School of Computing and Information Technology
University of Wollongong

Session: **Autumn 2016**
Lecturer: **Janusz R. Getta**

CSIT115/CSIT815 Data Management and Security
Assignment 2

Scope

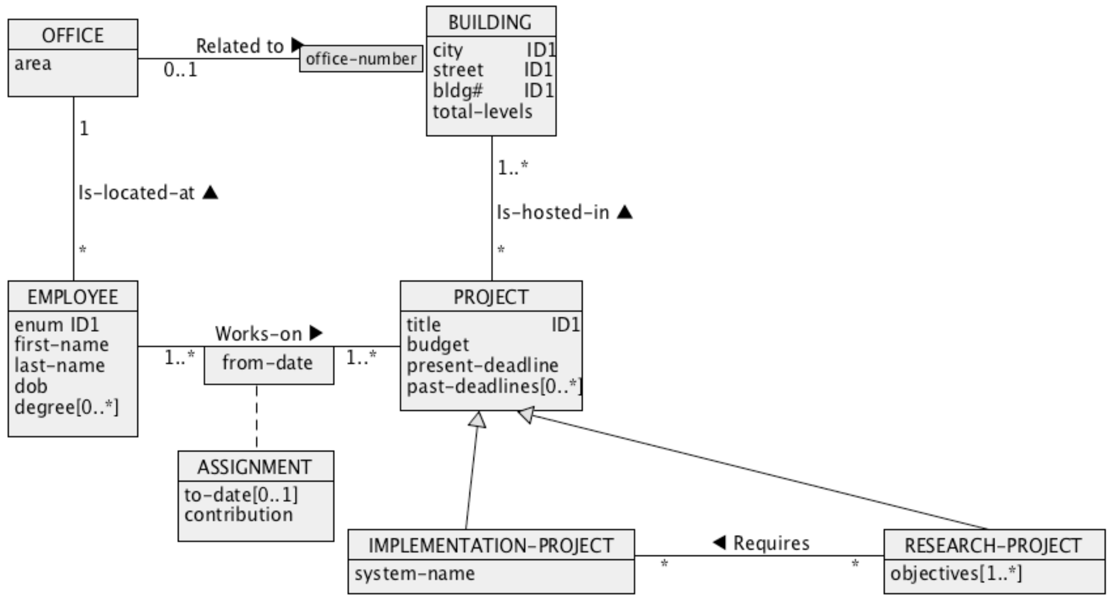
This assignment consists of the tasks related to the logical design of relational database, implementation of relational tables, and modification of the structures of relational database.

This assignment consists of 4 tasks and specification of each task starts from a new page.

Task

Task1 (2 marks)

Consider a conceptual schema given below.



Your task is to perform a step of logical database design, i.e. to transform a conceptual schema given above into a collection of relational schemas.

For each relational schema created clearly list the names of attributes, primary key, candidate keys (if any), and foreign keys (if any). Assume, that **superset method** must be used to implement a generalization. A way how a conceptual schema can be transformed into a collection of relational schemas is explained in a presentation 06 Logical Design.

The relational schemas must be listed in a format presented in the slides 43-44 in a presentation 06 Logical Design. There is no need to show all stages of the transformations. Only the final relational database design is expected.

Deliverables

A file `solution1.pdf` with a list of relational schemas, primary key for each relational schema, candidate keys (if any) for each relational schema, foreign keys (if any) for each relational schema

Task 2 (1 mark)
Consider the following collection of relational schemas.

```
SENDER(first-name, last-name, city, street, bldg-num,
       flat-num,total-parcel-sent)
primary key = (first-name, last-name, city, street,
               bldg-num, flat-num)
```

```
RECIPIENT(first-name, last-name, city, street, bldg-num,
          flat-num, total-parcel-received)
primary key = (first-name, last-name, city, street,
               bldg-num, flat-num)
```

```
PARCEL(send-date, deliver-date
       sender-first-name, sender-last-name, sender-city,
       sender-street, sender-bldg-num, sender-flat-num,
       recipient-first-name, recipient-last-name,
       recipient-city, recipient-street,
       recipient-bldg-num, recipient-flat-num,
       weight,fee)
primary key = (send-date,
       sender-first-name, sender-last-name, sender-city,
       sender-street, sender-bldg-num, sender-flat-num,
       recipient-first-name, recipient-last-name,
       recipient-city, recipient-street,
       recipient-bldg-num, recipient-flat-num)

foreign key 1 = (sender-first-name, sender-last-name,
                 sender-city,sender-street,
                 sender-bldg-num, sender-flat-num)
  references SENDER(first-name, last-name, city, street,
               bldg-num, flat-num)

foreign key 2 = (recipient-first-name, recipient-last-name,
                 recipient-city,recipient-street,
                 recipient-bldg-num, recipient-flat-num)
  references RECIPIENT(first-name, last-name, city, street,
               bldg-num, flat-num)
```

```
CONTENTS(send-date,
         sender-first-name, sender-last-name, sender-city,
         sender-street, sender-bldg-num, sender-flat-num,
         recipient-first-name, recipient-last-name,
         recipient-city, recipient-street,
          recipient-bldg-num, recipient-flat-num,
          item-name, total-items)

primary key = (send-date,
       sender-first-name, sender-last-name, sender-city,
       sender-street, sender-bldg-num, sender-flat-num,
       recipient-first-name, recipient-last-name,
       recipient-city, recipient-street,
       recipient-bldg-num, recipient-flat-num,
       item-name)

foreign key = (send-date,
       sender-first-name, sender-last-name, sender-city,
       sender-street, sender-bldg-num, sender-flat-num,
       recipient-first-name, recipient-last-name,
       recipient-city, recipient-street,
       recipient-bldg-num, recipient-flat-num)
   references PARCEL(send-date,
       sender-first-name, sender-last-name, sender-city,
       sender-street, sender-bldg-num, sender-flat-num,
       recipient-first-name, recipient-last-name,
       recipient-city, recipient-street,
       recipient-bldg-num, recipient-flat-num)
```

Your task is to perform *reverse database engineering*, i.e. to find a conceptual schema of a database that has a collection of relational schemas given above. Use UMLetLet to draw a conceptual schema found and save your design in a file `solution2.pdf` (you have to Export your design under UMLetLet).

Deliverables
A file `solution2.pdf` with the *reverse engineered* conceptual schema.

Task 3 (1 mark)
SQL script `dbcreate3.sql` contains `CREATE TABLE` statements that can be used to create a small relational that contains information about employees, department and locations of department (it is the same relational database as it is frequently used in Cookbook). A conceptual schema of the database is included in a file `schema3.pdf`.

Analyse, the contents of a script `dbcreate3.sql` to find out how the relational tables are implemented and then execute a script `dbcreate3.sql` to create the relational tables.

Your task is to implement SQL script `solution3.sql` with only (!) `ALTER TABLE` statements that implement the following modifications of the original relational tables included in the sample database.

(1) We would like to be able to add information about the total number of employees working at a particular location. Assume that no more than 1000 employees can work at the same location.

(2) We would like to increase the maximum length of department name to 50 characters;

(3) We would like to replace information about chairman of each department with information about manager of each department assuming that one of the employees is a manager of a department.

(4) We would like to increase maximum budget of a department to 5000.

(5) We would like to allow for automatic deletion of a row from `EMPLOYEE` table when a row from `DEPARTMENT` table is deleted. Additionally, when a row from a relational table `DEPARTMENT` is deleted then the respective values of foreign keys in a relational table `DEPTLOC` must be set to `NULL`.

You can find a lot of information about application of `ALTER TABLE` statements in a presentation 09 SQL - Data Definition Language (DDL) and in Cookbook, How to use data definition and basic data manipulation statements of SQL, Recipe 4.1 How to create and how to alter the relational tables ?

When a file `solution3.sql` is ready connect to MySQL either through command line client mysql or graphical user interface MySQL Workbench and execute your script file file.

It is recommended to use a script `drop3.sql` to drop all relational tables modified during execution of a script `solution3.sql` and it is recommend to execute `drop3.sql` after each execution of `solution3.sql` (you can also put `DROP TABLE` statements at the end of a script `solution3.sql`) and later re-create the original database with a script `dbcreate3.sql`. In such a way your script always operates on the original structures of the sample database.

When execution of your script returns no errors then connect to MySQL server using command based interface mysql and create a report from processing of the script `solution3.sql`. Save your report in a file `solution3.rpt`. You can find more information about creating reports from processing of SQL scripts in Cookbook, Recipe 3.1 How to use "mysql? Command based interface to MySQL database server ? Step 4 How to save the results of SQL processing in a file ?.

Your report must contain a listing of SQL statements processed.

You can find more information on how to display SQL statements while a script is processed in Cookbook, Recipe 3.1 How to use "mysql? Command based interface to MySQL database server ? Step 3 How to process SQL script ?.

A report that contains no listing of executed SQL statements scores no marks and report that contains errors also scores no marks !

Deliverables
A file `solution3.rpt` with a report from processing of SQL script `solution3.sql`. The report MUST have no errors and the report MUST list all SQL statements processed.

Task 4 (2 marks)
SQL script dbcreate4.sql contains CREATE TABLE statements of TPC-H benchmark database (http://www.tpc.org) that contains information about the parts, suppliers, customers, orders, and lines included within the orders. A conceptual schema of the database is included in a file schema4.pdf.

Analyse, the contents of a script dbcreate4.sql to find out how the relational tables are implemented and then execute a script dbcreate4.sql to create the relational tables.

Your task is to implement SQL script solution4.sql with CREATE TABLE, ALTER TABLE, and DROP TABLE statements that implement the following modifications of the original sample database. Note, that your script must operate on the already existing database and it must not create a new database from very beginning.

(1) It should be possible to store information about the total number of lines included in each order and about the total number of orders submitted by each customer so far.

(2) The following columns:

```
L_SHIPDATE          DATE        NOT NULL,
L_COMMITDATE        DATE        NOT NULL,
L_RECEIPTDATE       DATE        NOT NULL,
L_SHIPINSTRUCT      CHAR(25)    NOT NULL,
L_SHIPMODE          CHAR(10)    NOT NULL,
L_COMMENT           VARCHAR(44) NOT NULL,
```

included in a relational table LINEITEM must be moved to another relational table in order to decrease an average length of rows in a relational table LINEITEM. Such modification may have a positive impact on the performance of query processing on the table. A name of another relational table is up to you.

(3) A relational table PART must be decomposed into three relational tables PART_KEY, CHEAP_PART and EXPENSIVE_PART. A relational table PART_KEY must contain only information about all part numbers (P_PARTKEY column from a relational table PART). A relational table CHEAP_PART must contain all information about cheap parts, i.e. such that their retail price is below 100. A relational table EXPENSIVE_PART must contain all information about more expensive parts, i.e. such that their prices is equal or higher than 100. After the decomposition a relational table PART is no longer needed. Such modification may also have a positive impact on the performance when processing cheap or expensive parts.

(4) Information about regions and nations of the customers is no longer needed in the database.

Note, that you must enforce appropriate consistency constraints after all modifications !

When execution of your script returns no errors then connect to MySQL server using command based interface mysql and create a report from processing of the script `solution4.sql`. Save your report in a file `solution4.rpt`. You can find more information about creating reports from processing of SQL scripts in Cookbook, Recipe 3.1 How to use "mysql? Command based interface to MySQL database server ? Step 4 How to save the results of SQL processing in a file ?.

Your report must contain a listing of SQL statements processed.

You can find more information on how to display SQL statements while a script is processed in Cookbook, Recipe 3.1 How to use "mysql? Command based interface to MySQL database server ? Step 3 How to process SQL script ?.

A report that contains no listing of executed SQL statements scores no marks and report that contains errors also scores no marks !

Deliverables
A file `solution4.rpt` with a report from processing of SQL script `solution4.sql`. The report MUST have no errors and the report MUST list all SQL statements processed.

End of specification