9. MANIPULACIÓN DE DATOS

9.1.- LENGUAJE DE MANIPULACIÓN DE DATOS (DML)

El lenguaje de manipulación de datos o DML (Data Manipulation Languaje) es la parte central de SQL.

Actúa de dos formas en la base de datos:

1. Cuando agrega, modifica o elimina registros de las tablas de la base de datos
2. Como transacción, la cual consiste en una colección de sentencias DML que forman una unidad lógica de trabajo.

9.2.- INSERCIÓN DE REGISTROS EN LA BASE DE DATOS: INSERT

Añade nuevos registros a una tabla mediante la sentencia INSERT

Sintaxis:

```
INSERT INTO  nb_tabla [ ( columna1[ , columna2 ...] ) ]
VALUES (valor1 [ , valor2...]);
```

Para insertar registros es conveniente seguir los siguientes pasos:

- Insertar una nueva fila que contenga los valores de cada columna de la tabla.
- Opcionalmente, se puede listar las columnas en la cláusula INSERT.
- Colocar valores en el orden que las columnas tienen en la tabla por defecto.
- Encerrar los valores de datos de tipo carácter y fecha entre comillas simples.

Ejemplo:

Insertar un registro en la tabla DEPT con la siguiente información:
Número de departamento: 50
Nombre: DEVELOMENT
Localidad : DETROIT

```
INSERT INTO DEPT (DEPTNO,DNAME,LOC)
VALUES (50,'DEVELOMENT','DETROIT');
```

Métodos para insertar valores nulos

Existen dos métodos para insertar valores nulos.

- MÉTODO IMPLÍCITO.

Consiste en omitir la columna de la lista

Ejemplo:

```
INSERT INTO DEPT (DEPTNO,DNAME)
VALUES (60,'MIS');
```

Al haber sido omitida la columna loc, se crea el registro con la columna loc como NULL

- MÉTODO EXPLICITO

Se especifica con la palabra null o con el string vacío ‘ ’ en la lista de valores.

Ejemplo:

```
INSERT INTO DEPT
VALUES (70,'FINANCES','');

INSERT INTO DEPT
VALUES(60,'FINANCES',NULL);
```

Inserción de valores especiales

SYSDATE:

Cuando se inserta un valor de fecha normalmente se inserta con el formato por defecto.

Si queremos convertirlo a otro formato en la inserción se utilizará la función TO_DATE

Ejemplo 1:

Insertar en la tabla EMP el siguiente empleado (7196, GREEN, SALESMAN, 7782, SYSDATE, 2000, NULL, 10), teniendo en cuenta que la fecha de ingreso es la fecha actual.

```
INSERT INTO EMP
VALUES(7196,'GREEN','SALESMAN',7782,SYSDATE,2000,NULL,10);
```

Ejemplo 2:

```
INSERT INTO EMP
VALUES      ( 2296,'ARMANDO','SALESMAN',7782,
              TO_DATE('FEB 02,00','MON DD,YY'),1300,NULL,10);
```

Inserción de valores por variables de sustitución

Se puede crear un comando INSERT que permita al usuario agregar valores interactivamente usando variables de sustitución SQL*Plus, por lo que es conveniente crear un script interactivo usando parámetros de sustitución, además podemos ayudar de la variable de entorno ACCEPT.

Ejemplo:

Insertar registros en la tabla DEPT usando variables de sustitución.

```
INSERT INTO DEPT
VALUES('&NUMDEPT','&NOMDEPT','&LOCDEPT');
```

```
Introduzca un valor para numdept: 80
Introduzca un valor para nomdept: VENTAS
Introduzca un valor para locdept: MADRID
```

Copia de registros de otra tabla

Se puede usar la sentencia INSERT para agregar filas a una tabla donde los valores se derivan de otras tablas existentes, usando en vez de la cláusula VALUES una subconsulta, para ello se debe:

1. Escribir el comando INSERT con una subconsulta.
2. No usar la cláusula VALUES.
3. Observar que coincida el número de columnas de INSERT con el de la subconsulta.

Sintaxis:

```
INSERT INTO   nb_tabla [ columna ( , columna... ) ]
                              SUBQUERY;
```

Ejemplo:

Supongamos que la tabla MANAGER tiene una estructura idéntica a la tabla conocida EMP y queremos rellenar la tabla MANAGER con el contenido de la tabla EMP en la que JOB sea MANAGER.

```
INSERT INTO MANAGER
(SELECT * FROM EMP
WHERE JOB = 'MANAGER');
```

9.3.- MODIFICACIÓN DE REGISTROS EN UNA TABLA: UPDATE

Para modificar el contenido de los registros de una tabla se hace mediante la sentencia UPDATE, esta sentencia permite además modificar varios registros a la vez.

Sintaxis:

```
UPDATE nb_tabla
SET columna1 = valor1 [, columna1 = valor2 ...]
[WHERE condición];
```

Si no especificamos el WHERE la modificación se hará en todos los registros de la tabla.

Ejemplo:

Transferir al empleado 7782 al departamento 20.

```
UPDATE EMP
SET DEPTNO=20
WHERE EMPNO=7782;
```

Las consultas multicolumna pueden ser implementadas en la cláusula SET de una sentencia UPDATE.

Sintaxis:

```
UPDATE nb_tabla
SET (columna,columna...)  =  (SELECT columna, columna, ...
                                FROM  nb_tabla
                                WHERE condición)
[WHERE condición];
```

Ejemplo:

Modificar el oficio y el número de departamento del empleado 7698 con los valores que corresponden al empleado 7499.

```
UPDATE      EMP
SET         (JOB, DEPTNO)=      (SELECT JOB, DEPTNO
                                 FROM EMP
                                 WHERE EMPNO=7499)
WHERE EMPNO = 7698;
```

Si se intenta actualizar un registro con un determinado valor que viola una restricción de identidad se produce un error.

Ejemplo:

```
UPDATE EMP
SET DEPTNO = 55
WHERE DEPTNO = 10;
```

Se produce un error pues no existe el departamento 55 en la tabla DEPT.

9.4.- ELIMINACIÓN DE REGISTROS DE UNA TABLA: DELETE

Para eliminar registros existentes de una tabla, se hace por medio de la sentencia DELETE.

Sintaxis:

```
DELETE [FROM ] nb_tabla
[WHERE condición];
```

Si se omite la cláusula WHERE se eliminan todos los registros de la tabla.

Ejemplo 1:

Borrar de la tabla EMP todos los empleados que trabajen en el departamento 30:

```
DELETE FROM EMP
WHERE DEPTNO=30;
```

Se pueden eliminar registros basados en otra tabla utilizando subconsultas en la sentencia DELETE.

Ejemplo 2:

Eliminar de la tabla EMP, aquellos cuyo número de departamento sea igual al número de departamento que tiene como nombre RESEARCH.

```
DELETE FROM EMP
WHERE DEPTNO      =     (SELECT DEPTNO
                        FROM DEPT
                        WHERE DNAME='RESEARCH');
```

Si se intenta borrar un registro con un valor relacionado con otra tabla dará un error de integridad, ya que, no se pueden eliminar registros que contengan una clave primaria, usada como clave extranjera en otra tabla.

Ejemplo 3:

Borrar en la tabla DEPT el departamento de código 10.

```
DELETE FROM DEPT
WHERE DEPTNO=10;
```

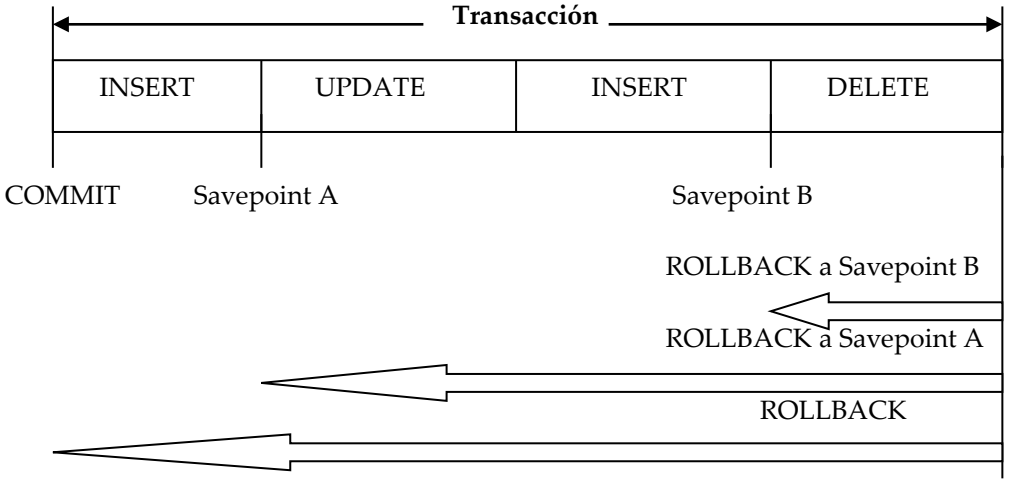
9.5.- TRANSACCIONES EN LA BASE DE DATOS

Existen tres tipos de sentencias

- DML(data manipulation lenguaje): INSERT, DELETE y UPDATE
- DDL (data definition lenguaje): CREATE y DROP.
- DCL (data control lenguaje): GRANT.

El servidor ORACLE asegura consistencia en los datos basándose en las transacciones, se entiende por TRANSACCIÓN al periodo de tiempo desde que el usuario ejecuta el primer comando SQL hasta que ocurre uno de los siguientes eventos:

- Un comando COMMIT o ROLLBACK.
- Un comando DDL o DCL.
- Al producirse ciertos errores.
- Finaliza la sesión SQL*Plus.
- Un fallo o caída del sistema.



Sentencia	Descripción
COMMIT;	- Finaliza la transacción actual haciendo que todos los cambios pendientes pasen a ser permanentes.
SAVEPOINT nombre;	- Realiza una marca dentro de la transacción en curso.
ROLLBACK [TO SAVEPOINT nombre];	- Finaliza la transacción en curso descartando a todos los cambios pendientes.

Ventajas de COMMIT y ROLLBACK.

1. Aseguran la consistencia de los datos.
2. Pueden visualizar los cambios sobre los datos antes de hacerlos permanentes.
3. Agrupan lógicamente las tareas relacionadas entre sí.

Estado de los datos antes de COMMIT y ROLLBACK.

1. El estado previo de los datos puede ser recuperado porque el afecto es el búfer de la base de datos.
2. El usuario actual debe revisar los resultados de sus operaciones DML usando la sentencia SELECT.
3. Otros usuarios no pueden ver los resultados de las sentencias DML ejecutadas por el usuario actual.
4. Las filas aceptadas son bloqueadas, otros usuarios no podrán cambiar los datos pertenecientes a esas filas.

Estado de los datos después del COMMIT:

1. Los datos manipulados son escritos en la BD.
2. Los datos anteriores se pierden definitivamente.
3. Todos los usuarios podrán ver los resultados.
4. Se liberan los bloqueos aplicados a las filas afectadas; esas filas están ahora disponibles para que otros usuarios las usen.
5. Se borran todos los SAVEPOINT.

Estado de los datos después de ROLLBACK.

1. Se descartan todos los cambios pendientes.
2. Se restaura el estado previo de los datos.
3. Los cambios de los datos se deshacen.
4. Se levantan todos los bloqueos sobre las filas afectadas.

Bloqueos en ORACLE 8i

Los bloqueos son mecanismos que previenen conflictos entre las transacciones, que acceden a los mismos recursos, bien sea un objeto de usuario (tabla o registro), o un objeto del sistema no visible por los usuarios (como estructuras de datos compartidas y registros del diccionario de datos).

El bloqueo de ORACLE 8i es completamente automático y no requiere acción por parte del usuario, estos previenen la interacción destructiva entre transacciones concurrentes, es además un mecanismo automático que utiliza el nivel más bajo aplicable de restricción, por tanto ofrece además un grado mas alto de concurrencia y máxima integridad de datos. También permite al usuario bloqueos de datos manuales.

ORACLE permite dos modos de bloqueo:

- Exclusivo: Previene la comparición de un recurso. La primera transacción que bloquea el recurso es la única que puede alterarlo, hasta liberar el bloqueo.

- Compartido: Permite la compartición de un recurso. Múltiples usuarios leyendo dato pueden compartir los datos, manteniendo bloqueos para prevenir acceso concurrente por una escritura.