FIT2094-FIT3171 Databases

Session 9 Tutorial Activities

Update, Delete and Transaction Management

FIT Database Teaching Team

Complete session 9 activities in week 5 listed below:

FIT2094-FIT3171 2021 Summer B

FIT2094-FIT3171 Databases

Author: FIT Database Teaching Team

Learning Objectives:

- understand the concept of a database transaction
- understand the properties that database transactions should exhibit for proper database operation
- understand basic transaction serialisation through locking.
- be able to change the content of records in a database according to a specification
- be able to remove records from database

Important

Remember before starting any lab activity which involves working with files, first use SQL Developer to pull from the FIT GitLab server so as to ensure your local files and the FIT GitLab server files are in sync. *During this activity, you will be creating a set of sql scripts which **MUST** be sent to the FIT GitLab server.*

9.1 Transactions Management

9.1.1 Lock Exercise - Table

Task 1
Given the following transaction sequence, complete the table by clearly indicating what locks are present at each of the indicated times (Time 0 to Time 12). Cell entries must have the form **S(Tn)** - for a shared lock by Tn, **X(Tn)** - for an exclusive lock by Tn, **Tn wait Tm** - for a wait of Tn due to Tm (where n and m are transaction numbers)

Time	Transaction	Activity	A	B	C	D
0	T1	READ A				
1	T2	READ B				
2	T3	READ C				
3	T1	READ D				
4	T4	READ A				
5	T4	READ B				
6	T4	UPDATE B				
7	T3	UPDATE C				
8	T2	READ A				
9	T1	UPDATE C				
10	T2	READ D				
11	T2	UPDATE D				
12	T3	READ A				

Task 2
Based on your answers in the above table, draw a wait for graph and check if a deadlock is present

9.1.2 Lock Exercise - Practice

In these exercises, you will examine the issues involved in updating shared data. You will need 2 connections to do this exercise.

Task 1

In SQL Developer, create a new connection to Oracle. Your current connection will be referenced as connection 1 and your new connection as connection 2.

Task 2

Create the CUSTBALANCE table shown below using connection 1. The table will have 2 attributes, cust_id and cust_bal. Both attribute data types are number. Insert two rows of data so that the table appears as below:

	CUST_ID	CUST_BAL
1	1	100
2	2	200

Task 3

Complete the following steps (maintain the order of the operations):

1. Connection 1 update the balance of customer 1 from 100 to 110 (without issuing a commit).
2. Connection 1 run a select statement to see whether the value of customer 1 has been updated
3. Connection 2 view the contents of the custbalance table (do you see the new value? - if not, why not?)
4. Connection 1 issue a commit command
5. Connection 2 view the contents of the custbalance table (do you notice any difference?)

Explain what is happening in the results of the above queries, in the context of atomic transactions.

Task 4

Complete the following steps, see what happens when two connections try concurrent updates of the table (keep the order of transactions the same as below):

1. Connection 1 update the balance of customer 2 from 200 to 150 (without issuing a commit).
2. Connection 2 tries to update the balance of customer 2 to 100 (what happens?)

Explain what is happening here. What should be done to allow the Connection 2 update to proceed?

Task 5

Complete the following steps, see what happens when two connections try to update different rows in the same table (keep the order of transactions the same as below):

1. Connection 1 update the balance of customer 2 to 175 (without issuing a commit).
2. Connection 2 tries to update the balance of customer 1 to 125 (what happens?)

How does this differ from the results of the transactions in Task 4? What does this tell you about the granularity of locking in Oracle? What must be done in order for the results of both updates to be visible to both users?

Task 6

Try and generate a deadlock between the two connections (hint: connection 2 will need to set up another table). Remember that a deadlock occurs when connection 1 holds a lock on table A and requests a lock on table B, but table B is locked by connection 2 who is also requesting a lock on table A.

9.2. Using UPDATE and DELETE

First, run your **session7_schema.sql** and **session7_insert.sql** (or download the sample solution from Moodle) to recreate and repopulate the student, unit and enrolment tables under your Oracle account.

Then create a new sql file and name the file as **session9_dml.sql** and save this file under Tut09 in your local repository. Write the required SQL statements for section 9.2.1 to 9.2.2 in this file.

Lastly, run the **session9_dml.sql** and save the output from this run as **session9_dml_output.txt**. To save the output, use of the inbuilt Oracle SPOOL command.

9.2.1 UPDATE

It is common for data to change in value across time. In a database, we use the SQL UPDATE statement to change the value of a cell or cells in a table.

The UPDATE statement consist of three main components:

- The name of the table where the data will be updated.
- The row or the set of rows where the value will be updated.
- The new value to replace the old value.

An example of an UPDATE statement for data in the database we have created by following the exercises in Tutorial 7 SQL Data Definition Language DDL is as follows:

```
UPDATE enrolment
SET enrol_mark = 60
WHERE stud_nbr = 11111111 AND
unit_code = 'FIT5132' AND
enrol_semester = '2' AND
enrol_year = 2014;
```

TASKS

1. Update the unit name of FIT9999 from 'FIT Last Unit' to 'place holder unit'.
2. Enter the mark and grade for the student with the student number of 11111113 for the unit code FIT5132 that the student enrolled in semester 2 of 2014. The mark is 75 and the grade is D.
3. The university has introduced a new grade classification scale. The new classifications are:
    1. 0 - 44 is N.
    2. 45 - 54 is P1.
    3. 55 - 64 is P2.
    4. 65 - 74 is C.
    5. 75 - 84 is D.
    6. 85 - 100 is HD.

    Change the database to reflect the new grade classification scale.

9.2.2 DELETE

The DELETE statement is used to remove data from the database.

It is important to consider the referential integrity issues when using a DELETE statement. In Oracle, a table can be created with a FOREIGN KEY ON DELETE clause which indicates what action should be taken when the parent of a child record is attempted to be deleted. The specified actions can be:

- CASCADE, or
- SET NULL

When the ON DELETE clause is not specified (the default), the action will be set to RESTRICT. RESTRICT means the delete of a row in a parent table (the table containing the PRIMARY KEY being referred to by a FOREIGN KEY) will not be permitted when there are related rows in the child table (the table with the FOREIGN KEY).

TASKS

1. A student with student number 11111114 has taken intermission in semester 2 2014, hence all the enrolments of this student for semester 2 2014 should be removed. Change the database to reflect this situation.
2. Assume that Wendy Wheat (student number 11111113) has withdrawn from the university. Remove her details from the database.
3. Add Wendy Wheat back to the database (use the INSERT statements you have created when completing module Tutorial 7 SQL Data Definition Language DDL).

   Change the FOREIGN KEY constraints definition for the STUDENT table so it will now include the ON DELETE clause to allow CASCADE delete. Hint: You need to use the ALTER TABLE statement to drop the FOREIGN KEY constraint first and then put it back using ALTER TABLE with the ADD CONSTRAINT clause. A brief description of using ALTER to drop a constraint is available here, the ADD CONSTRAINT syntax was covered in tutorial 7. For more details, you can check the SQL Reference Manual (available from Moodle) for the full syntax and a range of examples.

   Once you have changed the table, now, perform the deletion of the Wendy Wheat (student number 11111113) row in the STUDENT table. Examine the ENROLMENT table. What happens to the enrolment records of Wendy Wheat?

Important

You need to get into the habit of establishing this as a standard FIT2094-FIT3171 workflow - Pull at the start of your working session, work on the activities you wish to/are able to complete during this session, add all (stage), commit changes and then Push the changes back to the FIT GitLab server