Database Management Systems 2010/11
– Chapter 1: Introduction –

J. Gamper

- The Course
- The DB Field
- Basic Definitions
- DB Functionality and Characteristics
- History of DB Technology
- The Relational Data Model
- Accessing DBs

These slides were developed by:
– Michael Böhlen, University of Zurich, Switzerland
– Johann Gamper, University of Bozen-Bolzano, Italy

The Course

- Course page
  - http://www.inf.unibz.it/dis/teaching/DMS
  - Here you will find all important information, including lecture notes, exercises, schedule, rules for the exam, old exams, etc.
  - Slides and exercises will be uploaded 1 day before the lecture

- The slides are based on the following text books and associated material:
  - A. Silberschatz, H. Korth, and S. Sudarshan: Database System Concept, 5 edition, McGraw Hill, 2006.
  - R. Elmasri and S.B. Navathe: Fundamentals of Database Systems, 4th edition, Pearson Addison Wesley, 2004.
- Additional Book
  - Garcia-Molina, Ullman,Widom: Database Systems: The Complete Book, Prentice-Hall, 2002.

- What is important?
  - Understand the key concepts of database management systems.
  - Be able to apply your knowledge on relevant examples.
  - Doing the exercises is very important. It is the best preparation for the exam.

The Course Content

- Storage and File Structure
  - Physical Storage media, file and buffer manager
- Indexing and Hashing
  - Ordered indices, B-trees, hashing
- Query Processing
  - Measures of query cost, selection and join operation
- Query Optimization
  - Transformation of relational expressions, evaluation plans
- Transactions
  - ACID properties, SQL transactions
- Concurrency Control
  - Lock-/timestamp-/validation-based protocols
- Recovery System
  - Log-based recovery, shadow paging

The DB Field/1

- Journal Publications
  - ACM Transaction on Database System (TODS)
  - The VLDB Journal (VLDBJ)
  - IEEE Transactions on Knowledge and Data Engineering (TKDE)
  - Information Systems (IS)
- Conference Publications
  - SIGMOD
  - VLDB
  - ICDE
  - EDBT
- DB & LP Bibliography (Michael Ley, Uni Trier, Germany)
  - http://www.informatik.uni-trier.de/~ley/db/
- DBWorld mailing list
  - http://www.cs.wisc.edu/dbworld/

The DB Field/2

dblp.uni-trier.de
Computer Science Bibliography

The DBLP Computer Science Bibliography

maintained by Michael Ley - Welcome - FAQ

DBLP is available from several hosts: Trier I - Trier II - ACM SIGMOD - SunSITE CE

Search

- Author
- Faceted search (L3S Research Center, U. Hannover)
- CompleteSearch (Holger Bast, Max Planck Institut f. Inf.)

Bibliographies

- **Conferences**: SIGMOD, VLDB, PODS, ER, EDBT, ICDE, POPL, ...
- **Journals**: CACM, TODS, TOIS, TOPLAS, DKE, VLDB J., Inf. Systems, TPLP, TCS, ...
- **Series**: LNCS/LNAI, IFIP
- **Books**: Reference - Collections - DB Textbooks
- **By Subject**: Database Systems, Logic Prog., IR, ...

Full Text: ACM SIGMOD Anthology

The DB Field/3

- Commercial Products
  - Oracle
  - DB2 (IBM)
  - Microsoft SQL Server
  - Sybase
  - Ingres
  - Informix
  - PC "DBMSs": Paradox, Access, ...
  - ...
- Open Source Products
  - PostgreSQL
  - MySQL
  - MonetDB
  - ...

Typical Activities (aka Jobs) of Database People

- Data modeling
- Handling large volumes of complex data
- Distributed databases
- Design of migration strategies
- User interface design
- Development of algorithms
- Design of languages
- New data models and systems
  - XML/semi-structured databases
  - Stream data processing
  - Temporal and spatial databases
  - GIS systems
- etc.

Basic Definitions/1

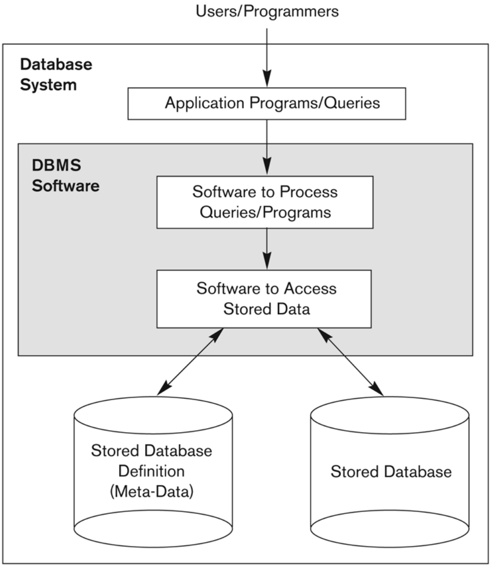
About, data, information, and knowledge:

- **Data** are facts that can be recorded:
  - book(Lord of the Rings, 3, 10)
- **Information** = data + meaning
  - book:
  - title = Lord of the rings,
  - volume nr = 3,
  - price in USD = 10
- **Knowledge** = information + application

Basic Definitions/2

- **Mini-world**: The part of the real world we are interested in
- **Data**: Known facts about the mini-world that can be recorded
- **Database (DB)**: A collection of related data
- **Database Management System (DBMS)**: A software package to facilitate the creation/maintenance of databases
- **Database System**: DB + DBMS
- **Meta Data**: Information about the structure of the DB.
  - Meta data is organized as a DB itself.

Basic Definitions/3



Basic Definitions/4

- A DBMS provides two kind of languages
  - A **data definition language** (DDL) for specifying the database schema
    - the database schema is stored in the data dictionary
    - the content of data dictionary is called metadata
  - A **data manipulation language** (DML) for updating and querying databases, i.e.,
    - retrieval of information
    - insertion of new information
    - deletion of information
    - modification of information
- The standard language for database systems is SQL
  - "Intergalactic data speak" [Michael Stonebraker]
- SQL offers a DDL and DML

Database Applications

- Traditional Applications
  - Numeric and Textual Databases
- More Recent Applications:
  - Multimedia Databases
  - Geographic Information Systems (GIS)
  - Data Warehouses
  - Real-time and Active Databases
  - Many other applications
- Examples:
  - Bank (accounts)
  - Stores (inventory, sales)
  - Reservation systems
  - University (students, courses, rooms)
  - online sales (amazon.com)
  - online newspapers (nzz.ch)

Typical DBMS Functionality/1

- **Define** a particular database in terms of its data types, structures, and constraints
- **Construct** or **load** the initial database contents on a secondary storage medium
- **Manipulating** the database:
  - Retrieval: Querying, generating reports
  - Modification: Insertions, deletions and updates to its content
  - Accessing the database through Web applications
- **Sharing** by a set of concurrent users and application programs while, at the same time, keeping all data valid and consistent

Typical DBMS Functionality/2

- Other features of DBMSs:
  - Protection or Security measures to prevent unauthorized access
  - Active processing to take internal actions on data
  - Presentation and Visualization of data
  - Maintaining the database and associated programs over the lifetime of the database application (called database, software, and system maintenance)

Main Characteristics of Database Approach/1

- **Self-describing** nature of a database system:
  - A DBMS catalog stores the description of a particular database (e.g. data types, data structures, and constraints)
  - The description is called **metadata**.
  - This allows the DBMS software to work with different database applications.
- Insulation between programs and data:
  - Called **data independence**.
  - Allows changing data structures and storage organization without having to change the DBMS access programs.

Main Characteristics of Database Approach/2

Example of a DBMS catalog (just the idea; oversimplified):

RELATIONS

RelationName	NrOfColumns
STUDENT	4
COURSE	4
SECTION	5
GRADE_REPORT	3
PRERQUISITE	2

COLUMNS

ColumnName	DataType	BelongsToRelation
Name	Character(30)	STUDENT
StudentNr	CHARACTER(4)	STUDENT
Class	INTEGER(1)	STUDENT
...	...	...

- PostgreSQL 8.3.9: 74 objects in the system catalog
- Oracle 10.2: 1821 objects in the system catalog

Main Characteristics of Database Approach/3

- **Data Abstraction**:
  - A data model is used to hide storage details and present the users with a conceptual view of the database.
  - Programs refer to the data model constructs rather than data storage details
- Support of **multiple views** of the data:
  - Each user may see a different view of the database, which describes only the data of interest to that user.

Main Characteristics of Database Approach/4

- **Sharing** of data and **multi-user** transaction processing:
  - Allowing a set of concurrent users to retrieve from and to update the database.
  - Concurrency control within the DBMS guarantees that each transaction is correctly executed or aborted
  - Recovery subsystem ensures each completed transaction has its effect permanently recorded in the database
  - OLTP (Online Transaction Processing) is a major part of database applications. This allows hundreds of concurrent transactions to execute per second.

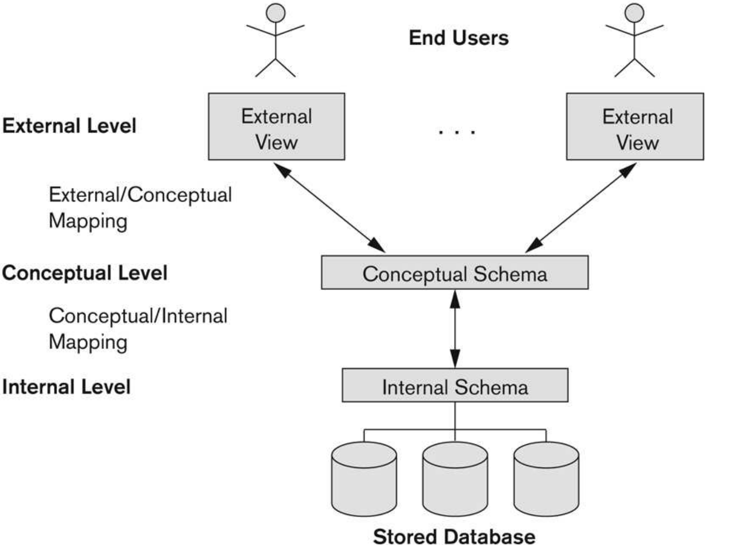
The ANSI/SPARC Three Schema Architecture/1

- Proposed to support DBMS characteristics of:
  - Data independence
  - Multiple views of the data
- Not explicitly used in commercial DBMS products, but has been useful in explaining database system organization
- Defines DBMS schemas at three levels:
  - **Internal schema** at the internal level to describe physical storage structures and access paths (e.g indexes).
    - Typically uses a physical data model.
  - **Conceptual schema** at the conceptual level to describe the structure and constraints for the whole database for a community of users.
    - Uses a conceptual or an implementation data model.
  - **External schemas** at the external level to describe the various user views.
    - Usually uses the same data model as the conceptual schema.

The ANSI/SPARC Three Schema Architecture/2

- Mappings among schema levels are needed to transform requests and data.
  - Programs refer to an external schema, and are mapped by the DBMS to the internal schema for execution.
  - Data extracted from the internal DBMS level is reformatted to match the user's external view (e.g., formatting the results of an SQL query for display in a Web page)

The ANSI/SPARC Three Schema Architecture/3



Databases – Pros and Cons

- **Pros**
  - Logical
  - Data abstraction
  - Meta reasoning
  - Self describing, e.g., data dictionary
  - Multiple user views
  - Data sharing
- **Cons**
  - Huge and complex systems
  - Restrict functionality
  - Substantial overhead
  - No direct data access

- When **not to use** a DBMS
  - Too high costs
    - High intitial investment (software, hardware, training)
    - Overhead for providing generality, security, recovery, integrity, and concurrency
  - Simple, well defined, and not-changing application
  - No multi-user access required
  - Stringent real-time requirements

History of Database Technology/1

- Early database applications:
  - The hierarchical model and the network model were introduced in mid 1960s and dominated during the seventies.
  - A bulk of the worldwide database processing still occurs using these models, particularly, the hierarchical model.
- Systems based on the relational model:
  - The relational model was originally introduced in 1970
  - The relational model was heavily researched and experimented within IBM Research and several universities
  - Relational DBMS Products emerged in the early 1980s.

History of Database Technology/2

- Object-oriented and emerging applications:
  - Object-oriented database management systems (OODBMSs) were introduced in late 1980s and early 1990s to cater to the need of complex data processing in CAD and other applications.
  - Pure OODBMSs have disappeared. Many relational DBMSs have incorporated object database concepts, leading to a new category called object-relational DBMSs (ORDBMSs).
  - Extended relational systems add further capabilities (e.g. for multimedia data, XML, and other data types)
- Data on the web and E-commerce applications:
  - Web contains data in HTML with links among pages.
  - This has given rise to a new set of applications and E-commerce is using new standards like XML.
  - Script programming languages such as PHP and JavaScript allow generation of dynamic Web pages that are partially generated from a database.

History of Database Technology/3

- New functionality is being added to DBMSs in the following areas:
  - Scientific Applications
  - XML (eXtensible Markup Language)
  - Image Storage and Management
  - Audio and Video Data Management
  - Data Warehousing and Data Mining
  - Spatial Data Management
  - Time Series and Historical Data Management
- The above gives rise to new research and development in incorporating new data types, complex data structures, new operations and storage and indexing schemes in database systems.

Relational Data Model/1

- Data are stored in relations/tables

employee

Name	Dept	Salary
Tom	SE	23K
Lena	DB	33K

department

Dname	Manager	Address
SE	Tom	Boston
DB	Lena	Tucson

project

Pid	Dept	From	To
14	SE	01.01.2005	31.12.2005
173	SE	15.04.2005	30.10.2006
201	DB	15.04.2005	31.03.2006

Relational Data Model/2

- A **domain** D is a set of atomic data values.
  - phone numbers,CPR numbers, names, grades, birthdates, departments, {i,o,x,?,-}
  - each domain includes the special value `null` for unknown or missing value
- With each domain a **data type** or format is specified.
  - 5 digit integers, yyyy-mm-dd, characters
- An **attribute** A_i describes the role of a domain in a relation schema.
  - PhoneNr, Age, DeptName

Relational Data Model/3

- A **relation schema** $R(A_1, ..., A_n)$ is made up of a relation name R and a list of attributes.
  - $employee(Name, Dept, Salary)$
- A **tuple** t is an ordered list of values, i.e., $t = (v_1, ..., v_n)$ with $v_i \in dom(A_i)$.
  - $t = (Tom, SE, 23K)$
- A **relation** r of the relation schema $R(A_1, ..., A_n)$ is a set of n-ary tuples.
  - $r = \{(Tom, SE, 23K), (Lene, DB, 33K)\}$
- A **database** DB is a set of relations.
  - $DB = \{r, s, ...\}$
  - $r = \{(Tom, SE, 23K), (Lene, DB, 33K)\}$
  - $s = \{(SE, Tom, Boston), (DB, Lena, Tucson)\}$

Relational Data Model/4

Properties of relations

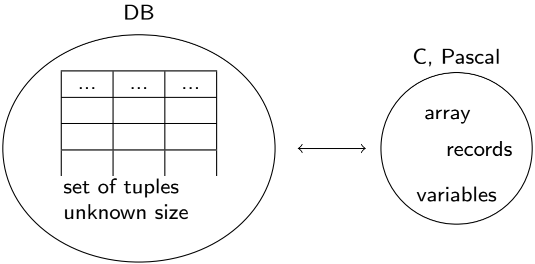
- A relation is a **set of tuples**, i.e.,
  - **no ordering** between tuples and
  - **no duplicates** (identical tuples) exist.
- Attributes within tuples are **ordered**
  - At the logical level it is possible to have unordered tuples if the correspondence between values and attributes is maintained
  - e.g., $\{Salary/23K, Name/Tom, Dept/SE\}$

Accessing DBs

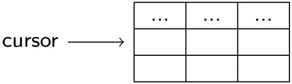
- The success of DBs also depends on the ease of data access.
- When accessing a (relational) DB two factors must be taken into account.
  - The impedence mismatch.
  - The interface to the DB.

Impedence Mismatch and Cursor

- The **impedence mismatch** refers to the difference between the data models of the DBMS and the programming (host) language (e.g., sets vs. records)



- **Cursor**: The most versatile way to access a DB.
  - Cursors are used to resolve the impedance mismatch.
  - A cursor runs through the tuples of a relation/table.



DB Interfaces

- Various interfaces to DBs exist, e.g.,
  - Terminal interface (sqlplus, etc.)
  - OCI (Oracle Call Interface)
  - X/Open SQL CLI (Call Level Interface)
  - ODBC (Open Data Base Connection), iODBC for Unix
  - JDBC (Java Database Connectivity)
  - DBI (Perl DB Interface)
  - Embedded SQL

Oracle's OCI/1

- The OCI is a set of C procedures to access an Oracle database e.g.,
  - `olon`
  - `oparse`
  - `oexec`
  - `ologof`
  - `odescr`
  - `ofetch`
  - `oopen`
  - `odefin`
  - `oclose`
  - `obndrn`

Oracle's OCI/2

```
#include <ocidfn.h>
Lda_Def lda;
Cda_Def cda;
main() {
  orlon(&lda,hda,"scott",-1,"tiger",-1,0);
  oopen(&cda,&lda,0,-1,-1,0,-1);
  oparse(&cda,"SELECT * FROM cat",-1,0,2);
  odefin(&cda,1,&name,30,,-1,0,0,-1,-1,0,0);
  odefin(&cda,2,&type,30,,-1,0,0,-1,-1,0,0);

  oexec(&cda);
  for (;;) {
    if (ofetch(&cda1)) break;
      printf(" %s %s ", name, type);
  }

  oclose(&cda);
  ologof(&lda);
}
```

ODBC/1

- ODBC is a set of C procedures to access any(!) SQL database, e.g.,
  - `SQLAllocEnv`
  - `SQLAllocStmt`
  - `SQLDescribeCol`
  - `SQLAllocConnect`
  - `SQLPrepare`
  - `SQLBindCol`
  - `SQLConnect`
  - `SQLSetParam`
  - `SQLFetch`
  - `SQLDisconnect`
  - `SQLExecute`
  - `SQLFreeConnect`
  - `SQLExecDirect`
  - `SQLFreeEnv`
  - `SQLFreeStmt`
- ODBC supports meta data.

ODBC/2

```
#include <sqlcli_.h>
SQLHENV e;
SQLHDBC c;
SQLHSTMT s;

int main() {
  SQLAllocEnv(&e);
  SQLAllocConnect(e,&c);
  SQLConnect(c, "ora1", SQL_NTS, "scott", SQL_NTS, "tiger", SQL_NTS);
  SQLAllocStmt(c,&s);
  SQLPrepare(s,"select * from cat", SQL_NTS);
  SQLExecute(s);
  SQLBindCol(s,1,SQL_C_CHAR,name,30,NULL);
  SQLBindCol(s,2,SQL_C_CHAR,type,30,NULL);

  SQLFetch(s);
  printf("%s %s", name, type);

  SQLFreeStmt(s,SQL_DROP);
  SQLDisconnect(c);
  SQLFreeConnect(c);
  SQLFreeEnv(e);
}
```

JDBC Interface/1

- JDBC is a set of Java procedures to access any(!) SQL database, e.g.,
  - `getConnection`
  - `execute`
  - `getColumnName`
  - `createStatement`
  - `exectueQuery`
  - `getColumnType`
  - `close`
  - `executeUpdate`
  - `getString`
  - `getResultSet`
  - `getObject`
- JDBC supports meta data.

JDBC Interface/2

```
import java.sql.*;
class demo {

  public static void main (String args [])
      throws SQLException,ClassNotFoundException {

    // Load the Oracle JDBC driver
    Class.forName("oracle.jdbc.driver.OracleDriver");

    // Connect to the database
    Connection conn = DriverManager.getConnection(
      "jdbc:oracle:thin:@femto:1526:ora1", "scott", "tiger");

    // Create a statement
    Statement stmt = conn.createStatement ();
    // Insert a tuple into a relation
    stmt.execute("insert into r values(1,'abc')");
    // Executes a query and displays the result
    ResultSet rset = stmt.executeQuery ("select * from r");
    while (rset.next())
      System.out.println(rset.getInt(1) + " " + rset.getString(2));
  }
}
```

Embedded SQL

- Extended versions of, e.g., C, Pascal, and Fortran allow to embed SQL statements.
- A precompiler compiles these languages to, e.g., C with OCI library calls.
- The idea is that C with embedded SQL is easier to use than C with OCI calls.
- Embedded SQL is standardized (ISO, ANSI).