DATABASTEKNIK - 1DL116

Fall 2003

An introductury course on database systems

http://user.it.uu.se/~udbl/dbt-ht2003/

Kjell Orsborn
Uppsala Database Laboratory
Department of Information Technology, Uppsala University,
Uppsala, Sweden

Personell - Spring 2003

- Kjell Orsborn, kursansvarig lärare
  - epost: kjell.orsborn@it.uu.se
  - tel: 471 1154
  - rum 1321
- Tore Risch,
  - epost: tore.risch@it.uu.se
  - tel: 471 6342
  - rum 1353
- Milena Koparanova, kursassistent
  - epost: milena.koparanova@it.uu.se
  - tel: 471 2846
  - rum 1316
- Johan Petrini, kursassistent
  - epost: johan.petrini@it.uu.se ,
  - tel: 471□1045
  - rum 1357

Preliminary course contents

- Course intro - overview of db technology
- DB terminology,
- ER-modeling, Extended ER
- Relational model and relational algebra
- ER/EER-to-relational mapping and Normalization
- SQL
- OO/OR DBMSs
- AMOS/AMOSQL
- Transactions, Concurrency Control
- Recovery Techniques
- Security / Authorization
- Storage and Index Structures
- Query optimization
- Distributed and Multi-DBMSs
- Active DBMSs
- Multimedia DBMSs
- Data warehousing / Data Mining
- Parallell DBMSs
- Relational calculus, QBE

Preliminary course contents

- Labs using Mimer
  - RDBMS
- Labs using AMOS II
  - OO/OR DBMS

- Lab Project XX
  - To be decided:
    Mimer alt. AMOS II

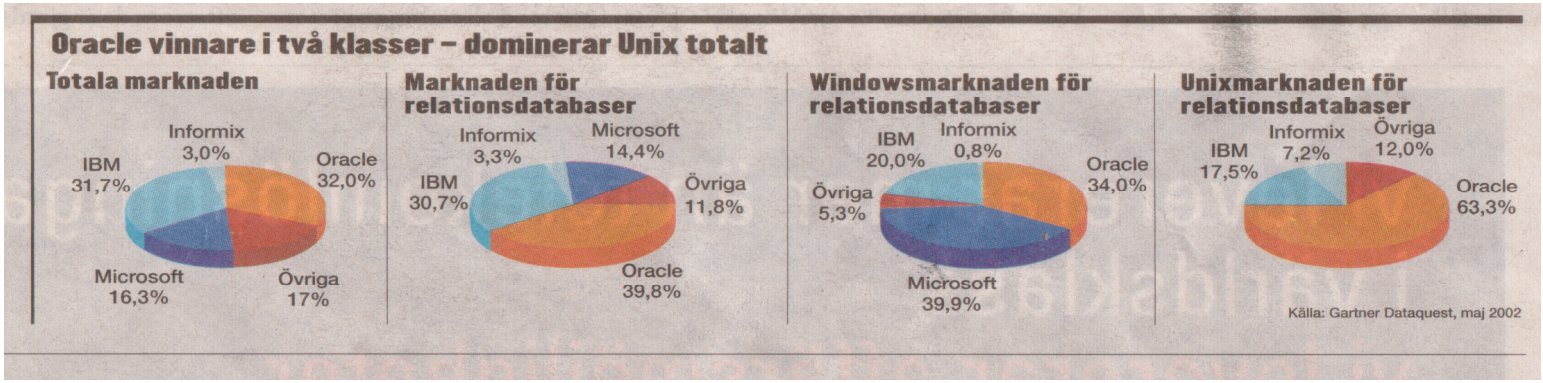
Introduction to Database Technology

Lecture 1

Kjell Orsborn

Department of Information Technology
Uppsala University, Uppsala, Sweden

The database market /CS 020524



ORACLE

Oracle9i Database

IBM

DB2 Universal Database

Informix Dynamic Server (IDS)

Microsoft

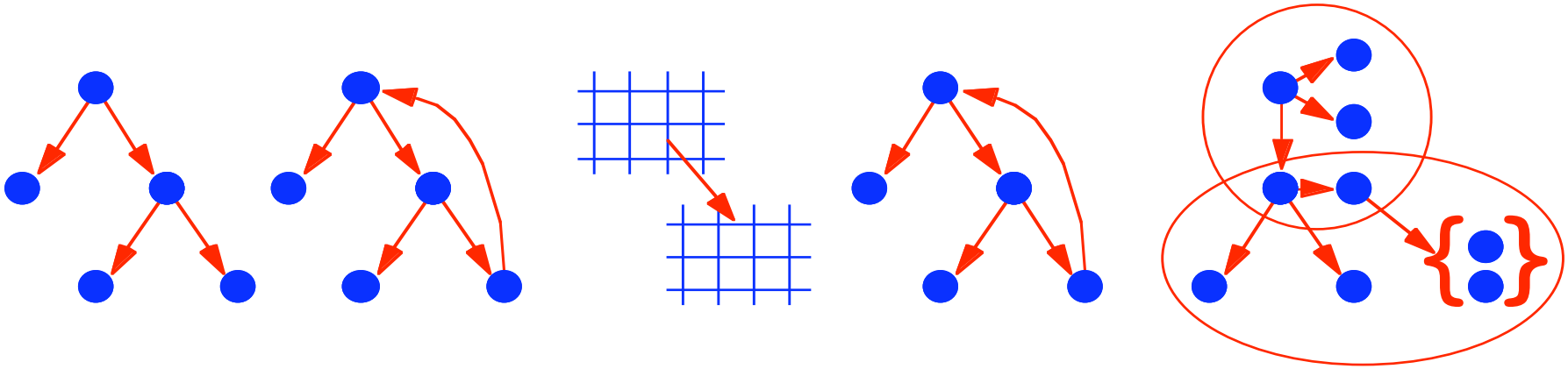
Microsoft SQL Server

Microsoft Access

The Office XP database solution

Evolution of Database Technology

1960	1970	1980	1990	1997
Hierarchical (IMS)	Network model (CODASYL)	Relational model (e.g. ORACLE)	1st Generation OODB (e.g. Objectivity)	Object Realtional DBMS (e.g. SQL99)
Trees	Complex data structures	Tables	OO data structures	Object model



Introduction to Database Terminology

Elmasri/Navathe chs 1-2

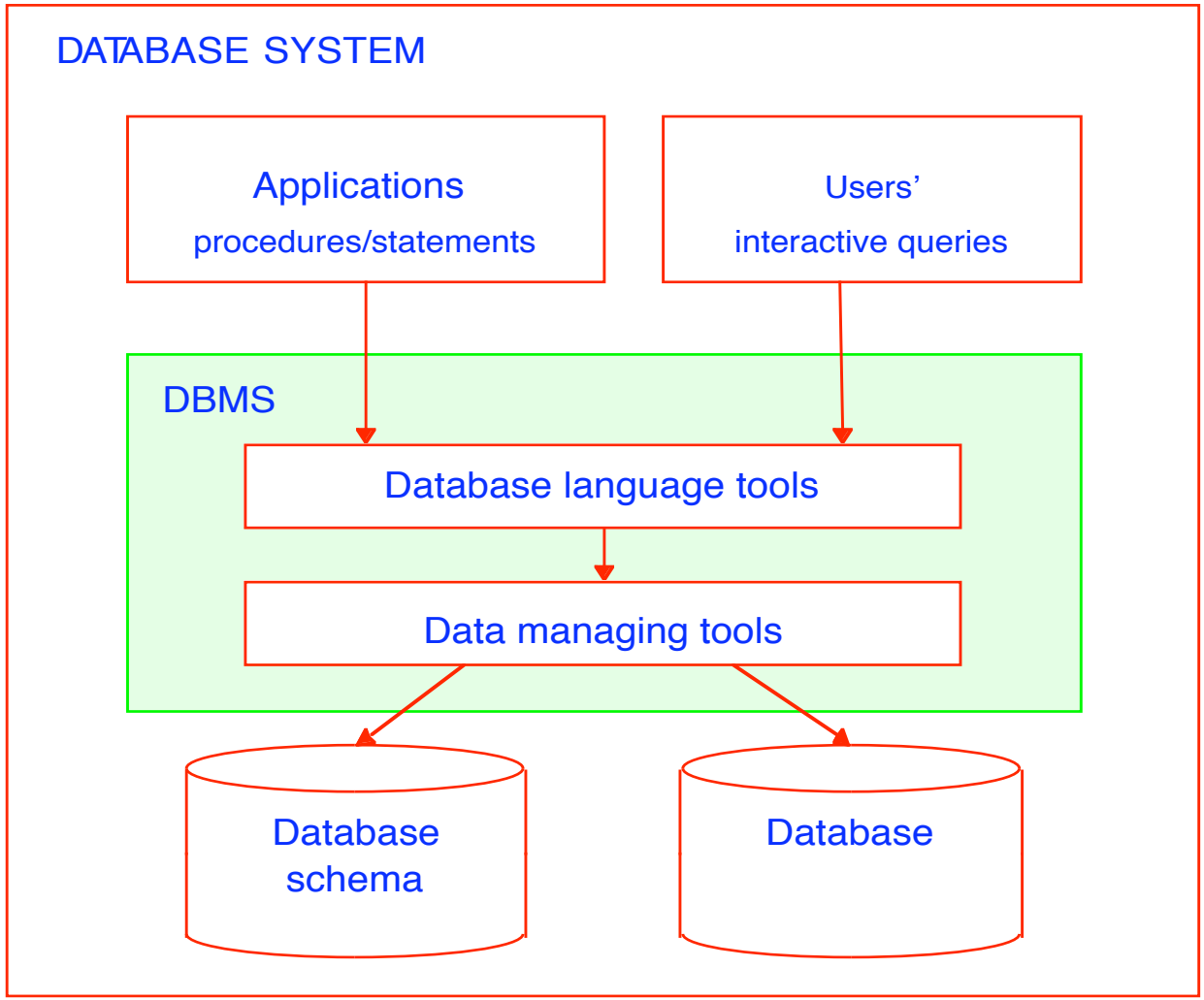
Kjell Orsborn

Department of Information Technology
Uppsala University, Uppsala, Sweden

Database?

- A **database** (DB) is a more or less well-organized collection of related *data*.
- The information in a database . . .
  - represents information within some subarea of "the reality" (i.e. objects, characteristics and relationships between objects)
  - is logically connected through the intended meaning
  - has been organized for a specific group of users and applications

Outline of a database system



An example database (Elmasri/Navathe fig. 1.2)

STUDENT

Name	StudentNumber	Class	Major
Smith	17	1	CS
Brown	8	2	CS

COURSE

CourseName	CourseNumber	CreditHours	Department
Intro to Computer Science	CS1310	4	CS
Data Structures	CS3320	4	CS
Discrete Mathematics	MATH2410	3	MATH
Database	CS3380	3	CS

SECTION

SectionIdentifier	CourseNumber	Semester	Year	Instructor
85	MATH2410	Fall	98	King
92	CS1310	Fall	98	Anderson
102	CS3320	Spring	99	Knuth
112	MATH2410	Fall	99	Chang
119	CS1310	Fall	99	Anderson
135	CS3380	Fall	99	Stone

GRADE_REPORT

StudentNumber	SectionIdentifier	Grade
17	112	B
17	119	C
8	85	A
8	92	A
8	102	B
8	135	A

PREREQUISITE

CourseNumber	PrerequisiteNumber
CS3380	CS3320
CS3380	MATH2410
CS3320	CS1310

Database management system?

- A **database management system** (DBMS) is one (or several) program that provides functionality for users to develop, use, and maintain a database.
- Thus, a DBMS is a *general* software system for *defining*, *populating (creating)*, *manipulating* and *sharing* databases for different types of applications.

Back to the example ...

- Defining this DB involve
  - declaration of files, records, fields and data types for each fields.
- Population of the DB means
  - that the files are filled with data about individual students, courses etc.
- Manipulation is then carried out by users directly via a query language or indirectly via application programs:
  - updates
  - queries to the DB
- Sharing a database
  - allows multiple users and applications to access the database concurrently

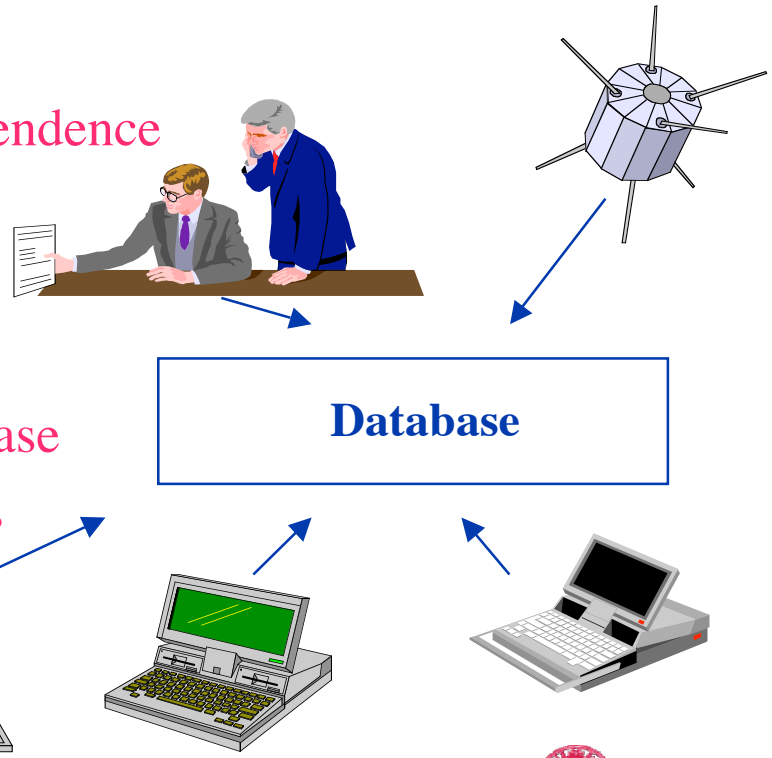
Database System?

- A **database system** consists of . . .
  - the physical database (instance)
  - a database management system
  - one or several database languages
    (means for communicating with the database)
  - one or several application program(s)
- A **database system** makes a *simple* and *efficient* manipulation of large data sets possible.
- The term DB can refer to both the content and to the system. The answer to this ambiguity is governed by the context.

Why DB?

- DB in comparison to conventional file management:
  - data model - data abstraction
  - meta-data - in catalog
  - program-data and program-operation independence
  - multiple views of data
  - sharing data - multiuser transactions

  Also:
  - high-level language for managing the database
  - efficient search and access of large data sets



Advantages of using a database approach

- Controlling redundancy and inconsistency
- Access control
- Persistent storage
- Indexes and query processing
- Backup and recovery
- Multiple user interfaces
- Complex relationships
- Integrity constraints
- Active behaviour
- Enforcing standards, reducing application development time, flexibility to evolve system, up-to-date info

Data model?

- Every DB has a **data model** which makes it possible to "hide" the physical representation of data.
- A **data model** is a formalism that defines a *notation* for describing data on an abstract level together with a set of *operations* to manipulate data represented using this data model.
- Data models are used for *data abstraction* - making it possible to define and manipulate data on an abstract level.

Data model continued . . .

- E.g. assume that information about employees in an enterprise exists in a file `employees` which is a sequence records of the type:

```
record
    name: char[30];
    manager: char[30]
end
```

An abstract model of this file is a relation:

```
employees(name, manager)
```

, where `employees` is the name of the relation and `name` and `manager` is the attribute of the relation.

Data models - examples

- Examples of data models within the database field are:
  - Hierarchical (IMS)
  - Network (IDMS)
  - Relational (ORACLE, DB2, SQL Server, InterBase, Mimer)
  - Object-oriented (ObjectStore, Objectivity, Versant, Poet)
  - Object-relational (Informix, Odapter, DB2)
- Conceptual data model
  - ER-model (Entity-Relationship model)
    (not an implementation model since there are no operations defined for the notation)

Meta-data, i.e. “data about data”

- Information about which information that exists
- Information about how/where data is stored
  - file structures
  - records
  - data types / formats
  - name of files, data types
- Information regarding mapping between different schemas
- Meta-data is stored in the, so called, *system catalog* or *data dictionary*.

Meta-data cont. ...

- Meta-data is used by the DBMS to answer questions such as:

  (Users)

  – Which information exists in the database?
  – Is the information “x” in the database?
  – What is the cost to access a specific piece of information.

  (Database administrator or DBMS)

  – How much is different parts of the database used?
  – How long is the response time for different types of queries?
  – Has any user tried to break the security system of the database?
  – Is optimization required of the physical organization of the database with regards to memory utilization or response times?

Schema and instance

To be able to separate data in the database and its description the terms **database instance** and **database schema** are used.

- The schema is created when a database is defined. A database schema is not changed frequently.
- The data in the database constitute an instance. Every change of data creates a new instance of the database.

Data independence

- Reduces the connection between:
  - the actual organization of data and
  - how the users/application programs process data (or “sees” data.)
- Why?
  - Data should be able to change without requiring a corresponding alteration of the application programs.
  - Different applications/users need different “views” of the same data.

Data dependencies

Conventional systems have, in general, a very low level of data independence:

- Even a small change of the data structure, e.g. the introduction or reduction of a field in a record structure, usually require that one has to make changes in several programs or routines.

Programs can be dependent of that:

- data is located on a specific storage medium
- data has a specific storage format (binary, compressed)
- fields have been coded according to certain rules (Man = 1, Woman = 2)
- the file records are sorted in a specific manner

etc . . .

Data independence - how?

By introducing a multi-level architecture where each level represents one abstraction level.

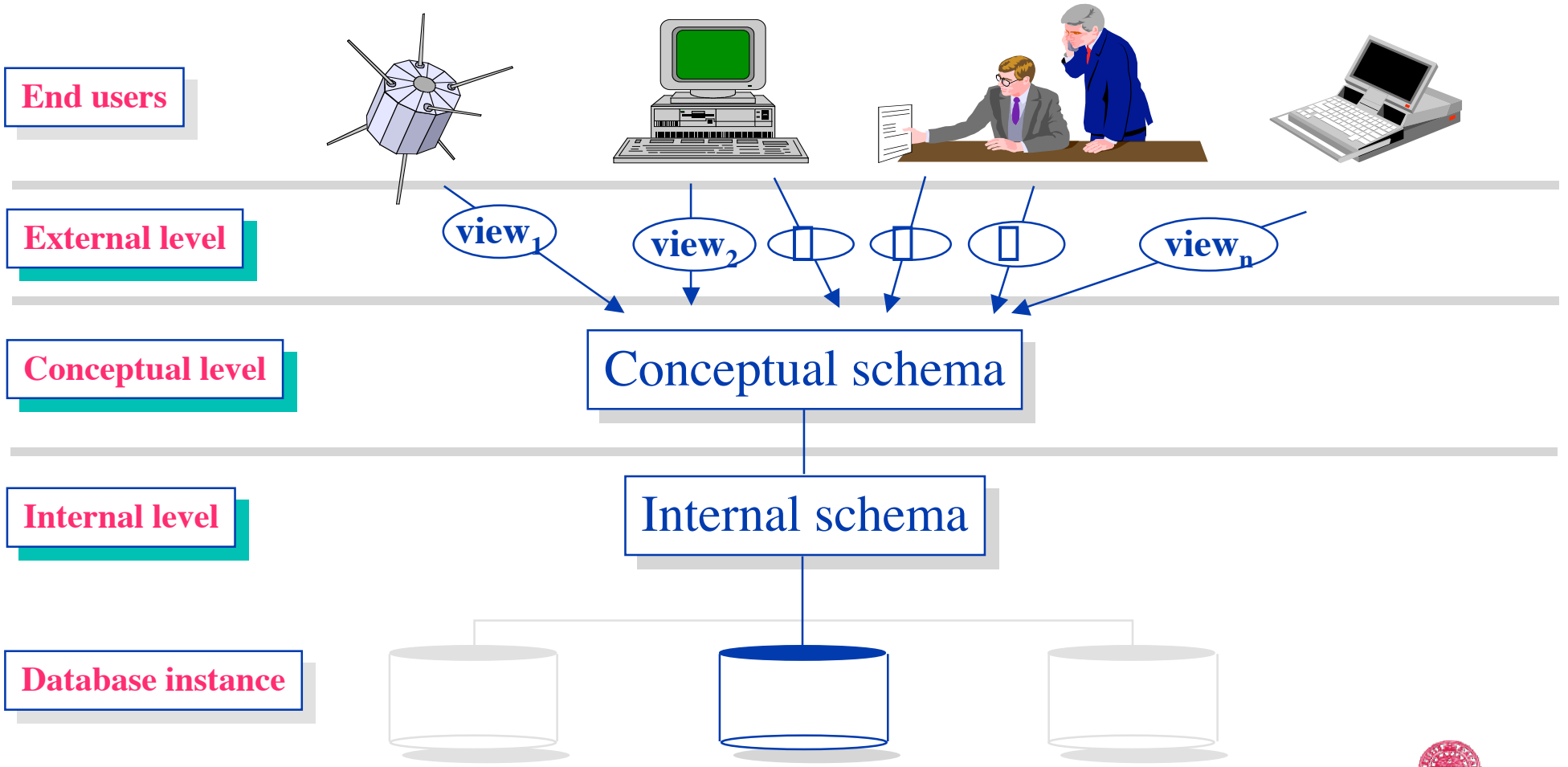
Three-schema architecture:

- In 1978 the following "standard" architecture (ANSI/SPARC architecture) for databases was introduced.

It consists of 3 levels:

1. Internal level
2. Conceptual level
3. External level
   - Each level introduces one abstraction layer and has a schema that describes how representations should be mapped to the next lower abstraction level.

Three-schema architecture



Internal schema

- Describes storage structures and access paths for the physical database.
  - Abstraction level: files, index files etc.
- Is usually defined through the data definition language (DDL) of the DBMS.

Conceptual schema

- An abstract description of the physical database.
- Constitute one, for all users, common basic model of the logical content of the database.
- This abstraction level corresponds to “the real world”: object, characteristics, relationships between objects etc.
- The schema is created in the DDL according to a specific data model.

External schemas, or views

- A typical DB has several users with varying needs, demands, access privileges etc.
- External schemas describes different views of the conceptual database with respect to what different user groups would like to/are allowed to se.
- Some DBMS's have a specific language for view definitions (else the DDL is used).

Views - example (Elmasri/Navathe fig 1.4)

(a)

TRANSCRIPT	StudentName	Student Transcript				
		CourseNumber	Grade	Semester	Year	SectionId
	Smith	CS1310	C	Fall	99	119
		MATH2410	B	Fall	99	112
	Brown	MATH2410	A	Fall	98	85
		CS1310	A	Fall	98	92
		CS3320	B	Spring	99	102
		CS3380	A	Fall	99	135

(b)

PREREQUISITES	CourseName	CourseNumber	Prerequisites
	Database	CS3380	CS3320
			MATH2410
	Data Structures	CS3320	CS1310

Views - example in SQL

- Assume that we have a relation (table) consisting information about employees in an enterprise:

```
employees(name,dept,salary,address)
```

- and wish to give a user group rights “to see” all information in the table except the SALARY field.
- This can be accomplished by the definition of a view called safe-emp:

```
create view safe-emps by
    select name, dept, address
    from employees;
```

Data independence in the three-schema architecture

1. Logical data independence
   - The possibility to change the conceptual schema without influencing the external schemas (views).
     - e.g. add another field to a conceptual schema.
2. Physical data independence
   - The possibility to change the internal schema without influencing the conceptual schema..
     - the effects of a physical reorganization of the database, such as adding an access path, is eliminated.

Database languages

- The term *database language* is a generic term for a class of languages used for defining, communicating with or manipulating a database.
- In conventional programming languages, declarations and program sentences is implemented in one and the same language.
- A DB system uses several different languages.
  - Storage Definition Language (SDL) - internal schema
  - Data Definition Language (DDL) - conceptual schema
  - View Definition Language (VDL) - external schema
  - Data Manipulation Language (DML)

DDL and DML

- DDL is used by the database administrator and others to define *internal* and *conceptual* schema.
- In this manner the database is designed. Subsequent modifications in the design is also made in DDL.
- DML is used by DB users and application programs to *retrieve*, *add*, *remove*, or *alter* the information in the database. The term *query language* is usually used as synonym to DML.

DDL example in SQL

```
create table
   flights(number: int,
           Date:char(6),
           Seats: int,
           from:char(3),
           to: char(3));
create index for flights on number;
```

- The first expression defines a relation, its attribute and their types.
- The second expression creates an index as part of the internal schema making search faster for flights, given a flight no. (e.g. this can be accomplished by creating a hash table with number as the key).

DML example in SQL

```
update flights
  set seats = seats -4
   where number = 123 and date = 'AUG 31'
```

"Decrease the no. of seats in flight no.. 123 on August 31 with 4."

```
insert into flights
   values(171,
      'AUG 21', 100, 'ROM', 'JFK')
```

"Add a new flight with flight no.. 171 and 100 seats, from Rom to New York (JFK) on August 21"

Classification criteria for DBMSs

- Type of data model
  - hierarchical, network, relational, object-oriented, object-relational
- Centralized vs. distributed DBMSs
  - Homogeneous vs. heterogeneous DDBMSs
  - Multidatabase systems
- Single-user vs. multi-user systems
- General-purpose vs. special-purpose DBMSs
  - specific applications such as airline reservation and phone directory systems.
- Cost

Components of a DBMS

- Query processor
  - DML compiler
  - Embedded DML precompiler
  - DDL interpreter
  - Query processing unit
- Storage manager
  - Authorization and integrity control
  - Transactions management
  - File management
  - Buffer management
- Physical storage
  - data files, meta-data (catalog), index, statistics

Comp. of a DBMS (fig 2.3 Elmasri/Navathe)