COMP 2714 – RELATIONAL DATABASE SYSTEMS

Lab02 Exercise - Concurrency Control Experiment with Oracle

The purpose of this lab tutorial is to explore part of the concurrency control features in Oracle relating to SQL transactions and locking. By default, Oracle runs in Read Committed level for concurrency control.

Follow the instructions below, note what is happening, and record your observations. Write your answers, in point form to the questions in a notepad text file, named Lab02_<Lastname><Initial>.txt (e.g. Lab02_TangK.txt). Label your answers according to the following section numbers. **Submit** it by the end of the 2-hour lab period, or as instructed by your lab instructor.

Browse through the whole tutorial first before starting!

1. Preparation

1.1. Creating the two simulated users

- Copy the shortcut 'Run SQL Command Line' from its location folder.
- Open D:\work\2714 folder, right-click inside and paste the shortcut. Rename it to User1.
  This simulates your first user (User1). Change shortcut properties to 'Start in:' D:\Work\2714.
- Right-click inside D:\work\2714 foler and paste the shortcut a second time. Rename it to User2.
  This simulates your second user (User2). Change shortcut properties to 'Start in:' D:\Work\2714.

1.2. Connecting to Oracle

- Right-click **User1** and Run as administrator. Login as user system: **connect system**.
- Right-click **User2** and Run as administrator. Login as user system: **connect system**
- Resize the two connection windows and tile them (vertically or horizontally as you prefer) so that you can switch between them easily.

1.3. Creating the sample database tables

- We are simulating a simple banking database, with 2 types of accounts, and 2 customers.
- In User1 window, type and execute the following:

```
DROP TABLE Checking;
DROP TABLE Saving;

CREATE TABLE Checking
(acctNo   DECIMAL(3)   NOT NULL
,lName    VARCHAR(15)  NOT NULL
,curBal   DECIMAL(6,2) NOT NULL
);

CREATE TABLE Saving
(acctNo   DECIMAL(3)   NOT NULL
,lName    VARCHAR(15)  NOT NULL
,curBal   DECIMAL(6,2) NOT NULL
);

INSERT INTO Checking VALUES (1, 'Smith', 500.00);
INSERT INTO Checking VALUES (2, 'Jones', 300.00);
INSERT INTO Saving   VALUES (1, 'Smith', 100.00);
INSERT INTO Saving   VALUES (2, 'Jones', 200.00);

COMMIT;
```

- When completed, clear the User1 window:

```
SQL> HOST CLS
```

2. ROLLBACK of an uncommitted transaction

2.1. Finding out Smith's checking account balance; Smith has acctNo=1

- In User2's query window, execute the following:

```
SELECT *
FROM Checking
WHERE acctNo = 1;
```

What is Smith's checking account balance?

2.2. Withdrawing from Smith's checking account

- In User1's query window, execute the following.
  Note that the transaction is started but not ended.

```
UPDATE Checking
SET curBal = curBal - 100.00
WHERE acctNo = 1;

SELECT *
FROM Checking
WHERE acctNo = 1;
```

What is Smith's checking account balance?

2.3. Finding out Smith's checking account balance

- In User2 window, execute the same SELECT query

```
SELECT *
FROM Checking
WHERE acctNo = 1;
```

Do you get the same query result on Smith's checking account balance as from step 2.1 or 2.2 above? Record briefly your observations on User2. Why the seeming discrepancy?

Try to think about why you see what you see without asking your instructor!

You may want to complete the next step (step 2.4) first and then come back to this question.

2.4. Rolling back the uncommitted account balance update

- In User1's window execute the following:

```
ROLLBACK;

SELECT *
FROM Checking
WHERE acctNo = 1;
```

-
- In User2 window, execute the same SELECT query

```
SELECT *
FROM Checking
WHERE acctNo = 1;
```

What is Smith's checking account balance seen from User1? From User2?
What happens in User2 when you execute ROLLBACK in User1?

For User1, why is this balance different from that displayed in step 2.2 above? Try to briefly explain your observations (i.e. what is happening?)

3. COMMIT of a transaction

This part III is similar to part II above, except we complete the update transaction with a COMMIT statement to finalize the changes, instead of a ROLLBACK statement to undo any uncommitted changes.

3.1. Finding out Smith's checking account balance

- In User2's query window, execute the following:

```
SELECT *
FROM Checking
WHERE acctNo = 1;
```

What is Smith's checking account balance?

3.2. Withdrawing from Smith's checking account

- In User1's query window, execute the following.
  Note that the transaction is started but not ended.

```
UPDATE Checking
SET curBal = curBal - 100.00
WHERE acctNo = 1;

SELECT *
FROM Checking
WHERE acctNo = 1;
```

What is Smith's checking account balance?

3.3. Finding out Smith's checking account balance

- In User2 window, execute the same SELECT query

```
SELECT *
FROM Checking
WHERE acctNo = 1;
```

Do you get the same query result on Smith's checking account balance as from step 3.1 or 3.2 above? Record briefly your observations on User2. Why the seeming discrepancy?

Try to think about why you see what you see without asking your instructor!

You may want to complete the next step (step 3.4) first and then come back to this question.

3.4. Committing the account balance update

- In User1's window execute the following:

```
COMMIT;

SELECT *
FROM Checking
WHERE acctNo = 1;
```

-

- In User2 window, execute the same SELECT query

```
SELECT *
FROM Checking
WHERE acctNo = 1;
```

What is Smith's checking account balance seen from User1? From User2?
What happens in User2 when you execute COMMIT in User1?

For User2, why is the balance now the same as that displayed in step 3.2 above, but different from step 3.3? Try to briefly explain your observations (i.e. what is happening?)

4. Transaction LOCKING and DEADLOCK

This part explores how Oracle handles transaction locking and deadlocks.

Clear the query windows in both User1 and User2 before starting the following. If needed, rerun step 1.4 in part 1 to recreate the checking and saving tables and initial amounts.

4.1. Bank is applying interest to checking accounts

- In User1's query window, execute the following.
  Note that the transaction is started but not ended.

```
UPDATE Checking
SET curBal = curBal * 1.02
;
```

4.2. Jones is withdrawing money from checking account

- In User2's query window, execute the following.

```
UPDATE Checking
SET curBal = curBal - 100.00
WHERE acctNo = 2;
```

What is happening here in User2? Why?

- In User1's query window, execute the following.

```
COMMIT;
```

What is happening here in User2? Why?

4.3. Processing the check Smith wrote Jones - At same time, processing another check from Jones to Smith

- In User1's window execute the following:

```
UPDATE Checking
SET curBal = curBal - 100
WHERE acctNo = 1;
```

-

- In User2's window, execute the following

```
UPDATE Checking
SET curBal = curBal - 200
WHERE acctNo = 2;
```

Then in User1 execute the following

```
UPDATE Checking
SET curBal = curBal + 100
WHERE acctNo = 2;
```

and in User2 execute the following

```
UPDATE Checking
SET curBal = curBal + 200
WHERE acctNo = 1;
```

Record briefly your initial observations on both User1, and User 2.
What happened after a couple of seconds?

4.4. How Oracle handles a deadlock?

- In User1, retype and execute the following:

```
UPDATE Checking
SET curBal = curBal + 100
WHERE acctNo = 2;
```

What happened after a couple of seconds? What can you conclude on the way Oracle handles a deadlock?

4.5. Updating the checking accounts again

- **Make a suggestion on what the execution order should possibly be for both User1 and User2, in order to complete their respective tasks in section 4.3 above, without any undesirable side-effects?**
  Write out the order of the steps / statements for both users in a single timeline.