

# Einführung in die Datenbanken

Felix Leitl

16. Juli 2024

## Inhaltsverzeichnis

<b>Grundlagen</b>	<b>4</b>
Modellierung . . . . .	4
Warum Datenbanken . . . . .	4
Vorteile einer Datenbank . . . . .	4
Nachteile . . . . .	4
Begriffe . . . . .	5
Datenbank . . . . .	5
Datenbank-Management-System . . . . .	5
Datenbanksystem . . . . .	5
Datenbankanwendung . . . . .	5
Datenmodell . . . . .	5
Datenbankschema . . . . .	5
Nutzdaten . . . . .	5
Metadaten . . . . .	5
Konzeptionelles Schema . . . . .	5
Externes Schema . . . . .	6
Internes Schema . . . . .	6
Phasen des Datenbankentwurfs . . . . .	6
<b>ERM</b>	<b>6</b>
<b>Relationenmodell</b>	<b>6</b>
Bestandteile eines Datenmodells . . . . .	6
Begriffe . . . . .	6
Erweiterte Attributdefinition . . . . .	7
Sicherstellung der Referenziellen Integrität . . . . .	7
Löschen eines referenzierten Primärschlüssels . . . . .	7
Ändern eines referenzierten Primärschlüssels . . . . .	7
Integritätsbedingungen . . . . .	7
„System-enforced Integrity“ . . . . .	7
Benutzerdefinierte oder „globale“ Integritätsbedingung . . . . .	8

<b>Mapping</b>	<b>8</b>
Abbildungskonzepte . . . . .	8
Algorithmus . . . . .	8
Reguläre Entity-Typen . . . . .	8
Schwache Entity-Typen . . . . .	8
M:N-Beziehungen . . . . .	8
N:1-Beziehungen . . . . .	9
1:1-Beziehungen . . . . .	9
Mehrwertige Attribute . . . . .	9
Mehrstellige Beziehungen . . . . .	9
Generalisierung/Spezialisierung . . . . .	9
Kategorien . . . . .	9
<b>Normalisierung</b>	<b>9</b>
Anomalien . . . . .	9
Funktionale Abhängigkeit $X \rightarrow Y$ . . . . .	10
Volle Funktionale Abhängigkeit . . . . .	10
Normalformen . . . . .	10
Erst Normalform (1NF) . . . . .	10
Zweite Normalform (2NF) . . . . .	10
Dritte Normalform (3NF) . . . . .	10
Boyce-Codd-Normalform (BCNF) . . . . .	10
Vierte Normalform (4NF) . . . . .	10
Denormalisierung . . . . .	10
Wann ist eine Denormalisierung angebracht? . . . . .	10
<b>Relationenalgebra</b>	<b>11</b>
<b>SQL</b>	<b>11</b>
Grundstruktur . . . . .	11
Neue Spalten . . . . .	11
Duplikate . . . . .	11
IN . . . . .	11
EXISTS . . . . .	11
Mengenvergleiche und Quantoren . . . . .	12
Join . . . . .	12
FROM-Klausel . . . . .	12
Auto-Join und Alias-Namen . . . . .	12
<b>Multidimensionale Datenmodellierung</b>	<b>12</b>
OLTP vs. OLAP . . . . .	12
OLTP . . . . .	12
OLAP . . . . .	12
Relationenmodell vs. Multidimensionales Datenmodell . . . . .	12
Relationenmodell . . . . .	12
Multidimensionales Datenmodell . . . . .	13
Charakterisierung der Datenanalyse . . . . .	13
Mikro-, Makro- und Meta-Daten . . . . .	13
Anforderungen . . . . .	13
MD Entwurf . . . . .	14

Logisches Schema einer Dimension . . . . .	14
Instanz einer Dimension . . . . .	14
Schema eines Datenwürfels . . . . .	14
Instanz eines Würfels . . . . .	15
Multidimensionale Operatoren . . . . .	15
Aggregation . . . . .	15
MD Schemaentwurf (Kimball) . . . . .	16
ROLAP . . . . .	16
Star Schema . . . . .	16
Snowflake Schema . . . . .	16
<b>Schichtenmodell</b> . . . . .	<b>16</b>
Konsequenzen . . . . .	16
Schichtenmodell eines DBVS . . . . .	17
Erweitertes Schichtenmodell . . . . .	19
Satzadresse . . . . .	19
<b>Transaktionen</b> . . . . .	<b>20</b>
Erwünschte Zustände auf einer Platte . . . . .	20
Physische Konsistenz . . . . .	20
Logische Konsistenz . . . . .	20
Annahmen . . . . .	20
Nach einem Fehler . . . . .	20
Systemunterstützung . . . . .	20
Der herzustellende konsistente Zustand kann sein . . . . .	20
Voraussetzungen . . . . .	20
<b>Pufferverwaltung</b> . . . . .	<b>21</b>

# Grundlagen

## Modellierung

Ein Modell ist ein zweckgerichtetes Abbild der Wirklichkeit

Zweck:

- Spezifizieren
- Konstruieren
- Visualisieren
- Dokumentieren

## Warum Datenbanken

- Große Software-Systeme
- Viele Anwendungen/Benutzer arbeiten mit den gleichen Daten
- Daten sollen auch nach Ende eines Programms verfügbar bleiben
- Daten sollen vor Verlust geschützt werden
- Daten sollen konsistent bleiben

## Vorteile einer Datenbank

- Anwendungsneutralität
- Vermeidung redundanter Daten
- Zentrale Kontrolle der Datenintegrität
- Synchronisation im Mehrnutzerbetrieb
- Fehlertoleranz
- Performance
- Skalierbarkeit
- Verkürzte Entwicklungszeiten für Anwendungen
- Umsetzung von Standards

## Nachteile

- Hohe initiale Kosten
- General purpose software
- Signifikanter Overhead

## **Begriffe**

### **Datenbank**

Eine Datenbank ist eine Sammlung zusammenhängender Daten.

- repräsentiert einen Ausschnitt der realen Welt (Miniwelt)
- Logisch kohärente Sammlung von Daten
- Hat definierten Zweck

### **Datenbank-Management-System**

Sammlung von Programmen zur Verwaltung einer Datenbank

- Erzeugung von DB
- Wartung von DB
- Konsistenter Zugriff auf DB

### **Datenbanksystem**

- DB + DBMS

### **Datenbankanwendung**

- DBS + Anwendungsprogramme

### **Datenmodell**

- Strukturierungsvorschrift für Daten (z.B. Tabellenform)

### **Datenbankschema**

- Beschreibung einer konkreten Datenbank

### **Nutzdaten**

- Eigentliche Datenbank

### **Metadaten**

- Struktur der DB
- Information über Speicherungsstrukturen

### **Konzeptionelles Schema**

- Beschreibt sämtliche Daten auf logischer Ebene
- z.B. Patient (NR. Krankenkasse, Laborwerte)

## Externes Schema

- Beschreibt den für die Anwendung relevanten Teil einer DB auf logischer Ebene
- z.B. für den Arzt: `Patient (Nr., Laborwerte)` und für die Verwaltung: `Patient (Nr., Krankenkasse)`

## Internes Schema

- Beschreibt die interne Speicherungsstrukturen einer Datenbank
- Unsichtbar für Anwendung
- z.B. Index über Attribut `Nr.` von `Patient`

## Phasen des Datenbankentwurfs

- Konzeptioneller Entwurf
  - Abbildung auf Semantisches Datenmodell (z.B. E/R-Modell)
- Logischer Entwurf
  - Abbildung auf Datenmodell

## ERM

Siehe Vorlesungsfolien

## Relationenmodell

### Bestandteile eines Datenmodells

- einfache Datentypen und Konstruktoren für zusammengesetzte Datentypen
- Konsistenzregeln:
  - inhärente Konsistenzregeln:  
gelten für ein Datenmodell per Konvention
  - explizite Konsistenzregeln:  
werden für eine Anwendung im Zuge der Datendefinition festgelegt
- Benennungskonvention für die Bezeichnung von Datenbankelementen

### Begriffe

- Relation: Menge von gleichartig aufgebauten Tupeln
- Tupel: Zeile einer Tabelle
- Kardinalität: Anzahl der Tupel in einer Relation
- Attribut: Spalte einer Tabelle

- Grad: Anzahl der Attribute
- Relationenschema:
  - Beschreibung einer Relation
  - besteht aus Relationennamen (z.B. **Personen**)
  - und einer Menge von Attributen (z.B. {PNr, Vorname, Nachname})
  - Jedes Attribut wird definiert über einen Attributnamen und einen Wertebereich
  - z.B. **Personen** (PRn, Vorname, Nachname)
- Relationales Datenbankschema: Menge von Relationendatenbankschemata
- Wertebereich: zulässige Attribute
- Superschlüssel: definiert ein Tupel eindeutig
- Schlüsselkandidat: Minimaler Superschlüssel
- Primärschlüssel: Ausgewählter Schlüsselkandidat
- Fremdschlüssel: Attribut, dass mit Primärschlüssel einer Tabelle auf ein bestimmtes Tupel verweist

### **Erweiterte Attributdefinition**

- NOT NULL
- UNIQUE
- PRIMARY KEY

### **Sicherstellung der Referenziellen Integrität**

#### **Löschen eines referenzierten Primärschlüssels**

- RESTRICTED: ablehnen der Operation
- CASCADES: Alle referenzierenden Tupel werden auch gelöscht
- NULLIFIE: Referenzen werden auf NULL gesetzt
- SET DEFAULT

#### **Ändern eines referenzierten Primärschlüssels**

- RESTRICTED
- CASCADES

### **Integritätsbedingungen**

#### **„System-enforced Integrity“**

- Primärschlüsseleigenschaft
- Referenzielle Integrität

### Benutzerdefinierte oder „globale“ Integritätsbedingung

- Bedingungen aus der Anwendungsdomäne, die explizit formuliert werden müssen
- Kontrolliert durch das DBMS
- Operationen, die die Integritätsbedingungen verletzen werden abgelehnt

## Mapping

### Abbildungskonzepte

ER-Modell	Relationenmodell
Entity-Typ	„Entity“-Relation
1:1- oder 1:N-Beziehungstyp	Fremdschlüssel oder
M:N-Beziehungstyp	Beziehungstabelle mit 2 FS
N-ärer Beziehungstyp	Beziehungstabelle mit N FS
Einfaches Attribut	Attribut
Zusammengesetztes Attribut	Menge von Attributen
Mehrwertiges Attribut	„Attribut“-Relation mit FS
Wertebereich	Wertebereich
Schlüsselattribut	Schlüsselkandidat → Primärschlüssel

### Algorithmus

#### Reguläre Entity-Typen

- Erzeuge eine Relation R, die alle einfachen Attribute von E umfasst
  - Bei zusammengesetzten Attributen nur Komponenten als eigenständige Attribute
- Wähle aus Schlüsselkandidaten einen Primärschlüssel
  - zusammengesetzt → Komponenten bilden zusammen den Primärschlüssel
  - Jeder Schlüsselkandidat, außer PS wird UNIQUE & NOT NULL

#### Schwache Entity-Typen

- Erzeuge eine Relation, die alle einfachen Attribute von W umfasst
- Füge als Fremdschlüssel alle PS-Attribute der Owner-Typen ein
- PS wird Kombination aller FSA, zusammen mit partiellem Schlüssel (falls vorhanden)

#### M:N-Beziehungen

- Erzeuge Relation die alle einfachen Attribute von X umfasst
- FS → PSA der beidem Relationen
- PS ist Kombination der FSA



### **N:1-Beziehungen**

- identifiziere die Relation, die dem Entity-Typ E auf der N-Seite des Beziehungstyps entspricht
- Füge den PS des anderen ET als FS in R ein
- Füge alle einfachen Attribute des Beziehungstyps X als Attribute in R ein

### **1:1-Beziehungen**

- Identifiziere Relationen R & S
- Nehme den PS von S bzw. R als FS von R bzw. S auf UNIQUE
- Füge alle einfachen Attribute in R bzw. S ein

### **Mehrwertige Attribute**

- Erzeuge Relation R mit folgenden Attributen:
  - Ein Attribut A, dass dem abzubildenden Attribut A entspricht
  - Den PS K der Relation S, die zu E gehört, als FS auf S
- Der PS der Relation R ist die Kombination von A & K

### **Mehrstellige Beziehungen**

- Erzeuge Relation R, die alle einfachen Attribute von B umfasst
- $FS \rightarrow PS$  aller Relationen
- $PS \rightarrow$  Kombination aller FS

### **Generalisierung/Spezialisierung**

siehe VL

### **Kategorien**

siehe VL

## **Normalisierung**

### **Anomalien**

- Einfüge-Anomalie (ohne hinzufügen von Info B, geht Info A nicht)
- Lösch-Anomalie
- Änderungs-Anomaile

## **Funktionale Abhängigkeit $X \rightarrow Y$**

$Y$  ist funktional abhängig von  $X$ , wenn es keine Tupel geben darf, in denen für gleiche  $X$ -Werte verschiedene  $Y$ -Werte auftreten

Linke Seite der FA wird „Determinante“ genannt

## **Volle Funktionale Abhängigkeit**

$Y$  ist voll funktional abhängig von  $X$ , wenn es keine echte Teilmenge  $Z \subset X$  gibt, für die gilt  $Z \rightarrow Y$

## **Normalformen**

### **Erst Normalform (1NF)**

Eine Relation, die nur atomare Attributwerte besitzt (keine Mengen als Attributwert)

### **Zweite Normalform (2NF)**

Eine Relation, in 1NF & deren Nicht-Schlüsselattribute voll funktional von jedem Schlüsselkandidaten abhängen

### **Dritte Normalform (3NF)**

Eine Relation, deren Nicht-Schlüsselkandidaten nicht transitiv abhängig von einem Schlüsselkandidaten sind

### **Boyce-Codd-Normalform (BCNF)**

Eine Relation, bei welcher jede Determinante einer FA ein Superschlüssel ist

### **Vierte Normalform (4NF)**

Eine Relation  $R$  ist in 4NF, wenn für jede nicht-triviale mehrwertige Abhängigkeit  $X \twoheadrightarrow A \in R$  gilt:  $X$  ist Superschlüssel von  $R$

Eine mehrwerte Abhängigkeit gilt, wenn die Attributwerte von  $C$  nur von  $A$  und nicht von  $B$  abhängig sind

$A \twoheadrightarrow C$  ist trivial, wenn  $C \in A$  oder  $B = \emptyset$

## **Denormalisierung**

Normalisierung kostet Zugriffszeit

### **Wann ist eine Denormalisierung angebracht?**

- Seltene Änderungen
- Viele Joins

Bei weiteren Fragen Anhang VL\_06 lesen

# Relationenalgebra

## SQL

### Grundstruktur

```
SELECT Personalnummer, Name
FROM Mitarbeiter
WHERE Name = 'Müller';
```

### Neue Spalten

```
SELECT MNR, Gehalt * 1.1 AS Gehaltsprognose
FROM Angestellte;
```

```
SELECT MNR, Gehalt + Werbeeinnahmen AS Einkünfte
FROM Angestellte;
```

```
(
    SELECT Name, Vorname, Gehalt
    FROM Angestellte)
UNION
(
    SELECT Name, Vorname, NULL AS Gehalt
    FROM Kunde);
```

### Duplikate

```
SELECT DISTINCT Wohnort
FROM Angestellte;
```

### IN

```
SELECT *
FROM Angestellte
WHERE AbtNr IN (6, 4, 2);
```

```
SELECT Nachname
FROM Angestellte
WHERE AbtNr IN
(
    SELECT AbtNr
    FROM Abteilungen
    WHERE Ort = 'Erlangen'
);
```

### EXISTS

```
SELECT *
FROM Angestellte
WHERE EXISTS (SELECT * FROM Abteilungen WHERE Ort = 'Erlangen');
```

Inner Anfrage muss Bezug zur äußeren Anfrage haben

## Mengenvergleiche und Quantoren

```
SELECT *  
FROM Angestellte  
WHERE Wohnort = ANY (SELECT Ort FROM Abteilungen);
```

```
SELECT *  
FROM Angestellte  
WHERE Gehalt >= ALL (SELECT Gehalt FROM Angestellte);
```

## Join

### FROM-Klausel

```
SELECT PersNr, Wohnort, Bezeichnung  
FROM Angestellte, Abteilung  
WHERE Angestellte.AbtNr = Abteilung.AbtNr  
      AND Gehalt > 30000  
      AND Ort = 'Nürnberg';
```

### Auto-Join und Alias-Namen

```
SELECT m.Nachname AS Mitarbeiter, v. Nachname AS Chef  
FROM Angestellte [AS] m, Angestellte [AS] v  
WHERE m.VorgesNr = v.PersNr  
AND m.Gehalt > v.gehalt;
```

## Multidimensionale Datenmodellierung

### OLTP vs. OLAP

#### OLTP

Online Transaction Processing

#### OLAP

Online Analytical Processing

### Relationenmodell vs. Multidimensionales Datenmodell

#### Relationenmodell

- Einfach, wenige Modellierungskonstrukte
- Anwendungsneutral
- Keine „eingebaute“ Anwendungssemantik

⇒ Nützlich in beliebigen Domänen, manchmal etwas komplizierter in der Anwendung

## **Multidimensionales Datenmodell**

- Komplexer, mehr Modellierungskonstrukte
- Speziell auf Anwendung zur Datenanalyse zugeschnitten

⇒ Nur nützlich für analytische Zwecke

## **Charakterisierung der Datenanalyse**

- Qualifizierende und Quantifizierende Daten
  - Spezielle funktionale Abhängigkeiten ⇒ spezifische Repräsentation
- Klassifikationshierarchien
  - Aggregierende Anfragen nutzen Hierarchien zu ihrem Vorteil
- Stabile Daten
  - Daten werden (fast) nie geändert
  - Nur neue Daten hinzugefügt
- Zugriff auf materialisierte Sichten
  - Voraggregierte Daten

## **Mikro-, Makro- und Meta-Daten**

- Mikro-Daten
  - Einzelne Observationen, beschreiben Elementarereignisse
  - Ergebnis der Ladephase → Basisdaten
- Makro-Daten
  - Aggregierte Daten für die Datenanalyse
  - Ergebnis der Auswertungsphase → Data Warehouse, Data Mart
- Meta-Daten
  - Beschreibungsdaten
  - Beschreiben die Eigenschaften von Mikro-Daten und Makro-Daten
  - Beschreiben auch den Entstehungsprozess

## **Anforderungen**

- Datenwürfel soll flexibel durchsucht werden können
- Qualifizierende Daten → Dimensionen des Würfels
- Quantifizierende Daten → Fakten (Zellen des Würfels)
- Dimensionen müssen unabhängig sein
- Eindeutige Trennung von Fakten

## MD Entwurf

1. Benutzer-Anforderungen
2. Konzeptionelle Schema
  - semi-formal: mE/R, mUML
3. Logisches Schema
  - formal: Dimensionen, Cubes
4. Physisches Schema
  - Relationen, MD-Strukturen

## Logisches Schema einer Dimension

- Partiiell geordnete Menge  $D$  von Klassifikationsstufen
- Partielle Ordnung erlaubt parallele Hierarchie: „Pfade“
- Orthogonalität: Verschiedene Dimensionen sind unabhängig

## Instanz einer Dimension

- Funktionale Abhängigkeiten  $\rightarrow$  Baumstruktur auf Instanzebene
- Jeder Pfad im Schema einer Dimension definiert eine Klassenhierarchie
- Klassenhierarchie ist ein balancierter Baum
- Instanz einer Dimension ist die Menge aller Klassenhierarchien

## Schema eines Datenwürfels

- Definition: Schema eines Datenwürfels  $C$ 
  - Struktur:  $C[G, M]$
  - Menge von Fakten:  $M = (M_1, \dots, M_m)$
  - Granularität:  $G = (G_1, \dots, G_n)$
  - Jedes  $G_i$  ist ein dimensionales Attribut
- Bsp.:
  - Sales und Turnover pro Article, Shop und Day
    - \*  $C_{\text{sales}}[(P.\text{Article}, S.\text{Shop}, T.\text{Day}), (\text{Sales}, \text{Turnover})]$
- Fakten (Kenngrößen)
  - können auch Eigenschaften zugesprochen werden
  - sind aber keine Datenstruktur an sich, eher analog zu einem Wertebereich
- Aggregatstyp

- nicht-triviale Eigenschaften neben Name und Wertebereich → definiert, welche Aggregationsoperationen auf einer Kenngröße ausgeführt werden dürfen
  1. beliebig aggregierbar (Sales, Turnover, ...): FLOW
  2. nicht temporal summierbar (Stock, Inventory, ...): STOCK
  3. nicht summierbar (Preis, Steuer, i.Allg. Faktoren): VPU (Value per Unit)
- MIN, MAX & AVG können immer durchgeführt werden

## **Instanz eines Würfels**

- Alle Zellen aus dem Definitionsbereich des Datenwürfels werden als existierend angenommen, egal ob ein Datensatz physisch vorhanden ist
- Gegenüberstellung: Im relationalen Datenmodell herrscht eine andere Grundannahme vor (→ „closed world“-Prinzip): Nichts wird angenommen, was nicht explizit als Datensatz vorhanden ist (→ „Intension vs. Extension“)

## **Multidimensionale Operatoren**

- Slice
- Dice
- Drill-Down: Abstieg in der Klassifikationshierarchie zu feinerem Granulat
- Roll-Up: Aufstieg in der Klassifikationshierarchie hin zu größerem Granulat
- Drill-Across: Verknüpfung mehrerer Datenwürfel mit gemeinsamen Dimensionen
- Drill-Through: Wechsel zu den Originaldaten
- Pivottisierung: Wechsel der Darstellung in einer Pivottabelle, entspricht Drehen des Würfels

## **Aggregation**

- Zusammenfassen mehrerer Zellen
- Notwendig beim Roll-Up
- Bsp.: vom Tag zum Monat, vom Produkt zur Kategorie
- Standard
  - SUM
  - AVG
  - MIN
  - MAX
  - COUNT
- Ordnungsbasiert
  - cumulating
  - ranking

## MD Schemaentwurf (Kimball)

1. Auswahl eines Geschäftsprozesses → Subjektorientierung
2. Auswahl der Erfassungsgranularität
3. Auswahl der Dimensionen
4. Auswahl der Kennziffern

## ROLAP

- Idee für die Speicherung multidimensionaler Daten
  - Tabelle mit zusammengesetztem Primärschlüssel aus den Dimensionen
  - (Nur) für jede vorhandene Datenzeile wird ein Tupel abgespeichert
- Trennung von Struktur und Inhalt führt zur Aufteilung in
  - zentrale „Fact Table“ und
  - Dimensionstabellen

## Star Schema

- Eine Tabelle für jede Dimension

## Snowflake Schema

- Normalisierung der Dimensionstabellen
- Viele Tabellen je Dimension

## Schichtenmodell

### Konsequenzen

- Höhere Ebenen werden einfacher
- Änderungen auf höheren Ebenen haben keinen Einfluss auf niedrigere
- Änderungen sind unproblematisch, solange die Schnittstelle gleich bleibt
- Ebenen sind wiederverwendbar
- Tiefere Ebenen können vor höheren getestet werden

⇒

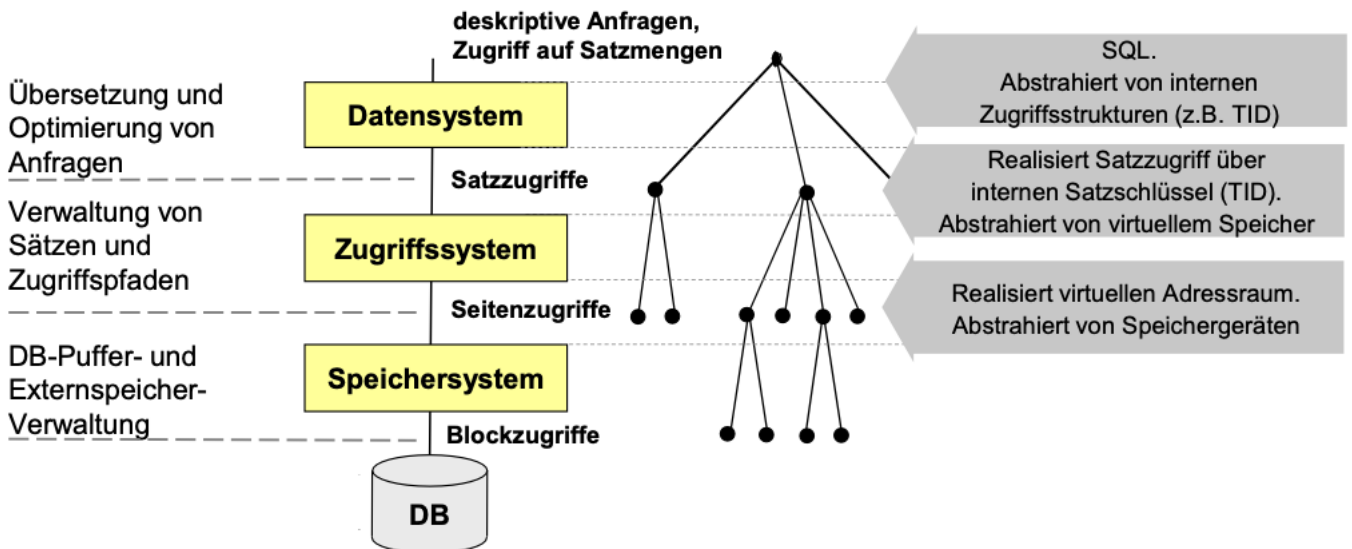
- Reduzierte Komplexität
- Verbesserung der Wartbarkeit
- Verbesserung der Portierbarkeit
- Verbesserung der Wiederverwendbarkeit



## Schichtenmodell eines DBVS

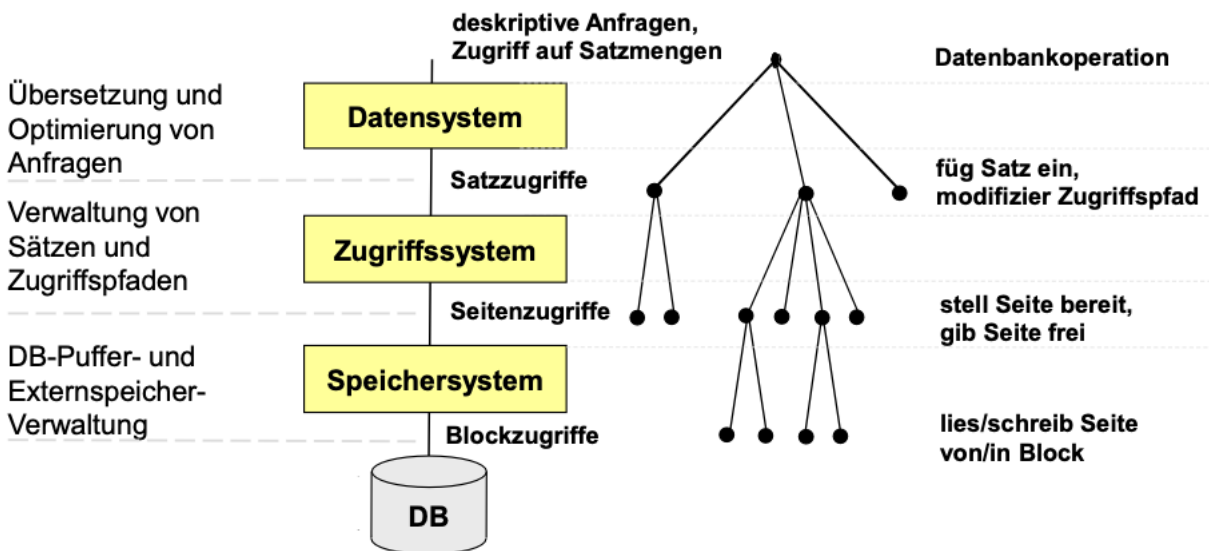
### Aufgaben der Schicht

### Abstraktionen der Schnittstelle

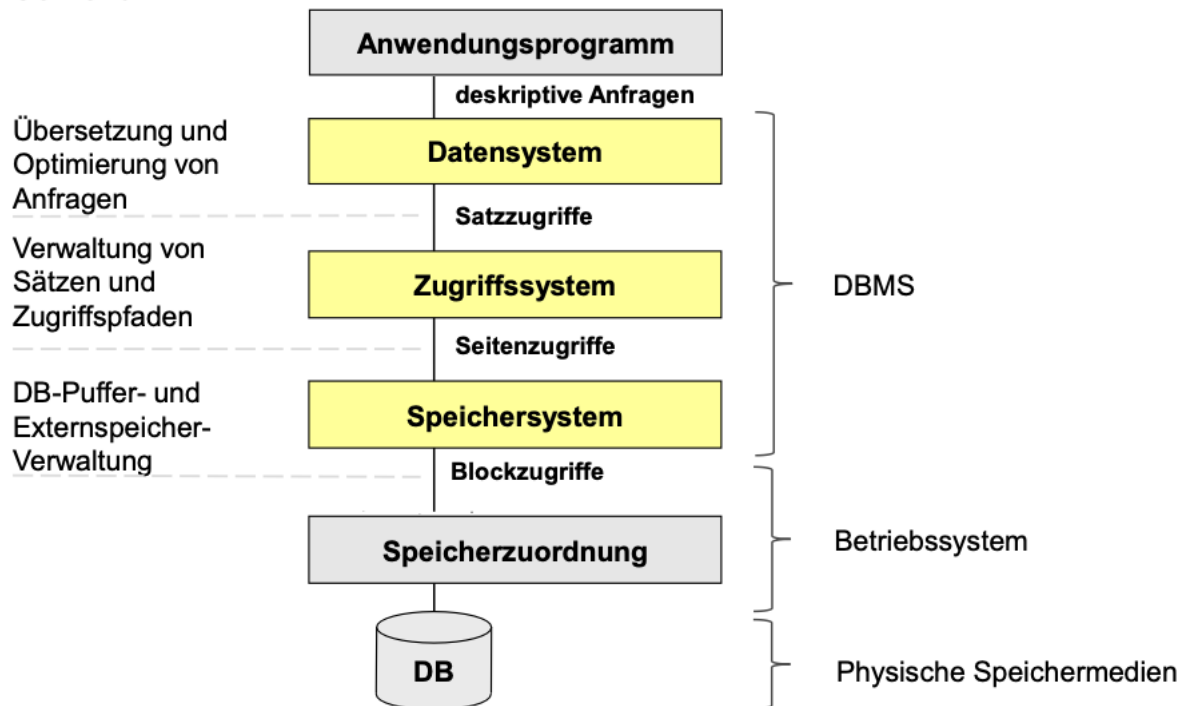


### Aufgaben der Schicht

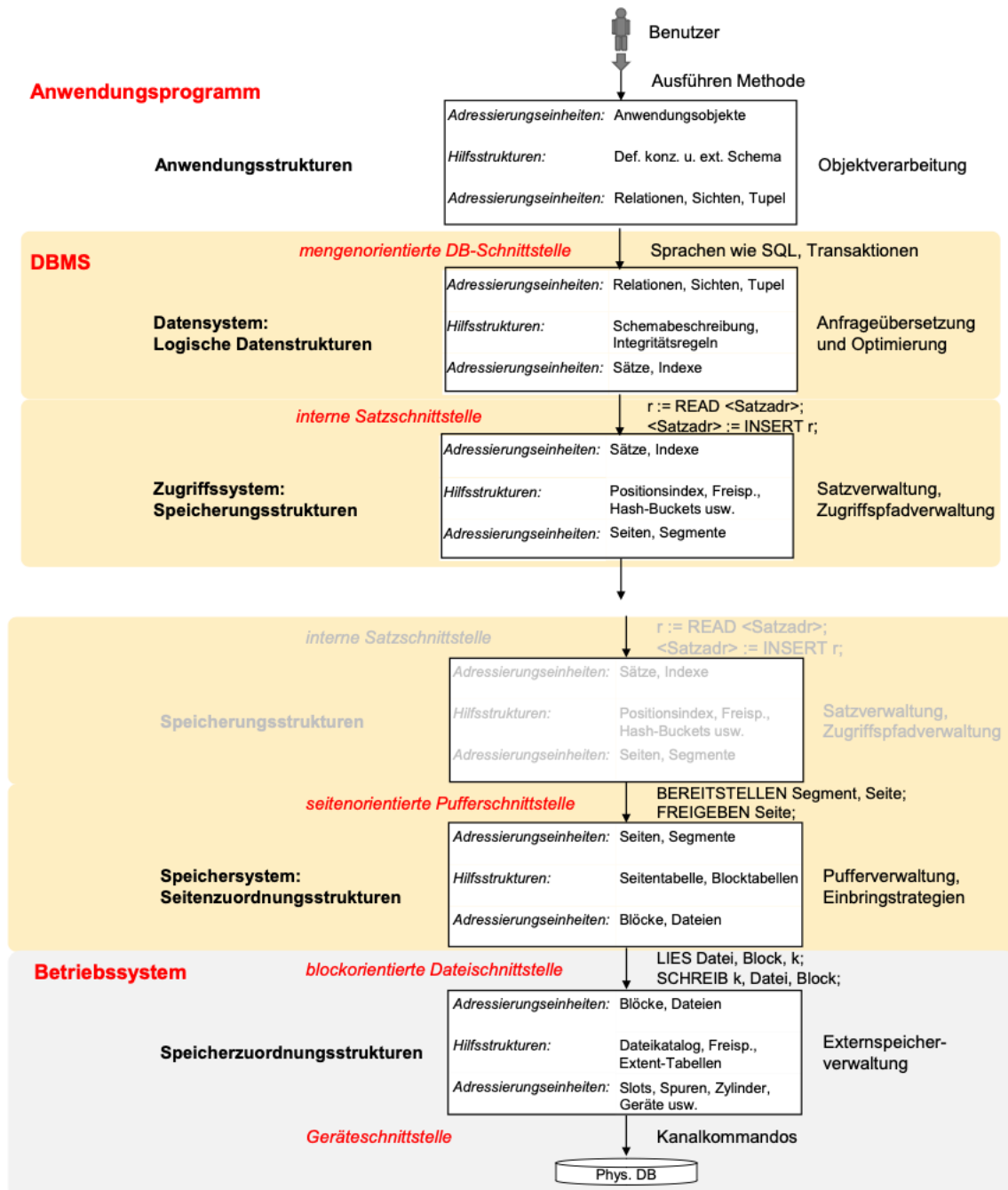
### Operationen an der Schnittstelle



## Aufgaben der Schicht



## Erweitertes Schichtenmodell



## Satzadresse

Eine Satzadresse sollte

- eindeutig

- unveränderlich
- Paar aus Seitennummer und Feldindex

sein

## **Transaktionen**

### **Erwünschte Zustände auf einer Platte**

#### **Physische Konsistenz**

- Korrektheit der Speicherkonstrukte
- Alle Verweise und Adressen (TIDs) stimmen
- Alle Indexe sind vollständig und stimmen mit Primärdaten überein

#### **Logische Konsistenz**

- Korrektheit der Dateninhalte
- Alle im Datenbankschema formulierten Integritätsbedingungen sind erfüllt

#### **Annahmen**

- Alle vollständig ausgeführten DB-Operationen hinterlassen einen physisch konsistenten Zustand
- Alle vollständig ausgeführten Anwendungsprogramme hinterlassen einen logisch konsistenten Zustand

#### **Nach einem Fehler**

- Die Daten sind i.Allg. weder physisch noch logisch konsistent

#### **Systemunterstützung**

- zur Wiederherstellung eines logischen und physischen konsistenten Zustands der Daten nach einem Fehlerfall

#### **Der herzustellende konsistente Zustand kann sein**

- Vor Beginn der Änderungen eines unvollständig ausgeführten Programms
  - Rückgängigmachen der bereits ausgeführten Änderungen
- Nach Abschluss aller Änderungen eines Programms
  - Kompletieren der unvollständigen Änderungen bzw. Wiederholen verlorengegangener Änderungen

#### **Voraussetzungen**

- Geeignete Sicherungs- und Protokollierungsmaßnahmen im laufenden Betrieb (Log)
- Protokolldatei

## Pufferverwaltung