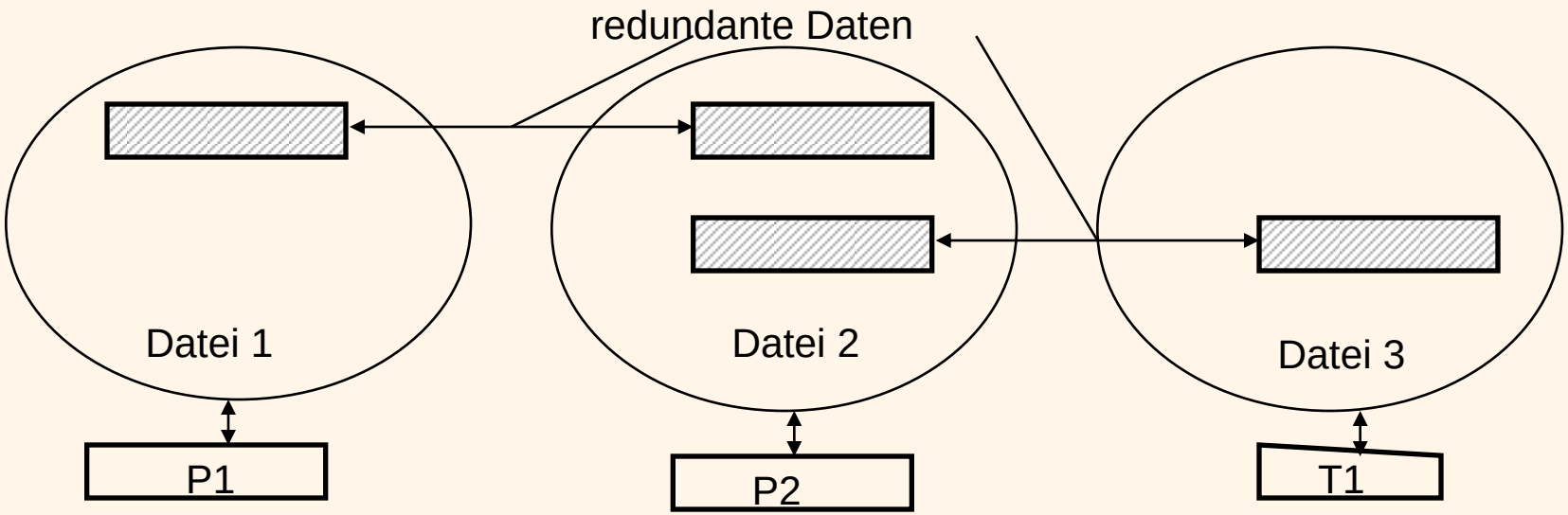
Einführung

Applikationen mit Dateien



- einfache Dateioperationen
  - zur Verwaltung: create/drop, open/close
  - zum Zugriff: read/write
- Verschiedene Formen der Dateiorganisation
  - Direkter Zugriff (relativ)
  - Wertbasierter (assoziativer) Zugriff (hash, key)
  - Unstrukturierte Dateien (byte stream)
- Synchronisation
  - Kommunikation notwendig für Änderungen

Nachteile von Dateisystemen

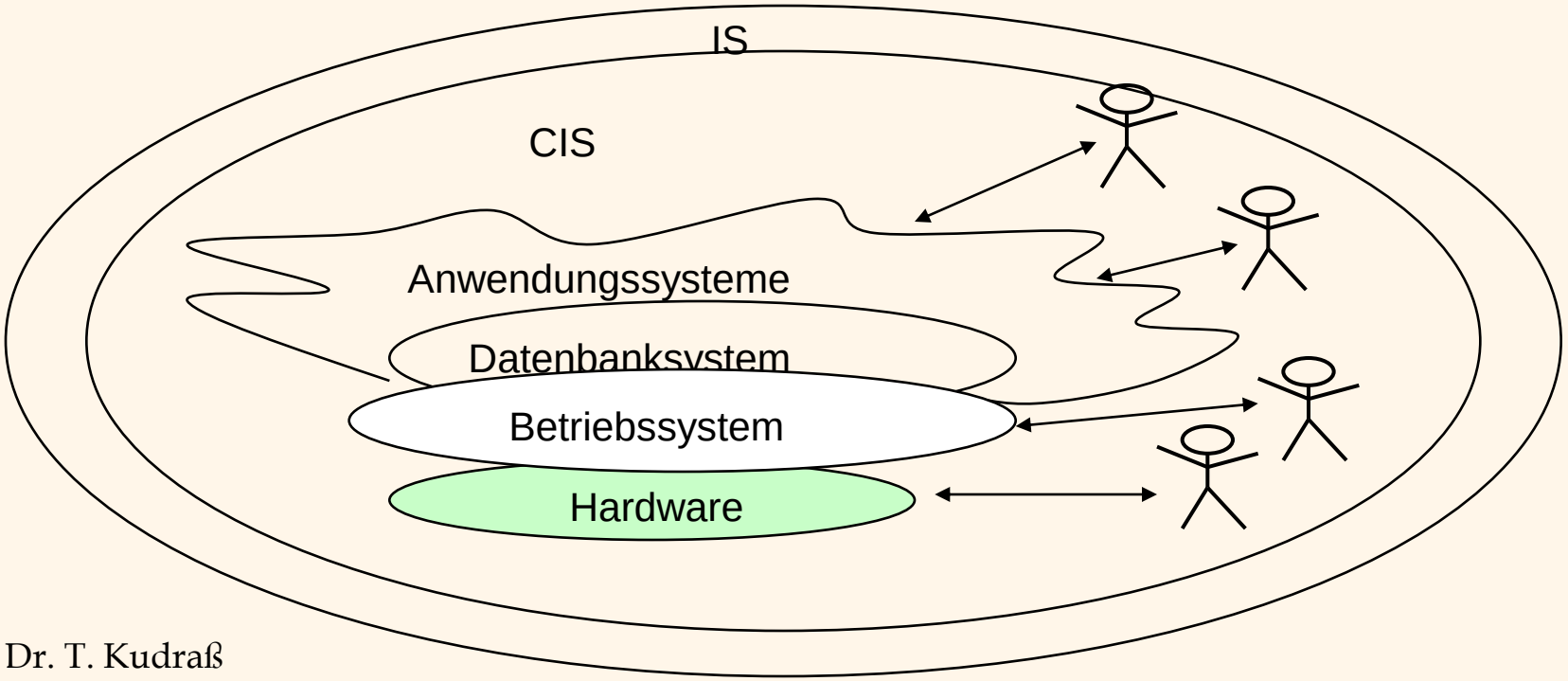
- Wiederholte Speicherung gleicher Daten (Redundanz)
  - Integritätsprobleme
  - erhöhter Speicherplatzbedarf
- Verantwortung des Programmierers für
  - Datenintegrität
  - Datensicherheit
  - effizienten Zugriff
- Bindung von Datenstrukturen an Programmstrukturen (hoher Änderungsaufwand)
- Lösung gleicher Aufgaben in einem Anwendungsprogramm
  - Speicherverwaltung
  - Datenverwaltung (Ändern und Retrieval)
  - Schutzfunktionen
- Annahmen:
  - Alles bleibt stabil!

Grundbegriffe

- Datenbank (DB)
  - eine sehr große integrierte Sammlung von Daten
  - Beschreibt einen Ausschnitt aus der realen Welt:
    - Entitäten (z.B. Studenten, Kurse)
    - Beziehungen (z.B. Professor hält Kurs)
- Datenbank-Management-System (DBMS)
  - Software-Paket zum Speichern und Verarbeiten von Datenbanken (Einfügen, Lesen, Ändern und Löschen von Daten)
  - Beispiele: Oracle, DB2 (IBM), MS SQL Server
- Datenbank-System (DBS)
  - Ermöglicht die anwendungsübergreifende Nutzung von Daten
  - Isoliert Anwendungsprogramme von Hardware und Betriebssystem (und deren Änderungen)
  - DBS = DBMS + DB
- Datenmodell (DM)
  - Struktur
  - Operationen
  - Konsistenzregeln der Daten

Informationssystem vs. Datenbanksystem

- Ein Informationssystem (IS) besteht aus Menschen und Maschinen, die Informationen erzeugen und/oder benutzen und die durch Kommunikationsbeziehungen miteinander verbunden sind.
- Ein rechnergestütztes Informationssystem (CIS) ist ein System, bei dem die Erfassung, Speicherung und/oder Transformation von Informationen durch den Einsatz von EDV teilweise automatisiert ist.

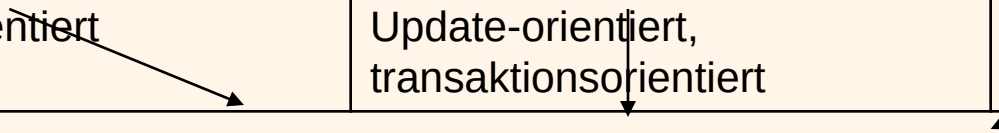


Beispiele für Informationssysteme

- Universitätsdatenbank
  - Objekte: Fachbereiche, Studenten, Professoren, Mitarbeiter, Vorlesungen, Prüfungen
  - Anwendungen:
    - Immatrikulation
    - Ausfertigung von Studienbescheinigungen
    - Stundenplanerstellung
    - Raumbelegung
    - Ausstellung von Zeugnissen
    - Statistiken
- Bank-Informationssystem
  - Objekte: Partner (Kunden, Geschäftspartner), Produkte (bestehend aus Features), Tarife, Standardkosten, Konten, Finanzinstrumente, Geschäftsprozesse (Beschreibungen und Logs), Referenzdaten (z.B. Kalender)
  - Anwendungen:
    - Buchung von Zahlungsvorgängen auf verschiedenen Konten
    - Einrichten und Auflösen von Konten
    - Zinsberechnung und Verbuchung
    - Personalverwaltung (Gehaltsabrechnung)
    - Bereitstellung von Statistiken über Kundenverhalten zu Marketing-Zwecken

Typen von Informationssystemen

Information Retrieval	Kommerzielle EDV	Wiss.-techn. Anwendungen
große Datenmengen (unformatiert)	große Datenmengen (formatiert) Einfache Daten-struktur/Datentypen	Kleine Datenmengen, numerische Daten
Komplexe Suchalgorithmen	Einfache Algorithmen	Komplexe Algorithmen
Retrieval-orientiert	Update-orientiert, transaktionsorientiert	prozeßorientiert



Datenbankmanagementsysteme

Non-Standard-Anwendungen
große Datenmengen (formatiert und unformatiert)
Komplexe Datentypen
Komplexe Algorithmen
Komplexe Transaktionen

Entwicklung der Datenbanken

- Bis 1960
  - Daten in nicht-magnetischen Medien (z.B. Lochkarten) gespeichert
  - Daten auf sequentiellen magnetischen Medien (Magnetband)
- 1960-65
  - Plattenspeicher und Trommeln: Direktzugriff auf Daten über deren Adressen
  - große Datenmengen weiter auf Magnetband
  - Daten werden als anwendungsspezifische Dateien gespeichert, auch Zugriffsmechanismen in Anwendungen integriert
- 1965-70
  - Erstes DBMS von Bachman: Integrated Data Store (IDS), Netzwerk-Datenmodell, CODASYL-Standard
  - IBM: Information Management System (IMS), hierarch. Datenmodell
- 1970-80
  - Datenverwaltungssysteme mit Data Dictionary zur Kontrolle von Redundanz, Sichten und Benutzer
- 1980-
  - Relationale DBMS dominieren (Relationenmodell von Codd, 1970): Oracle, DB2, Informix, Sybase
  - SQL wird zur Standard-Anfragesprache (aus IBM System R hervorgegangen)

Entwicklung der Datenbanken (Forts.)

- 1985-1995
  - Objektorientierte Datenbanksysteme (z.B. ObjectStore)
  - Erweiterbare Datenbanksysteme mit neuen Datentypen für multimediale Anwendungen (Image, Text, Video): z.B. Oracle 8, Informix UDS
  - Mächtigere Query-Languages (Erweiterungen von SQL)
- 1995-
  - Generische anwendungsorientierte Schichten on top of a DBMS (z.B. SAP, Oracle), die applikationsspezifisch angepaßt werden können:
    - Inventory Management
    - Human Resource Management
  - Data Warehouse Systeme (z.B. SAS)
  - Anbindung von Datenbanken ans Internet
- 2005-
  - Big Data: Steigende Datenmengen in Vielfalt und Umfang $\rightarrow$ wachsende Anforderungen an DBMS
  - Datenbanken im Web 2.0 (NoSQL-Datenbanken, Semantic Web)
  - Scientific Databases (z.B. Genom-Datenbanken)
  - Embedded Databases
  - Data Management on New Hardware (Flash-Speicher, Hauptspeicher-DB)
  - Konvergenz mit Information Retrieval (Information Extraction)

Vorteile eines DBMS

- Datenunabhängigkeit
- Effizienter Zugriff
- Verminderte Entwicklungszeit
- Kontrolle der Datenintegrität
- Zugriffskontrolle auf die Daten
- Datensicherheit und Zugriffskontrolle auf die Daten (Korrektheit bei fehlerhaftem Ablauf einzelner Anwendungen und System-absturz)
- Einheitliche Datenadministration
- Unterstützung von Nebenläufigkeit (Concurrency Control)
- Recovery-Fähigkeiten (Korrektheit bei fehlerhaftem Ablauf einzelner Anwendungen und Wiederherstellung der DB nach System-Crash)

Datenmodelle

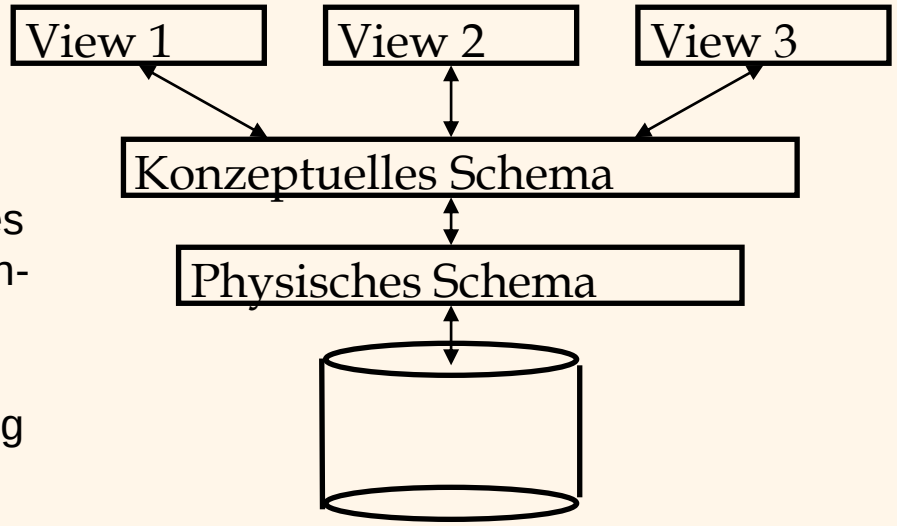
- Daten haben
  - Struktur, die im Typ definiert wird (Intension)
  - Werte oder Instanzen (Extension)
- Datenmodell
  - Menge von Konstrukten zur Beschreibung von Daten
  - Low-Level Details der Datenspeicherung werden abstrahiert
  - Ein DBMS erlaubt die Definition von Daten in einem bestimmten Datenmodell (meist relational)
- Semantisches Datenmodell
  - Abstraktere Beschreibung von Daten unab-hängig von einem konkreten DBMS
  - Beispiele: Entity-Relationship-Modell, Objektmodell (UML)
- Schema
  - Beschreibung einer Menge von Daten mit Hilfe eines bestimmten Datenmodells
- Relationales Datenmodell
  - Basiskonstrukt: Relation, d.h. eine Tabelle mit Zeilen und Spalten
  - Jede Relation hat ein Schema zur Beschreibung der Spalten (oder Felder)

Abstraktionsebenen

- Mehrere *Sichten (Views)*, ein *konzeptuelles (logisches) Schema* und *physisches Schema*.
- Views:

  Anwendungsspezifische Ausschnitte des konzeptuellen Schemas zur Filterung un-nötiger Daten oder zum Schutz vor nichtautorisiertem Zugriff. Auch zur anwendungsspezi-fischen Strukturierung verwendet
- Konzeptuelles Schema: Beschreibt eine integrierte logische Sicht des gesamten Datenbestandes ohne Details über Speicherstrukturen und Zugriffspfade
- Physisches Schema:

  Beschreibt die Daten in Form von Datensätzen (Records), spezifischen Zugriffspfaden, die Abbildung der logischen Records auf die Speicherstrukturen (physische Records, Seiten)



Schemas werden in DDL definiert

Beispiel: Datenbank *Hochschule*

- Konzeptuelles Schema
  - *Studenten (sid: string, name: string, login: string, alter: integer)*
  - *Kurse (kid: string, kname: string, stunden: integer)*
  - *Fachbereich (fid: string, fname: string, budget: real)*
  - *Lehrt (fid: string, kid: string)*
  - *Eingeschrieben (sid: string, cid: string, grade:string)*
- Physisches Schema
  - Speicherung der Relationen als Files: unsortierte Menge von physischen Records
  - Index auf der ersten Spalte von *Studenten* und *Kurse* zur Beschleunigung des Datenzugriffs
- Externes Schema (View)
  - Wieviele Studenten haben sich in jedem Kurs eingeschrieben?
  - *Kurs_Info (kid: string, einschreibanz: integer)*

Datenunabhängigkeit

- Applikationen sind isoliert davon, wie Daten strukturiert und gespeichert werden.
- Logische Datenunabhängigkeit

  Änderungen in der logischen Struktur der Daten sind für Applikationen und ad-hoc Queries irrelevant

  Beispiel: Aufsplitten der Relation *Fachbereich*:

  Fachbereich_public (fid: string, fname: string, büro: integer)

  Fachbereich_private (fid: string, budget: real)

  Benutzer der Query *Kurs_Info* sind davon nicht betroffen
- Physische Datenunabhängigkeit

  Änderungen an den Speicherstrukturen und Zugriffspfaden sind für das konzeptuelle Schema irrelevant

  Es gibt verschiedene Arten von Unabhängigkeit:

  Zugriffspfad

  Datenstruktur

  Speicherungsstruktur

  Seitenzuordnungsstruktur

  Gerät

Concurrency Control

- Nebenläufige Ausführung von Anwendungsprogrammen wichtig für Performance
  - –Nebenläufige Arbeit wichtig für gute CPU-Auslastung
- Überlappende Aktionen von verschiedenen Programmen können zu Inkonsistenz führen

  z.B. Berechnung eines Kontostandes durch eine Transaktion und gleichzeitige Ausführung einer Überweisung
- DBMS garantiert Isolation und Konsistenz
- Isolation

  Illusion einer Transaktion, allein Zugriff auf die Datenbank zu haben. Eine Transaktion sieht nur einen konsistenten Zustand der Datenbank.
- Konsistenz

  Korrekter Ablauf einer Transaktion

Transaktionskonzept

- Transaktion
  - Atomare Sequenz von Datenbank-Aktionen (read / write)
- Atomizität = All or Nothing
- Jede Transaktion hinterläßt die DB in einem konsistenten Zustand, wenn diese bei Transaktionsbeginn schon konsistent war.
  - Benutzer können Integritätsbedingungen auf den Daten formulieren, die vom DBMS kontrolliert werden.
  - Das DBMS “versteht“ nicht die Semantik der Daten $\rightarrow$ Benutzer ist für den korrekten Ablauf einer Transaktionen verantwortlich
  - DBMS garantiert, daß die verschachtelte Ausführung einer Menge von Transaktionen {T1, ..., Tn} äquivalent zu irgendeiner seriellen Ausführung T1... Tn sind.

Datensicherheit

- DBMS sichert Atomizität auch, wenn das System mitten in einer Transaktion abstürzt
- Idee: Protokollierung aller Aktionen (Logging), die vom DBMS ausgeführt werden während der Ausführung der Transaktionen
  - Vor dem Ändern der DB wird der entsprechende Log-Eintrag auf einen sicheren Platz geschrieben.
  - Nach einem Crash werden die Effekte von teilweise ausgeführten Transaktionen zurückgesetzt (undo)
- Logging von
  - Write (alter und neuer Wert) durch Ti
  - Commit oder Abort von Ti
- Logs werden oft zusätzlich gesichert auf einem anderen Datenträger
- Alle Aktivitäten von Logging und Concurrency Control werden transparent durch das DBMS behandelt.

Rollen beim Entwurf und Betrieb von DB

- Endbenutzer
  - Meist ein computer-naiver Benutzer (z.B. Verkäufer, Verwaltungs-angestellter)
  - Liest Daten ein, fügt Daten ein oder ändert Daten mit vorgegebenen AP und Masken
- DB-Anwendungsprogrammierer
  - Entwickelt AP mit vorgegebenem Schema
- Schema-Designer
  - Ermittelt die Anforderungen der Benutzer und Anwender
  - Entwickelt Teilsichten und integriert diese zum globalen Schema (konzeptuelles Schema)
  - Bildet Schema zusammen mit DB-Administrator auf spezifisches DBMS ab
- DB-Administrator
  - Mitverantwortlich für das Mapping konzeptuelles $\rightarrow$ logisches Schema
  - Autorisierung neuer Benutzer (Zugriffsrechte)
  - Verantwortlich für reibungslosen Ablauf (Verfügbarkeit)
  - Sicherungen / Archivkopien
  - Installation neuer DBMS-Versionen
- DBMS-Entwickler
  - Entwickelt die DBMS-Software

Zusammenfassung

- Vorteile eines Datenbanksystems
  - Integrierter Datenbestand Vermeidung von Redundanz
  - Konsistenz, höhere Qualität des Datenbestandes
  - Einheitliche Mechanismen für Datenschutz, Datensicherheit (Recovery) und Nebenläufigkeit (Concurrency Control)
  - Physische und logische Datenunabhängigkeit
  - Leichtere und schnellere Programmentwicklung und -wartung durch einheitliche und explizite Strukturdarstellung, Nutzung von 4GL, Form- und Reportgeneratoren
  - Optimierbare Anfragesprache (Query Language)
- Nachteile eines Datenbanksystems
  - General-Purpose Software oft weniger effizient als spezialisierte Software
  - Bei konkurrierenden Anforderungen kann DBS nur für einen Teil der AP optimiert werden
  - Kosten: DBMS und zusätzliche Hardware
  - Hochqualifiziertes Personal (DB-Administration)
  - Verwundbarkeit durch Zentralisierung (Ausweg: Verteilung)