DML STATEMENTS

Data Manipulation Language is use to modify a table in several different ways. Any changes we make to a table or other object in our schema are done with DML.

We can INSERT rows, UPDATE rows, DELETE rows, or use Transaction Control statements such as: COMMIT, SAVEPOINT, and ROLLBACK.

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

Inserting a NEW Row

Before we can insert a new row we need to know the layout and datatypes of the table. We have to account for each column in the table and we also have to be aware of any constraints or restriction that the table has.

We do not need to list the columns individually as long as we are aware of the default order or the table.

If we list the columns they can be listed in any order as long as the values match the datatypes and the column name.

Make note of the fact that IF we violate any of the table constrains the insert statement will not be executed and your procedure will cease to function.

Inserting a NEW Row

We have two different types of INSERT statement.

IMPLICIT - We omit the unused columns from the column list, however we have to list the column names in the insert statement.

EXPLICIT – If we just have the table name we have to have a value for each column in the table and we have to list the values in the same order as the table. If we are inserting a null value we use the word NULL.

Inserting a NEW Row

If we have a table called DEPT that contains the columns of deptno, dname, and loc we could use the following statement.

If we use the column names:

```
INSERT INTO dept(deptno, dname, loc)
VALUES (50,'DEVELOPMENT', 'DETROIT');
```

OR

```
INSERT INTO dept(dname, deptno, loc)
VALUES ('DEVELOPMENT', 50,'DETROIT');
```

Both accomplish the same thing since we are using the column names it does not HAVE to be in the same order as the table layout.

Inserting a NEW Row

If we have a table called DEPT that contains the columns of deptno, dname, and loc we could use the following statement.

If we DO NOT use the column names the data has to be in the same order as the table layout.

```
INSERT INTO dept
values (50, 'DEVELOPMENT', 'DETRIOT');
```

If a column names are not listed, any column with an "UNASSIGNED" will have a NULL valued placed in it.

Changing data in a Table

If the table already exists and the columns are define and we need to change the data in a column we use the UPDATE command.

UPDATE tablename

SET columnname = to the value it is being changed to

WHERE condition that allows the change.

Updating data in a Table

EXAMPLE

If we want to change just one row:

```
UPDATE emp
SET deptno = 20
WHERE empno = 7782;
```

Changing all rows:

```
UPDATE emp
SET deptno = 20;
```

Updating data in a Table

EXAMPLE of using a subquery to update

If we want to change just one row making the job and deptno equal to another employee in the table:

```
UPDATE emp
SET (job, deptno) = (SELECT job, deptno
                     FROM emp
                     WHERE empno = 7499)
WHERE empno = 7698;
```

Deleting a row from a table

To remove an existing row or rows from a table we use the DELETE statement. Care should be used when deleting since there is this consideration: once the command is executed the data is gone.

This command is usually used in conjunction with the WHERE clause.

```
DELETE FROM department
WHERE dname = 'DEVELOPMENT';
```

If required we can also delete a row based on a subquery from another table.

Database Transactions

A transaction consists of the execution on one of the following statements: DML, DDL, and DCL.

DML: Data Manipulation Language - Consists of any number of DML statements that the server treats as a single entity or a logical unit of work.

DDL: Data Definition Language – Consists of ONLY one DDL statement.

DCL: Data Control Language – Consists of only one DCL statement.

DCL Statements

DCL: Data Control Language

COMMIT – Ends the current transaction by making all pending data changes permanent.

SAVEPOINT – Marks a savepoint within the current transaction.

ROLLBACK – Ends the current transaction by discarding all pending data changes.

ROLLBACK to SAVEPOINTA – By setting a savepoint and providing a name like A, B and so forth we can rollback to that savepoint name disregarding the changes after the savepoint was set. HOWEVER if a COMMIT has been issued the data is already made permanent.

Record Locking

Record locking is usually set up at the server level for the following reasons:

Prevents destructive interaction between concurrent transactions.

Required no user action.

Automatically uses the lowest level of restrictiveness.

Are held for the duration of the transaction.

There are two types.

Record Locking

The two types of record locking are: Exclusive and Share Lock.

Exclusive: Prevents a resource from being shared. The first transaction to lock the resource exclusively, is the ONLY transaction that can alter the resource until the exclusive lock is released.

Share Lock: Allows the resource to be shared. Multiple users reading data can share the data, holding share locks to prevent concurrent access by a WRITER (Who has the exclusive lock), Several transactions can acquire share locks on the same resource.