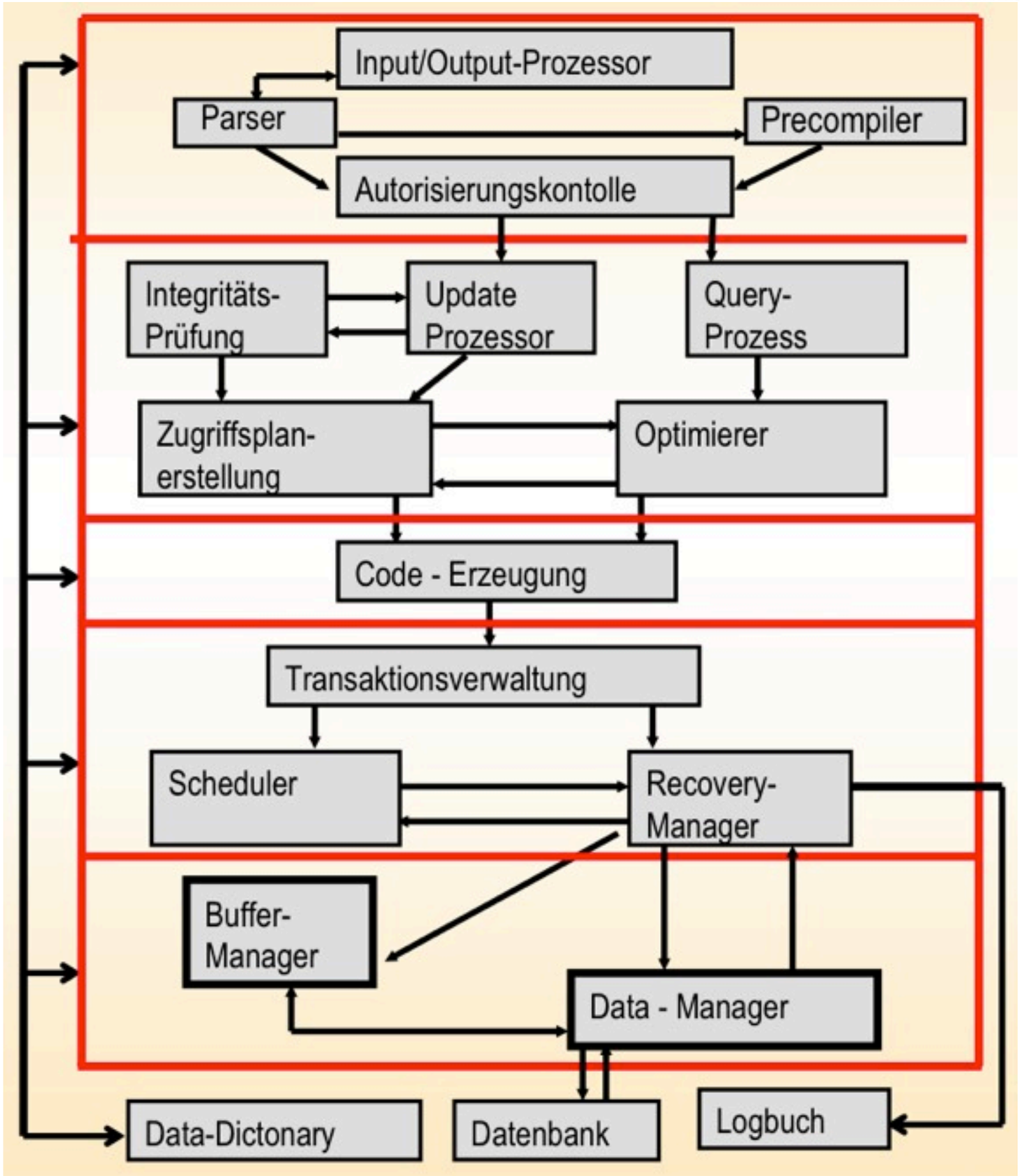
e) E = $\pi_{A\,B}(\sigma_{A=5}(R_1 \bowtie R_2)$

E' = $\sigma_{A=5}(R_1)$

$aw_E(d) = \pi_{A\,B}(\sigma_{A=5}(r_1 \bowtie r_2)) = \pi_{A\,B}(\sigma_{A=5}(r_1 * r_2))$

$= \sigma_{A=5}(\pi_{A\,B}(r_1 * r_2)) = \sigma_{A=5}(r_1) = aw_{E'}(d)$

6.4 Komponenten eines DBMS



Im Hauptspeicher: das DBMS

Auf den Sekundärspeichern: drei Datenbestände:

- Datenbank
- Schemainformationen im Data Dictionary
- (internes) Logbuch der Datenbank

Dem Benutzer zugeordnet: **I/O-Prozessor:**

- nimmt Kommandos entgegen,
- liefert Antworten oder Fehlermeldungen,
- ist häufig ein Monitor, ruft andere Komponenten auf bzw. stößt Prozesse an.

Parser :

syntaktische Analyse

Precompiler

Muss bei eingebetteten Kommandos aufgerufen werden.

Beide Kommandoarten erfordern die Ausführung einer **Autorisierungskontrolle.**

Ergebnis: Benutzerauftrag in einer „internen Form“

Query - Prozessor:

Übersetzt eine **Anfrage** in das konzeptionelle Schema.

Optimierer:

Erzeugt eine äquivalente Anfrage, die kostengünstiger sein soll.

Update - Operation:

an **Integritätsbedingungen** gebunden

Die interne Zwischenform wird entsprechend der **Integritätsbedingungen** erweitert.

Zugriffsplanerstellung:

Feststellung der auf die auszuwählenden Daten verfügbaren **Zugriffsstrukturen** und **Auswahl eines effizienten Zugriffspfades.**

Erstellung eines Zugriffs- bzw. Ausführungsprogramms:
Code - Generierung für den Benutzerauftrag

Transaktionsmanager:
Datenbank erscheint für jeden Nutzer als ein exklusiv verfügbares Betriebsmittel.

Scheduler :
Verzahnt und synchronisiert Transaktionen miteinander.

Transaktions - Manager :
Überwacht die Einhaltung der ACID – Eigenschaften für jede Transaktion.

Recovery – Manager:

- aktiv, wenn ein Hard- oder Softwarefehler auftritt.
- setzt die Datenbank in einen konsistenten Ausgangszustand zurück.
- ist für den **Wideranlauf** der Datenbank zuständig
- übernimmt Transaktionen, die nicht erfolgreich beendet werden können.
- verwendet das **Logbuch** der Datenbank, indem alle Veränderungen aufgezeichnet sind.
- setzt die Datenbank in den Zustand zurück, der vor Start der Transaktion bestand.
- macht alle bereits durchgeführten Veränderungen in der Datenbank rückgängig .

6.5 DBS-BETRIEB