Part 1: Introduction

References:

- Elmasri/Navathe:Fundamentals of Database Systems, 3rd Edition, 1999.
  Chapter 1, "Databases and Database Users"
  Chapter 2, "Database System Concepts and Architecture"
- Kemper/Eickler: Datenbanksysteme (in German), 3rd Edition, 1999.
  Chapter 1: "Einleitung und Übersicht" ("Introduction and Overview")
- Heuer/Saake: Datenbanken, Konzepte und Sprachen (in German), Thomson, 1995.
- Lipeck: Skript zur Vorlesung Datenbanksysteme (in German), Univ. Hannover, 1996.
- Silberschatz/Korth/Sudarshan: Database System Concepts, 3rd Edition.
  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, (Oct. 1991), 110–120.
- Silberschatz/Stonebraker/Ullman: Database research: achivements and opportunities into the 21st century. ACM SIGMOD Record, Vol. 25, Issue 1, (March 1996), 52–63.

Objectives

After completing this chapter, you should be able to:

- explain basic notions: Database State, Schema, Query, Update, Data Model, DDL, DML.
- explain the role of the DBMS.
- explain data independence, declarativity, and the three schema architecture.
- name some DBMS vendors.
- name different classes of users of a database application system.
- name some DBMS tools.

Overview

1. Basic Database Notions

2. Database Management Systems (DBMS)

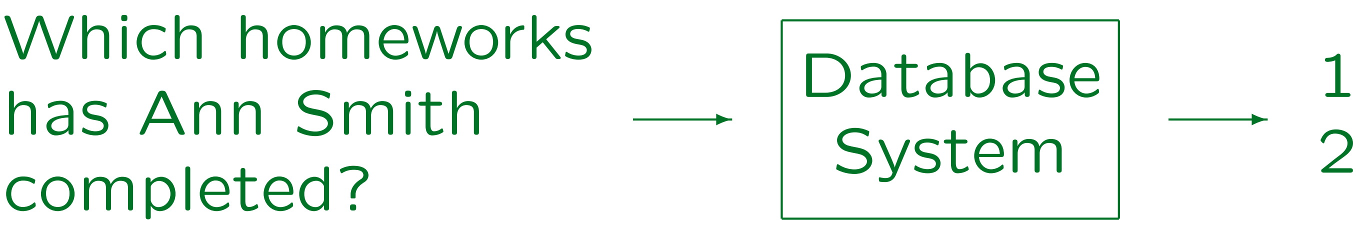
3. Programmer's View, Data Independence

4. DBMS Vendors

5. Database Users and Database Tools

Task of a Database (1)

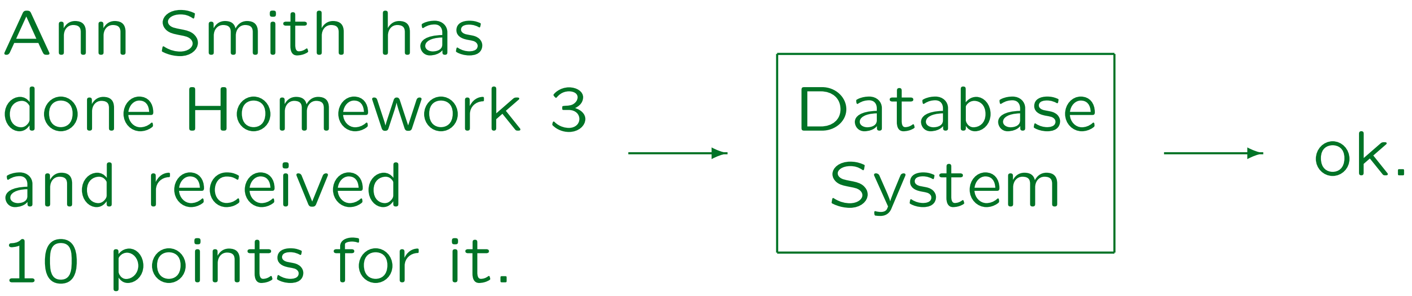
- What is a database? Difficult question. There is no precise and generally accepted definition.
- Naive approach: The main task of a database system (DBS) is to answer certain questions about a subset of the real world, e.g.



Which homeworks has Ann Smith completed? → Database System → 1 2

Task of a Database (2)

- The DBS acts only as storage for information. The information must first be entered and then kept current.



Ann Smith has done Homework 3 and received 10 points for it. → Database System → ok.

- A database system is a computerized version of a card-index box/filing cabinet (but more powerful).
- A spreadsheet can be considered a small DBS.

Task of a Database (3)

- Normal database systems do not perform very complicated computations on the stored data in order to answer questions.

  But there are e.g. knowledge bases and data mining tools.

- However, they can find/retrieve the requested data quickly in/from a large set of data (Gigabytes or Terrabytes — larger than main memory).

- They can also aggregate/combine several pieces of stored data for one answer, e.g. compute the average points for Homework 3.

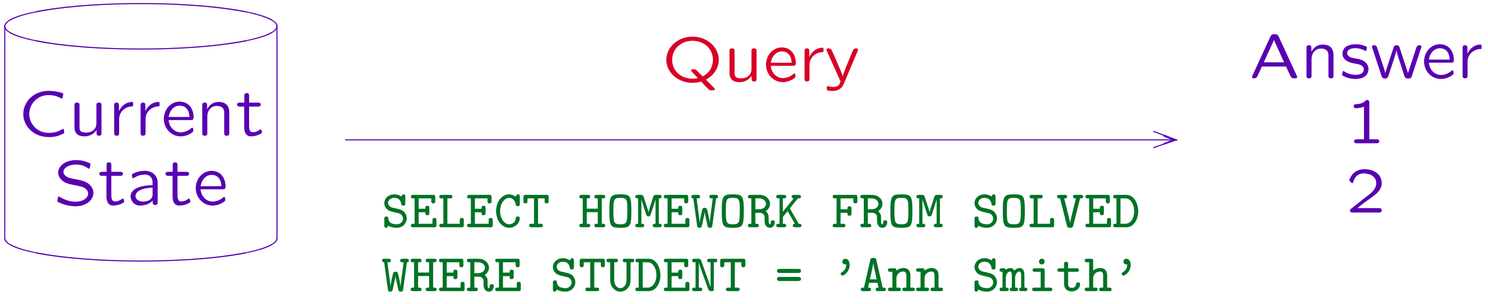
Task of a Database (4)

- Above, the question “Which homeworks has Ann Smith completed?” was shown in natural language.
- Making computers understand natural language is not easy (large potential for misunderstandings).
- Therefore, questions (“queries”) are normally written in a formal language, today typically in SQL.

  One can view SQL as a programming language designed especially for data retrieval tasks. However, in contrast to languages like Pascal, C, or Java, one cannot write arbitrary programs in SQL.
- But there are natural language interfaces for DBS.

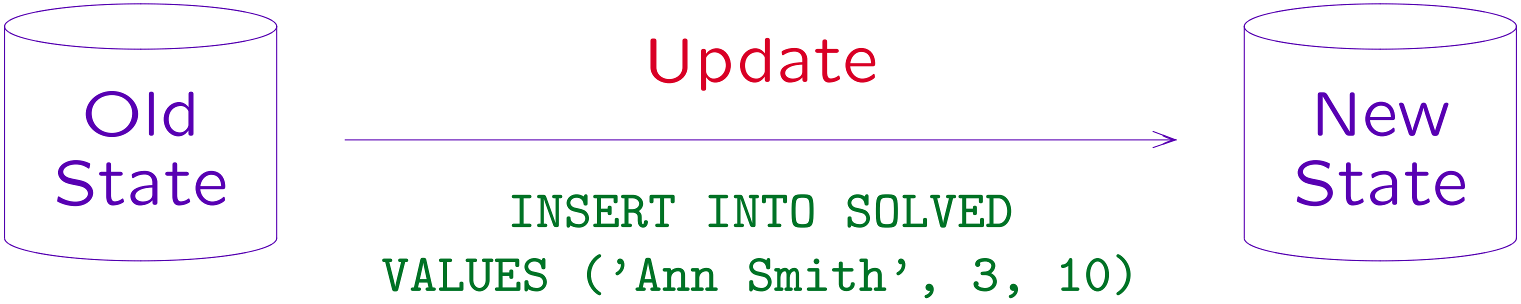
State, Query, Update

- The set of stored data is called the database state:



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

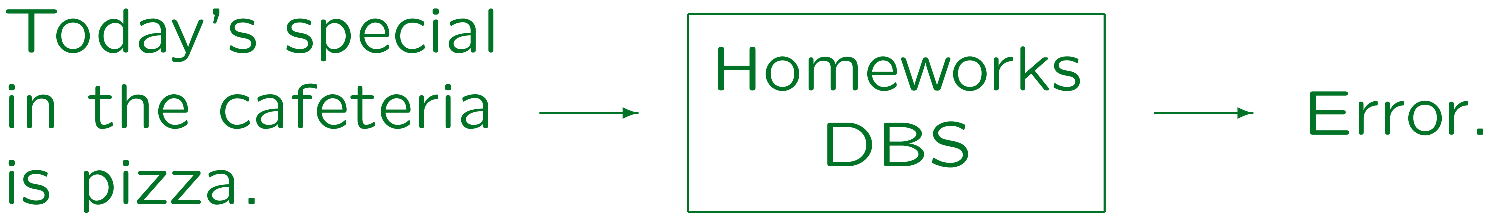
- Entering, modifying, or deleting information changes the state:



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

Structured Information (1)

- Each database can store only information of a pre-declared structure (a limited domain of discourse):



Today's special in the cafeteria is pizza. → Homeworks DBS → Error.

- Because the data are structured, not simply text, more complex evaluations are possible, e.g.:

How many homeworks has each student done?

Structured Information (2)

- Actually, a database system stores only data (character strings, numbers), and not information.
- Data become information by interpretation.
- Therefore, concepts like students and exercises must be defined/declared before the database can be used.

  Of course, it is possible to create a databases in which arbitrary texts can be stored. But then the DBS can only search for substrings, not answer more advanced queries. The more the DBMS "knows" about the structure of the data, the better it can support the user.

State vs. Schema (1)

Database Schema:

- Formal definition of the structure of the database contents.
- Determines the possible database states.
- Defined only once (when the DB is created).

  In practice, it might sometimes be necessary to modify the schema. However, this happens seldom and causes some difficulties.

- Corresponds to variable declaration (type information).

  E.g. if a variable `i` is declared as `short int`, the possible states of `i` are normally `-32768 .. +32767`.

State vs. Schema (2)

Database State (Instance of the Schema):

- Contains the actual data, structured according to the schema.
- Changes often (whenever database-information is updated).
- Corresponds to current contents/value of a variable.

E.g. in the current state s, `i` might have value 5. An update, e.g. `i := i + 1`, changes the state to s', where `i` has value 6.

State vs. Schema (3)

- In the relational data model, the data is structured in form of tables (relations).
- Each table has a name, sequence of named columns (attributes) and a set of rows (tuples).

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

DB Schema

DB State (Instance)

Exercise

Name some typical applications of databases, e.g. companies or organizations which use them, or any private uses you might have for a database:

1. ______________________________
2. ______________________________
3. ______________________________
4. ______________________________
5. ______________________________

Data Model (1)

- Defines formal language (syntax&semantics) for
  - ◇ declaring database schemas

    Data Definition Language (DDL)
  - ◇ querying the current database state

    Query Language (QL), part of the Data Manipulation Language.
  - ◇ changing the database state.

    Forms together with the query language the Data Manipulation Language (DML).
- Examples: Relational Model, Entity-Relationship M., Object-Oriented/Object-Relational Models, XML.

  Historical, but still in use: Hierarchical Model, Network Model.

Data Model (2)

- Often, the term “data model” is used in a more abstract way for the general concepts underlying the concrete language.

  For instance, one speaks of the “relational data model”, and not of some specific version of the SQL language. Unfortunately, the term “data model” is then very fuzzy.

- Some people use the term “Data Model” for “Database Schema”.

  E.g. the Enterprise Data Model is the database schema for all information of a company.

Overview

1. Basic Database Notions

2. Database Management Systems (DBMS)

3. Programmer's View, Data Independence

4. DBMS Vendors

5. Database Users and Database Tools

DBMS (1)

A Database Management System (DBMS) is an application-independent software that implements a data model, i.e. allows

- definition of a DB schema for some concrete application,

  Since the DBMS itself is application-independent, it stores the schema typically on the disk, often together with the database state in special "system tables".

- storage of an instance of this schema on e.g. a disk,
- querying the current instance (database state),
- changing the database state.

DBMS (2)

- Of course, normal users do not need to use SQL for their daily tasks of data entry or lookup.
- They use application programs that have been developed especially for this task and offer a nicer user interface.

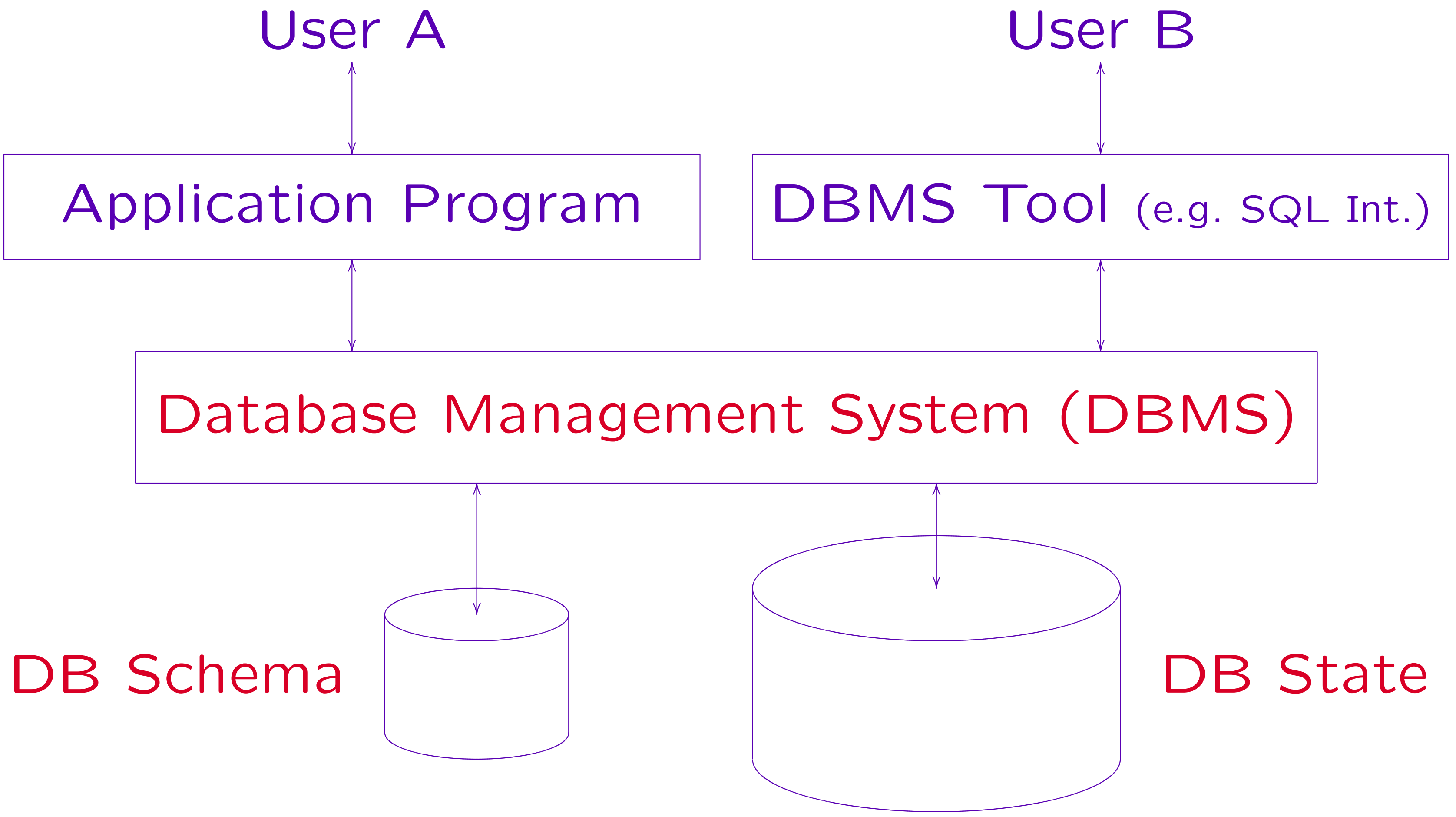
  E.g. a form, in which fields can be filled out, and then the user clicks on the "Submit" button.

- However, internally, the application program contains SQL statements (queries, updates) in order to communicate with the DBMS.

DBMS (3)

- Often, several different application programs are used to access the same centralized database.
- E.g. the homeworks DB might have:
  - ◇ A web interface for students.
  - ◇ A program used by the GSA (graduate student assistant) to load homework and exam points.
  - ◇ A program that prints a report for the professor used to assign grades.
- The interactive SQL interface that comes with the DBMS is simply another way to access the DB.

DBMS (4)


User A
User B
Application Program
DBMS Tool (e.g. SQL Int.)
Database Management System (DBMS)
DB Schema
DB State

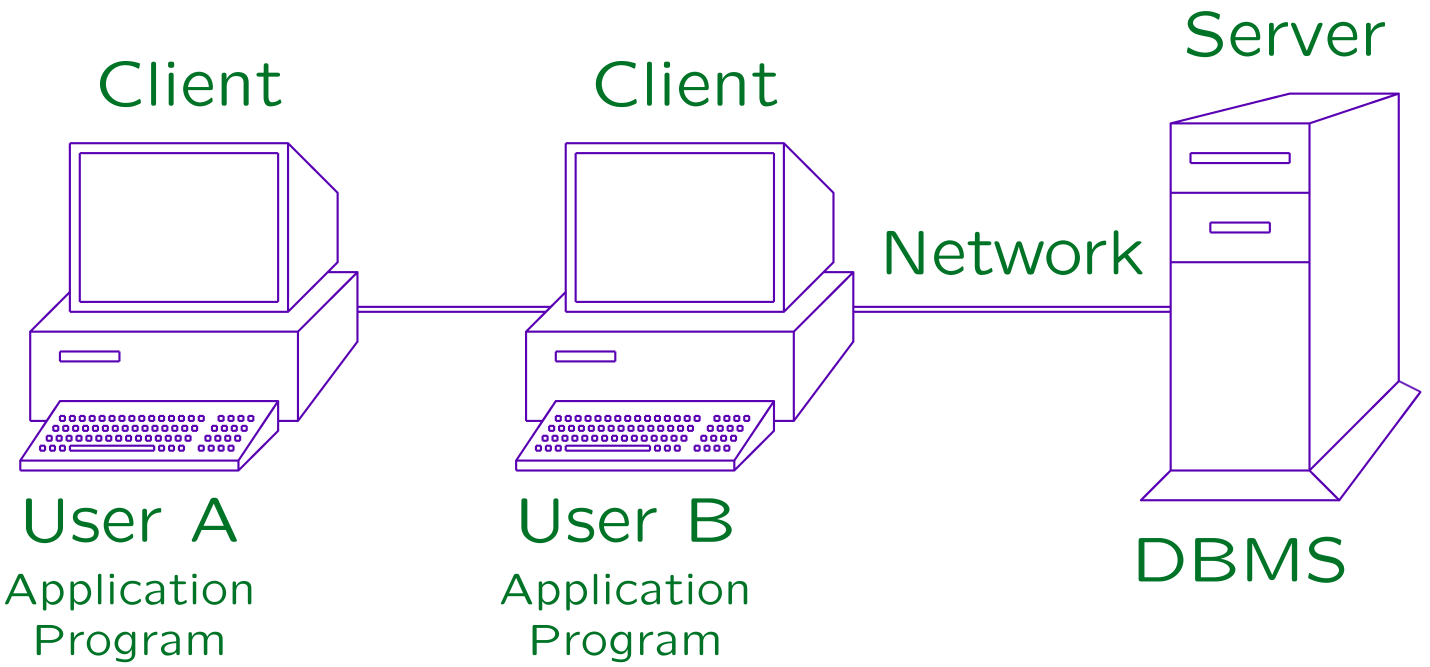
DB Application Systems (1)

- Often, different users access the same database concurrently (i.e. at the same time).
- The DBMS is usually a background server process (or set of such processes) that is accessed over the network by application programs (clients).

  Very similar to a web server. For some small PC DBMS, there is only a single program that acts as DBMS and as interpreter for the application programs.
- One can also view the DBMS as an extension of the operating system (more powerful file system).

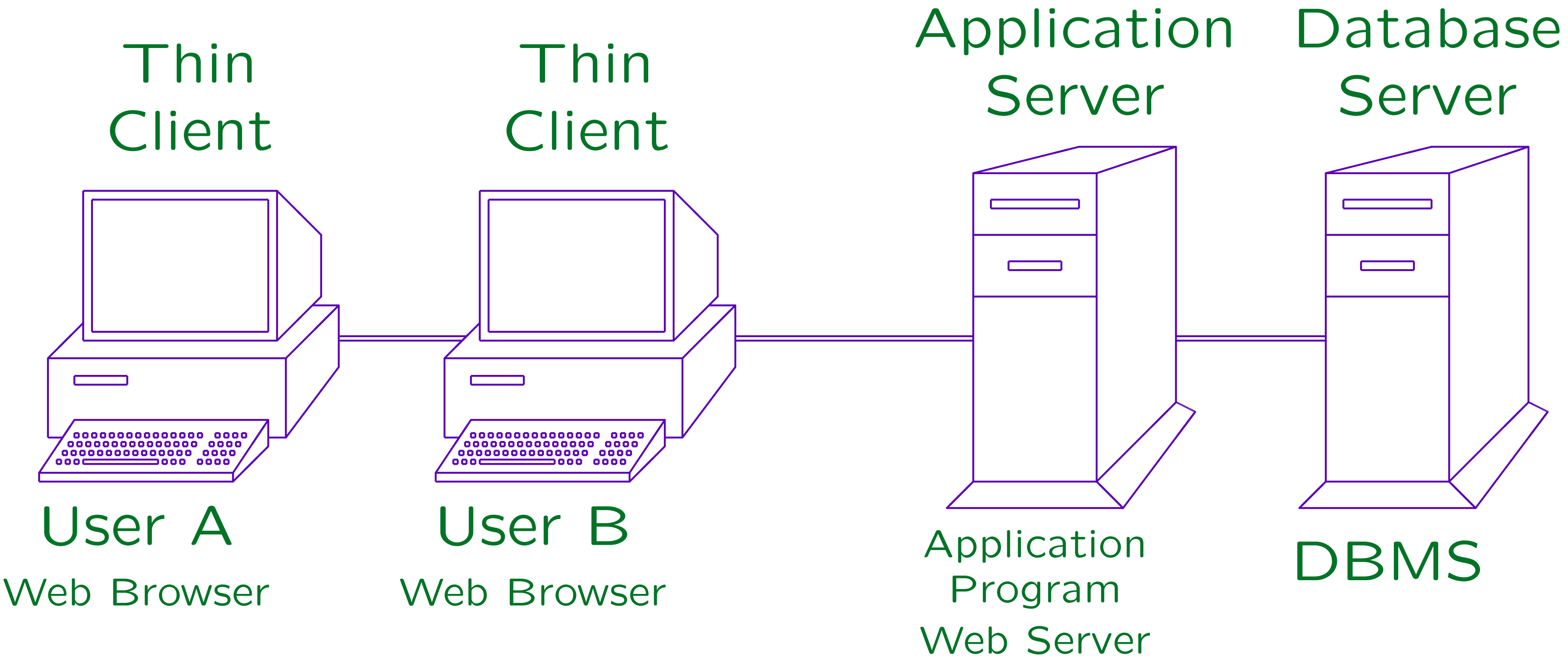
DB Application Systems (2)

Client-Server Architecture:



DB Application Systems (3)

Three-Tier Architecture:



DB Application Systems (4)

Exercise:

- Consider a database used by a bank for managing checking accounts.
- Which tasks might be supported by different application programs that access this database?
  - ◇ ______________________________
  - ◇ ______________________________
  - ◇ ______________________________
  - ◇ ______________________________

DB Application Systems (5)

Some Database Vocabulary:

- A database consists of DB schema and DB state.

  E.g. one says "The homeworks database". It depends on the context whether one means the current state, or basically only the schema and the storage space or location on the net, i.e. a place where one can always look for the current state. It is wrong to call a single table or a single file a database unless that contains all data of the schema.

- A database system (DBS) consists of a DBMS and a database.

  But database systems is also often used as an abbreviation for DBMS.

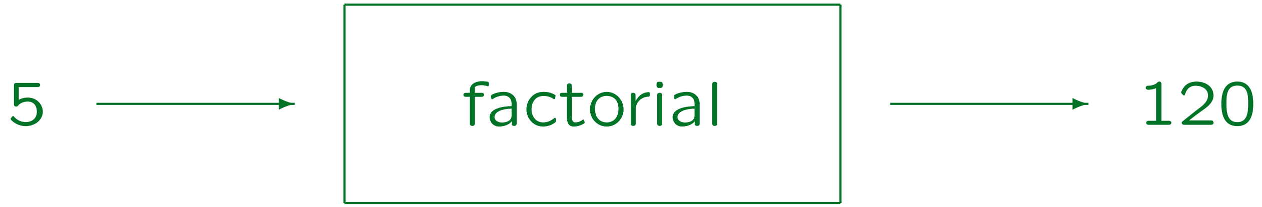
- A database application system consists of a DBS and a set of application programs.

Overview

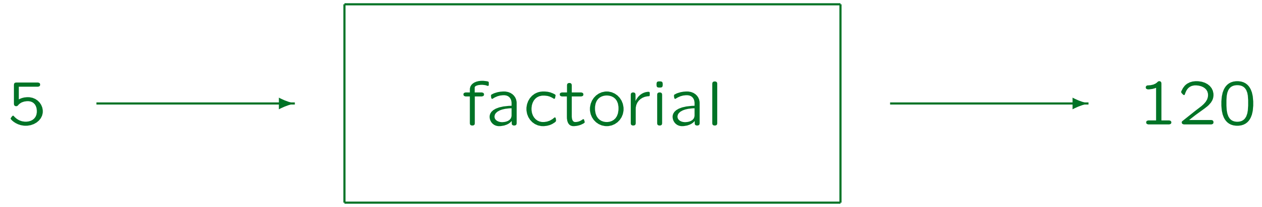
1. Basic Database Notions
2. Database Management Systems (DBMS)
3. Programmer's View, Data Independence
4. DBMS Vendors
5. Database Users and Database Tools

Persistent Storage (1)

Today:



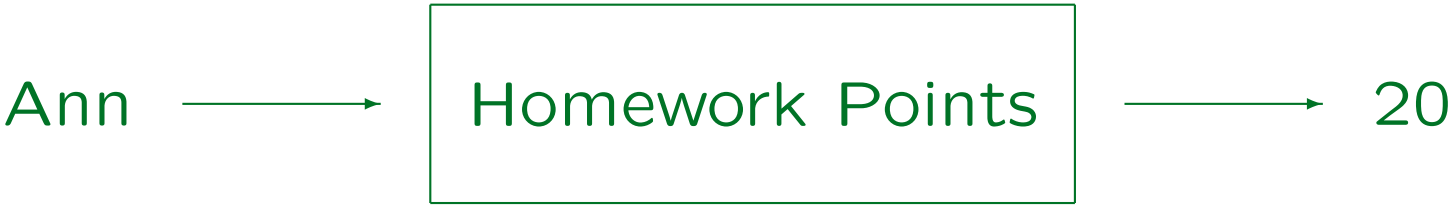
Tomorrow:



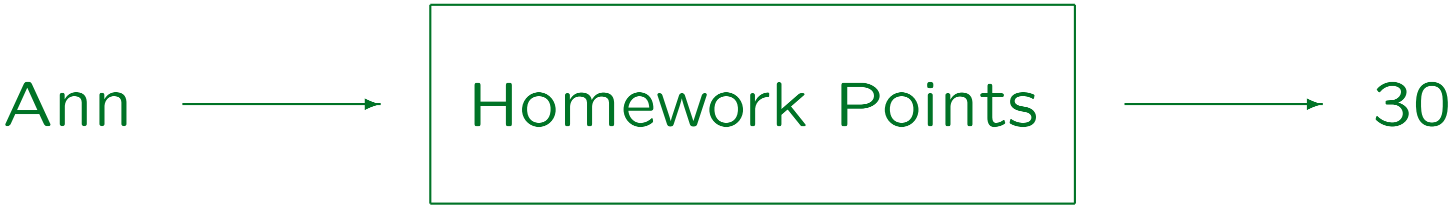
⇒ No persistent storage necessary.
The output is a function of the input only.

Persistent Storage (2)

Today:

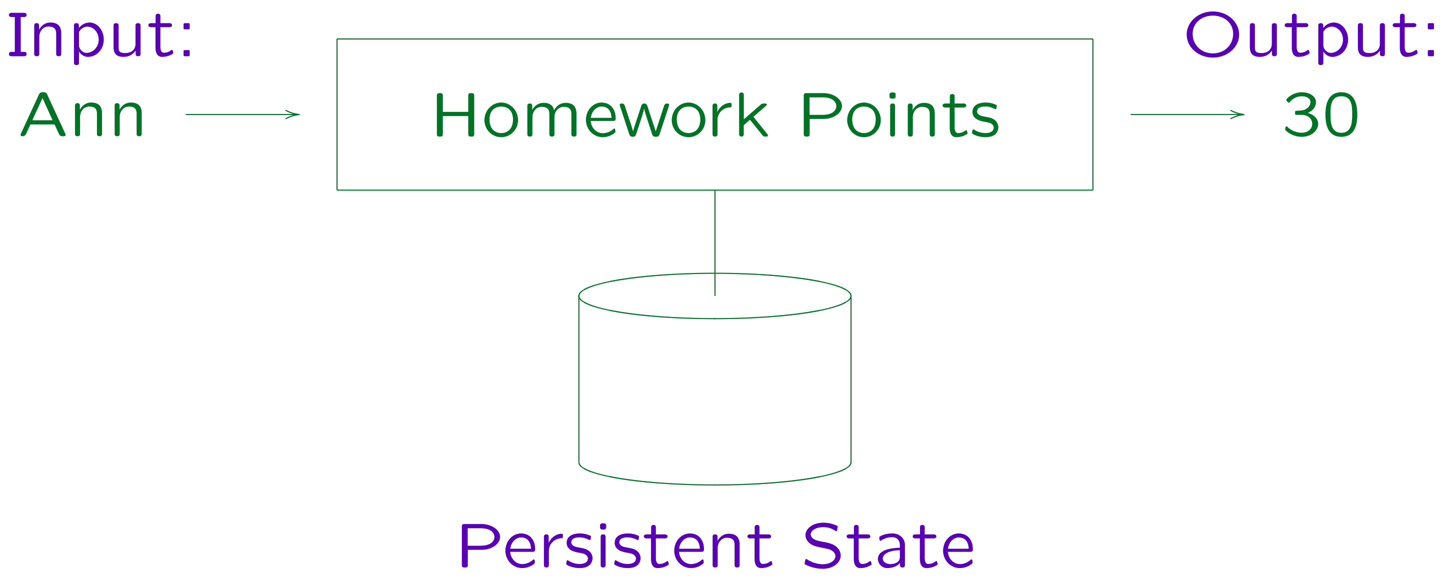


Tomorrow:



⇒ Output is a function of the input and a persistent state.

Persistent Storage (3)



Persistent Information:

- Information that lives longer than a single process (program execution). Survives power outage and a reboot of the operating system.

Persistent Storage (4)

Exercise:

- Which of the following programs need persistent storage? Why?
  - ◇ Pocket Calculator (non programmable)
  - ◇ Web Browser
  - ◇ Screen Saver
- How persistent is the memory of your video recorder for channels?
- Is information in the Windows Registry persistent?

Typed Persistent Data (1)

Classical Way to Implement Persistence:

- Information needed in other program invocations is saved in a file.
- The operating system (OS) stores the file on a disk.
- Disk is persistent memory: The contents is not lost if the computer is switched off or the operating system is rebooted.
- File systems are predecessors of modern database management systems.

Typed Persistent Data (2)

Implementing Persistence with Files:

- OS files are usually only sequences of bytes.
- A record structure must be defined as in Assembler languages.

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

- File structure information is contained only in the programmers' heads.
- The system cannot prevent errors because it does not know the file structure.

Typed Persistent Data (3)

Implementing Persistence with a DBMS:

- The structure of the information to be stored must be defined in a way the system understands:

```
CREATE TABLE SOLVED(STUDENT  VARCHAR(40),
                    HOMEWORK NUMERIC(2),
                    POINTS   NUMERIC(2))
```

- Thus, the file structure is formally documented.
- The system can detect type errors in application programs.
- Simplified programming (higher abstraction level).

A Subprogram Library (1)

- Most DBMS use OS files to store the data.

  Some use direct disk access for performance reasons.

- One can view a DBMS as a subprogram library that can be used for file accesses.

- Compared with the direct OS calls for file accesses, the DBMS offers higher level operations.

- I.e. it contains already many algorithms that one would otherwise have to program.

  Reusing well-debugged code in the DBMS can save a lot of programming efforts!

A Subprogram Library (2)

- For instance, a DBMS contains routines for
  - ◇ Sorting (Mergesort)
  - ◇ Searching (B-Trees)
  - ◇ File Space Management, Buffer Management
  - ◇ Aggregation, Statistical Evaluation
- Optimized for large data sets (that do not fit into main memory).
- It also has multi-user support (automatic locking) and safety measures to protect the data in case of system crashes (see below).

Data Independence (1)

- The DBMS is a layer of software above the OS files. The files can be accessed only via the DBMS.
- Indirection gives the possibility of hiding internal changes.
- Idea of abstract data types:
  Change the implementation, but keep the interface.
- Here the implementation is the file structure, which has to be changed for performance reasons.
  The application program interface is kept stable.

Data Independence (2)

Typical Example:

- At the beginning, a professor used the homeworks database only for his courses in the current term.

  For simplicity, the table "SOLVED" shown above permits only one course. But there might be an additional column to distinguish courses.

- Since the database was small, and there were relatively few accesses, it was sufficient to store the data as a "heap file".

  I.e. the rows of the table (records) are stored without any particular order. In order to evaluate queries, the DBMS must then do a "full table scan", i.e. read every row of the table and check whether it satisfies the query condition. For small tables, this is no problem.

Data Independence (3)

- Later the entire university used the database, and information of previous courses had to be kept for some time.
- Thus, the DB became much bigger and was also accessed more frequently.

  The “system workload” has changed.
- An index (e.g. B-tree) is needed for faster access.

  An index in a book is an ordered list of keywords together with references to page numbers where these keywords occur. A B-tree basically contains an ordered list of column values together with references to rows that contain these values (see chapter on physical design).

Data Independence (4)

Without DBMS (or with a Pre-Relational DBMS):

- Using the index to access the file must be explicitly mentioned in the query command.
- Thus, application programs must be changed if the file structure is changed.
- If one forgets to change a seldom used application program, and it does not update the index when the table is changed, the DB becomes inconsistent.

  This can happen only if one works directly with OS files. Already e.g. DBMS for the network data model did update indexes automatically. However, query commands had to refer explicitly to the index.

Data Independence (5)

With Relational DBMS:

- The system hides the existence of indexes at the interface.
- Queries and updates do not have to (and cannot) refer to the index.
- The system automatically
  - ◇ modifies the index in case of updates,
  - ◇ uses the index to evaluate queries when advantageous.

Data Independence (6)

Conceptual Schema (“Interface”):

- Only logical information content of the database
- Simplified View: Storage details hidden.

Internal/Physical Schema (“Implementation”):

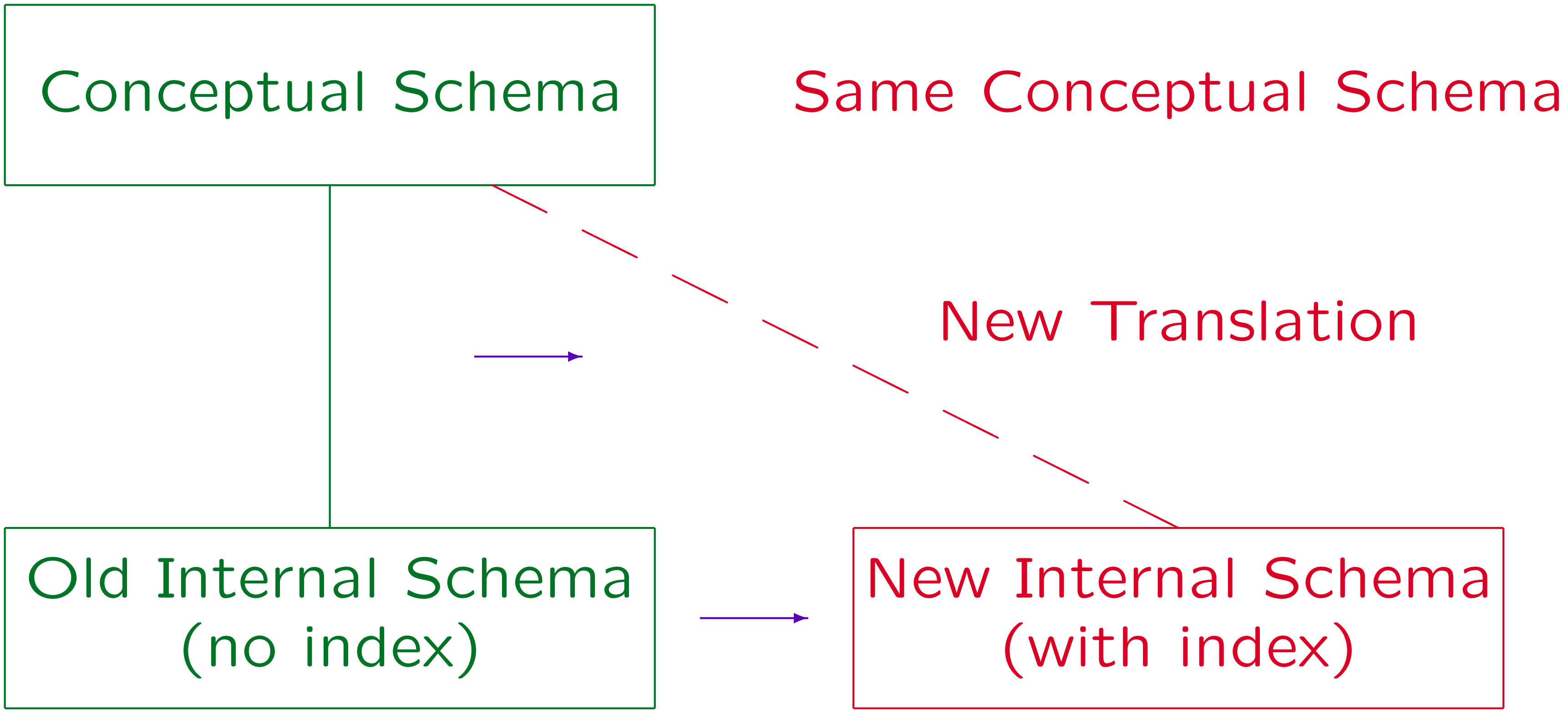
- Indexes
- Division of tables among disks
- Storage management if tables grow or shrink
- Physical placement of new rows in a table.

  E.g. store rows with the same column value in the same disk block.

Data Independence (7)

1. The user enters a query (e.g. in SQL) that refers to the conceptual schema.
2. The DBMS translates this into a query/program ("execution plan") which refers to the internal schema (this is done by the "query optimizer").
3. The DBMS executes the translated query on the stored instance of the internal schema.
4. The DBMS translates the result back to the conceptual level.

Data Independence (8)



Conceptual Schema

Same Conceptual Schema

New Translation

Old Internal Schema (no index)

New Internal Schema (with index)

Declarative Languages (1)

- Physical data independence requires that the query language cannot refer to indexes.
- Declarative query languages go one step further:
  - ◇ Queries should describe only what information is sought,
  - ◇ but should not prescribe any particular method how to compute this information.
- Algorithm = Logic + Control (Kowalski)

  Imperative/Procedural Language: Explicit Control, Implicit Logic.
  Declarative/Descriptive Language: Explicit Logic, Implicit Control.

Declarative Languages (2)

- SQL is a declarative query language. The user specifies only conditions for the requested data:

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

- Often simpler formulations: The user does not have to think about efficient execution.
- Much shorter than imperative programming: Less expensive program development/maintainance.

Declarative Languages (3)

- Declarative query languages
  - ◇ allow powerful optimizers

    because they do not prescribe a query evaluation method.
  - ◇ need powerful optimizers

    because the naive evaluation algorithm would be too inefficient.
- Larger independence of current hardware/software technology:
  - ◇ Simpler Parallelization
  - ◇ Today's queries will use tomorrow's algorithms when a new version of the DBMS is released.

“Data Independence”

- Decoupling between programs and data.
- Data is an independent resource by itself.

  Earlier it had meaning only in the context of the application programs for which it was originally collected.
- Physical data independence:
  - ◇ Programs should not depend on data storage methods.
  - ◇ Vice versa, the file structures are not determined by the programs.

  ⇒ Protects investments in programs and data.

Logical Data Independence (1)

- Logical data independence allows changes to the logical information content of the database.
- Of course, information can only be added, e.g. add a column “SUBMISSION_DATE” to the table SOLVED.

  Or represent information differently, e.g. use 24h-notation instead of AM/PM, inch instead of cm, combined first and last name instead of separate columns.

- This may be required for new applications.
- It should not be necessary to change old applications, although the records are now longer.

Logical Data Independence (2)

- Logical data independence is only important when there are application programs with distinct, but overlapping information needs.
- Logical data independence also helps to integrate previously distinct databases.
  - ◇ In earlier times, every department of a company had its own database / data files.
  - ◇ Now, businesses generally aim at one central DB.

    It may be distributed, but this is another issue.

Logical Data Independence (3)

- If a company has more than one DB, the information in these databases will normally overlap, i.e. some pieces of information are stored several times.
- Data is called redundant if it can be derived from other data and knowledge about the application.
- Problems:
  - ◇ Duplicates data entry and update efforts.
  - ◇ Sooner or later one will forget to modify one copy (data becomes inconsistent).
  - ◇ Wastes storage space, also on backup tapes.

Logical Data Independence (4)

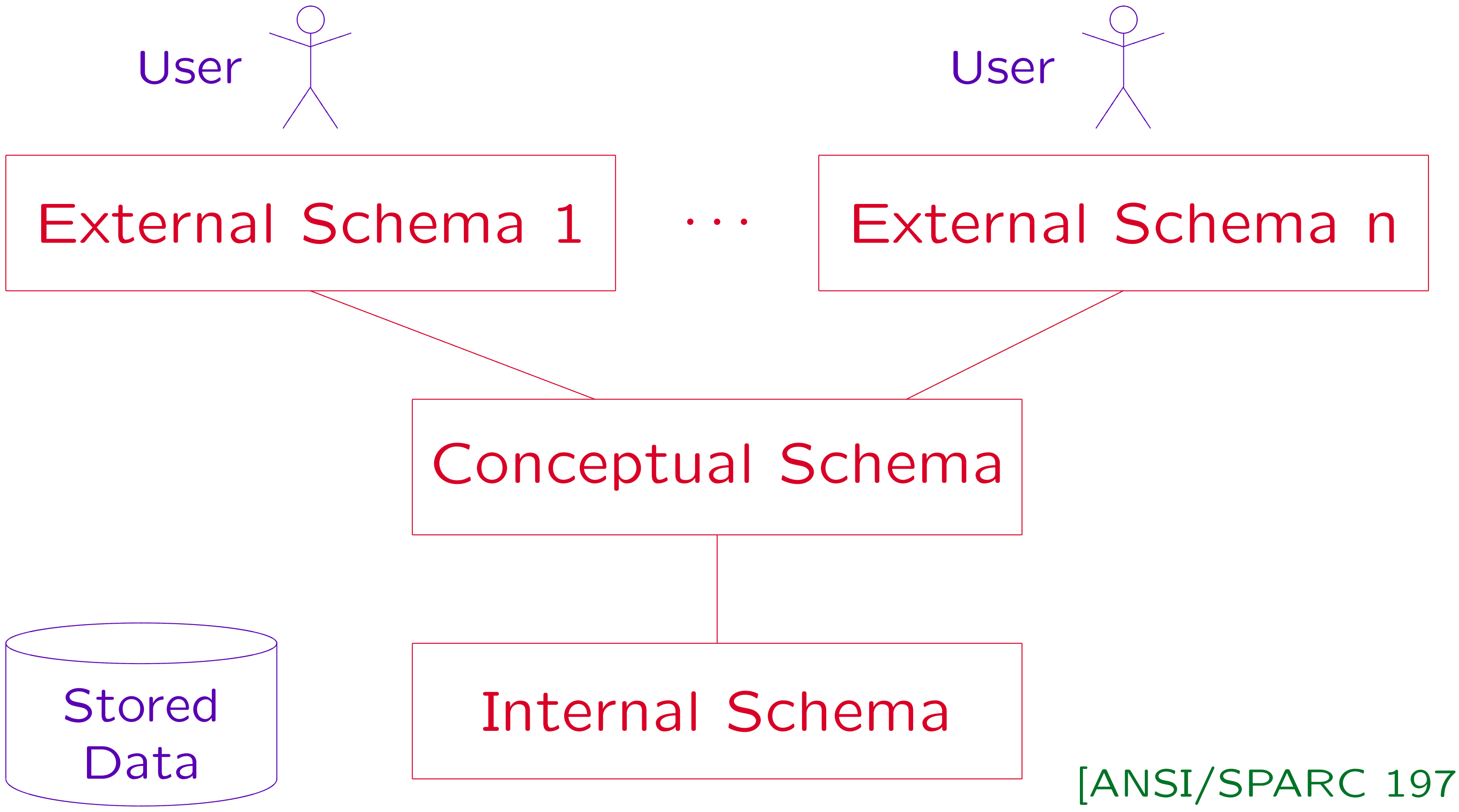
External Schemas/Views:

- Logical data independence requires a third level of database schemas, the external schemas/views.
- Each user can have an individual view of the data.
- An external view contains a subset of the information in the database, maybe slightly restructured.

  External views are also important for security reasons: They describe the subset of the information a user (group) can access.

- In contrast, the conceptual schema describes the complete information content of the database.

Three-Schema Architecture



[ANSI/SPARC 1978]

More DBMS Functions (1)

Transactions:

- Sequence of DB commands that are executed as an atomic unit ("all or nothing").

  If the system should crash in the middle of a transaction, all changes are undone ("rolled back") when the DBMS starts the next time. If the system should crash after a transaction is completed ("committed"), all changes are guaranteed to be stored in the DB state.

- Support for Backup and Recovery

  Data of completed transactions should survive single disk failures.

- Support of concurrent users

  Users/Programmers should not have to think about concurrent accesses by other users (DBMS does e.g. automatic locking).

More DBMS Functions (2)

Security:

- Access rights: Who may do what on which table.

  Actually, it is even possible to permit accesses only to part of a table or only to aggregated data.

- Auditing: The DBMS may remember who did what.

Integrity:

- It is possible to let the DBMS check that the entered data are plausible.
- The DBMS can also reject updates that would violate defined business rules.

More DBMS Functions (3)

Data Dictionary:

- Information about the data (e.g. schema, user list, access rights) is available in system tables, e.g.:

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

Overview

1. Basic Database Notions

2. Database Management Systems (DBMS)

3. Programmer's View, Data Independence

4. DBMS Vendors

5. Database Users and Database Tools

DBMS Vendors (1)

- Oracle: Oracle 9i

  Standard Edition, Enterprise Edition, Personal Oracle, Oracle Lite.

- IBM: DB2 Universal Database V8.1 (plus e.g. IMS)
- Informix: e.g. Informix Dynamic Server 9.30

  Informix was bought in 2001 by IBM (for $ 1000 Mio).

- Sybase: Adaptive Server Enterprise 12.5

  Other DBMS products: SQL Anywhere Studio 8, Adaptive Server IQ.

- Microsoft: SQL Server 2000 (plus Access, FoxPro)

  SQL Server started 1988 as a port of Sybase to OS/2. Later, Microsoft developed it further on its own (1994: formal end of partnership).

DBMS Vendors (2)

DBMS Market 1998 [Dataquest Study]

Vendor	Market Share	Change to 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%
Others	20.5%	−2.2%

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

DBMS Vendors (3)

RDBMS Market Share (1998) [Dataquest]

Vendor	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%
Others	14%	11.8%	12%
Market Size:	$5400 Mio.	$2200 Mio	$1200 Mio.
Growth Rate:	+18%	+10%	+46%

DBMS Vendors (4)

Database Market Share 2001 [in %, New License Sales]:

Vendor	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
old IBM	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
Others	14.4	−2.8	8.5	−7.5	7.4	−2.0	3.7	−10.7
Total	8844 Mio $		7108 Mio $		3014 Mio $		2555 Mio $	
Growth	1.4%		1.6%		-1.4%		11.0%	

Source: Gartner Dataquest (May 2002) [UNIX, Windows: only RDBMS]

DBMS Vendors (5)

- Oracle cites a survey of Fortune 100 companies (by the FactPoint Group) in which 400 IT managers were asked for their primary database choice:
  - ◇ Oracle: 51%
  - ◇ IBM DB2: 22% (19% on mainframe, 3% on UNIX/NT)
  - ◇ Microsoft SQL Server: 8%
  - ◇ Combinations of Vendors: 15%
  - ◇ Other: 4%
- Oracle is used by 93% of these companies.
- 76% of their SAP installations run on top of Oracle.

Selection Criteria (1)

- Price, Licence conditions.

  One should also consider the price for support and later updates. There are also "Total Cost of Ownership" calculations.

- Availability for different hardware platforms.
- Performance [http://www.tpc.org], scalability.
- Availability of tools.

  For developing application programs.

- Knowledge of employees, cost of training.
- Amount of time needed for administration.

Selection Criteria (2)

- Reliability, support for 7 × 24 operation.

  How good is the support for backup and recovery? How quick is the recovery if it should be needed? Is a failover system supported?

- Support for security.

  And how likely are bugs that permit hackers to break in? How fast are security problems solved? How well is the DBA informed about security problems? How easy is it to apply patches?

- SQL is more or less standard.

  Every modern relational DBMS (RDBMS) supports at least the SQL-86 standard or the entry level of the SQL-92 standard. But switching from one DBMS to another one can nevertheless be costly. Every vendor has certain extensions to the SQL standard. Also many programming tools exist only for a specific vendor.

Disadvantages of DBMS

- Expensive
- Dependence on the DBMS supplier

  There are standards for SQL, but every system has extensions and special tools. DBMS software is also often tightly coupled with the OS because they perform in part similar tasks. When the OS is updated, it might be necessary to get an update of the DBMS, too.

- Requires quite a lot of time to learn.

  Oracle 8 had $\geq$ 95 volumes (= 1.70 m) of documentation.

- Overhead: A hand-optimized C-routine is faster than the general purpose code in the DBMS.

When Not to Use a DBMS

- The data will be processed by a single program.

  No other applications on the same data are planned.
- A DBMS seems unusable for various reasons, e.g.:
  - ◇ Response time must be very short (realtime),
  - ◇ non-standard locking needed.
- The redevelopment is not too expensive:
  - ◇ The structure of the data is simple.
  - ◇ All data fits into main memory.
  - ◇ A simple backup strategy is sufficient.

Overview

1. Basic Database Notions

2. Database Management Systems (DBMS)

3. Programmer's View, Data Independence

4. DBMS Vendors

5. Database Users and Database Tools

Database Users (1)

Database Administrator (DBA):

- Should know the complete database schema.

  Changes the DB schema if needed, documents such changes.

- Gives access rights to users. Ensures security.
- Monitors system performance.

  Does performance tuning.

- Monitors available disk space and installs new disks.
- Ensures that backup copies of the data are made. Does the recovery after disk failures etc.

Database Users (2)

Database Administrator, Continued:

- Installs new versions of the DBMS software.
- Tries to ensure data correctness.
- Responsible for licence agreement.
- Contact for support / DBMS vendor.
- Expert on the DBMS software.
- Can damage everything.

  Sometimes needs powerful privileges for the operating system.

Database Users (3)

Application Programmer:

- Writes programs for standard tasks.

  "Application programs" to be used by the "naive users" (see below). Today, this might also include a web interface.

- Knows SQL well, plus some programming languages and development tools.
- Usually supervised by the database administrator.

  Or outside consultant who leaves after the project is finished.

- Might do DB design (i.e. develop the DB schema).

  But there are specialists for this task (analysts, data modelers).

Database Users (4)

Sophisticated User (one kind of “End User”):

- Knows SQL and/or some query tools.
- Does non-standard evaluations of the data without help from application programmers.

  E.g. some kind middle manager who needs some new statistics for decision support.

Naive User (the other kind of “End User”):

- Uses the database only via application programs.
- Does the real work of entering data.

Database Tools

- Interactive SQL interpreter
- Graphical/Menu-based query tools
- Interface for DB access from standard programming languages (C, Pascal, Java, . . .)
- Tools for form-based DB applications (4GL)
- Report generator
- WWW interface
- Tools for import/export of data, backup&recovery, performance monitoring, . . .

Knowing Oracle (1)

Core:

- Relational Model, Fundamentals of DB Design
- SQL: SQL-92 (standard) plus Oracle deviations
- Data Types and Data Type Functions
- Data Dictionary (system tables)
- PL/SQL for server-side stored procedures, triggers.

  PL/SQL is Oracle's own programming language (similar to Ada), tightly coupled with SQL. Now Java can be used as an alternative.
- SQL*Plus: Output Formatting, Scripts
- Where to look in the Oracle Documentation.

Knowing Oracle (2)

Database Administration:

- SQL: Oracle Extensions, e.g. for physical storage parameters, user management.
- Oracle Process Structure, Installation Parameters, Tuning, Internal Data Structures.
- Administration Tools: Starting and Stopping the DB Server, Adding Disk Space, etc.
- Backup & Recovery in Oracle
- Import/Export-Utilities, SQL*Loader

Knowing Oracle (3)

Application Programming:

- Oracle Pro*C (SQL embedded in C), ODBC
- Java-Interfaces: JDBC, JSQL
- Oracle Developer (iDS): Visual programming tools for creating end-user application programs.

  E.g. Form Builder: Creates programs which are something like specialized editors for specific relations, e.g. showing one record at a time in a screen mask. Oracle Developer contains also Report Builder, Graphics Builder, etc.
- Oracle Application Server (iAS): for web interfaces
- Some knowledge about security, e.g. for web interfaces.

Summary (1)

Functions of Database Systems:

- Persistence
- Integration / No Redundancy (duplicate storage)
- Data Independence
- Less programming effort: Many algorithms built-in, especially for external memory (disks)
- Ad-hoc Queries

  I.e. when somebody gets an idea for a new query, he/she can immediately pose it. In earlier times, he/she had to ask the programming department for a new program.

Summary (2)

Functions of Database Systems, continued:

- High data security (Backup & Recovery)
- Combinition of updates into atomic transactions

  Transactions are completely executed or not at all
- Multi-User: synchronization of concurrent accesses
- Integrity Enforcement
- Views for different users (user groups)
- Data Access Control
- System Catalog (Data Dictionary)

Summary (3)

- The main goal of the DBMS is to give the user a simplified view on the persistant storage, i.e. to make complications "transparent".
- The user does not have to worry about:
  - ◇ Physical storage details
  - ◇ Different information needs of different users
  - ◇ Efficient query formulation
  - ◇ Possibility of system crashes / disk failures
  - ◇ Possibility of concurrent access by other users

Exercise

- Take the task of developing a system where students can vote about the quality of lectures.

  There is a form in the world wide web, where students can enter their data, and when they press the "submit" button, these data are stored on the web server. Later evaluation is performed on the collected data, e.g. computing average values.

- I have proposed writing C programs for this task, and storing the data in UNIX files.
- What arguments can you make as to why using a database system might be better?

Exercises (1)

- Take the task of developing a system where students can vote about the quality of lectures.

  There is a form in the world wide web, where students can enter their data, and when they press the "submit" button, these data are stored on the web server. Later evaluation is performed on the collected data, e.g. computing average values.

- I have proposed writing C programs for this task, and storing the data in UNIX files.
- What arguments can you make as to why using a database system might be better?

Exercises (2)

- Suppose the homework points are stored in a text file with the format

  Student Name:Homework Number:Points

  (i.e. one row of the table `SOLVED` per line).
- Suppose you have to write a C program that prints the total number of points per student (students ordered alphabetically).
- How many lines and how much programming time do you need?
- In SQL, this needs 4 lines and 2 minutes work.