Database Systems LAB # 10

TCL, Nested Queries and Set Operators in RDBMS

What is a Transaction?

A transaction is a set of SQL statements which Oracle treats as a Single Unit. I.e. all the statements should execute successfully or none of the statements should execute.

To control transactions Oracle does not made permanent any DML statements unless you commit it. If you don't commit the transaction and power goes off or system crashes then the transaction is roll backed.

TCL Statements available in Oracle are

COMMIT: Make changes done in transaction permanent.
ROLLBACK: Rollbacks the state of database to the last commit point.
SAVEPOINT: Use to specify a point in transaction to which later you can rollback.

1. COMMIT

To make the changes done in a transaction permanent issue the COMMIT statement.

The syntax of COMMIT Statement is

Commit;

2. ROLLBACK

To rollback the changes done in a transaction give rollback statement. Rollback restore the state of the database to the last commit point.

Example :

```
delete from emp;

rollback;        /* undo the changes */
```

3. SAVEPOINT

Specify a point in a transaction to which later you can roll back.

Example

```
insert into emp (empno,ename,sal) values (109,'Sami',3000);
savepoint a;
insert into dept values (10,'Sales','Hyd');
savepoint b;
insert into salgrade values ('III',9000,12000);
```

Now if you give

```
rollback to a;
```

Then row from salgrade table and dept will be roll backed. At this point you can commit the row inserted into emp table or rollback the transaction.

If you give

```
rollback to b;
```

Then row inserted into salgrade table will be roll backed. At this point you can commit the row inserted into dept table and emp table or rollback to savepoint a or completely roll backed the transaction.

If you give

```
rollback;
```

Then the whole transactions is roll backed.

If you give

```
commit;
```

Then the whole transaction is committed and all savepoints are removed.

Nested Queries in RDBMS.

Nested Queries:

Nesting of queries one within another is known as a nested queries.

Subqueries:

The query within another is known as a subquery. A statement containing subquery is called parent statement. The rows returned by subquery are used by the parent statement.

Example: *select ename, eno, address where salary > (select salary from employee where ename ='jones');*

Types:

1. Subqueries that return several values

Subqueries can also return more than one value. Such results should be made use along with the operators in and any.

Example: *select ename, eno, from employee where salary <any (select salary from employee where deptno =10');*

2. Multiple queries

Here more than one subquery is used. These multiple Subqueries are combined by means of 'and' & 'or' keywords.

Example: *Select * from test where job= (select job from test where empno=2) or Empno= (select empno from test where job='Ins');*

3. Correlated subquery

A subquery is evaluated once for the entire parent statement whereas a correlated subquery is evaluated once per row processed by the parent statement.

Example: *select * from emp x where x.salary > (select avg(salary) from emp where deptno =x.deptno);*

Above query selects the employees details from emp table such that the salary of employee is > the average salary of his own department.

1. Set operators

The Set operator combines the result of 2 queries into a single result. The following are the operators:

- Union
- Union all
- Intersect
- Minus

The rules to which the set operators are strictly adhere to:

- The queries which are related by the set operators should have a same number of column and column definition.
- Labels under which the result is displayed are those from the first select statement.

1. Union:

Returns all distinct rows selected by both the queries.

Syntax:

Query1 Union Query2;

Example:

select empno from emp union select empno from test;

2. Union all:

Returns all rows selected by either query including the duplicates.

Syntax:

Query1 Union all Query2;

Example:

select empno from emp union all select empno from test;

3. Intersect:

Returns rows selected that are common to both queries.

Syntax:

Query1 Intersect Query2;

Example:

select empno from emp intersect select empno from test;

4. Minus:

Returns all distinct rows selected by the first query and are not by the second

Syntax:

Query1 minus Query2;

Example:

select empno from emp minus select empno from test;