1.2 Requirements and Advantages of Database Systems

Data independence

- independence of application programs from the details of data representation and data storage (abstract view on data)

Efficient data access

- multitude of sophisticated techniques for the efficient storage of and efficient access to persistent data
- application of **index structures**

Common data basis

- common data basis for all current and future application programs

Concurrent data access

- simultaneous access to the same data by different users
- each user gets the impression of exclusively accessing data
- concept of **transaction** for the **synchronisation** of concurrent data accesses

Lacking or **controlled redundancy**

- avoiding copies of the same data by an integrated view on data
- controlled redundancy for improving **performance**

Consistency of data

- caused by lacking redundancy
- DBMS must ensure consistency of data for controlled redundancy

Integrity of data

- correctness and completeness of data (semantical aspect)
- formulation of **integrity constraints** or **integrity rules**
- DBMS checks constraints for each insertion, change and deletion of data

Data security

- protection of the database against unauthorized access (**view** on data)
- **access control** with authentication and encoding as possible protection mechanisms

Backup and **recovery**

- protection against the consequences of system errors
- backup: copies on external storage media (e.g. tapes)
- recovery: automatic reconstruction of the last, up-to-date, and consistent database status with the aid of tapes and a protocol listing the executed changes

Posing **queries**

- traditional data organization: posing queries by an application program, extremely inflexible
- DBMS: **query language**, which allows, to pose *ad hoc* queries by using the keyboard and to get an immediate answer
- importance of an efficient **query optimization**, **query processing** and **query execution**

Provision of diversified **user interfaces**

- query languages for occasional users, programming interfaces for implementors, menu-driven interfaces for naive users, window- and graphic-oriented user interfaces
- **data definition languages**, **data manipulation languages**

Flexibility

- change of the database structure without change of existing application programs (e.g., extension of a record by a new field, insertion of a new collection of data into the database)
- different evaluations of data possible in an easy way

Faster developments of new applications

- use of the powerful functions of a DBMS for new applications

1.3 Fundamental Terms

Database (DB)

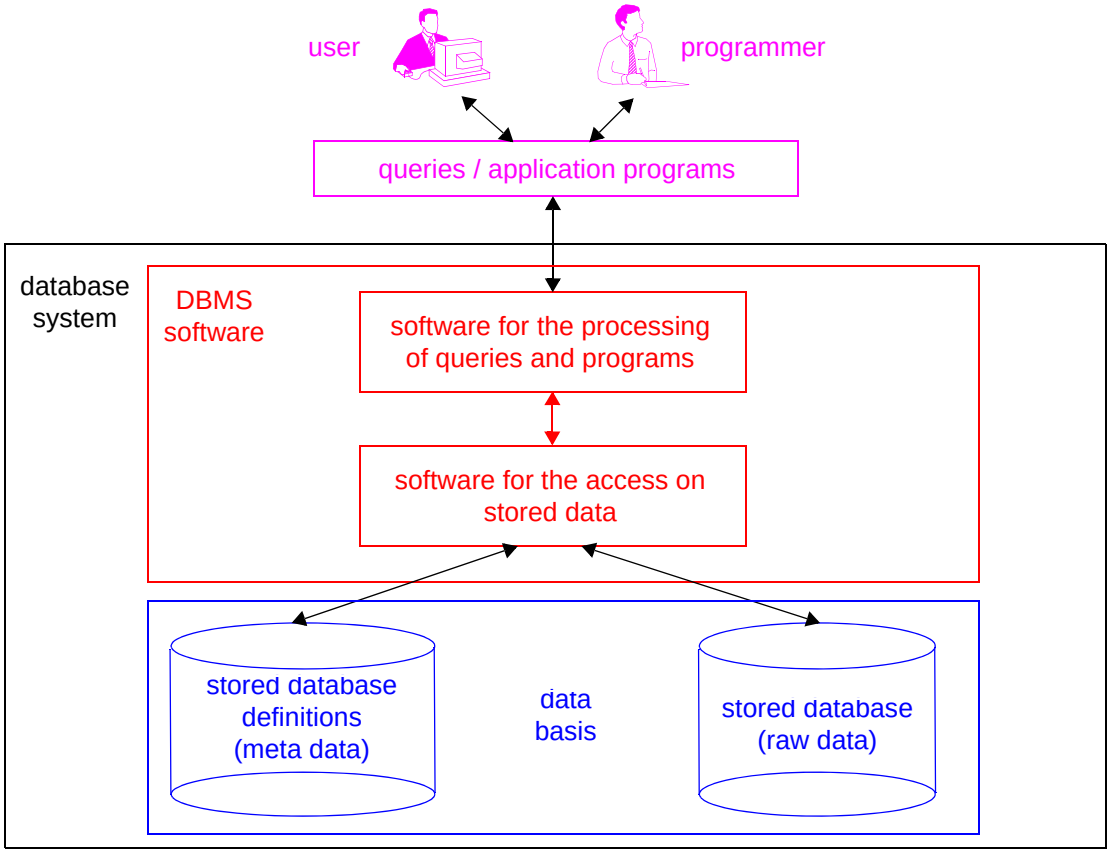
- integrated and structured repository of large collections of persistent data, which serves for all users of an application area as a common and reliable basis of up-to-date information
- frequently used synonymous term: **data basis**

Database Management System (DBMS)

- all-purpose software system, which supports the user in the definition, construction and manipulation of databases for different applications in an application-neutral and efficient manner
- set of programs for the management of and access to the data in the DB
- software level between physical database and user

Database System(DBS)

- DBS = DBMS + DB



Data Model

- mathematical formalism consisting of a notation for describing the data of interest and of a set of operations for manipulating these data
- description of the structure of a database (data types, relationships, conditions)

1.4 Data Model

- a data model offers facilities
  - for the specification of data objects
  - for the specification of the relationship between data and
  - for the specification of operations on data objects together with their semantics

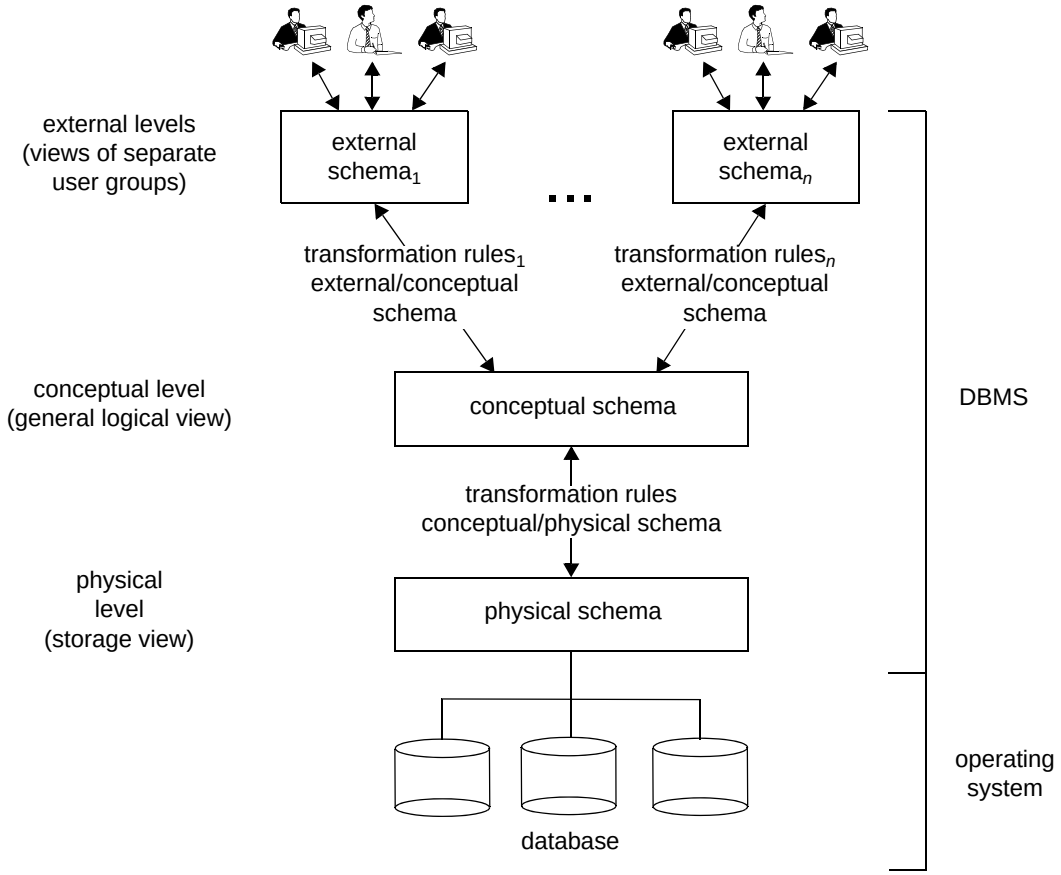
- usually a DBS has at least two data models
  - **physical data models** for the storage-oriented representation of data
  - **logical data models** for the user-oriented representation of data

- logical data models
  - object-based, e.g.
    entity-relationship model
    object-oriented data model
    object-relational data model
  - record-based, e.g.
    object-relational data model
    relational data model
    network data model
    hierarchical data model

1.5 Data Abstraction: The Three-Level Model

DBS has several abstraction levels

- **external/view levels** describe the part of the DB, which is relevant for the user
- **conceptual/logical level** gives information about existing data and relationships in the DB
- **physical/internal level** describes how data are physically stored



Database schema and state

- a schema describes the structure/the design of a DB
- a state describes a concrete instance of a DB

1.6 Data Independence

- denotes the property that higher levels of the model are not influenced by changes of lower levels

- **logical data independence**
  - changes of the conceptual schema (e.g., information about new types of entities, further information about existing entities) do not have impact on external schemas (e.g., existing application programs)
  - example: extension of the class data for an additional boolean value expressing whether the merits required for a master thesis have been performed

- **physical data independence**
  - changes of the physical schema (e.g., change of an access structure to a more efficient one, use of other data structures, exchange of algorithms) do not have impact on the conceptual schema and thus also not to external schemas
  - example: the class data, which are so far stored in an unsorted file, are to be reorganized in a B-tree to enable efficient access by “registration number”.

1.7 Database Languages

Data definition language (**DDL**)

- language to manipulate a database schema
- (meta) data for the description of a schema (**data dictionary**, **system catalog**)
- permits the specification of implementation details

Data manipulation language (**DML**)

- **query language** for the retrieval of data objects in a database
- "actual" data manipulation language for the change of stored data objects, for the insertion of new data, and for the deletion of stored data
- a query, which is formulated by a user with the objects of his/her external level, is translated into an efficient query, which rests on the objects of the physical level
- in general realized as a non-procedural language
  - user specifies *which* data are searched for but *not how* data can be found

Overview of the Database Software Project (I)

- Semester-long database software group project
- Main goal: Apply and practice the theoretically learned concepts in class in a professional and commercial database environment by means of the design and implementation of a web-based database application program

Project Objectives

- Transfer the database concepts learned in class, such as
  - the conceptual database design with the Entity-Relationship (ER) Model,
  - the transformation of an ER diagram into a relational database schema with a provided algorithm, and
  - the application of the synthesis algorithm for 3NF normalization,

  step by step into practice by deploying the professional and commercial database system Oracle

Overview of the Database Software Project (II)

Project Objectives (*continued*)

- In homework assignments students learn the formulation of SQL queries in an *ad hoc* mode
- In the project students learn how to embed SQL queries into database application programs
- Special aspect: Database application includes a *web-based user interface* as its front-end and a supporting Oracle database as its back end
- Preparation for database industry:
  - Provide students with a real hands-on database experience that will enable them to work later in industry in the database field
  - By working in groups, enable students to argue, discuss, compromise, write technical documents, and solve arising social conflict situations in the group at a professional level.

Overview of the Database Software Project (III)

General Description and Project Activities in Detail

- Identify an application area for which a DBMS may prove beneficial to store the data it processes
- Consider factors such as the need to store and query large data volumes, support multiple users, provide concurrent access, maintain consistency
- Find appropriate real world data that match your application and can be used to populate your database with at least 250,000 records
- Understand and analyze the available data
- Find problems in the data such as missing data and inconsistent data
- Determine the main functionalities and operations of the database application
- Think about the various requirements of the user of your application and the various data attributes that need to be stored and later queried

Overview of the Database Software Project (IV)

General Description and Project Activities in Detail (*continued*)

- Application development
  - Database development
    - Model the data to be stored in the database, i.e., identify the various entities, relationships, constraints, etc. by creating an ER diagram.
    - Transform the ER diagram into a relational database schema
    - Design, normalize, and perfect the relational database schema
    - Transform and upload the real world data according to the database schema into the database by using SQL queries, spreadsheets, CSV files, and/or stand-alone programs written in Java, C, C++, etc.
    - Design the SQL queries that will be embedded into the application program

Overview of the Database Software Project (V)

General Description and Project Activities in Detail (*continued*)

- Application development (*continued*)
  - User Interface (UI) development
    - Design the web interface for the application by considering the various "screens" and the "flow of control" of your application
    - Example “photo manager application”: Start with a user login screen, then a web page to display the user's photo in a gallery, then another one to display a specific picture with additional descriptive information, a search page, logout page, etc.
    - Select web-based technologies for designing web-based user interfaces, e.g., PHP and Ruby on Rails.
    - Implement the web interface and write supporting code to embed the designed SQL queries to retrieve data from the DBMS.
- Test your database application software and check if it works as desired

Organizational Issues (I)

Topic of the Group Project

- Each group selects an application topic on its own
- The instructor will not provide a topic
- Enables a group to be creative and follow its interests
- Important: Not sufficient to identify an interesting topic that is worthwhile to be supported by a database system
  → Find real world data that support the selected application

Project Group Size and Formation

- Students can form own 4-student groups
- Depending on the class size, 3-student groups or 5-student groups possible
- If the class size should be low, 3-students per group will be the default
- Each group will select tools from the Internet (e.g., email, Skype) for group communication

Organizational Issues (II)

Grading

- 4 project deliverables (discussed later in detail)
- Deliverables are produced by each group together
- All members of a group will get the same grade
- Exception: If it turns out that a group member has not adequately contributed to the group's efforts and therefore harmed the group, the instructor will take the right to assign a different and adequate grade to such a group member
- General information about the grading of the four project deliverables can be found in the syllabus

"250,000 Tuple" Rule

- A group's database must store at least 250,000 tuples (records) as the sum of all records stored in all database tables.

Organizational Issues (III)

Programming Environment

- User interface
  - Group's choice to use any high-level web programming language
  - Options: Ruby on Rails, PHP, .NET, JSP, Javascript, etc.
  - Teaching of web-based programming technologies is beyond the scope of this class
    → Self-study required

- Database interface
  - We will use the CISE Oracle database server
  - Needed: CISE account, Oracle account (see syllabus, course web page)
  - General information available on the CISE Oracle Database Help web page

Organizational Issues (IV)

Programming Environment (*continued*)

- Database interface (*continued*)
  - Connection between application program and database possible by
    - connectivity protocols such as ODBC and JDBC
    - web programming language specific methods provided by PHP, Perl, and Ruby on Rails
  - Simple coding examples for Java using JDBC, PHP using OCI8, Perl using DBD::Oracle, and Ruby using DBI available on the CISE Oracle Database Help web page
  - CISE help pages also contain information how to remotely access CISE Oracle
  - The project database must run on Oracle and use the *orcl* instance running on the CISE Oracle database server
  - Clients for testing SQL queries: SQL*Plus (command line) or Oracle SQL Developer (graphical)