MODIFICACIÓN

Inserción de registros

```
INSERT INTO nombre_tabla (lista_campos) VALUES (lista_valores);
```

Hay que tener en cuenta que cada campo de *lista_campos* debe tener un valor válido en la posición correspondiente de la *lista_valores,* sino se obtendrá un error en la ejecución. *(Para recordar los valores válidos para cada campo, utiliza la sentencia* **DESC**RIBE *seguida del nombre de la tabla).*
Al hacer un INSERT en el que no se especifiquen los valores de todos los campos, se obtendrá el valor NULL en aquellos campos que no se han indicado.

```
INSERT INTO USUARIOS (Login, Password, Nombre, Apellidos,
Direccion, CP, Localidad, Provincia, Pais, F_Nacimiento,
F_Ingreo, Correo, Credito, Sexo) VALUES ('migrod86',
'6PX5=V', 'MIGUEL ANGEL', 'RODRIGUEZ RODRIGUEZ', 'ARCO DEL
LADRILLO,PASEO', '47001', 'VALLADOLID', 'VALLADOLID',
'ESPAÑA', '27/04/1977', '10/01/2008', 'migrod86@gmail.com',
200, 'H');
```

Modificación de registros

```
UPDATE nombre_tabla SET nombre_campo=valor [, nombre_ campo=
valor]... [ WHERE condición ];
```

```
UPDATE USUARIOS SET Credito = 200;
UPDATE USUARIOS SET Credito = 0, Pais = NULL;
UPDATE USUARIOS SET Credito = 300 WHERE Sexo = 'M';
```

Borrado de registros

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

Si no se indica WHERE (opcional), se borrará todo el contenido de la tabla, aunque la tabla seguirá existiendo con la estructura que tenía hasta el momento.

```
DELETE FROM USUARIOS;
DELETE FROM USUARIOS WHERE Credito = 0;
```

Integridad referencial

Dos tablas pueden ser relacionadas entre ellas si tienen en común uno o más campos. Cada valor del campo que forma parte de la integridad referencial definida, debe corresponderse, en la otra tabla, con otro registro que contenga el mismo valor en el campo referenciado..
La relación existente entre la clave ajena (tabla hija) y la clave de referencia (tabla padre) tiene implicaciones en el borrado y modificación de sus valores. Podemos configurar la clave ajena para:

- **No permitir supresión**: Por defecto. En caso de que se intente se produce un error.
- **Supresión en cascada**: Los registros de la tabla hija también son borrados.
- **Definir nulo en suprimir**: Los valores de la clave ajena son cambiados al valor NULL.

```
CONSTRAINT JUEGOS_CON FOREIGN KEY (Cod_Juego) REFERENCES
JUEGO (Codigo) ON DELETE CASCADE –
```
(o también ON DELETE SET NULL)

Subconsultas y composiciones en órdenes de edición

Inserción de registros a partir de una consulta

```
INSERT INTO USUARIOS (LOGIN, PASSWORD, NOMBRE, APELLIDOS,
CORREO) VALUES ('natsan63', 'VBROMI', 'NATALIA', 'SANCHEZ
GARCIA', 'natsan63@hotmail.com');
```

Podemos escribirlo también como:

```
INSERT INTO (SELECT LOGIN, PASSWORD, NOMBRE, APELLIDOS,
CORREO FROM USUARIOS) VALUES ('natsan63', 'VBROMI',
'NATALIA', 'SANCHEZ GARCIA', 'natsan63@hotmail.com');
```

Viendo que es posible sustituir los datos por una consulta. Lo que nos permite hacer algo como:

```
INSERT INTO USUARIOS_SIN_CREDITO SELECT * FROM USUARIOS WHERE Credito=0;
```

Aquí no se debe especificar la palabra *VALUES*, pues no se está especificando una lista de valores.

```
INSERT INTO CLIENTES (Nombre_CLI) VALUES (SELECT Nombre_NCLI
FROM NUEVOS_CLIENTES);
```

Modificación de registros a partir de una consulta

Las consultas pueden formar parte de cualquiera de los elementos de la sentencia *UPDATE*.

Modifica el crédito de los usuarios que tienen una partida creada y cuyo estado es 1 (activa). El valor del crédito asignado es el valor más alto de los créditos de todos los usuarios.

```
UPDATE USUARIOS SET Credito = (SELECT MAX(Credito) FROM
USUARIOS) WHERE Login IN (SELECT Cod_Crea FROM PARTIDAS WHERE
Estado=1);
```

Actualizar en la tabla USUARIOS el crédito del usuario con código 3 para asignarle el mismo crédito que el del usuario con código 5

```
UPDATE USUARIOS SET Credito = (SELECT Credito FROM USUARIOS WHERE
Codigo = 3) WHERE Codigo = 5;
```

Supresión de registros a partir de una consulta

```
DELETE FROM (SELECT * FROM USUARIOS, PARTIDAS WHERE
Login=Cod_Crea AND Estado=0);
```

o también:

```
DELETE FROM (SELECT * FROM USUARIOS, PARTIDAS WHERE
Login=Cod_Crea) WHERE Estado=0;
```

```
DELETE FROM USUARIOS WHERE Codigo IN (SELECT Codigo FROM
ANTIGUOS);
```

Transacciones

Una transacción es una unidad atómica (no se puede dividir) de trabajo que contiene una o más sentencias SQL. A ellas se le aplica una operación `COMMIT(`aplicadas o guardadas en la base de datos), o `ROLLBACK` (deshacer las operaciones que deberían hacer sobre la base de datos.) Mientras que sobre una transacción no se haga `COMMIT`, los resultados de ésta pueden deshacerse.
Oracle cumplen con las propiedades básicas de las transacciones en base de datos:

Atomicidad: No hay transacciones parciales. O se hacen todas las tareas o se deshacen cambios.
Consistencia: La transacción se inicia y termina en un estado consistente de los datos.
Aislamiento: El efecto de una transacción no es visible por otras transacciones hasta que finaliza.
Durabilidad: Los cambios que han volcado sus modificaciones, se hacen permanentes.

Las sentencias de control de transacciones gestionan los cambios que realizan las sentencias DML y las agrupa en transacciones, permitiendo: Hacer permanentes los cambios, deshacerlos (`ROLLBACK`) desde que fue iniciada o desde un punto de restauración (`ROLLBACK TO SAVEPOINT`), establecer un punto intermedio (`SAVEPOINT`), indicar propiedades (`SET TRANSACTION`), especificar si una restricción de integridad aplazable se comprueba después de cada sentencia DML o cuando se ha realizado el `COMMIT` de la transacción (`SET CONSTRAINT`).

Hacer cambios permanentes en una transacción

Una transacción comienza cuando se encuentra la primera sentencia SQL ejecutable. Para que los cambios producidos durante la transacción se hagan permanentes (no puedan deshacerse):

- Utilizar `COMMIT`, que ordena a la base de datos que haga permanentes las sentencias.
- Ejecutar una sentencia DDL (como `CREATE`, `DROP`, `RENAME`, o `ALTER`). La base de datos ejecuta implícitamente una orden `COMMIT` antes y después de cada sentencia DDL.
- Cerrar adecuadamente el SGBD Oracle, que produce un volcado permanente de las cambios.

Deshacer cambios de una transacción: `ROLLBACK;`

La sentencia `ROLLBACK` permite deshacer los cambios efectuados por la transacción actual, dándola además por finalizada y volviendo al estado de la última transacción volcada.
si una transacción termina de forma anormal, por ejemplo, por un fallo de ejecución, los cambios que hasta el momento hubiera realizado la transacción son deshechos de forma automática.

Deshacer cambios de una transacción parcialmente

Un punto de restauración (`SAVEPOINT`) es un marcador intermedio declarado por el usuario dentro de una transacción. Los puntos de restauración dividen una transacción grande en pequeñas partes.
Establecerlo: `SAVEPOINT nombre_punto_restauración;`
Restaurar: `ROLLBACK TO SAVEPOINT nombre_punto_restauración;`

Acceso simultaneo a los datos: Problemática

Una base de datos multiusuario, las sentencias contenidas en varias transacciones simultáneas pueden actualizar los datos simultáneamente. Las transacciones ejecutadas simultáneamente, deben generar resultados consistentes. Por tanto, una base de datos multiusuario debe asegurar:

- **Concurrencia de datos**: asegura q. los usuarios pueden acceder a los datos al mismo tiempo
- **Consistencia de dato**s: asegura que cada usuario tiene una vista consistente de los datos, incluyendo cambios visibles realizados por este usuario y las finalizadas de otros usuarios.

El SGBD tiene mecanismos para prevenir la modificación concurrente del mismo dato. Los bloqueos se realizan de forma automática y no requiere la actuación del usuario y logran:

- **Consistencia**: Los datos que están siendo consultados o modificados por un usuario no pueden ser cambiados por otros hasta que el usuario haya finalizado la operación completa.
- **Integridad**: Los datos y sus estructuras deben reflejar todos los cambios efectuados sobre ellos en el orden correcto.

Políticas de bloqueo

Los bloqueos afectan a la interacción de lectores y escritores. Un lector es una consulta sobre un recurso, mientras que un escritor es una sentencia que realiza un modificación sobre un recurso:

- Un registro es bloqueado sólo cuando es modificado por un escritor: Cuando una sentencia actualiza un registro, la transacción obtiene un bloqueo sólo para ese registro.
- Un escritor de un registro bloquea a otro escritor concurrente del mismo registro: Si una transacción está modificando una fila, un bloqueo del registro impide que otra transacción modifique el mismo registro simultáneamente.
- Un lector nunca bloquea a un escritor, quien puede modificar dicho registro. La única excepción es la sentencia `SELECT … FOR UPDATE`, que es un tipo especial de sentencia `SELECT` que bloquea el registro que está siendo consultado.
- Un escritor nunca bloquea a un lector: Cuando un registro está siendo modificado, la base de datos proporciona al lector una vista del registro sin los cambios que se están realizando.

Hay dos mecanismos para el bloqueo de los datos en una base de datos: el bloqueo **pesimista** y el bloqueo **optimista**. El bloqueo pesimista (de registro o tabla) se realiza inmediatamente, en cuanto se solicita, mientras que en un bloqueo optimista el acceso al registro o la tabla sólo está cerrado en el momento en que los cambios realizados a ese registro se actualizan en el disco.
Esta última situación sólo es apropiada cuando hay menos posibilidad de que alguien necesite

acceder al registro mientras está bloqueado, de lo contrario no podemos estar seguros de que la actualización tenga éxito, porque el intento de actualizar el registro producirá un error si otro usuario actualiza antes el registro. Con el pesimista se garantiza que el registro será actualizado.

Bloqueos exclusivos y compartidos

Un recurso, p. ej. un registro de una tabla, sólo puede obtener un bloqueo exclusivo, pero puede conseguir varios bloqueos compartidos.

- **bloqueo exclusivo**: Este modo previene que sea compartido el recurso asociado. Una transacción obtiene un bloqueo exclusivo cuando modifica los datos. La primera transacción que bloquea un recurso exclusivamente, es la única transacción que puede modificar el recurso hasta que el bloqueo exclusivo es liberado.
- **bloqueo compartido**: Este modo permite que sea compartido el recurso asociado, dependiendo de la operación en la que se encuentra involucrado. Varios usuarios que estén leyendo datos pueden compartir los datos, realizando bloqueos compartidos para prevenir el acceso concurrente de un escritor que necesita un bloqueo exclusivo. Varias transacciones pueden obtener bloqueos compartidos del mismo recurso.

Ejemplo: Una transacción usa la sentencia `SELECT … FOR UPDATE` para consultar un registro de una tabla, y obtiene un bloqueo exclusivo del registro y un bloqueo compartido de la tabla. El bloqueo del registro permite a otras sesiones que modifiquen cualquier otro registro que no sea el bloqueado, mientras que el bloqueo de la tabla previene que otras sesiones modifiquen la estructura de la tabla. Así, la base de datos permite la ejecución de todas las sentencias que sean posibles.

Bloqueos automáticos

Oracle bloquea automáticamente un recurso usado por una transacción para prevenir que otras transacciones realicen alguna acción que requiera acceso exclusivo sobre el mismo recurso. La base de datos adquiere automáticamente diferentes tipo de bloqueos con diferentes niveles de restricción dependiendo del recurso y de la operación que se realice. Están divididos en las categorías:

- Bloqueos DML: Garantiza la integridad de los datos accedidos de forma concurrente por varios usuarios. P. ej, evitan que dos clientes compren el último artículo disponible en una tienda on-line. Estos bloqueos pueden ser sobre un sólo registro o sobre la tabla completa.
- Bloqueos DDL: Protegen la definición del esquema de un objeto mientras una operación DDL actúa sobre él. Los bloqueos se realizan de manera automática por cualquier transacción DDL que lo requiera. Los usuarios no pueden solicitarlo explícitamente.
- Bloqueos del sistema: Usa var. tipos para proteger BBDD interna y estructuras de memoria.

Bloqueos manuales

Se puede omitir los mecanismos de bloqueo que realiza por defecto la base de datos cuando:

- En aplicaciones que requieren consistencia en la consulta de datos a nivel de transacciones o en lecturas repetitivas: En este caso, las consultas obtienen datos consistentes en la duración de la transacción, sin reflejar los cambios realizados por otras transacciones.
- En aplicaciones que requieren que una transacción tenga acceso exclusivo a un recurso con el fin de que no tenga que esperar que otras transacciones finalicen.

La cancelación de los bloqueos automáticos se pueden realizar a nivel de sesión o de transacción. En el nivel de sesión, una sesión puede establecer el nivel requerido de aislamiento de la transacción con la sentencia `ALTER SESSION`. En el nivel de transacción, las transacciones que incluyan las siguientes sentencias SQL omiten el bloqueo por defecto:

```
SET TRANSACTION ISOLATION LEVEL
LOCK TABLE
SELECT...FOR UPDATE
```

Los bloqueos de las sentencias anteriores terminan una vez que la transacción ha finalizado.