1. Einführung / Grundlagen von DBS

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

- Aufbau von DBS
  - Schemaarchitektur
  - Schichtenmodell

- Einsatzformen von DBS: OLTP vs. OLAP

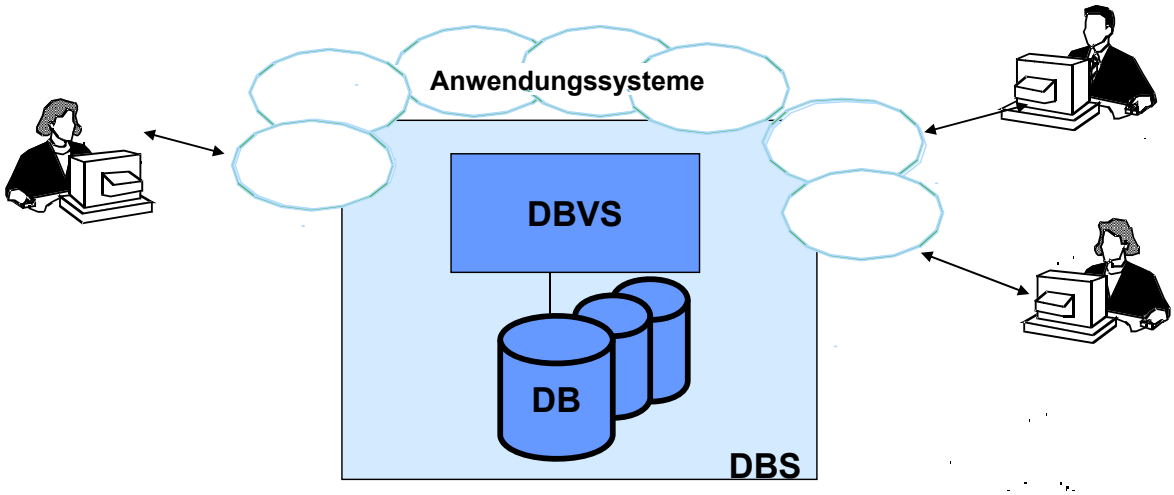
- Historische Entwicklung
  - Datenmodelle
  - Architekturen

Datenbanksysteme

- System zur dauerhaften Speicherung und Verwaltung großer Datenmengen
- Realisierung datenintensiver Anwendungen

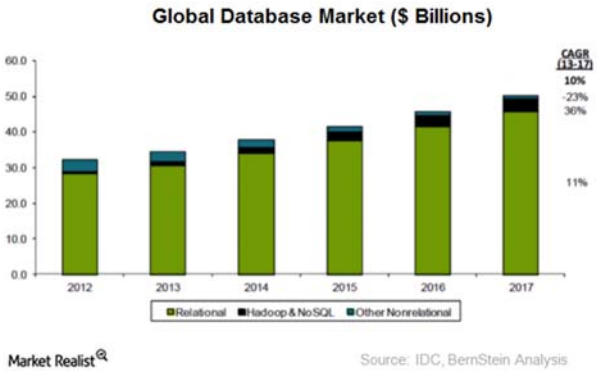
- typische Anwendungen:
  - Personaldatenbank
  - Kundenverwaltung
  - Bankanwendungen mit Konten und Zahlungsverkehr
  - Buchungssysteme/Warenverwaltung in Online-Shops
  - soziale Netzwerke (Nutzer, Aktivitäten …)
  - Infektionsgeschehen
  - ….
- Daten sind wertvoll

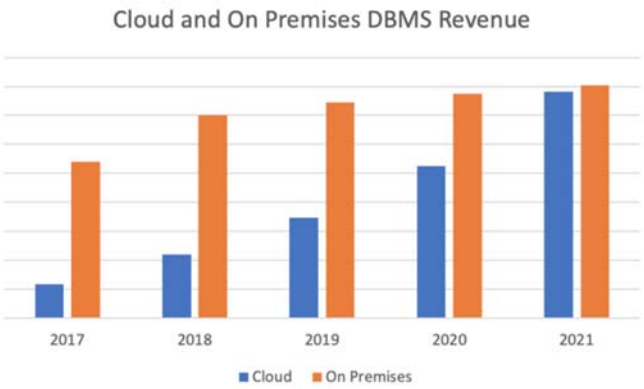
DBS als Kern von Informationssystemen (IS)



- IS = DBS + Anwendungssysteme + Benutzerschnittstellen
- DBS = DB + Datenbankverwaltungssystem (DBVS, DBMS)
  - DB: Menge der gespeicherten Daten
  - Datenbankverwaltungssystem: generisches Software-System zur Definition, Verwaltung, Verarbeitung und Auswertung der DB-Daten. Einsatz für unterschiedlichste Anwendungen

DBMS-Markt





2017		2018		2019		2020		2021	
Vendor	Share	Vendor	Share	Vendor	Share	Vendor	Share	Vendor	Share
Oracle	36.1%	Oracle	31.1%	Oracle	27.4%	Microsoft	24.3%	Microsoft	24.0%
Microsoft	21.5%	Microsoft	23.6%	Microsoft	24.7%	Oracle	23.8%	AWS	23.9%
IBM	12.7%	AWS	13.5%	AWS	17.1%	AWS	20.6%	Oracle	20.6%
AWS	9.2%	IBM	10.4%	IBM	8.8%	IBM	6.8%	Google	6.5%
SAP	7.4%	SAP	6.9%	SAP	6.5%	SAP	5.6%	IBM	5.6%

source: Gartner.com, 2022

Beispiele für Informationssysteme

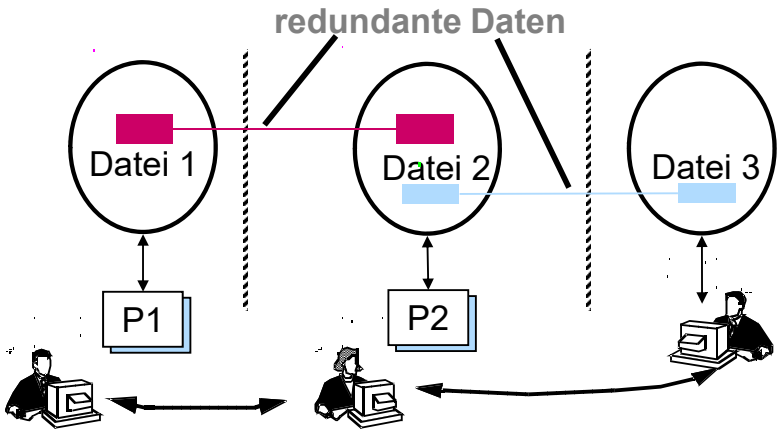
- Hochschulinformationssystem (Universitäts-DB)
  - Verwaltung von Studenten, Fakultäten, Professoren, Mitarbeitern
  - Studenten belegen Vorlesungen von Professoren und legen bei ihnen Prüfungen ab
  - Anwendungsvorgänge: Im/Exmatrikulation, Rückmeldung, Prüfungsverwaltung, Stundenplanerstellung, Raumplanung, etc.

- Datenbank eines Produktionsbetriebes
  - Verwaltung von Abteilungen und deren Beschäftigten
  - Produktdaten: Zusammensetzung von Endprodukten aus Baugruppen und Einzelteilen. Lieferbeziehungen für Teile
  - 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 die selben Daten
- Probleme beim Mehrbenutzerbetrieb
- Verlust von Daten
- Integritätsverletzung
- Sicherheitsprobleme
  - Annahmen: Alles bleibt stabil ! Alles geht gut !

Aufgaben/Eigenschaften von DBS

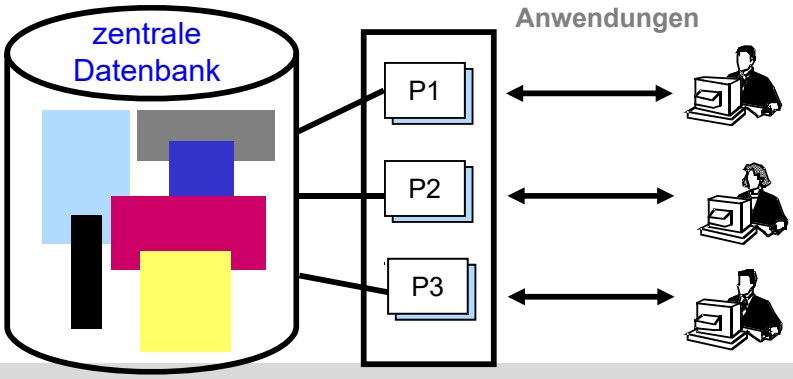
- generell: effiziente und flexible Verwaltung großer Mengen persistenter Daten (z. B. T Bytes)

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
- Verwaltung durch Datenbankadministrator (DBA)

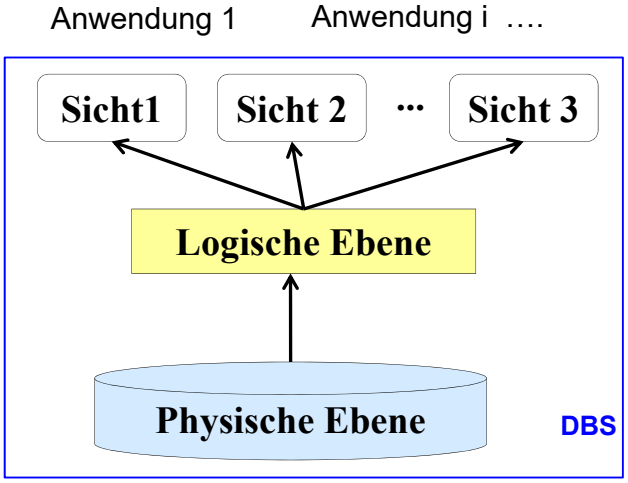
zentrale DB
statt verteilter Dateien



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 Art der physischen Datenspeicherung (Geräteeigenschaften, Indexstrukturen ...
- logische Datenunabhängigkeit
  - Unabhängigkeit gegenüber logischer Strukturierung der Daten (z.B. durch Sichten)
  - 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)
  - automatische Nutzung von Mehrprozessorsystemen, großen Hauptspeichern, parallelen Plattensystemen (Disk Arrays) etc.
  - Tuning (z.B. Festlegung von Indexstrukturen) automatisch durch DBS oder manuell durch DBA
- hohe Skalierbarkeit
  - Nutzung zusätzlicher/schnellerer Hardware-Ressourcen
  - Anpassung an steigende Leistungsanforderungen (wachsende Datenmengen und Anzahl der Benutzer)

4. Mächtige Datenmodelle

- Datenmodell/DBS-Schnittstelle
  - Definition des Datenbankaufbaus durch **DB-Schema**
  - Definition von Integritätsbedingungen und Zugriffskontrollbedingungen (Datenschutz)
  - DDL (Data Definition Language): Operationen zur Definition von Datenstrukturen
  - DML (Data Manipulation Language): Operationen zum Aufsuchen und Verändern von Daten

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 S_i, Attribute A_j und Attributwerte AW_k
  - jeder Attributwert AW_k wird durch Beschreibungsinformation (Metadaten) A_j und S_i in seiner Bedeutung festgelegt

Universität

Name	Gründungsjahr	Studenten	Fakultäten

Relationenmodell

Beispiel: Universitäts-DB (*Schema*)

FAK

FNR	FNAME	DEKAN

PROF

PNR	PNAME	FNR	FACHGEB

STUDENT

MATNR	SNAME	FNR	W-ORT

PRÜFUNG

PNR	MATNR	FACH	DATUM	NOTE

Relationenmodell (2)

Beispiel: Universitäts-DB (*Daten / Instanzen*)

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
```

Anfragesprachen

- Anfragesprache (query language) abhängig vom Datenmodell
  - Hierarchisches / Netzwerk-Datenmodell:
    navigierende / satzorientierte Operationen
  - Relationenmodell: deskriptive / mengenorientierte Operationen

- wünschenswert
  - deskriptive Problemformulierung, leichte Erlernbarkeit
  - hohe Auswahlmächtigkeit, z.B. einfache Verknüpfung mehrerer Satztypen („typübergreifende“ Operationen)
  - Standardisierung (SQL)
  - interaktiver DB-Zugriff sowie DB-Zugriff von Programmen aus
  - Unterstützung verschiedener Nutzerklassen

DBS-Nutzer/Personen

- Endbenutzer von DB-Anwendungen
  - kommuniziert nur mit Anwendung
  - benötigt keine Datenbankkenntnisse
- Anwendungsprogrammierer
  - intensive Nutzung der Datenbank mit DML
- DB-Analyst
  - interaktive Nutzung der Datenbank, z.B. mit SQL
- DB-Modellierer
  - entwirft DB-Schema
- DB-Administrator
  - richtet DB ein, vergibt Zugriffsrechte
  - führt DB-Sicherungen durch
  - optimiert die Leistung durch Tuning-Maßnahmen …
- (DBMS-Implementierer)

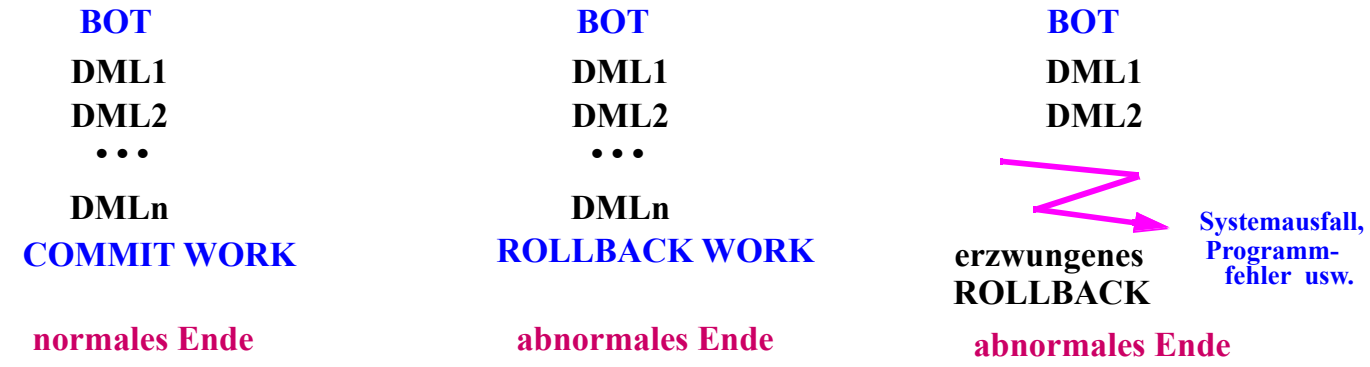
5. Transaktionskonzept

Eine Transaktion ist eine Folge von DB-Operationen (DML-Befehlen), für die das DBS die vier sogenannten ACID-Eigenschaften gewährleistet:

- **A**tomicity: 'Alles oder Nichts'-Eigenschaft (Fehlerisolierung)
- **C**onsistency: eine erfolgreiche Transaktion erhält die DB-Konsistenz (Gewährleistung der definierten Integritätsbedingungen), so dass sie die DB von einem logisch konsistenten Zustand in einen (möglicherweise geänderten) logisch konsistenten Zustand überführt.
- **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



Datenintegrität

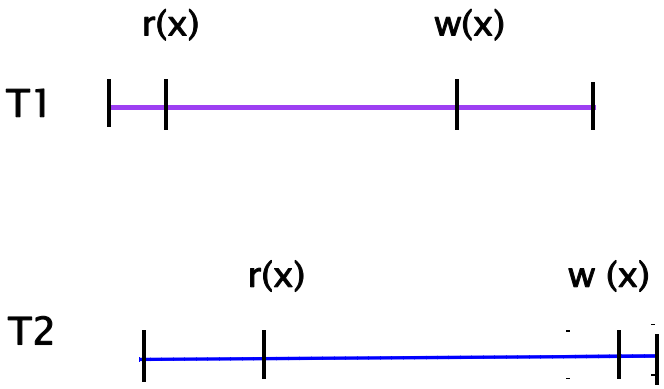
- ACID impliziert verschiedene Arten an Datenintegrität
- Consistency: Erhaltung der *logischen Datenintegrität*

- Erhaltung der *physischen Datenintegrität*
  - Führen von Änderungsprotokollen für den Fehlerfall (*Logging*)
  - Bereitstellen von Wiederherstellungsalgorithmen im Fehlerfall (*Recovery*)

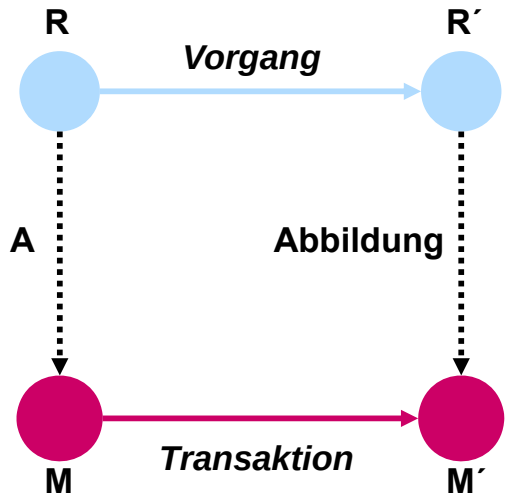
- kontrollierter Mehrbenutzerbetrieb (*Ablaufintegrität*)
  - *logischer Einbenutzerbetrieb* für jeden von n parallelen Benutzern (Leser + Schreiber)
  - Synchronisation i. a. durch Sperren (*Locking*)
  - Ziel: möglichst geringe gegenseitige Behinderung

Synchronisationsbeispiel

- Konto x mit Kontostand 100 Euro
- zwei gleichzeitige Buchungstransaktionen
  - T1: Einzahlung 100 Euro
  - T2: Abhebung 50 Euro



Modell einer Miniwelt: Grobe Zusammenhänge



R: Realitätsausschnitt (Miniwelt)

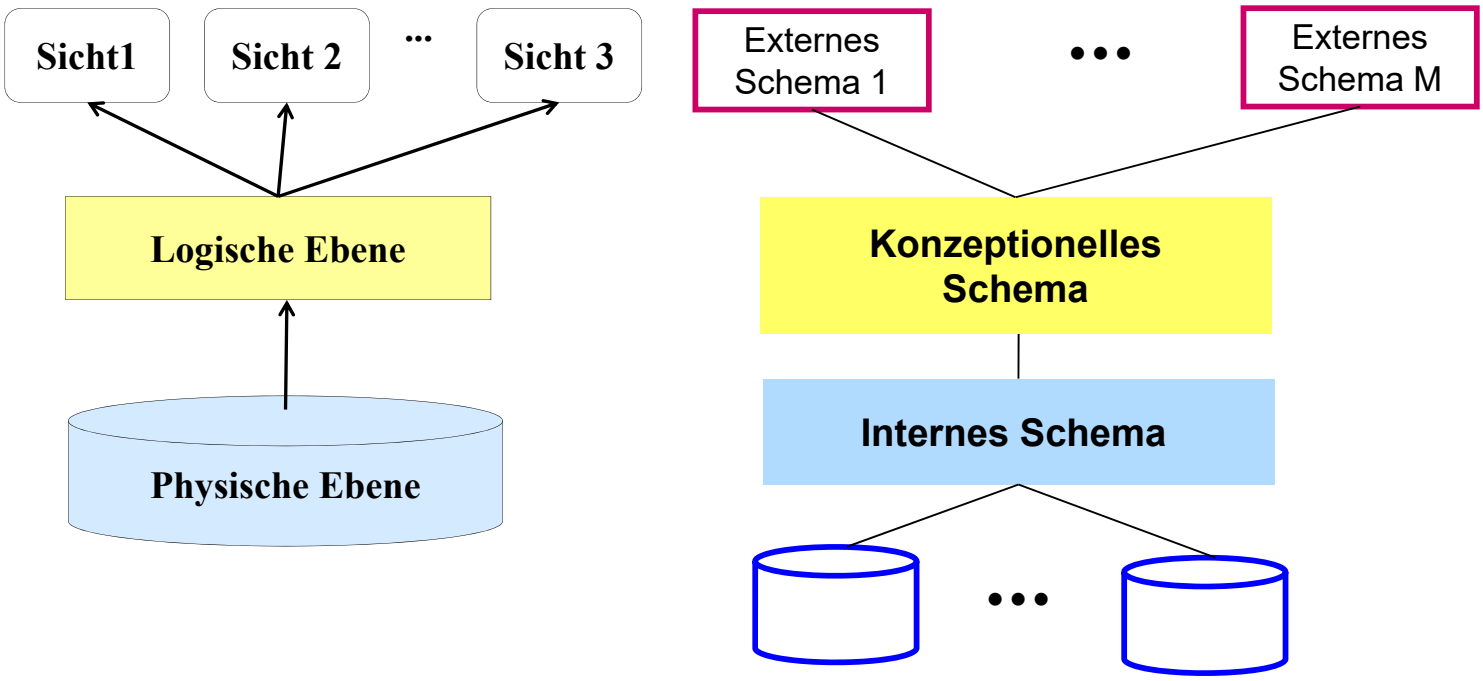
M: Modell der Miniwelt
(beschrieben durch DB-Schema)

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 / Sichtenbildung**
  - definieren spezielle Benutzersichten auf DB-Struktur (für Anwendungsprogramm bzw. Analysten)
  - abstrahieren von konzeptionellem Schema: ermöglicht partiell *logische Datenunabhängigkeit*
  - Zugriffsschutz: nur ausgewählte Relationen und Attribute bleiben zugänglich
  - reduzierte Komplexität: Anwendung sieht nur die erforderlichen Daten

Beispiel-Datenbeschreibung (vereinfacht)

Externes Schema 1

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

Externes Schema 2

```
ABT-INFO
     ANR  CHAR(4)
     ANZ-MA  INT ...
```

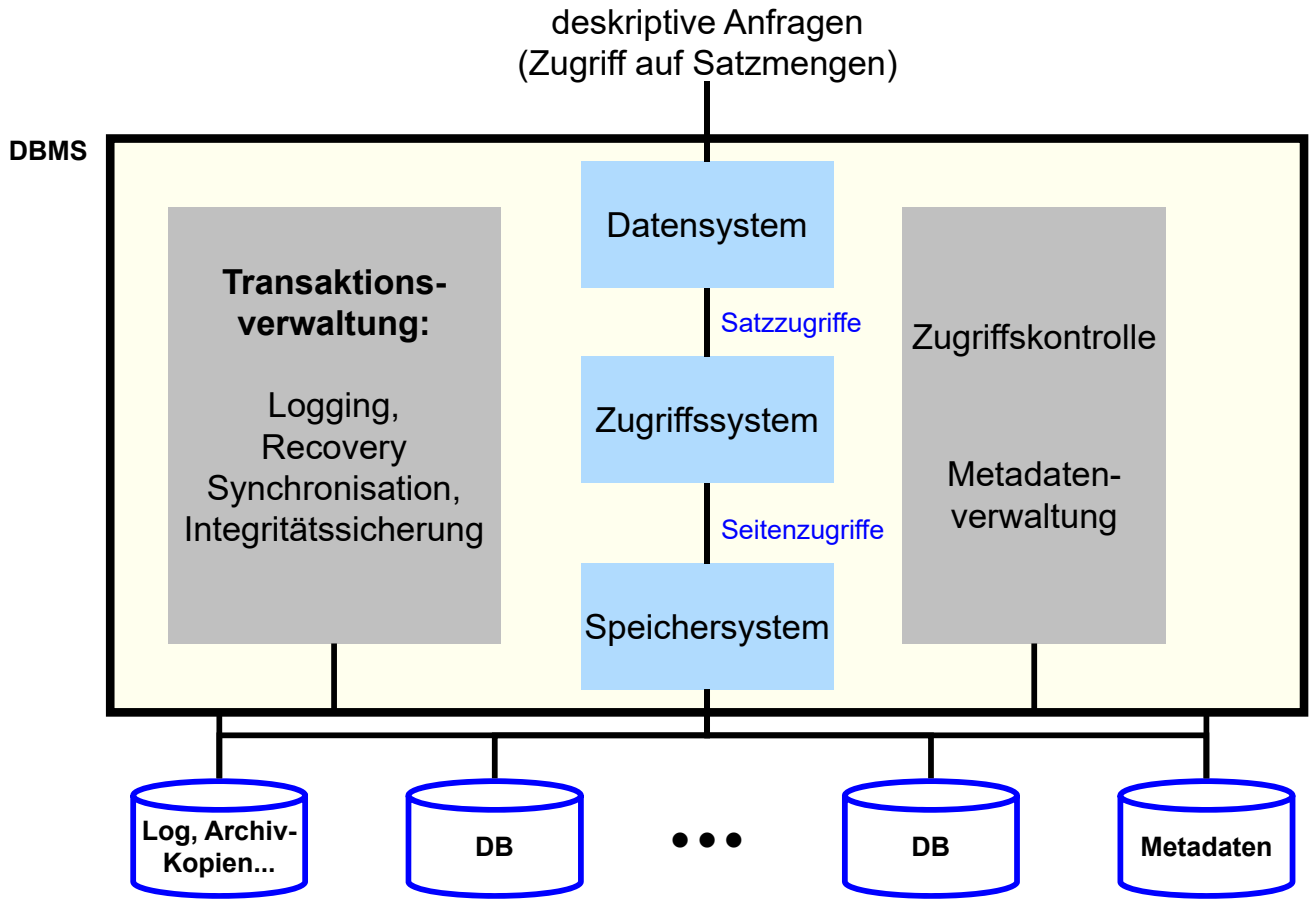
Konzeptionelles Schema:

```
PERSONAL                                   ABT
     (PERSONAL_NUMMER    CHAR (6)            (ANR   ...
     ABT_NUMMER          CHAR (4)             ABTNAME ...
     ...                      )
```

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



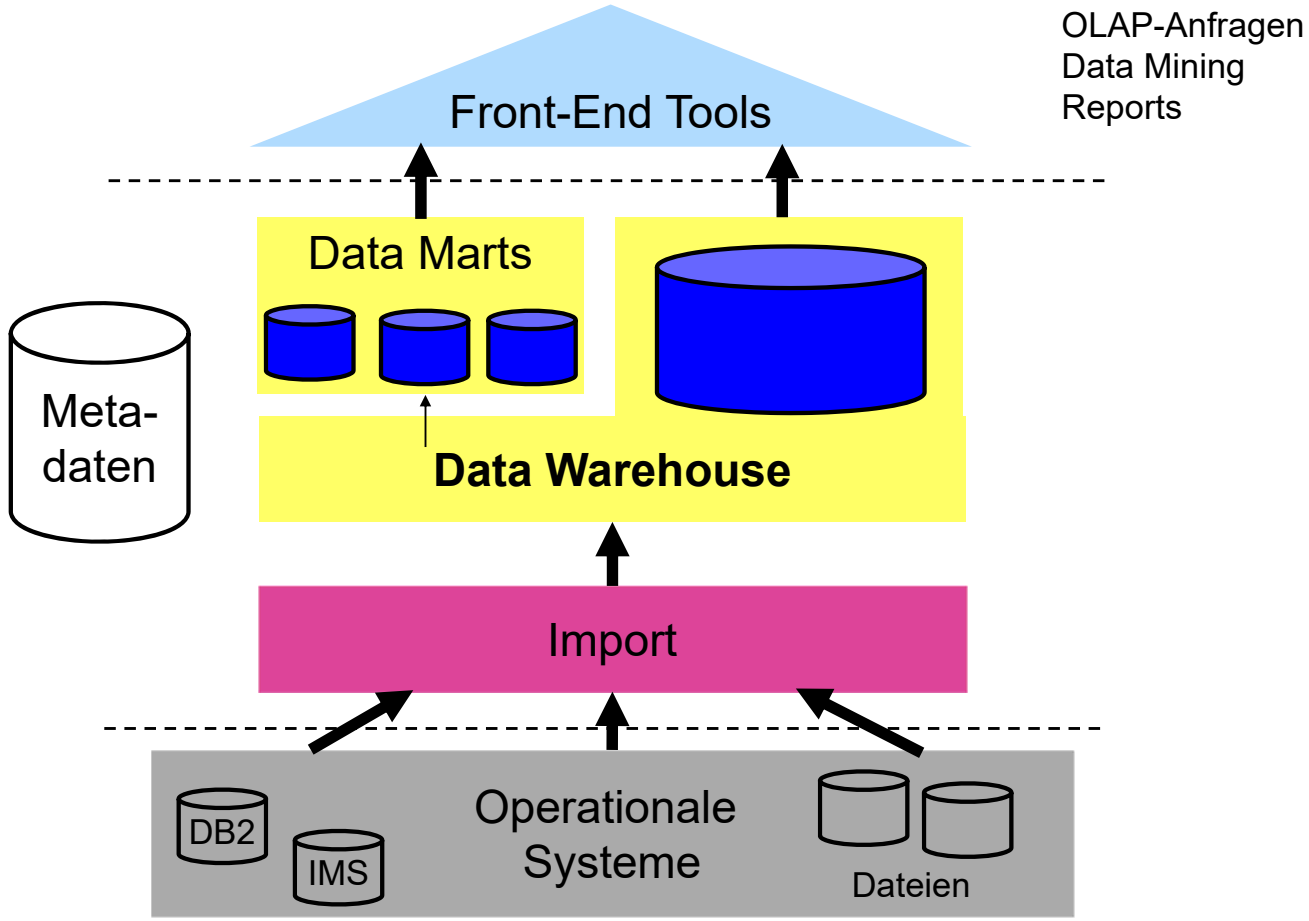
OLTP vs. OLAP (1)

- OLTP (Online Transaction Processing)
  - dominierende Einsatzform von DBS, u.a. für E-Business
  - Ausführung vorgeplanter Anwendungsprogramme mit DB-Zugriff
- Beispiele für OLTP-Transaktionen:
  - Produktbestellung, Platzreservierung (Flug, Hotel, etc.)
  - Kontostandsabfrage; Überweisung
  - Kreditkartenbezahlung
  - Abwickeln eines Telefonanrufes, ...
- Transaktionsmerkmale
  - kurze Bearbeitungszeit notwendig
  - wenige Daten pro Transaktion
  - häufige Änderungen
  - Einhaltung von ACID essenziell

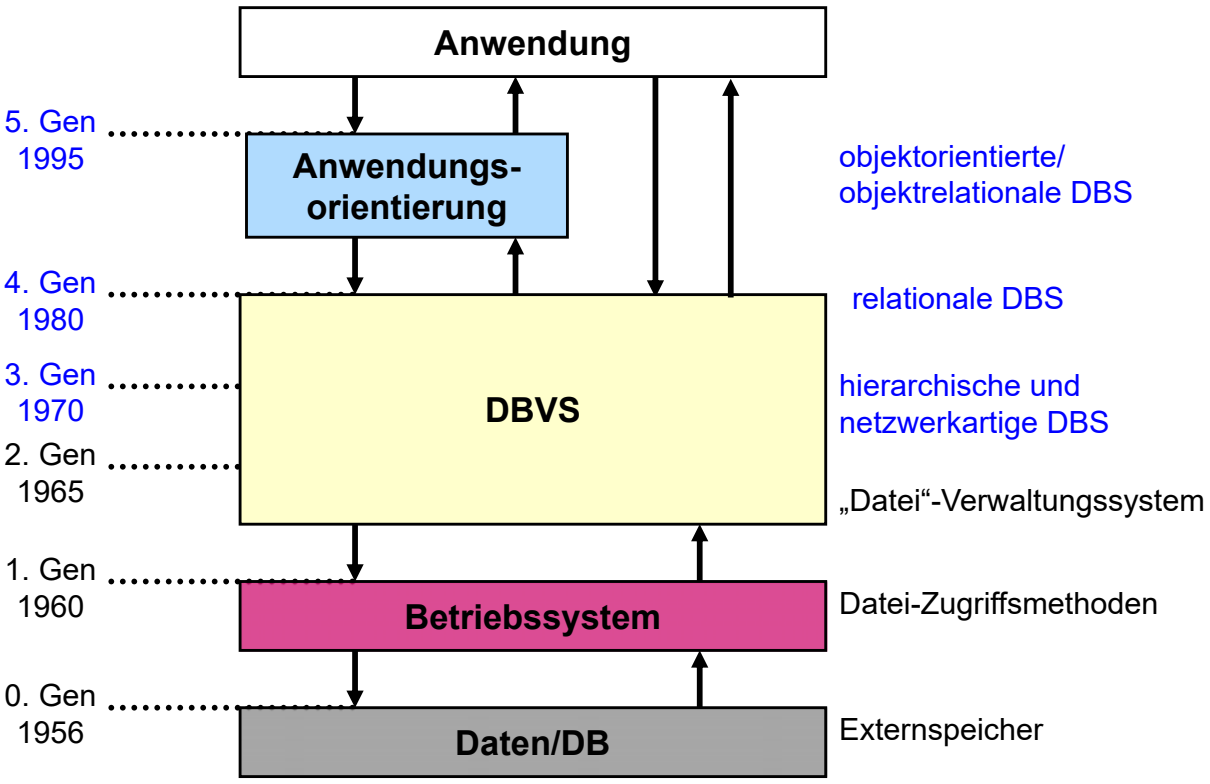
OLTP vs. OLAP (2)

- OLAP (Online Analytical Processing)
  - umfassende Auswertung/Analyse großer Datenbestände
  - Vorbereitung von Geschäftsentscheidungen (Decision Support)
  - Anwendungen: Vertriebskontrolle, Preisoptimierung, Kundenbindung …
- häufiger Einsatz von **Data Warehouses**
  - Konsolidierung und Integration der Datenbestände eines Unternehmens für Analysen
  - Bsp.: Umsatzentwicklung nach Zeit, Produktklasse, Region, etc.
- **Data Mining / Machine Learning**
  - eigenständiges Aufspüren von inhärenten Mustern in großen Datenbeständen (nicht nur Beantwortung gestellter Fragen)
  - Vorhersagen, z.B. auf Basis von Beispieldaten

Data-Warehouse-Umfeld



Historische Entwicklung von DBS



Historische Entwicklung (2)

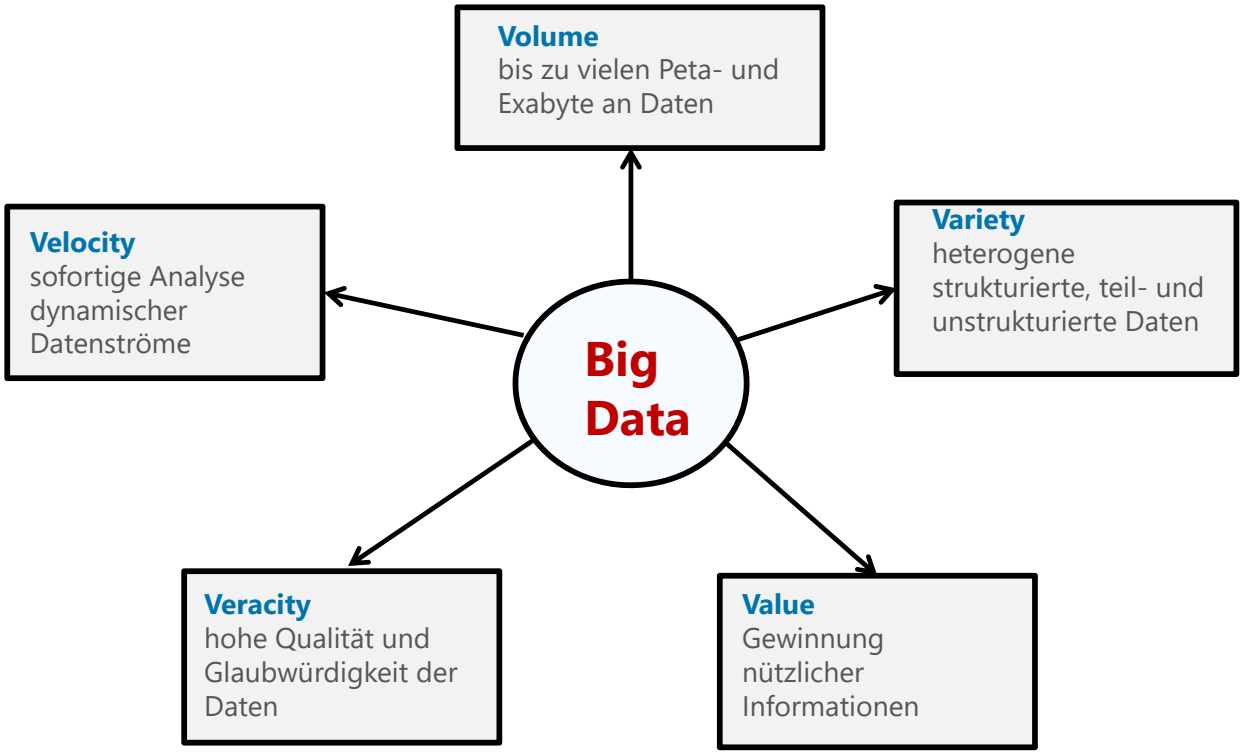
- semistrukturierte Daten / XML-Dokumente
  - optionales Schema
  - Mischung von strukturierten Daten und Text
  - Anfragesprache XQuery für XML

```
<Vorlesung >
    <Thema>DBS1</Thema>
    <Dozent>
      <Name>Prof. Rahm</Name>
      <Einrichtung>
          Uni Leipzig</Einrichtung>
    </Dozent>
</Vorlesung>
```

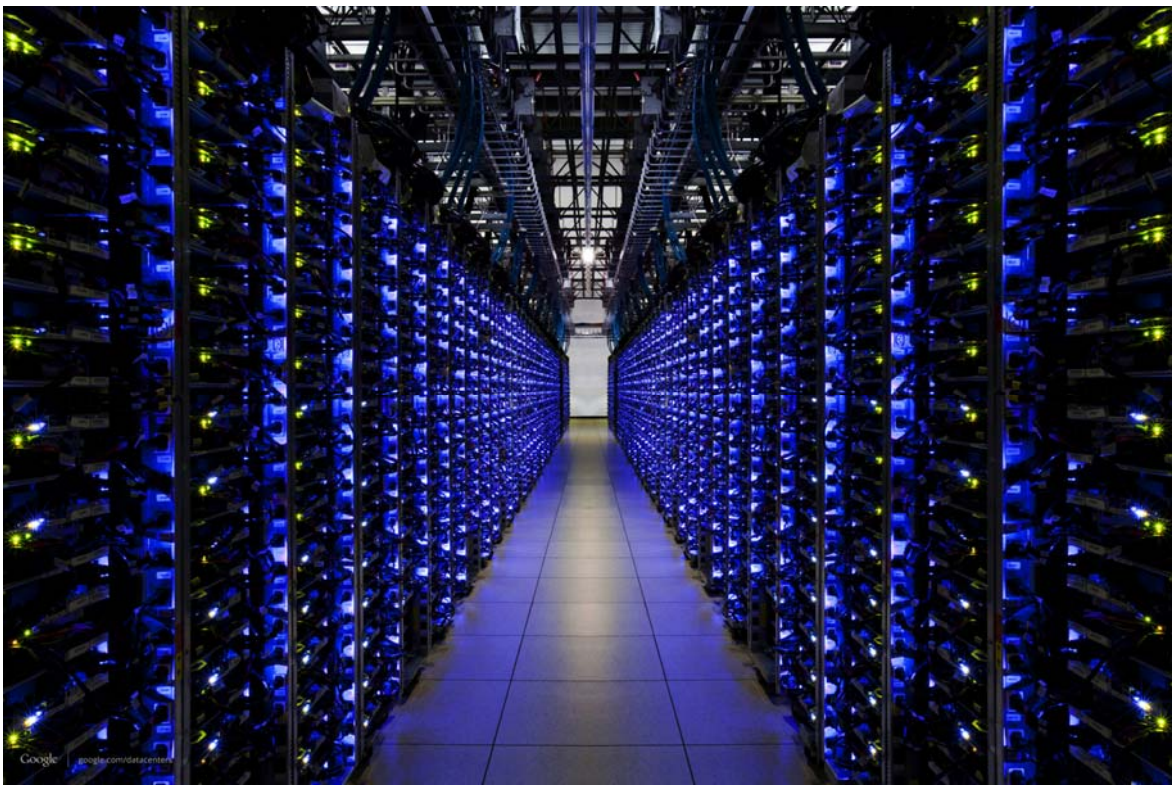
- NoSQL-Datenbanken / Datenbanken für „Big Data“ (seit ca 2009)
  - verteilte Key Value Stores
  - Dokumenten-Stores, z.B. MongoDB
  - Graph-Datenbanken, z.B. Neo4j

- NewSQL-Systeme
  - effizientere Implementierungen relationaler DBS
  - In-Memory-Datenbanken, Column Store statt RecordStore, …

Big Data Challenges



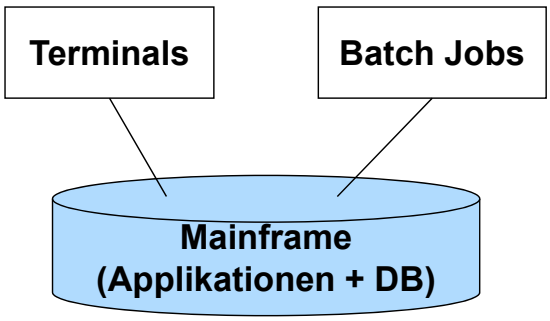
Data Center



Quelle: Google

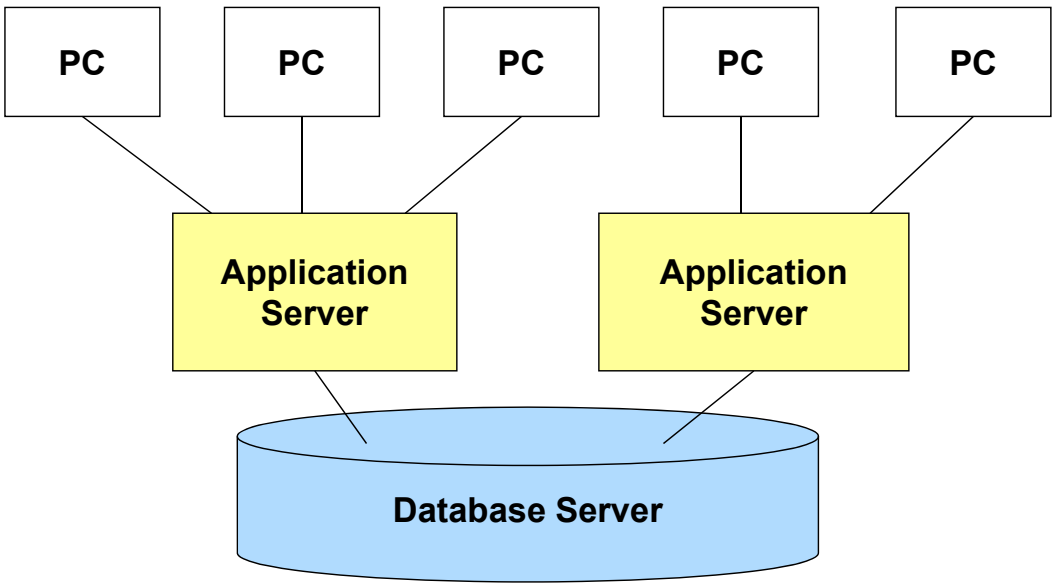
Entwicklung von DBS-Architekturen

- DBS auf Mainframe (1960er/70er Jahre)



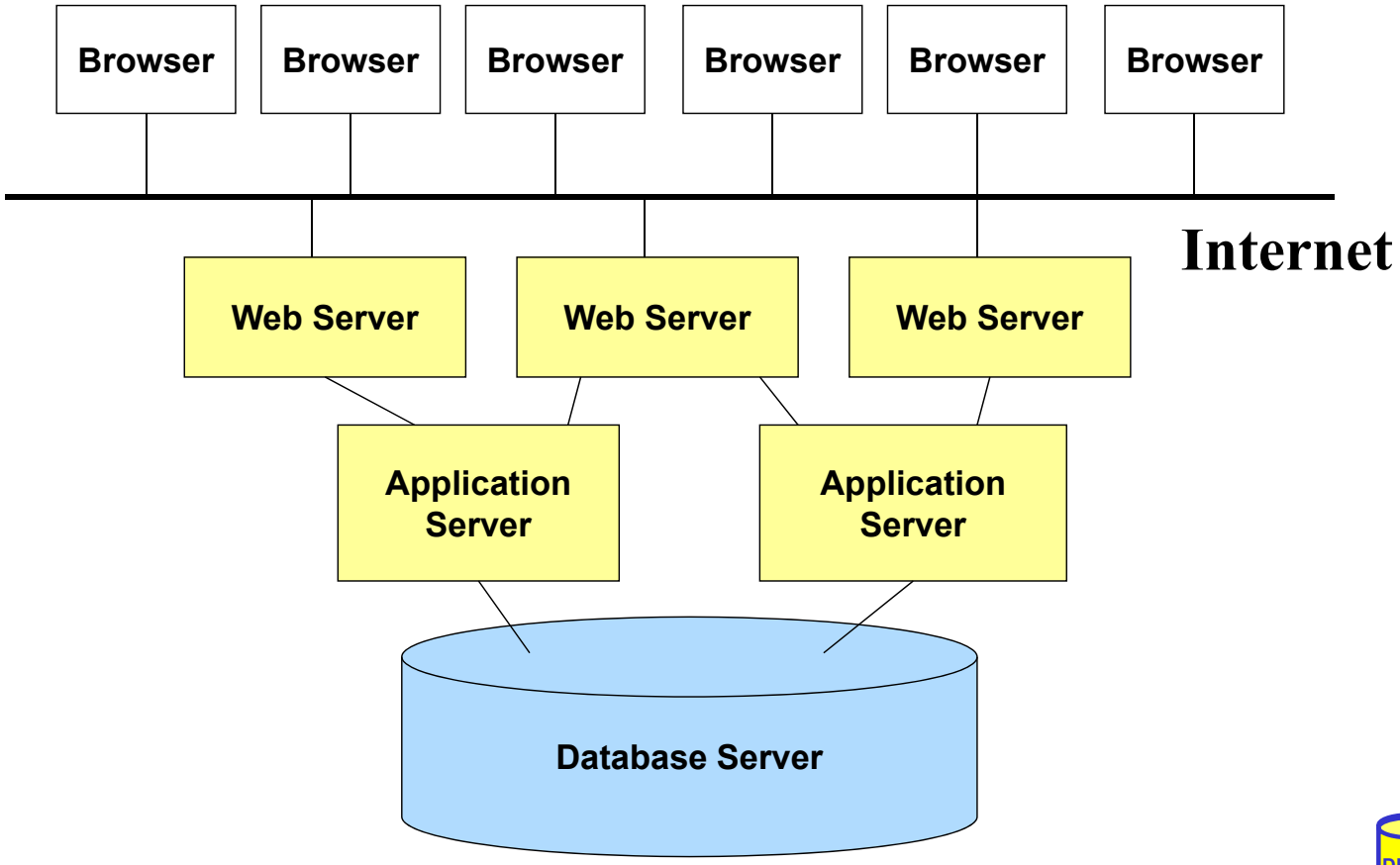
- Client/Server-Systeme
  - 2-stufig (80er Jahre)
  - 3-stufig (90er Jahre)
- Mehrrechner-DBS / Parallele DBS (seit 90er Jahre)
  - mehrere DB-Server
  - Cluster oder ortsverteilt
- Web-Einbindung
- Cloud-Einsatz

Three-Tier Client/Server

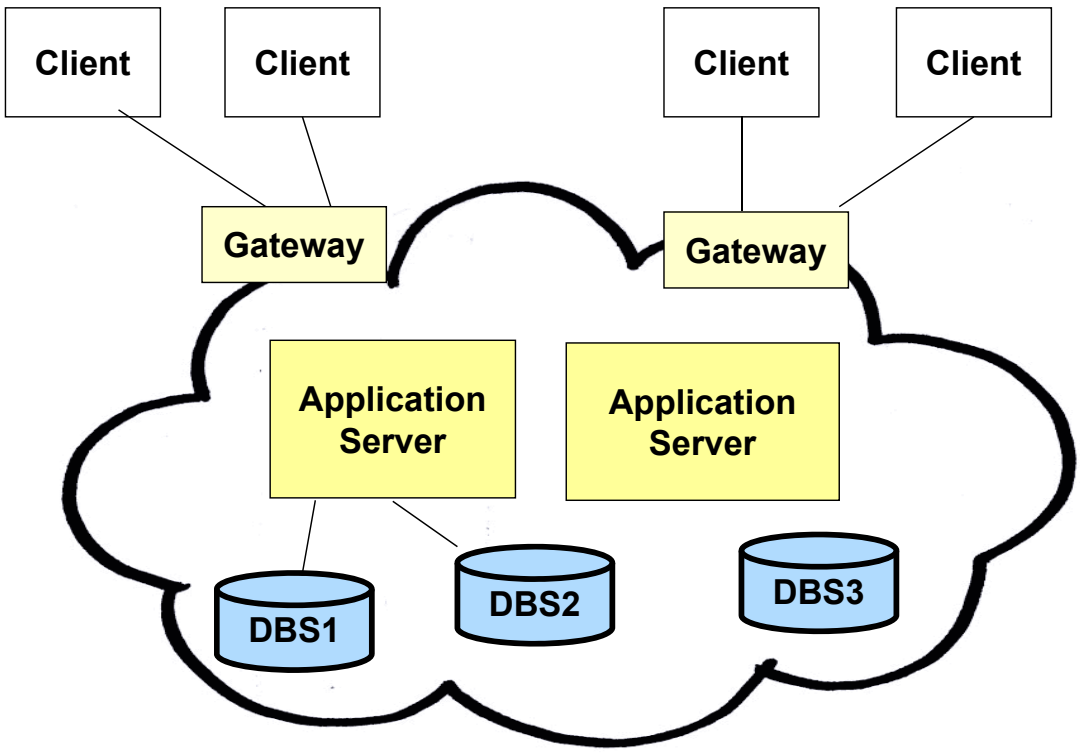


- mehrere Applikationsserver zur Skalierbarkeit auf viele Nutzer
- DB-Zugriffe erfordern Kommunikation zwischen Application und DB Server
- DB-Server kann paralleles DBS in einem Cluster sein

Datenbanken im Web



Cloud-DBS



Zusammenfassung (1)

- 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
- dominierendes Datenmodell: Relationenmodell
  - mengenorientierte DB-Schnittstelle
  - standardisierte Anfragesprache SQL

Zusammenfassung (2)

- Schema-Architektur mit 3 Ebenen:
  externes, konzeptionelles, internes Schema
- Schichtenmodell eines DBVS
  - interne Schichten für Seiten, Sätze und Satzmengen
  - Querschnittsaufgaben: Transaktionsverwaltung und Metadaten
- Haupt-Einsatzformen von DBS in Unternehmen:
  - OLTP-Transaktionsverarbeitung / E-Business
  - Entscheidungsunterstützung: OLAP, Data Mining, maschinelles Lernen
- Architekturvarianten
  - DB-Zugriff über Applikations-Server und Web-Interfaces
  - zentrale DB-Server bzw. Parallele DBS
  - Server im Unternehmen oder extern, z.B. in Cloud