CS1555 Recitation 12

Objective: 1. To get started with JDBC and demonstrate transaction concurrency control on oracle.
2. To practice Views

PART 1:

Oracle supports multi-version concurrency control. Therefore, a normal select statement does not acquire a lock on the data item it reads and never reads dirty (uncommitted) data.

Transactions are characterized by the ACID properties:
o Atomicity
o Consistency
o Isolation
o Durability

Isolation is ensured by Concurrency Control, which synchronizes the execution of transactions to ensure each executes in an isolation manner. The default, more restricted isolation level is serializability in which transactions execute as if they executed serially, one after the other.

In general, SQL transactions can execute under different isolation levels.
o Serializable
o Repeatable reads
o Read committed
o Read uncommitted (execute like OS processes)

1. Typical synchronization is based on locks: (1) read or shared and (2) write or exclusive. How do these locks work?

2. Oracle supports multi-version concurrency control. What does multi-version mean?

SQL select reads the most recently committed version of a table/tuple. That is, a SQL select statement does not acquire a lock on the data item it reads and never reads dirty (uncommitted) data.

PART 2:

Before we start:

- In order to set the PATH and CLASSPATH environmental variables to point to JAVA and Oracle JDBC library, just do the following:
  ```
  source ~panos/1555/bash.env.class3
  ```

- Copy the following files to your working directory:
  - `cp ~panos/1555/recitation/rec8db.sql .`
  - `cp ~panos/1555/recitation/TranDemo1.java .`
  - `cp ~panos/1555/recitation/TranDemo2.java .`

- Open 3 terminal windows, ssh to class3 and set the environment variables.
  - In the first terminal, we'll be running sqlplus to keep track of what changes are happening to the database.
  - In the second terminal, we'll be running TranDemo1.java
  - In the third terminal, we'll be running TranDemo2.java

Example 0: Getting Started

- Edit the TranDemo1.java file and TranDemo2.java, change the username and password to your username and password that you use to login to Oracle.
- Compile the files using the command: `javac TranDemo1.java; javac TranDemo2.java`
- Execute rec8db.sql under sqlplus in the first terminal.
- Run the program using the command: `java TranDemo1 0`
- Now read the demo source file to learn how it works. Note in the file:
  - How to connect to the DB.
  - How to execute an SQL statement.
  - How to iterate through the results set.

Notes:

- To run any of the examples, pass as an argument the example number.
- We will start running 2 transactions concurrently by running TranDemo1 and TranDemo2.
  - Run TranDemo1 first and then run TranDemo2 while TranDemo1 is still running.
  - Notice in the source codes how to group SQL statements into one transaction, commit/rollback the transaction and how to set isolation level for the transaction.
  - The sleep (milliseconds) function is used to force the statements in both transactions to execute in the order we want.

Example 1: Multi-version Concurrency of Oracle

TranDemo1 (read committed)	TranDemo2 (read committed)
update class set max_num_students = 5 where classid = 1 sleep... **rollback**	SELECT * FROM class where classid = 1

Question: What is the max_num_students as read by TranDemo2?

Example 2: (Implicit) Unrepeatable Read Problem

TranDemo1 (read committed)	TranDemo2 (read committed)
select max_num_students, cur_num_students **from** class **where** classid = 1 **sleep...** **if**(cur_num_students <max_num_students) **update** class **set** cur_num_students = cur_num_students +1 **where** classid = 1 **else** **print** 'the class is full' **commit**	**select** max_num_students, cur_num_students **from** class where classid = 1 **sleep...** **if**(cur_num_students <max_num_students) **update** class **set** cur_num_students = cur_num_students +1 **where** classid = 1

	else **print** 'the class is full' **commit**

Question: What is the value of cur_num_students for class with classid = 1 ? Compare it to the max_num_classes.

Example 3: Serializable Isolation Level:

The same as Example 2, but each transaction has the isolation level of **serializable**
Before running Example 3, reset the database by rerunning rec8db.sql in the first terminal window.

Question: What is the value of cur_num_students for class with classid = 1 now? Do both transactions perform the update?

Example 4: Using for Update of

The same as Example 2, but each transaction uses the following statement to select max and current number of students:

```
SELECT max_num_students, cur_num_students
FROM class where classid = 1
for update of cur_num_students
```

Again before running Example 4, reset the database by rerunning rec8db.sql in the first terminal window.

Question: What is the value of cur_num_students for class with classid = 1 now? Do both transactions perform the update?

Example 5: Deadlock

TranDemo1 (read committed)	TranDemo2 (read committed)
update class **set** max_num_students = 10 **where** classid = ***1*** sleep...	**update** class **set** max_num_students = 20 **where** classid = ***2***
update class **set** max_num_students = 10 **where** classid = ***2*** **commit**	sleep...
	update class **set** max_num_students = 20 **where** classid = ***1*** **commit**

Question: What is the value of max_num_students ? Do both transactions perform their updates?

PART 3 (Optional Review):

Before we start:
- Copy and run the file creating the Student database using:
    host cp ~panos/1555/recitation/studentdb.sql studentdb.sql
    @ studentdb

1. Create a view called student_courses that lists the SIDs, student names, number of courses in the Course_taken table.

2. Create a materialized view called mv_student_courses that lists the SIDs, student names, number of courses in the Course_taken table.

3. Execute the following commands. Compare the query results and time used of the two select statements.

```
insert into course_taken
values('CS1555', '129','Fall 18', null);
commit;

--execute DBMS_MVIEW.REFRESH('mv_student_courses');
set timing on
select * from mv_student_courses;
set timing on
set timing off
select * from student_courses;
set timing off
commit;
```

4. Reset the database by running the studentdb.sql and recreate the views. Comment back the line beginning with “execute” in the above commands and execute the commands. Compare the query results of the two select statements.