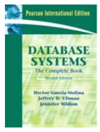
Databases TDA357/DIT620

Graham Kemp
kemp@chalmers.se
Room 6475, EDIT Building

Course Book

"Database Systems:
The Complete Book, 2E",
by Hector Garcia-Molina,
Jeffrey D. Ullman,
and Jennifer Widom



Approx. chapters 1-12

Learning outcomes ("goals")

- Discuss and use features of different data models: the entity-relationship model, the relational model and the semi-structured model.
- Apply design theory for relational databases.
- Describe the effect of indexes and transactions in a relational database.
- Describe how access can be controlled via user authorisation.
- Implement a database design using a data definition language.
- Query and modify data using a data manipulation language.
- Express queries in relational algebra.
- Implement a database application in a host language.
- Construct an entity-relationship diagram for a given domain.
- Design and implement a database application that meets given requirements.

Examination

- Written exam
  - Thursday 20 December 2012, 14:00-18:00 (but check Student Portal)
  - 60 points (3/4/5 = 24/36/48, G/VG = 24/42)
- Four assignments to be submitted
  - we recommend that you work in pairs
  - work must be submitted via the 'fire' system
  - obtain Oracle username and password via 'fire' system

Course Web Page

`http://www.cse.chalmers.se/edu/course/TDA357/`

A database is …

- a collection of data
- managed by specialised software called a **database management system** (DBMS) (or, informally, a "database system")
- needed for large amounts of persistent, structured, reliable and shared data

Why a whole course in Databases?

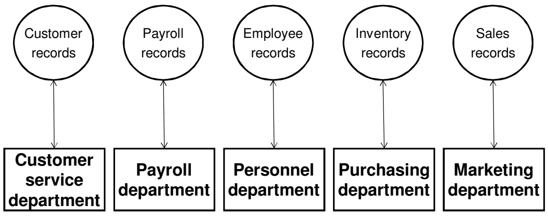
Banking, ticket reservations, customer records, sales records, product records, inventories, employee records, address books, demographic records, student records, course plans, schedules, surveys, test suites, research data, genome bank, medical records, time tables, news archives, sports results, e-commerce, user authentication systems, web forums, www.imdb.com, the world wide web, …

Databases are everywhere!

Examples

- Banking
  - Drove the development of DBMS
- Industry
  - Inventories, personnel records, sales …
  - Production Control
  - Test data
- Research
  - Sensor data
  - Geographical data
  - Laboratory information management systems
  - Biological data (e.g. genome data)

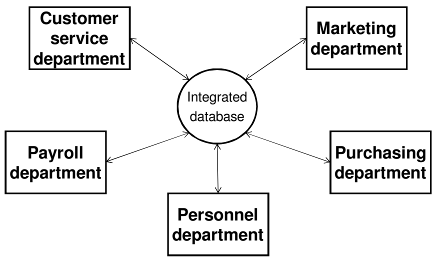
File-oriented information system



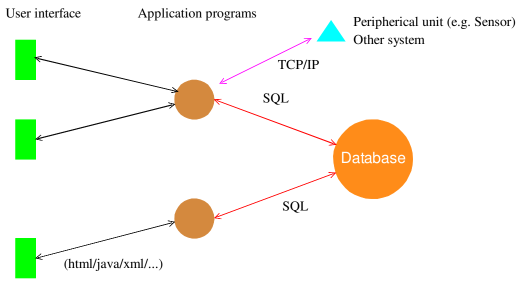
Problems with working with files

- Redundancy
  - Updates
  - Wasted space
- Changing a data format will require all application programs that read/write these files to be changed.
- Sharing information between departments can be difficult.

Database-oriented information system



Typical Computer System



Centralised control of data

- amount of redundancy can be reduced
  - less inconsistency in the stored data
- stored data can be shared
- standards can be enforced
- security restrictions can be applied
- data integrity can be maintained
  - validation done in one place
- conflicting requirements can be balanced
- provides **data independence**
  - can change storage structure without affecting applications

Motivation for database systems

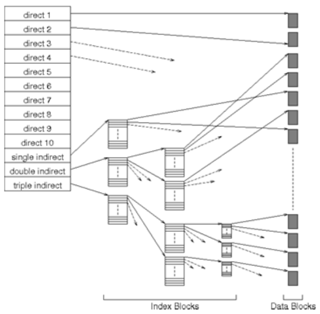
Needed for large amounts of persistent, structured, reliable and shared data (Ted Codd, 1973)

- Large amounts:
  - needs indexing for fast access
  - needs a load utility
- Persistent:
  - needs schema definition of types which evolves
- Structured:
  - storage schema held with data
  - query language (e.g. SQL) independent of storage
- Shared:
  - locking mechanism for concurrent update
  - access control via DBMS
  - centralised integrity checking
- Reliable:
  - changes to disc pages are logged
  - commit protects against program of disc crash
  - can undo (rollback) uncommitted updates

Traditional File Structures

A short digression …

UNIX file management



Actual organisation is hidden

- Just as the file management system in an operating system gives the users the illusion that a text file is stored on disc as a long consecutive sequence of characters …
- … a database management system gives the users the illusion that their data are stored on disc in accordance with a **data model**.

Data models

- Storing data in a computer system requires describing the data according to some **data model**, in a form which can be represented directly within the computer.
- A **data model** specifies the rules according to which data are structured and also the associated operations that are permitted.

Why not a file system?

File systems are

- Structured
- Persistant
- Changable
- Digital

… but oh so inefficient!

Data models: brief overview

- "No data model"
  - Flat files
- "Classical" data models
  - Hierarchical (tree)
  - Network (e.g. CODASYL) (graph)
  - **Relational** (Codd, 1970) (tables)
- Semantic data models, e.g.
  - Entity-Relationship model (Chen, 1976)
  - Functional Data Model (Shipman, 1981)
  - SDM (Hammer and McLeod, 1981)

Database Management Systems

- Hierarchical databases:
  - "Easy" to design if only one hierarchy
  - Efficient access
  - Low-level view of stored data
  - Hard to write queries
- Network databases:
  - "Easy" to design
  - Efficient access
  - Low-level view of stored data
  - Very hard to write queries

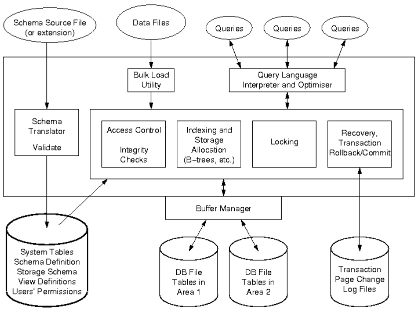
Database Management Systems

Relational databases:
  - **Hard to design**
  - Use specialized storage techniques
  - Efficient access
  - Provides high-level views of stored data based on mathematical concepts
  - **Easy to write queries**
  - Not all data fit naturally into a tabular structure
- Other databases:
  - Some based on a semantic data models
  - Object-oriented database management systems (OODBMS)
  - "NoSQL" ("not only SQL")

Relational DBMSs

- Very simple model
- Familiar tabular structure
- Has a good theoretical foundation from mathematics (set theory)
- Industrial strength implementations, e.g.
  - Oracle, Sybase, MySQL, PostgreSQL, Microsoft SQL Server, DB2 (IBM mainframes)
- Large user community

Database system architecture



Data Definition Language

"A language that allows the DBA [database administrator] or user to describe and name the entities, attributes and relationships required for the application, together with any associated integrity or security constraints."

[Definition from Connolly and Begg (2002) Database Systems: A Practical Approach to Design Implementation and Management. Third Edition. Addison Wesley.]

DDL statements are compiled into **metadata** ("data about data").

Data Manipulation Language

"A language that provides a set of operations to support the basic data manipulation operations on data held in the database."

[Definition from Connolly and Begg (2002) Database Systems: A Practical Approach to Design, Implementation and Management. Third Edition. Addison Wesley.]

Data manipulation operations include:

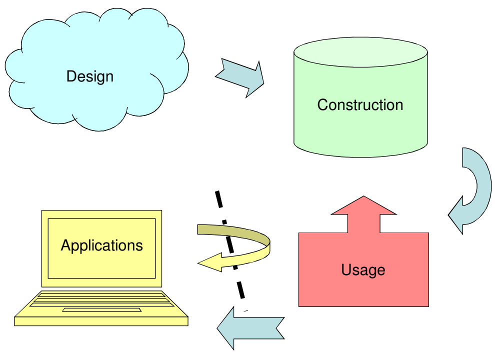
- inserting new data into the database;
- modifying data stored in the database;
- deleting data from the database;
- retrieving data from the database.

The part of the DML involved with data retrieval is called the **query language**.

Database system studies

1. Design of databases, e.g.
   - Entity-Relationship modelling
   - relational data model
   - dependencies and normalisation
   - XML and its data model
2. Database programming, e.g.
   - relational algebra
   - data manipulation and querying in SQL
   - application programs
   - querying XML
3. Database implementation, e.g.
   - indexes, transaction management, concurrency control, recovery, etc.

Course Objectives



Course Objectives – Design

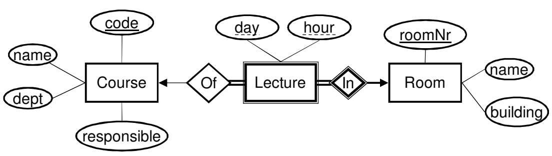
When the course is through, you should

- Given a domain, know how to design a database that correctly models the domain and its constraints

"We want a database that we can use for scheduling courses and lectures. This is how it's supposed to work: …"

Course Objectives – Design

- Entity-relationship (E-R) diagrams
- Functional Dependencies
- Normal Forms



Course Objectives – Construction

When the course is through, you should

- Given a database schema with related constraints, implement the database in a relational DBMS

```
Courses(code, name, dept, examiner)
Rooms(roomNr, name, building)
Lectures(roomNr, day, hour, course)
  roomNr -> Rooms.roomNr
  course -> Courses.code
```

Course Objectives – Construction

- SQL Data Definition Language (DDL)

```
CREATE TABLE Lectures
(
  lectureId INT       PRIMARY KEY,
  roomId    REFERENCES Rooms(roomId),
  day       INT       CHECK (day BETWEEN 1 AND 7),
  hour      INT       CHECK (hour BETWEEN 0 AND 23),
  course    REFERENCES Courses(code),
  UNIQUE (roomId, day, hour)
);
```

Course Objectives – Usage

When the course is through, you should

- Know how to query a database for relevant data using SQL
- Know how to change the contents of a database using SQL

”Add a course ’Databases’ with course code ’TDA357’, given by …”
”Give me all information about the course ’TDA357’”

Course Objectives – Usage

- SQL Data Manipulation Language (DML)

```
INSERT INTO Courses VALUES
('TDA357', 'Databases','CS', 'Niklas Broberg');
```

- Querying with SQL

```
SELECT * FROM Courses WHERE code = 'TDA357';
```

Course Objectives – Applications

When the course is through, you should

- Know how to connect to and use a database from external applications

”We want a GUI application for booking rooms for lectures …”

Course Objectives – Applications

- JDBC

```
// Assemble the SQL command for inserting the
// newly booked lecture.
String myInsert = "INSERT INTO Lectures "
  + "VALUES (" + room + ", "
  + day + ", " + hour + ", " + course + ")";

// Execute the SQL command on the database
Statement stmt = myDbConn.createStatement();
stmt.executeUpdate(myInsert);
```

Course Objectives - Summary

You will learn how to

- design a database
- construct a database from a schema
- use a database through queries and updates
- use a database from an external application

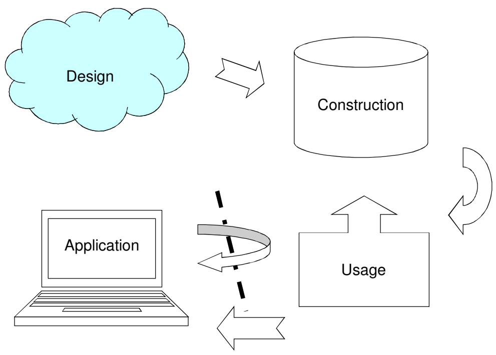
Lab Assignment

- Write a ”student portal” application in Java
  - Part I: Design
    - Given a domain description, design a database schema using an E-R diagram and functional dependencies.
  - Part II: Construction and Usage
    - Implement the schema from Part I in Oracle.
    - Insert relevant data.
    - Create views.
  - Part III: Construction
    - Create triggers.
  - Part IV: Interfacing from external Application
    - Write a Java application that uses the database from Part III.

Database design

Relations

Course Objectives



Course Objectives – Design

When the course is through, you should

- Given a domain, know how to design a database that correctly models the domain and its constraints

”We want a database that we can use for scheduling courses and lectures. This is how it’s supposed to work: …”

Designing a database

- ”Map” the domain, find out what the database is intended to model
  - The database should accept all data that are possible in reality
  - The database should agree with reality and not accept impossible or unwanted data
- Construct the ”blueprint” for the database – the database ***schema***

Relation Schemas

- In the relational data model, a design consists of a set of **relation schemas**.
- A relation schema has
  - a name, and
  - a set of attributes (+ types):

```
Courses(code, name, teacher)
```

name

attributes

Schema vs Instance

- ***Schema*** (or ***intension*** of a relation)
  - name and attributes of a relation

```
Courses(code, name, teacher)
```

- ***Instances*** (or ***extension*** of a relation)
  - the actual data
  - a set of **tuples**:

```
{ ('TDA357', 'Databases', 'Niklas Broberg'),
  ('TIN090', 'Algorithms', 'Devadatt Dubhashi') }
```

tuples

(Like a blueprint for a house, and the actual house built from it.)

From schema to database

- The relations of the database schema become the tables when we implement the database in a DBMS. The tuples become the rows:

```
Courses(code, name, teacher)
```

relation schema

table instance

code	*name*	*teacher*
'TDA357'	'Databases'	'Niklas Broberg'
'TIN090'	'Algorithms'	'Devatt Dubhashi'

Keys

- Relations have **keys** – attributes whose values uniquely determine the values of all other attributes in the relation.

```
Courses(code, name, teacher)
```

key

{('TDA357', 'Databases', 'Niklas Broberg'),
~~('TDA357', 'Algorithms', 'Devdatt Dubhashi')~~}

Composite keys

- Keys can consist of several attributes

```
Courses(code, period, name, teacher)

{('TDA357', 2, 'Databases', 'Niklas Broberg'),
 ('TDA357', 4, 'Databases', 'Rogardt Heldal')}
```

Quiz time!

What's wrong with this schema?

```
Courses(code, period, name, teacher)

{('TDA357', 2, 'Databases', 'Niklas Broberg'),
 ('TDA357', 4, 'Databases', 'Rogardt Heldal')}
```

Redundancy!

```
Courses(code, name)
CourseTeachers(code, period, teacher)
```

Next Lecture

More on Relations
Entity-Relationship diagrams