1. Einführung / Grundlagen von DBS

- DBS vs. Dateisysteme
- Eigenschaften von DBS
- Datenmodelle
- Transaktionskonzept (ACID)

- Aufbau von DBS
  - Schemaarchitektur
  - Schichtenmodell
- Historische Entwicklung

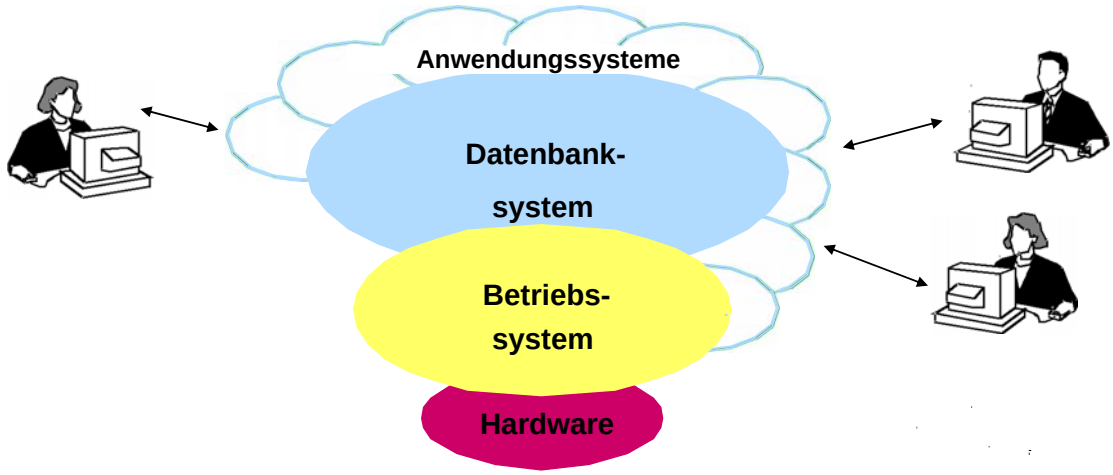
- Einsatzformen von DBS (OLTP, OLAP)

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 + Anwendungssysteme + Benutzerschnittstellen
- DBS = DB + 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.

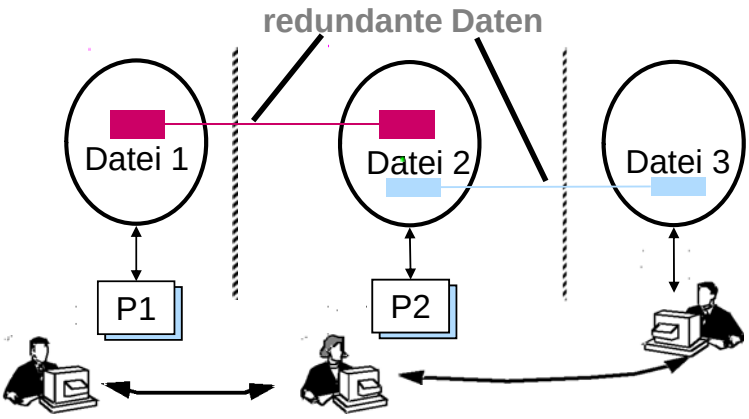
Beispiele für Informationssysteme

- Hochschulinformationssystem (Universitäts-DB)
  - Verwaltung von Studenten, Fakultäten, Professoren, Mitarbeitern
  - Studenten belegen Vorlesungen von Professoren und legen bei ihnen Prüfungen ab
  - Anwendungsvorgänge: Immatrikulation, Rückmeldung, Exmatrikulationen, Prüfungsverwaltung, Stundenplanerstellung, Planung der Raumbelegung, Ausstellen von Zeugnissen, etc.
- Datenbank eines Produktionsbetriebes
  - Verwaltung verschiedener Abteilungen und deren Beschäftigte
  - Produktdaten: die in einem Betrieb hergestellten Endprodukte setzen sich i.a. aus mehreren Baugruppen und vielen Einzelteilen zusammen. Jedes Teil kann von Lieferanten bezogen werden.
  - Typische Anwendungsvorgänge : Personalverwaltung (Einstellung / Entlassung, Lohn- und Gehaltsabrechnung), Bestellung und Lieferung von Einzelteilen, Verkauf von Fertigprodukten, Lagerhaltung, Bedarfsplanung, Stücklistenauflösung

Motivation für Einsatz eines DBS

Typische Probleme bei Informationsverarbeitung ohne DBVS (z.B. Nutzung von Dateisystemen)

- Redundanz und Inkonsistenz
- Beschränkte Zugriffsmöglichkeiten
- hohe Entwicklungskosten für Anwendungsprogramme



DBS-Motivation / Probleme Dateisysteme (2)

- enge Bindung von Dateistrukturen an Programmstrukturen (geringe „Datenunabhängigkeit“)
  - Änderungen im Informationsbedarf sowie bei Leistungsanforderungen erfordern Anpassungen der Datenstrukturen, die auf Anwendungen durchschlagen
  - verschiedene Anwendungen brauchen verschiedene Sichten auf dieselben Daten
- Probleme beim Mehrbenutzerbetrieb
- Verlust von Daten
- Integritätsverletzung
- Sicherheitsprobleme
  - Annahmen: Alles bleibt stabil ! Alles geht gut !

Aufgaben/Eigenschaften von DBS

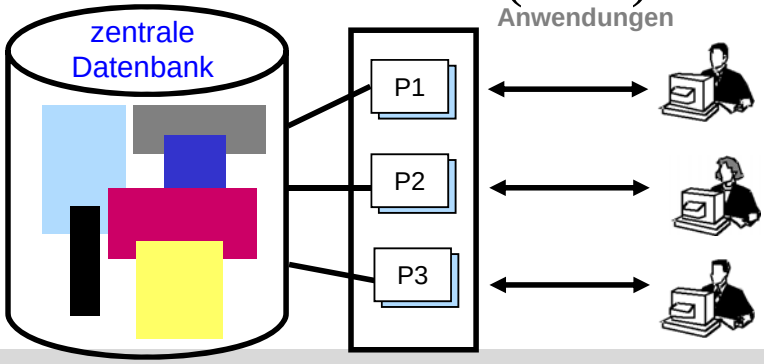
- Generell: effiziente und flexible Verwaltung großer Mengen persistenter Daten (z. B. T Bytes)

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)
6. **Automatisierte Zugriffskontrolle / Datenschutz**
   - Zugriffsrechte für einzelne DB-Objekte
7. **Ständige Verfügbarkeit / Betriebsbereitschaft**
   - 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
- Verwaltung durch Datenbankadministrator (DBA)

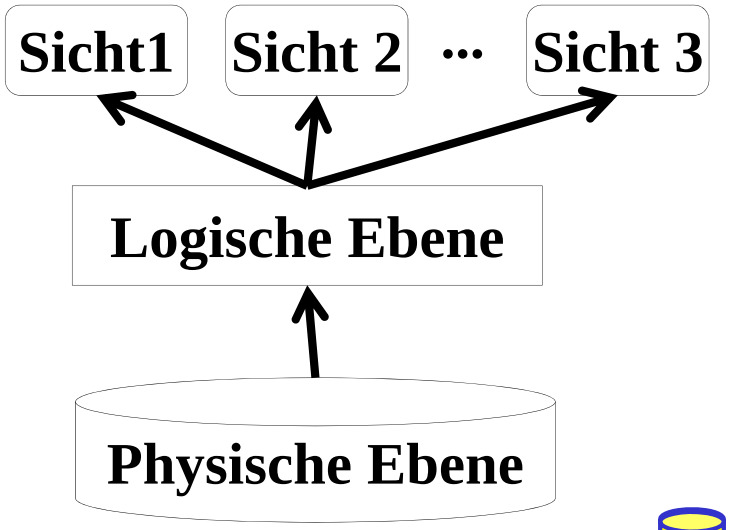
zentrale DB
statt verteilter Dateien



2. Hohe Datenunabhängigkeit

- Datenunabhängigkeit = Maß für die Isolation zwischen Anwendungsprogrammen und Daten
- Gefordert: Isolation der Anwendungsprogramme von den Daten

  sonst: extremer Wartungsaufwand für die Anwendungsprogramme
- physische Datenunabhängigkeit
  - Unabhängigkeit gegenüber Geräteeigenschaften, Speicherungsstrukturen ...
- logische Datenunabhängigkeit
  - Unabhängigkeit gegenüber logischer Strukturierung der Daten
  - i. a. nur teilweise erreichbar



Abstraktionsebenen eines DBS

3. Hohe Leistung und Skalierbarkeit

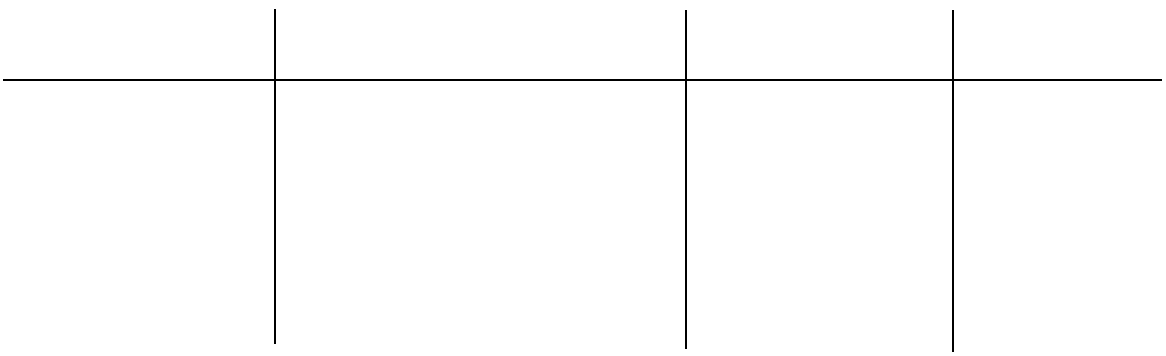
- Hoher Durchsatz / kurze Antwortzeiten für DB-Operationen auf großen Datenmengen
  - „trotz“ loser Bindung der Programme an die Daten (Datenunabhängigkeit)
- Leistung ist DBS-Problem, nicht Anwendungsproblem
  - Optimierung von DB-Anfragen durch das DBS (Query-Optimierung)
  - Festlegung von Indexstrukturen, Datenallokation etc. durch 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



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)
- Abdeckung verschiedener Nutzerklassen: Systempersonal, Anwendungsprogrammierer, „anspruchsvolle Laien“

Relationenmodell

Beispiel: Universitäts-DB

FAK

FNR	FNAME	DEKAN

PROF

PNR	PNAME	FNR	FACHGEB

STUDENT

MATNR	SNAME	FNR	W-ORT

PRÜFUNG

PNR	MATNR	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

Eine Transaktion ist eine Folge von DB-Operationen (DML-Befehlen), welche die Datenbank von einem logisch konsistenten Zustand in einen (möglicherweise geänderten) logisch konsistenten Zustand überführt.

Das DBS gewährleistet für Transaktionen die vier sogenannten ACID-Eigenschaften.

- **A**tomicity: 'Alles oder Nichts'-Eigenschaft (Fehlerisolierung)
- **C**onsistency: eine erfolgreiche Transaktion erhält die DB-Konsistenz (Gewährleistung der definierten Integritätsbedingungen)
- **I**solation: alle Aktionen innerhalb einer Transaktion müssen vor parallel ablaufenden Transaktionen verborgen werden („logischer Einbenutzerbetrieb")
- **D**urability: Überleben von Änderungen erfolgreich beendeter Transaktionen trotz beliebiger (erwarteter) Fehler garantieren (*Persistenz*).

Transaktionskonzept (2)

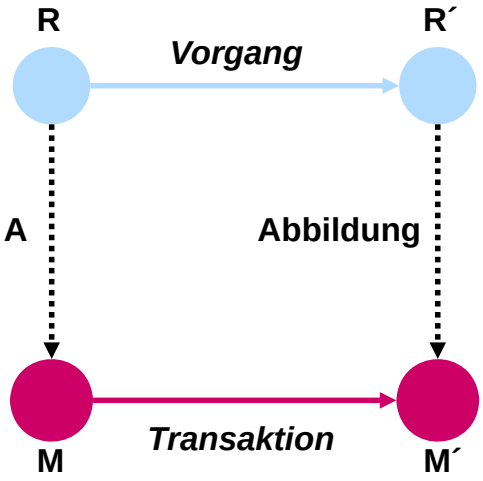
- Programmierschnittstelle für Transaktionen
  - begin of transaction (BOT)
  - commit transaction („commit work" in SQL)
  - rollback transaction („rollback work" in SQL)
- Mögliche Ausgänge einer Transaktion

BOT	BOT	BOT
DML1	DML1	DML1
DML2	DML2	DML2
• • •	• • •	
DMLn	DMLn	
COMMIT WORK	ROLLBACK WORK	erzwungenes ROLLBACK
		Systemausfall, Programmfehler usw.
normales Ende	abnormales Ende	abnormales Ende

Datenintegrität

- ACID impliziert v erschiedene Arten an Datenintegrität
- 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*)
  - Ziel: möglichst geringe gegenseitige Behinderung

Modell einer Miniwelt: Grobe Zusammenhänge



R: Realitätsausschnitt (Miniwelt)

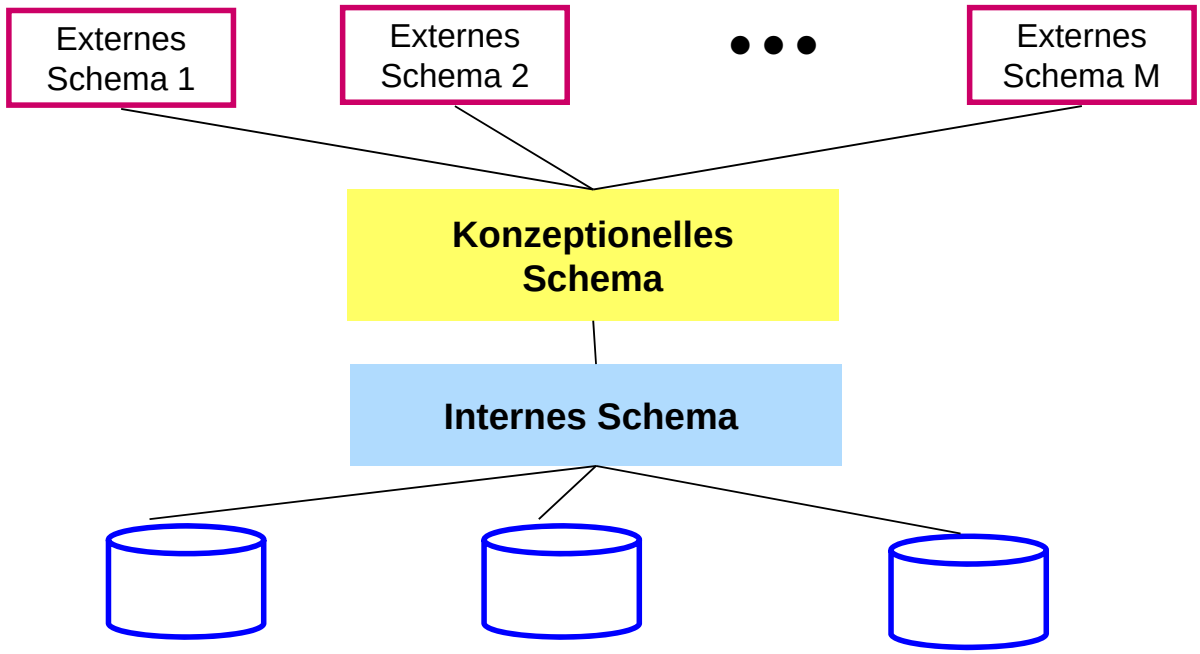
M: Modell der Miniwelt
(beschrieben durch DB-Schema)

A: Abbildung aller wichtigen Objekte und Beziehungen (Entities und Relationsships)
=> 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
  - abtrahiert 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)
  - abtrahieren 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) ...
```

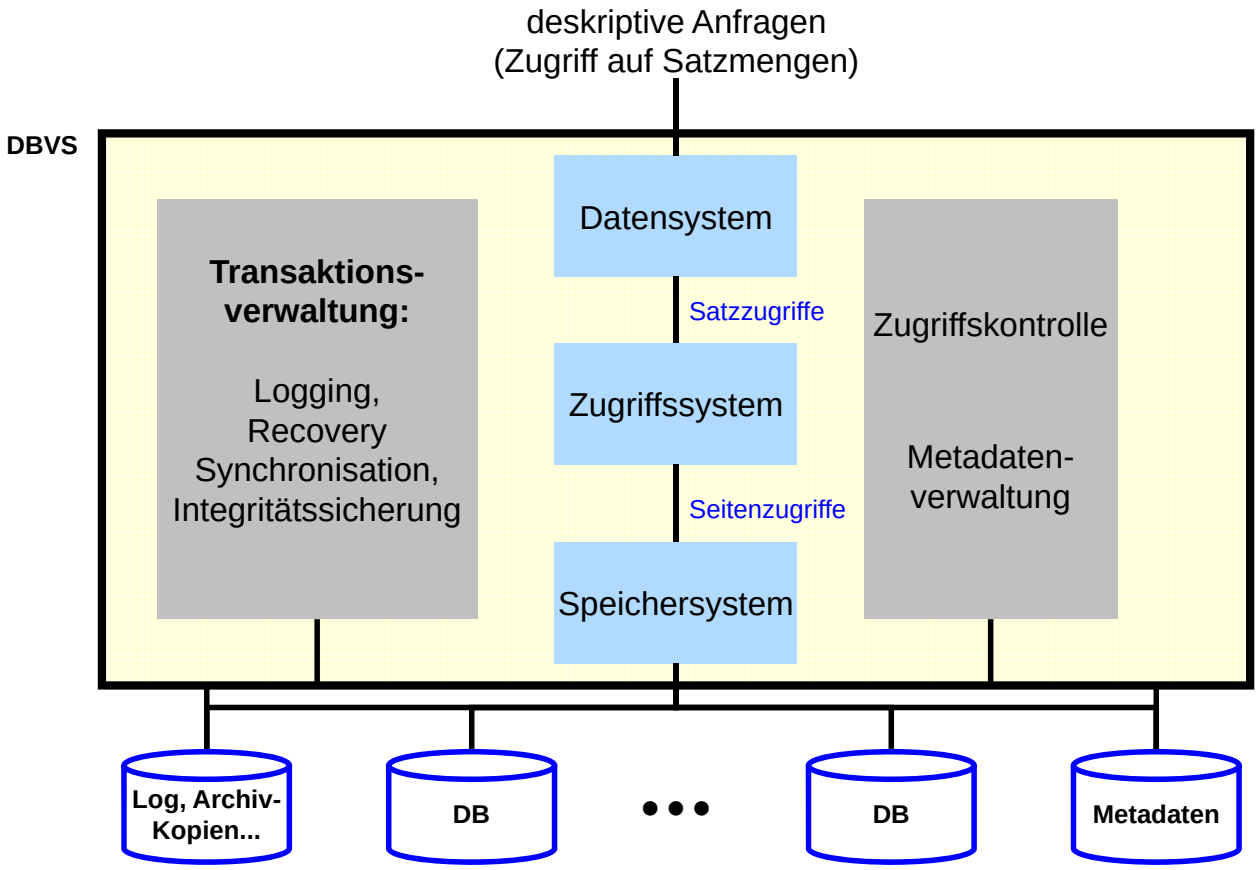
Konzeptionelles Schema:

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

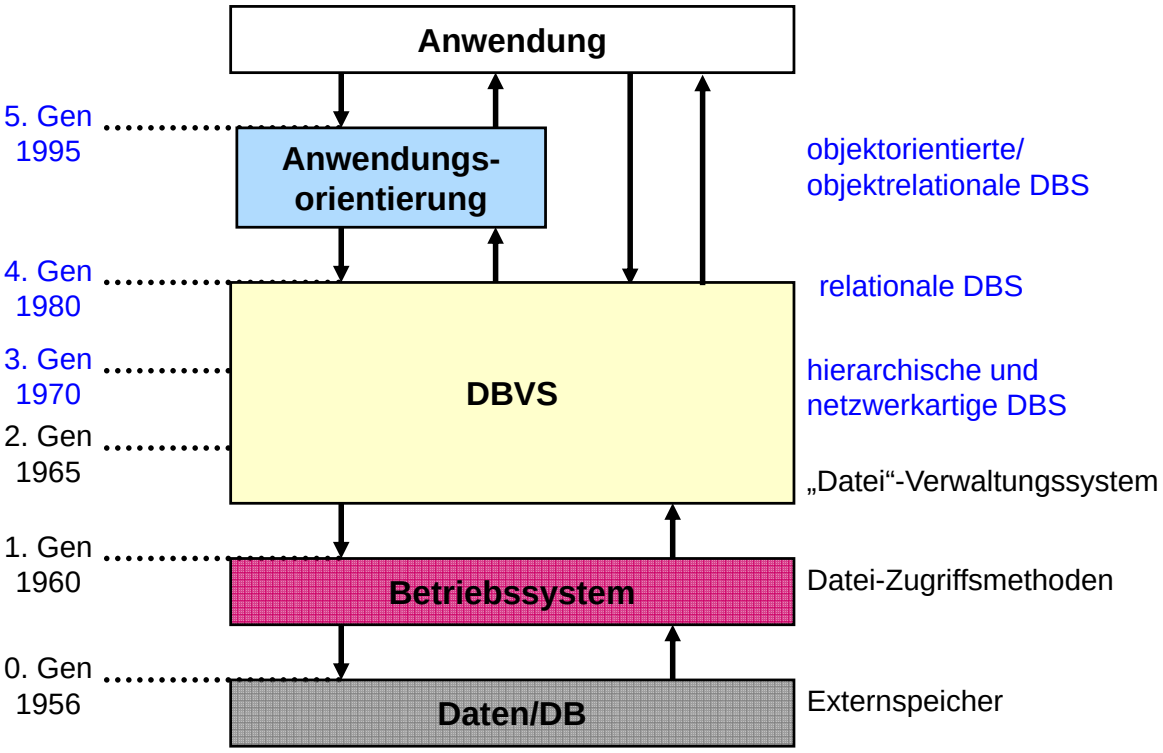
Internes Schema:

```
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



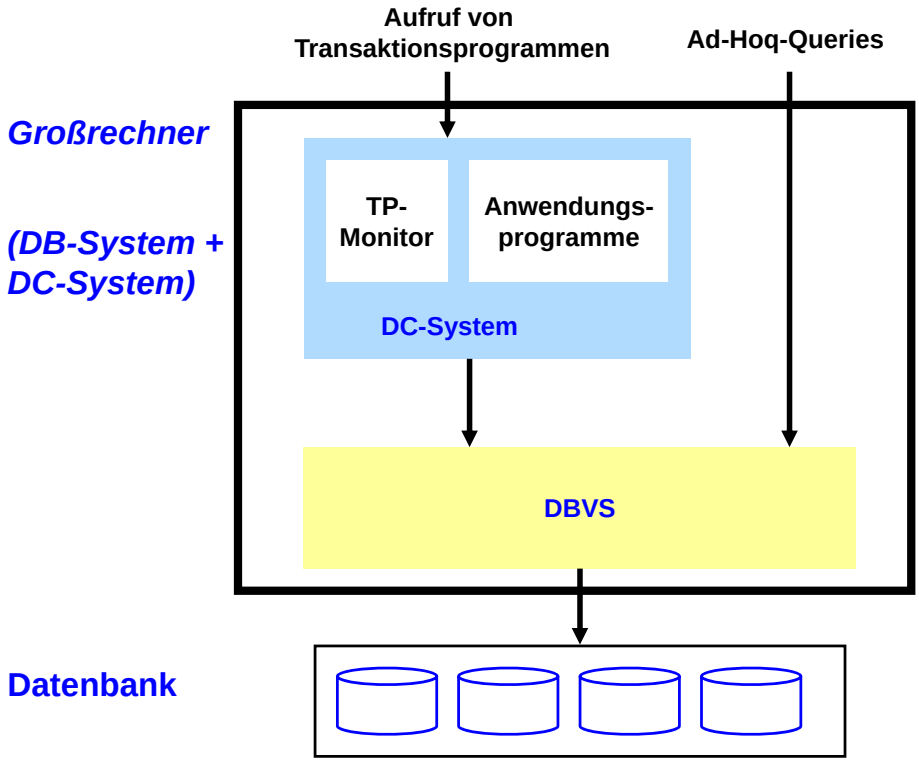
Historische Entwicklung von DBS



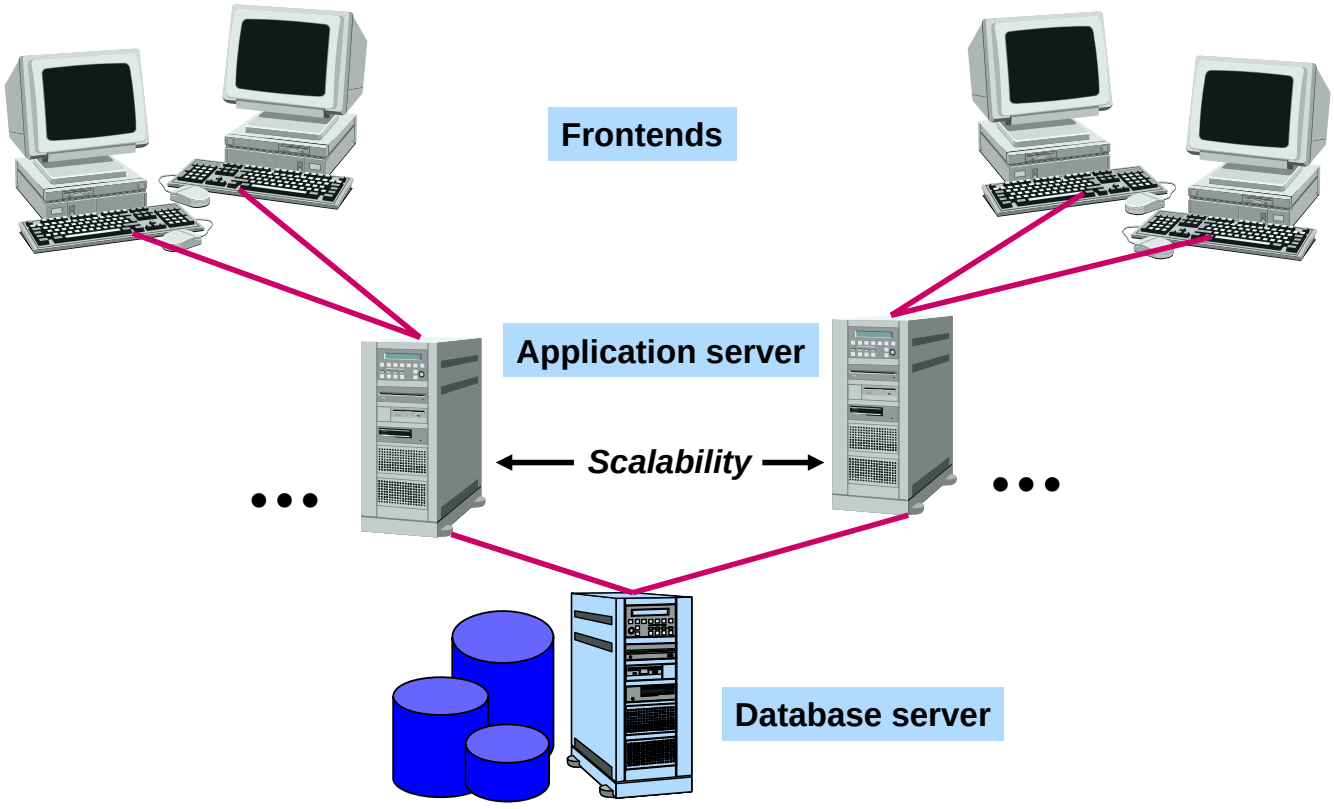
Einsatzformen von DBS

- dominierende DBS-Nutzung im Rahmen von **Transaktionssystemen** (OLTP, Online Transaction Processing) sowie E-Business: Ausführung vorgeplanter Anwendungen
- Online-Transaktion: Ausführung eines Programmes, das auf gemeinsam genutzte Datenbank zugreift, um eine Anwendungsfunktion zu realisieren
  - 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
  - Content Management Systeme (Verwaltung von Webseiten, Dokumenten, Multimediadaten)
  - Geodatenbanken, Wissensbanksysteme …

Grobaufbau eines zentralisierten Transaktionssystemes (ca. 1985)



3-stufige Client/Server-Architektur zur Transaktionsverarbeitung

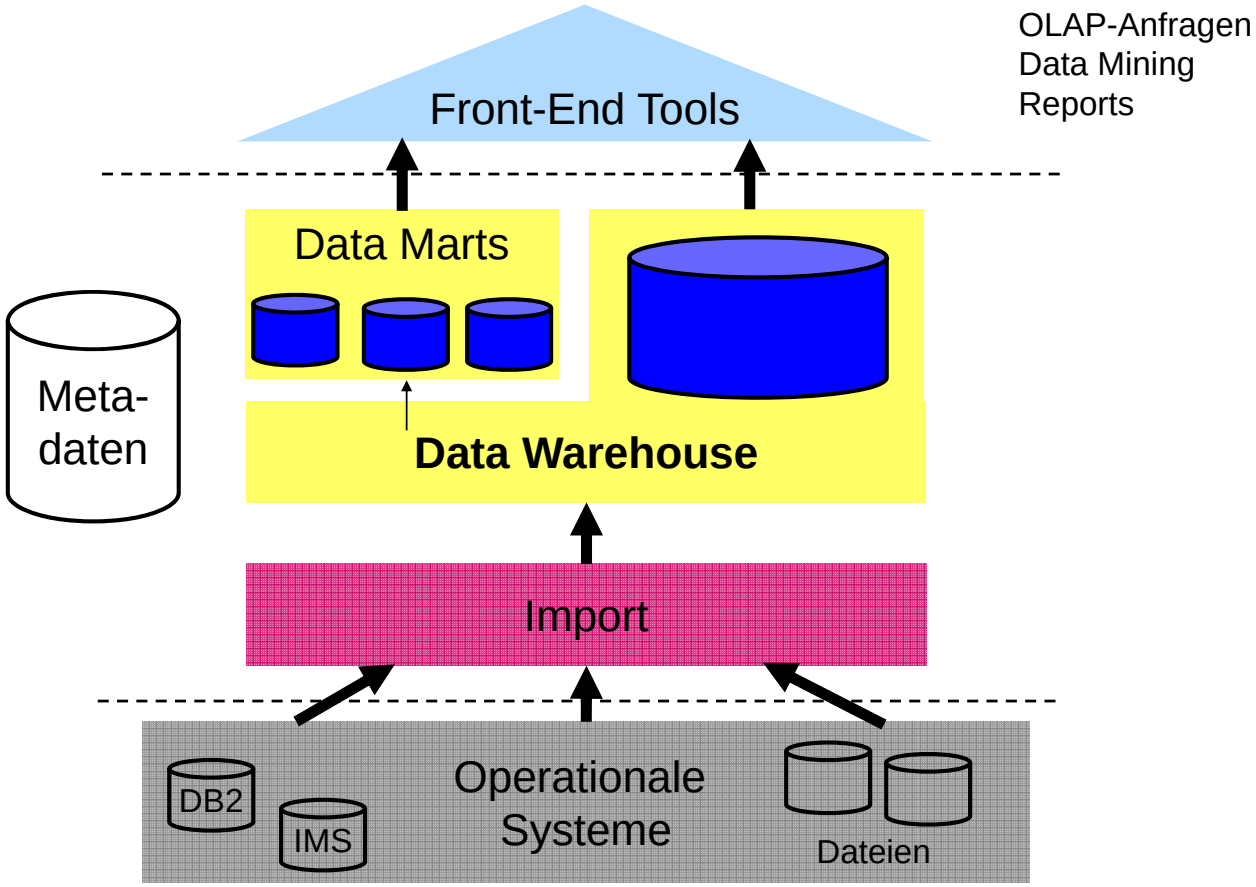


Entscheidungsunterstützende Systeme

(Decision Support Systems, DSS)

- **OLAP** (Online Analytical Processing) vs. OLTP (Online Transaction Processing)
  - Umfassende Auswertung und Analyse betrieblicher Datenbestände
- häufiger Einsatz von **Data Warehouses**
  - Integration der Datenbestände eines Unternehmens für Analysen aus Sicht der Endbenutzer
  - Bsp.: Umsatzentwicklung nach Zeit, Produktklasse, Region, etc.
  - physisches Kopieren und Transformieren der Daten
- **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)