School of Computing and Information Technology
University of Wollongong

Session: **Spring 2020**
Lecturer: **Janusz R. Getta**

CSCI235/CSCI835 Database Systems
Assignment 1
17 August 2020

Scope
This assignment includes the tasks related to database normalization, indexing of relational tables, and using cursors to search a database.

The outcomes of the laboratory work are due by **Saturday 5 September, 2020, 7.00 pm (sharp).**

Please read very carefully information listed below.

This assignment contributes to 20% of the total evaluation in a subject CSCI235 and it contributes to 16% of the total evaluation in a subject CSCI835.

A submission procedure is explained at the end of specification.

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

It is recommended to solve the problems before attending the laboratory classes in order to efficiently use supervised laboratory time.

A submission marked by Moodle as "late" is treated as a late submission no matter how many seconds it is late.

A policy regarding late submissions is included in the subject outline.

A submission of compressed files (zipped, gzipped, rared, tared, 7-zipped, lhzed, … etc) is not allowed. The compressed files will not be evaluated.

All files left on Moodle in a state `"Draft(not submitted)"` will not be evaluated.

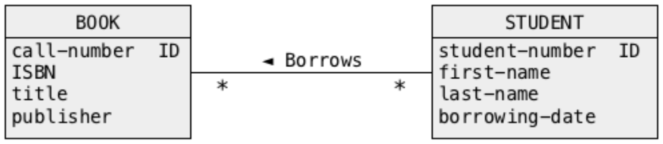
An implementation that does not compile due to one or more syntactical and/or run time errors scores no marks.

It is expected that all tasks included within **Assignment 1** will be solved **individually without any cooperation** with the other students. If you have any doubts, questions, etc. please consult your lecturer or tutor during lab classes or office hours. Plagiarism will result in a **FAIL** grade being recorded for the assessment task.

Task 1 (6 marks)
Normalization of relational tables

Consider the following conceptual schema of a sample database domain where the students borrow the books from a library. A book is described by a call-number that uniquely identifies each copy of a book kept in a library. A library purchases several identical textbooks for the subjects enrolled by the larger number of students. A book is identified by ISBN and it is described by a title and publisher.

The students are described by a student number, first name and last name. A date is recorded when a student borrows a book.



A database designer made few mistakes at both conceptual modelling stage and logical design stage (i.e. transformation of a conceptual schema into the relational schemas).

At present, the relational schemas created by a database designer are the following.

```
BOOK(call-number, ISBN, title, publisher)
primary key = (call-number)

STUDENT(student-number, first-name,last-name)
primary key = (student-number)

BORROW(call-number, ISBN, student-number, borrow-date)
primary key = (call-number, student-number)
```

Your task is to use the analysis of functional dependencies and normalization of relational schemas to find the highest normal form valid for each one of the relational schemas listed above. If a relational schema is not in BCNF then you must decompose it into the relational schemas in BCNF.

The process of normalization of relational schemas must be performed in the following way. First, find functional dependencies valid in a relational schema. Next, find the minimal keys. Next find, the highest normal form valid for a relational schema, and finally if a normal form found is not BCNF decompose a schema into BCNF.

Repeat such process for every relational schema listed above.

Please remember, that both conceptual schema and relational schemas are not completely correct, so be very careful when finding functional dependencies. Using a conceptual schema and relational schemas only is like "walking over a mine field".

Finally, please do not send to me emails saying that the design is incorrect, I know that it is incorrect !

Deliverables
A file `solution1.pdf` with a report from normalization of the relational schemas given above. A report must include a list of functional dependencies found, complete derivations of minimal keys, complete identification of the highest normal form valid and decomposition into BCNF whenever it is necessary.

Task 2 (3 marks)
Normalization of relational tables

(1) Consider the following relational schema in BCNF.

```
ROOM(bldg-number, room-number, room-area)
primary key = (bldg-number, room-number)
```

Add to a relational schema `ROOM` a **meaningful attribute** such that after addition the relational schema will be at most in 1NF and not in 2NF. Prove, that after addition of an attribute the relational schema is not in 2NF.

(2) Consider the following relational schema in BCNF.

```
FLIGHT(flight-number, from-city, to-city)
primary key = (flight-number)
```

Add to a relational schema `FLIGHT` a **meaningful attribute** such that after addition the relational schema will be at most in 2NF and not in 3NF. Prove, that after addition of an attribute the relational schema is not in 3NF.

(3) Consider the following relational schema in BCNF.

```
ORDER(order-number, supplier)
primary key = (order-number)
```

Add to a relational schema `RDER` a **meaningful attribute** such that after addition the relational schema will be at most in 3NF and not in BCNF. Prove, that after addition of an attribute the relational schema is not in BCNF.

Deliverables

A file `solution2.pdf` with a report from denormalization of relational schemas (1), (2), and (3). For each one of the cases listed above provide a proof that after addition of a meaningful attribute a relational schema is in a required normal form.

Task 3 (5 marks)
Indexing

Prologue

Download the files `dbcreate.sql`, `dbload.sql`, and `dbdrop.sql` included in a section SAMPLE DATABASES on Moodle. To drop a sample database, process a script `dbdrop.sql`. To create a sample database, process as script `dbcreate.sql`. To drop a sample database, process a script `dbdrop.sql`. To load data into a sample database, process as script `dbload.sql`.

Connect to Oracle database server and process the following SQL statement that saves a query processing plan for a given `SELECT` statement in `PLAN_TABLE`.

```
EXPLAIN PLAN FOR
 SELECT R_COMMENT
 FROM REGION
 WHERE R_NAME = 'Africa';
```

Next, process the following `SELECT` statement to display a query processing plan stored in `PLAN_TABLE`.

```
SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY);
```

Among the others, you should get the following results.

```
PLAN_TABLE_OUTPUT
----------------------------------------------------------------------------
| Id  | Operation          | Name    | Rows  | Bytes | Cost  (%CPU)| Time     |
----------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |         |     1 |   105 |     3   (0)| 00:00:01 |
|*  1 |  TABLE ACCESS FULL| REGION |     1 |   105 |     3   (0)| 00:00:01 |
----------------------------------------------------------------------------

Predicate Information (identified by operation id):

   1 - filter("R_NAME"='Africa')
```

A line `TABLE ACCESS FULL| REGION` in a plan given above indicates that a database system plans to read entire table `REGION` to compute the query.

Next, create an index on a column `R_NAME` in a relational table `REGION`.

```
CREATE INDEX REGION_IDX ON REGION(R_NAME);
```

Again, process the following SQL statements that save a query processing plan for the same `SELECT` statement as before in `PLAN_TABLE` and display a query processing plan stored in `PLAN_TABLE`.

```
EXPLAIN PLAN FOR
 SELECT R_COMMENT
 FROM REGION
 WHERE R_NAME = 'Africa';
```

Among the others, you should get the following results.

```
PLAN_TABLE_OUTPUT
---------------------------------------------------------------------------------------------------
| Id  | Operation                                    | Name        | Rows  | Bytes | Cost (%CPU)| Time     |
---------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                             |             |     1 |   105 |     1   (0)| 00:00:01 |
|   1 |  TABLE ACCESS BY INDEX ROWID BATCHED| REGION      |     1 |   105 |     1   (0)| 00:00:01 |
|*  2 |   INDEX RANGE SCAN                           | REGION_IDX |     1 |       |     1   (0)| 00:00:01 |
---------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):

   2 - access("R_NAME"='Africa')
```

This time a database system plans to use an index `REGION_IDX` created a moment ago to process the same query. Note, a line `INDEX RANGE SCAN    | REGION_IDX` in a plan given above means, that a database system plans to vertically traverse an index `REGION_IDX` to find the identifiers of rows that satisfy a condition `R_NAME = 'Africa';`. A line `TABLE ACCESS BY INDEX ROWID BATCHED| REGION` means, that the system plans to group row identifiers pointing to the same data blocks in order to minimize the total number of read block operations and to access data blocks of a relational table `REGION` pointed by the row identifiers to find the values of an attribute `R_COMMENT`.

Conclusions
`EXPLAIN PLAN` statement of SQL can be used to get information about a processing plan created by a query processor for a given `SELECT` statement. A query processing plan provides information on whether and index created earlier will be used for processing of SQL statement. We shall use `EXPLAIN PLAN` statement to check whether an index created to speed up `SELECT` statement will be used for processing of the statement.

To drop an index, process a statement

```
DROP INDEX REGION_IDX;
```

No report is expected from processing of SQL statements given above.

Problem
Your task is to find what indexes should be created to speed up processing of `SELECT` statements listed below. You are expected to create one index for one `SELECT` statement. To simplify the problem, assume that any index, later on used by a query processor to speed up processing of `SELECT` statement, will do.

(1)
```
SELECT O_ORDERDATE, O_CUSTKEY
FROM ORDERS
WHERE O_ORDERDATE > '31-DEC-1994' AND
      O_ORDERDATE < '01-JAN-1996';
```

(3)
```
SELECT L_PARTKEY, L_SUPPKEY
FROM LINEITEM
WHERE ( L_TAX IN (0.02, 0.06) OR L_DISCOUNT> 0.4 ) AND
      L_EXTENDEDPRICE = 7232;
```

(3)
```
SELECT DISTINCT PS_AVAILQTY, PS_SUPPLYCOST, PS_SUPPKEY
FROM PARTSUPP;
```

(4)
```
SELECT P_BRAND, AVG(P_RETAILPRICE)
FROM PART
GROUP BY P_BRAND;
```

(5)
```
SELECT *
FROM SUPPLIER
ORDER BY S_NAME, S_NATIONKEY;
```

Implement SQL script `solution3.sql` such that for each one of `SELECT` statements given above the script performs the following actions.

(i) Find and list a query processing plan for `SELECT` statement without an index.
(ii) Create an index.
(iii) Find and list a query processing plan for `SELECT` statement with an index.
(iv) Drop an index.

When ready process SQL script file `solution3.sql` and save a report from processing in a file `solution3.lst`.

Your report must include a listing of all PL/SQL statements processed. To achieve that put the following SQLcl commands:

```
SPOOL solution3
SET ECHO ON
SET FEEDBACK ON
SET LINESIZE 300
SET PAGESIZE 200
```

at the beginning of SQL script and

```
SPOOL OFF
```

at the end of SQL script.

Deliverables
A file `solution3.lst` with a report from processing of a script file `solution3.sql` that lists query processing plans before and after indexing. A report must have no errors and it must list all SQL statements processed.

Task 4 (6 marks)
Implement an anonymous PL/SQL block that lists in a format listed below the names of regions (attribute `R_NAME` in a relational table `REGION`), and the names of at most 3 nations (attribute `n_nation` in a relational table `NATION`) included in each region. The names of regions must be sorted in ascending order and the names of nations must be sorted in descending order. All name must be listed in uppercase letters.

```
REGION-NAME-1 : NATION-NAME-1 NATION-NAME-2 NATION-NAME-3
REGION-NAME-2 : NATION-NAME-1 NATION-NAME-2 NATION-NAME-3
REGION-NAME-3 : NATION-NAME-1 NATION-NAME-2 NATION-NAME-3
     ...          ...           ...           ...           ...
```

To list information retrieved from a sample database use PL/SQL package `DBMS_OUTPUT`. It is explained in the Cookbook, Recipe 7.1 How to start programming in PL/SQL how to use `DBMS_OUTPUT` package. Remember about `SET SERVEROUTPUT ON` at the beginning of a script file that contains your anonymous PL/SQL block.

Your implementation must use at least one cursor and at least one exception handler. In fact, such constraints make your implementation easier.

To test your solution put an implemented anonymous PL/SQL block into SQL script file `solution4.sql` and process the script.

Your report must include a listing of all PL/SQL statements processed. To achieve that put the following SQLcl commands:

```
SPOOL solution4
SET SERVEROUTPUT ON
SET ECHO ON
SET FEEDBACK ON
SET LINESIZE 200
SET PAGESIZE 400
SET SERVEROUTPUT ON
```

at the beginning of SQL script and

```
SPOOL OFF
```

at the end of SQL script.

Deliverables
A file `solution4.lst` with a report from testing of an anonymous PL/SQL block implemented in this task. A report must have no errors and it must list all PL/SQL and SQL statements processed.

Submission

Submit the files **`solution1.pdf`**, **`solution2.pdf`**, **`solution3.lst`**, and **`solution4.lst`** through Moodle in the following way:

(1) Access Moodle at **`http://moodle.uowplatform.edu.au/`**
(2) To login use a **`Login`** link located in the right upper corner the Web page or in the middle of the bottom of the Web page
(3) When logged select a site **`CSCI835/CSCI235 (S220) Database Systems`**
(4) Scroll down to a section **SUBMISSIONS**
(5) Click at a link **`In this place you can submit the outcomes of Assignment 1`**
(6) Click at a button **`Add Submission`**
(7) Move a file **`solution1.pdf`** into an area **`You can drag and drop files here to add them`**. You can also use a link **Add**...
(8) Repeat a step (7) for the files **`solution2.pdf`**, **`solution3.lst`**, and **`solution4.lst`**.
(9) Click at a button **`Save changes`**
(10) Click at a button **`Submit assignment`**
(11) Click at the checkbox with a text attached: **`By checking this box, I confirm that this submission is my own work,`** ... in order to confirm the authorship of your submission.
(12) Click at a button **`Continue`**

End of specification