

## Experiment no: - 5

**Experiment Name:** - DML and Transaction control statements.

**Aim:** - Performing practical by using DML statements and control transaction.

**Resource required:** - Oracle 9i - iSQLplus

**Theory:** -

- **DATA MANIPULATION LANGUAGE:**

- **DML statements:**

A DML statement is executed when:

- Add new rows to a table
- Modify existing rows in a table
- Remove existing rows from a table

A transaction consists of a collection of DML statements that form a logical unit of work.

**INSERT Statement:** add new rows to a table

**Syntax:** INSERT INTO table [ (column [, column....])]   
VALUES (value [, value...]);

1. Insert one – row at a time

e.g.: INSERT INTO departments (department\_id, department\_name,   
manager\_id, location-id)   
VALUES (70, „Public Relations“, 100, 1700);

2. Inserting rows with Null values

e.g.: INSERT INTO departments   
VALUES (100, „Finance“, NULL, NULL);

3. Inserting specific Date values

e.g.: INSERT INTO employees   
VALUES (114, „Den“, „Raphealy“, „DRAPHEAL“, „515.127.4561“, TO\_DATE   
(„FEB 3, 1999“, „MON DD, YYYY“)“   
„AC\_ACCOUNT“, 11000, NULL, 100, 30);

**UPDATE Statement:** modify existing rows

**Syntax:** UPDATE table   
SET column = value [, column = value ...]   
[WHERE condition];

1. Updating rows in a table

e.g.: UPDATE employees   
SET department\_id = 70   
WHERE employee\_id = 113;

2. Updating rows based on another table: use subqueries

e.g.: UPDATE copy\_emp  
SET department\_id = (SELECT department\_id  
FROM employees  
WHERE employee\_id = 100)  
WHERE job\_id = (SELECT job\_id  
FROM employees  
WHERE employee\_id = 200);

**DELETE Statement:** remove existing rows from a table

**Syntax:** DELETE [FROM] table  
[WHERE condition];

1. Deleting rows from a table

e.g.: DELETE FROM departments  
WHERE department\_name = „Finance“;

2. Deleting rows based on another table

e.g.: DELETE FROM employees  
WHERE department\_id =  
(SELECT department-id  
FROM departments  
WHERE department\_name LIKE „%Public %“);

**MERGE Statement:** ability to insert and update rows in a table

**Syntax:** MERGE INTO table\_name table\_alias  
USING (table/ view/ sub\_query) alias  
ON (join condition)  
WHEN MATCHED THEN  
UPDATE SET  
Col1 = col\_val1,  
Col2 = col2\_val  
WHEN NOT MATCHED THEN  
INSERT (column\_list)  
VALUES (column\_values);

e.g.: MERGE INTO copy\_emp c  
USING employees e  
ON (c.employee-id = e.employee\_id)  
WHEN MATCHED THEN  
UPDATE SET  
...  
WHEN NOT MATCHED THEN  
INSERT VALUES.....;

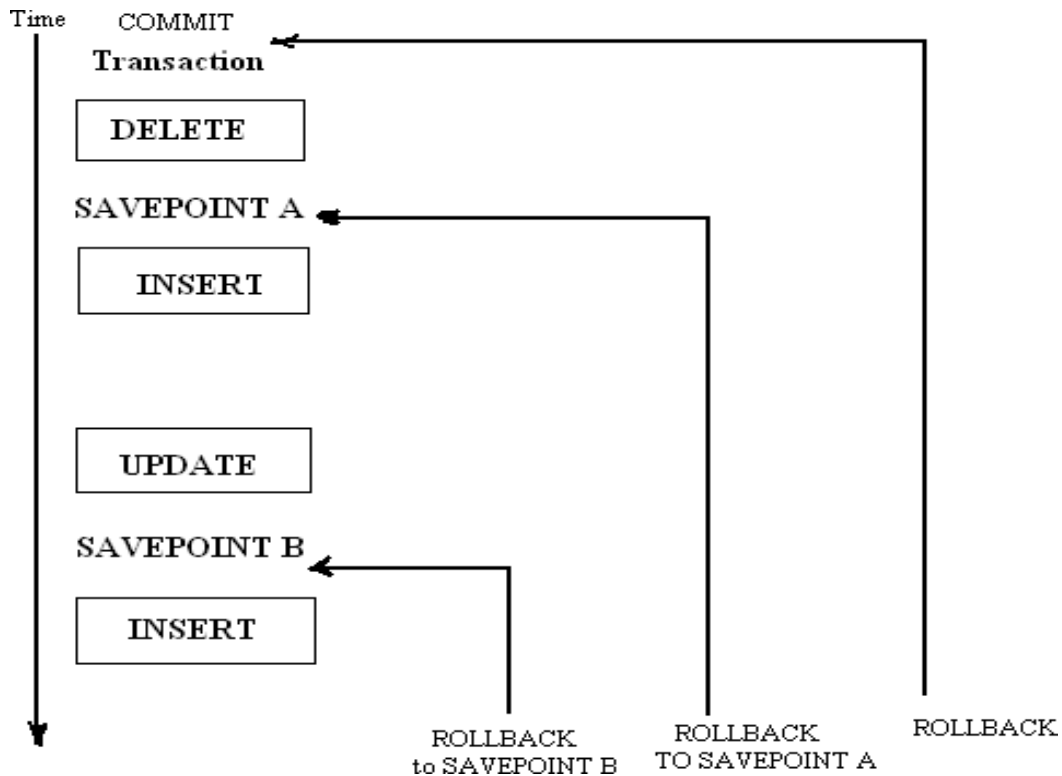
- **Database Transactions:**

Consists of one of the following:

- DML statement with one consistent change to the data
- One DDL statement
- One DCL statement

Begin when the first DML SQL statement is executed and end with a COMMIT or ROLLBACK statement is issued.

### **Controlling Transactions:**



Control the logic of a transaction by using COMMIT, ROLLBACK, and SAVEPOINT statements.

**COMMIT:** Ends the current transaction by making all pending data changes

**SAVEPOINT:** Marks a savepoint within the current transaction

**ROLLBACK:** ends the current transaction by discarding all pending data changes.

### **State of the data before COMMIT or ROLLBACK:**

- The previous state of data can be recovered.
- The current user review the result of DML operations by using the SELECT statement
- Other user can't views the result of the DML statements by the current user.
- The affected rows are locked: other user can't change the data within affected rows.

**Sate of data after COMMIT;**

- Data change may be permanent in the database.
- The previous sate of the data is permanently lost
- All users can view the results.
- Locks on affected rows are released
- All savepoints are erased.

**Committing Data:**

- make the changes

e.g.: DELETE FROM employees  
WHERE employee\_id = 99999;

INSERT INTO departments  
VALUES (290, „Corporate Tax“, NULL, 1700);

- Commit the changes

e.g.: COMMIT;  
Commit complete.

**Sate of data after ROLLBACK:**

- Discard all pending changes by using the ROLLBACK statement.

e.g.: DELETE FROM copy\_emp;  
22 rows deleted  
ROLLBACK;  
Rollback complete.

**LOCKING:**

In an Oracle database, locks:

- Prevent destructive interaction between concurrent transactions
- Require no user action
- Automatically use the lowest level of restrictiveness
- Are held for the duration of the transaction
- Two types of locking: explicit locking and implicit locking
- Locks held until COMMIT and ROLLBACK.

**Conclusion:**

In this practical, learned how to use DML statements and control transactions.

## LAB ASSIGNMENT -5

Roll. No. B74	Name: Akash Kinage
Class: SE-B	Batch: B4
Date of Experiment: 11-03-2022	Date of Submission: 13-03-2022
Grade:	

1. Add the first row of data to the MY\_EMPLOYEE table from the following sample Data. Do not list the column in the INSERT clause.

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	Patel	Ralph	Rpatel	895
2	Dance	Betty	Bdancs	860
3	Biri	Ben	Bbiri	1100
4	Newman	Chad	Cnewman	750
5	Ropeburn	Audrey	Aropebur	1550

### Query:

```
CREATE TABLE MY_EMPLOYEE (  
  ID number(10),  
  Last_name varchar(40),  
  First_name varchar(40),  
  Userid varchar(10),  
  Salary number(10)  
);
```

```
INSERT INTO MY_EMPLOYEE VALUES (1, 'Patel', 'Ralph', 'rpatel', 895);  
select * from MY_EMPLOYEE
```

### Output:

The screenshot shows the Oracle Live SQL interface. The SQL Worksheet contains the following queries:

```
1 CREATE TABLE MY_EMPLOYEE (  
2   ID number(10),  
3   Last_name varchar(40),  
4   First_name varchar(40),  
5   Userid varchar(10),  
6   Salary number(10)  
7 );  
8  
9 INSERT INTO MY_EMPLOYEE VALUES (1, 'Patel', 'Ralph', 'rpatel', 895);  
10  
11 select * from MY_EMPLOYEE;  
12
```

The output shows "Table created." and "1 row(s) inserted." Below this, a table displays the data:

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	Patel	Ralph	rpatel	895

At the bottom, there is a "Download CSV" link. The footer of the interface includes copyright information for Oracle and mentions "Built with using Oracle APEX - Privacy - Terms of Use".

2. Populate the MY\_EMPLOYEE table with the second row of sample data from the preceding list. This time, list the column explicitly in the INSERT clause.

### Query:

```
INSERT INTO MY_EMPLOYEE (id, last_name, first_name, userid, salary)
VALUES (2, 'Dancs', 'Betty', 'bdancs', 860);
```

### Output:

The screenshot shows the Oracle Live SQL interface. The SQL Worksheet contains the following code:

```
1 INSERT INTO MY_EMPLOYEE (id, last_name, first_name, userid, salary)
2 VALUES (2, 'Dancs', 'Betty', 'bdancs', 860);
3
4 select * from MY_EMPLOYEE
5
```

The output shows "1 row(s) inserted." and a table with the following data:

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
2	Dancs	Betty	bdancs	860
1	Patel	Ralph	rpatel	895

Below the table, it says "Download CSV" and "2 rows selected."

At the bottom, the footer indicates: "© 2022 Oracle - Live SQL 22.1.2, running Oracle Database 19c Enterprise Edition - 19.8.0.0.0 - Database Documentation - Ask Tom - Dev Gym. Built with using Oracle APEX - Privacy - Terms of Use."

3. Change the name of employee 3 to Drexler.

### Query:

```
INSERT INTO MY_EMPLOYEE
VALUES (3, 'Biri', 'Ben ', 'Bbiri', 1100);
```

```
UPDATE MY_EMPLOYEE
SET first_name = 'Drexler'
WHERE id = 3;
```

```
select * from MY_EMPLOYEE
```

## Output:

The screenshot shows the Oracle Live SQL interface. The SQL Worksheet contains the following code:

```
1 INSERT INTO MY_EMPLOYEE
2 VALUES (3, 'Biri', 'Ben ', 'Bbiri', 1100);
3
4 UPDATE MY_EMPLOYEE
5 SET first_name = 'Drexler'
6 WHERE id = 3;
7
8 select * from MY_EMPLOYEE
9
```

The execution results show 1 row(s) updated. Below the message is a table with 5 columns: ID, LAST\_NAME, FIRST\_NAME, USERID, and SALARY.

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
2	Dancs	Betty	bdancs	860
3	Biri	Drexler	Bbiri	1100
1	Patel	Ralph	rpate1	895

Below the table, it says "Download CSV" and "3 rows selected."

4. Change the salary to 1000 for all employee with a salary less than 900.

## Query:

```
UPDATE my_employeee
SET salary = 1000
WHERE salary < 900;
```

## Output:

The screenshot shows the Oracle Live SQL interface. The SQL Worksheet contains the following code:

```
1 UPDATE MY_EMPLOYEE
2 SET salary = 1000
3 WHERE salary < 900;
4
5
6 select * from MY_EMPLOYEE
7
```

The execution results show 2 row(s) updated. Below the message is a table with 5 columns: ID, LAST\_NAME, FIRST\_NAME, USERID, and SALARY.

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
2	Dancs	Betty	bdancs	1000
3	Biri	Drexler	Bbiri	1100
1	Patel	Ralph	rpate1	1000

Below the table, it says "Download CSV" and "3 rows selected."

5. Confirm your addition to the table.

### Query:

select \* from MY\_EMPLOYEE;

### Output:

The screenshot shows the Oracle Live SQL interface. The SQL Worksheet contains the query: `select * from MY_EMPLOYEE`. The output displays a table with 3 rows selected:

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
2	Dancs	Betty	bdancs	1000
3	Biri	Drexler	Bbiri	1100
1	Patel	Ralph	rpate1	1000

Below the table, it says "Download CSV" and "3 rows selected." The footer indicates the environment is Oracle Live SQL 22.1.2 running Oracle Database 19c Enterprise Edition.

6. Commit all pending changes. Control data transaction to the MY\_EMPLOYEE table.

### Query:

Commit;

### Output:

The screenshot shows the Oracle Live SQL interface. The SQL Worksheet contains the query: `commit;`. The output displays the message: "Statement processed." The footer indicates the environment is Oracle Live SQL 22.1.2 running Oracle Database 19c Enterprise Edition.

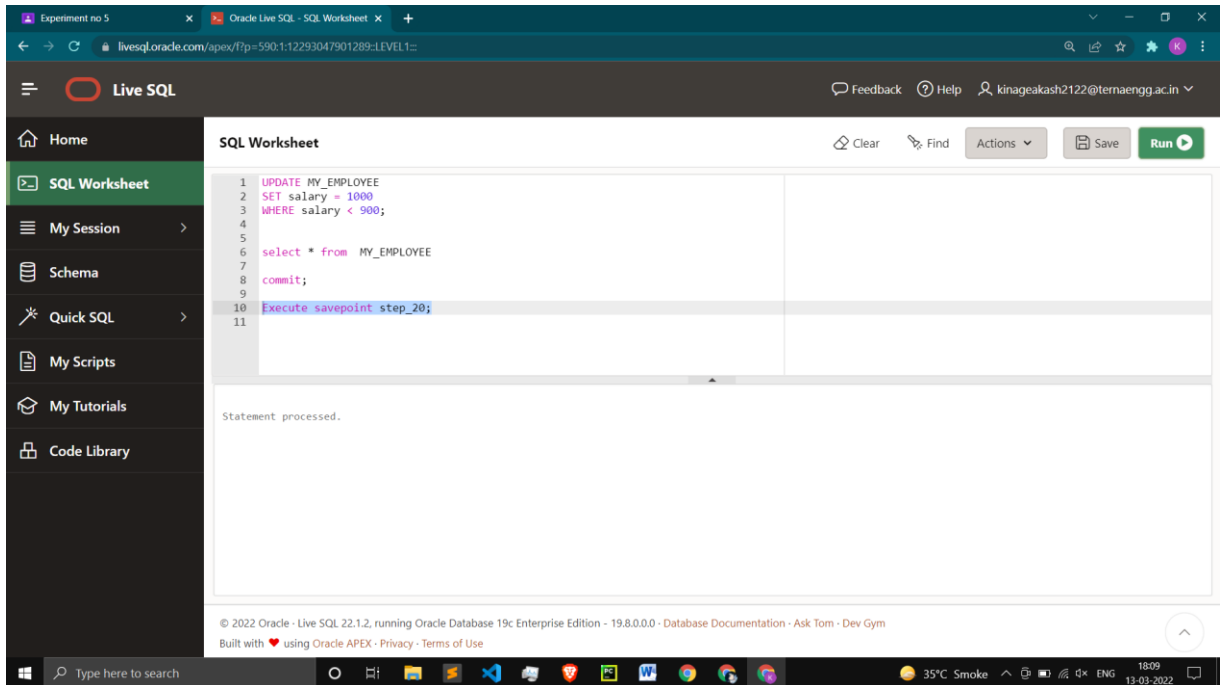


- Mark an intermediate point in the processing of the transaction.

**Query:**

Execute savepoint step\_20;

**Output:**

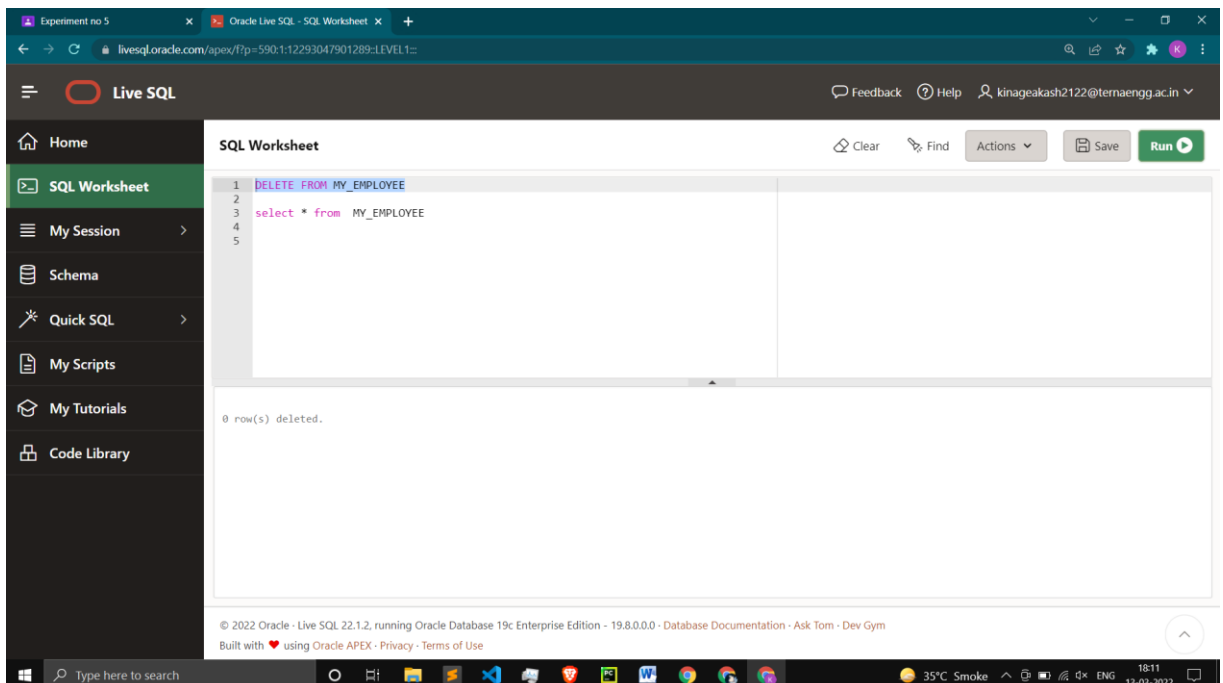


- Empty the entire table.

**Query:**

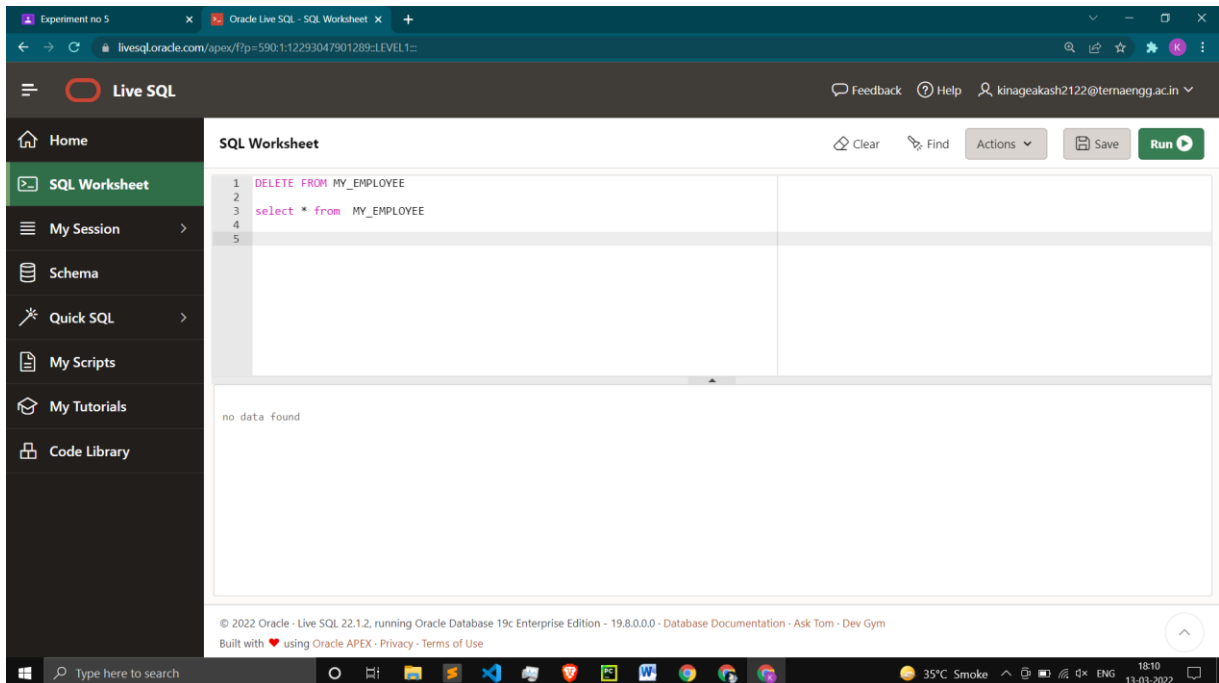
DELETE FROM MY\_EMPLOYEE

**Output:**



9. Confirm that table is empty.

### Output:



10. Discard the recent DELETE operation without discarding the earlier INSET Operation.

### Query:

Rollback;

### Output:

