School of Computing and Information Technology
University of Wollongong

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

CSCI235/CSCI835 Database Systems
Laboratory 3
31 August 2020

Scope

This laboratory includes implementation of a stored procedure that enforces the consistency constraints and adds derived data and implementation of a stored function that simplifies SQL programming.

The outcomes of the laboratory work are due by **Saturday 12 September, 2020, 7.00 pm (sharp).**

Please read very carefully information listed below.

This laboratory contributes to 2% of the total evaluation in the subject.

A submission procedure is explained at the end of specification.

This laboratory consists of 2 tasks and specification of each task starts from a new page.

It is recommended to solve the problems before attending a laboratory class 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 errors scores no marks.

It is expected that all tasks included within **Laboratory 3** 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.

Prologue

Download the files `dbschema.bmp`, `dbcreate.sql`, `dbload.sql`, and `dbdrop.sql` included in a section SAMPLE DATABASES on Moodle. 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`. A conceptual schema of a sample database is included in a file `dbschema.bmp`.

No report is expected from the implementations of the actions listed in Prologue.

Tasks

Task 1 (1 mark)

Verification of consistency constraints and generation of derived data

Implement a stored PL/SQL procedure

```
INSERT_ITEM(order_key,part_key,supplier_key,quantity,price,
            discount,tax)
```

that enforces a consistency constraint listed below before inserting a row to a relational table `LINEITEM`.

(1) The same part manufactured by the same supplier cannot be included more than one time in the same order.

The procedure must also generate the values of missing attributes before insertion of a row.

(2) A value of an attribute `L_LINENUMBER` must be automatically generated.

(3) A value of attribute `L_SHIPDATE` must be tomorrow's date.

(4) A value of attribute `L_RECEIPTDATE` must be one week after shipment date.

(5) The values of the other attributes are up to you. A simple solution is to use `0` for the columns of type `NUMBER`, empty string `''` for `CHAR` or `VARCHAR` and today's date for `DATE`.

If a consistency constraint listed above validates well, then the procedure must make an insertion of a row permanent in a database. If a consistency constraint does not validate well, the procedure must not insert a row and it must display an error message.

When the implementation of a procedure `INSERT_ITEM` is completed, create SQL script `solution1.sql` that stores the procedure in a data dictionary and tests the procedure with `EXECUTE` statements. Testing must be performed in the following way.

(1) One successful insertion must be performed into an already existing order.
(2) One successful insertion must be performed into a new order, and it means that you must insert a new order into a relational table `ORDERS` first.
(3) One unsuccessful insertion must be performed into an already existing order.
(4) Use `SELECT` statement to display the successfully inserted rows.

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

```
SPOOL solution1
```

```
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 `solution1.lst` with a report from the implementation and testing of a stored procedure `INSERT_ITEM`. A report must have no errors and it must list all PL/SQL and SQL statements processed.

Task 4 (1 mark)
Simplification of SQL with a stored function

The following SELECT statement lists the names and addresses of customers who either submitted more than 20 orders or who submitted no orders at all.

```
SELECT CUSTOMER.C_CUSTKEY, COUNT(ORDERS.O_CUSTKEY) CNT
FROM CUSTOMER LEFT OUTER JOIN ORDERS
              ON CUSTOMER.C_CUSTKEY = ORDERS.O_CUSTKEY
GROUP BY CUSTOMER.C_CUSTKEY
HAVING COUNT(ORDERS.O_CUSTKEY) > 20 OR
       COUNT(ORDERS.O_CUSTKEY) = 0
ORDER BY CNT ASC;
```

Implement a stored PL/SQL function that can be used in SELECT statement such that the new SELECT statement retrieves the same information as the old one and it consists of the clauses SELECT, FROM, WHERE, and ORDER BY only. The new SELECT does not use GROUP BY and HAVING clauses. Additionally, the new SELECT statement does not use an operation of LEFT OUTER JOIN.

When ready, implement a script solution2.sql that performs the following sequence of actions.

(1) First, the script processes SELECT statement give above,
(2) Next, the script stores PL/SQL function in a data dictionary,
(3) Next, the script processes SELECT statement that uses the stored function to retrieve the same results as the original statement,
(4) Finally, the script drops the stored function.

If you have the temptations to use DBMS_OUTPUT with PUT_LINE method then it is a very bad idea and such idea will not bring you a good evaluation.

Process SQL script solution2.sql and save a report from processing in a file solution2.lst.

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

```
SPOOL solution2
SET ECHO ON
SET FEEDBACK ON
SET LINESIZE 100
SET PAGESIZE 200
SET SERVEROUTPUT ON
```

at the beginning of SQL script and

```
SPOOL OFF
```

at the end of SQL script.

Deliverables
A file `solution2.lst` with a report from processing of SQL script `solution2.sql`. A report must have no errors and it must list all PL/SQL and SQL statements processed.

Submission
Submit the files **`solution1.lst`** and **`solution2.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 Laboratory 3`**
(6) Click at a button **`Add Submission`**
(7) Move a file **`solution1.lst`** 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 a file **`solution2.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