Kapitel DB:I

I. Einführung

- Datenintensive Anwendungen
- Entstehung von Datenbanksystemen
- Begriffsbildung und Einordnung
- Datenbank-Management-Systeme
- Relationale Datenbanksysteme

©STEIN 2004-2018

Datenintensive Anwendungen

- Content-Management-System. Textdokumente, Bilder, sonstige „Assets“
- Buchhaltung. Personaldaten, Lohndaten, Kostenstellen
- Auftragsverwaltung. Aufträge, Kundenadressen, Artikelnummern
- Soziale Software. Benutzerprofile, Life-Logging-Daten
- CAD-System. Zeichnungen, Makro-Bibliotheken, technische Daten
- PPS-System. Maschinenbelegungspläne, Produktionspläne, Teilenummern
- Krankenversicherung. Patientenakten, Krankenhausstatistiken
- Online-Shop. Produktbeschreibungen, Kundendaten
- . . .

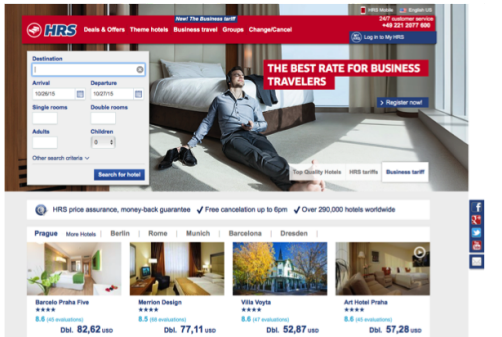
©STEIN 2004-2018

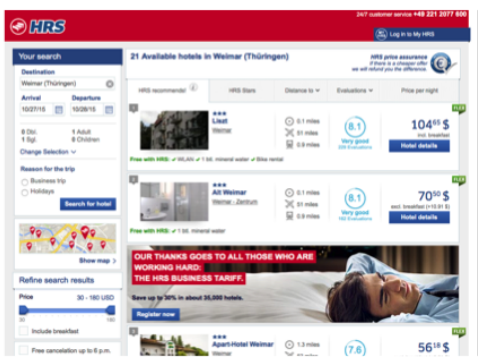
Datenintensive Anwendungen

Beispiel: Hotelreservierung

- Hotel Reservation Service HRS
  - zweitgrößter Hotel-Anbieter der Welt
  - zweitgrößte Tourismus-Webseite Europas
  - basiert auf riesigen Datenbanken

- Daten bei HRS (Stand 2011)
  - 250 000+ Hotels
    ca. 100 Metadaten und 1 000+ Amenities pro Hotel, dynamische Angebote
  - 30 000 000+ Buchungen
    ca. 100 Metadaten pro Buchung, 3 000 000+ jährlich, 50% mit Bewertung
  - 3 000 000+ Benutzer
    200 000+ jährlich, diverse Suchanfragen pro Benutzer

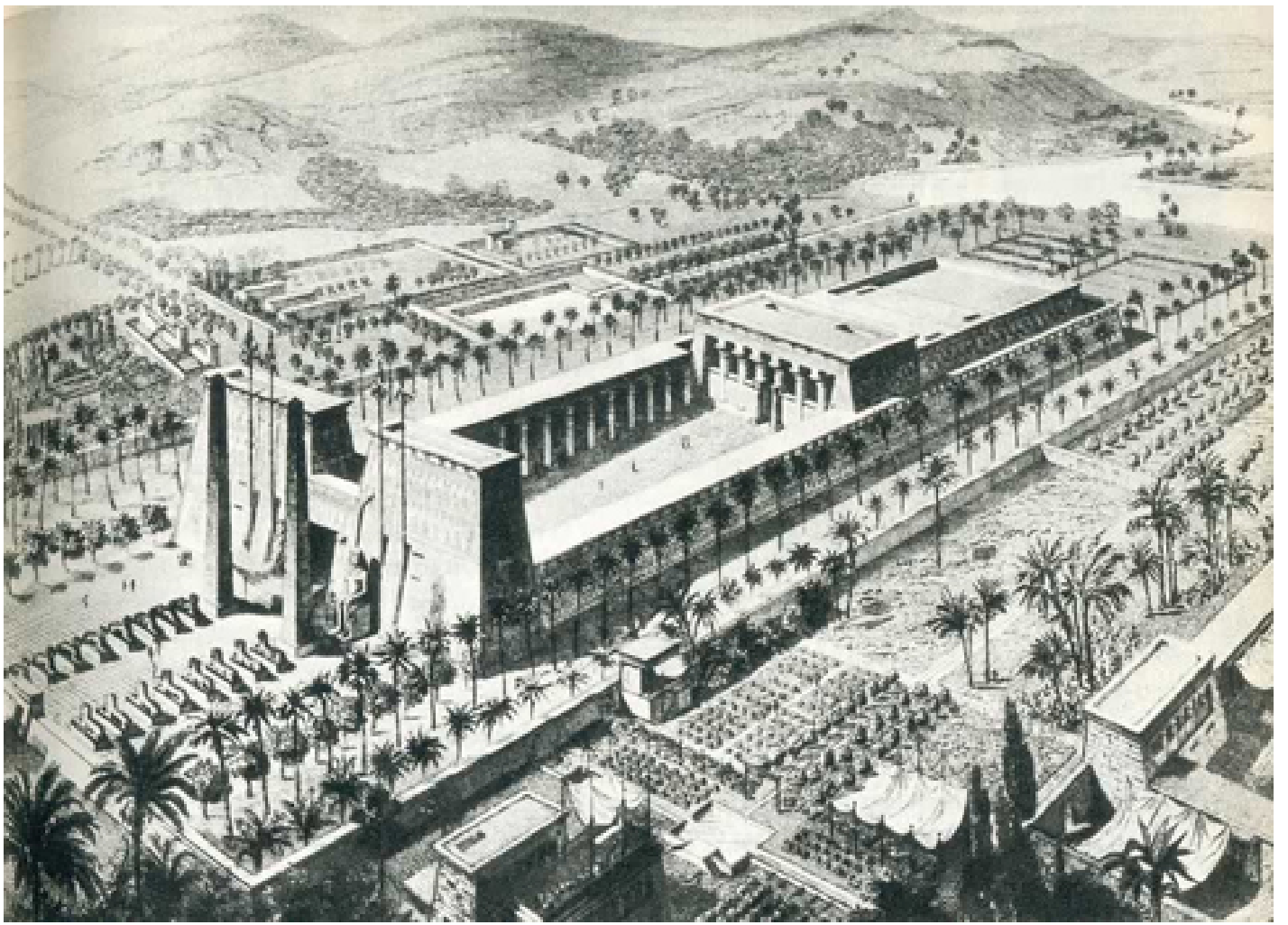




©STEIN 2004-2018

Entstehung von Datenbanksystemen

Alte Bibliothek von Alexandria



©STEIN 2004-2018

Entstehung von Datenbanksystemen

Neue Bibliothek von Alexandria

©STEIN 2004-2018

Bemerkungen:

- Die alte Bibliothek von Alexandria wurde von Ptolemeus I 288 v. Chr. gegründet. Gelehrte, Intellektuelle, Wissenschaftler und Schüler fanden hier ein Umfeld, um über das damalige Wissen zu diskutieren und zu lernen.

  Aristarchus war einer der ersten, der behauptete, dass sich die Erde um die Sonne dreht. Hipparchus war der erste, der das Solarjahr mit einer Abweichung von 6½ Minuten messen konnte. Eratosthenes war der erste, der den Durchmesser der Erde berechnete. Euclid, der Vater der Geometrie, war ebenfalls hier tätig. Archimedes, der größte Mathematiker der alten Welt, lehrte hier. Der Gründer des Bibliothekswesens, Callimachus, erfand hier ein Sortierungssystem: einen Katalog, um Schriftrollen nach Thema oder Autor zu finden.

- Die Bibliothek war offen für jedermann und die besten Werke der Zeit wurden hier gesammelt. Das Alte Testament wurde erstmalig aus dem Hebräischen ins Griechische übersetzt. Es gab eine Mischung von diversen Kulturen und Sprachen unter den Gelehrten, aber die griechische Sprache setzte sich durch, da für die griechischen Philosophen Alexandrien mit ihrer Bibliothek die intellektuelle Hauptstadt der Zeit war.

- Die alte Bibliothek (es waren eigentlich zwei) wurde durch die Römer 49 v. Chr. und ein Feuer zerstört. Hier war das gesamte Wissen der Antike vorhanden; es soll an die 700 000 Papyrusschriftrollen gegeben haben.

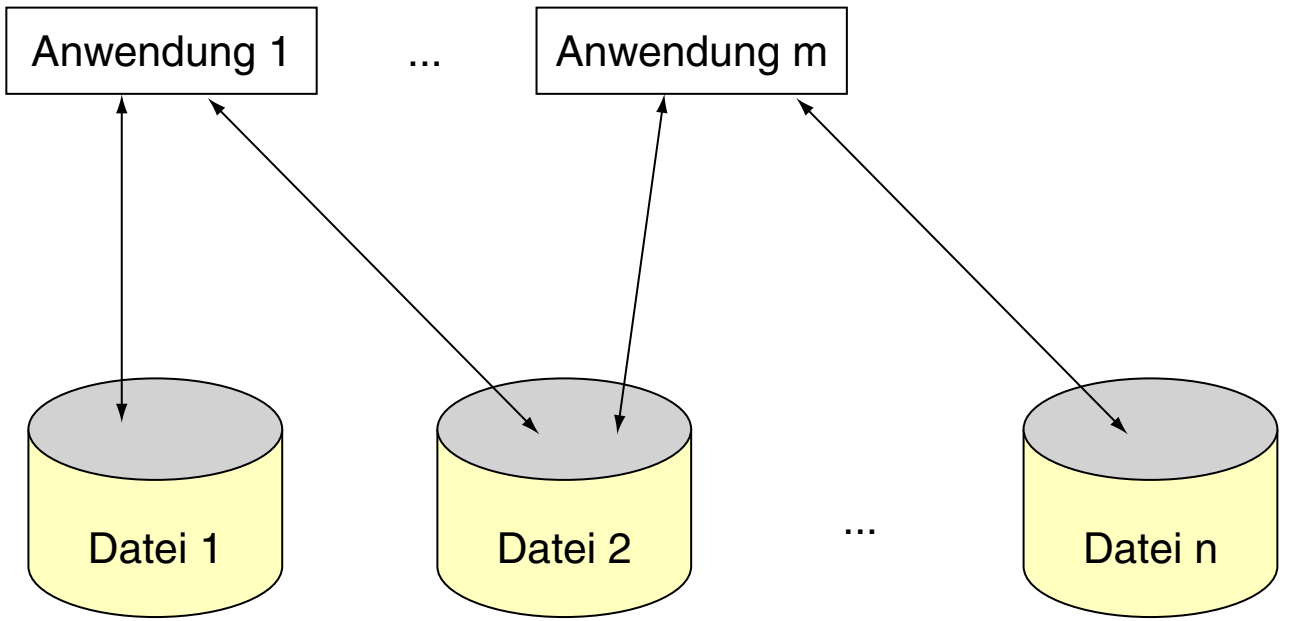
  Die UNESCO setzte sich für einen Neubau der Bibliothek ein. Baubeginn war 1995, Fertigstellung 2001, Kosten ca. 250 Millionen Dollar. Die Gesamtfläche der Bibliothek beträgt 45 000 Quadratmeter; ihre Kapazität ist auf 8 Millionen Bücher ausgelegt.

[www.bibliothek-alexandria.de, www.bibalex.org]

©STEIN 2004-2018

Entstehung von Datenbanksystemen

Anfang 60er Jahre: direkter Dateizugriff



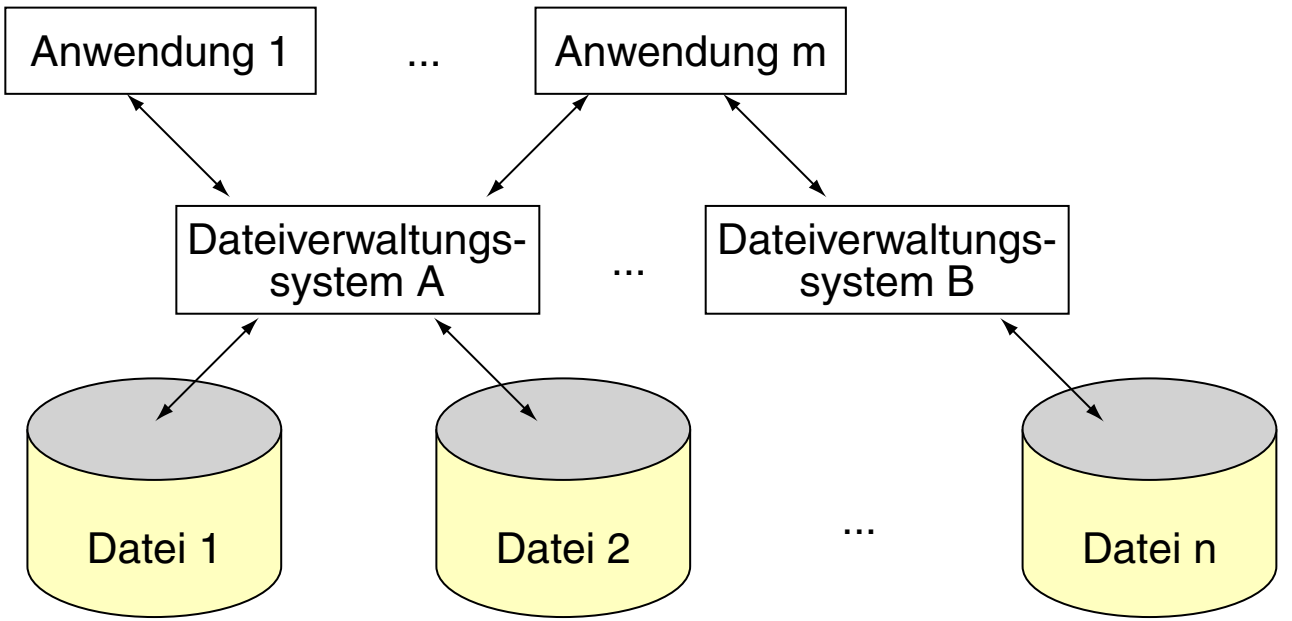
Probleme:

- Geräteabhängigkeit
- redundante Speicherung
- Dateninkonsistenz

©STEIN 2004-2018

Entstehung von Datenbanksystemen

Ende 60er Jahre: Dateizugriff mit Verwaltungssystem



Fortschritt:

- Geräteunabhängigkeit

Probleme:

- redundante Speicherung
- Dateninkonsistenz

©STEIN 2004-2018

Entstehung von Datenbanksystemen

Probleme der frühen Datenverarbeitung

- Redundanz

  Erhöhter Verwaltungs- und Verarbeitungsaufwand, bedingt durch Speichern gleicher Daten aus verschiedenen Anwendungsprogrammen in verschiedenen Dateien.

- Inkonsistenz

  Fehlende logische Übereinstimmung von Daten, weil Änderungen gleicher Inhalte in verschiedenen Dateien nicht nachvollzogen werden.

- Datenverteilung (Replikation)

  Daten werden an verschiedenen Orten benötigt, jedoch wird gezieltes Anlegen und Pflegen von Kopien nicht unterstützt.

- Daten-Programm-Abhängigkeit

  Das direkte Schreiben und Lesen der Daten durch Anwendungsprogramme hat viele negative Konsequenzen: Änderung am Dateiaufbau (der gespeicherten Daten) erfordert Änderung des Anwendungsprogramms, Erweiterung des Anwendungsprogramms erfordert Änderung des Dateiaufbaus, Anwendungsprogrammierer und Benutzer nutzen die versteckte Semantik der gespeicherten Daten.

©STEIN 2004-2018

Entstehung von Datenbanksystemen

Probleme der frühen Datenverarbeitung

- Redundanz
  Erhöhter Verwaltungs- und Verarbeitungsaufwand, bedingt durch Speichern gleicher Daten aus verschiedenen Anwendungsprogrammen in verschiedenen Dateien.

- Inkonsistenz
  Fehlende logische Übereinstimmung von Daten, weil Änderungen gleicher Inhalte in verschiedenen Dateien nicht nachvollzogen werden.

- Datenverteilung (Replikation)
  Daten werden an verschiedenen Orten benötigt, jedoch wird gezieltes Anlegen und Pflegen von Kopien nicht unterstützt.

- Daten-Programm-Abhängigkeit
  Das direkte Schreiben und Lesen der Daten durch Anwendungsprogramme hat viele negative Konsequenzen: Änderung am Dateiaufbau (der gespeicherten Daten) erfordert Änderung des Anwendungsprogramms, Erweiterung des Anwendungsprogramms erfordert Änderung des Dateiaufbaus, Anwendungsprogrammierer und Benutzer nutzen die versteckte Semantik der gespeicherten Daten.

©STEIN 2004-2018

Entstehung von Datenbanksystemen

Probleme der frühen Datenverarbeitung (Fortsetzung)

- Inflexibilität
  Auswertung von Daten sowie Realisierung neuer Anwendungen ist schwierig, weil Inhalte aus mehreren Dateien nur mit Aufwand kombinierbar sind.
- große Datenmengen
  Anwendungen (Mail-Client, Tabellenkalkulation, etc.) sind nicht für die effiziente Verwaltung großer Datenmengen konzipiert.
- paralleler Zugriff
  Mehrere Benutzer oder Anwendungen können nicht ohne gegenseitige Beeinflussung gleichzeitig auf denselben Daten arbeiten.
- Datensicherheit
  Datenverlust durch Absturz von Anwendungen, Platten-Crash, Stromausfall, etc.
- Datenschutz
  Fehlende Mechanismen zum Schutz der Daten vor unberechtigten Zugriffen.

©STEIN 2004-2018

Entstehung von Datenbanksystemen

Probleme der frühen Datenverarbeitung (Fortsetzung)

- Inflexibilität
  Auswertung von Daten sowie Realisierung neuer Anwendungen ist schwierig, weil Inhalte aus mehreren Dateien nur mit Aufwand kombinierbar sind.

- große Datenmengen
  Anwendungen (Mail-Client, Tabellenkalkulation, etc.) sind nicht für die effiziente Verwaltung großer Datenmengen konzipiert.

- paralleler Zugriff
  Mehrere Benutzer oder Anwendungen können nicht ohne gegenseitige Beeinflussung gleichzeitig auf denselben Daten arbeiten.

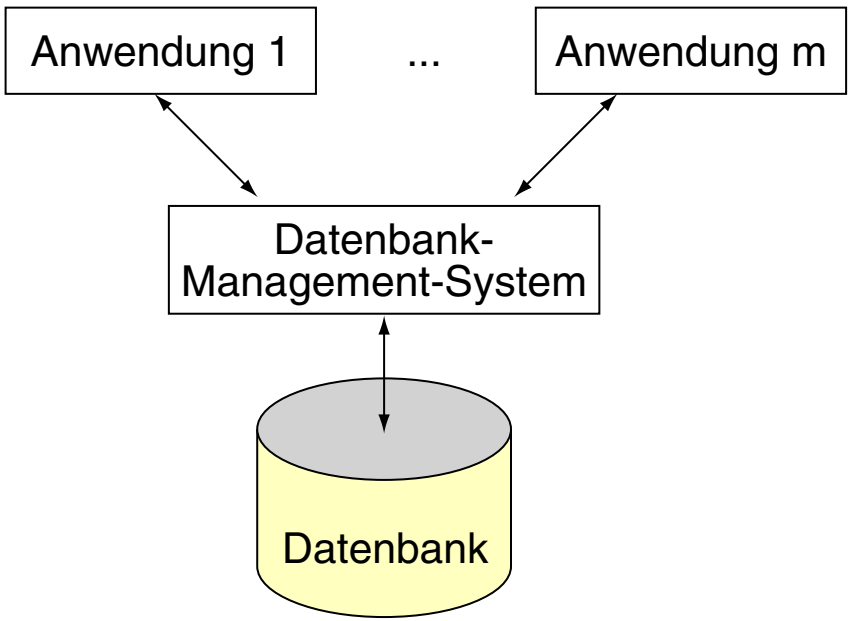
- Datensicherheit
  Datenverlust durch Absturz von Anwendungen, Platten-Crash, Stromausfall, etc.

- Datenschutz
  Fehlende Mechanismen zum Schutz der Daten vor unberechtigten Zugriffen.

©STEIN 2004-2018

Entstehung von Datenbanksystemen

Mitte 70er Jahre: explizites Daten-Management



Fortschritt:

- Datenintegration (statt Redundanz)
- einheitliche Operatoren
- Konsistenz

Herausforderungen:

- Transaktionsabwicklung
- Effizienz
- Datenschutz

©STEIN 2004-2018

Begriffsbildung und Einordnung

Definition 1 (Datenbank = Datenbasis, DB)

Eine Datenbank ist eine strukturierte Sammlung einer umfangreichen Menge persistenter (= dauerhaft zur Verfügung stehender) Daten in elektronischer Form. Diese Daten sind nach bestimmten Merkmalen und Regeln erfasst, geordnet und abgelegt. Der Zugriff auf die Daten und deren Änderung ist ohne großen Aufwand möglich.

©STEIN 2004-2018

Begriffsbildung und Einordnung

Definition 1 (Datenbank = Datenbasis, DB)

Eine Datenbank ist eine strukturierte Sammlung einer umfangreichen Menge persistenter (= dauerhaft zur Verfügung stehender) Daten in elektronischer Form. Diese Daten sind nach bestimmten Merkmalen und Regeln erfasst, geordnet und abgelegt. Der Zugriff auf die Daten und deren Änderung ist ohne großen Aufwand möglich.

Definition 2 (Datenbank-Management-System, DBMS)

Gesamtheit der Programme zum Zugriff auf die Datenbasis, zur Datenmodifikation und zur Gewährleistung der Konsistenz. Das Datenbank-Management-System bildet eine Softwareschicht zwischen dem Benutzer und der physischen Darstellung der Daten.

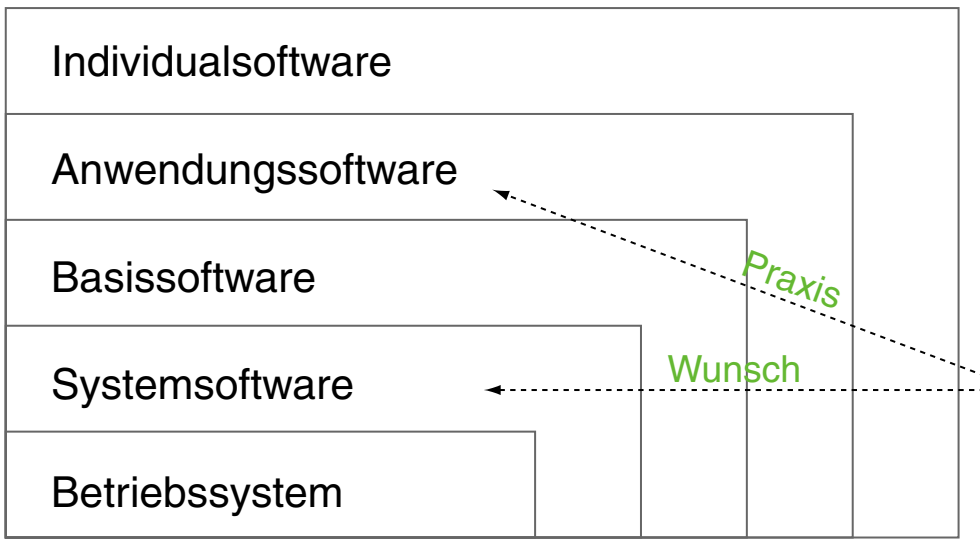
Definition 3 (Datenbanksystem, DBS)

Datenbanksystem = DB + DBMS

©STEIN 2004-2018

Begriffsbildung und Einordnung

Softwareschichten eines Computer-Systems



Speziallösung

CAD-System, DTP-System

Textverarbeitung, Spread-Sheet

Datenbanksystem, Graphik-API

Linux, Windows, MacOS

©STEIN 2004-2018

Bemerkungen:

- Jede Softwareschicht im Computer baut auf den weiter innenliegenden Schichten auf.
- Idealerweise sollte Basissoftware, Anwendungssoftware oder Individualsoftware ihre Datenverwaltung über ein standardisiertes Application-Programming-Interface (API) auf der Systemsoftwareebene realisieren. Stichwort: Datenbanksystem

©STEIN 2004-2018

Datenbank-Management-Systeme

Neun Anforderungen an ein DBMS [E.F. Codd 1982]

1. Integration

   Einheitliche (= nicht-redundante) Verwaltung der Daten aller Anwendungen.

2. Operationen

   Operationen zum Speichern, Suchen und Ändern des Datenbankinhaltes existieren.

3. Katalog bzw. Data-Dictionary

   Der Katalog ermöglicht den Zugriff auf die Datenbankbeschreibung (Metasicht).

4. Benutzersichten

   Realisierbarkeit unterschiedlicher Sichten auf den Datenbankinhalt.

5. Konsistenzüberwachung bzw. Integritätssicherung

   Gewährleistung der Korrektheit des Datenbankinhaltes.

6. Zugriffskontrolle bzw. Datenschutz

   Verhinderung unautorisierter Zugriffe auf den Datenbankinhalt.

©STEIN 2004-2018

Datenbank-Management-Systeme

Neun Anforderungen an ein DBMS (Fortsetzung)

7. Transaktionen

   Zusammenfassung mehrerer Operationen zu einer Funktionseinheit, die unteilbar ausgeführt wird.

8. Synchronisation

   Koordination konkurrierender Transaktionen mehrerer Benutzer.

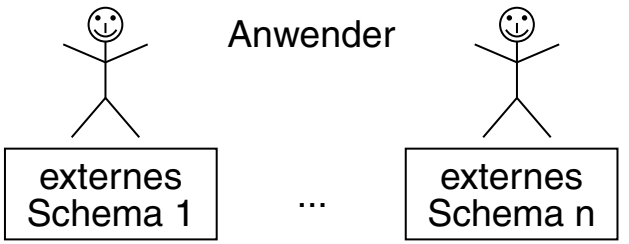
9. Datensicherung

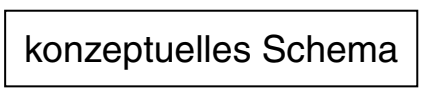
   Wiederherstellung von Daten nach Systemfehlern.

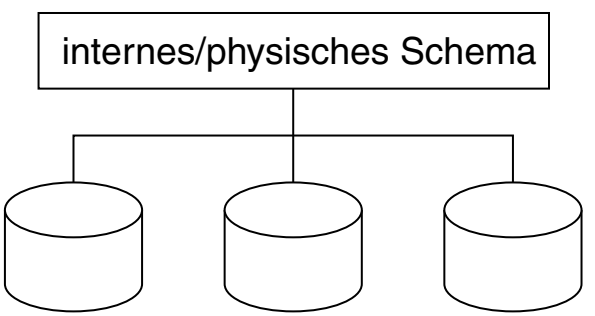
©STEIN 2004-2018

Datenbank-Management-Systeme

Drei-Schichten-Schema-Architektur [ANSI/SPARC 1975]



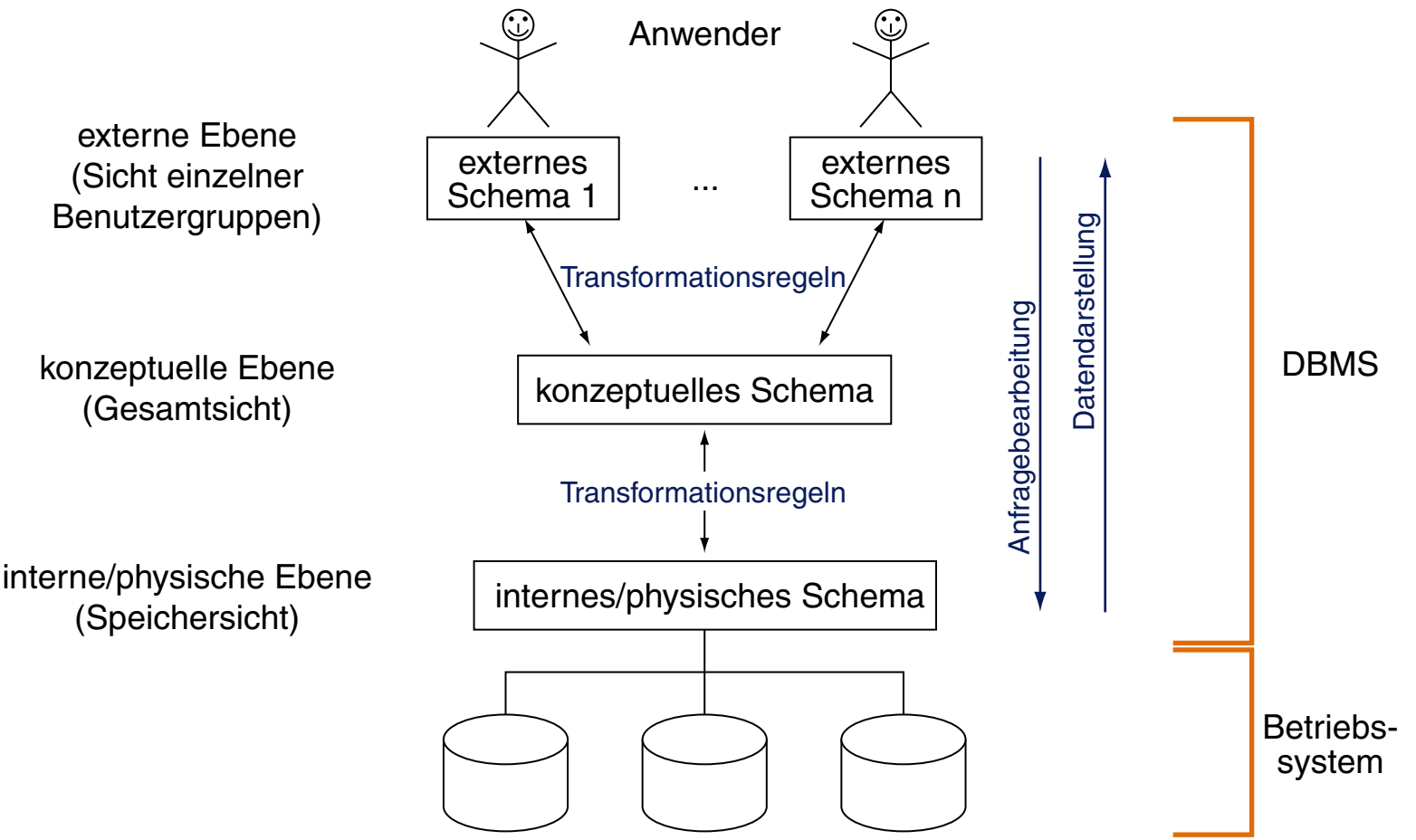




©STEIN 2004-2018

Datenbank-Management-Systeme

Drei-Schichten-Schema-Architektur [ANSI/SPARC 1975]



©STEIN 2004-2018

Bemerkungen:

- In der Praxis kommt die dargestellte Aufteilung nicht in ihrer Reinform zum Einsatz: statt eines konzeptuellen Schemas (realisiert mit einem semantisch reichen Datenmodell) ist im DBMS ein logisches Schema (realisiert mit einem logischen Datenmodell wie dem Relationenmodell) implementiert.
- Die Anfragebearbeitung und die Datendarstellung realisieren Abbildungen zwischen den drei Ebenen.
- konzeptuell im Sinne von „paradigmenunabhängig“
- ANSI = American National Standards Institute
- SPARC = Standards Planning and Requirements Committee

©STEIN 2004-2018

Datenbank-Management-Systeme

Drei-Schichten-Schema-Architektur

Die Schema-Architektur nach ANSI/SPARC garantiert Datenunabhängigkeit.

Datenunabhängigkeit bezeichnet die Eigenschaft, dass höhere Ebenen des Modells unbeeinflusst von Änderungen auf niedrigeren Ebenen bleiben.

- logische Datenunabhängigkeit, auch Anwendungsunabhängigkeit

  Änderungen des konzeptuellen Schemas (z.B. Informationen über neue Typen von Objekten, weitere Eigenschaften für existierende Objekte) haben keine Auswirkungen auf externe Schemata (z.B. existierende Anwendungsprogramme).

- physische Datenunabhängigkeit

  Änderungen des physischen Schemas (z.B. Wechsel von einer Zugriffsstruktur zu einer effizienteren, Benutzung anderer Datenstrukturen, Austausch von Algorithmen) lassen die konzeptuelle Ebene unverändert und haben somit auch keine Auswirkungen auf externe Schemata.

©STEIN 2004-2018

Datenbank-Management-Systeme

Drei-Schichten-Schema-Architektur

Die Schema-Architektur nach ANSI/SPARC garantiert Datenunabhängigkeit.

Datenunabhängigkeit bezeichnet die Eigenschaft, dass höhere Ebenen des Modells unbeeinflusst von Änderungen auf niedrigeren Ebenen bleiben.

- logische Datenunabhängigkeit, auch Anwendungsunabhängigkeit

  Änderungen des konzeptuellen Schemas (z.B. Informationen über neue Typen von Objekten, weitere Eigenschaften für existierende Objekte) haben keine Auswirkungen auf externe Schemata (z.B. existierende Anwendungsprogramme).

- physische Datenunabhängigkeit

  Änderungen des physischen Schemas (z.B. Wechsel von einer Zugriffsstruktur zu einer effizienteren, Benutzung anderer Datenstrukturen, Austausch von Algorithmen) lassen die konzeptuelle Ebene unverändert und haben somit auch keine Auswirkungen auf externe Schemata.

©STEIN 2004-2018

Datenbank-Management-Systeme

Was ein DBMS zu tun hat

Beispielanfrage mittels SQL an eine relationale DB:

```
select  Titel
from    Buch
where   Autor = Pearl
```

Buch			
Inv_Nr	Titel	ISBN	Autor
0110	Lesebuch	2-341...	Popper
1201	C++	2-123...	Stroustrup
3309	Längengrad	2-123...	Sobel
4711	Glücksformel	2-679...	Klein
7510	Heuristics	9-212...	Pearl

©STEIN 2004-2018

Datenbank-Management-Systeme

Was ein DBMS zu tun hat (Fortsetzung)

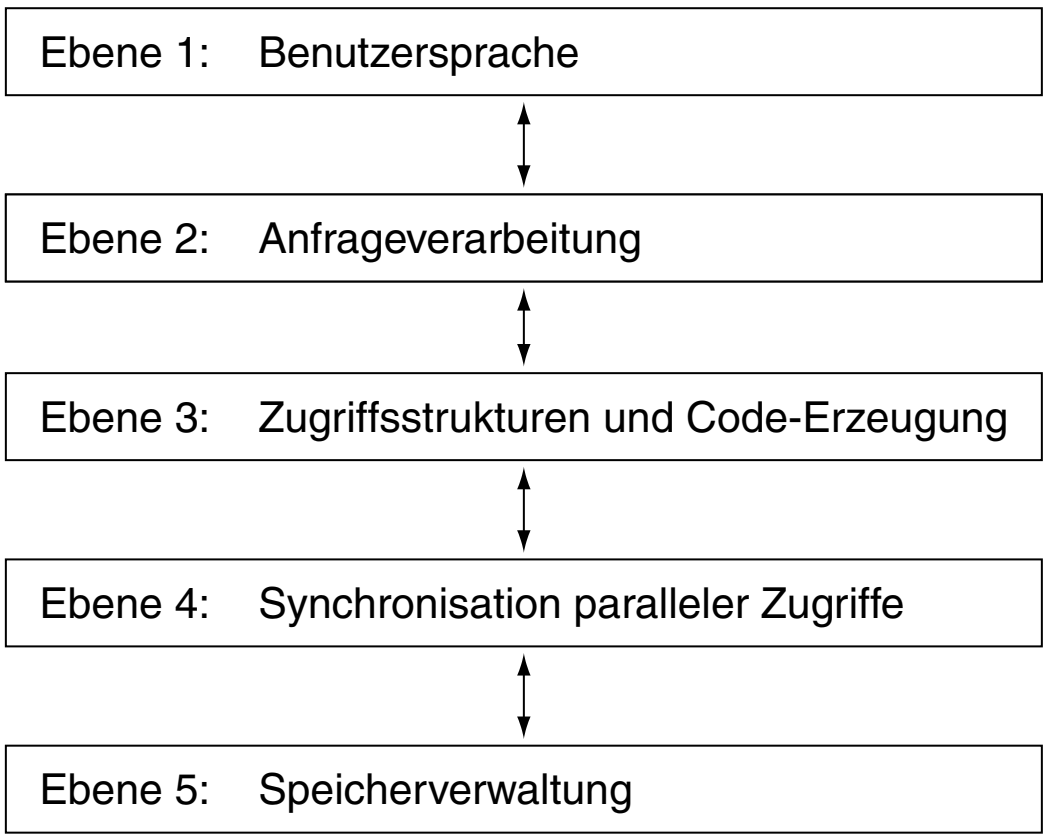
1. Überprüfen der Syntax.
2. Feststellen, ob die entsprechende Relation in der Datenbank definiert ist und ob der anfragende Benutzer deren Information lesen darf.
3. Feststellen, welche Operationen zur Beantwortung der Anfrage intern auszuführen sind und wie der Anfrageoperand, die Relation Buch, gespeichert ist.
4. Erstellen eines (effizienten) Programms zur Berechnung der Antwort.
5. Holen des Operanden aus der Datenbank.
6. Aufbereiten des Operanden zum Zweck der Ausgabe.
7. Sicherstellen, dass der Operand während der Ausführung dieses Programms nicht durch eine andere Operation verändert wird.

©STEIN 2004-2018

Datenbank-Management-Systeme

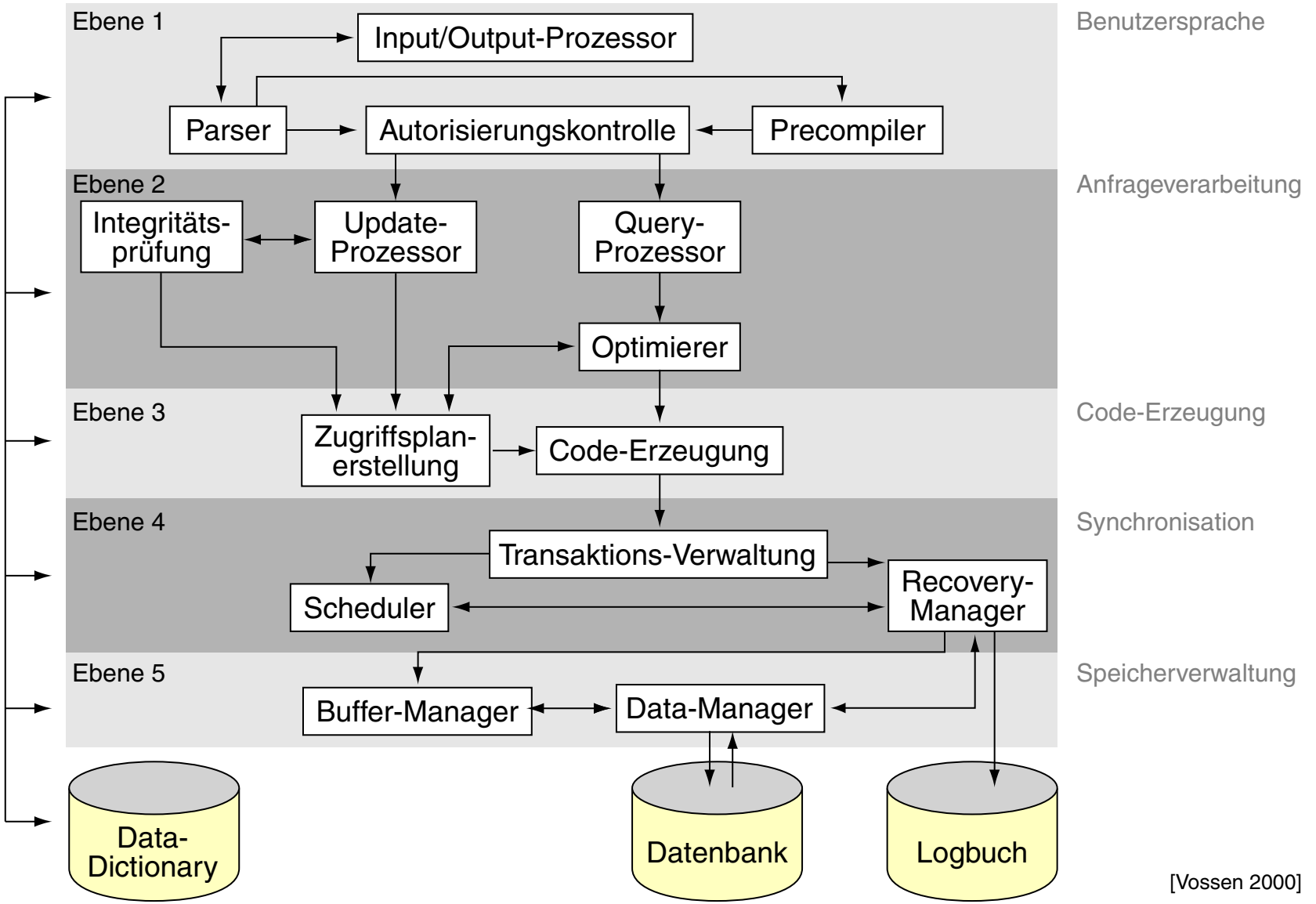
Systemarchitektur und Komponenten eines DBMS

➜ Unterscheidung von fünf Verarbeitungsschichten in einem DBMS:



©STEIN 2004-2018

Datenbank-Management-Systeme



[Vossen 2000]

©STEIN 2004-2018

Bemerkungen:

- Das Data-Dictionary enthält die drei Schemata der Ebenen der ANSI/SPARC-Architektur.
- Das Logbuch dient zum Zweck des Wiederanlaufs nach Systemfehlern.
- Der Parser führt die syntaktische Analyse der Kommandos durch.
- Der Precompiler ist u.a. für die Verarbeitung eingebetteter Kommandos zuständig.
- Update-Operationen erfordern eine Integritätsprüfung, um die semantische Korrektheit der Datenbank zu gewährleisten.
- Der Optimierer untersucht die Formulierung von Anfragen dahingehend, ob sie sich in eine effizientere Form bringen lassen.
- Die Zugriffsplanerstellung enthält das Finden und die Auswahl möglichst effizienter Zugriffspfade auf die benötigten Daten (Stichwort: Index).
- Bei der Code-Erzeugung wird der bisher generierte Zwischen-Code in eine Folge von Lese- und Schreibbefehlen für den Sekundärspeicher übersetzt.
- Die Transaktionsverwaltung erledigt die Synchronisation von quasi parallel ablaufenden Transaktionen mehrerer Benutzer. Jede Transaktion wird entweder vollständig oder gar nicht ausgeführt.
- Eine Transaktion, die nicht erfolgreich zu Ende gebracht werden kann, wird dem Recovery-Manager übergeben. Er setzt die Datenbank in den Zustand zurück, in dem sie sich vor dem Start der Transaktion befand; Basis ist das Logbuch.
- Buffer-Manager und Data-Manager realisieren die Speicherverwaltung des Systems. Der Buffer-Manager verwaltet im Hauptspeicher des Rechners den für jede Transaktion bereitgestellten Puffer; der Data-Manager verwaltet die dem DBMS zur Verfügung gestellten Hardware-Betriebsmittel.

©STEIN 2004-2018

Datenbank-Management-Systeme

Systemarchitektur und Komponenten eines DBMS (Fortsetzung)

Die Sprachschnittstellen eines DBMS [Ebene 1] :

- Datendefinitionssprache *(Data Definition Language)*, DDL

  Dient dem Benutzer, der die Schemata der Datenbank definieren möchte.
  Manchmal auch Unterscheidung einer Subschema-DDL (externe Ebene), einer Schema-DDL (konzeptuelle Ebene) und einer Data-Storage-Definition-Language, DSDL, (interne Ebene).

- Datenmanipulationssprache *(Data Manipulation Language)*, DML

  Sprache, die zum Arbeiten mit einer Datenbank zur Verfügung steht.
  Die „eigentliche“ Datenmanipulationssprache zur Änderung von gespeicherten Daten, zum Einfügen von neuen Daten und zum Löschen von gespeicherten Daten.
  Im Allgemeinen als deklarative (nicht-prozedurale) Sprache realisiert: der Benutzer spezifiziert, *was* für Daten gesucht werden, aber nicht *wie* sie gefunden werden sollen.

- Datenverwaltungssprache *(Data Administration Language)*, DAL

  Sprache zur Manipulation der internen Ebene.

©STEIN 2004-2018

Datenbank-Management-Systeme

Systemarchitektur und Komponenten eines DBMS (Fortsetzung)

Die Sprachschnittstellen eines DBMS [Ebene 1] :

- Datendefinitionssprache *(Data Definition Language)*, DDL

  Dient dem Benutzer, der die Schemata der Datenbank definieren möchte.
  Manchmal auch Unterscheidung einer Subschema-DDL (externe Ebene), einer Schema-DDL (konzeptuelle Ebene) und einer Data-Storage-Definition-Language, DSDL, (interne Ebene).

- Datenmanipulationssprache *(Data Manipulation Language)*, DML

  Sprache, die zum Arbeiten mit einer Datenbank zur Verfügung steht.
  Die „eigentliche“ Datenmanipulationssprache zur Änderung von gespeicherten Daten, zum Einfügen von neuen Daten und zum Löschen von gespeicherten Daten.
  Im Allgemeinen als deklarative (nicht-prozedurale) Sprache realisiert: der Benutzer spezifiziert, *was* für Daten gesucht werden, aber nicht *wie* sie gefunden werden sollen.

- Datenverwaltungssprache *(Data Administration Language)*, DAL

  Sprache zur Manipulation der internen Ebene.

©STEIN 2004-2018

Relationale Datenbanksysteme

©STEIN 2004-2018

Relationale Datenbanksysteme

Merkmale relationaler Datenbank-Management-Systeme (RDBMS)

- seit Mitte der 70er Jahre
- Daten in Tabellenstruktur
- Drei-Schichten-Schema-Architektur nach ANSI/SPARC
- standardisierte Datenbanksprache SQL *(Structured Query Language)*
- kontrollierter Mehrbenutzerbetrieb
- Gewährleistung von Datenschutz und Datensicherheit

- Werkzeuge
  - für die interaktive Definition/Anfrage/Darstellung von Daten
  - für die Erstellung von Benutzeroberflächen
  - für den Entwurf von Anwendungsprogrammen

©STEIN 2004-2018

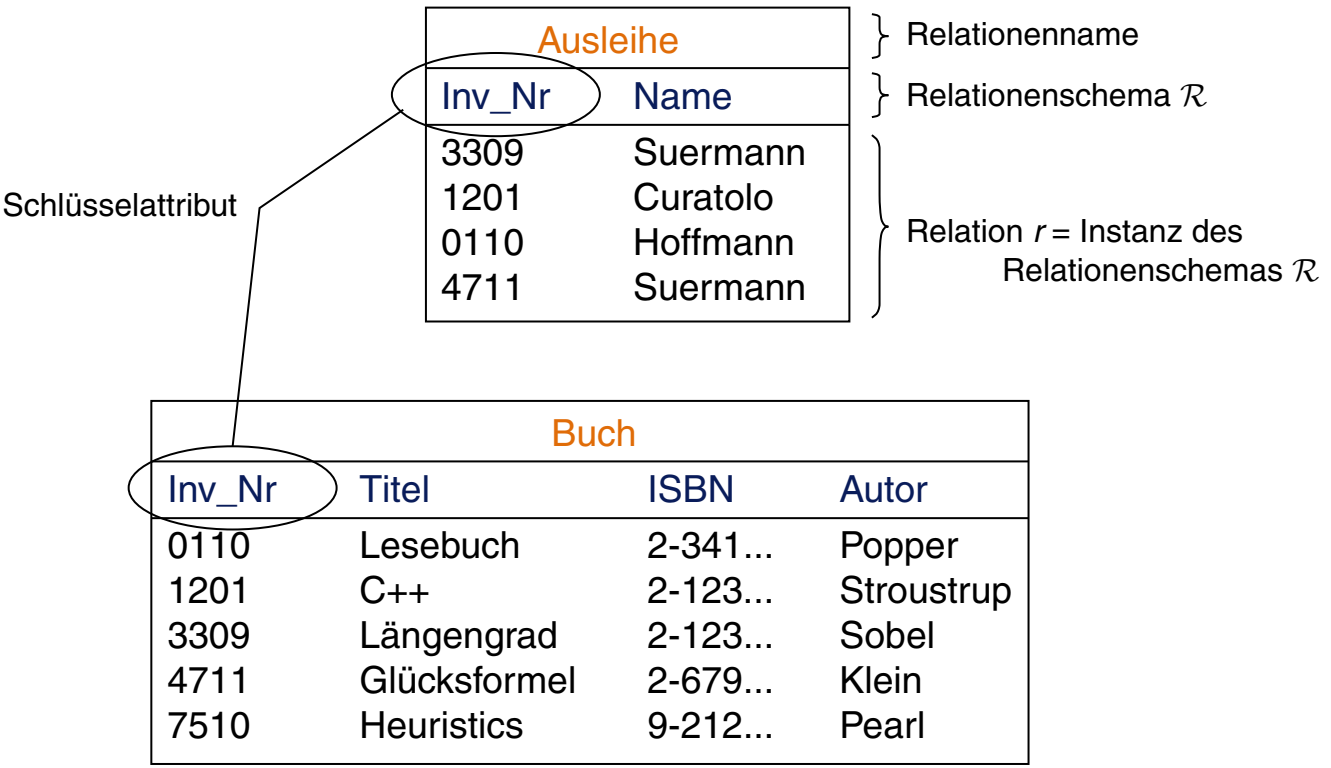
Bemerkungen:

- Beispiele für leistungsstarke RDBMS: IBM DB2, MS SQL-Server, MySQL, Oracle DB, PostgreSQL, Sybase
- Beispiele für „Pseudo“-RDBMS: MS Access, dBASE. Hier fehlen u.a. eine explizite Abbildung der Drei-Schichten-Schema-Architektur und leistungsfähige Algorithmen zur Optimierung.

©STEIN 2004-2018

Relationale Datenbanksysteme

Beispiel: Daten- und Datenbankdefinition



- Datenbank = Menge von Relationen
- Datenbankschema = Menge von Relationenschemata

©STEIN 2004-2018

Relationale Datenbanksysteme

Beispiel: Integritätsbedingungen

Integritätsbedingungen sind Konsistenzregeln.

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 Ausleihe:
  in Ausleihe existieren keine zwei Tupel mit demselben Inv_Nr-Wert.
- Attribut Inv_Nr ist (Primär)Schlüssel von Buch:
  in Buch existieren keine zwei Tupel mit demselben Inv_Nr-Wert.

globale Integritätsbedingung:

- Jeder Inv_Nr-Wert in der Relation Ausleihe muss in der Relation Buch auftreten und eindeutig sein. Stichwort: Fremdschlüsselbedingung

©STEIN 2004-2018

Relationale Datenbanksysteme

Beispiel: Anfrageoperationen

- Selektion = Auswahl von Zeilen bzw. Tupeln:

$\sigma_{\text{Name}='\text{Suermann}'}(\text{Ausleihe}) \quad \leadsto$

Inv_Nr	Name
3309	Suermann
4711	Suermann

- Projektion = Auswahl von Spalten bzw. Attributen:

$\pi_{\text{Inv_Nr,Titel}}(\text{Buch}) \quad \leadsto$

Inv_Nr	Titel
0110	Lesebuch
1201	C++
3309	Längengrad
4711	Glücksformel
7510	Heuristics

©STEIN 2004-2018

Relationale Datenbanksysteme

Beispiel: Anfrageoperationen (Fortsetzung)

- Verbund *(Natural Join)* = Verknüpfung von Relationen über gleichbenannte Spalten mit gleichen Werten:

$$\pi_{\text{Inv_Nr,Titel}}(\text{Buch}) \bowtie \sigma_{\text{Name}='\text{Suermann}'}(\text{Ausleihe})$$

$\rightsquigarrow$

Inv_Nr	Titel	Name
3309	Längengrad	Suermann
4711	Glücksformel	Suermann

- Vereinigung, Durchschnitt und Differenz von gleich strukturierten Relationen
- Umbenennung von Spalten bzw. Attributen

Die genannten Operationen lassen sich beliebig kombinieren.
Stichwort: Relationenalgebra

©STEIN 2004-2018

Relationale Datenbanksysteme

Beispiel: Anfrageoperationen (Fortsetzung)

Deklarativ-mengenorientierte Formulierung mit SQL:

```
select  Buch.Inv_Nr, Titel, Name
from    Buch, Ausleihe
where   Name = Suermann
        and
        Buch.Inv_Nr = Ausleihe.Inv_Nr
```

Weiterhin können definiert werden:

- Relationen und Sichten
- Konsistenzbedingungen
- Update-Operationen
- Trigger

©STEIN 2004-2018

Relationale Datenbanksysteme

Anfrageoptimierung

Gegeben sei ein Relationenalgebra-Ausdruck α.

Finde einen zu α äquivalenten Ausdruck β, der effizienter als α auswertbar ist.

Beispiel:

a) $\sigma_{\text{Attribut}=\text{Wert}}(r_1 \bowtie r_2)$

b) $\sigma_{\text{Attribut}=\text{Wert}}(r_1) \bowtie r_2$

$|r_1|$ = 100

$|r_2|$ = 50

Attribut $\in \mathcal{R}_1$

10 Tupel in r_1 erfüllen Attribut=Wert

©STEIN 2004-2018

Relationale Datenbanksysteme

Anfrageoptimierung

Gegeben sei ein Relationenalgebra-Ausdruck α.

Finde einen zu α äquivalenten Ausdruck β, der effizienter als α auswertbar ist.

Beispiel:

a) $\sigma_{\text{Attribut}=\text{Wert}}(r_1 \bowtie r_2)$

b) $\sigma_{\text{Attribut}=\text{Wert}}(r_1) \bowtie r_2$

$|r_1|$ = 100
$|r_2|$ = 50
Attribut $\in \mathcal{R}_1$
10 Tupel in r_1 erfüllen Attribut=Wert

Feststellung: a) und b) sind äquivalente Anfragen

Auswertungsaufwand:

zu a) 100·50 Join-Operationen + 100·50 Select-Operation = 10 000 Operationen

zu b) 100 Select-Operationen + 10·50 Join-Operationen = 600 Operationen

©STEIN 2004-2018

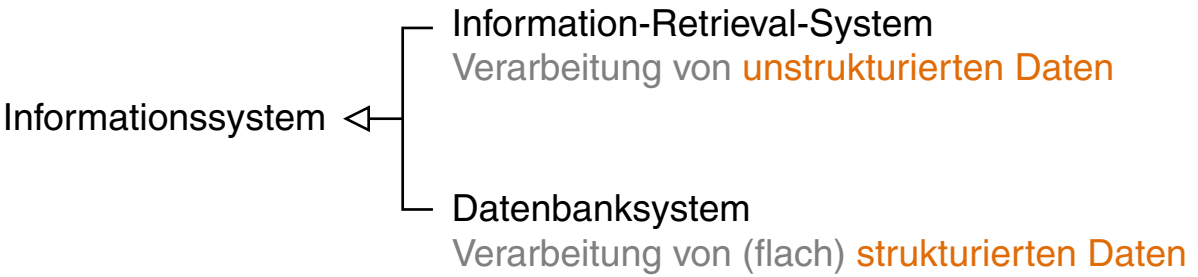
Bemerkungen:

- Algebraische Optimierung bedeutet die Umformung von Ausdrücken.
- Generelle Regel: Select-Operation vor Join-Operation durchführen.

©STEIN 2004-2018

Relationale Datenbanksysteme

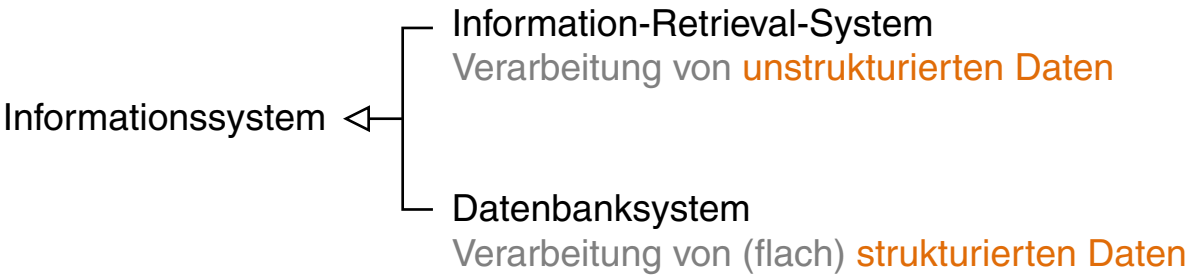
Grenzen von RDBMS



©STEIN 2004-2018

Relationale Datenbanksysteme

Grenzen von RDBMS



- Realzeitanwendungen für Steuer- und Regelaufgaben

  verlangen Reaktionszeitgarantien → Echtzeit-DBS

- Speicherung und Verarbeitung von Expertenwissen

  wenige Objekte, viele Objekttypen, Anfragen ≈ Inferenzoperationen → deduktive DBS

- Speicherung von Landkarten, annotierte Gebäudeinformationen

  komplex strukturierte Objekte, räumliche Informationen → Geo-Informationsysteme

- Verarbeitung von Zeitreihen

  Informationen über regelmäßig wiederkehrende Ereignisse (tägliche Umsatzzahlen, etc.)

- Data Mining

  Data Warehouse, Online Analytical Processing OLAP

©STEIN 2004-2018

Relationale Datenbanksysteme

Erstellung von RDBMS

Aufgabe	Tätigkeit	Sprache, Werkzeug
konzeptueller Entwurf	Beschreibung der Daten unabhängig von der Realisierung	Entity-Relationship-Modell, UML
relationaler Entwurf	Umsetzung in Relationendefinitionen	Relationenmodell
Datendefinition	Beschreibung der Relationen in einem SQL-Dialekt	SQL, DDL
Implementierung	Realisieren von Anfragen und Änderungen in SQL	SQL, DQL und DML
Qualitätssicherung	Analyse, Beweis, Dokumentation	formale Kriterien für Konsistenz, Effizienz, Wartbarkeit

©STEIN 2004-2018

Bemerkungen:

- Die Begriffe „Datenbankentwurf“, „konzeptueller Datenbankentwurf“ und „konzeptioneller Datenbankentwurf“ werden in der Literatur oft synonym benutzt.
- Die industrielle und universitäre Forschung beschäftigt sich mit der Weiterentwicklung klassischer Datenbanktechnologie, um die genannten Nicht-Standardaufgaben adäquat – d.h. effizient und auf die Daten abgestimmt – zu unterstützen.

©STEIN 2004-2018