CO527 Advanced Database Systems

Lab Number : **04**

Topic : **Transaction Processing**

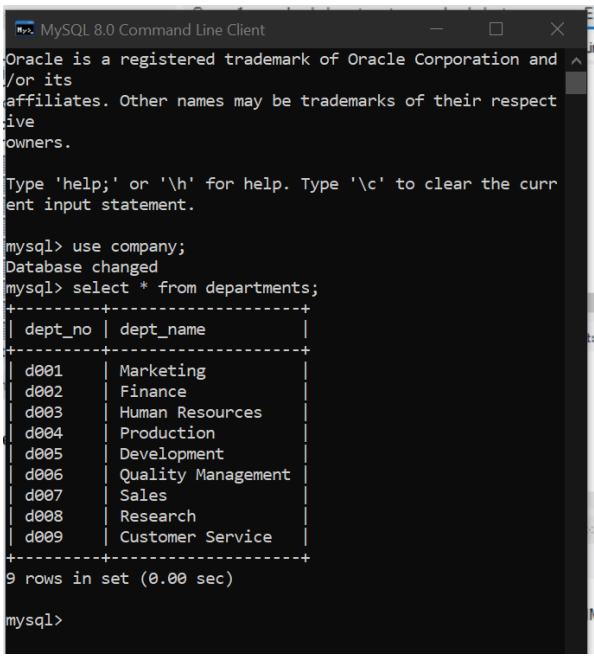
Name : **Dharmathilaka A.L.V.H.**

Registration Number : **E/16/086**

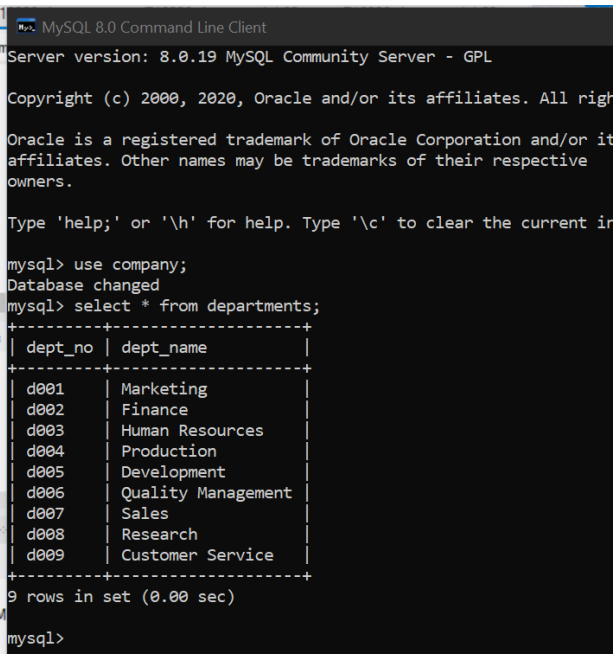
Concurrent Accesses

1. I of ACID

I. Issue a select query to view the current status of the departments table in both sessions.



Session I



Session II

II. Now, start transaction running start transaction in both sessions.

III. Insert a new row into the departments table from the 1st session and check if the changes are visible in the second session.

```
MySQL 8.0 Command Line Client
Query OK, 0 rows affected (0.00 sec)

mysql> insert into departments values
    -> ('d010','Computer Engineering');
Query OK, 1 row affected (0.01 sec)

mysql> select * from departments;
+---------+---------------------+
| dept_no | dept_name           |
+---------+---------------------+
| d001    | Marketing           |
| d002    | Finance             |
| d003    | Human Resources     |
| d004    | Production          |
| d005    | Development         |
| d006    | Quality Management  |
| d007    | Sales               |
| d008    | Research            |
| d009    | Customer Service    |
| d010    | Computer Engineering |
+---------+---------------------+
10 rows in set (0.00 sec)

mysql>
```

```
MySQL 8.0 Command Line Client
| d009    | Customer Service    |
+---------+--------------------+
9 rows in set (0.00 sec)

mysql> start transaction;
Query OK, 0 rows affected (0.00 sec)

mysql> select * from departments;
+---------+--------------------+
| dept_no | dept_name          |
+---------+--------------------+
| d001    | Marketing          |
| d002    | Finance            |
| d003    | Human Resources    |
| d004    | Production         |
| d005    | Development        |
| d006    | Quality Management |
| d007    | Sales              |
| d008    | Research           |
| d009    | Customer Service   |
+---------+--------------------+
9 rows in set (0.00 sec)

mysql>
```

Session I Session II

In both client sides new transactions are started. Session I client insert a one row. But it is not committed yet. So it is not visible in second session.

IV. Commit changes in the 1st command window and check if you can see the updates done at 1st window in 2nd command window.

```
mysql> commit;
Query OK, 0 rows affected (0.01 sec)

mysql> select * from departments;
+---------+---------------------+
| dept_no | dept_name           |
+---------+---------------------+
| d001    | Marketing           |
| d002    | Finance             |
| d003    | Human Resources     |
| d004    | Production          |
| d005    | Development         |
| d006    | Quality Management  |
| d007    | Sales               |
| d008    | Research            |
| d009    | Customer Service    |
| d010    | Computer Engineering |
+---------+---------------------+
10 rows in set (0.00 sec)
```

```
| d009    | Customer Service    |
+---------+--------------------+
9 rows in set (0.00 sec)

mysql> select * from departments;
+---------+--------------------+
| dept_no | dept_name          |
+---------+--------------------+
| d001    | Marketing          |
| d002    | Finance            |
| d003    | Human Resources    |
| d004    | Production         |
| d005    | Development        |
| d006    | Quality Management |
| d007    | Sales              |
| d008    | Research           |
| d009    | Customer Service   |
+---------+--------------------+
9 rows in set (0.00 sec)
```

Session I Session II

After committing in session I also, the changes are not reflected in session II as in above figure. It happens because the second session also in a transaction process and to get further updates it should finish the current session.

V. Explain your observations before and after running the commit in the 1st window.

Before the commit in 1st window:

The updates were not visible in the 2nd window since the updates done at the 1st window were not permanently saved in the database as the transaction was not committed.

After the commit in 1st window:

The updates were not visible in the 2nd command window since it is in the middle of the transaction. Once we committed the ongoing transaction in session 02 and start a new transaction then we must be able to see the updates done at session I.

2. Concurrent Updates

I. Try to do a concurrent update to the same row in departments table during two transactions

```
mysql> start transaction;
Query OK, 0 rows affected (0.00 sec)

mysql> update departments
    -> set dept_name = "Chemical and Process Engineering"
    ->  where dept_no = "d010";
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from departments;
+---------+--------------------------------+
| dept_no | dept_name                      |
+---------+--------------------------------+
| d001    | Marketing                      |
| d002    | Finance                        |
| d003    | Human Resources                |
| d004    | Production                     |
| d005    | Development                    |
| d006    | Quality Management             |
| d007    | Sales                          |
| d008    | Research                       |
| d009    | Customer Service               |
| d010    | Chemical and Process Engineering |
+---------+--------------------------------+
```

Session I

```
mysql> start transaction;
Query OK, 0 rows affected (0.00 sec)

mysql> update departments
    -> set dept_name = "Mechanical Engineering"
    -> where dept_no = "d010";
ERROR 1205 (HY000): Lock wait timeout exceeded; try restarting transaction
mysql>
```

Session II

Not allowed to updates in session 02, before committing the transaction in session 01. Tried updates:

session 01

update departments
set dept_name = "Chemical and Process Engineering"
where dept_no = "d010";

Done this update.

session 02

update departments
set dept_name = "Mechanical Engineering"
where dept_no = "d010";

Unable to complete.

II. Explain what happens before ending any of the transactions.

When session I doing the transaction, it locks the attribute that they are updating. So that any other transaction cannot use that for update/delete. As a result, session II cannot do the update while session I doing the update. Once the transaction finish in session I and do the commit, the attribute can be used by the transaction in session II for update query.

III. What happens when you commit your changes in the 1st session?

```
mysql> commit;
Query OK, 0 rows affected (0.02 sec)

mysql> select * from departments;
+---------+---------------------------------+
| dept_no | dept_name                       |
+---------+---------------------------------+
| d001    | Marketing                       |
| d002    | Finance                         |
| d003    | Human Resources                 |
| d004    | Production                      |
| d005    | Development                     |
| d006    | Quality Management              |
| d007    | Sales                           |
| d008    | Research                        |
| d009    | Customer Service                |
| d010    | Chemical and Process Engineering |
+---------+---------------------------------+
10 rows in set (0.00 sec)

mysql>
```

Session I has committed the transaction. So the corresponding field is updated.

```
mysql> update departments
    -> set dept_name = "Mechanical Engineering"
    -> where dept_no = "d010";
ERROR 1205 (HY000): Lock wait timeout exceeded; try restarting transaction
mysql> update departments
    -> set dept_name = "Mechanical Engineering"
    ->  where dept_no = "d010";
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from departments;
+---------+-----------------------+
| dept_no | dept_name             |
+---------+-----------------------+
| d001    | Marketing             |
| d002    | Finance               |
| d003    | Human Resources       |
| d004    | Production            |
| d005    | Development           |
| d006    | Quality Management    |
| d007    | Sales                 |
| d008    | Research              |
| d009    | Customer Service      |
| d010    | Mechanical Engineering |
+---------+-----------------------+
10 rows in set (0.00 sec)

mysql>
```

Since session II committed, transaction can done in session II, but second transaction is not aware about the first transaction. So lost update problem can be occurred.

Use your imagination and words to write a scenario where using transactions is essential and then create the required tables and test how the transaction will effect your tables,

1. during the transaction execution.

2. after rollback statement.

3. after the commit statement.

4. during 2 concurrent transactions, both of them update a record and both of them commit it.

Scenario:

Company has several stationary shops that sell products. All shops(shop A, shop B, shop C…) connected to same Database. Company add products, change prices according to the demand. At the same time shops can add the products that they have taken from outside to the shop. These shops are popular among students. So many transactions happen at time.

01. Update name of the pencil by company;

```
mysql> start transaction;
Query OK, 0 rows affected (0.00 sec)

mysql> update products set PName="Pencil B1" where ProductID=003;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

How shop A see is in below figure. It did not change the name since company did not commit the change.

```
mysql> start transaction;
Query OK, 0 rows affected (0.00 sec)

mysql> select * from products where ProductId=003
    -> ;
+-----------+--------+-------+
| ProductID | PName  | Price |
+-----------+--------+-------+
|         3 | Pencil |    30 |
+-----------+--------+-------+
1 row in set (0.00 sec)
```

02. Rollback

```
mysql> select * from products where productID=003;
+-----------+-----------+-------+
| ProductID | PName     | Price |
+-----------+-----------+-------+
|         3 | Pencil B1 |    30 |
+-----------+-----------+-------+
1 row in set (0.00 sec)

mysql> rollback;
Query OK, 0 rows affected (0.01 sec)

mysql> select * from products where productID=003;
+-----------+--------+-------+
| ProductID | PName  | Price |
+-----------+--------+-------+
|         3 | Pencil |    30 |
+-----------+--------+-------+
1 row in set (0.00 sec)
```

In company after updating the product name it is visible but after rollback it is go to the previous state. It is similar to the result in transaction at shop A.

```
+-----------+--------+-------+
| ProductID | PName  | Price |
+-----------+--------+-------+
|         3 | Pencil |    30 |
+-----------+--------+-------+
1 row in set (0.00 sec)

mysql> select * from products where ProductId=003
    -> ;
+-----------+--------+-------+
| ProductID | PName  | Price |
+-----------+--------+-------+
|         3 | Pencil |    30 |
+-----------+--------+-------+
1 row in set (0.00 sec)
```

03. commit

Company add a new item “Eraser” to the table.

```
mysql> start transaction;
Query OK, 0 rows affected (0.00 sec)

mysql> insert into products values (002,"Eraser B",100);
ERROR 1062 (23000): Duplicate entry '2' for key 'products.PRIMARY'
mysql> insert into products values (005,"Eraser B",100);
Query OK, 1 row affected (0.00 sec)
```

In shop A transaction state it shows only old products. Eraser is not added as company did not commit changes.

```
mysql> select * from products
    -> ;
+-----------+---------+-------+
| ProductID | PName   | Price |
+-----------+---------+-------+
|         1 | Pen     |    30 |
|         2 | Book    |   250 |
|         3 | Pencil  |    30 |
|         4 | A4sheet |     5 |
+-----------+---------+-------+
4 rows in set (0.00 sec)
```

Company commit changes. But still shopA did not get the changes since it is in another transaction.

```
mysql> start transaction;
Query OK, 0 rows affected (0.00 sec)

mysql> insert into products values (002,"Eraser B",100);
ERROR 1062 (23000): Duplicate entry '2' for key 'products.PRIMARY'
mysql> insert into products values (005,"Eraser B",100);
Query OK, 1 row affected (0.00 sec)

mysql> commit;
Query OK, 0 rows affected (0.02 sec)
```

```
mysql> select * from products;
+-----------+----------+-------+
| ProductID | PName    | Price |
+-----------+----------+-------+
|         1 | Pen      |    30 |
|         2 | Book     |   250 |
|         3 | Pencil   |    30 |
|         4 | A4sheet  |     5 |
|         5 | Eraser B |   100 |
+-----------+----------+-------+
5 rows in set (0.00 sec)
```

Company after commit.

```
mysql> select * from products
    -> ;
+-----------+---------+-------+
| ProductID | PName   | Price |
+-----------+---------+-------+
|         1 | Pen     |    30 |
|         2 | Book    |   250 |
|         3 | Pencil  |    30 |
|         4 | A4sheet |     5 |
+-----------+---------+-------+
4 rows in set (0.00 sec)
```

Shop A

After shop A commits and new transaction started, changes are taken.

```
mysql> start transaction;
Query OK, 0 rows affected (0.00 sec)

mysql> select * from products;
+-----------+----------+-------+
| ProductID | PName    | Price |
+-----------+----------+-------+
|         1 | Pen      |    30 |
|         2 | Book     |   250 |
|         3 | Pencil   |    30 |
|         4 | A4sheet  |     5 |
|         5 | Eraser B |   100 |
+-----------+----------+-------+
5 rows in set (0.00 sec)
```

04. during 2 concurrent transactions, both of them update a record and both of them commit it.

Generally if same record is updated by two transactions, lost update problem may occurred. Let's check in clients.

Two transactions started. Company changed the pencil name to "Pencil H2"

```
mysql> start transaction;
Query OK, 0 rows affected (0.00 sec)

mysql> update products set PName="Pencil H2" where ProductID=003;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from products;
+-----------+-----------+-------+
| ProductID | PName     | Price |
+-----------+-----------+-------+
|         1 | Pen       |    30 |
|         2 | Book      |   250 |
|         3 | Pencil H2 |    30 |
|         4 | A4sheet   |     5 |
|         5 | Eraser B  |   100 |
+-----------+-----------+-------+
5 rows in set (0.00 sec)
```

At the same time shop A also try to update the same row. Pencil to "Pencil HBB". It makes error.

```
mysql> start transaction;
Query OK, 0 rows affected (0.00 sec)

mysql> update products set PName="Pencil HBB" where ProductID=003;
ERROR 1205 (HY000): Lock wait timeout exceeded; try restarting transaction
```

Now company commit changes;

```
mysql> select * from products;
+-----------+-----------+-------+
| ProductID | PName     | Price |
+-----------+-----------+-------+
|         1 | Pen       |    30 |
|         2 | Book      |   250 |
|         3 | Pencil H2 |    30 |
|         4 | A4sheet   |     5 |
|         5 | Eraser B  |   100 |
+-----------+-----------+-------+
5 rows in set (0.00 sec)

mysql> commit;
Query OK, 0 rows affected (0.01 sec)
```

Shop A see the changes now. Also it can do the update.

```
mysql> select * from products;
+-----------+-----------+-------+
| ProductID | PName     | Price |
+-----------+-----------+-------+
|         1 | Pen       |    30 |
|         2 | Book      |   250 |
|         3 | Pencil H2 |    30 |
|         4 | A4sheet   |     5 |
|         5 | Eraser B  |   100 |
+-----------+-----------+-------+
5 rows in set (0.00 sec)

mysql> update products set PName="Pencil HBB" where ProductID=003;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from products;
+-----------+-----------+-------+
| ProductID | PName     | Price |
+-----------+-----------+-------+
|         1 | Pen       |    30 |
|         2 | Book      |   250 |
|         3 | Pencil HBB |    30 |
|         4 | A4sheet   |     5 |
|         5 | Eraser B  |   100 |
+-----------+-----------+-------+
5 rows in set (0.00 sec)
```

Summary: In two transactions, same tuple cannot modify concurrently.