Teil I

Was sind Datenbanken?

Was sind Datenbanken?

1. Überblick & Motivation
2. Architekturen
3. Einsatzgebiete
4. Historisches

Lernziele für heute . . .

- Motivation für den Einsatz von Datenbanksystemen
- Kenntnis grundlegender Architekturen

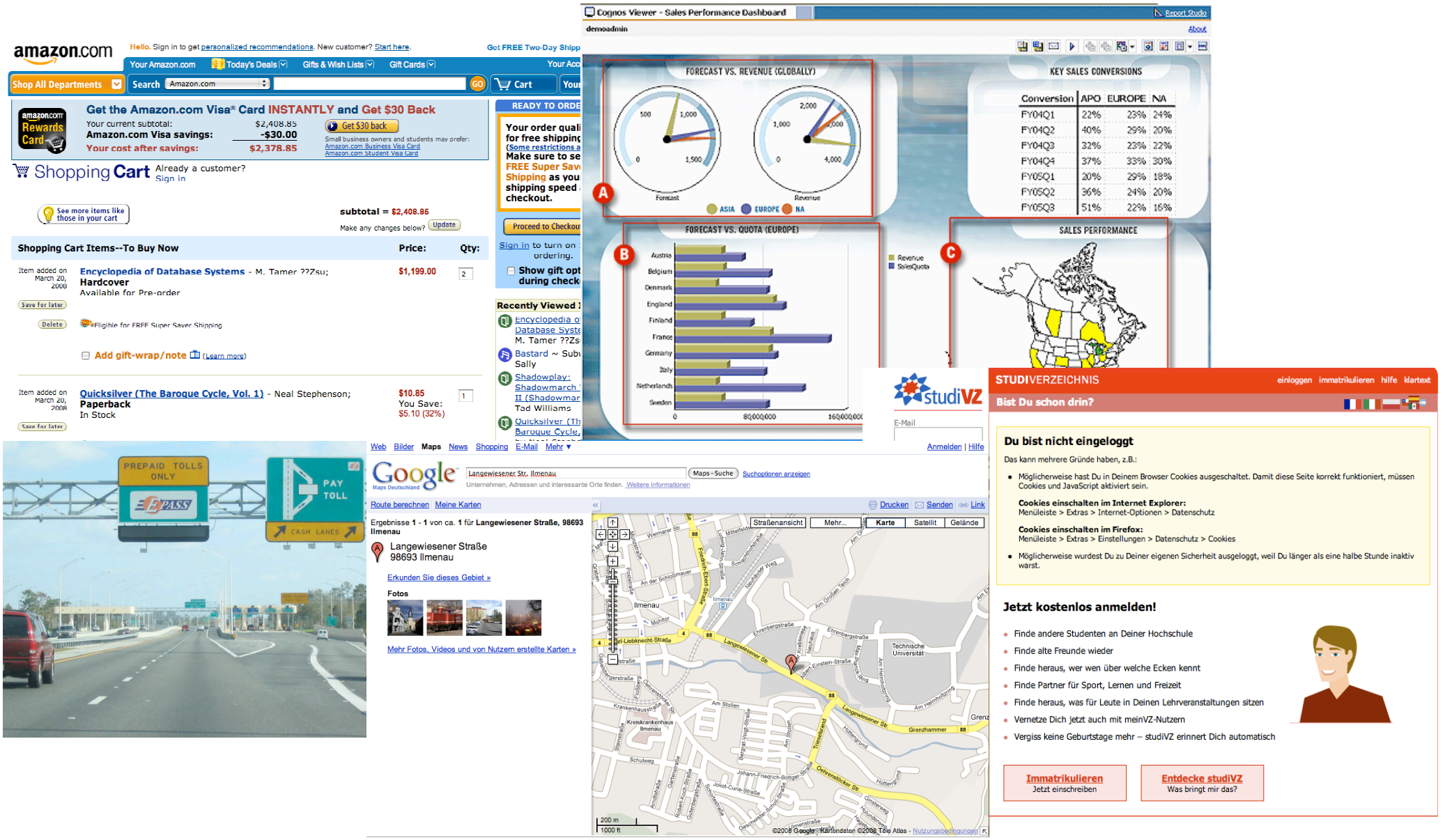
Überblick & Motivation

Was sind Datenbanken?

- Daten = logisch gruppierte Informationseinheiten
- Bank = ...

- Die **Sicherheit vor Verlusten** ist eine Hauptmotivation, etwas „auf die Bank zu bringen“.
- Eine Bank bietet **Dienstleistungen für mehrere Kunden** an, um effizient arbeiten zu können.
- Eine Datenbank hat die (langfristige) **Aufbewahrung** von Daten als Aufgabe.

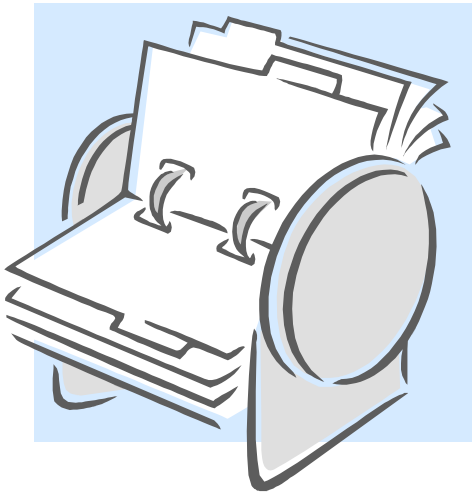
Anwendungsbeispiele

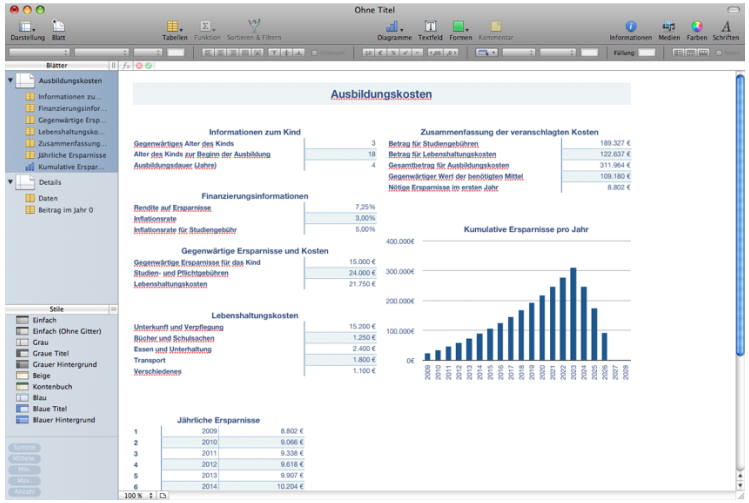


Bedeutung der Datenverwaltung

- Daten als das Öl des 21. Jahrhunderts (Toonders @Wired 2014)
- Ash Ashutosh (CEO Actifio):
  - das größte Hotelunternehmen (AirBnB) hat keine Hotels bzw. Zimmer
  - das größte Taxiunternehmen (Uber) hat keine eigenen Taxis!

Wie verwaltet man Datenbanken?





Ohne Datenbanken

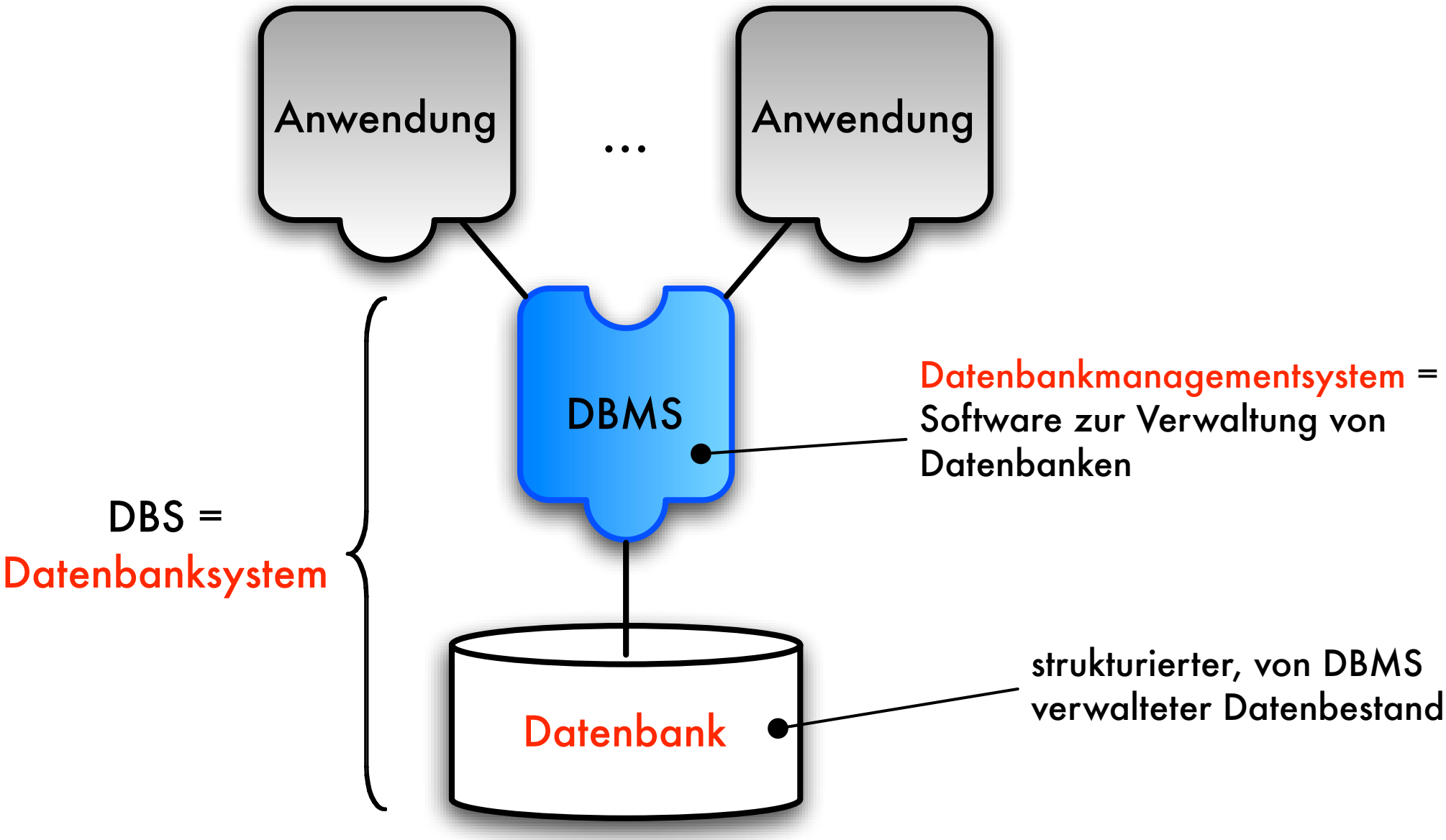
- jedes Anwendungssystem verwaltet seine eigenen Daten
- Daten sind mehrfach gespeichert $\rightsquigarrow$ redundant
- Probleme
  - Verschwendung von Speicherplatz
  - „Vergessen“ von Änderungen
  - keine zentrale, „genormte“ Datenhaltung

Probleme der Datenredundanz

- Andere Softwaresysteme können große Mengen von Daten nicht effizient verarbeiten
- Mehrere Benutzer oder Anwendungen können nicht parallel auf den gleichen Daten arbeiten, ohne sich zu stören
- Anwendungsprogrammierer / Benutzer können Anwendungen nicht programmieren / benutzen, ohne
  - interne Darstellung der Daten
  - Speichermedien oder Rechner

  zu kennen (**Datenunabhängigkeit** nicht gewährleistet)
- Datenschutz und Datensicherheit sind nicht gewährleistet

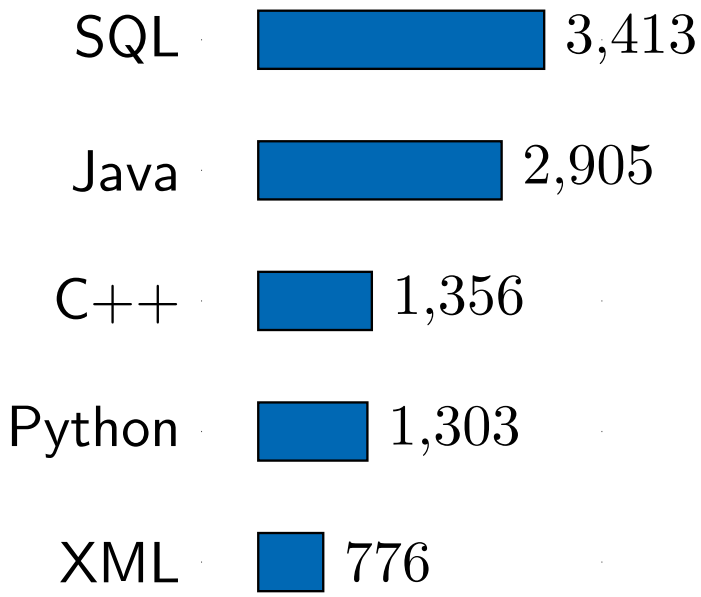
Idee: Datenintegration durch Datenbanksysteme

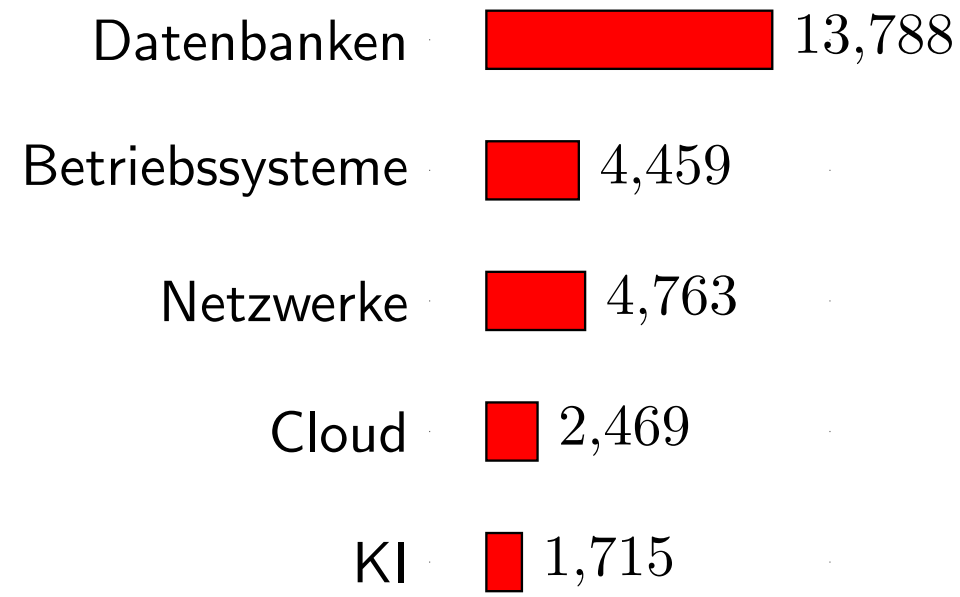


Motivation

- Datenbanksysteme sind Herzstück heutiger IT-Infrastrukturen
- . . . allgegenwärtig
- Datenbank-spezialisten sind gefragt

Motivation: Stellenangebote (Quelle: monster.de 2020)





Fragestellungen

1. **Wie organisiert (modelliert und nutzt) man Daten?**
2. Wie werden Daten dauerhaft verlässlich gespeichert?
3. Wie kann man riesige Datenmengen ($\geq$ Terabytes) effizient verarbeiten?
4. Wie können viele Nutzer ($\geq$ 10.000) gleichzeitig mit den Daten arbeiten?

Architekturen

Prinzipien: Die neun Codd'schen Regeln

1. **Integration**: einheitliche, nichtredundante Datenverwaltung
2. **Operationen**: Speichern, Suchen, Ändern
3. **Katalog**: Zugriffe auf Datenbankbeschreibungen im Data Dictionary
4. **Benutzersichten**
5. **Integritätssicherung**: Korrektheit des Datenbankinhalts
6. **Datenschutz**: Ausschluss unauthorisierter Zugriffe
7. **Transaktionen**: mehrere DB-Operationen als Funktionseinheit
8. **Synchronisation**: parallele Transaktionen koordinieren
9. **Datensicherung**: Wiederherstellung von Daten nach Systemfehlern

Datenunabhängigkeit und Schemata

- Basierend auf DBMS-Grobarchitektur
- Entkopplung von Benutzer- und Implementierungssicht
- Ziele u.a.:
  - Trennung von Modellierungssicht und interner Speicherung
  - Portierbarkeit
  - Tuning vereinfachen
  - standardisierte Schnittstellen

Schemaarchitektur

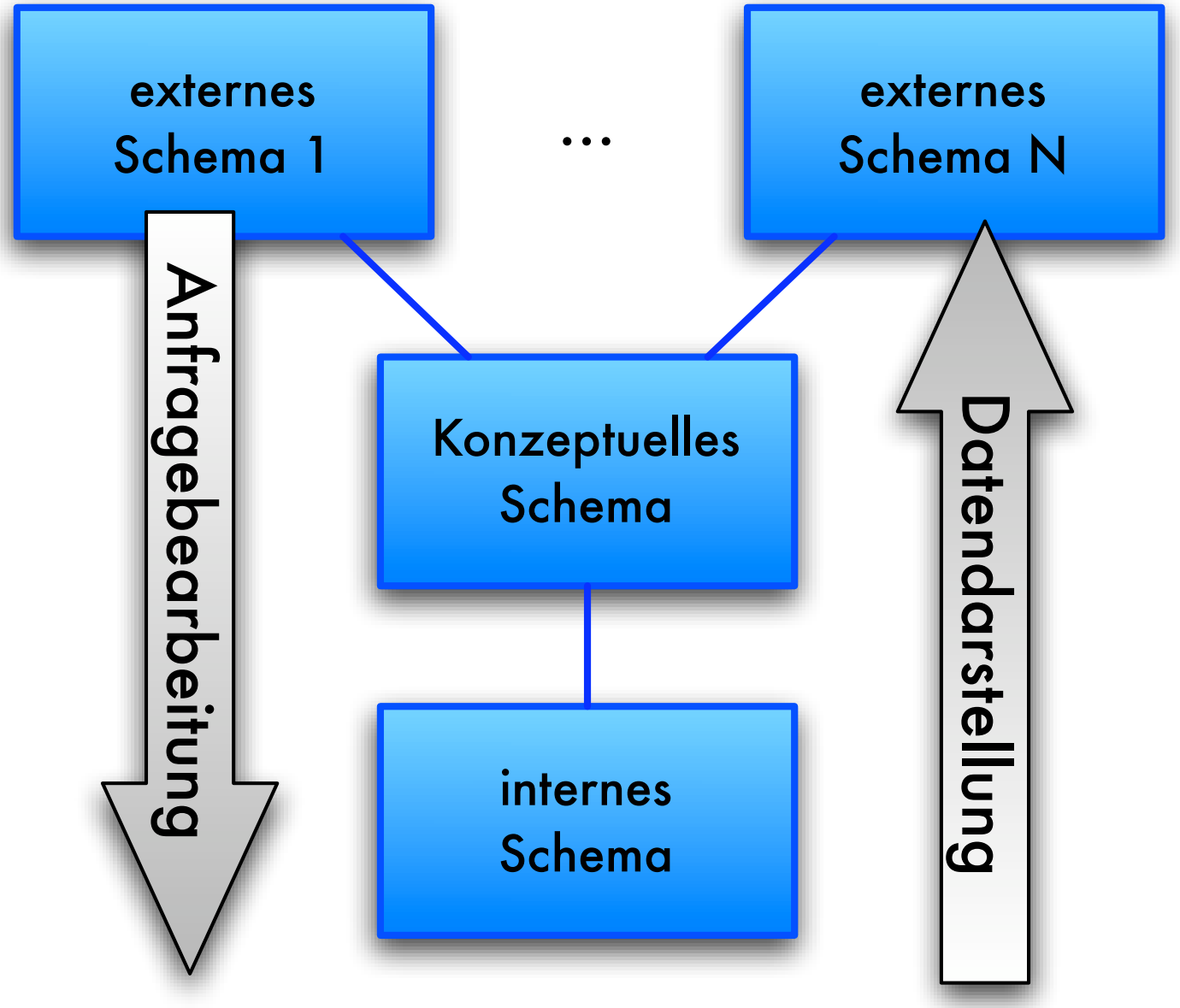
- Zusammenhang zwischen
  - Konzeptuellem Schema (Ergebnis der Datendefinition)
  - Internem Schema (Festlegung der Dateiorganisationen und Zugriffspfade)
  - Externen Schemata (Ergebnis der Sichtdefinition)
  - Anwendungsprogrammen (Ergebnis der Anwendungsprogrammierung)

Schemaarchitektur /2

OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

- Trennung Schema — Instanz
  - Schema (Metadaten, Datenbeschreibungen)
  - Instanz (Anwenderdaten, Datenbankzustand oder -ausprägung)
- Datenbankschema besteht aus
  - internem, konzeptuellem, externen Schemata und den Anwendungsprogrammen
- im konzeptuellen Schema etwa:
  - Strukturbeschreibungen
  - Integritätsbedingungen
  - Autorisierungsregeln (pro Benutzer für erlaubte DB-Zugriffe)

Schemaarchitektur /3



Datenunabhängigkeit /2

OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

- Stabilität der Benutzerschnittstelle gegen Änderungen
- **physisch**: Änderungen der Dateiorganisationen und Zugriffspfade haben keinen Einfluss auf das konzeptuelle Schema
- **logisch**: Änderungen am konzeptuellen und gewissen externen Schemata haben keine Auswirkungen auf andere externe Schemata und Anwendungsprogramme

Datenunabhängigkeit /3

- mögliche Auswirkungen von Änderungen am konzeptuellen Schema:
  - eventuell externe Schemata betroffen (Ändern von Attributen)
  - eventuell Anwendungsprogramme betroffen (Rekompilieren der Anwendungsprogramme, eventuell Änderungen nötig)
- nötige Änderungen werden jedoch vom DBMS erkannt und überwacht

Anwendungsbeispiel: Mitfahrgelegenheit

- Angebote von Mitfahrgelegenheiten
  - Wann? Von wo? Wohin? Wer? Plätze?
  - Kontaktdaten
  - Reservierungsmöglichkeiten

CC-BY-2.0: Luo Shaoyang

Ebenen-Architektur am Beispiel

- Konzeptuelle Sicht: Darstellung in Tabellen (Relationen)

Fahrer

FahrerID	Name	Telefon
103	Lilo Pause	01234
104	Just Vorfan	01246
105	Heiko Heizer	01756

Mitfahrangebot

ANr	Abfahrt	Ankunft	Zeit	FahrerID
1014	Ilmenau	Erfurt	Freitag 10:00	103
1015	Magdeburg	Halle	Freitag 15:00	104
1016	Magdeburg	Leipzig	Freitag 15:00	104
1021	Magdeburg	Ilmenau	Freitag 14:00	105
1025	Ilmenau	Jena	Freitag 10:00	103

Ebenen-Architektur am Beispiel /2

- Externe Sicht: Daten in einer flachen Relation

ANr	**Abfahrt**	**Ankunft**	**Zeit**	**Fahrer**
1014	Ilmenau	Erfurt	Freitag 10:00	Lilo Pause
1015	Magdeburg	Halle	Freitag 15:00	Just Vorfan
1016	Magdeburg	Leipzig	Freitag 15:00	Just Vorfan
1021	Magdeburg	Ilmenau	Freitag 14:00	Heiko Heizer
1025	Ilmenau	Jena	Freitag 10:00	Lilo Pause

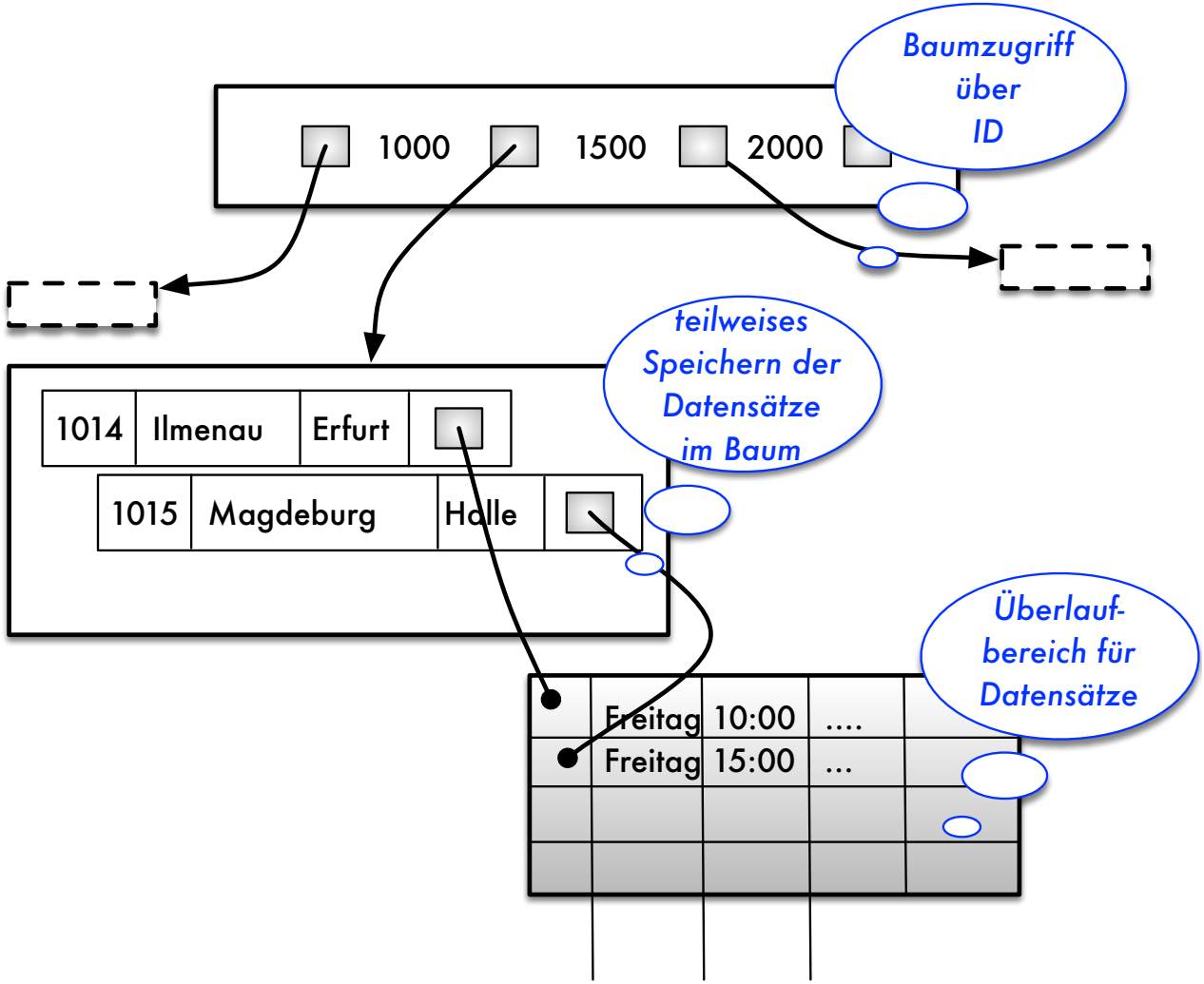
Ebenen-Architektur am Beispiel /3

- Externe Sicht: Daten in einer hierarchisch aufgebauten Relation

Fahrer	**Mitfahrangebot**		
	Abfahrt	**Ankunft**	**Zeit**
Lilo Pause	Ilmenau	Erfurt	Freitag 10:00
		Jena	Freitag 10:00
Just Vorfan	Magdeburg	Halle	Freitag 15:00
		Leipzig	Freitag 15:00
Heiko Heizer	Magdeburg	Ilmenau	Freitag 14:00

Ebenen-Architektur am Beispiel /4

- Interne Darstellung



System-Architekturen

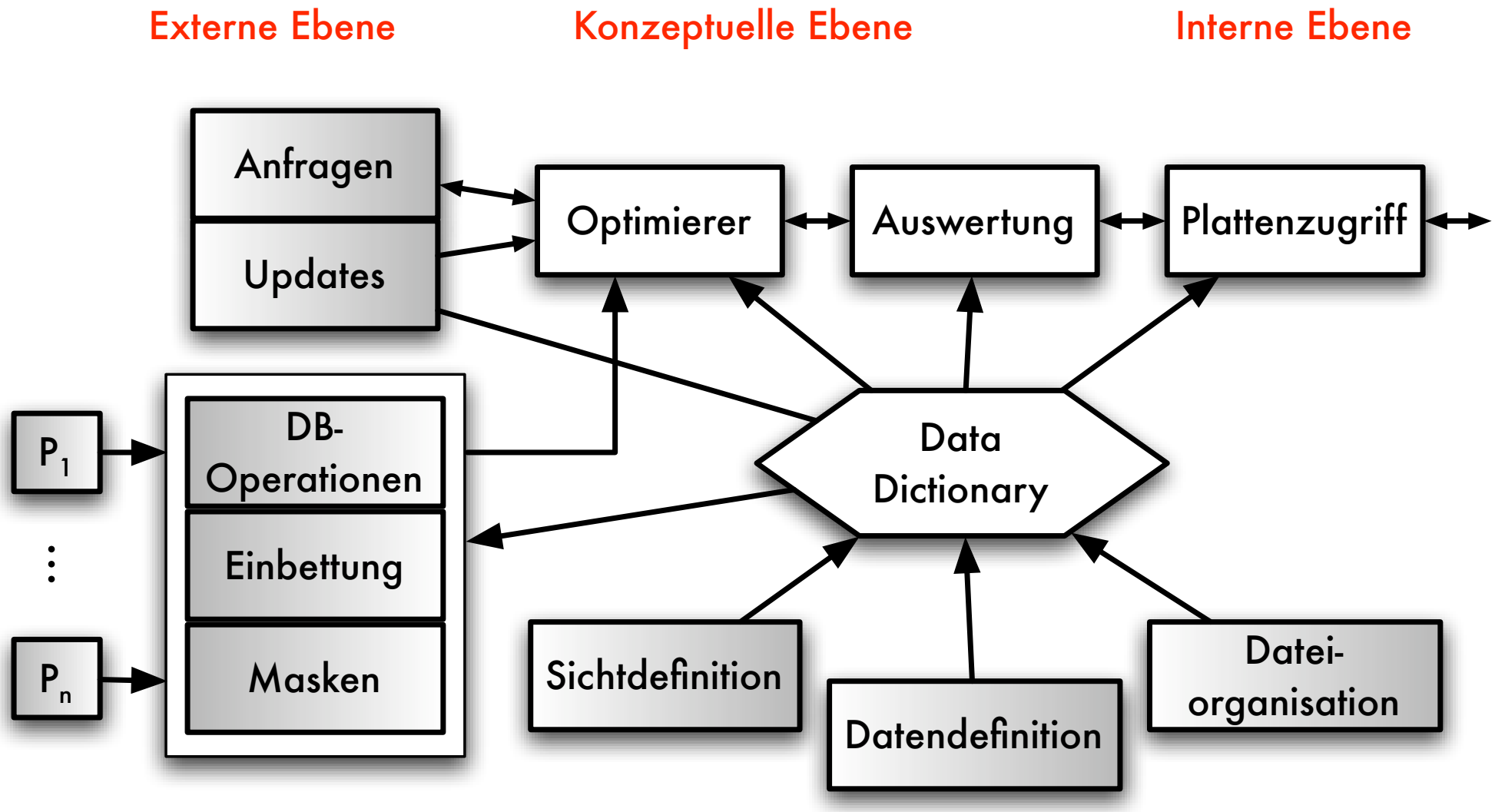
- Beschreibung der Komponenten eines Datenbanksystems
- Standardisierung der Schnittstellen zwischen Komponenten

- Architekturvorschläge
  - ANSI-SPARC-Architektur
    $\rightsquigarrow$ Drei-Ebenen-Architektur
  - Fünf-Schichten-Architektur
    $\rightsquigarrow$ beschreibt Transformationskomponenten im Detail
    Vorlesung „Datenbank-Implementierungstechniken“

ANSI-SPARC-Architektur

- ANSI: American National Standards Institute
- SPARC: Standards Planning and Requirement Committee
- Vorschlag von 1978
- Im Wesentlichen Grobarchitektur verfeinert
  - Interne Ebene / Betriebssystem verfeinert
  - Mehr Interaktive und Programmier-Komponenten
  - Schnittstellen bezeichnet und normiert

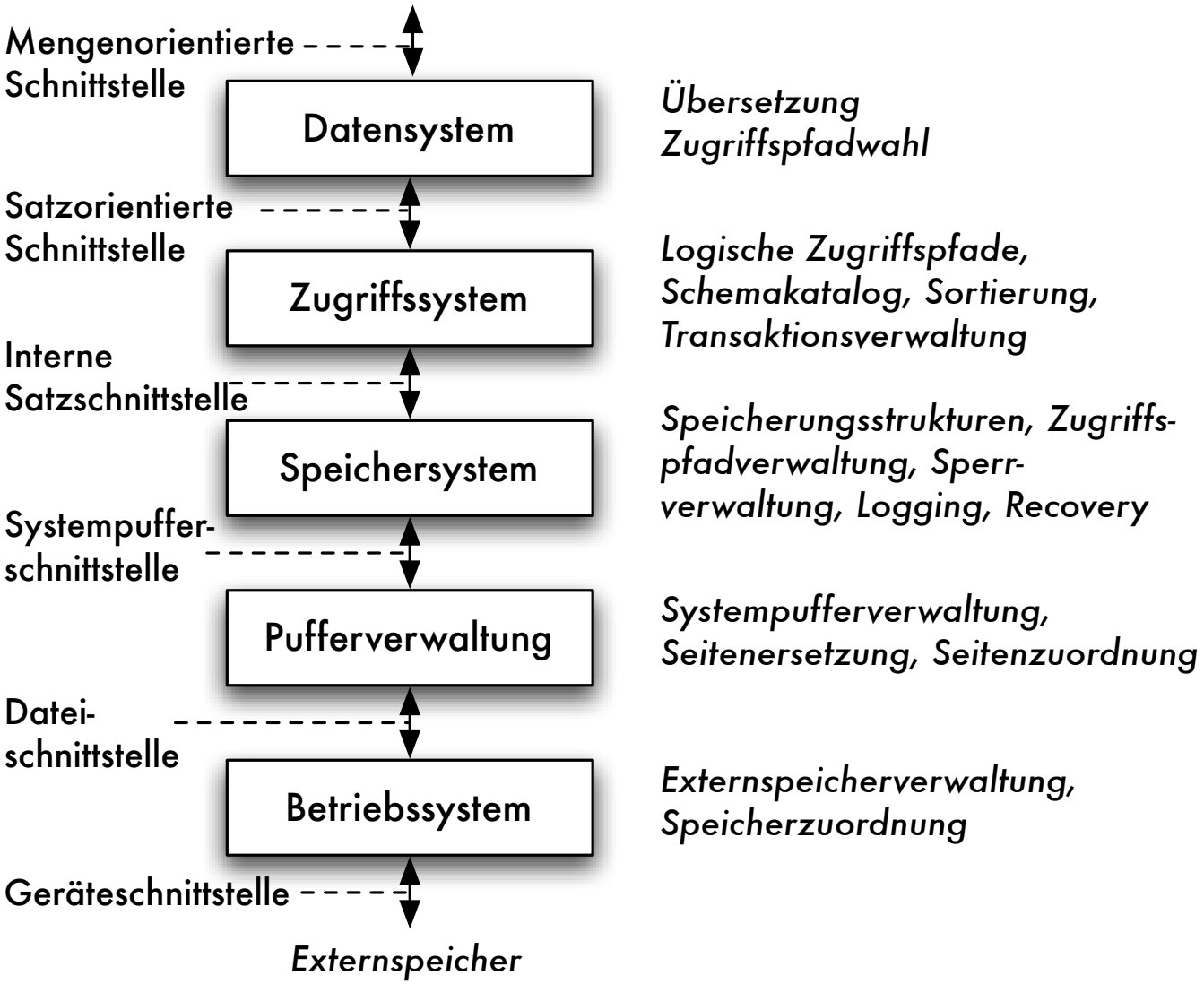
ANSI-SPARC-Architektur /2



Klassifizierung der Komponenten

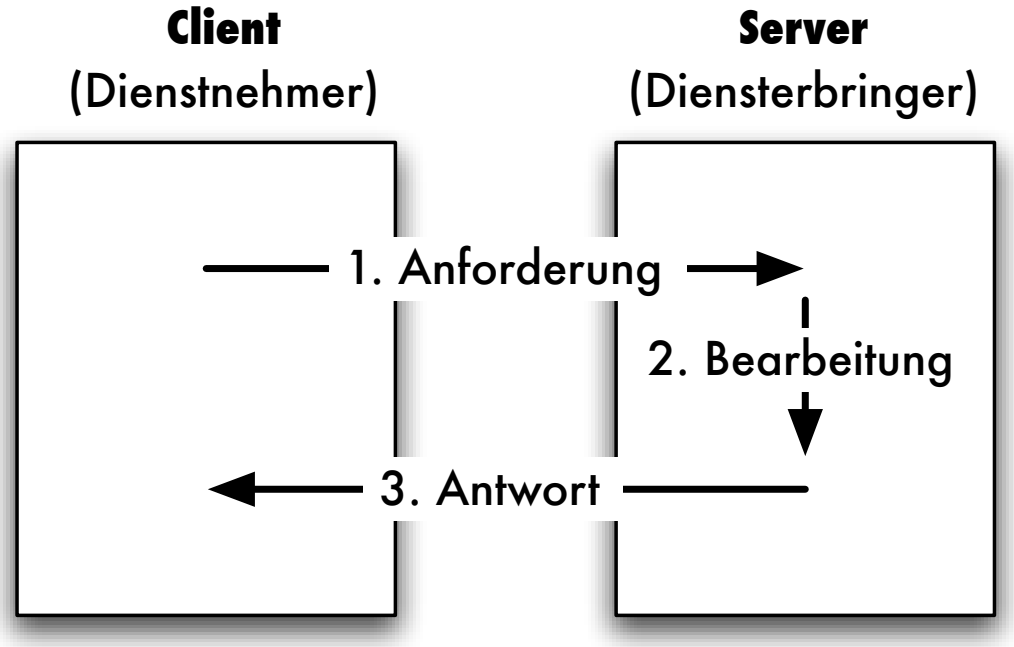
- Definitionskomponenten: Datendefinition, Dateiorganisation, Sichtdefinition
- Programmierkomponenten: DB-Programmierung mit eingebetteten DB-Operationen
- Benutzerkomponenten: Anwendungsprogramme, Anfrage und Update interaktiv
- Transformationskomponenten: Optimierer, Auswertung, Plattenzugriffssteuerung
- Data Dictionary (Datenwörterbuch): Aufnahme der Daten aus Definitionskomponenten, Versorgung der anderen Komponenten

Fünf-Schichten-Architektur: Verfeinerung der Transformation



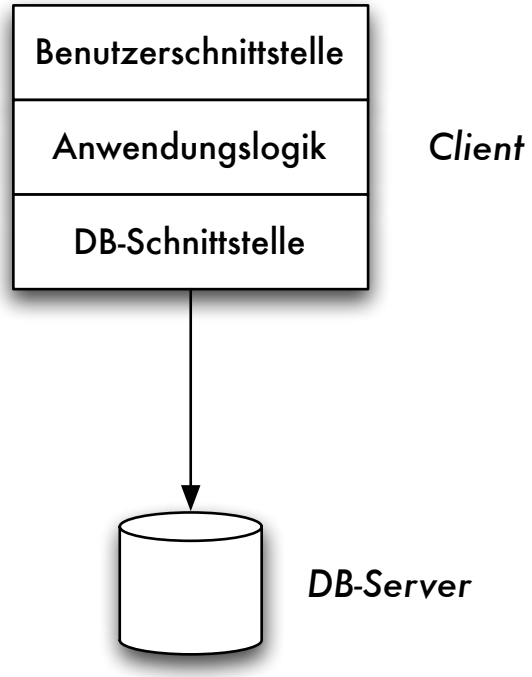
Anwendungsarchitekturen

- Architektur von Datenbankanwendungen typischerweise auf Basis des Client-Server-Modells: Server ≡ Datenbanksystem

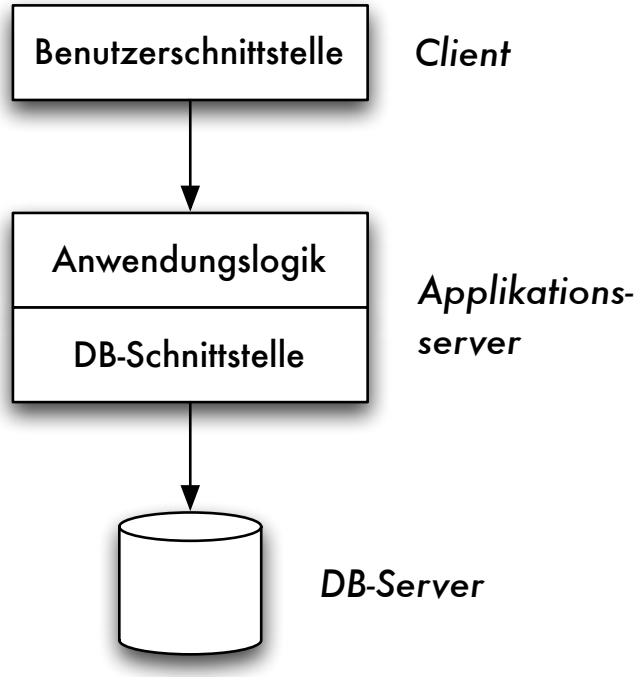


Anwendungsarchitekturen /2

- Aufteilung der Funktionalitäten einer Anwendung
  - Präsentation und Benutzerinteraktion
  - Anwendungslogik („Business“-Logik)
  - Datenmanagementfunktionen (Speichern, Anfragen, . . .).



Zwei-Schichten-Architektur



Drei-Schichten-Architektur

Einsatzgebiete

Einige konkrete Systeme

- (Objekt-)Relationale DBMS
  - Oracle21c, IBM DB2 UDB V.11, Microsoft SQL Server 2017, SAP HANA
  - MySQL (`www.mysql.org`) & Friends (MariaDB), PostgreSQL (`www.postgresql.org`)
  - Amazon Aurora, Amazon Redshift, Microsoft Azure SQL, ...
- Pseudo-DBMS
  - MS Access
- NoSQL-Systeme
  - Graph-Datenbanksysteme (InfiniteGraph, neo4j), Dokument-Datenbanken (MongoDB), Key-Value-Stores, ...

Einsatzgebiete

- Klassische Einsatzgebiete:
  - viele Objekte (100.000 Bücher, 10.000 Benutzer, 1000 Ausleihvorgänge pro Tag, . . .)
  - wenige Objekttypen (BUCH, BENUTZER, AUSLEIHUNG)
  - etwa Buchhaltungs- bzw. ERP-Systeme, Bestellsysteme, Bibliothekssysteme, . . .
- Aktuelle Anwendungen:
  - E-Commerce, Buchung- und Abrechnungssysteme, entscheidungsunterstützende Systeme (Data Warehouses, OLAP), wiss. Anwendungen (Erdbeobachtung, Klima), . . .

Datenbankgrößen

Kommerzielle Systeme, Zahlen aus 2010

eBay Data Warehouse 10 PB ($\approx 10 \cdot 10^{15}$ Bytes)
Teradata DBMS, 72 Knoten, 10.000 Nutzer,
mehrere Millionen Anfragen/Tag

WalMart Data Warehouse 2,5 PB
Teradata DBMS, NCR MPP-Hardware;
Produktinfos (Verkäufe etc.) von 2.900 Märkten;
50.000 Anfragen/Woche

OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

Historisches

Entwicklungslinien: 60er Jahre

- Anfang 60er Jahre: elementare Dateien, anwendungsspezifische Datenorganisation (geräteabhängig, redundant, inkonsistent)
- Ende 60er Jahre: Dateiverwaltungssysteme (SAM, ISAM) mit Dienstprogrammen (Sortieren) (geräteunabhängig, aber redundant und inkonsistent)
- DBS basierend auf **hierarchischem Modell**, **Netzwerkmodell**
  - Zeigerstrukturen zwischen Daten
  - Schwache Trennung interne / konzeptuelle Ebene
  - Navigierende DML
  - Trennung DML / Programmiersprache

Entwicklungslinien: 70er und 80er Jahre

- 70er Jahre: Datenbanksysteme (Geräte- und Datenunabhängigkeit, redundanzfrei, konsistent)
- **Relationale Datenbanksysteme**
  - Daten in Tabellenstrukturen
  - 3-Ebenen-Konzept
  - Deklarative DML
  - Trennung DML / Programmiersprache

Historie von RDBMS

- 1970: Ted Codd (IBM) → Relationenmodell als konzeptionelle Grundlage relationaler DBS
- 1974: System R (IBM) → erster Prototyp eines RDBMS
  - zwei Module: RDS, RSS; ca. 80.000 LOC (PL/1, PL/S, Assembler), ca. 1,2 MB Codegröße
  - Anfragesprache SEQUEL
  - erste Installation 1977
- 1975: University of California at Berkeley (UCB) → Ingres
  - Anfragesprache QUEL
  - Vorgänger von Postgres, Sybase, ...
- 1979: Oracle Version 2

Entwicklungslinien: (80er und) 90er Jahre

- **Wissensbanksysteme**
  - Daten in Tabellenstrukturen
  - Stark deklarative DML, integrierte Datenbankprogrammiersprache
- **Objektorientierte Datenbanksysteme**
  - Daten in komplexeren Objektstrukturen (Trennung Objekt und seine Daten)
  - Deklarative oder navigierende DML
  - Oft integrierte Datenbankprogrammiersprache
  - Oft keine vollständige Ebenentrennung

Entwicklungslinien: heute

- Neue Hardwarearchitekturen
  - Multicore-Prozessoren, Hauptspeicher im TB-Bereich: **In-Memory-Datenbanksysteme** (z.B. SAP HANA)
  - Hardwarebeschleunigung durch FPGA oder GPU
- Unterstützung für spezielle Anwendungen
  - **Cloud-Datenbanken**: Hosting von Datenbanken, Skalierbare Datenmanagementlösungen (Amazon RDS, Microsoft Azure)
  - **Datenstromverarbeitung**: Online-Verarbeitung von Live-Daten, z.B. Börseninfos, Sensordaten, RFID-Daten, ... (StreamBase, MS StreamInsight, IBM Infosphere Streams)
  - **Big Data**: Umgang mit Datenmengen im PB-Bereich durch hochskalierbare, parallele Verarbeitung, Datenanalyse (Hadoop, Hive, Google Spanner & F1, ...)

Entwicklungslinien: NoSQL

- **NoSQL-Datenbanken** („Not only SQL“):
  - nicht-relationale Datenbanken, flexibles Schema (dokumentenzentriert)
  - „leichtgewichtig“ durch Weglassen von SQL-Funktionalitäten wie Transaktionen, mächtige deklarative Anfragesprachen mit Verbunden etc.
  - Beispiele: CouchDB, MongoDB, Cassandra, ...

Trends

- Nutzergenerierte Inhalte, z.B. Google:
  - Verarbeitung von 20 PB täglich
  - 15h Video-Upload auf YouTube in jeder Minute
  - Lesen von 20 PB würde 12 Jahre benötigen bei 50 MB/s-Festplatte
- Linked Data und Data Web
  - Bereitstellung, Austausch und Verknüpfung von strukturierten Daten im Web
  - ermöglicht Abfrage (mit Anfragesprachen wie SPARQL) und Weiterverarbeitung
  - Beispiele: DBpedia, GeoNames

Zusammenfassung

- Motivation für Einsatz von Datenbanksystemen
- Codd'sche Regeln
- 3-Ebenen-Schemaarchitektur & Datenunabhängigkeit
- Einsatzgebiete

Kontrollfragen

- Welchen Vorteil bieten Datenbanksysteme gegenüber einer anwendungsspezifischen Speicherung von Daten?
- Was versteht man unter Datenunabhängigkeit und wie wird sie erreicht?
- In welchen Bereichen kommen Datenbanksysteme zum Einsatz?