

# 1. Einführung / Grundlagen von DBS

- DBS vs. Dateisysteme
- Eigenschaften von DBS
- Datenmodelle
- Transaktionskonzept (ACID)
  
- Aufbau von DBS
  - Schemaarchitektur
  - Schichtenmodell
- Historische Entwicklung
  
- Einsatzformen von DBS (OLTP, Decision Support)

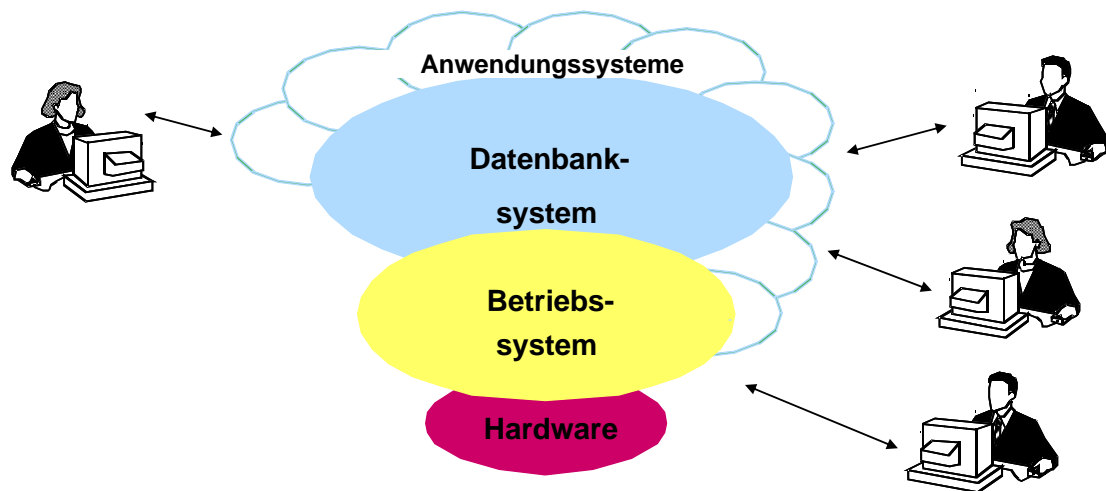


## Persistente Datenhaltung

- persistente Datenspeicherung: Dateien oder Datenbanken
  - Nutzung von Hintergrundspeicher (Magnetplattenspeicher)
  - Daten bleiben über Programmende, Rechnereinschaltung etc. hinaus erhalten
  - inhaltsbasierter Zugriff auf Daten vielfach erforderlich
  
- Daten sind wertvoll!
  - Adressdaten
  - Personaldaten
  - Kundendaten
  - Transaktionsdaten (Bestellungen, Lieferungen, ...)
  - Konstruktionsdaten (Auto, Motor, ...)
  - Geoinformationsdaten (Straßen, Flüsse, Leitungen, ...)



# DBS als Kern von Informationssystemen



- $IS = DBS + \text{Anwendungssysteme} + \text{Benutzerschnittstellen}$
- $DBS = DB + \text{Datenbankverwaltungssystem (DBVS, DBMS)}$ 
  - **DB:** Menge der gespeicherten Daten
  - **Datenbankverwaltungssystem (DBVS):** Generisches Software-System zur Definition, Verwaltung, Verarbeitung und Auswertung der DB-Daten. Es kann für unterschiedlichste Anwendungen eingesetzt werden.

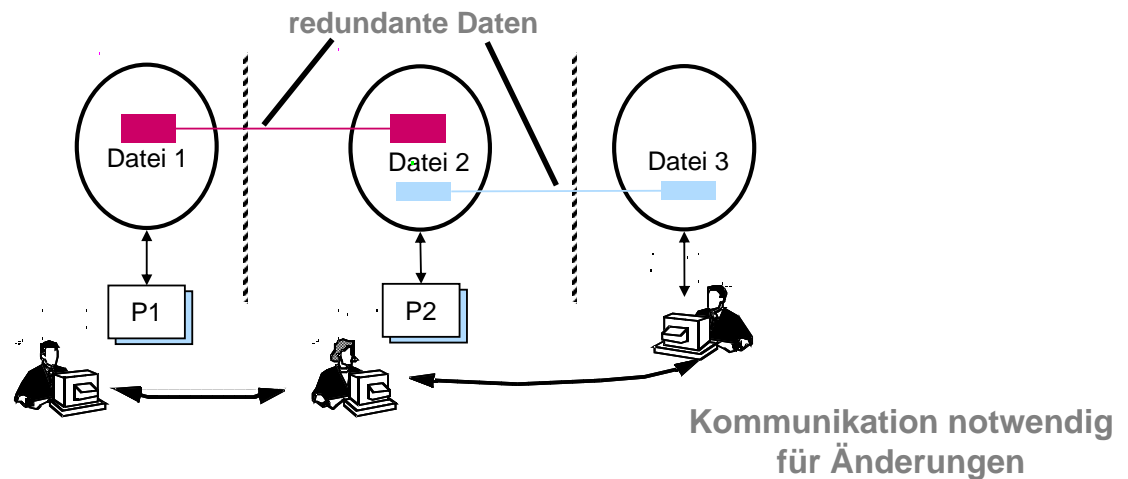


## Beispiele für Informationssysteme

- **Universitätsdatenbank**
  - Verwaltung von Fakultäten und ihren Studenten sowie Professoren
  - Studenten belegen Vorlesungen von Professoren und legen bei ihnen Prüfungen ab
  - Anwendungsvorgänge sind z. B.: Immatrikulation, Rückmeldung, Exmatrikulationen, Stundenplanerstellung, Planung der Raumbelegung, Ausstellen von Zeugnissen, Statistiken über Prüfungsergebnisse, etc.
- **Datenbank eines Produktionsbetriebes**
  - Verwaltung verschiedener Abteilungen und deren Beschäftigte
  - Die in einem Betrieb hergestellten Endprodukte setzen sich i.a. aus mehreren Baugruppen und Einzelteilen zusammen. Jedes Teil kann von Lieferanten bezogen werden.
  - Typische Anwendungsvorgänge sind z. B.: Personalverwaltung (Einstellung / Entlassung, Lohn- und Gehaltsabrechnung), Bestellung und Lieferung von Einzelteilen, Verkauf von Fertigprodukten, Lagerhaltung, Bedarfsplanung, Stücklistenauflösung



# Probleme mit Dateisystemen



- wiederholte Speicherung gleicher Daten => Redundanz
  - erhöhter Speicherplatzbedarf
  - Konsistenzprobleme !

## Probleme mit Dateisystemen (2)

- hoher Entwicklungsaufwand für Anwendungen
  - Programmierer verantwortlich für Aufbau/Inhalt der Dateien
  - Lösung gleicher Aufgaben in allen Anwendungsprogrammen: Suchaufgaben, Änderungsdienst, Speicherverwaltung ...
- enge Bindung von Datenstrukturen an Programmstrukturen (geringe „Datenunabhängigkeit“)
  - Kenntnisse der Datenorganisation kann gutes Leistungsverhalten ermöglichen, **aber**
  - Änderungen im Informationsbedarf sowie bei Leistungsanforderungen erfordern Anpassungen, die auf Anwendungen durchschlagen
  - verschiedene Anwendungen brauchen verschiedene Sichten auf dieselben Daten
- Mehrbenutzerbetrieb
- Verlust von Daten, Datensicherheit
  - Annahmen: Alles bleibt stabil ! Alles geht gut !

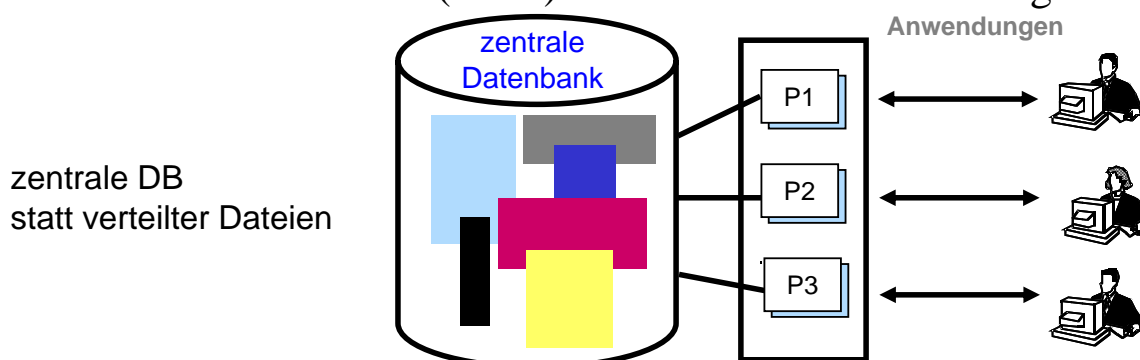
# Aufgaben/Eigenschaften von DBS

- Generell: effiziente und flexible Verwaltung großer Mengen persistenter Daten (z. B. GBytes - T Bytes)
1. Zentrale Kontrolle über die operationalen Daten
  2. Hoher Grad an Datenunabhängigkeit
  3. Hohe Leistung und Skalierbarkeit
  4. Mächtige Datenmodelle und Anfragesprachen / leichte Handhabbarkeit
  5. Transaktionskonzept (ACID), Datenkontrolle
  6. **Ständige Betriebsbereitschaft (hohe Verfügbarkeit und Fehlertoleranz)**
    - 24-Stundenbetrieb
    - keine Offline-Zeiten für DB-Reorganisation u. ä.



## 1. Zentrale Kontrolle der Daten

- Alle (operationalen) Daten können gemeinsam benutzt werden
  - keine verstreuten privaten Dateien
  - ermöglicht inhaltliche Querauswertungen
- Eliminierung der Redundanz
  - Vermeidung von Inkonsistenzen
  - keine unterschiedlichen Änderungsstände
- einfache Erweiterung/Anpassung der DB (Änderung des Informationsbedarfs)
- Datenbankadministrator (DBA) hat zentrale Verantwortung für Daten



## 2. Datenunabhängigkeit

- Datenunabhängigkeit = Maß für die Isolation zwischen Anwendungsprogrammen und Daten
- Gefordert ist eine möglichst starke Isolation der Anwendungsprogramme von den Daten
  - sonst: extremer Wartungsaufwand für die Anwendungsprogramme
- Minimalziel: **physische Datenunabhängigkeit**
  - Unabhängigkeit gegenüber Geräteeigenschaften, Speicherungsstrukturen, Indexstrukturen, ...
- **logische Datenunabhängigkeit**
  - Unabhängigkeit gegenüber logischer Strukturierung der Daten
  - i. a. nur teilweise erreichbar



## 3. Hohe Leistung und Skalierbarkeit

- Hoher Durchsatz / kurze Antwortzeiten für DB-Operationen auf großen Datenmengen
  - „trotz“ loser Bindung der Programme an die Daten (Datenunabhängigkeit)
- Leistungsverhalten ist DBS-Problem, nicht Anwendungsproblem
  - Zugriffsoptimierung für DB-Anfragen durch das DBS (Query-Optimierung)
  - Festlegung von Zugriffspfaden (Indexstrukturen), Datenallokation etc. durch den DBA (idealerweise durch das DBS)
  - automatische Nutzung von Mehrprozessorsystemen, parallelen Plattensystemen etc. (-> Parallele DBS)
- Hohe Skalierbarkeit
  - Nutzung zusätzlicher/schnellerer Hardware-Ressourcen
  - Anpassung an steigende Leistungsanforderungen (wachsende Datenmengen und Anzahl der Benutzer)



# 4. Mächtige Datenmodelle

## ■ Datenmodell/DBS-Schnittstelle

- Operationen zur Definition von Datenstrukturen (Data Definition Language, DDL), Festlegung eines **DB-Schemas**
- Definition von Integritätsbedingungen und Zugriffskontrollbedingungen (Datenschutz)
- Operationen zum Aufsuchen und Verändern von Daten (Data Manipulation Language DML)



## Datenstrukturierung

### ■ Beschreibung der logischen Aspekte der Daten, neutral gegenüber Anwendungen

- Anwendung erhält logische auf ihren Bedarf ausgerichtete Sicht auf die Daten

### ■ formatierte Datenstrukturen, feste Satzstruktur

- Beschreibung der Objekte durch Satztyp, Attribute und Attributwerte ( $S_i/A_j/AW_k$ )
- jeder Attributwert  $AW_k$  wird durch Beschreibungsinformation (Metadaten)  $A_j$  und  $S_i$  in seiner Bedeutung festgelegt

--	--	--	--



# Mächtige Anfragesprachen

- Art der Anfragesprache (query language)
  - formale Sprache
  - abhängig von Datenmodell: navigierend / satzorientiert vs. **deskriptiv / mengenorientiert**
  - einfache Verknüpfung mehrerer Satztypen („typübergreifende“ Operationen)
- Strukturierung ermöglicht Einschränkung des Suchraumes für Anfragen sowie effiziente Indexunterstützung
- Wünschenswert
  - deskriptive Problemformulierung, leichte Erlernbarkeit
  - hohe Auswahlmächtigkeit
  - DB-Zugriff im Dialog und von Programmen aus
  - Standardisierung (SQL)
- Nutzerklassen einer Anfragesprache: Systempersonal, Anwendungsprogrammierer, „anspruchsvolle Laien“



## Relationenmodell

### Beispiel: Universitäts-DB

*FAK*

<u>FNR</u>	FNAME	DEKAN
------------	-------	-------

*PROF*

<u>PNR</u>	PNAME	FNR	FACHGEB
------------	-------	-----	---------

*STUDENT*

<u>MATNR</u>	SNAME	FNR	W-ORT
--------------	-------	-----	-------

*PRÜFUNG*

<u>PNR</u>	<u>MATNR</u>	FACH	DATUM	NOTE
------------	--------------	------	-------	------



## Relationenmodell (2)

**FAK**

FNR	FNAME	DEKAN
MI	Mathematik/ Informatik	2223

**STUDENT**

MATNR	SNAME	FNR	W-ORT
654 711	ABEL	MI	Leipzig
196 481	MAIER	MI	Delitzsch
225 332	MÜLLER	MI	Leipzig

**PROF**

PNR	PNAME	FNR	FACHGEB
1234	RAHM	MI	DBS
2223	MEYER	MI	AN
6780	BREWKA	MI	KI

**PRÜFUNG**

PNR	MATNR	FACH	DATUM	NOTE
6780	654 711	FA	19.9.	2
1234	196 481	DBS	15.10.	1
1234	654 711	DBS	17.4.	2
6780	196 481	KI	25.3.	3



## Relationenmodell (3)

### ■ Beispielanfragen mit SQL

Finde alle Studenten der Fakultät MI mit Wohnort Leipzig:

```
SELECT *  
FROM STUDENT  
WHERE FNR = 'MI' AND W-ORT = 'Leipzig'
```

Finde alle Studenten der Fakultät MI, die im Fach DBS eine Note 2 oder besser erhielten:

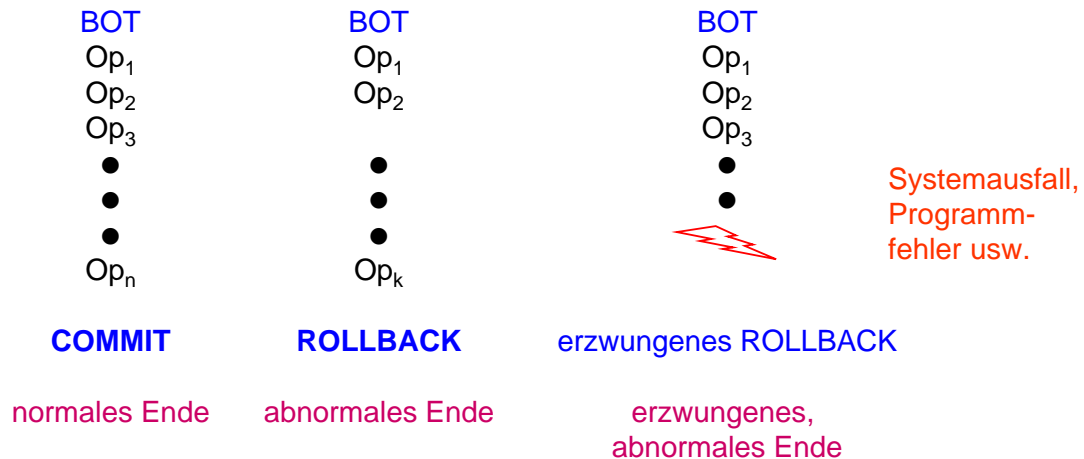
```
SELECT S.*  
FROM STUDENT S, PRUEFUNG P  
WHERE S.FNR = 'MI' AND P.FACH = 'DBS'  
AND P.NOTE <= 2 AND S.MATNR = P.MATNR
```





# 5. Transaktionskonzept

- **Kontrollstruktur:** Transaktionen mit den vier **ACID**-Eigenschaften
  - Eine Transaktion besteht aus einer Folge von DB-Operationen, für die das DBS folgende Eigenschaften garantiert
    - **A**tomicity: Alles-oder-Nichts
    - **C**onsistency: Gewährleistung der Integritätsbedingungen
    - **I**solated Execution: „logischer Einbenutzerbetrieb“
    - **D**urability: Persistenz aller Änderungen
- **Atomarität:** mögliche Ausgänge einer Transaktion

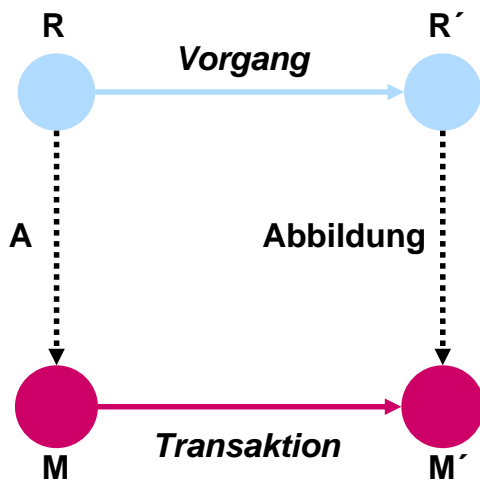


## Transaktionskonzept/Zugriffskontrolle

- **Consistency:** Erhaltung der logischen Datenintegrität
- **Erhaltung der physischen Datenintegrität**
  - Führen von Änderungsprotokollen für den Fehlerfall (*Logging*)
  - Bereitstellen von Wiederherstellungsalgorithmen im Fehlerfall (*Recovery*)
- **Kontrollierter Mehrbenutzerbetrieb (Ablaufintegrität)**
  - *logischer Einbenutzerbetrieb* für jeden von n parallelen Benutzern (Leser + Schreiber)
  - Synchronisation i. a. durch Sperren (*Locking*)
  - wichtig: Lese- und Schreibsperren mit angepassten Sperreinheiten (-granulate)
  - **Ziel:** möglichst geringe gegenseitige Behinderung
- **Automatisierte Zugriffskontrollen (Datenschutz)**
  - separat für jedes Datenobjekt
  - unterschiedliche Rechte für verschiedene Arten des Zugriff



# Modell einer Miniwelt: Grobe Zusammenhänge



R: Realitätsausschnitt (Miniwelt)

M: Modell der Miniwelt  
(beschrieben durch DB-Schema)

A: Abbildung aller wichtigen Objekte und  
Beziehungen (Entities und Relationships)  
=> Abstraktionsvorgang

## ■ Transaktion:

- garantiert ununterbrechbaren Übergang von M nach M'
- implementiert durch Folge von DB-Operationen

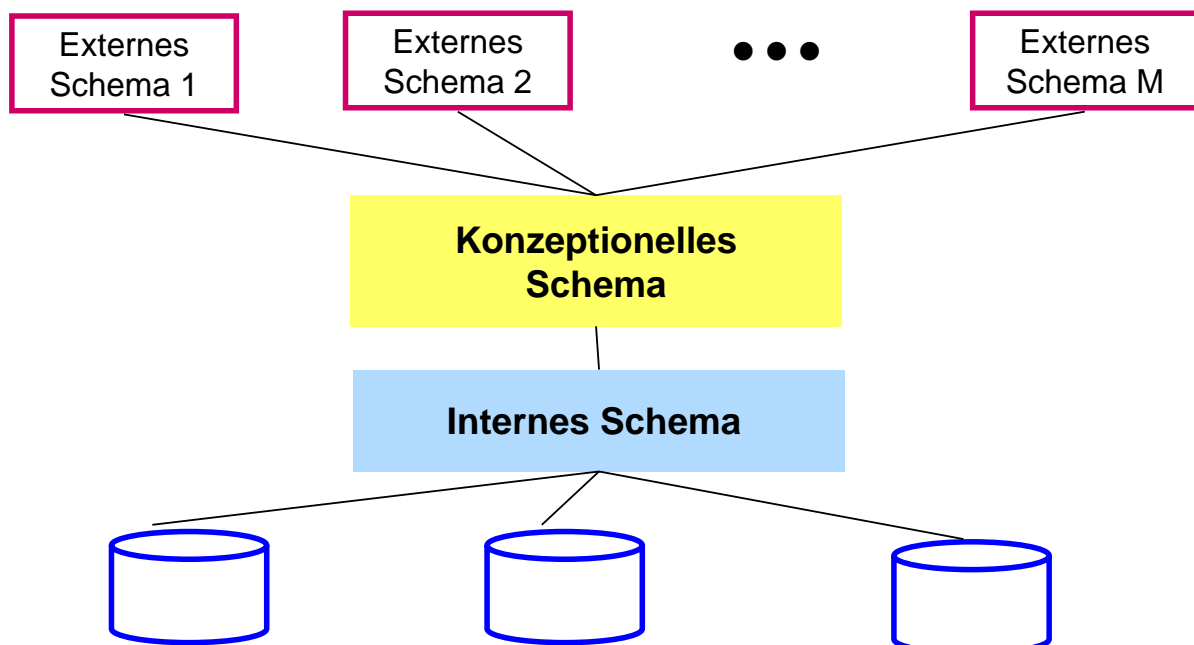
## ■ Integritätsbedingungen:

- Zusicherungen über A und M
- Ziel: möglichst gute Übereinstimmung von R und M



# Schemaarchitektur

## ■ 3-Ebenen-Architektur nach ANSI / SPARC



# Schemaarchitektur (2)

## ■ Konzeptionelles Schema:

- logische Gesamtsicht auf die Struktur der Datenbank
- abstrahiert von internem Schema  
-> *physische Datenunabhängigkeit*

## ■ Internes Schema

- legt physische Struktur der DB fest (physische Satzformate, Indexstrukturen etc.)

## ■ Externe Schemata

- definieren spezielle Benutzersichten auf DB-Struktur (für Anwendungsprogramm bzw. Endbenutzer)
- abstrahieren von konzeptionellem Schema: ermöglicht partiell *logische Datenunabhängigkeit*
- Sichtenbildung unterstützt Zugriffsschutz: Isolation von Attributen, Relationen, ...
- Reduktion der Komplexität: Anwendung sieht nur die erforderlichen Daten



# Beispiel-Datenbeschreibung (vereinfacht)

### Externe Sicht

```
MITARBEITER
  PNR   CHAR      (6)
  ABT   CHAR      (30) ...
```

### Konzeptionelles Schema:

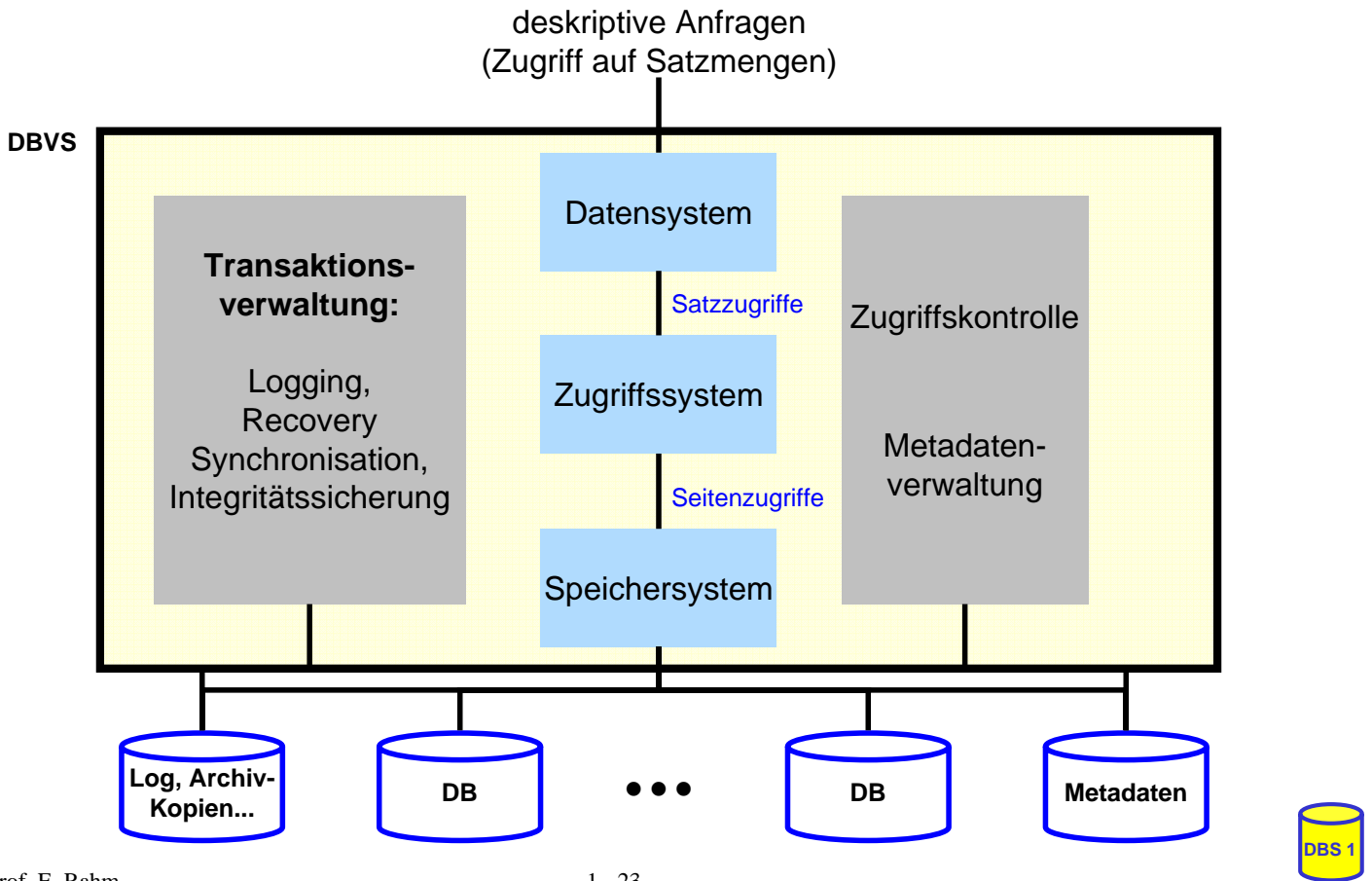
```
PERSONAL
  (PERSONAL_NUMMER CHAR (6)
  ABT_NUMMER      CHAR (4)
  ...             )
```

### Internes Schema:

```
STORED_PERS LENGTH=18
  PREFIX     TYPE=BYTE(6), OFFSET=0
  PNUM       TYPE=BYTE(6), OFFSET=6, INDEX=PNR
  ABT#       TYPE=BYTE(4), OFFSET=12
  PAY        TYPE=FULLWORD, OFFSET=16
  ...
```

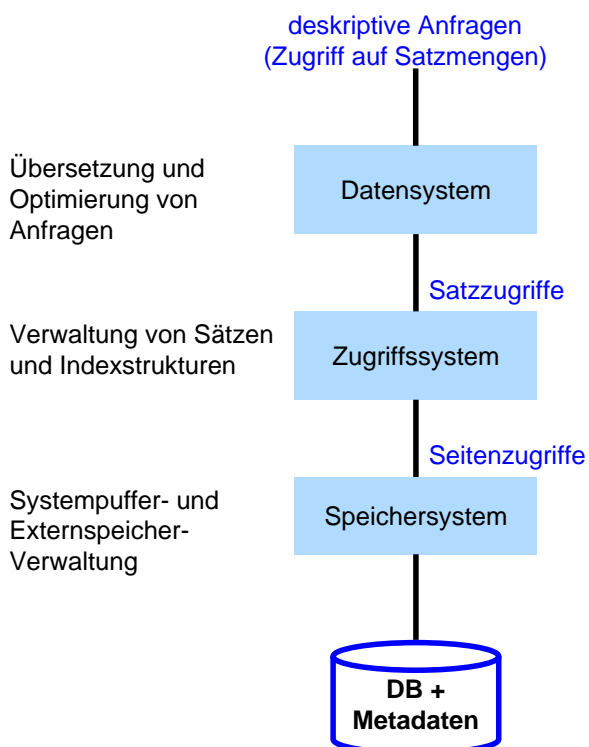


# Grobaufbau eines DBS

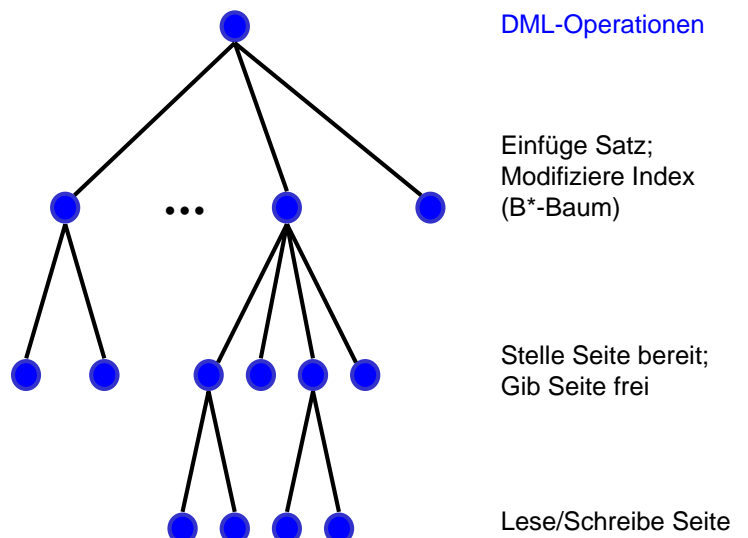


# Grobaufbau eines DBS (2)

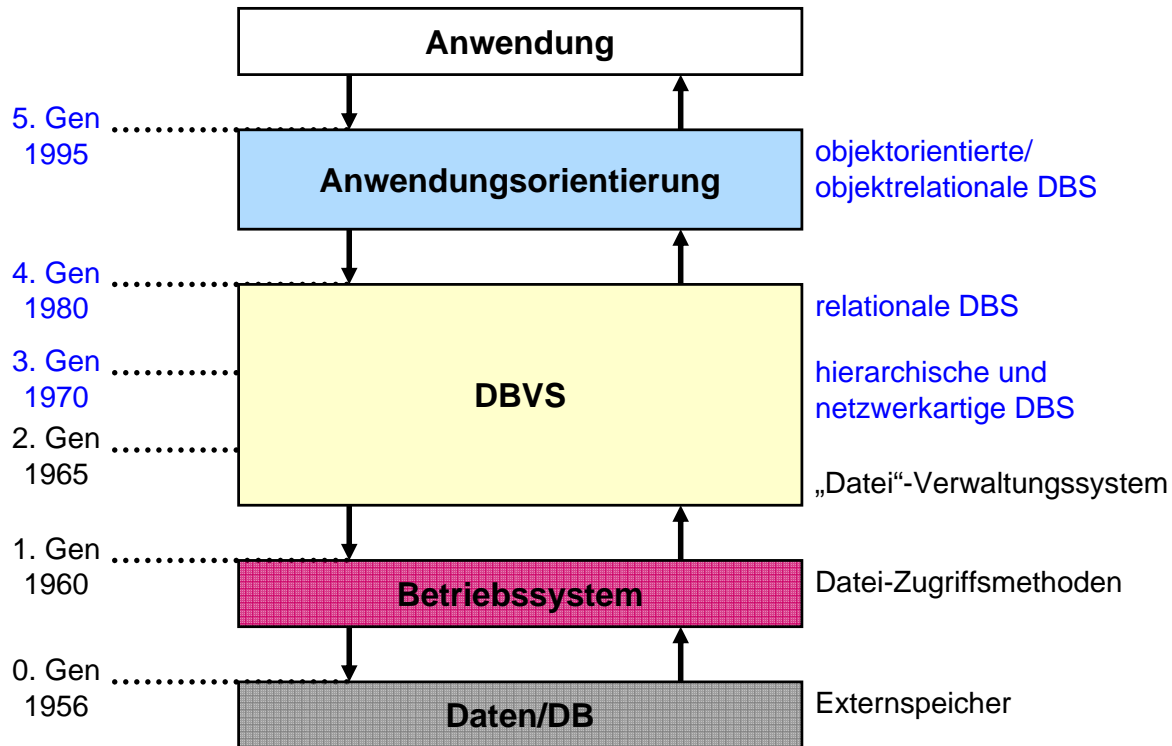
## Schichtenmodell



## Dynamischer Kontrollfluss einer DB-Operation



# Historische Entwicklung

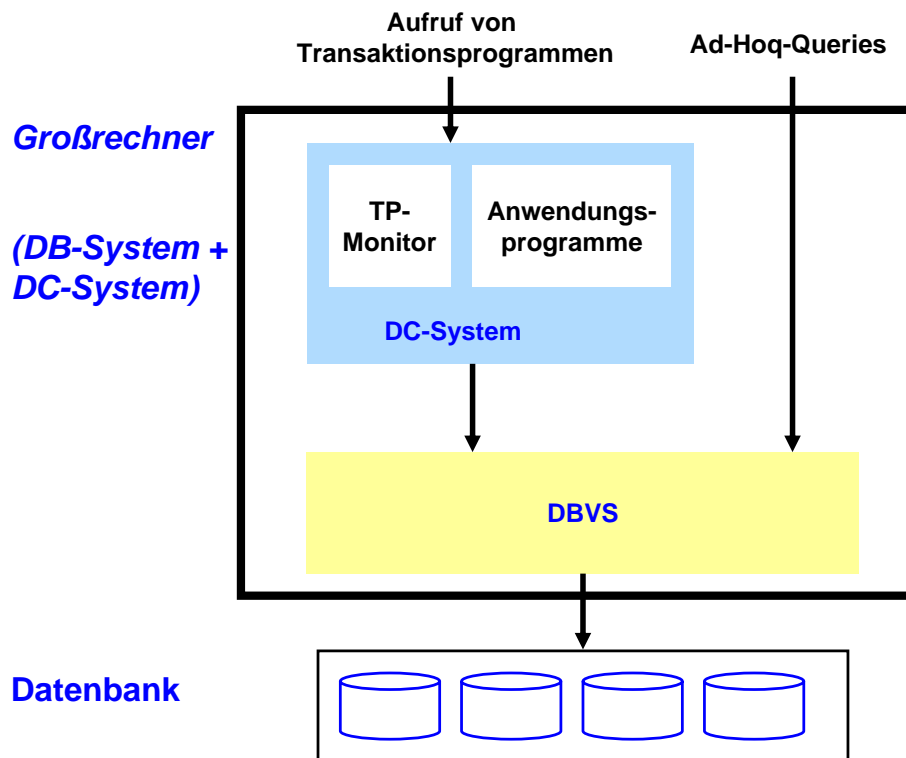


## Einsatzformen von DBS

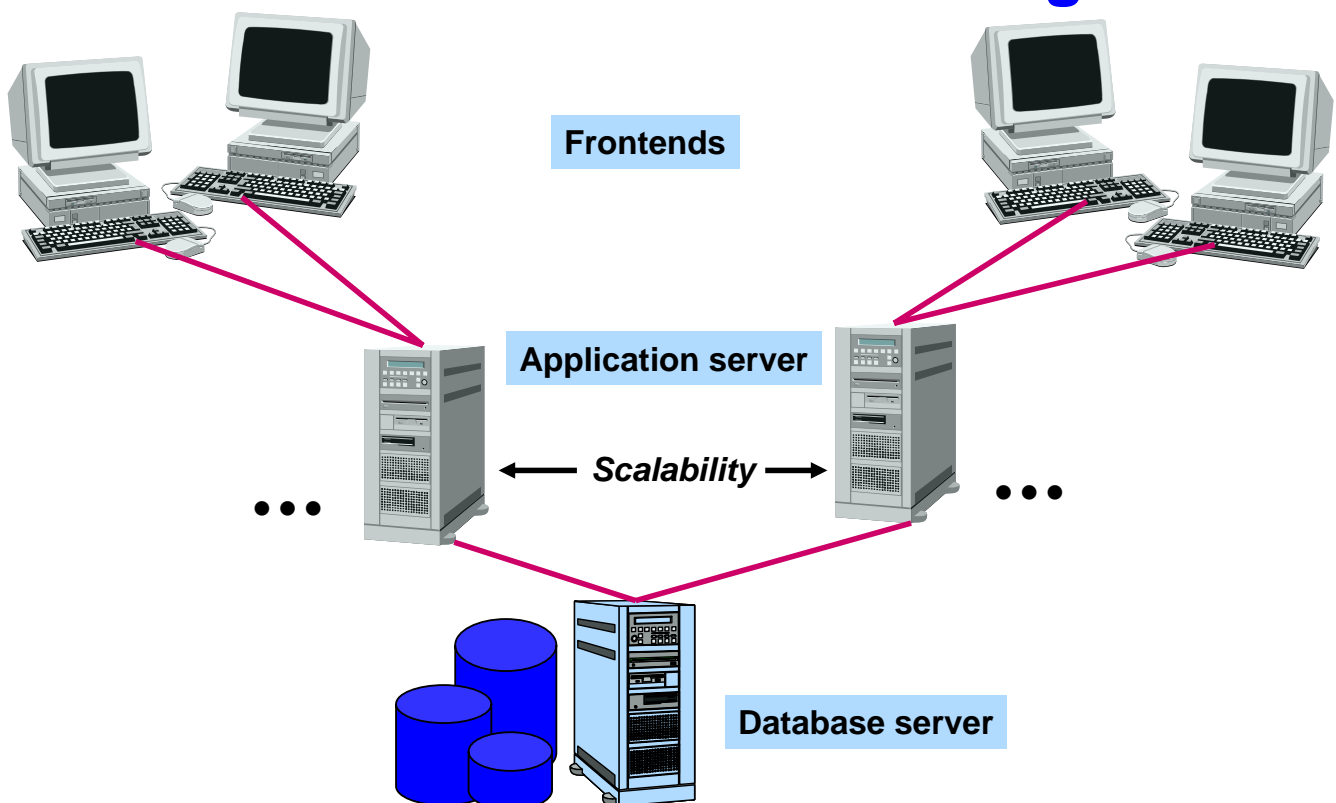
- dominierende DBS-Nutzung im Rahmen von **Transaktionssystemen (OLTP, Online Transaction Processing)** sowie E-Business: Ausführung vorgeplanter Anwendungen
- Online-Transaktion: Ausführung eines Programmes, das mit Hilfe von Zugriffen auf gemeinsam genutzte Datenbank eine i. a. nicht-triviale Anwendungsfunktion erfüllt, z. B.
  - Bearbeiten einer Bestellung. Platzreservierung für einen Flug
  - Kontostandsabfrage; Abbuchen eines Geldbetrages; Überweisung
  - Anmelden eines Autos, Abwickeln eines Telefonanrufes, ...
- weitere DBS-Einsatzfelder / -Ausprägungen
  - Decision Support: **OLAP (Online Analytical Processing)**, Data Warehousing, Data Mining
  - Multimedia-, Geo-, Volltext-DBS, XML-DBS
  - Deduktive DBS, Wissensbankverwaltungssysteme
- Architektur: zentralisierte DBS vs. Mehrrechner-DBS



# Grobaufbau eines zentralisierten Transaktionssystems (ca. 1985)



# 3-stufige Client/Server-Architektur zur Transaktionsverarbeitung



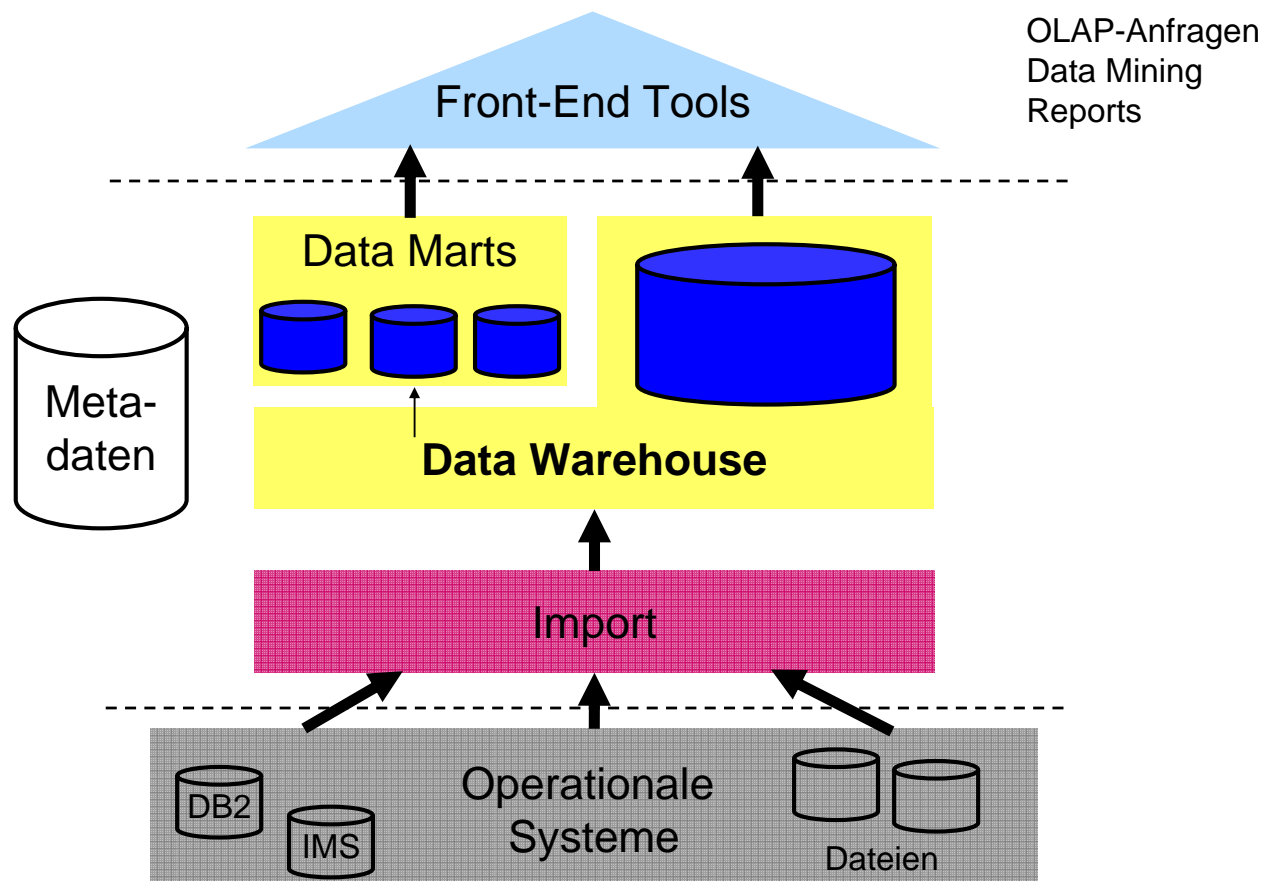
# Entscheidungsunterstützende Systeme

(Decision Support Systems, DSS)

- **OLAP** (Online Analytical Processing) vs. OLTP (Online Transaction Processing)
  - Analyse betrieblicher Datenbestände
- häufiger Einsatz von **Data Warehouses**
  - Integration der Datenbestände eines Unternehmens für Analysen aus Sicht der Endbenutzer
  - physisches Kopieren und Transformieren der Daten
  - Nutzung unterschiedlicher Analysewerkzeuge
  - Bsp.: Umsatzentwicklung nach Zeit, Produktklasse, Region, etc.
- **Data Mining**: Aufspüren von inhärenten Daten-/Informationsmustern aus großen Datenbeständen
  - oft synonym: KDD (Knowledge and Data Discovery)
  - eigenständiges Entdecken von „interessanten“ Mustern (nicht nur Beantwortung gestellter Fragen)



## Data-Warehouse-Umfeld



# Zusammenfassung

- Datenverwaltung durch Dateisysteme unzureichend
- DBS-Charakteristika
  - Effiziente Verwaltung persistenter und strukturierter Daten
  - Datenstrukturierung und Operationen gemäß Datenmodell/DB-Sprache
  - Transaktionskonzept (ACID): Atomarität, Konsistenzerhaltung, kontrollierter Mehrbenutzerbetrieb, Persistenz erfolgreicher Änderungen
  - zentrale (integrierte) Datenbank mit hohem Grad an Datenunabhängigkeit
- relationale DBS: mengenorientierte DB-Schnittstelle
- 3-Ebenen-Architektur: externes, konzeptionelles, internes Schema
- Schichtenmodell eines DBVS
  - interne Schichten für Seiten, Sätze und Satzmengen
  - Querschnittsaufgaben: Transaktionsverwaltung und Metadaten
- Haupt-Einsatzformen von DBS in Unternehmen:
  - Transaktionssysteme (OLTP) / E-Business
  - Entscheidungsunterstützung (OLAP, Data Mining)

