DATABASTEKNIK - 1DL116

Spring 2004

An introductury course on database systems

http://user.it.uu.se/~udbl/dbt-vt2004/

Kjell Orsborn
Uppsala Database Laboratory
Department of Information Technology, Uppsala University,
Uppsala, Sweden
Personell - Spring 2004

- Kjell Orsborn, kursansvarig lämare
  - 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
- Ruslan Fomkin, kursassistent
  - epost: ruslan.fomkin@it.uu.se
  - tel: 471 1062
  - rum 1341
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
- Security / Authorization
  Advanced SQL
- OO/OR DBMSs
- AMOS/AMOSQL Query optimization
- Active DBMSs
- Multimedia DBMSs
- Data warehousing / Data Mining
- Relational calculus, QBE
Preliminary course contents

• Labs using Mimer SQL Engine
  – RDBMS
• Labs using AMOS II
  – OO/OR DBMS
• Small lab project in AMOS II
Introduction to Database Technology

Lecture 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

DB2 Universal Database

Informix Dynamic Server (IDS)

MySQL

Microsoft Access

The Office XP database solution
Evolution of Database Technology

1960
Hierarchical
(IMS)
Trees

1970
Network model
(CODASYL)
Complex data structures

1980
Relational model
(e.g. ORACLE)
Tables

1990
1st Generation OODB
(e.g. Objectivity)
OO data structures

1997
Object Realntional DBMS
(e.g. SQL99)
Object model
Introduction to Database Terminology

Elmasri/Navathe chs 1-2

Kjell Orsborn

Department of Information Technology
Uppsala University, Uppsala, Sweden
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
An example database (Elmasri/Navathe fig. 1.2)

<table>
<thead>
<tr>
<th>Student</th>
<th>StudentNumber</th>
<th>Class</th>
<th>Major</th>
<th>Course</th>
<th>CourseNumber</th>
<th>CreditHours</th>
<th>Department</th>
<th>Instructor</th>
<th>Year</th>
<th>Semester</th>
<th>CourseNumber</th>
<th>SectionIdentifier</th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>Smith</td>
<td>17</td>
<td>1</td>
<td>CS</td>
<td>Intro to Computer Science</td>
<td>CS1310</td>
<td>4</td>
<td>CS</td>
<td>King</td>
<td>99</td>
<td>Fall</td>
<td>MATH2410</td>
<td>112</td>
<td>B</td>
</tr>
<tr>
<td>Brown</td>
<td>18</td>
<td>2</td>
<td>CS</td>
<td>Data Structures</td>
<td>CS3220</td>
<td>4</td>
<td>CS</td>
<td>Anderson</td>
<td>99</td>
<td>Fall</td>
<td>CS3130</td>
<td>102</td>
<td>B</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Discrete Mathematics</td>
<td>MATH2110</td>
<td>3</td>
<td></td>
<td>Knuth</td>
<td>99</td>
<td>Spring</td>
<td>CS3130</td>
<td>119</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Database</td>
<td>CS3380</td>
<td>3</td>
<td></td>
<td>Chang</td>
<td>99</td>
<td>Fall</td>
<td>CS3380</td>
<td>135</td>
<td>A</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Course</th>
<th>CourseNumber</th>
<th>SectionIdentifier</th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>CS1310</td>
<td>112</td>
<td>B</td>
</tr>
<tr>
<td>18</td>
<td>CS3220</td>
<td>119</td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>MATH2110</td>
<td>99</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>CS3380</td>
<td>135</td>
<td>A</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PrerequisiteCourse</th>
<th>PrerequisiteNumber</th>
<th>CS1310</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CS3220</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MATH2110</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CS3380</td>
<td></td>
</tr>
</tbody>
</table>

Kjell Orsborn 3/23/04
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:

```plaintext
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 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 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.

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:

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

- **End users**
  - External level
  - Conceptual level
  - Internal level
- **Database instance**
  - Internal schema
  - Conceptual schema
  - Views: $\text{view}_1$, $\text{view}_2$, $\text{view}_n$
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 see.
- Some DBMS’s have a specific language for view definitions (else the DDL is used).
Views - example (Elmasri/Navathe fig 1.4)

<table>
<thead>
<tr>
<th>Student Name</th>
<th>Course Number</th>
<th>Grade</th>
<th>Semester</th>
<th>Year</th>
<th>Section Id</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>

<table>
<thead>
<tr>
<th>Course Name</th>
<th>Course Number</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>
Views - example in SQL

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

```sql
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:

```sql
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

```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)