

## Who, where, when

Database Management Systems  
(LIX022B05)

Instructor: Çağrı Çöltekin  
c.coltekin@rug.nl

Information science/Informatiekunde

Fall 2012

Course Databases (LIX022B05) 2012/13  
Instructor Çağrı Çöltekin  
Email c.coltekin@rug.nl  
Lectures Mon 13:00–15:00, A weg 30, room 103  
Labs Wed 15:00–17:00, 1312.0107 MB  
Office hours Wed 13:00–15:00, H1311.0426  
Course page <http://www.let.rug.nl/coltekin/db2012/>

## Literature

## Textbook:

*Database System Concepts* by A. Silberschatz, H. F. Korth and S. Sudarshan. McGraw-Hill (2010), ISBN 978-007-128959-7 (6th international ed.)

## More References:

- ▶ *Database Management Systems* by Ramakrishnan & Gehrke
- ▶ *A First Course in Database Systems* by Ullman & Widom
- ▶ *Fundamentals of Database Systems* by Elmasri & Navathe
- ▶ *Database Systems: The Complete Book* by Garcia-Molina, Ullman & Widom
- ▶ *Database Systems: A Practical Approach to Design, Implementation, and Management* by Connolly & Begg

## Evaluation

## Grading:

Homeworks & labs 30%  
Final exam (Tentamination) 70%

## Homework assignments:

- ▶ Six homeworks: a homework each week (except this week)
- ▶ Homeworks will be combination of theoretical questions with some practical (mostly SQL) exercises.
- ▶ Practical part of the homeworks will be related to the lab exercises.
- ▶ You have a week for each homework: **no extensions!**
- ▶ Evaluation will be based on best five homework scores.

## Lab Sessions

## Homework assignments:

- ▶ You need an account on [siegfried.let.rug.nl](http://siegfried.let.rug.nl). If you do not have an account apply to A. da Costa  
Room: 1313.336  
[Mo-Thu, 10:30-12:00](#) and [14:00-15:30](#)
- ▶ You also need a MySQL user account and a database for the exercises.
- ▶ No programming, mostly exercises with SQL.

## About this course

This is an *introductory* course on database management systems. The particular focus will be on relational database management systems.

- ▶ No initial knowledge of databases required.
- ▶ There is no programming in this course.
- ▶ We will have a practical focus, but the theories behind the relational database design practices and queries are also introduced.

## After this course . . .

You should be able to

- ▶ develop a conceptual data model that reflects an organization's database requirements
- ▶ convert the conceptual data model into a relational database schema
- ▶ apply normalization techniques
- ▶ identify data integrity and security requirements
- ▶ be able to construct complex SQL queries
- ▶ be familiar with fundamentals of database administration, performance and optimization
- ▶ gain hands-on experience with a database management system (MySQL)

## Time plan

Week	Lecture (Mon)	Lab (Wed)
1	Introduction	No Lab this week.
2	Conceptual DB design, E-R diagrams	DB Design with E-R diagrams
3	Logical DB design, normalization	Implement a DB in MySQL
4	SQL 1: simple queries	Query exercises
5	SQL 2: more complex queries	More query exercises
6	SQL 3: views, indexes, access control ...	Query optimization, Indexes, DBA tasks
7	Summary & introduction to SQL and programming	Q&A

## Next...

Next half-semester course 'Database-driven web technology', will cover:

- ▶ A more practical approach to the subjects in this course.
- ▶ Some programming for web applications (PHP).
- ▶ Using relational databases from web applications.
- ▶ More/practical topics including
  - ▶ transaction processing
  - ▶ security
  - ▶ performance

## Introduction

## What is a database?

A *database* is a collection of related data.

- ▶ A company database: employees, departments, salaries, ...
- ▶ A bank database: customers, accounts, loans, credits, ...
- ▶ Airline flight reservation database: flights, seats, tickets, ...
- ▶ A library catalog: books, authors, ...
- ▶ University student database: students, instructors, grades, ...
- ▶ A database of DNS records: domain names, IP addresses, ...
- ▶ The collection of documents in Wikipedia: documents, authors, revisions, ...
- ▶ The phone book on your mobile phone: contacts, phone numbers, email addresses, ...
- ▶ ...

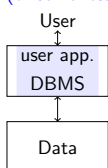
## Introduction

## Why use a DBMS

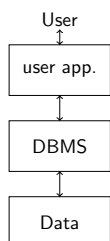
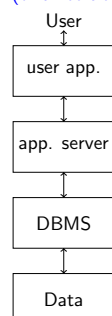
- ▶ Insulation between program and data
- ▶ Multiple views of the same data
- ▶ Sharing data in multi-user environments
- ▶ Controlling redundancy
- ▶ Enforcing data integrity
- ▶ Access control
- ▶ Efficient query processing
- ▶ Backup and recovery
- ▶ Multiple user interfaces

## Introduction

## DBMS architectures

Self-contained  
(or serverless)

## Client/server

Multi tier  
(or three tier)

## Today

- ▶ What is a database? and database management system (DBMS)?
- ▶ Why use a DBMS?
- ▶ Why not use a DBMS?
- ▶ Ways of organizing data
- ▶ Common DB architectures
- ▶ A quick introduction to RDBMSs: tables, keys, queries.

## Introduction

## What is a database management system?

A database management system (DBMS) is a *general purpose* software system for *creating, maintaining and sharing* data.

A DBMS,

- ▶ allows creating a database
- ▶ allows populating the database and manipulating the data
- ▶ enables queries on the data stored in the database
- ▶ enforces data integrity
- ▶ provides data access control

## Introduction

## Why not use a DBMS?

- ▶ The overhead of DBMS
- ▶ Specialized data access
- ▶ Cost of DBMS
- ▶ Simple and well-defined read-only data
- ▶ Real-time systems
- ▶ Single-user environments.

## Introduction

## Typical roles in a DBMS environment

- ▶ User
- ▶ Application programmer
- ▶ Database designer
- ▶ Database administrator

## Relational DBMSs

In this course we will focus on *relational* database management systems (RDBMS), where data and the relations between the data is organized in the form of *tables* (or, relations).

Book			
ISBN	title	year	pages
0330258648	The Hitchhiker's Guide to the Galaxy	1979	180
055338256X	I, Robot	1950	272
0553383043	A Wizard of Earthsea	1968	192

Author	
name	ISBN
Douglas Adams	0330258648
Isaac Asimov	055338256X
Ursula K. LeGuin	0553383043

Genre	
genre	author
comedy	0330258648
sci-fi	0330258648
fantasy	0553383043

## Types of DBMSs

Relational data model and RDBMSs are the dominant method of modeling and managing databases. However, it is not the *only* way. Historical precursors:

- ▶ Hierarchical
- ▶ Network

Somewhat new:

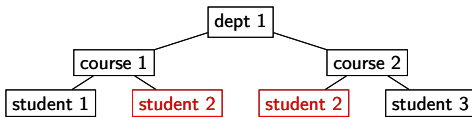
- ▶ Object-oriented or object-relational

.. and becoming popular:

- ▶ So-called NoSQL databases covering a wide range of methods of organizing data.

## Hierarchical databases

Initial DBMSs followed a *hierarchical* data organization. All records in a hierarchical database is organized according to a hierarchy.

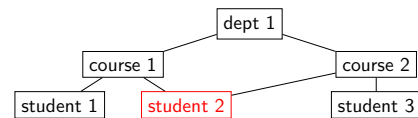


Main problem: data replication.

Although this is a serious problem for typical database applications. The hierarchical databases are preferable, and still popular, for certain applications (e.g. DNS, LDAP).

## Network databases

To overcome the replication problem with the hierarchical databases, network databases allow arbitrary links (representing relations) between the data records.



Main problem: complexity.

## Object-based databases

With the popularity of object-oriented programming languages, object oriented database management systems are suggested.

**Object relational:** extension of relational database systems to support object-oriented notions like user-defined data types, inheritance, encapsulation etc.

**Object oriented:** supports objects from an object-oriented programming language to be stored in a database.

The object-based databases are still not standardized, and relational databases are still the dominant approach to standard database applications.

## NoSQL databases

A large range of database management system that are collectively called *NoSQL* databases has (re)gained popularity in recent years.

Examples include:

- ▶ Key-value stores (Berkeley DB)
- ▶ Document stores (Apache CouchDB)
- ▶ Graph (FlockDB)
- ▶ Tabular (Google BigTable)
- ▶ Tuple store (Apache River)

## Structure of a relational database

- ▶ A relational database consists of multiple relations, or tables.
- ▶ Information is broken into multiple tables.
- ▶ The relevant information is accessed through references between tables.
- ▶ A bad database design results in problems such as data replication, or inconsistency.

## Anatomy of a table (or relation)

Rows (records, tuples)	Column names	Columns (attributes, fields)		
	ISBN	title	year	pages
	0330258648	The Hitchhiker's ...	1979	180
	055338256X	I, Robot	1950	272
	0553383043	A Wizard of ...	1968	192

- ▶ **Domain** of an attribute is the set of allowed values the attribute can take.

## More on relations

ISBN	title	year	pages
0330258648	The Hitchhiker's Guide to the Galaxy	1979	180
055338256X	I, Robot	1950	272
0553383043	A Wizard of Earthsea	1968	null

- ▶ No two rows are identical.
- ▶ Order of rows and columns are not important. (NB. order may be important in some SQL statements)
- ▶ A domain is said to be *atomic* if the elements of the domain are considered indivisible.
- ▶ A special value '**null**' is allowed for unknown or inapplicable values.

## Foreign Key

Book			
ISBN	title	year	pages
0330258648	The Hitchhiker's Guide to the Galaxy	1979	180
055338256X	I, Robot	1950	272
0553383043	A Wizard of Earthsea	1968	192

Author	
name	ISBN
Douglas Adams	0330258648
Isaac Asimov	055338256X
Ursula K. LeGuin	0553383043

Genre	
genre	ISBN
comedy	0330258648
sci-fi	0330258648
sci-fi	0553383043

A **foreign key** is used for cross-referencing in a RDBMS. The set of attributes that form the foreign key in a (referencing) table is the primary key of another (referenced) table.

## SQL: create/drop table

To create a table:

```
create table table_name (attribute1 domain1, ...,
attributeN domainN,
(constraint1), ... (constraintm));
```

Example:

```
create table book (ISBN int,
title varchar(50),
year int,
pages varchar(50),
primary key (ISBN));
```

To drop (remove) a table:

```
drop table table_name;
```

Example:

```
drop table book;
```

## SQL: update records

```
update table_name
set attribute1=value1, ..., attributeN=valueN
where condition;
```

Example:

```
update book set title='I, Robot'
where ISBN='055338256X';
```

## Primary Key

ISBN	title	year	pages
0330258648	The Hitchhiker's Guide to the Galaxy	1979	180
055338256X	I, Robot	1950	272
0553383043	A Wizard of Earthsea	1968	192

A **primary key**, formed by one or more attributes, uniquely identifies a row in a table.

- ▶ In worst case, the values of all attributes in a row has to be unique.
- ▶ A *candidate key* is one with no redundant attributes.
- ▶ There may be more than one candidate keys. Choice of primary key is a database design decision.

## Languages for creating, changing, querying

- ▶ Data Definition Language (DDL) allows creating relations that form a database.
- ▶ Data Manipulation Language (DML) allows changing the data in the database.
- ▶ A query language allows finding specific information in the database.

The *the* standard language for all above purposes (and more) is called *SQL*. Others exist (e.g. QBE), but SQL is the language supported by all main RDBMSs.

## SQL: insert/delete records

To insert a new record:

```
insert into table_name (attribute1, ..., attributeN)
values (value1, ..., valueN);
```

Example:

```
insert into book
values ('055338256X', 'I, Robot',
1950, 272);
```

To remove record(s):

```
delete from table_name where condition;
```

Example:

```
delete from book where ISBN = '055338256X';
```

## SQL: queries

```
select attribute1, ..., attributeN
from table1, ..., tableM
where condition;
```

Examples:

```
select ISBN, title from book
where year > 1960;
```

```
select * from instructor
where ISBN='055338256X';
```

## Summary / Next week

Today:

- ▶ A general introduction to databases and database management systems, relational databases and SQL.
- ▶ We will return to (almost) all subjects introduced today in later weeks.

Wednesday:

- ▶ No lab this week.
- ▶ [Obtain a database account on siegfried, or check if you already have one.](#)

Next Week:

- ▶ Conceptual database design.
- ▶ Read Chapter 7. (We will not study Section 7.8 about extended E-R features).