General Information

- **Professor:** Leonid Libkin
  - **Contact:** libkin@ed.ac.uk

- **Lectures:** Tuesday, 11:10am – 1 pm, AT LT4
- **Website:**
Course Information

- Email:
  - Use your UoE account
  - Use ‘DBS’ in the subject line
  - Always sign using your name
  - No attachments
  - Never send me code via email
  - I will try to answer by the end of the next business day

- Academic offences: Don't!
Course information cont’d

- 3 assignments + exam
- 1st assignment (5%) - pencil/paper, relational algebra/calculus, SQL
- 2nd assignment (15%) - use PostgreSQL (available on dice machines). Extensive SQL programming exercise
- 3rd assignments (5%) - pencil/paper again, database design, query processing, basics of XML
- Exam - 75%
Textbook

- **Recommended:**
The goal:

You will learn:

- General principles of database systems that apply to all (certainly most) products you are likely to deal with.

- However, you will not learn system-specific issues
What will you learn in this course

- **Database Design Methodology**
  - start from a general application description (in verbal form)
  - abstract and optimize the requirements (ER modeling)
  - map the requirements into entities that an RDBMS understands (extract database relations)
  - optimize the relations (normalization)
What will you learn in this course (cont'd)

- **Use of an RDBMS**
  - write queries in a language that a DBMS understands (SQL)
  - implement your application using a language you are familiar with, suitably enhanced with SQL statements with the help of the DBMS

- **Basics of XML**
  - Schemas (DTDs), query languages (XPath, XQuery)
What you will not learn in this course

- **System-specific issues**
  - what is it that I can do in PostgreSQL but not in DB2 or Oracle, and vice versa
  - what level of query nesting is permitted in the latest version of Sybase
  - the exact syntax of Oracle's `connect by prior` clause
  - comparison with the `with recursive` clause in the SQL3 standard
  - etc etc
Why learn about databases?

- It used to be about boring things: employee records, bank records, etc.
- Today, the field covers all aspects of working with data:
  - Web search
  - Data mining
  - Scientific and medical databases
  - Integrating information
- Databases are behind almost everything you do on the Web
  - Google searches
  - Queries at Amazon, eBay, etc.
  - Trip planning (expedia etc)
What is Data Management?

- Find data (search and query)
- Update or modify data
- Ensure data consistency
- Protect data
  - from unauthorized access (access control)
  - from failures (recovery)
  - from other programs or users (concurrency control)
Finding data?

- **Query:** *Find the average enrollment in database courses at UoE?*
- **How could we find this using a conventional search within file system?**
  - Do we get what we want?
  - Why is this hard?
- **How could we find this using a Database Management System (DBMS)?**
What is a DBMS?

- **Database**: A large collection of data.
  - Examples: databases of customers, products,...
- A database usually models (some part of) a real-world **enterprise**.
  - Entities (e.g., students, courses)
  - Relationships (e.g., John Doe is taking DBS)
- A **Database Management System (DBMS)** is a software package designed to store and manage databases.
- Many vendors: IBM, Sybase, Oracle, Microsoft, etc
Simplified database system environment
Typical DBMS Functionality

- **Define** a particular database in terms of its data types, structures, and constraints
- **Construct** or Load the initial database contents on a secondary storage device
- **Manipulating** the database:
  - Retrieval: Querying, generating reports
  - Modification: Insertions, deletions, and updates to its content
  - Accessing the database through Web applications
- **Processing and Sharing** by a set of concurrent users and application programs – yet, keeping all data valid and consistent
Typical DBMS Functionality

- Other features:
  - Protection or Security measures to prevent unauthorized access
  - Presentation and Visualization of data
  - Maintaining the database and associated programs over the lifetime of the database application
Why Use a DBMS?

- **Self-describing nature of a database system:**
  - A DBMS **catalog** stores the description of a particular database (e.g. data structures, types, and constraints).
  - The description is called **meta-data**.

- **Data independence**
  - You don’t need to know the implementation of the database to access the data.
Why use a DBMS - Data Independence

- Applications insulated from how data is structured and stored
  - change the order of tuples
  - add or modify other columns
  - add or modify indexes
- Note that query does not change when physical structure changes

One of the most important benefits of using a DBMS
Why Use a DBMS?

- **Efficient access**
  - queries are optimized.

- **Reduced application development time**
  - Queries can be expressed declaratively, we do not need to indicate how to execute them.

- **Data integrity and security**
  - Some constraints on the data are enforced automatically.
Why use a DBMS - Data Consistency

- **Data Constraints:**
  - All students must have a student ID (sID)
  - No two students can have the same sID (uniqueness)
  - A student may only have one grade per course
  - Etc.
Why Use a DBMS?

- Concurrent access, recovery from crashes
  - Many users can access/update the database at the same time without any interference.
- Speed – even when the data is huge, i.e.
  - IRS: 150 TB (1 TB ≈ 10^{12} B)
  - Yahoo: 2 PB (1 PB ≈ 10^{15} B)
  - National Energy Research Scientific Computing Center (USA): 3.5 PB
Why use a DBMS - Concurrency Control

- Concurrent execution of user programs is essential for good DBMS performance.
  - Because disk accesses are frequent, and relatively slow
- Interleaving actions of different user programs can lead to inconsistency:
  - A cheque is cleared while account balance is being computed.
- DBMS ensures that such problems do not arise: users can behave as if they were using a single-user system.
Why use a DBMS - Data Abstraction

Many **views**, single **logical schema** and **physical schema**.

- Views (external schemas) describe how users see the data.
- Logical schema defines logical structure.
- Physical schema describes the files and indexes used.
Example: University Database

- **Conceptual schema:**
  - `Student(sid: string, name: string, login: string, age: integer, gpa: real)`
  - `Course(cid: string, cname: string, credits: integer)`
  - `Enrolled(sid: string, cid: string, grade: string)`

- **Physical schema:**
  - Relations stored as unordered files.
  - Index on first column of Students.

- **External Schema (View):**
  - `Course_info(cid: string, enrollment: integer)`

- describes data in terms of the data model of the DBMS
Describing Data: What is a Data Model?

- **Mathematical representation of data**
  - relational model = tables;
  - semistructured model = trees/graphs.

AND

- **Operations on data**
- **Constraints**
Describing Data: Data Models

- A *schema* is a description of a particular collection of data, using a given data model.
- The *relational model of data* is the most widely used model today.
  - Main concept: *relation*, basically a table with rows and columns.
  - Every relation has a *schema*, which describes the columns, or fields.
Example Instance of Student Relation

<table>
<thead>
<tr>
<th>sID</th>
<th>Name</th>
<th>Login</th>
<th>Age</th>
<th>GPA</th>
</tr>
</thead>
<tbody>
<tr>
<td>53666</td>
<td>Jones</td>
<td>Jones@cs</td>
<td>20</td>
<td>3.2</td>
</tr>
<tr>
<td>45453</td>
<td>Smith</td>
<td>Smith@ai</td>
<td>19</td>
<td>3.1</td>
</tr>
</tbody>
</table>

- Columns are *attributes*
- Rows are *tuples*
The SQL Query Language

- Find all students who are 20 years old

```sql
SELECT * 
FROM Students 
WHERE age = 20
```

<table>
<thead>
<tr>
<th>sID</th>
<th>Name</th>
<th>Login</th>
<th>Age</th>
<th>GPA</th>
</tr>
</thead>
<tbody>
<tr>
<td>53666</td>
<td>Jones</td>
<td>Jones@cs</td>
<td>20</td>
<td>3.2</td>
</tr>
</tbody>
</table>
Database Users

- Three groups:
  - End users
  - Database administrators
  - Database developers

- This course prepares you to be end-users of DBMSs. But to be an intelligent end-user you need to know how a DBMS operates.
Historical Development of Database Technology

- Early Database Applications:
  - The Hierarchical and Network Models were introduced in the mid 1960s and dominated during the 1970s.

- Relational Model based Systems:
  - Relational model was originally introduced in 1969 (40th anniversary is celebrated this year!) by E.F. Codd at IBM.
  - Relational DBMS Products emerged in the early 1980s.
Object-oriented and emerging applications:

- Object-Oriented Database Management Systems (OODBMSs) were introduced in the late 1980s and early 1990s to cater to the need of complex data processing - but failed to take off.

- 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 (eXtended Markup Language).
- Script programming languages such as PHP and JavaScript allow generation of dynamic Web pages that are partially generated from a database
  - Also allow database updates through Web pages
Extending Database Capabilities

- 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
  - Interoperability, integrating data from different sources