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

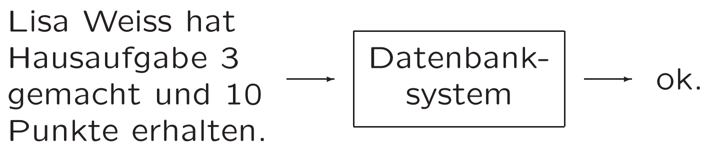
Aufgabe einer Datenbank (1)

- 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? → $\boxed{\text{Datenbank-system}}$ → 1, 2

Aufgabe einer Datenbank (2)

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



- 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)

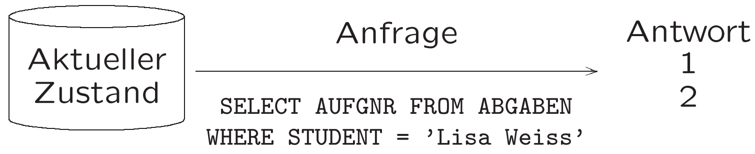
- Obige Frage "Welche Hausaufgaben hat Lisa Weiss gemacht?" war in natürlicher Sprache (Deutsch).
- Das Verstehen natürlicher Sprache durch Computer ist ein schwieriges Problem (Missverständnisse).
- Daher werden Fragen ("Anfragen") normalerweise in formaler Sprache gestellt, heute meist in SQL.

  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:



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

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.
- Legt mögliche Datenbankzustände fest.
- 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.

- Entspricht der Variablendeklaration (Informationen über Datentypen).

  Z.B. wenn eine Variable `i` als **`short int`** (16 Bit) deklariert ist, sind die möglichen Zustände von `i` normalerweise `-32768 .. +32767`.

Zustand vs. Schema (2)

Datenbank-Zustand (Ausprägung eines Schemas):

- Beinhaltet die aktuellen Daten, dem Schema entsprechend strukturiert.
- Ändert sich oft (wenn Datenbank-Informationen geändert werden).
- Entspricht dem Inhalt/Wert der Variablen.

  Z.B. hat `i` im aktuellen Zustand s den Wert `5`. Ein Update, z.B. `i := i + 1`, verändert den Zustand zu s', in dem `i` den Wert `6` hat.

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	DB-Schema
Lisa Weiss	1	10	DB-Zustand
Lisa Weiss	2	8	
Daniel Sommer	1	9	
Daniel Sommer	2	9	

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 definiert
  - ◇ eine Menge $\mathcal{SCH}$ möglicher DB-Schemata,
  - ◇ für jedes DB-Schema $\mathcal{S} \in \mathcal{SCH}$ ein Menge $\mathcal{ST}(\mathcal{S})$ möglicher DB-Zustände.
- Oft (aber nicht immer) wird ein Datenmodell in eine Menge von Basis-Datentypen parametrisiert.
- Z.B. für relationales Modell nicht wichtig, ob die Tabelleneinträge nur Zeichenketten oder auch Zahlen, Datums- oder Zeitwerte usw. sein können.

Datenmodell (2)

- Ein Datenmodell definiert also normalerweise Typ-Konstruktoren, um komplexe Datenstrukturen aus elementaren Datentypen zu bilden.

  Z.B. `RECORD/STRUCT`, `SET`, `LIST`, `ARRAY`.
  Oft gibt es zusätzlich auch die Möglichkeit, mittels Integritätsbedingungen (Constraints) die möglichen Zustände einzuschränken: Z.B.: "Die Punktzahl muß ≥ 0 sein." "Es kann keine zwei Einträge für den gleichen Studenten und die gleiche Aufgabe geben."

- Es ist auch sehr typisch, daß DB-Schemata Symbole einführen (wie z.B. Tabellennamen), die der DB-Zustand auf Werte oder Funktionen abbildet.

  Diese Namen verwendet der Benutzer dann in Anfragen/Updates.

Datenmodell (3)

- Es gibt keine allgemein anerkannte formale Definition für den Begriff "Datenmodell".

  Obige Definition ist mein Vorschlag. Man könnte mehr ins Detail gehen, aber das würde darin enden, Oracle 8.1.3 als ein Datenmodell zu definieren. Die meisten Bücher behandeln den Begriff sehr ungenau.

- Man könnte speziell darüber streiten, ob die Anfragesprache zum Datenmodell gehört.

  Z.B. gab es am Anfang kleine PC-DBMS, die Daten in Tabellen strukturierten, aber keine Anfragen zuließen, die Daten mehrerer Tabellen kombinierten. Diese Systeme galten als nicht ganz relational. Um die Ausdrucksstärke verschiedener Anfragesprachen für ein Datenmodell zu vergleichen, muss man beides unterscheiden. Außerdem wird das ER-Modell meist ohne Anfragesprache dargestellt.

Datenmodell (4)

- 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 (5)

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)

Datenmodell (6)

- Es gibt auch Datenmodelle (im weiteren Sinn), bei denen nicht explizit ein Schema deklariert wird:

  Formal hat $\mathcal{SCH}$ dann genau ein Element.

  - ◇ Well-formed XML: DOM-Schnittstelle

    Dieses Beispiel zeigt, daß man ein Datenbank-Schema (auch allgemein) als zusätzliche Einschränkung auffassen kann, die eine Typprüfung erlaubt.

  - ◇ Dateisystem eines Betriebssystems
  - ◇ DNS (Domain Name System)
  - ◇ Windows Registry

Inhalt

1. Grundlegende Datenbankbegriffe
2. Datenbankmanagementsysteme (DBMS)
3. Sicht der Programmierer, Datenunabhängigkeit
4. DBMS-Anbieter
5. Datenbanknutzer und Datenbank-Tools

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, ...

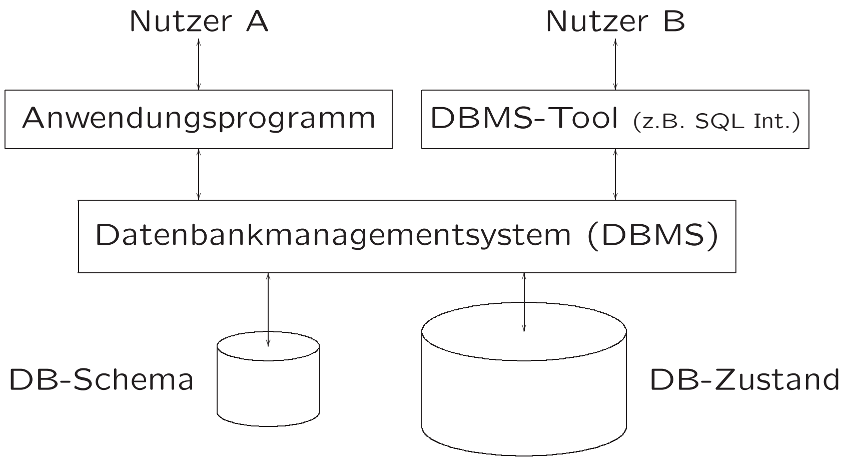
  - ◇ Programm zum Eintragen von Punkten für Klausur und Hausaufgaben (für Übungsleiter/Tutor).
  - ◇ Programm, das einen Bericht (Übersicht) für den Professor ausdruckt, um Noten zu vergeben.

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)



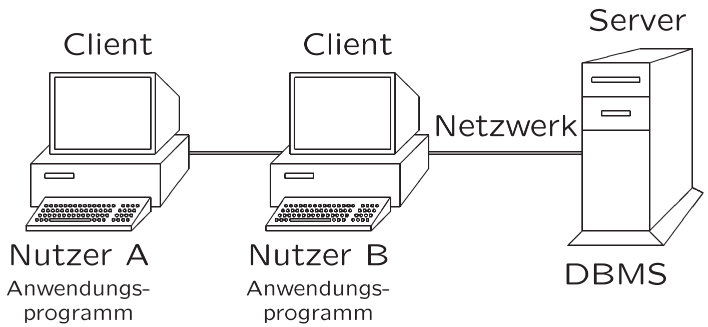
DB-Anwendungssysteme (1)

- 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).

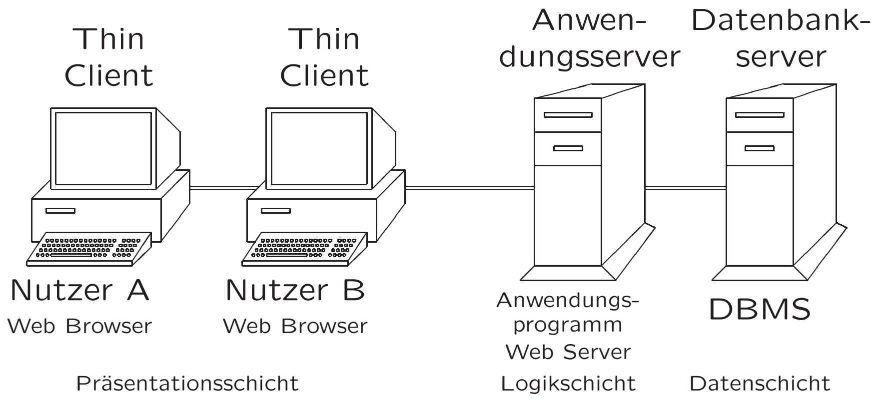
DB-Anwendungssysteme (2)

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 Daten (1)

Heute:

5 ⟶ $n!$ (Fakultät) ⟶ 120

Morgen:

5 ⟶ $n!$ (Fakultät) ⟶ 120

⇒ Kein persistenter Speicher nötig.
Die Ausgabe ist nur eine Funktion der Eingabe.

Persistente Daten (2)

Heute:

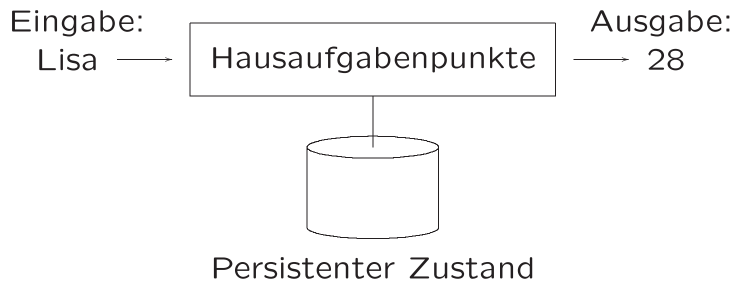
Lisa ⟶ Hausaufgabenpunkte ⟶ 18

Morgen:

Lisa ⟶ Hausaufgabenpunkte ⟶ 28

⇒ Die Ausgabe ist eine Funktion der Eingabe und eines persistenten Zustands.

Persistente Daten (3)



Persistente Informationen:

- Informationen, die länger leben als ein einziger Prozess (Programmausführung). Überleben auch Stromausfall oder Neustart des Betriebssystems.

Persistente Daten (4)

Übung:

- Welche der folgenden Programme brauchen persistenten Speicher und warum?
  - ◇ Taschenrechner (nicht programmierbar)
  - ◇ Web Browser
  - ◇ Bildschirmschoner
- Wie persistent ist der Speicher Ihres Videorekorders für Kanäle?
- Sind Informationen in der Windows Registry persistent?

Persistente Speicherung (1)

Klassischer Weg, Persistenz zu implementieren:

- Informationen, die in zukünftigen Programmläufen benötigt werden, werden in einer Datei gespeichert.
- Das Betriebssystem speichert die Datei auf einer Festplatte.
- Festplatten sind persistente Speicher: Der Inhalt geht nicht verloren, wenn der Rechner ausgeschaltet oder das Betriebssystem neu gestartet wird.
- Dateisysteme sind Vorgänger moderner DBMS.

Persistente Speicherung (2)

Implementierung von Persistenz mit Dateien:

- Dateien sind normalerweise nur Folgen von Bytes.
- Die Struktur der Datensätze (Records) muß wie in Assembler-Sprachen festgelegt werden.

0: 40: 42: 44:

L	i	s	a		W	e	i	s	s		...	0	3	1	0

- Die Information über die Dateistruktur existiert nur in den Köpfen der Programmierer.
- Das System kann nicht vor Fehlern schützen, da es die Dateistruktur nicht kennt.

Persistente Speicherung (3)

Implementierung von Persistenz mit einem DBMS:

- Die Struktur der zu speichernden Daten wird so definiert, dass sie das System versteht:

```
CREATE TABLE ABGABEN(STUDENT VARCHAR(40),
                     AUFGNR  NUMERIC(2),
                     PUNKTE  NUMERIC(2))
```

- Somit ist die Dateistruktur formal dokumentiert.
- Das System kann Typfehler in Anwendungsprogrammen erkennen.
- Vereinfachte Programmierung, höhere Abstraktion

Unterprogrammbibliothek (1)

- Die meisten DBMS nutzen Dateien (von Windows, UNIX: Betriebssystem), um Daten zu speichern.

  Manche nutzen aus Effizienzgründen direkten Plattenzugriff.
- Man kann ein DBMS als Unterprogrammbibliothek für Dateizugriffe verstehen.
- Verglichen mit dem Dateisystem bietet ein DBMS Operationen auf höherer Ebene an.

  Man arbeitet z.B. mit Punktzahl statt Bytes an bestimmer Position.
- Es enthält bereits viele Algorithmen, die man sonst selbst programmieren müsste.

Unterprogrammbibliothek (2)

- Zum Beispiel enthält ein DBMS Routinen zum
  - ◇ Sortieren (Mergesort)
  - ◇ Suchen (B-Bäume)
  - ◇ Freispeicherverwaltung in Datei, Pufferverwaltung
  - ◇ Aggregationen, statistische Auswertungen
- Optimiert für große Datenmengen (die nicht in den Hauptspeicher passen).
- Es unterstützt auch mehrere Nutzer (automatische Sperren, Locks) und Sicherheitsmaßnahmen, um die Daten bei Abstürzen zu schützen (siehe unten).

Datenunabhängigkeit (1)

- DBMS = Software-Schicht über dem Dateisystem. Zugriff auf die Dateien nur über DBMS.
- Die Indirektion erlaubt es, interne Veränderungen zu verstecken.
- Idee abstrakter Datentypen: Implementierung kann geändert werden, aber die Schnittstelle bleibt stabil.
- Hier ist die Implementierung die Dateistruktur, die aus Effizienzgründen geändert werden muss.
  Das Anwendungsprogramm muß nicht angepasst werden: Schnittstelle zum Datenzugriff ist stabil.

Datenunabhängigkeit (2)

Typisches Beispiel:

- Anfangs nutzte ein Professor die Hausaufgaben-DB nur für seine Vorlesungen im aktuellen Semester.

  Zur Vereinfachung lässt die obige Tabelle "ABGABEN" nur eine Vorlesung zu. Aber es könnte eine zusätzliche Spalte geben, um verschiedene Vorlesungen zu unterscheiden.

- Da die DB klein war und es wenige Zugriffe gab, genügte es, die Daten als "heap file" zu speichern.

  D.h. die Zeilen der Tabelle (Records) werden ohne bestimmte Reihenfolge gespeichert. Um Anfragen auszuwerten, muss das DBMS die ganze Tabelle durchsuchen, d.h. jede Zeile der Tabelle lesen und überprüfen, ob sie die Anfragebedingung erfüllt ("full table scan"). Für kleine Tabellen ist das kein Problem.

Datenunabhängigkeit (3)

- Später nutzte die ganze Uni die DB und Daten vergangener Vorlesungen mussten aufbewahrt werden.
- Die DB wurde viel größer und hatte mehr Zugriffe.

  Das "Lastprofil" hat sich geändert.

- Für schnelleren Zugriff wird ein Index (typischerweise ein B-Baum) benötigt.

  Ein Index eines Buches ist eine sortierte Liste mit Stichwörtern und den zugehörigen Seitenzahlen, wo die Wörter vorkommen. Ein B-Baum enthält im Wesentlichen eine sortierte Liste von Spaltenwerten zusammen mit Verweisen auf die Tabellenzeilen, die diese Werte enthalten (vgl. Kapitel über physischen DB-Entwurf).

Datenunabhängigkeit (4)

Fußnote: Indexe vs. Indices

- Als Plural von "Index" ist in der DB-Literatur (wenn es um B-Bäume etc. geht) "Indexe" üblich, obwohl "Indices" auch vorkommt (selten).

  Z.B. verwenden Kemper/Eickler, Vossen, Härder/Rahm und die deutsche Übersetzung von Elmasri/Navathe den Plural "Indexe". In meinem deutschen Wörterbuch steht als Plural nur "Indices" bzw. "Indizes", aber das deckt sich nicht mit meinem persönlichen Sprachempfinden. In der Wikipedia stehen "Indexe", "Indices", "Indizes" als (offenbar gleichberechtigte) Alternativen (speziell für Datenbanken). Als Genitiv sind "des Indexes" und "des Index" beide möglich.

- Dagegen besteht Einigkeit, daß die kleinen tiefgestellten Buchstaben (wie i in x_i) "Indices" sind.

Datenunabhängigkeit (5)

Ohne DBMS (oder mit Prä-Relationalem DBMS):

- Die Verwendung eines Indexes für den Datenzugriff muss explizit in der Anfrage verlangt werden.
- Somit müssen Anwendungsprogramme geändert werden, wenn sich die Dateistruktur ändert.
- Vergisst man, ein selten verwendetes Anwendungsprogramm zu ändern, wird die DB inkonsistent.

  Das kann jedoch nur passieren, wenn man direkt mit Betriebssystem-Dateien arbeitet. Schon ein DBMS für das Netzwerk-Datenmodell (z.B.) machte Index-Updates automatisch. Die Anfrage-Programme mussten sich jedoch explizit auf den Index beziehen.

Datenunabhängigkeit(6)

Mit Relationalem DBMS:

- Das System versteckt die Existenz von Indexen an der Schnittstelle.
- Anfragen und Updates müssen (und können) sich nicht auf Indexe beziehen.
- Folgendes geschieht automatisch durch das DBMS:
  - ◇ Aktualisierung des Indexes bei einem Update,
  - ◇ Nutzung des Indexes zur Anfrageauswertung, wenn dies von Vorteil (d.h. schneller) ist.

Datenunabhängigkeit (7)

Konzeptuelles Schema ("Schnittstelle"):

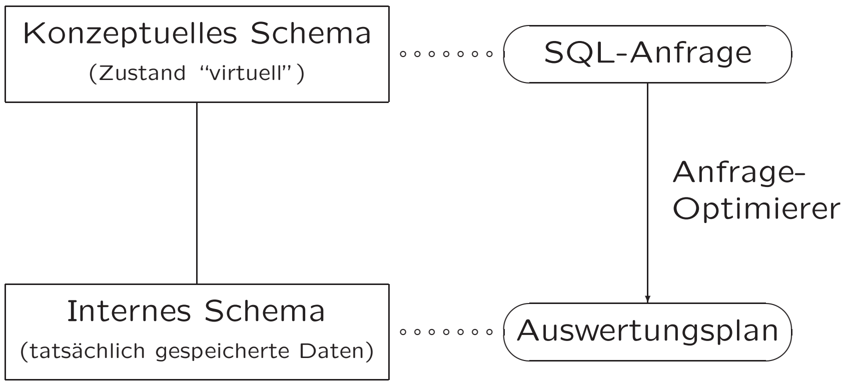
- Nur der logische Informationsgehalt der DB
- Vereinfachte Sicht: Speicherdetails sind versteckt.

Internes/Physisches Schema ("Implementierung"):

- Indexe, Aufteilung der Tabellen auf Festplatten
- Parameter für Speicher-Management, wenn Tabellen wachsen oder schrumpfen
- Physische Plazierung neuer Zeilen in einer Tabelle.

  Z.B. Zeilen mit gleichem Spaltenwert in den gleichen Plattenblock.

Datenunabhängigkeit (8)

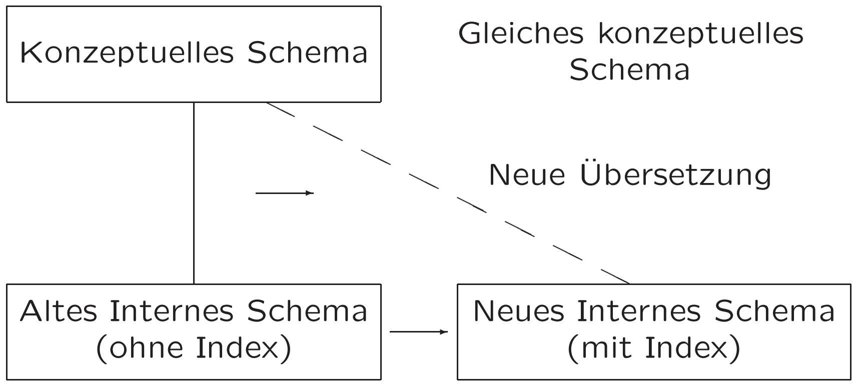


Datenunabhängigkeit (9)

1. Der Nutzer gibt eine Anfrage ein (z.B. in SQL), die sich auf das konzeptuelle Schema bezieht.
2. Das DBMS übersetzt dies in eine Anfrage/ein Programm ("Auswertungsplan"), die/das sich auf das interne Schema bezieht.

   Hierfür ist der "Anfrageoptimierer" im DBMS zuständig.
3. Das DBMS führt die übersetzte Anfrage auf dem gespeicherten Zustand des internen Schemas aus.
4. Das DBMS übersetzt das Resultat zurück in das konzeptuelle Schema.

Datenunabhängigkeit (10)



Deklarative Sprachen (1)

- Physische Datenunabhängigkeit verlangt, dass sich die Anfragesprache nicht auf Indexe beziehen kann.
- Deklarative Anfragesprachen gehen noch weiter:
  - ◇ Anfragen sollen nur beschreiben, was (welche Information) gesucht wird,
  - ◇ aber sollen keine spezielle Methode vorschreiben, wie diese Information berechnet werden soll.
- Algorithmus = Logik + Kontrolle (Kowalski)

  Imperative/Prozedurale Sprachen: Kontrolle explizit, Logik implizit.
  Deklarative/Deskriptive Sprachen: Logik explizit, Kontrolle implizit.

Deklarative Sprachen (2)

- SQL ist eine deklarative Anfragesprache. Man gibt nur Bedingungen für die gesuchten Daten an:

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

- Häufig leichtere Formulierungen: Der Nutzer muss nicht über eine effiziente Auswertung nachdenken.
- Oft viel kürzer als imperative Programmierung. Daher ist z.B. die Programmentwicklung billiger.

Deklarative Sprachen (3)

- Deklarative Anfragesprachen
  - ◇ erlauben leistungsstarke Optimierer

    weil sie keine Auswertungsmethode vorschreiben.
  - ◇ benötigen leistungsstarke Optimierer

    weil ein naiver Auswertungsalgorithmus ineffizient wäre.
- Größere Unabhängigkeit von aktueller Hardware/Software-Technologie:
  - ◇ Einfache Parallelisierung
  - ◇ Heutige Anfragen können zukünftige Algorithmen verwenden (bei neuer DBMS-Version).

"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.

  $\Rightarrow$ Schützt Investitionen in Programme und Daten.

Logische Datenunabh. (1)

- Logische Datenunabhängigkeit ermöglicht Änderungen des logischen Inhalts einer Datenbank.
- Informationen können nur erweitert werden, z.B. in Tabelle `ABGABEN` Spalte `ABGABE_DATUM` hinzufügen.

  Oder Informationen anders darstellen, z.B. eine 24h-Notation anstelle von AM/PM verwenden, inch statt cm, kombinierter Vor- und Nachname statt zwei einzelne Spalten.
- Das könnte für neue Anwendungen nötig sein.
- Es sollte nicht nötig sein, alte Anwendungen zu ändern, auch wenn die Zeilen nun länger sind.

Logische Datenunabh.(2)

- Logische Datenunabhängigkeit nur wichtig, wenn es Anwendungsprogramme mit unterschiedlichem, aber überlappendem Informationsbedarf gibt.
- Logische Datenunabhängigkeit hilft auch bei der Integration unterschiedlicher Datenbestände.
  - ◇ Früher hatte jede Abteilung einer Firma eine eigene Datenbank oder eigene Dateien.
  - ◇ Heute ist eine einzige zentrale DB das Ziel.

    Sie kann verteilt sein, aber das ist ein anderes Thema.

Logische Datenunabh.(3)

- Wenn eine Firma mehr als eine DB hat, werden die Daten in den einzelnen Datenbanken meist überlappen, d.h. manches ist mehrfach gespeichert.
- Daten heißen redundant, wenn sie aus anderen Daten und Wissen über die Anwendung ableitbar sind.
- Probleme:
  - ◇ Doppelter Aufwand für Datenerfassung/Updates
  - ◇ Irgendwann vergisst man, eine der Kopien zu ändern (Daten werden inkonsistent)
  - ◇ Verschwendet Speicherplatz, auch bei Backups

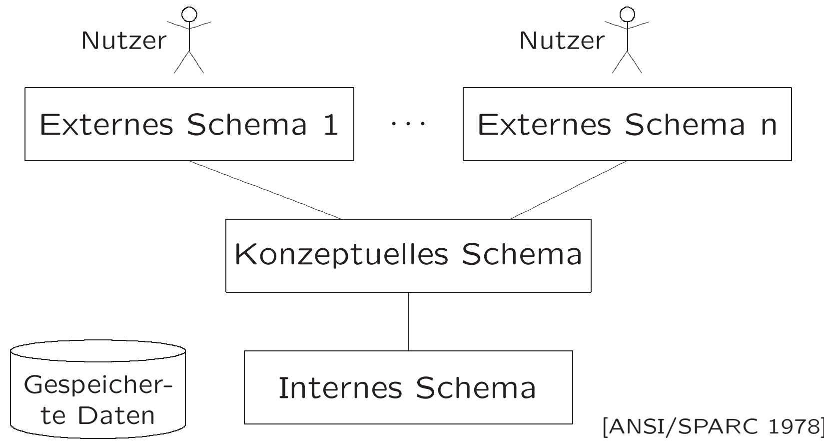
Logische Datenunabh.(4)

Externe Schemata/Sichten:

- Logische Datenunabhängigkeit benötigt eine dritte Schema-Ebene, die externen Schemata/Sichten.
- Jeder Nutzer (-gruppe) hat eigene Sicht auf Daten
- Eine externe Sicht enthält eine Teilmenge der Daten der DB, eventuell leicht umstrukturiert.

  Externe Sichten sind auch aus Sicherheitsgründen wichtig: Sie enthalten nur Informationen, auf die ein Nutzer zugreifen darf.
- Dagegen beschreibt das konzeptuelle Schema den gesamten Inhalt der Datenbank.

Drei-Schema Architektur



[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 10g

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

- IBM: DB2 Universal Database V8.2 (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 12.5 (plus ...)

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

DBMS-Anbieter (2)

- Informix: z.B. Informix Dynamic Server 10.0

  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)

DBMS-Markt 1998 [Dataquest-Studie]

Anbieter	Marktanteil	Änderung zu 1997
IBM	32.3%	+3.4%
Oracle	29.3%	−0.1%
Microsoft	10.2%	+0.3%
Informix	4.4%	−0.4%
Sybase	3.5%	−1.0%
Andere	20.5%	−2.2%

Marktgröße (1998): 7100 Mio. US-Dollar (+15%).

DBMS-Anbieter (4)

RDBMS-Marktanteile (1998) [Dataquest]

Anbieter	RDBMS	UNIX	NT
Oracle	38.5%	60.9%	46.1%
IBM	30.8%	7.3%	10%
Microsoft	7%	—	29.7%
Informix	6%	13%	—
Sybase	5%	7%	3%
Andere	14%	11.8%	12%
Marktgröße:	$5400 Mio.	$2200 Mio	$1200 Mio.
Wachstum:	+18%	+10%	+46%

DBMS-Anbieter (5)

DB-Marktanteile 2001 [in %, Verkauf neuer Lizenzen]:

Anbieter	DBMS	RDBMS	UNIX	Windows
Oracle	32.0 −4.9	$\boxed{39.8}$ −4.9	$\boxed{63.3}$ −5.7	34.0 −1.0
IBM	$\boxed{34.6}$ +4.3	34.1 +6.2	24.7 +15.4	20.7 +15.8
IBM, alt	31.7 +5.7	30.7 +5.9	17.5 +19.4	20.0 +15.0
Informix	3.0 −9.4	3.3 +8.8	7.2 +6.8	0.8 +39.6
Microsoft	16.3 +17.8	14.4 +25.3	—	$\boxed{39.9}$ +25.3
Sybase	2.6 −16.1	3.3 −16.1	4.6 −14.3	1.6 −11.9
Andere	14.4 −2.8	8.5 −7.5	7.4 −2.0	3.7 −10.7
Gesamt	8844 Mio \$	7108 Mio \$	3014 Mio \$	2555 Mio \$
Entwickl.	1.4%	1.6%	-1.4%	11.0%

Quelle: Gartner Dataquest (Mai 2002) [UNIX, Windows: nur RDBMS]

DBMS-Anbieter (6)

DB-Marktanteile 2002 [in %, Verkauf neuer Lizenzen]:

Anbieter	DBMS	RDBMS
Oracle	26.9	33.9
IBM	$\boxed{36.9}$	$\boxed{36.2}$
davon Informix	2.3	
Microsoft	18.8	18.0
Sybase	2.2	
NCR		2.7
Andere	15.2	9.2
Gesamt	8363 Mio \$	6629 Mio \$

Quelle: Gartner Dataquest (Mai 2003)

DBMS-Anbieter (7)

- Oracle führt eine IDC-Studie über Datenbank-Marktanteile an (2003 veröffentlicht):

Oracle	39.4%
IBM	33.6%
Microsoft SQL Server	11.1%

- Diese Studie zeigt jedoch auch, wie sich die Erlöse von 2001 zu 2002 entwickelt haben:

Oracle	−5%
IBM	+9%
Microsoft	+15%

DBMS-Anbieter (8)

- Oracle führt auch Umfrage der Fortune 100 companies an (von FactPoint Group); 400 IT-Manager wurden nach der Wahl ihrer Primär-DB gefragt:
  - ◇ Oracle: 51%
  - ◇ IBM DB2: 22% (19% Großrechner, 3% UNIX/NT)
  - ◇ Microsoft SQL Server: 8%
  - ◇ Kombination von Anbietern: 15%
  - ◇ Andere: 4%
- Oracle wird von 93% dieser Firmen verwendet. 76% ihrer SAP-Installationen laufen auf Oracle-Basis.

DBMS-Anbieter (9)

- IDC-Studie (relationale/objektrelationale DBMS):

  Quelle: [http://www.techweb.com/wire/26805179], 2004.

Oracle	39.8%
IBM	31.3%
Microsoft	12.1%

Markt insgesamt: 13.6 Mrd. Dollar (2003).

- Gartner (2004, Relationale DBMS):

IBM (DB2, Informix)	35.2%
Oracle	32.6%
Microsoft	18.7%

Markt insgesamt: 7.1 Mrd. Dollar (2003).

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™ 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 könnten für nicht-klassische DB-Anwendungen wichtig sein.

  Hier unterscheiden sich die Anbieter.

Nachteile eines DBMS

- Oft teuer
- Abhängigkeit vom DBMS-Anbieter

  Es gibt Standards für SQL, aber jedes System hat Erweiterungen und spezielle Tools. DBMS-Software ist auch oft eng mit dem Betriebssystem verknüpft, weil sie teilweise ähnliche Aufgaben haben. Betriebssystem Update ⇒ evtl. auch DBMS-Update nötig.

- Benötigt viel Zeit zum Lernen.

  Oracle 8 hatte $\geq$95 Bände (=1.70 m) Dokumentation.

- Eine handoptimierte C-Routine ist praktisch immer schneller als der allgemeine DBMS-Code.

Wann kein DBMS verwenden

- Daten nur von einem Programm verarbeitet. Keine anderen Anwendungen mit diesen Daten geplant.
- Standard-DBMS nicht möglich, z.B. wenn:
  - ◇ Antwortzeit sehr kurz sein muss (Realzeit),
  - ◇ Nicht-Standard-Sperren benötigt.
- Die Neuentwicklung ist nicht zu teuer:
  - ◇ Die Struktur der Daten ist einfach.
  - ◇ Alle Daten passen in den Hauptspeicher.
  - ◇ Eine einfache Backup-Strategie reicht aus.

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 Backups der Daten. Stellt Daten nach einem Plattenfehler, etc. 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, ...

Oracle kennen (1)

Kern:

- Relationales Modell, Grundlagen des DB-Entwurfs
- SQL: SQL-92 (Standard) + Oracle-Erweiterungen
- Datentypen und Datentypfunktionen
- Data Dictionary (Systemtabellen)
- PL/SQL für gespeicherte Prozeduren, Trigger

  PL/SQL ist Oracle's eigene Programmiersprache (ähnlich wie Ada), eng mit SQL verbunden. Als Alternative kann Java genutzt werden.
- SQL*Plus: Output-Formatierung, Skripte
- Orientierung in der Oracle-Dokumentation.

Oracle kennen (2)

Datenbank-Administration:

- SQL: Oracle-Erweiterungen, z.B. für physische Speicherparameter, Nutzer-Management.
- Oracle-Prozesstruktur, Installationsparameter, Tuning, interne Datenstrukturen.
- Administrations-Tools: den DB-Server starten/ anhalten, Plattenspeicher zufügen, etc.
- Backup & Recovery in Oracle
- Import/Export-Utilities, SQL*Loader

Oracle kennen(3)

Anwendungsprogrammierung:

- Oracle Pro*C (SQL eingebettet in C), ODBC
- Java-Interfaces: JDBC, JSQL
- Oracle Developer (iDS): Visuelle Werkzeuge für Entwicklung von Anwendungsprogrammen

  Z.B. Form Builder: Erstellt Programme, die spezialisierte Editoren für bestimmte Tabellen sind, z.B. Darstellung einzelner Tupel in einem Bildschirmfenster. Oracle Developer enthält auch Report Builder, Graphics Builder, etc.
- Oracle Anwendungsserver (iAS) für Web-Interfaces
- Etwas Wissen über Sicherheit, z.B. für Web-Interfaces.

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

Übungen (1)

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 in einer UNIX-Datei speichern.
- Welche Argumente gibt es, stattdessen ein DBMS zu verwenden?

Übungen (2)

- Stellen Sie sich vor, die Hausaufgabenpunkte sind in einer Textdatei gespeichert mit dem Format

  Vorname:Nachname:Aufgabennummer:Punkte

  (d.h. ein Tupel der Tabelle `ABGABEN` pro Zeile).
- Welchen Aufwand schätzen Sie für die Entwicklung eines C-Programms, das die Gesamtpunktzahl je Student ausgibt (alphabetisch geordnet).

  Anzahl Zeilen: ________ (ohne Kommentare)
  Arbeitszeit: ________
- In SQL braucht man 4 Zeilen und 2 Minuten Zeit.