School of Computing and Information Technology
University of Wollongong

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

CSCI835 Database Systems
Assignment 3
4 July 2020

Scope

This assignment includes the tasks related to transaction processing in database systems.

The outcomes of the assignment are due by **Saturday, 11 July, 2020, 9.00 pm (sharp).**

Please read very carefully information listed below.

This Assignment contributes to 13% of the total evaluation in a subject CSCI835 Database Systems.

A submission procedure is explained at the end of specification.

This assignment consists of 3 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 errors scores no marks.

It is expected that all tasks included within **Assignment 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 the laboratory classes. Plagiarism will result in a **FAIL** grade being recorded for the assessment task.

Prologue

Download the files `dbcreate.sql`, `dbload.sql`, and `dbdrop.sql` included in a section SAMPLE DATABASES on Moodle. A conceptual schema of a sample database is available in the same section on Moodle. It is recommended to familiarize yourself with the conceptual schema and implementations of relational tables.

SQL script `dbcreate.sql` creates a sample database. To drop a sample database, you can use a script `dbdrop.sql`. To load data into a sample database, process as script `dbcreate.sql`. It is strongly recommended to drop a sample database and to re-create it before implementation each task.

No report is expected from the actions performed in the Prologue.

Tasks

Task 1 (3 marks)

Conflict serialization, 2PL and timestamp ordering protocols

This task does not need any implementation. It is a pure "pen&paper" or rather "keyboard&pdf " exercise.

Consider the following history of processing of three database transactions `T1`, `T2`, and `T3`. The transactions interleaved their operations and no controller (scheduler) has been used to control the processing of the transactions. A transaction `T1` started first, followed by a transaction `T2` and finally a transaction `T3` started as the last one. Therefore, `timestamp(T1) < timestamp(T2) < timestamp(T3).`

T1	T2	T3
read(x)		
write(x,x+10)		
	read(u)	
		read(y)
		write(y,y+1)
	read(z)	
read(z)		
write(z,z+1)		
	write(y,z+2)	
		read(v)
		write(v,v+y)
write(v,v+z)		
	read(x)	

(1) Draw a conflict serialization graph for the concurrent processing of database transactions given above.

(2) Show a history of concurrent processing when the database transactions `T1`, `T2`, and `T3` interleave their operations in the same way as above and the concurrent processing is controlled by a scheduler implementing 2PL protocol. A sample processing of database transactions controlled by a scheduler implementing 2PL protocol can be found in the lectures slides. Apply the same two-dimensional visualisation of concurrent processing of database transactions as above for a presentation if your solution.

(3) Show a history of concurrent processing when the database transactions `T1`, `T2`, and `T3` interleave their operations in the same way as above and the concurrent processing is controlled by a scheduler implementing timestamp ordering protocol. A sample processing of database transactions controlled by a scheduler implementing 2PL protocol can be found in the lectures slides. Apply the same

two-dimensional presentation of concurrent database transactions as above for a presentation if your solution.

Deliverables
A file `solution1.pdf` with conflict serialization graph (step 1), a history of concurrent processing of database transactions when controlled by 2PL protocol (step 2), and a history of database transactions controlled by timestamp ordering protocol (step 3).

Task 2 (5 marks)
Deadlocks

Use SQL Developer to create **two separate connections** to your Oracle account. It is very important that you create **two connections** and do not operate on two tabs within the same connection.

Next, use both connections to connect to Oracle database server.

Assume that two SQL Developer connections simulate 2 different users concurrently changing the contents of a sample database.

The users plan to update the relational tables `EMPLOYEE` and `ORDERS`. All details of the updates are up to you.

Use the connections created in this task to process `UPDATE` statements on the relational tables `EMPLOYEE` and `ORDERS` such that the simulated concurrent processing is leading to a deadlock that involves both users/transactions.

You can simulate concurrent processing of database transactions through processing of SQL statement in the first connection, then processing SQL statement in the second connection, then returning to the first connection and processing SQL statement in the first connection, then processing SQL statement in the second connection and so on, and so on.

`UPDATE` statements can be processed in any order and as it has been already mentioned, all other details of `UPDATE` statements are up to you. Note, that the details and the orders of `UPDATE` statements may be used to evaluate originality of your solution.

When ready, i.e. when the simulated processing of two transactions ends up in a deadlock copy and paste the contents of both windows with SQL Developer connections and save one by one in a file `solution2.lst`. Do not forget to save an evidence that actually a deadlock has happened.

Deliverables
A file `solution2.lst` with a report from the simulated processing of two transactions that ends up in a deadlock.

Task 3 (5 marks)
`READ COMMITTED` versus `SERIALIZABLE` isolation levels

This task does not need any implementation. It is a pure "pen&paper" or rather "keyboard&pdf " exercise.

Assume that the structures and the contents of a sample database have been changed in the following way.

```
ALTER TABLE CUSTOMER ADD (TOTAL_ORDERS NUMBER(7));

UPDATE CUSTOMER
SET TOTAL_ORDERS = ( SELECT count(*)
                     FROM ORDERS
                     WHEREORDERS.CUSTOMER_CODE = CUSTOMER.CUSTOMER_CODE );
```

A database developer implemented the following stored PL/SQL procedure that can be used to create a new order.

```
CREATE OR REPLACE PROCEDURE INSERT_ORDER(oid IN NUMBER,
                                         ccode IN VARCHAR,
                                         eid IN NUMBER) IS

total   CUSTOMER.TOTAL_ORDERS%TYPE;

BEGIN
 INSERT INTO ORDERS VALUES(oid, ccode, eid, sysdate, sysdate + 5, sysdate + 1,
                           NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL);

 SELECT TOTAL_ORDERS
 INTO total
 FROM CUSTOMER
 WHERE CUSTOMER.CUSTOMER_CODE = ccode;

 total := total + 1;

 UPDATE CUSTOMER
 SET TOTAL_ORDERS = total
 WHERE CUSTOMER.CUSTOMER_CODE = ccode;

 COMMIT;

EXCEPTION
  WHEN OTHERS THEN
   DBMS_OUTPUT.PUT_LINE( SQLERRM );
   ROLLBACK;
END;
/
```

The procedure INSERT_ORDER can be used to insert information about a new order submitted by a customer. Assume that the procedure INSERT_ORDER can be processed concurrently by many different users.

Your task is to decided what isolation level the procedure INSERT_ORDER should be processed and to justify your decision.

You have the following two options: READ COMMITTED or SERIALIZABLE. I am sure that there is no need to explain why the procedure cannot be processed at READ ONLY isolation level ;).

If you decide that the procedure can be processed at READ COMMITTED level then as justification of your decision provide a proof that any concurrent processing of the procedure at READ COMMITTED level does not corrupt a sample database.

If you decide that the procedure can be processed at SERIALIZABLE level then as a justification provide a concurrent processing of the procedure at READ COMMITTED level that corrupts a database. You can apply two-dimensional visualisation of concurrent execution of database transactions as it has been already used in this assignment, please see a diagram in Task 1 or the contents of lecture slides related to transaction processing in database systems.

Deliverables
A file solution3.pdf with a decision what isolation level a stored procedure INSERT_ORDER should be processed at and with comprehensive justification of the decision. Note, that a decision without justification, i.e. "educated guess" scores no marks.

<u>Submission</u>
Submit the files **`solution1.pdf`**, **`solution2.lst`**, and **`solution3.pdf`** 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 (JIS20) Database Systems`**
(4) Scroll down to a section **SUBMISSIONS**
(5) Click at a link **`In this place you can submit the outcomes of Assignment 3`**
(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.lst`**, and **`solution3.pdf`**.
(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