Database Management Systems
(LIX022B05)

Instructor: Çağrı Çöltekin

c.coltekin@rug.nl

Information science/Informatiekunde

Fall 2012
### Who, where, when

<table>
<thead>
<tr>
<th>Course</th>
<th>Databases (LIX022B05) 2012/13</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instructor</td>
<td>Çağrı Çöltekin</td>
</tr>
<tr>
<td>Email</td>
<td><a href="mailto:c.coltekin@rug.nl">c.coltekin@rug.nl</a></td>
</tr>
<tr>
<td>Lectures</td>
<td>Mon 13:00–15:00, A weg 30, room 103</td>
</tr>
<tr>
<td>Labs</td>
<td>Wed 15:00–17:00, 1312.0107 MB</td>
</tr>
<tr>
<td>Office hours</td>
<td>Wed 13:00–15:00, H1311.0426</td>
</tr>
<tr>
<td>Course page</td>
<td><a href="http://www.let.rug.nl/coltekin/db2012/">http://www.let.rug.nl/coltekin/db2012/</a></td>
</tr>
</tbody>
</table>
Literature

Textbook:

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: 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. 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

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

A *database* is a collection of related data.
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, . . .
- . . .
What is a database management system?

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

Self-contained (or serverless)

Client/server

Multi tier (or three tier)

User

user app.

DBMS

Data
DBMS architectures

Self-contained (or serverless)  Client/server  Multi tier (or three tier)

User  

↓  ↓  ↓

user app. user app. user app.

DBMS  DBMS  DBMS

Data  Data  Data
DBMS architectures

Self-contained (or serverless)

User
↓
user app.

DBMS
↓
Data

Client/server

User
↓
user app.

DBMS
↓
Data

Multi tier (or three tier)

User
↓
user app.
DBMS architectures

Self-contained (or serverless)

User

user app.

DBMS

Data

Client/server

User

user app.

DBMS

Data

Multi tier (or three tier)

User

user app.

app. server

DBMS

Data
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).

<table>
<thead>
<tr>
<th>ISBN</th>
<th>title</th>
<th>year</th>
<th>pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>0330258648</td>
<td>The Hitchhiker’s Guide to the Galaxy</td>
<td>1979</td>
<td>180</td>
</tr>
<tr>
<td>055338256X</td>
<td>I, Robot</td>
<td>1950</td>
<td>272</td>
</tr>
<tr>
<td>0553383043</td>
<td>A Wizard of Earthsea</td>
<td>1968</td>
<td>192</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>name</th>
<th>ISBN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Douglas Adams</td>
<td>0330258648</td>
</tr>
<tr>
<td>Isaac Asimov</td>
<td>055338256X</td>
</tr>
<tr>
<td>Ursula K. LeGuin</td>
<td>0553383043</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>genre</th>
<th>author</th>
</tr>
</thead>
<tbody>
<tr>
<td>comedy</td>
<td>0330258648</td>
</tr>
<tr>
<td>sci-fi</td>
<td>0330258648</td>
</tr>
<tr>
<td>fantasy</td>
<td>0553383043</td>
</tr>
</tbody>
</table>
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.
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.
Hierarchical databases

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

![Diagram showing a hierarchical database structure with a department (dept 1), courses (course 1, course 2), and students (student 1, student 2, student 2, student 3).]

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.
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.
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)

- **Column names**
- **Columns (attributes, fields)**

### Rows (records, tuples)

<table>
<thead>
<tr>
<th>ISBN</th>
<th>title</th>
<th>year</th>
<th>pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>0330258648</td>
<td>The Hitchhiker’s . . .</td>
<td>1979</td>
<td>180</td>
</tr>
<tr>
<td>055338256X</td>
<td>I, Robot</td>
<td>1950</td>
<td>272</td>
</tr>
<tr>
<td>0553383043</td>
<td>A Wizard of . . .</td>
<td>1968</td>
<td>192</td>
</tr>
</tbody>
</table>

▶ **Domain** of an attribute is the set of allowed values the attribute can take.
More on relations

<table>
<thead>
<tr>
<th>ISBN</th>
<th>title</th>
<th>year</th>
<th>pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>0330258648</td>
<td>The Hitchhiker’s Guide to the Galaxy</td>
<td>1979</td>
<td>180</td>
</tr>
<tr>
<td>055338256X</td>
<td>I, Robot</td>
<td>1950</td>
<td>272</td>
</tr>
<tr>
<td>0553383043</td>
<td>A Wizard of Earthsea</td>
<td>1968</td>
<td>null</td>
</tr>
</tbody>
</table>

- 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.
Primary Key

<table>
<thead>
<tr>
<th>ISBN</th>
<th>title</th>
<th>year</th>
<th>pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>0330258648</td>
<td>The Hitchhiker’s Guide to the Galaxy</td>
<td>1979</td>
<td>180</td>
</tr>
<tr>
<td>055338256X</td>
<td>I, Robot</td>
<td>1950</td>
<td>272</td>
</tr>
<tr>
<td>0553383043</td>
<td>A Wizard of Earthsea</td>
<td>1968</td>
<td>192</td>
</tr>
</tbody>
</table>

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.
A quick introduction to RDBMSs

Foreign Key

<table>
<thead>
<tr>
<th>ISBN</th>
<th>title</th>
<th>year</th>
<th>pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>0330258648</td>
<td>The Hitchhiker’s Guide to the Galaxy</td>
<td>1979</td>
<td>180</td>
</tr>
<tr>
<td>055338256X</td>
<td>I, Robot</td>
<td>1950</td>
<td>272</td>
</tr>
<tr>
<td>0553383043</td>
<td>A Wizard of Earthsea</td>
<td>1968</td>
<td>192</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>name</th>
<th>ISBN</th>
<th>genre</th>
<th>ISBN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Douglas Adams</td>
<td>0330258648</td>
<td>comedy</td>
<td>0330258648</td>
</tr>
<tr>
<td>Isaac Asimov</td>
<td>055338256X</td>
<td>sci-fi</td>
<td>0330258648</td>
</tr>
<tr>
<td>Ursula K. LeGuin</td>
<td>0553383043</td>
<td>sci-fi</td>
<td>0553383043</td>
</tr>
</tbody>
</table>
A quick introduction to RDBMSs

Foreign Key

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.
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: create/drop table

To create a table:

```
create table table_name (attribute_1 domain_1, ..., attribute_N domain_N, (constraint_1), ... (constraint_m));
```

Example:

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

To create a table:
```
create table table_name (attribute_1 domain_1, \ldots, attribute_N domain_N, (constraint_1), \ldots (constraint_m));
```
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: insert/delete records

To insert a new record:

\[
\text{insert into table\_name (attribute}_1, \ldots \text{attribute}_N) \\
\text{values (value}_1, \ldots \text{value}_N);
\]

Example:

\[
\text{insert into book values ('055338256X', 'I\_Robot', 1950, 272);
}\]
SQL: insert/delete records

To insert a new record:

```
insert into table_name (attribute_1, ... attribute_N)
values (value_1, ... value_N);
```

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: update records

```
update table_name
    set attribute_1=value_1, ..., attribute_N=value_N
where condition;
```
SQL: update records

```
update table_name
    set attribute_1=value_1, . . . , attribute_N=value_N
where condition;
```

Example:

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

\[
\text{select } \text{attribute}_1, \ldots, \text{attribute}_N \\
\text{from } \text{table}_1, \ldots, \text{table}_M \\
\text{where } \text{condition};
\]
**SQL: queries**

```sql
select attribute_1, \ldots, attribute_N 
from table_1, \ldots, table_M 
where condition;
```

Examples:

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

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