



Einführung in Datenbanken

Themen dieses Kapitels:

- Funktionsumfang von Datenbanksystemen
- Historie und Arten von Datenbanksystemen
- relationale Datenbanksysteme
- Architektur von Datenbanksystemen



Verschiedene Definitionen des Begriffs "Datenbank(-system)":

"Eine Datenbank (auch Datenbanksystem) ist ein System zur Beschreibung, Speicherung und Wiedergewinnung umfangreicher Datenmengen, die von mehreren Anwendungsprogrammen oder Anwendern benutzt werden. Es besteht aus einer Datenbasis, in der die Daten abgelegt werden und dem Verwaltungsprogramm (Datenbasismanagement), das die Daten entsprechend der vorgegebenen Beschreibung abspeichern, auffinden oder weitere Operationen durchführen kann."

[Informatikduden]

"Eine Datenbank ist eine Sammlung von Informationen zu einem bestimmten Thema oder Zweck, wie z.B. dem Verfolgen von Bestellungen oder dem Verwalten einer Musiksammlung. Wenn Ihre Datenbank nicht oder nur teilweise in einem Computer gespeichert ist, müssen Sie die Informationen aus den verschiedenen Quellen selbst koordinieren und organisieren."

[MS-Access-Online-Hilfe]



Einführung in Datenbanken

Typische Beispiele für Datenbankanwendungen:

- (Lohn-)Buchhaltung: Artikel, Adressen, Mitarbeiter, Abteilungen, ...
- Buchungssystem: freie/reservierte Plätze, Preise, ...
- Auftragsverwaltung: Aufträge, Artikel, Kundenadressen, ...
- Textverarbeitung: Textbausteine, Artikel, Literaturverweise, Adressen, ...
- GEO-Informationssystem: Lagepläne, Routen, Gebäudeinformationen, ...
- CASE-Tool: Diagramme, Programmteile, Fehlermeldungen, ...

Anfänge der Datenbanken:

- ① Anfang der 60er Jahre: elementare und anwendungsspezifische Dateien
- ② Ende der 60er Jahre: Dateiverwaltungssysteme (SAM, ISAM) als BS-Aufsatz
- ③ 70er Jahre: erste Datenbanksysteme



Probleme bei Datenverwaltung ohne Datenbanksysteme:

- große Datenmengen** : andere Softwaresysteme (Dateiverwaltung, Tabellenkalkulation, ...) können große Datenmengen nicht effizient verwalten
- paralleler Zugriff** : mehrere Benutzer oder Anwendungen können nicht gleichzeitig ungestört auf denselben Daten arbeiten
- Datenredundanz** : verschiedene Anwendungsprogramme speichern dieselben Daten, also Verschwendung von Speicherplatz, Vergessen von Änderungen, ...
- Datenschutz** : Mechanismen zum (teilweisen) Schutz der Daten vor unberechtigten Zugriffen (für verschiedene Benutzergruppen) fehlen
- Datensicherheit**: bei Absturz von Anwendungen, Stromausfall, Plattencrash, ... gehen hinzugefügte oder geänderte Daten verloren
- Datenunabhängigkeit** : Anwendungsprogrammierer oder Benutzer kennen und benutzen internen Aufbau (Format) der gespeicherten Daten

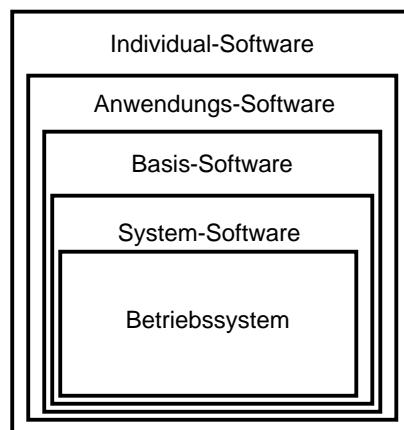


Ohne Datenbanken: Datenredundanz

- ❑ Basis- oder Anwendungssoftware verwaltet ihre eigenen Daten in ihren eigenen (Datei-) Formaten; *Bsp.:*
 - ❑ **Textverarbeitung: Texte, Artikel und Adressen**
 - ❑ **Buchhaltung: Artikel, Adressen**
 - ❑ **Lagerverwaltung: Artikel, Aufträge**
 - ❑ **Auftragsverwaltung: Aufträge, Artikel, Kundenadressen**
 - ❑ **CAD-System: Artikel, Technische Daten, Technische Bausteine**
 - ❑ **Produktion: ..., Bestelleingang: ..., Kalkulation: ...**
- ❑ Daten sind redundant: mehrfach gespeichert; Probleme: Verschwendung von Speicherplatz, „Vergessen“ von Änderungen; keine zentrale, „genormte“ Datenhaltung



Software-Schichten





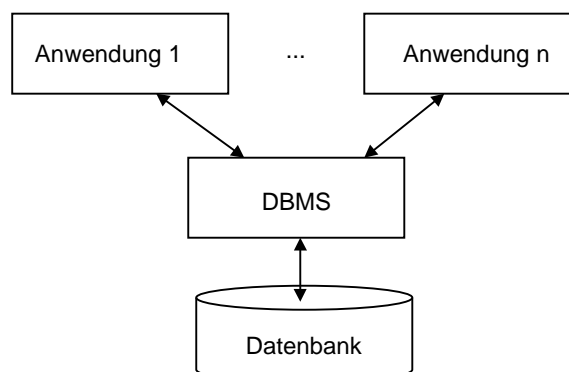
Mit Datenbanken: Datenintegration

- Die gesamte Basis- und Anwendungssoftware arbeitet auf denselben Daten (Datenbankentwurf, Datendefinition)
- Bsp.:* Adressen und Artikel werden nur einmal gespeichert
- Auch andere Probleme (Effizienz, Parallelität, Datenschutz, Datensicherheit) werden gelöst
 - Datenbanksysteme können große Datenmengen effizient verwalten (Anfragesprachen, Optimierung, Interne Ebene)**
 - Benutzer können parallel auf Datenbanken arbeiten (Transaktionskonzept)**
 - Datenunabhängigkeit durch 3-Ebenen-Konzept**
 - Datenschutz (kein unbefugter Zugriff) und Datensicherheit (kein ungewollter Datenverlust) werden vom System gewährleistet**



Begriffe

- DBMS: Datenbank-Management-System
- DBS: Datenbanksystem (DBMS + Datenbank)





Prinzipien I

Grundmerkmale von Datenbanksystemen

- verwalten persistente (langfristig zu haltende) Daten**
- verwalten große Datenmengen effizient**
- Datenbankmodell, mit dessen Konzepten alle Daten einheitlich beschrieben werden (Integration)**
- Operationen und Sprachen deskriptiv, getrennt von einer Programmiersprache**
- Transaktionskonzept, Concurrency Control: logisch**
- zusammenhängende Operationen atomar (unteilbar), Auswirkungen langlebig, können parallel durchgeführt werden**
- Datenschutz, Datenintegrität (Konsistenz), Datensicherheit**



Prinzipien II

- Grundprinzip moderner Datenbanksysteme
 - ◆ **3-Ebenen-Architektur (physische Datenunabhängigkeit, logische Datenunabhängigkeit)**
 - ◆ **Trennen zwischen Schema (etwa Tabellenstruktur) und Instanz (etwa Tabelleninhalt)**

angelehnt an 9 Codd'sche Regeln:



Die neun Codd'schen Anforderungen für Datenbanksysteme

- ❶ **Integration** : einheitliche nichtredundante Datenverwaltung
- ❷ **Operationen** : Speichern, Suchen, Ändern (ad hoc und "fest verdrahtet")
- ❸ **Katalog** : Zugriffe auf Datenbankbeschreibung im "Data Dictionary"
- ❹ **Benutzersichten** : unterschiedliche Sichten auf Datenbankausschnitte
- ❺ **Konsistenzüberwachung** : Gewährleistung der Korrektheit des Datenbankinhaltes
- ❻ **Datenschutz** : Ausschluß unautorisierter Zugriffe
- ❼ **Transaktionen** : mehrere Operationen als Funktionseinheit
- ❽ **Synchronisation** : parallele Transaktionen koordinieren
- ❾ **Datensicherung** : Wiederherstellung von Daten nach Systemfehlern



Entwicklungslinien = Historie der Datenbanksysteme:

ab 60er Jahre:

Datenbanksysteme basierend auf **hierarchischem Modell, Netzwerkmodell:**

- Zeigerstrukturen zwischen Daten
- schwache Trennung interne/konzeptuelle Ebene
- navigierende Anfragesprachen (entlang Zeigerstrukturen)
- Trennung Datenbanksprache/Programmiersprache

ab 70er Jahre:

relationale Datenbanksysteme :

- Daten in Tabellenstruktur
- 3-Ebenen-Konzept (externe, konzeptuelle, interne Ebene)
- deklarative standardisierte Datenbanksprache (SQL)
- Trennung Datenbanksprache/Programmiersprache

**Einführung in Datenbanken****ab 80er Jahre:****Wissensbanksysteme** (deduktive Datenbanken):

- Daten in Tabellenstruktur
- stark deklarative Anfragesprachen (logikbasiert)
- integrierte Datenbankprogrammiersprache

objektorientierte Datenbanksysteme

- Daten in komplexeren Objektstrukturen
- navigierende (oder deklarative) Anfragesprache
- oft OO-Programmiersprache = Datenbankprogrammiersprache
- oft keine vollständige Ebenentrennung

geographische Datenbanksysteme :

- meist spezielle Erweiterungen relationaler oder objektorientierter Systeme
- neben "normalen" Daten gibt es geometrische Daten (2D, 3D)
- spezielle Indexstrukturen (QuadTrees, ...) unterstützen räumliche Anfragen

**Einsatzgebiete und Grenzen von (relationalen) Datenbanksystemen:****Klassische Einsatzgebiete:**

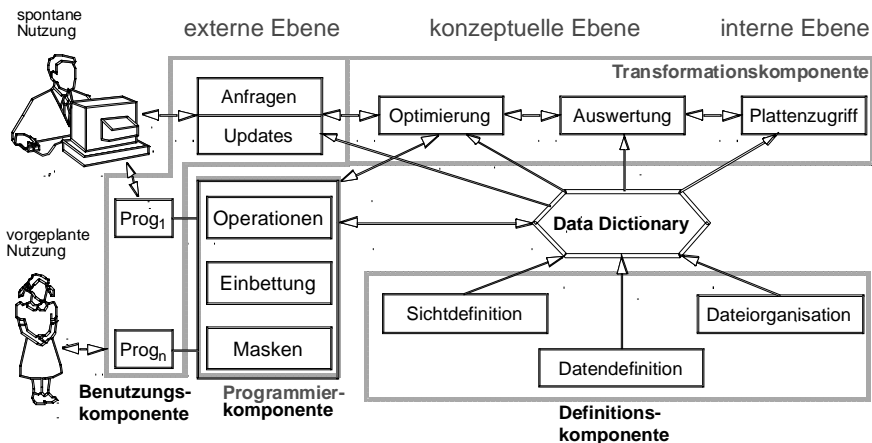
- etwa Buchhaltungssystem, Bibliothekssysteme, ...
- viele Objekte (15000 Bücher, 300 Benutzer, 100 Ausleihvorgänge pro Woche)
- wenige Objekttypen (BUCH, BENUTZER, AUSLEIHUNG, ...)

“Nicht-Standard“-Anwendungen:

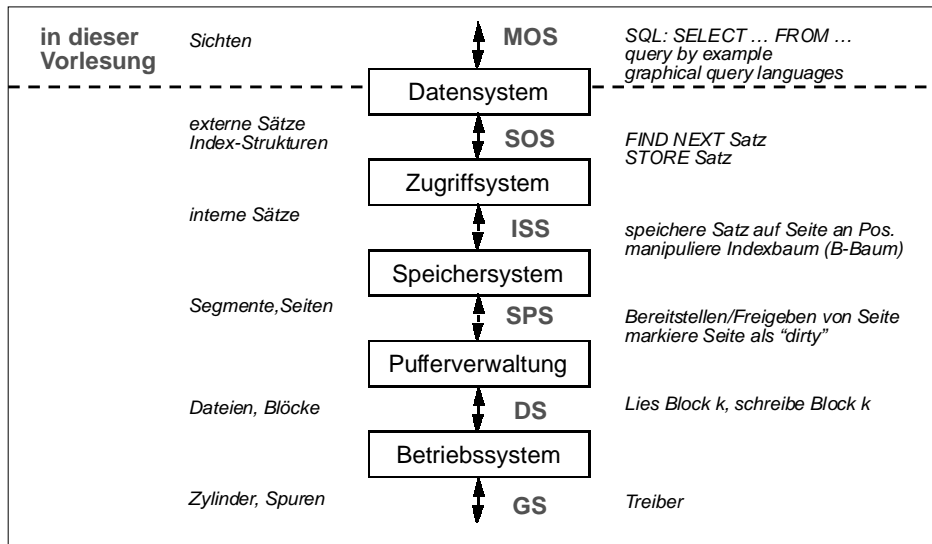
- Speicherung von CAD-System- oder CASE-Tool-Daten:
viele Objekte, viele (ähnliche) stark strukturierte Objekttypen + Operationen
 - objektorientierte Datenbanksysteme**
- Speicherung und Bearbeitung von Expertenwissen:
wenige Objekte, viele Objekttypen, komplizierte (Inferenz-)Operationen
 - deduktive Datenbanksysteme**
- Speicherung von Landkarten, Gebäudeinformationen mit Eintragungen
viele, stark strukturierte Objekte + räumliche Informationen
 - geographische Datenbanksysteme**

**Systemarchitektur eines Datenbanksystems:**

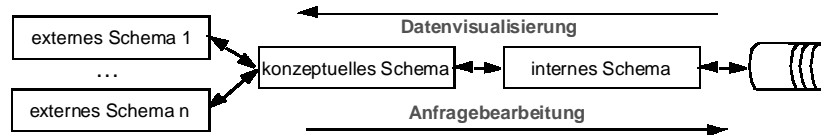
Komponentenzusammenfassung gemäß ANSI-SPARC-Standard von 1978
(SPARC = Standards Planning and Requirements Engineering Committee)

**Komponenten eines Datenbanksystems:**

- Datendefinition** : konzeptuelle Definition aller Daten (konzeptuelles Schema)
- Sichtdefinition** : Definition von Benutzersichten (externes Schema)
- Dateiorganisation** : Definition des (internen) Dateiaufbaus und Zugriffsstrukturen (internes Schema)
- Masken** : Entwurf der Benutzeroberfläche für Datenbankanwender
- Operationen** : Anfragen und Änderungsoperationen (Updates) in Programmen
- Einbettung** : Einbettung von Datenbankoperationen in Programme
- Prog_i** : verschiedene Anwendungsprogramme
- Optimierung** : Umformung von Anfragen (Updates) in effiziente Gestalt
- Auswertung** : Durchführung der umgeformten Anfragen (Updates)
- Plattenzugriff** : effiziente Ablage der Daten im Sekundärspeicher

**Fünf-Schichtenarchitektur (Senko 1973, Härder 1987):****Fünf-Schichtenarchitektur (Fortsetzung):**

- mengenorientierte Schnittstelle **MOS** :
deklarative Datenmanipulationssprache auf Tabellen, Sichten, Zeilen
- satzorientierte Schnittstelle **SOS** :
Abbildung der Daten auf Sätze, navigierender logischer Zugriff auf Sätze
- interne Satzschnittstelle **ISS** :
Abbildung Sätze auf Seiten und Verwaltung interner Indexstrukturen
(Suchbäume, Hashtabellen, ...)
- Pufferschnittstelle **SPS** :
Bereitstellung logischer Seiten für Sätze, Indexstrukturen, ...
- Datei- oder Seitenschnittstelle **DS** :
Abbildung logischer Seiten auf Blöcke im Dateisystem
- Geräteschnittstelle **GS** :
gerätespezifische Schnittstelle (als Bestandteil des Betriebssystems)

**Drei-Schichten-Schemaarchitektur (nach ANSI-SPARC):****Datenbankschema besteht aus**

- einem internen Schema (Datenverwaltung auf Platte, Indexstrukturen, ...)
- einem konzeptuellen Schema (Beschreibung der Gesamtstruktur, Integritätsbedingungen, ...)
- i.a. mehreren externen Schemata (Sichten für verschiedene Anwendungen, Benutzergruppen, ...)

Trennung Schema - Instanz

- Schema = Beschreibung der Daten (Metadaten)
- Instanz = Anwenderdaten (Datenbankzustand, -ausprägung)

**Relationale Datenbank(-Management)-Systeme (RDBMS):****Gemeinsame Merkmale:**

- Drei-Ebenen-Architektur nach ANSI-SPARC
- standardisierte Datenbanksprache (SQL = Structured Query Language)
- Einbettung von SQL in kommerzielle Programmiersprachen
- Werkzeuge für (interaktive) Definition, Anfrage und Darstellung von Daten; Entwurf von Anwendungsprogrammen, Benutzeroberflächen (Web)
- kontrollierter Mehrbenutzerbetrieb, Datenschutz- und Datensicherheit

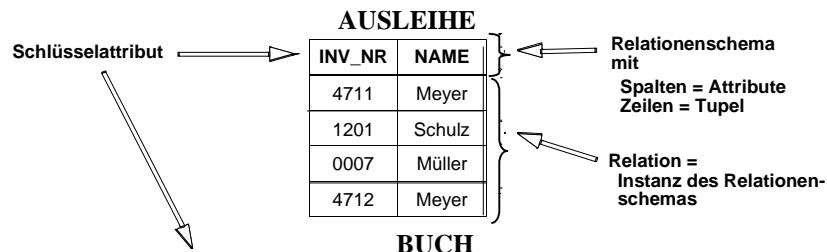
Beispiele echter RDBMS:

ORACLE, DB2, Ingres, Informix, ...

Beispiele für "Pseudo"-RDBMS:

MS-Access, dBASE, ...

(es fehlen u.a. Drei-Ebenen-Architektur, Optimierungen)

**Relationale Daten(-bank)-Definition mit Datenbankschema:****BUCH**

INV_NR	TITEL	ISBN	AUTOR
0007	Dr. No	3-125...	James Bond
1201	Objektbanken	3-111...	Heuer
4711	Datenbanken	3-765...	Vossen
4712	Datenbanken	3-891...	Ullmann
4717	PASCAL	3-999...	Wirth

Datenbank (DB) = Menge von Relationen**DB-Schema = Menge von Relationenschemata****Integritätsbedingungen = Konsistenzregeln zum Ausleih-Beispiel:****lokale Integritätsbedingungen:**

- zu jedem Attribut in jedem Relationenschema gibt es eine Typdefinition, die die zulässigen Werte der entsprechenden Spalte festlegt (String, Integer, ...)
- Attribut INV_NR ist **(Primär-)Schlüssel** von BUCH
 - in BUCH keine zwei Tupel mit demselben INV_NR-Wert
- Attribut INV_NR ist **(Primär-)Schlüssel** von AUSLEIHE
 - in AUSLEIHE keine zwei Tupel mit demselben INV_NR-Wert

globale Integritätsbedingungen:

- jeder INV_NR-Wert des „Fremdschlüssels“ von AUSLEIHE muß auch in BUCH auftreten

**Anfrageoperationen:**

-
- Selektion**
- : Zeilen (Tupel) auswählen

SEL [NAME = 'Meyer'](AUSLEIHE) ergibt:

INV_NR	NAME
4711	Meyer
4712	Meyer

-
- Projektion**
- : Spalten (Attribute) auswählen:

PROJ[INV_NR, TITEL](BUCH) ergibt:

INV_NR	TITEL
0007	Dr. No
1201	Objektbanken
4711	Datenbanken
4712	Datenbanken
4717	PASCAL

**Anfrageoperationen:**

-
- Verbund (natural join)**
- : Tabellen verknüpfen über gleichbenannten Spalten und gleichen Werten:

PROJ[INV_NR, TITEL](BUCH) JOIN SEL [NAME = 'Meyer'](AUSLEIHE)

ergibt:

INV_NR	TITEL	NAME
4711	Datenbanken	Meyer
4712	Datenbanken	Meyer

-
- weitere Operationen**
- :

-
- Vereinigung, Durchschnitt, Differenz von gleich strukturierten Tabellen
-
-
- Umbenennung von Spalten (Attributen)

Achtung:Operationen lassen sich beliebig kombinieren (schachteln) = **Relationenalgebra**



Sprachen und Sichten (für relationale Datenbanken):

Abfragesprache:

- SQL-Standard (Structured Query Language)
- interaktive Formulierung von Datenbankabfragen
- Relationenalgebra + Funktionen (SUM, MAX, COUNT, ...)
- ggf. mit grafischer Benutzeroberfläche oder "query by example"

Update-Komponente:

- interaktive Eingabe, Löschung, Änderung von Tupeln
- Prüfung lokaler und globaler Integritätsbedingungen

Definition von Sichten:

- häufig benötigte Anfragen werden als Sicht = virtuelle Tabelle gespeichert

```
MEYERS := PROJ[INV_NR, TITEL](BUCH) JOIN  
        SEL [NAME = 'Meyer'](AUSLEIHE)
```



SQL als Standard

```
select BUCH.INV_NR, TITEL, NAME  
from BUCH, AUSLEIHE  
where Name = „Meyer“ and  
        BUCH.INV_NR = AUSLEIHE.INV_NR
```

**Optimierung von Anfragen:****Problem:**

Finde zu einem gegebenen Relationenalgebra-Ausdruck einen äquivalenten Ausdruck, der effizienter auswertbar ist.

Beispiel: algebraische Optimierung = Umformung von Ausdrücken

$SEL [A = \text{Konstante}] (REL1 JOIN REL2)$ und A aus $REL1$

$SEL [A = \text{Konstante}] (REL1) JOIN REL2$

sind äquivalente Anfragen, aber mit 100 Tupel in $REL1$ und 50 Tupel in $REL2$:

Auswertung erster Variante:

$5000 JOIN + 5000 SEL = 10000$ Operationen

Auswertung zweiter Variante, falls 10 Tupel in $REL1$ $A = \text{Konstante}$ erfüllen:

$100 SEL + 10 * 50 JOIN = 600$ Operationen

Generelle Regel: SELEKTION vor JOIN durchführen

**Übersicht über Vorlesungsinhalt** **Datenbankmodelle**

Entity-Relationship-Modell, Relationenmodell

 Grundlagen von Anfragen und Änderungen

Relationenalgebra, Tupel-/Bereichskalkül

 SQL **Datenbankentwurf**

Normalformen