Teil 1: Einführung

Literatur:

- Elmasri/Navathe: Fundamentals of Database Systems, 3. Auflage, 1999.
  Chapter 1, "Databases and Database Users"
  Chapter 2, "Database System Concepts and Architecture"
- Kemper/Eickler: Datenbanksysteme, 3. Auflage, 1999.
  Kapitel 1: "Einleitung und Übersicht"
- Heuer/Saake: Datenbanken, Konzepte und Sprachen, Thomson, 1995.
- Lipeck: Skript zur Vorlesung Datenbanksysteme, Univ. Hannover, 1996.
- Silberschatz/Korth/Sudarshan: Database System Concepts, 3. Auflage.
  Chapter 1: "Introduction".
- Fry/Sibley: Evolution of data-base management systems.
  ACM Computing Surveys 8(1), 7–42, 1976.
- Steel: Interim report of the ANSI-SPARC study group.
  In ACM SIGMOD Conf. on the Management of Data, 1975.
- Codd: Relational database: a practical foundation for productivity.
  Communications of the ACM, Vol. 25, Issue 2, (Feb. 1982), 109–117.
- Silberschatz/Stonebraker/Ullman (Eds.): Database systems: achivements and opportunities. Communications of the ACM, Vol. 34, Issue 10, (Okt. 1991), 110–120.
- Silberschatz/Stonebraker/Ullman: Database research: achivements and opportunities into the 21st century. ACM SIGMOD Record, Vol. 25, Issue 1, (März 1996), 52–63.

Lernziele

Nach diesem Kapitel sollten Sie Folgendes können:

- Grundbegriffe erklären: Datenbankzustand, Schema, Anfrage, Update, Datenmodell, DDL/DML
- Die Rolle und den Nutzen eines DBMS erklären
- Datenunabhängigkeit, Deklarativität und die Drei-Schema-Architektur erklären
- Einige DBMS-Anbieter und DBMS-Tools nennen
- Unterschiedliche Nutzergruppen von Datenbankanwendungssystemen nennen

Inhalt

Aufgabe einer Datenbank (1)

- Was ist eine Datenbank? Schwierige Frage. Es gibt keine präzise und allgemein anerkannte Definition.
- Naive Näherung: Die Hauptaufgabe eines Datenbanksystems (DBS) ist die Beantwortung gewisser Fragen über eine Teilmenge der realen Welt, z.B.

Welche Hausaufgaben hat Lisa Weiss gemacht? → Datenbank-system → 1, 2

Aufgabe einer Datenbank (2)

- Das DBS dient nur als Speicher für Informationen. Die Informationen müssen zuerst eingegeben und dann immer aktualisiert werden.

  Lisa Weiss hat Hausaufgabe 3 gemacht und 10 Punkte erhalten. ⟶ Datenbank-system ⟶ ok.

- Ein Datenbanksystem ist eine Computer-Version eines Karteikastens (nur mächtiger).
- Eine Tabellenkalkulation ist (fast) ein kleines DBS.

Aufgabe einer Datenbank (3)

- Normalerweise machen Datenbanksysteme keine besonders komplizierten Berechnungen auf den gespeicherten Daten.

  Aber es gibt z.B. Wissensbanken und Data Mining-Werkzeuge.

- Sie können jedoch die gesuchten Daten schnell in einer großen Datenmenge finden (Gigabytes oder Terabytes – größer als der Hauptspeicher).
- Sie können auch Daten aggregieren/kombinieren, um eine Antwort aus vielen Einzeldaten zusammenzusetzen (z.B. Summe der Punkte von Lisa Weiss).

Aufgabe einer Datenbank (4)

- Obige Frage "Welche Hausaufgaben hat Lisa Weiss gemacht?" war in natürlicher Sprache (Deutsch).
- Das Verstehen natürlicher Sprache durch Computer ist ein schwieriges Problem (Missverständnisse).
- Daher werden Fragen ("Anfragen") normalerweise in formaler Sprache gestellt, heute meist in SQL.

  Man kann SQL als spezielle Programmiersprache für Anfragen an Datenbanken verstehen. Im Gegensatz zu Sprachen wie Pascal, C oder Java kann man in SQL aber keine beliebigen Algorithmen notieren.

- Einige DBS erlauben natürlichsprachliche Anfragen.

Zustand, Anfrage, Update

- Menge der gespeicherten Daten = DB-Zustand:

  Aktueller Zustand —Anfrage⟶ Antwort: 1, 2

  ```
  SELECT AUFGNR FROM ABGABEN
  WHERE STUDENT = 'Lisa Weiss'
  ```

- Eingabe, Modifikation oder Löschung von Daten ändert den Zustand:

  Alter Zustand —Update⟶ Neuer Zustand

  ```
  INSERT INTO ABGABEN
  VALUES ('Lisa Weiss', 3, 10)
  ```

Strukturierte Information (1)

- Jede Datenbank kann nur Informationen einer vorher definierten Struktur speichern:

  Heute gibt es Pizza in der Mensa. ⟶ Hausaufgaben-DBS ⟶ Fehler.

- Da die Daten strukturiert sind (nicht nur Text), sind komplexere Auswertungen möglich, z.B.:

  Gib für jede Hausaufgabe aus, wie viele Studenten sie bearbeitet haben, sowie die durchschnittlich erreichte Punktzahl.

Strukturierte Information (2)

- Eigentlich speichert ein DBS nur Daten (Zeichenketten, Zahlen), und keine Informationen.
- Daten werden durch Interpretation zu Information.
- Begriffe wie Studenten und Aufgaben müssen dem System formal bekanntgemacht (deklariert) werden, sowie für den Nutzer erklärt/definiert werden.

  Natürlich ist es möglich, eine Datenbank zu entwickeln, in der beliebige Texte gespeichert werden können. Dann kann aber das DBS nur nach Substrings suchen und keine komplexere Anfragen bearbeiten. Je mehr ein DBS über die Struktur der Daten "weiß", desto besser kann es den Nutzer bei der Suche und Auswertung unterstützen.

Zustand vs. Schema (1)

Datenbank-Schema:

- Formale Definition der Struktur des DB-Inhalts.
- Führt Namen (Bezeichner) ein, mit denen man auf die Daten zugreifen kann (z.B. `STUDENT`, `AUFGNR`).
- Legt mögliche Datenbankzustände fest.

  Hierzu gehört insbesondere Datentyp-Information. Z.B. ist `AUFGNR` eine positive ganze Zahl `1..99`.

- Nur einmal definiert (wenn die DB erstellt wird).

  In der Praxis kann es manchmal notwendig sein, das Schema zu verändern. Das passiert jedoch selten und ist problematisch.

Zustand vs. Schema (2)

Datenbank-Zustand (Ausprägung eines Schemas):

- Beinhaltet die aktuellen Daten, dem Schema entsprechend strukturiert.
- Legt konkrete Werte für die Schema-Elemente fest.

  Z.B. gibt es einen Eintrag mit `STUDENT = 'Lisa Weiss'`, `AUFGNR = 1` und `PUNKTE = 10`.

- Ändert sich oft (wenn Datenbank-Informationen geändert werden).
- Die Datenbank speichert jeweils nur einen (den aktuellen, derzeitigen) Zustand.

Zustand vs. Schema (3)

- Im relationalen Datenmodell sind die Daten in Form von Tabellen (Relationen) strukturiert.
- Jede Tabelle hat: Name, Folge von benannten Spalten (Attribute) und Menge von Zeilen (Tupel)

ABGABEN		
STUDENT	AUFGNR	PUNKTE
Lisa Weiss	1	10
Lisa Weiss	2	8
Daniel Sommer	1	9
Daniel Sommer	2	9

DB-Schema (Tabellenname und Spaltenköpfe)

DB-Zustand (Zeilen)

Übung

- Der Professor möchte ein Programm entwickeln, das an jeden Studenten eine E-Mail folgender Art verschickt:

```
Sehr geehrte Frau Weiss,
folgende Bewertungen sind über Sie gespeichert:
    - Aufgabe 1: 10 Punkte
    - Aufgabe 2:  8 Punkte
Melden Sie sich bitte, falls ein Fehler vorliegt.
Mit freundlichen Grüßen, ...
```

- Muss er dazu obige Tabelle erweitern? Sollte er vielleicht die Daten auf mehrere Tabellen verteilen?

Datenmodell (1)

- Ein Datenmodell bestimmt die Konzepte, die zur Strukturierung der Daten (also zur Definition von Datenbank-Schemata) verwendet werden können.
- Ein Datenmodell legt also folgendes fest:
  - ◇ Was sind mögliche Datenbank-Schemata?

    Ein Schema ist im wesentlichen ein Text in einer formalen Sprache (Computer-Sprache), alternativ auch ein Diagramm (graphische Syntax), oder mathematische Strukturen (abstrakte Syntax).
  - ◇ Was sind die möglichen Datenbank-Zustände (Instanzen) zu einem gegebenen Schema?

    Die Menge der möglichen Zustände ist die Bedeutung (Semantik) eines Schemas.

Datenmodell (2)

- Manche Wissenschaftler sind der Ansicht, daß auch Basisoperationen für Anfragen und Updates zum Datenmodell dazu gehören.

  Z.B. wird beim relationalen Modell gefordert, daß man in Anfragen Tabellen auch verknüpfen kann, nicht nur Anfragen an einzelne Tabellen stellen. Es gibt leider keine allgemein anerkannte formale Definition von Datenmodell.
- Die meisten Datenbankler würden auch sagen, daß syntaktische Feinheiten der Definition von Schemata für das Datenmodell nicht wichtig sind.

  Wichtig sind nur die Konzepte (z.B. beim relationalen Modell die Strukturierung in Form von Tabellen). Das macht den Begriff "fuzzy".

Datenmodell (3)

- Data Definition Language (DDL): Sprache zur Definition von Datenbank-Schemata.
- Data Manipulation Language (DML): Sprache für Anfragen und Updates.

  SQL, die Standardsprache für das relationale Modell, kombiniert DDL und DML. Der Anfrage-Teil der DML wird Anfragesprache (QL) genannt. Er ist meist komplizierter als der Update-Teil. Aber Updates können auch Anfragen enthalten (um neue Werte zu ermitteln).
- Manchmal wird der Begriff "Datenmodell" statt "Datenbank-Schema" verwendet.

  Z.B. Unternehmens-Datenmodell: Schema für alle Daten einer Firma.

Datenmodell (4)

Beispiele für Datenmodelle:

- Relationales Modell
- Entity-Relationship-Modell (viele Erweiterungen)
- Objekt-Orientiertes Datenmodell (z.B. ODMG)
- Objekt-Relationales Datenmodell
- XML (DTDs, XML Schemata)
- Netzwerk-Modell (CODASYL) (historisch)
- Hierarchisches Modell (historisch)

Inhalt

DBMS (1)

Ein Datenbankmanagementsystem (DBMS) ist ein anwendungsunabhängiges Softwarepaket, das ein Datenmodell implementiert, d.h. Folgendes ermöglicht:

- Definition eines DB-Schemas für eine konkrete Anwendung,

  Da das DBMS anwendungsunabhängig ist, speichert es das Schema normalerweise auf der Festplatte, oft zusammen mit dem DB-Zustand (in speziellen "System-Tabellen").
- Speichern eines DB-Zustands, z.B. auf Festplatte,
- Abfragen an den aktuellen DB-Zustand,
- Änderung des DB-Zustands.

DBMS (2)

- Natürlich verwenden normale Nutzer kein SQL für den täglichen Umgang mit einer Datenbank.
- Sie arbeiten mit Anwendungsprogrammen, die eine bequemere / einfachere Benutzerschnittstelle für Standard-Aufgaben bieten.

  Z.B. ein Formular, in dem Felder ausgefüllt werden können.
- Intern enthält das Anwendungsprogramm jedoch ebenfalls SQL-Befehle (Anfragen, Updates), um mit dem DBMS zu kommunizieren.

DBMS (3)

- Oft greifen eine ganze Reihe verschiedener Anwendungsprogramme auf eine zentrale Datenbank zu.
- Beispiel: Anwendungsprogramme für Punkte-DB:
  - ◇ Web-Interface für Studenten.

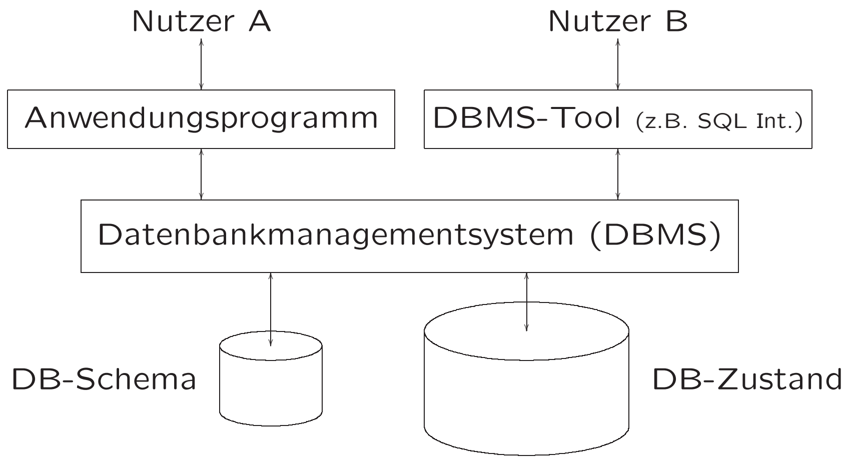
    Mit Funktionen zum Eintragen, Anschauen der Punkte, ...
  - ◇ Programm zum Eintragen von Punkten für Klausur und Hausaufgaben (für Übungsleiter/Tutor).
  - ◇ Programm, das einen Bericht (Übersicht) für den Professor ausdruckt, um Noten zu vergeben.

DBMS (4)

- Mit einem DBMS wird normalerweise eine interaktive SQL-Schnittstelle mitgeliefert:
  - ◇ Man kann einen SQL-Befehl (z.B. Anfrage) eintippen
  - ◇ und bekommt dann das Ergebnis (Tabelle) auf dem Bildschirm angezeigt.
- Dieses Programm kommuniziert mit dem DBMS wie andere Anwendungsprogramme auch.

  Wenn man will, kann man so ein Programm auch selbst schreiben. Es schickt die SQL-Befehle als Zeichenketten an das DBMS.

DBMS (5)



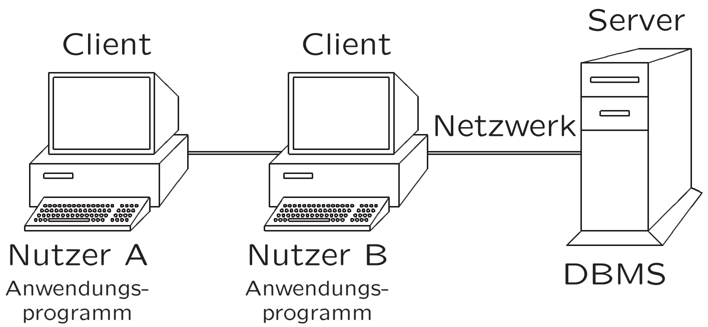
DB-Anwendungssysteme (1)

- Oft greifen verschiedene Nutzer gleichzeitig auf die gleiche Datenbank zu.
- Das DBMS ist normalerweise ein im Hintergrund laufender Server-Prozess (eventuell auch mehrere), auf den über das Netzwerk mit Anwendungsprogrammen (Clients) zugegriffen wird.

  Einem Web-Server sehr ähnlich. Für einige kleine PC-DBMS gibt es nur ein einziges Programm, das als DBMS und als Interpreter für die Anwendungsprogramme dient.
- Man kann das DBMS als Erweiterung des Betriebssystems sehen (leistungsfähigeres Dateisystem).

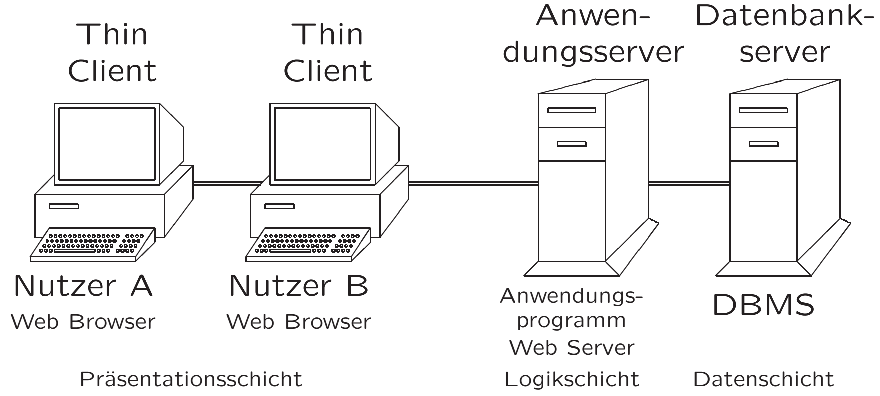
DB-Anwendungssysteme (2)

Client-Server-Architektur:



DB-Anwendungssysteme (3)

Dreischichten-Architektur (Three-Tier Architecture):



DB-Anwendungssysteme (4)

Übung:

- Betrachten Sie die Datenbank einer Bank zur Verwaltung von Girokonten.
- Für welche Aufgaben benötigt die Bank Anwendungsprogramme, die auf die Datenbank zugreifen?
  - ◇ ______________________
  - ◇ ______________________
  - ◇ ______________________
  - ◇ ______________________

DB-Anwendungssysteme (5)

Etwas Datenbank-Vokabular:

- Eine DB besteht aus DB-Schema und DB-Zustand.

  Z.B. sagt man "Hausaufgaben-Datenbank". Es hängt vom Kontext ab, ob man den derzeitigen Zustand meint, oder nur das Schema und den Speicherplatz oder -ort (Netzwerk-Adresse des Servers).
  Es ist falsch, eine einzige Tabelle oder Datei Datenbank zu nennen, außer sie beinhaltet alle Daten des Schemas.

- Ein Datenbank-System (DBS) besteht aus einem DBMS und einer Datenbank.

  Aber Datenbank-System wird auch als Abkürzung für DBMS genutzt.

- Ein Datenbank-Anwendungssystem besteht aus einem DBS und Anwendungsprogrammen.

Inhalt

Persistente Speicherung (1)

- Die Hauptaufgabe von Datenbanksystemen ist es, Daten persistent (dauerhaft) zu speichern:
  - ◇ Daten heißen persistent, wenn sie länger leben als eine einzelne Programm-Ausführung.
  - ◇ Die Daten sind auch noch da, wenn man den Rechner aus- und wieder eingeschaltet hat.
- Normalerweise werden persistente Daten auf der Festplatte des Rechners gespeichert.

  Der Hauptspeicher des Rechners ist dagegen flüchtig: Sein Inhalt ist bei einem Stromausfall verloren.

Persistente Speicherung (2)

- Natürlich könnte man die Daten auch direkt in Dateien auf der Festplatte speichern, sie wären dann auch persistent.

  Allerdings sorgt die Datenbank bei richtiger Installation automatisch für ständige Sicherungskopien auf einer zweiten Platte, die Daten sind dann also noch etwas "persistenter".

- Die Dateien werden vom Betriebssystem des Rechners verwaltet (z.B. Windows, Linux, UNIX).
- Aus Sicht des Betriebssystems sind Dateien aber nur Folgen von Bitmustern (Bytes), z.B. `01001011`.

Persistente Speicherung (3)

- Die Bedeutung und Struktur der Daten wird durch Programme bestimmt, die die Dateien erstellen, lesen und verändern.
- Nicht selten ist die Struktur der Daten in der Datei (das Dateiformat) nur den Autoren des Programms bekannt (nicht offiziell dokumentiert).
- Die Datei kann dann nur von diesem einen Programm verarbeitet werden.

  Fehler im Programm können leicht zur Zerstörung der Daten führen, da keine unabhängige Kontrolle stattfindet.

Persistente Speicherung (4)

- Datenbank-Managementsysteme legen die Daten normalerweise auch in Betriebssystem-Dateien ab.

  Bei besonderen Leistungsanforderungen können einige DBMS auch direkt auf die "Raw Devices" zugreifen (an Teilen des Betriebssystems vorbei).
- Datenbank-Managementsysteme sind Mittler zwischen Programmen und den Dateien.

  Das DBMS ist gewissermaßen eine Schicht über dem Betriebssystem. Anwendungsprogramme greifen nur über das DBMS auf die Datenbank-Dateien zu.
- Das ist besonders wichtig, wenn mehrere Programme auf die gleichen Daten zugreifen.

Persistente Speicherung (5)

- Das DBMS kennt die Struktur der Daten. Es macht diese Information auch den Benutzern verfügbar.

  Weil es die Struktur der Daten kennt, kann es Operationen verhindern, die diese Struktur zerstören würden. Die Datenstruktur ist jetzt nicht mehr nur in den Köpfen der Programmierer.
- Der Vorteil für die Programme ist, daß sie mit den Daten auf höherer Abstraktionsebene arbeiten können: Tabellen statt Folgen von Bytes.
- Das DBMS hat auch viele Verfahren eingebaut, die der Programmierer sonst selbst schreiben müßte (etwa zum Suchen und Sortieren).

"Datenunabhängigkeit"

- Entkopplung von Programmen und Daten.
- Daten sind eine unabhängige Ressource.

  Früher hatten sie nur im Zusammenhang mit den Anwendungsprogrammen eine Bedeutung, mit denen sie ursprünglich erfasst wurden.
- (Physische) Datenunabhängigkeit:
  - ◇ Programme sollten nicht von Speichermethoden (Datenstrukturen) abhängen.
  - ◇ Umgekehrt werden die Dateistrukturen nicht durch die Programme festgelegt.

  ⇒ Schützt Investitionen in Programme und Daten.

Deklarative Sprachen (1)

- SQL ist eine deklarative Abfragesprache:
  Man beschreibt nur,
  - was (welche Information) gesucht wird,
    aber nicht,
  - wie diese Information berechnet werden soll.
- Man gibt nur Bedingungen für die Daten an:

```
SELECT X.PUNKTE
FROM   ABGABEN X   -- X ist Tabellenzeile
WHERE  X.STUDENT = 'Lisa Weiss'
AND    X.AUFGNR = 3
```


Deklarative Sprachen (2)

- Das DBMS wählt dann selbst ein effizientes Verfahren, um die gesuchten Daten zu bestimmen.

  Das Verfahren hängt von den genauen Datenstrukturen ab, die intern zur Speicherung der Daten verwendet werden. Insofern verlangt physische Datenunabhängigkeit eine deklarative Sprache.
- Häufig leichtere Formulierungen: Der Nutzer muss nicht über eine effiziente Auswertung nachdenken.

  Es kommt nur darauf an, daß man logisch korrekt beschreibt, was man haben will. Um die effiziente Auswertung kümmert sich das DBMS.
- Oft viel kürzer als klassische Programmierung. Daher ist z.B. die Programmentwicklung billiger.

Deklarative Sprachen (3)

- Der Benutzer schreibt in deklarativen Sprachen kein bestimmtes Auswertungsverfahren vor.
- Daher ergibt sich eine größere Unabhängigkeit von aktueller Hardware/Software-Technologie:
  - Einfache Parallelisierung
  - Auch heutige Anwendungsprogramme profitieren von neuen Verfahren in zukünftigen DBMSen.

    Die Anfrage im Anwendungsprogramm braucht nicht geändert zu werden. Die neue Version des DBMS wertet sie nur anders (effizienter) aus, aber die Ergebnisse bleiben selbstverständlich gleich (bei gleichem Datenbank-Zustand).

Ad-hoc Anfragen

- In vor-relationaler Zeit mußte man, wenn man eine neue Auswertung der Daten brauchte, einen Antrag an die IT-Abteilung stellen.

  Dort haben die Programmierer dann ein entsprechendes Programm geschrieben, das durch die Daten navigierte, in Schleifen viele Datensätze verarbeitete, und am Ende die gewünschten Werte berechnete. So ca. drei Wochen später hatte man dann seine Ergebnisse.
- Heute kann man (z.B. nach dieser Vorlesung) selbst eine SQL-Anfrage schreiben und erhält sofort seine Ergebnisse.

  Eventuell sollte man das nicht zu den Haupt-Geschäftszeiten machen, wenn die Datenbank schon voll ausgelastet ist.

Redundanz-Vermeidung (1)

- Daten heißen redundant, wenn sie aus anderen Daten und Wissen über die Anwendung ableitbar sind.
  - ◇ D.h. Daten werden doppelt gespeichert.

    Auch Zusammenfassungen sind redundant (überflüssig), wenn die Einzeldaten vorliegen (z.B. Summe der HA-Punkte pro Student).
- Probleme:
  - ◇ Doppelter Aufwand für Datenerfassung/Updates
  - ◇ Irgendwann vergisst man, eine der Kopien zu ändern (Daten werden inkonsistent)

    Oder Fehler gleich beim Eintragen: Passt nicht zusammen.
  - ◇ Verschwendet Speicherplatz

Redundanz-Vermeidung (2)

- Manchmal sind redundante Daten aber bequem:
  - ◇ Man braucht die Formel zu ihrer Berechnung nicht jedesmal neu eintippen.
  - ◇ Unterschiedliche Benutzer hätten die Daten gern verschieden strukturiert (sie brauchen auch nicht immer alle Daten).

    Z.B. braucht das Prüfungsamt nur die Information, ob 50% der Hausaufgabenpunkte erreicht wurden. Die Punkte für die einzelnen Aufgaben sind nicht relevant. Manchmal soll ein Benutzer auch nur eine globale Statistik einsehen dürfen (z.B. Durchschnittspunktzahl, aber nicht die Ergebnisse einzelner Studierender). Diese Statistik ist aber redundant, weil sie aus den einzelnen Ergebnissen, die auch in der Datenbank stehen, zu berechnen ist.

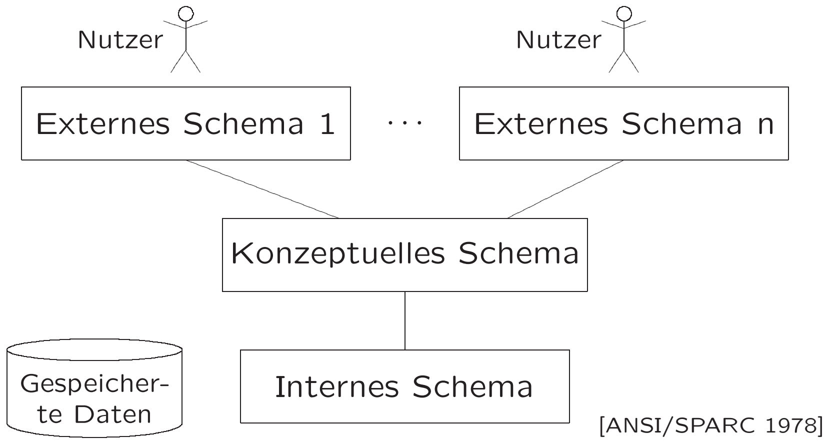
Redundanz-Vermeidung (3)

- Man kann verschiedene Sichten auf die Datenbank für verschiedene Benutzer zu definieren.
- Sichten sind virtuelle Tabellen:
  - ◇ Sie sind nicht wirklich abgespeichert (deswegen ist Redundanz kein Problem).
  - ◇ Aber man kann sie in Abfragen so verwenden, als wären es normale Tabellen.
- Sichten sind intern über Abfragen definiert:
  - ◇ Im relationalen Modell ist das Ergebnis einer Abfrage wieder eine Tabelle (Abschluss).

Drei-Schema Architektur (1)

- Es wurde eine Architektur mit drei Schema-Ebenen (Schichten) vorgeschlagen:
  - ◇ Jeder Benutzer (oder jede Anwendung) hat ein eigenes externes Schema (eigene Sicht auf die Daten: Teilmenge, umstrukturiert/aggregiert).
  - ◇ Das konzeptuelle (oder konzeptionelle) Schema ist die logische Gesamtsicht auf die Daten.
  - ◇ Das interne Schema beschreibt Datenstrukturen, die intern zur Abspeicherung der Daten genutzt werden (nur wichtig für “Performance Tuning”).

Drei-Schema Architektur (2)



[ANSI/SPARC 1978]

Weitere DBMS-Funktionen (1)

Transaktionen:

- Folge von DB-Befehlen, die als atomare Einheit ausgeführt werden ("ganz oder gar nicht")

  Wenn das System während einer Transaktion abstürzen sollte, werden alle Änderungen rückgängig gemacht ("roll back"), sobald das DBMS neu startet. Stürzt das System nach einer Transaktion ab ("commit"), sind alle Änderungen garantiert im DB-Zustand gespeichert.

- Unterstützung von Backup und Recovery

  Daten vollendeter Transaktionen sollten Plattenfehler überleben.

- Unterstützung von gleichzeitigen Nutzern

  Nutzer/Programmierer sollen nicht über parallele Zugriffe anderer Nutzer nachdenken müssen (z.B. durch automatische Sperren).

Weitere DBMS-Funktionen (2)

Sicherheit:

- Zugriffsrechte: Wer darf was womit machen?

  Es ist sogar möglich, Zugriffe nur für einen Teil der Tabelle oder nur für aggregierte Daten zuzulassen.

- Auditing: Das DBMS speichert ab, wer was mit welchen Daten gemacht hat.

Integrität:

- Das DBMS prüft, ob die Daten plausibel bzw. konsistent sind. Es lehnt Updates ab, die definierte Geschäftsregeln verletzen würden.

Weitere DBMS-Funktionen (3)

Data Dictionary / Systemkatalog (Meta-Daten):

- Informationen über Daten (z.B. Schema, Nutzer, Zugriffsrechte) in Systemtabellen verfügbar, z.B.:

SYS_TABLES	
TABLE_NAME	OWNER
ABGABEN	BRASS
SYS_TABLES	SYS
SYS_COLUMNS	SYS

SYS_COLUMNS		
TABLE_NAME	SEQ	COL_NAME
ABGABEN	1	STUDENT
ABGABEN	2	AUFGNR
ABGABEN	3	PUNKTE
SYS_TABLES	1	TABLE_NAME
SYS_TABLES	2	OWNER
SYS_COLUMNS	1	TABLE_NAME
SYS_COLUMNS	2	SEQ
SYS_COLUMNS	3	COL_NAME

Inhalt

DBMS-Anbieter (1)

- Oracle: Oracle 11g

  Enterprise Edition, Standard Edition, Standard Edition One, Express Edition, Lite Edition. [http://www.oracle.com/database/index.html]

- IBM: DB2 Universal Database V9 (plus z.B. IMS)

  [http://www.ibm.com/software/data/db2/]

- Microsoft: SQL Server 2005 (plus Access, FoxPro)

  SQL Server entstand 1988 als Portierung der Sybase-DB. Später als selbständiges DBMS weiterentwickelt (1994 formales Ende der Partnerschaft). Testversion: [www.microsoft.com/sql/evaluation/trial/]

- Sybase: Adaptive Server Enterprise 15 (plus . . .)

  Gratis Express Edition f. Linux: [http://www.sybase.com/linux/ase/]

DBMS-Anbieter (2)

- Informix: z.B. Informix Dynamic Server 11

  Informix wurde 2001 von IBM gekauft (für $ 1000 Mio). [http://www.ibm.com/software/data/informix/]

- Transaction Software: Transbase

  [http://www.transaction.de/products/tb/overview/?lang=en]

- Gupta SQLBase [http://www.guptaworldwide.com/]
- Borland InterBase [http://www.borland.com/interbase/]
- Pervasive SQL [http://www.pervasive.com/psql/]

DBMS-Anbieter (3)

Relationale DBMS 2006 [Gartner Dataquest-Studie]

Anbieter	Mio $	Marktanteil	Wachstum
Oracle	7186	47.1%	14.9%
IBM	3204	21.1%	8.8%
Microsoft	2654	17.4%	28.0%
Teradata	494	3.2%	5.7%
Sybase	487	3.2%	8.2%
Andere	1206	7.9%	5.0%
Gesamt	15 Mrd$	100.0%	14.2%

Quelle: [http://www.gartner.com/it/page.jsp?id=507466]
Marktanteil bezieht sich auf Einnahmen (Lizenzen, Updates, Support, Hosting). Platform: UNIX 34.8%, Windows Server 34.5%, Linux 15.5%.

Open Source-DBMS

- MySQL [http://dev.mysql.com]
- PostgreSQL [http://www.postgresql.org]
- MaxDB [http://www.mysql.com/products/maxdb/]

  Wurde vorher SAP DB genannt, und davor war es Adabas.
- IBM Cloudscape

  [http://publib.boulder.ibm.com/infocenter/cldscp10/index.jsp]
- Firebird [http://firebird.sourceforge.net/]

  Das war Borland InterBase. Borland entwickelt immer noch eine kommerzielle Version.
- CA Ingres [http://opensource.ca.com/projects/ingres/]

Größte Datenbanken (1)

- "TopTen^{TM} Program" der Winter Corporation

  [http://www.wintercorp.com/], hier Ergebnisse für 2005.
- Größtes Data Warehouse: Yahoo! (100.4 TB)

  Gewinner der Kategorie: DB Size, All Environments, Data Warehouse. Bei der Größe zählen Tabellendaten, Aggregate, Zusammenfassungen, Indexe, aber keine redundanten Kopien von Daten. Die Datenbank hat 385 Milliarden Tabellenzeilen. Ausstattung der Yahoo! Datenbank: Oracle, UNIX, Fujitsu Siemens PrimePower System, EMC Storage. Unter den ersten zehn sind Oracle (2∗), Oracle RAC (2∗), AT&T Daytona (2∗, beides AT&T Datenbanken), IBM DB2 (2∗), SQL Server, Sybase IQ. Sieben der Systeme laufen unter UNIX, zwei unter Linux, eins unter Windows. Die Rechnerhardware stammt von HP (4∗), IBM (2∗), Sun (2∗), Fujitsu Siemens, Unisys. Die Plattensysteme stammen von EMC (4∗), HP (4∗), Hitachi, Sun.

Größte Datenbanken (2)

- Größtes OLTP System:

  Land Registry for England and Wales: 23.1 TB

  OLTP ("OnLine Transaction Processing"): Klassische Transaktionsdatenbank (Gegensatz zu "Data Warehouse").
  Ausstattung: IBM DB2 Universal Database für z/OS, IBM eServer z990, IBM und Hitachi Storage.
  Zweiter: United States Patent and Trademark Office (16 TB).
  Dritter: Elsevier (Verlag) (9 TB). Vierter: UPS (9 TB).
  DBMS der ersten zehn: IBM DB2 (2∗), Oracle (2∗), Oracle RAC, SQL Server (3∗), Sybase ASE, CA-Datacom.
  Betriebssystem: UNIX (4∗), z/OS (3∗), Windows (3∗).
  Recher-Hardware: IBM (6∗), Sun (2∗), HP, Unisys.
  Plattensysteme: EMC (4∗), IBM (3∗), Hitachi (3∗).

Größte Datenbanken (3)

- Größte Anzahl Zeilen in Data Warehouse:

  2 Billionen (10^{12}): Sprint (Telekommunikation)

  DBMS: NonStop SQL von HP, Rechner und Platten auch von HP.
- Größte Anzahl Zeilen in OLTP Datenbank:

  90 Milliarden: UPS

  IBM DB2 Universal DB für z/OS, IBM eServer z990, IBM Storage
- Größte "Peak Workload":

  UPS: 1.1 Mrd. SQL Anweisungen pro Sekunde

  Ausstattung: siehe "Größte Anzahl Zeilen in OLTP Datenbank".
  Zweiter: US Bureau of Customs and Border Protection (340 Mio).

DBMS: Auswahlkriterien (1)

- Preis, Lizenzbedingungen.

  Es gibt auch "Total Cost of Ownership"-Berechnungen (inklusive Aufwand für Administration). Man sollte auch den Preis für Support und Updates berücksichtigen. Bei späteren Updates, Upgrades, oder Vergrößerung der Hardware (mehr CPUs) ist man dem DBMS-Hersteller mehr oder weniger ausgeliefert. Wie ist die Reputation des Herstellers?

- Wissen der Mitarbeiter, Weiterbildungskosten.
- Benötigter Zeitaufwand für Administration.
- Verfügbarkeit für mehrere Hardware-Plattformen.
- Performance [http://www.tpc.org], Skalierbarkeit.

DBMS: Auswahlkriterien (2)

- Zuverlässigkeit, Unterstützung für 7×24-Betrieb.

  Wie gut ist Support für Backup&Recovery? Wie schnell ist Wiederherstellung, falls benötigt? Wird ein "Notfallsystem" unterstützt?

- Sicherheit

  Wie wahrscheinlich sind Bugs, die Hackern Zutritt gewähren? Wie schnell werden Sicherheitsprobleme gelöst? Wie gut wird der DBA über solche informiert? Wie leicht kann man Patches anwenden?

- Wie gut ist der Support? Gibt es wirklich schnelle, qualifizierte Hilfe bei Problemen?
- Wie lange werden alte Versionen noch gewartet?

DBMS: Auswahlkriterien (3)

- Verfügbarkeit von Tools.

  Zur Entwicklung von Anwendungsprogrammen.

- SQL ist mehr oder weniger Standard.

  Jedes moderne relationale DBMS (RDBMS) unterstützt mindestens den SQL-86 Standard oder das Entry-Level des SQL-92-Standards. Aber der Wechsel eines DBMS kann trotzdem teuer sein. Jeder Anbieter hat gewisse Erweiterungen zum SQL-Standard. Auch viele Programmier-Tools existieren nur für einen speziellen Anbieter.

- Objektrelationale Erweiterungen und XML könnten für nichtklassische Anwendungen wichtig sein.

  Klassische Anwendungen: Banken, Lager, Versandhäuser. Nichtklassisch: z.B. CAD-Systeme. In den objektrelationalen Erweitungen und der XML-Unterstützung unterscheiden sich die Anbieter.

Inhalt

Datenbanknutzer (1)

Datenbank-Administrator (DBA):

- Sollte das komplette DB-Schema kennen.

  Ändert das DB-Schema wenn nötig, dokumentiert Änderungen.
- Verantwortlich für Sicherheit und Zugriffsrechte.
- Überwacht die Systemleistung.

  Führt Performance-Tuning durch, falls nötig.
- Überwacht verfügbaren Speicherplatz, installiert ggf. neue Festplatten.
- Verantwortlich für Sicherungskopien (Backups). Stellt Daten nach einem Plattenfehler wieder her.

Datenbanknutzer(2)

Datenbank-Administrator, fortgesetzt:

- Installiert neue Versionen der DBMS-Software.
- Versucht Korrektheit der Daten zu gewährleisten.
- Verantwortlich für Lizenzen.
- Kontaktperson für Support / DBMS-Anbieter.
- Experte für die DBMS-Software.
- Kann alles zerstören.

  Braucht normalerweise mächtige Rechte für das Betriebssystem.

Datenbanknutzer (3)

Anwendungsprogrammierer:

- Schreibt Programme für Standardaufgaben.

  "Anwendungsprogramme", die von den "naiven Nutzern" (s.u.) verwendet werden. Das beinhaltet heute häufig auch ein Web-Interface.
- Kennt SQL gut und zusätzlich einige Programmiersprachen und Entwicklungs-Tools.
- Normalerweise dem DBA unterstellt.

  Oder externer Berater, der nur für ein Projekt angestellt ist.
- Macht evtl. DB-Design (d.h. entwirft DB-Schema).

  Aber es gibt Spezialisten für diese Aufgabe.

Datenbanknutzer (4)

Fortgeschrittener Nutzer (ein Endbenutzer):

- Kennt SQL und/oder einige Anfrage-Tools.
- Kann Nicht-Standard-Auswertungen der Daten ohne Hilfe von Programmierern durchführen.

  Z.B. Manager: Benötigt neue Statistiken zur Entscheidungsfindung.

Naiver Nutzer (auch ein Endbenutzer):

- Nutzt die DB nur über Anwendungsprogramme.
- Macht die eigentliche Arbeit der Dateneingabe.

Datenbank-Werkzeuge

- Interaktiver SQL-Interpreter
- Graphische/Menü-basierte Anfrage-Tools
- Schnittstellen für Datenbank-Zugriff aus Standard-Programmiersprachen (C, Pascal, Java, . . .)
- Tools für formularbasierte DB-Anwendungen (4GL)
- Report Generator (für gedruckte Übersichten)
- WWW-Interface
- Tools für Import/Export von Daten, Überwachung der Leistung, Backup&Recovery, . . .

Zusammenfassung (1)

Funktionen von Datenbanksystemen:

- Persistenz
- Integration, keine Redundanz/Duplikatspeicherung
- Datenunabhängigkeit
- Weniger Programmieraufwand: Viele Algorithmen enthalten, besonders für Externspeicher (Platten).
- Ad-hoc-Anfragen

  D.h. wenn jemand eine neue Anfrage im Kopf hat, kann er sie sofort stellen. Früher benötigte man dafür ein neues Programm.

Zusammenfassung (2)

Funktionen von Datenbanksystemen, Fortsetzung:

- Hohe Datensicherheit (Backup & Recovery)
- Zusammenfassung von Updates zu Transaktionen

  Transaktionen werden ganz oder gar nicht ausgeführt (sind atomar).
- Mehrbenutzerbetrieb: Synchronisation von Zugriffen
- Integritätsüberwachung
- Sichten für verschiedene Nutzer (Nutzergruppen)
- Datenzugriffskontrolle
- System-Katalog (Data Dictionary)

Zusammenfassung (3)

- Hauptziel eines DBMS: dem Nutzer eine vereinfachte Sicht auf den persistenten Speicher geben.
- Der Nutzer muss nicht nachdenken über:
  - ◇ Physische Speicherdetails
  - ◇ Unterschiedlichen Informationsbedarf von verschiedenen Nutzern
  - ◇ Effiziente Anfrageformulierung
  - ◇ Möglichkeit von Systemabsturz/Plattenfehlern
  - ◇ Möglichkeit von störenden gleichzeitigen Zugriffen anderer Nutzer

Übung

Aufgabe:

- Angenommen, Sie sollen ein System zur Evaluation der Lehre an dieser Universität entwickeln: Studierende stimmen über die Vorlesungs-Qualität ab.

  Es gibt ein Formular im Internet, in das Studierende ihre Daten eingeben können. Diese werden auf dem Web-Server abgespeichert. Später werden die gesammelten Daten ausgewertet, z.B. Berechnung von Durchschnittswerten.

- Vorschlag: Daten direkt in einer Datei speichern.
- Welche Argumente gibt es, stattdessen ein DBMS zu verwenden?