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 (Ed.): 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/Bedeutung eines DBMS erklären
- Datenunabhängigkeit, Deklarativität und die Drei-Stufen-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 Ann Smith 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.

  Ann Smith hat Hausaufgabe 3 gemacht und 10 Punkte erhalten. → Datenbank-system → ok.

- Ein Datenbanksystem ist eine Computer-Version einer Karteikartenbox (nur mächtiger). Eine Tabellenkalkulation kann man als kleines DBS ansehen.

Aufgabe einer Datenbank (3)

- Normale DBS machen keine schwierigen Auswertungen auf den gespeicherten Daten, um Fragen zu beantworten.

  Aber es gibt z.B. Wissensbanken und Datamining-Werkzeuge.

- Sie können jedoch benötigte Daten schnell in/von einer großen Datenmenge finden/abfragen (Gigabytes oder Terrabytes – größer als Hauptspeicher).
- Sie können auch mehrere Teildaten für eine Antwort aggregieren/kombinieren, z.B. Durchschnitt für Hausaufgabe 3 berechnen.

Aufgabe einer Datenbank (4)

- Obige Frage "Welche Hausaufgaben hat Ann Smith gemacht?" war in natürlicher Sprache (Deutsch).
- Es ist nicht leicht, Computern natürliche Sprache verständlich zu machen (Missverständnisse!).
- Daher werden Fragen ("Anfragen") normalerweise in formaler Sprache gestellt, heute meist in SQL.

  Man kann SQL als Programmiersprache verstehen, entwickelt um spezielle Anfrageaufgaben zu erfüllen. Im Gegensatz zu Pascal, C oder Java kann man keine beliebigen Programme in SQL schreiben.

- Einige DBS erlauben natürlichsprachliche Anfragen.

Zustand, Anfrage, Update

- Menge der gespeicherten Daten = DB-Zustand:

  Aktueller Zustand —Anfrage→ Antwort 1 2

```
SELECT HOMEWORK FROM SOLVED
WHERE STUDENT = 'Ann Smith'
```

- Eingabe, Änderung oder Löschen von Informationen ändert den Zustand:

  Alter Zustand —Update→ Neuer Zustand

```
INSERT INTO SOLVED
VALUES ('Ann Smith', 3, 10)
```

Strukturierte Information (1)

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

  Heutiges Angebot in der Mensa ist Pizza. $\longrightarrow$ Hausaufgaben-DBS $\longrightarrow$ Fehler.

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

  Wieviele Hausaufgaben hat ein Student gemacht?

Strukturierte Information (2)

- Eigentlich speichert ein DBS nur Daten (Zeichenketten, Zahlen), und keine Informationen.
- Daten werden durch Interpretation zu Information.
- Daher müssen Begriffe wie Studenten und Aufgaben definiert/erklärt werden, bevor die DB genutzt werden kann.

  Natürlich ist es möglich, eine DB zu entwickeln, in der beliebige Texte gespeichert werden können. Dann kann aber das DBS nur nach Substrings suchen und keine höherwertigen Anfragen bearbeiten. Je mehr ein DBMS über die Struktur der Daten "weiß", desto besser kann es den Nutzer 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 kann zu Problemen führen.

- Entspricht der Variablendeklaration (Informationen über Datentypen).

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

Zustand vs. Schema (2)

Databank-Zustand:

- 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', wobei `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)

SOLVED		
STUDENT	HOMEWORK	POINTS
Ann Smith	1	10
Ann Smith	2	8
Michael Jones	1	9
Michael Jones	2	9

DB-Schema

DB-Zustand

Übung

- Ein Professor möchte mehr Daten über seine Studenten speichern. Er schreibt ein Programm, das an jeden Studenten folgende E-Mail verschickt:

```
Sehr geehrte Frau Smith,
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, ...
```

- Wie kann er die nötigen Daten in Tabellen speichern?

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.

  Formal wird eine Menge von Namen für Datentypen zusammen mit Operationen auf diesen Datentypen wird durch eine Signatur Σ beschrieben (siehe Kapitel 2). Diese Symbole werden interpretiert (indem sie auf Mengen von Werten und Funktionen auf diesen Mengen abgebildet werden) durch eine logische Interpretation $\mathcal{I}$. Ein Datenmodell ist nun eigentlich parametrisiert in der Datentyp-Signatur und der Interpretation der Datentypen: $(\mathcal{SCH}_\Sigma, \mathcal{ST}_\mathcal{I})$.

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

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, die verwendet wird, um DB-Schemata zu definieren.
- Data Manipulation Language (DML): Sprache zum Schreiben von 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 komlizierter 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 (historisch)
- Hierarchisches Modell (historisch)

Inhalt

DBMS (1)

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

- Definition eines DB-Schemas für konkrete Anwendungen,

  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,
- Abfrage des derzeitigen DB-Zustands,
- Änderung des DB-Zustands.

DBMS (2)

- Natürlich verwenden normale Nutzer kein SQL für den täglichen Umgang mit einer Datenbank.
- Sie verwenden Anwendungsprogramme, die speziell entwickelt wurden, um eine schönere Benutzerschnittstelle zu bieten.

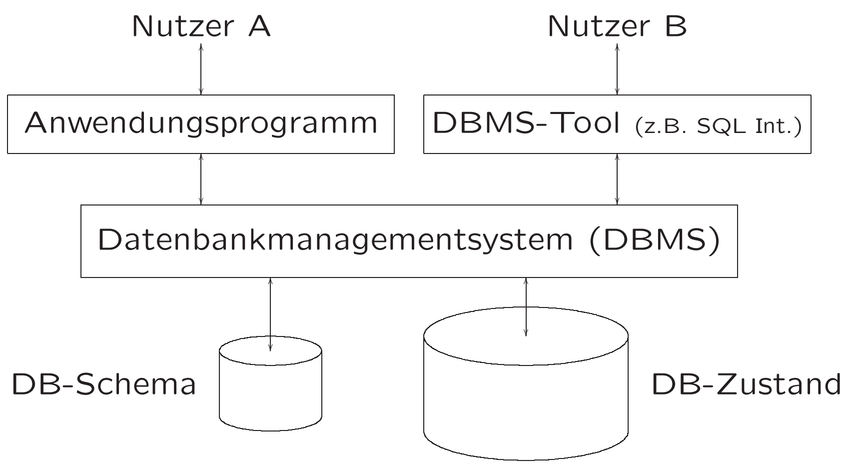
  Z.B. ein Formular, in dem Felder ausgefüllt werden können.

- Intern enthält das Anwendungsprogramm jedoch ebenfalls SQL-Befehle (Anfragen, Updates), um mit dem DBMS zu kommunizieren.

DBMS (3)

- Oft viele verschiedene Anwendungsprogramme genutzt, um auf die selbe zentrale DB zuzugreifen.
- Z.B. könnte die Hausaufgaben-DB bestehen aus:
  - ◇ Einem Web-Interface für Studenten.
  - ◇ Einem Programm um Klausur- und Hausaufgabenpunkte zu verwalten.
  - ◇ Einem Programm, das einen Report für den Professor ausdruckt, um Noten zu vergeben.
- Das interaktive SQL-Interface, das zum DBMS gehört, ist nur eine andere Art auf die DB zuzugreifen.

DBMS (4)



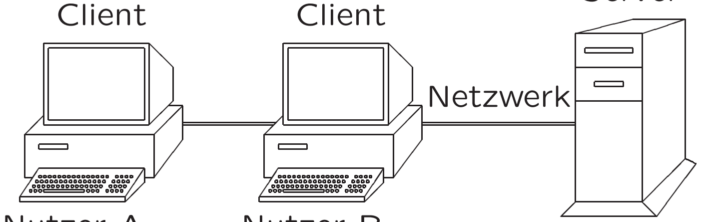
DB-Anwendungssysteme (1)

- Oft greifen verschiedene Nutzer gleichzeitig auf die gleiche Datenbank zu.
- Das DBMS ist normalerweise ein im Hintergrund laufender Server-Prozess (oder mehrere Prozesse), 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).

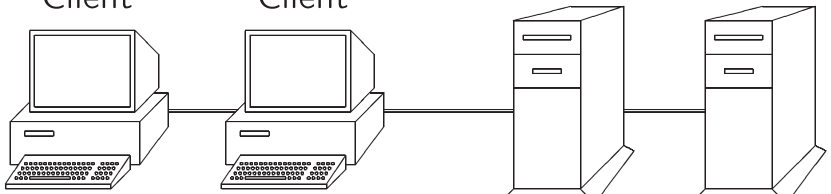
DB-Anwendungssysteme (2)

Client-Server-Architektur:



DB-Anwendungssysteme (3)

Three-Tier Architektur:



DB-Anwendungssysteme (4)

Übung:

- Betrachten wir eine DB, die von einer Bank genutzt wird um Konten zu verwalten.
- Welche Aufgaben könnten von verschiedenen Anwendungsprogrammen, die auf die DB zugreifen, unterstützt werden?
  - ◇ ______________________________
  - ◇ ______________________________
  - ◇ ______________________________
  - ◇ ______________________________

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. 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 einigen Anwendungsprogrammen.

Inhalt

Persistenter Speicher (1)

Heute:

5 → Fakultät → 120

Morgen:

5 → Fakultät → 120

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

Persistenter Speicher (2)

Heute:

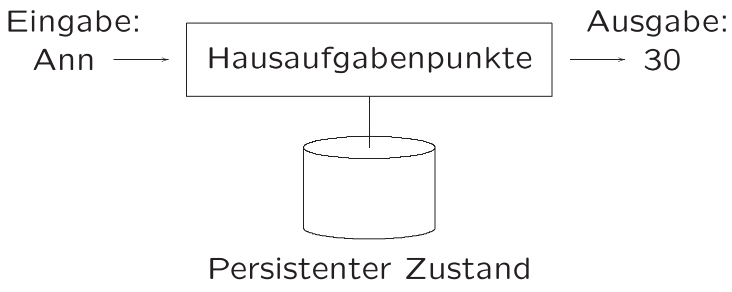
Ann → Hausaufgabenpunkte → 20

Morgen:

Ann → Hausaufgabenpunkte → 30

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

Persistenter Speicher (3)



Persistente Informationen:

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

Persistenter Speicher (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 Daten (1)

Klassischer Weg, Persistenz zu implementieren:

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

Persistente Daten (2)

Implementierung von Persistenz mit Dateien:

- OS-Dateien sind gewöhnlich nur Folgen von Bytes.
- Eine Record-Struktur muss wie in Assembler-Sprachen definiert werden.

0:										40:		42:	44:
A	n	n		S	m	i	t	h		...	0	3	1	0

- Informationen über die Dateistruktur ist nur den Programmierern bekannt.
- Das System kann nicht vor Fehlern schützen, da es die Dateistruktur nicht kennt.

Persistente Daten (3)

Implementierung von Persistenz mit einem DBMS:

- Die Struktur der zu speichernden Infos muss für das System verständlich definiert werden:

```
CREATE TABLE SOLVED(STUDENT  VARCHAR(40),
                    HOMEWORK NUMERIC(2),
                    POINTS   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 OS-Dateien, um Daten zu speichern.

  Manche nutzen aus Effizienzgründen direkten Plattenzugriff.
- Man kann DBMS als Unterprogrammbibliothek sehen, die für Dateizugriffe genutzt werden kann.
- Verglichen mit einem OS bietet ein DBMS Operationen auf höherem Level an.
- D.h. es enthält bereits viele Algorithmen, die man sonst programmieren müsste.

Unterprogrammbibliothek (2)

- Zum Beispiel enthält ein DBMS Routinen zum
  - ◇ Sortieren (Mergesort)
  - ◇ Suchen (B-Bäume)
  - ◇ Dateiverwaltung, 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" den OS-Dateien. Zugriff auf die Dateien nur über DBMS.
- "Indirektheit" macht es möglich, interne Veränderungen zu verstecken.
- Idee der abstrakten Datentypen: Die Implementierung ändern, aber das Interface beibehalten.
- Hier ist die Implementierung die Dateistruktur, die aus Effizienzgründen geändert werden muss. Das Interface des Anwendungsprogramms bleibt jedoch erhalten.

Datenunabhängigkeit (2)

Typisches Beispiel:

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

  Zur Vereinfachung lässt die obige Tabelle "SOLVED" 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. Für kleine Tabellen ist das kein Problem.

Datenunabhängigkeit (3)

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

  Das "Lastprofil" hat sich geändert.

- Für schnelleren Zugriff wird ein Index (z.B. 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 den Zeilen, die diese Werte enthalten (vgl. Kapitel über physischen DB-Entwurf).

Datenunabhängigkeit (4)

Ohne DBMS (oder mit Pre-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 OS-Dateien arbeitet. Schon ein DBMS für das Netzwerk-Datenmodell (z.B.) machte Index-Updates automatisch. Die Anfrage-Befehle mussten sich jedoch explizit auf den Index beziehen.

Datenunabhängigkeit(5)

Mit Relationalem DBMS:

- Das System versteckt die Existenz von Indexen auf dem Interface.
- Anfragen und Updates müssen (und können) sich nicht auf den Index beziehen.
- Das System macht automatisch:
  - ◇ Änderung des Indexes im Falle eines Updates,
  - ◇ Nutzung des Indexes zur Anfrageauswertung, wenn dies von Vorteil ist.

Datenunabhängigkeit (6)

Konzeptuelles Schema ("Interface"):

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

Internes/Physisches Schema ("Implementierung"):

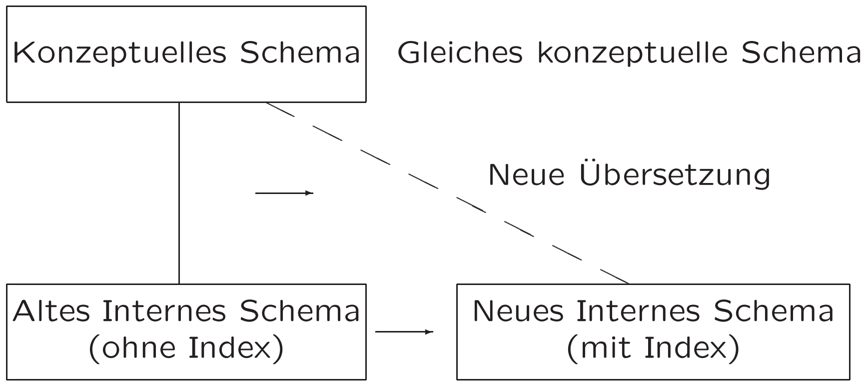
- Indexe, Aufteilung der Tabellen auf Festplatten
- Speicher-Management, wenn Tabellen wachsen oder schrumpfen
- Physische Platzierung neuer Zeilen in einer Tabelle.

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

Datenunabhängigkeit (7)

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 (das macht der "Anfrageoptimierer").
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 (8)



Deklarative Sprachen (1)

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

  Imperative/Prozedurale Sprachen: Explizite Steuerung, Implizite Logik.
  Deklarative/Deskriptive Sprachen: Explizite Logik, Implizite Steuerung.

Deklarative Sprachen (2)

- SQL = deklarative Anfragesprache. Der Nutzer legt nur Bedingungen für die gesuchten Daten fest:

```
SELECT X.POINTS
FROM   SOLVED X
WHERE  X.STUDENT = 'Ann Smith'
AND    X.HOMEWORK = 3
```

- Häufig leichtere Formulierungen: Der Nutzer muss nicht über eine effiziente Auswertung nachdenken.
- 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.
  - ⋄ brauchen 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 Anwendungsprogrammen, für die sie ursprünglich erfasst wurden, eine Bedeutung.
- Physische Datenunabhängigkeit:
  - ⋄ Programme sollten nicht von Datenspeichermethoden 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 DB.
- Informationen können nur erweitert werden, z.B. in Tabelle `SOLVED` Spalte `SUBMISSION_DATE` 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 dabei, ursprüglich verschiedene Datenbanken zu vereinen.
  - ◇ 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 mehrmals gespeichert.
- Daten heißen redundant, wenn sie aus anderen Daten und Wissen über die Anwendung ableitbar sind.
- Probleme:
  - ◇ Verdoppelt Aufwand für Dateneinträge/Updates
  - ◇ Irgendwann vergisst man, eine der Kopien zu ändern (Daten werden inkonsistent)
  - ◇ Veschwendet Speicherplatz, auch bei Backups

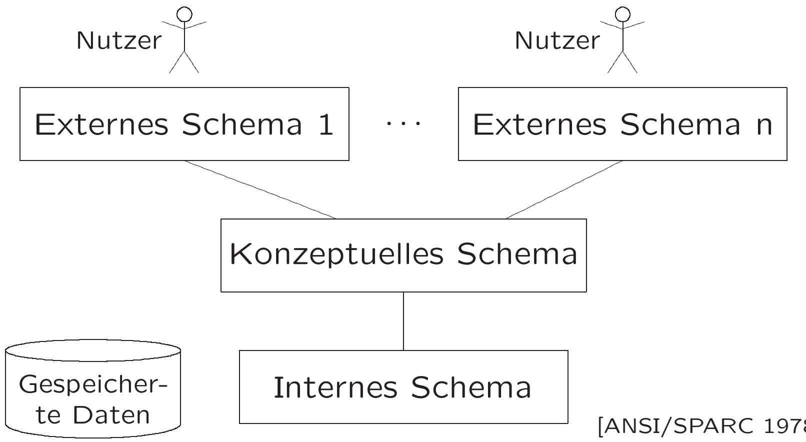
Logische Datenunabh.(4)

Externe Schemata/Sichten:

- Logische Datenunabhängigkeit benötigt ein drittes DB-Schema, die externen Schemata/Sichten.
- Jeder Nutzer 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 ("alles oder nichts").

  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 zeitgleiche Zugriffe anderer Nutzer nachdenken müssen (z.B. 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 kann sich erinnern, wer was gemacht hat.

Integrität:

- Das DBMS kann prüfen, ob Daten plausibel sind.
- Das DBMS kann auch Updates ablehnen, die vorhandene Geschäftsregeln verletzen würden.

Weitere DBMS-Funktionen (3)

Data Dictionary:

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

SYS_TABLES

TABLE_NAME	OWNER
SOLVED	BRASS
SYS_TABLES	SYS
SYS_COLUMNS	SYS

SYS_COLUMNS

TABLE_NAME	SEQ	COL_NAME
SOLVED	1	STUDENT
SOLVED	2	HOMEWORK
SOLVED	3	POINTS
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 2000 (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 9.4

  Informix wurde 2001 von IBM gekauft (für $ 1000 Mio). [http://www-306.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	39.8	−4.9	63.3	−5.7	34.0	−1.0
IBM	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	—		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	36.9	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.

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/]

Auswahlkriterien (1)

- Preis, Lizenzbedingungen.

  Man sollte auch den Preis für Support und spätere Updates beachten. Es gibt auch "Total Cost of Ownership"-Berechnungen.
- Verfügbarkeit für mehrere Hardware-Plattformen.
- Performance [http://www.tpc.org], Skalierbarkeit.
- Verfügbarkeit von Tools.

  Zur Entwicklung von Anwendungsprogrammen.
- Wissen der Mitarbeiter, Weiterbildungskosten.
- Benötigter Zeitaufwand für Administration.

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?
- Support für 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?
- 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.

Nachteile eines DBMS

- 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 OS verknüpft, weil sie teilweise ähnliche Aufgaben haben. OS-Update $\Rightarrow$ evtl. auch DBMS-Update nötig.
- Benötigt viel Zeit zum Lernen.

  Oracle 8 hatte ≥ 95 Bände ($= 1.70$ m) Dokumentation.
- Eine handoptimierte C-Routine ist oft schneller als der DBMS-Code.

Wann kein DBMS verwenden

- Daten von einem Programm verarbeitet. Keine anderen Anwendungen mit diesen Daten geplant.
- DBMS aus vielen Gründen unnützlich, z.B. wenn:
  - ◇ Antwortzeit sehr kurz sein muss (Realzeit),
  - ◇ non-Standard-Sperren benötigt werden.
- 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

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

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 Systemperformance.

  Macht Performance-Tuning.
- Überwacht verfügbaren Speicherplatz und installiert 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 Datenkorrektheit zu gewährleisten.
- Verantwortlich für Lizenzen.
- Kontaktperson für Support / DBMS-Anbieter.
- Experte für die DBMS-Software.
- Kann alles zerstören.

  Braucht manchmal viele Rechte für das Betriebssystem.

Datenbanknutzer (3)

Anwendungsprogrammierer:

- Schreibt Programme für Standardaufgaben.

  "Anwendungsprogramme", von den "naiven Nutzern" verwendet (siehe unten). Kann heutzutage auch ein Web-Interface beinhalten.

- 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 (Analytiker,etc.).

Datenbanknutzer (4)

Fortgeschrittener Nutzer (eine Art des "Endnutzers"):

- Kennt SQL und / oder einige Anfrage-Tools.
- Macht nicht-standard-Auswertungen der Daten ohne Hilfe von Anwendungsprogrammierern.

  Z.B. ein mittlerer Manager, der neue Statistiken zur Entscheidungsfindung benötigt.

Naiver Nutzer (die andere Art des "Endnutzers"):

- Nutzt die DB nur über Anwendungsprogramme.
- Gibt die meisten Daten ein.

Datenbank-Tools

- Interaktiver SQL-Interpreter
- Grafik/Menü-basierte Anfrage-Tools
- Interface für DB-Zugriff für Standard-Programmiersprachen (C, Pascal, Java, ...)
- Tools für formularbasierte DB-Anwendungen (4GL)
- Report generator
- WWW-Interface
- Tools für Import/Export von Daten, Überwachung der Performance, 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 Programmier-Tools um Anwendungsprogramme zu schreiben

  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 Datanbanksystemen:

- Persistenz
- Integration, keine Redundanz/Duplikatspeicherung
- Datenunabhängigkeit
- Weniger Programmieraufwand: Viele Algorithmen enthalten, v.a. für externen Speicher (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
  - ◇ Unterschiedlicher Informationsbedarf von verschiedenen Nutzern
  - ◇ Effiziente Anfrageformulierung
  - ◇ Möglichkeit von Systemabsturz/Plattenfehlern
  - ◇ Möglichkeit von gleichzeitigen Zugriffen verschiedener Nutzer

Übungen (1)

- Aufgabe: Entwicklung eines Systems, in dem Studenten über die Qualität von Vorlesungen abstimmen können.

  Es gibt ein Formular im Internet, in das Studenten ihre Daten eingeben können. Diese werden auf dem Web-Server abgespeichert. Später werden die gesammelten Daten ausgewertet, z.B. Berechnung von Durchschnittswerten.
- Vorschlag: für diese Aufgabe C-Programm schreiben und die Daten in UNIX-Dateien 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

  Name des Studenten:Aufgabennummer:Punkte

  (d.h. ein Tupel der Tabelle `SOLVED` pro Zeile).
- Stellen Sie sich vor, Sie müssen ein C-Programm schreiben, das die Gesamtpunktzahl je Student ausgibt (Studenten alphabetisch geordnet).
- Wie viele Zeilen und wieviel Programmierzeit brauchen Sie?
- In SQL braucht man 4 Zeilen und 2 Minuten Zeit.