DATABASE TECHNOLOGY - 1MB025

Fall 2005

An introductory course on database systems

http://user.it.uu.se/~udbl/dbt-ht2005/
alt. http://www.it.uu.se/edu/course/homepage/dbastekn/ht05/

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

• Milena Ivanova, course assistant
  – email: milena.ivanova@it.uu.se, phone: 471 2846, room 1319

• Johan Petrini, course assistant
  – email: johan.petrini@it.uu.se, phone: 471 6345, room 1316

• Erik Zeitler, course assistant
  – email: erik.zeitler@it.uu.se, phone: 471 3390, room 1310
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

Oracle9i Database

IBM

DB2 Universal Database

Microsoft Access

Informix Dynamic Server (IDS)

MySQL
Evolution of Database Technology

1960
Hierarchical
(IMS)
Trees

1970
Network model
(CODASYL)
Graph

1980
Relational model
(e.g. ORACLE)
Tables

1990
Object-oriented DBMS
(e.g. ObjectStore)
OO data structures

1997
Object-relational DBMS
(e.g. SQL:99)
Object model
An example database (Elmasri/Navathe fig. 1.2)

<table>
<thead>
<tr>
<th>STUDENT</th>
<th>COURSE</th>
<th>SECTION</th>
<th>GRADE REPORT</th>
<th>PREREQUISITE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>StudentNumber</td>
<td>CourseNumber</td>
<td>CreditHours</td>
<td>Department</td>
</tr>
<tr>
<td>Smith</td>
<td>17</td>
<td>CS1110</td>
<td>4</td>
<td>CS</td>
</tr>
<tr>
<td>Brown</td>
<td>8</td>
<td>CS3220</td>
<td>4</td>
<td>CS</td>
</tr>
<tr>
<td>CourseName</td>
<td>CourseNumber</td>
<td>CreditHours</td>
<td>Semester</td>
<td>Instructor</td>
</tr>
<tr>
<td>Intro to Computer Science</td>
<td>CS1110</td>
<td>4</td>
<td>Fall</td>
<td>King</td>
</tr>
<tr>
<td>Data Structures</td>
<td>CS3220</td>
<td>4</td>
<td>Fall</td>
<td>Anderson</td>
</tr>
<tr>
<td>Discrete Mathematics</td>
<td>MATH2110</td>
<td>3</td>
<td>Spring</td>
<td>Knuth</td>
</tr>
<tr>
<td>Database</td>
<td>CS3810</td>
<td>3</td>
<td>Fall</td>
<td>Chang</td>
</tr>
<tr>
<td>CourseNumber</td>
<td>SectionIdentifier</td>
<td>CourseNumber</td>
<td>Semester</td>
<td>Instructor</td>
</tr>
<tr>
<td>112</td>
<td>112</td>
<td>MATH2140</td>
<td>Fall</td>
<td>Anderson</td>
</tr>
<tr>
<td>119</td>
<td>119</td>
<td>CS3230</td>
<td>Fall</td>
<td>Stone</td>
</tr>
<tr>
<td>CourseNumber</td>
<td>SectionIdentifier</td>
<td>CourseNumber</td>
<td>Grade</td>
<td></td>
</tr>
<tr>
<td>CS3220</td>
<td>112</td>
<td>MATH2140</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>CS3810</td>
<td>119</td>
<td>CS3230</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>85</td>
<td>85</td>
<td>CS3220</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>92</td>
<td>92</td>
<td>MATH2140</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>102</td>
<td>102</td>
<td>CS3230</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>135</td>
<td>135</td>
<td>CS3810</td>
<td>C</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PreRequisiteNumber</th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS3320</td>
<td>B</td>
</tr>
<tr>
<td>MATH2110</td>
<td>B</td>
</tr>
<tr>
<td>CS1310</td>
<td>B</td>
</tr>
<tr>
<td>CS3390</td>
<td>C</td>
</tr>
<tr>
<td>CS3320</td>
<td>C</td>
</tr>
</tbody>
</table>
Outline of a database system

DATABASE SYSTEM

Applications
procedures/statements

Users’
interactive queries

DBMS

Database language tools

Data managing tools

Database schema

Database
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 (creating), manipulating and sharing databases for different types of applications.
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:
2. Internal level
3. Conceptual level
4. 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

End users

External level

Conceptual level

Internal level

Database instance

View 1

View 2

... 

View n

Conceptual schema

Internal schema

Database instance
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 see.
  – Some DBMS’s have a specific language for view definitions (else the DDL is used).
Views - example (Elmasri/Navathe fig 1.4)

(a) **TRANSCRIPT**

<table>
<thead>
<tr>
<th>StudentName</th>
<th>CourseNumber</th>
<th>Grade</th>
<th>Semester</th>
<th>Year</th>
<th>SectionId</th>
</tr>
</thead>
<tbody>
<tr>
<td>Smith</td>
<td>CS1310</td>
<td>C</td>
<td>Fall</td>
<td>99</td>
<td>119</td>
</tr>
<tr>
<td></td>
<td>MATH2410</td>
<td>B</td>
<td>Fall</td>
<td>99</td>
<td>112</td>
</tr>
<tr>
<td>Brown</td>
<td>MATH2410</td>
<td>A</td>
<td>Fall</td>
<td>98</td>
<td>85</td>
</tr>
<tr>
<td></td>
<td>CS1310</td>
<td>A</td>
<td>Fall</td>
<td>98</td>
<td>92</td>
</tr>
<tr>
<td></td>
<td>CS3320</td>
<td>B</td>
<td>Spring</td>
<td>99</td>
<td>102</td>
</tr>
<tr>
<td></td>
<td>CS3380</td>
<td>A</td>
<td>Fall</td>
<td>99</td>
<td>135</td>
</tr>
</tbody>
</table>

(b) **PREREQUISITES**

<table>
<thead>
<tr>
<th>CourseName</th>
<th>CourseNumber</th>
<th>Prerequisites</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database</td>
<td>CS3380</td>
<td>CS3320</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MATH2410</td>
</tr>
<tr>
<td>Data Structures</td>
<td>CS3320</td>
<td>CS1310</td>
</tr>
</tbody>
</table>
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 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.
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 (fig 2.3 Elmasri/Navathe)