DATABASE TECHNOLOGY - 1MB025

(also 1DL300+1DL400)

Spring 2008

An introductury course on database systems

http://user.it.uu.se/~udbl/dbt-vt2008/

alt. http://www.it.uu.se/edu/course/homepage/dbastekn/vt08/

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

Personell

- Kjell Orsborn, lecturer, examiner
  - email: kjell.orsborn@it.uu.se, phone: 471 1154, room: 1321
- Tore Risch, lecturer
  - email : tore.risch@it.uu.se, phone : 471 6342, room: 1353
- Silvia Stefanova, course assistant
  - email: ruslan.fomkin@it.uu.se, phone: 471 2846, room 1319
- Sabesan Manivasakan, course assistant
  - email: manivasakan.sabesan@it.uu.se, phone: 471 7778, room 1339

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
- Transactions, Concurrency Control
- Recovery Techniques
- Storage and Index Structures
- Authorization and security
- OO/OR DBMSs
- AMOS/AMOSQL
- Query optimization
- Multimedia DBMSs
- Data warehousing

Preliminary course contents cont...

- Database assignments using Mimer SQL Engine
  - RDBMS
- Database assignment using AMOS II
  - OO/OR DBMS
- Small assignment project in AMOS II

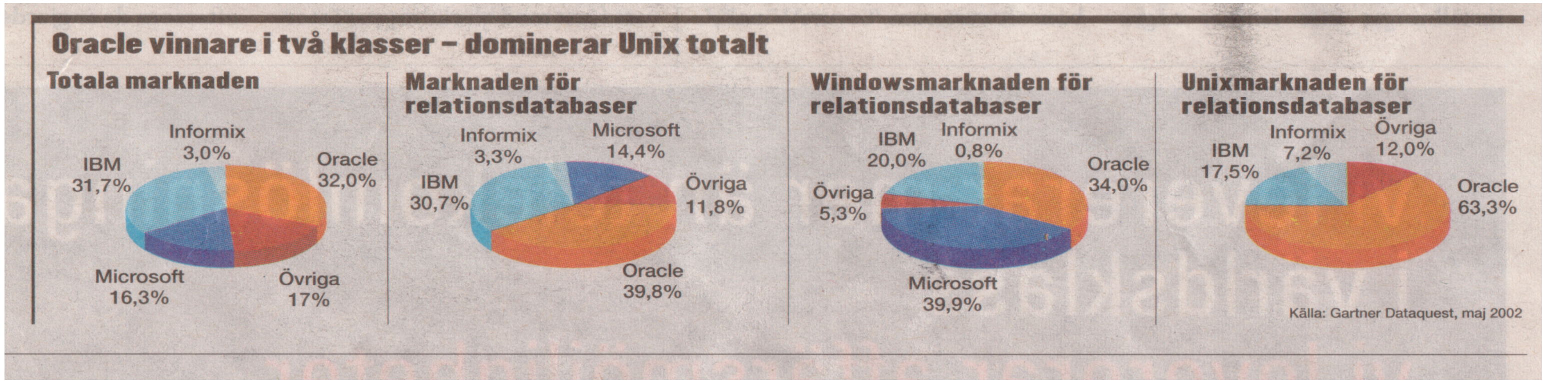
Introduction to Database Terminology

Elmasri/Navathe chs 1-2
Padron-McCarthy/Risch ch 1

Kjell Orsborn

Department of Information Technology
Uppsala University, Uppsala, Sweden

The database market /CS 020524



Oracle vinnare i två klasser – dominerar Unix totalt

Källa: Gartner Dataquest, maj 2002

ORACLE

Oracle9i Database

IBM

DB2 Universal Database

Informix Dynamic Server (IDS)

MySQL

Microsoft

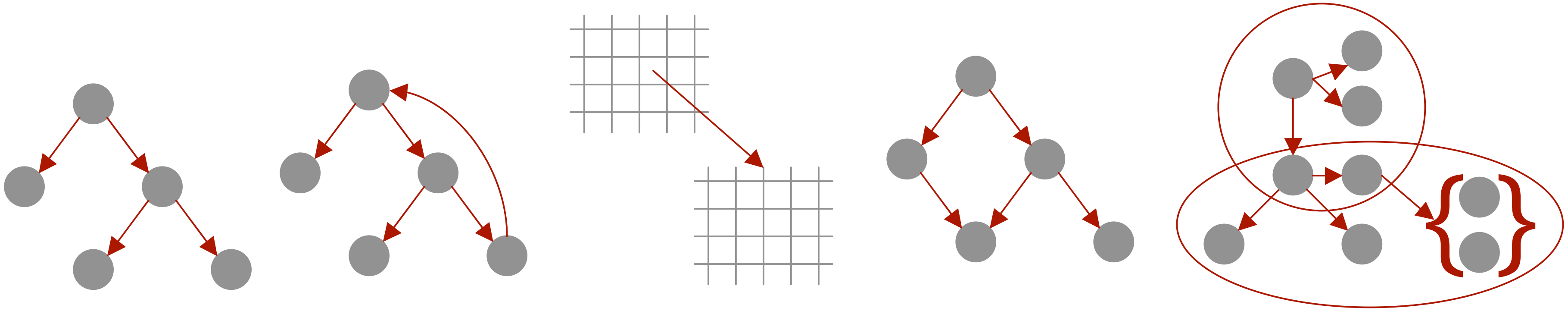
Microsoft SQL Server

Microsoft Access
The Office XP database solution

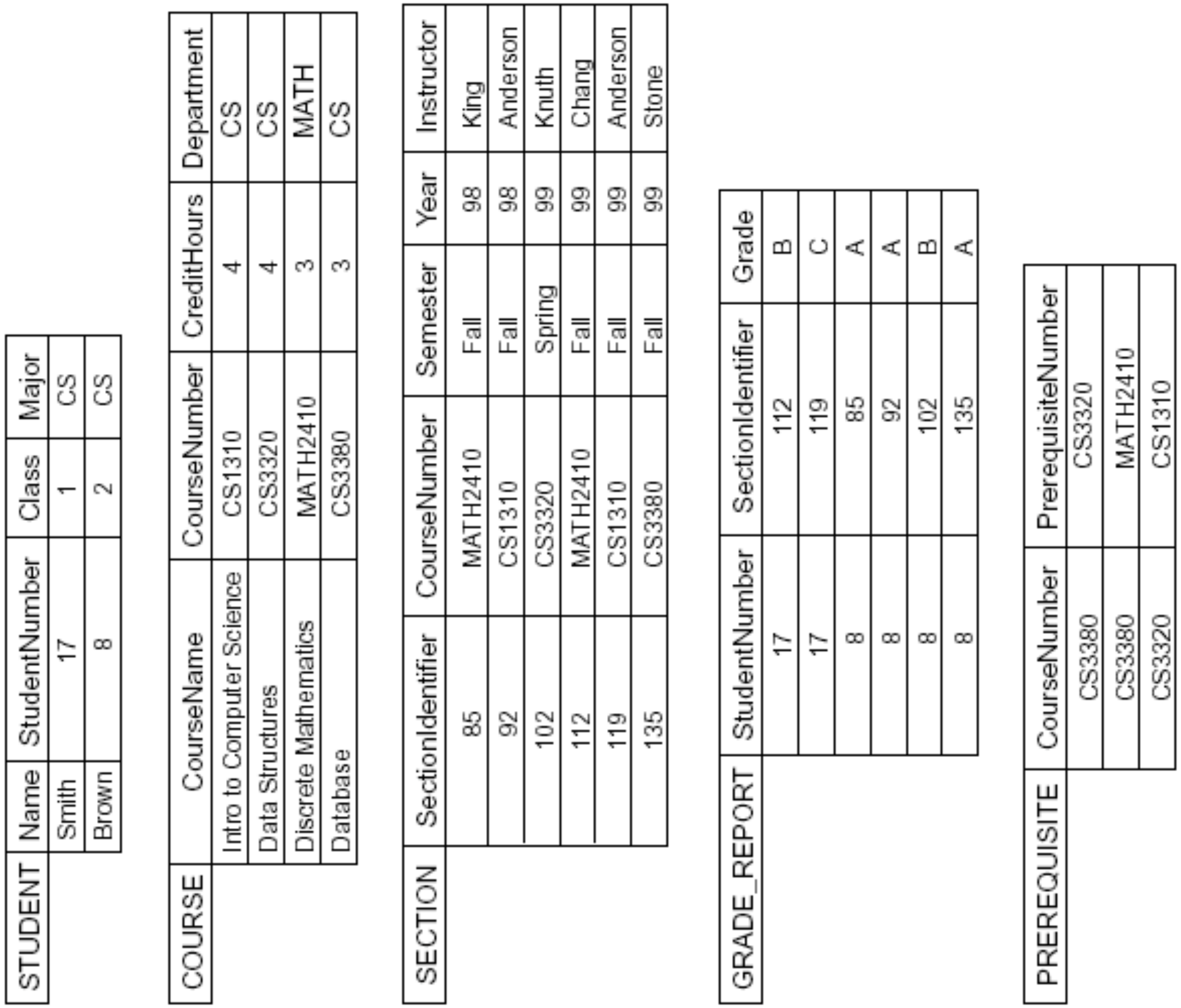
mimer SQL

Evolution of Database Technology

1960	1970	1980	1990	1997
Hierarchical	Network model	Relational model	Object-oriented DBMS	Object-relational DBMS
(IMS)	(CODASYL)	(e.g. ORACLE)	(e.g. ObjectStore)	(e.g. SQL:99)
Trees	Graph	Tables	OO data structures	Object model



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

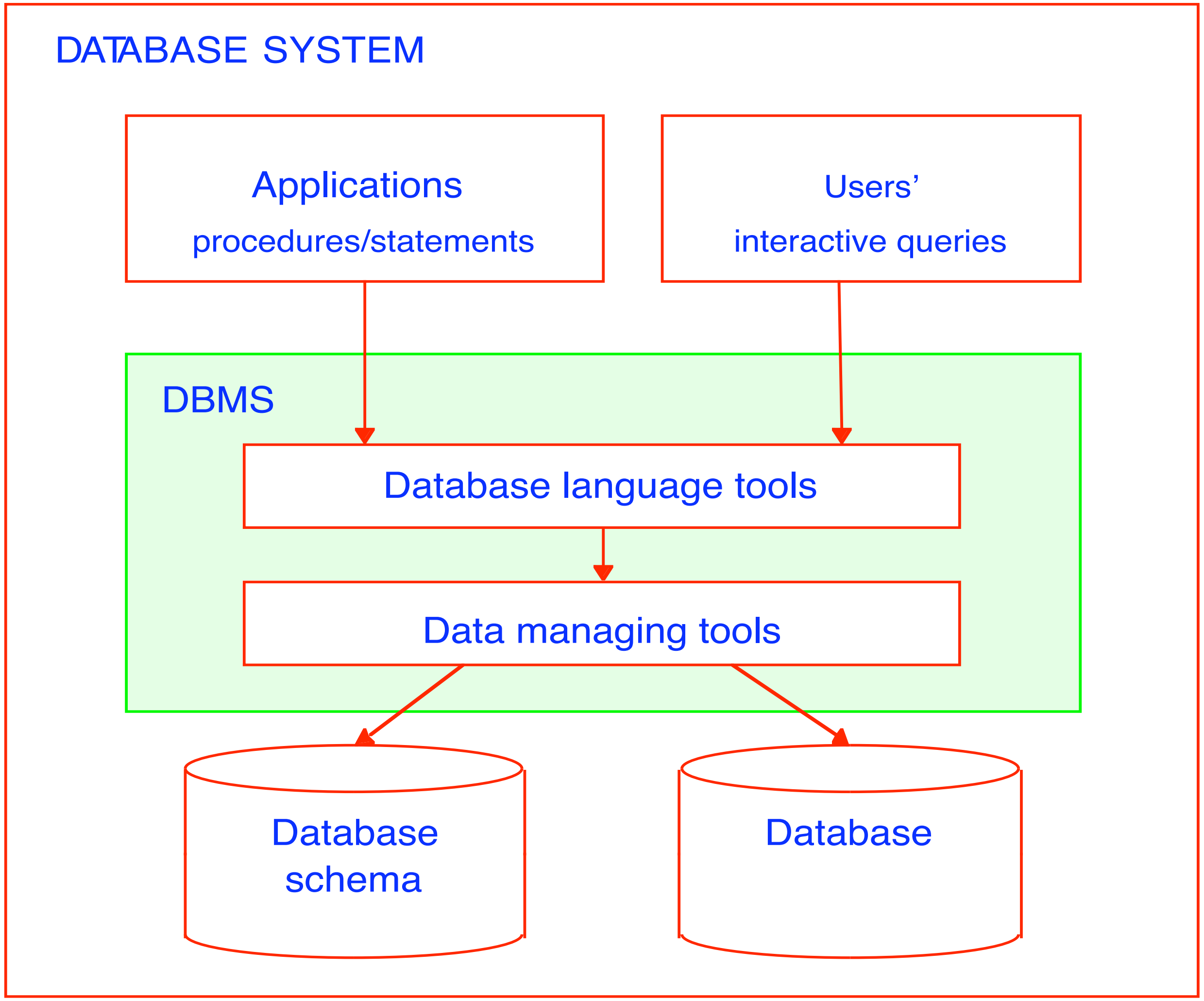
GRADE_REPORT

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

Outline of a database system



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

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 (constructing)*, *manipulating* and *sharing* databases for different types of applications.
- Also supports protection (system and security) and maintenance to evolve the system.

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
  - high-level language for managing data in the database

Advantages of using a database approach

- Efficient search and access of large data sets
- 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 models - examples

- Examples of representational (implementation) 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 and about how/where data is stored
  - names and data types of data items
  - names and sizes of files
  - storage details of each file
  - mapping information among schemas
  - constraints
- Meta-data is stored in the, so called, *system catalog* (or the more general term *data dictionary)*.

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

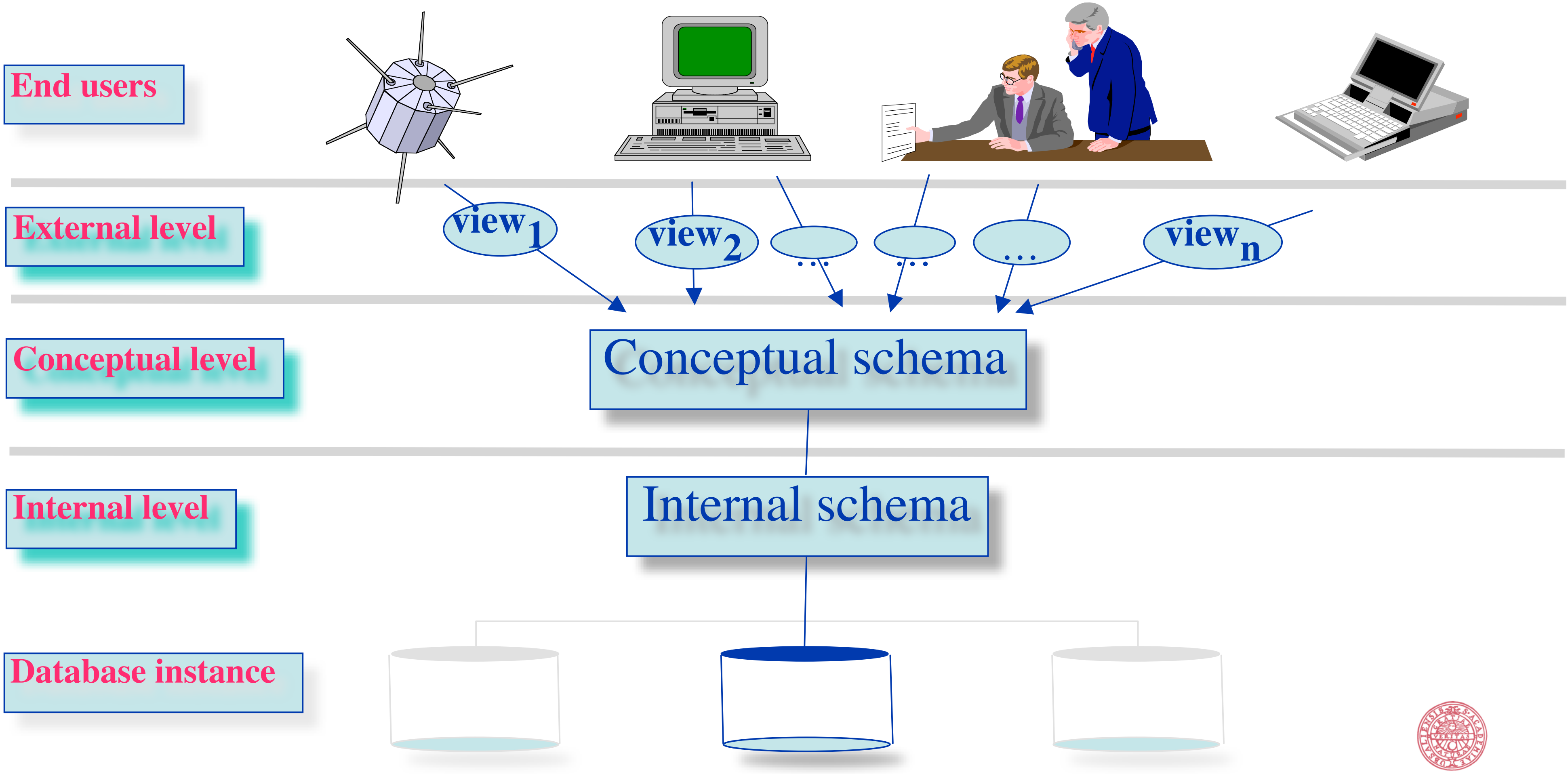
- Reduces the connection between:
  - the actual organization of data and
  - how the users/application programs process data (or “sees” data.)
- Why?
  - Data should be able to change without requiring a corresponding alteration of the application programs.
  - Different applications/users need different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

Data independence - how?

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

- The three-schema architecture:
  - In 1971 the “standard” three-schema architecture (also known as the ANSI/SPARC architecture) for databases was introduced by the CODASYL Data Base Task Group.
- It consists of 3 levels:
  - Internal level
  - Conceptual level
  - 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.

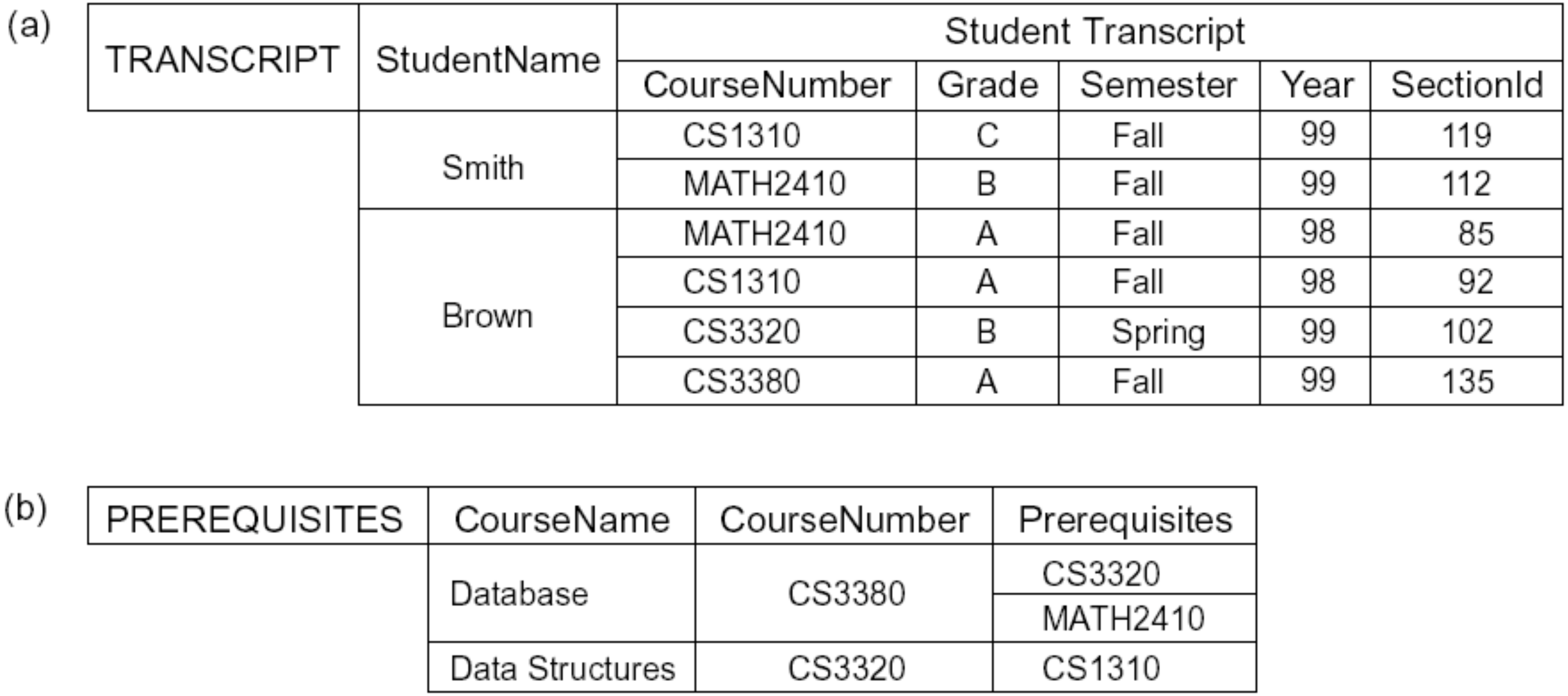
Three-schema architecture



Internal, conceptual and external schemas

- 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 schema (or views): a (restricted) view over the conceptual schema
  - A typical DB has several users with varying needs, demands, access privileges etc. and external schemas describes different views of the conceptual database with respect to what the 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

Possible 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.

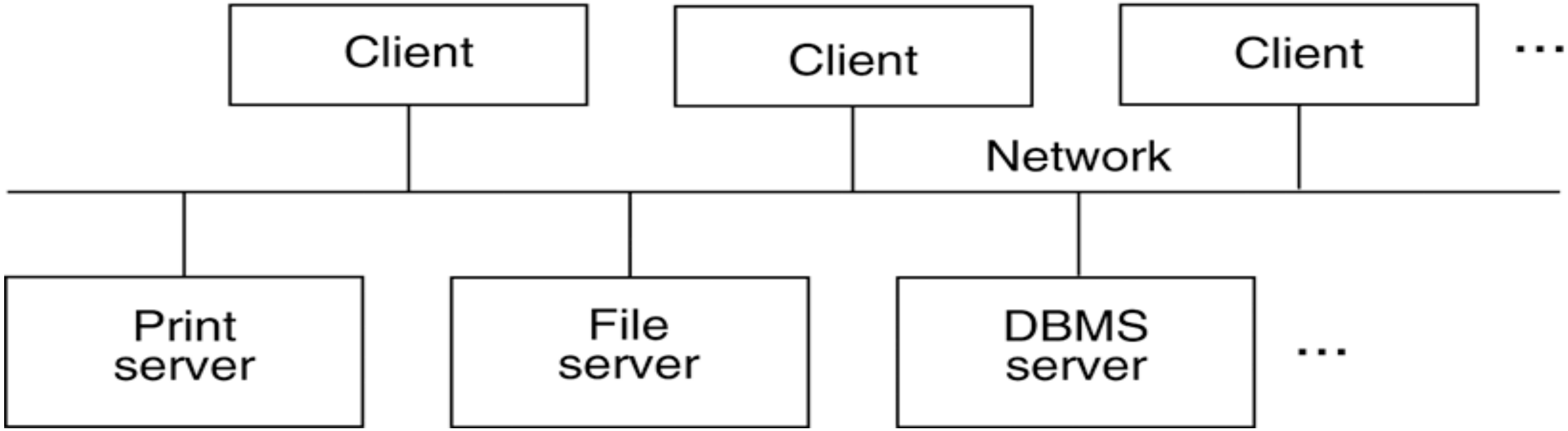
Database languages

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

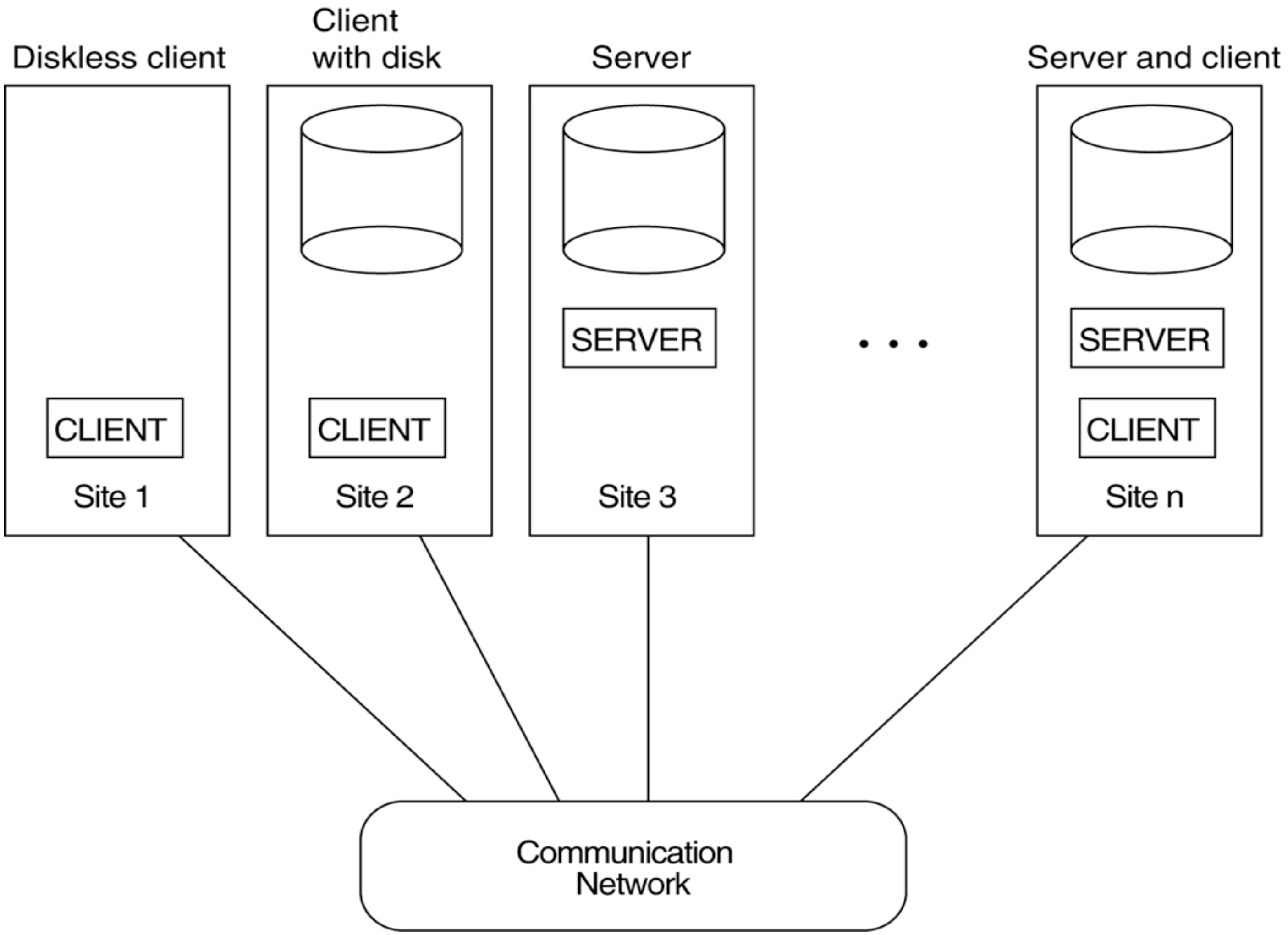
Classification criteria for DBMSs

- 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

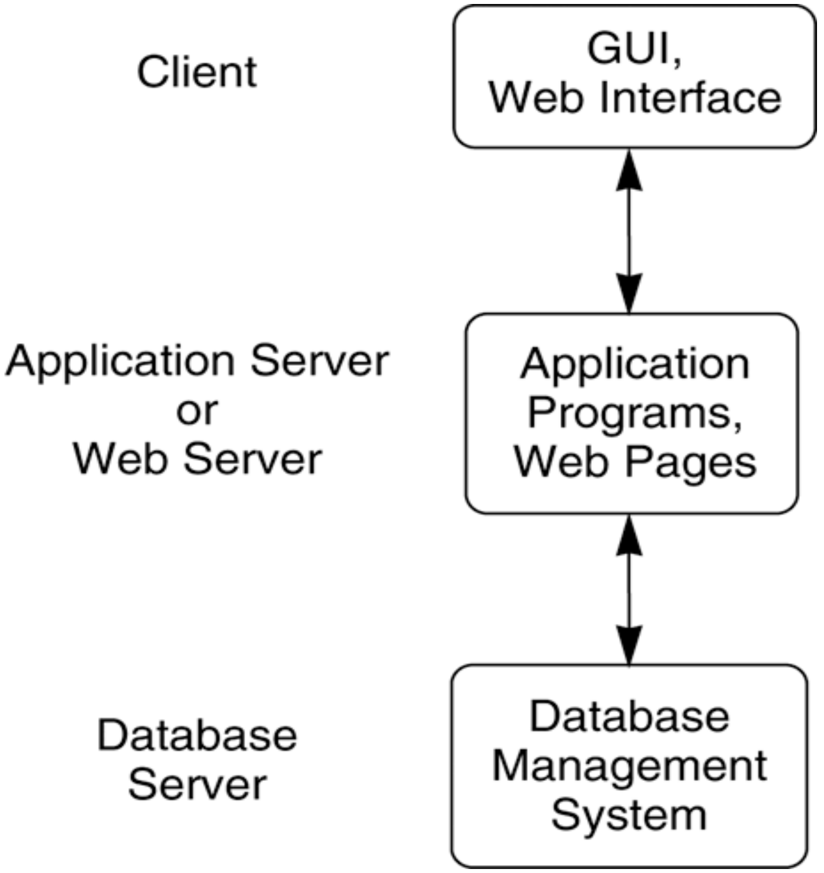
Logical two-tier client/server architecture.



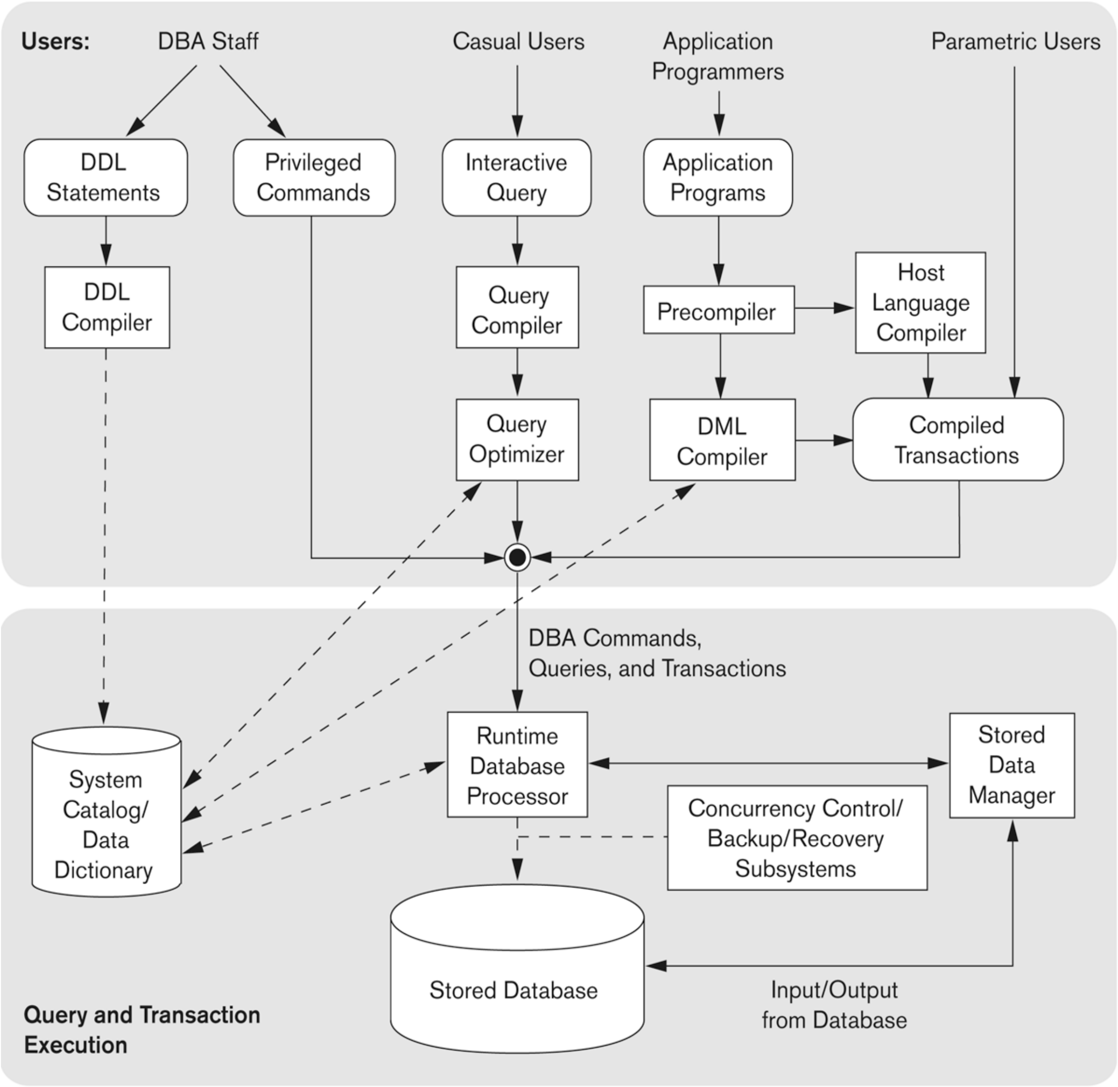
Physical two-tier client-server architecture



Logical three-tier client/server architecture


Client
GUI, Web Interface
Application Server or Web Server
Application Programs, Web Pages
Database Server
Database Management System

Components of a DBMS (fig 2.3 Elmasri/Navathe)


Users:
DBA Staff
Casual Users
Application Programmers
Parametric Users
DDL Statements
Privileged Commands
Interactive Query
Application Programs
DDL Compiler
Query Compiler
Precompiler
Host Language Compiler
Query Optimizer
DML Compiler
Compiled Transactions
DBA Commands, Queries, and Transactions
System Catalog/ Data Dictionary
Runtime Database Processor
Stored Data Manager
Concurrency Control/ Backup/Recovery Subsystems
Stored Database
Input/Output from Database
Query and Transaction Execution