

Gestión de Bases de Datos

Unidad 8: Modificación de los datos.

DML ORACLE.



IES Gonzalo Nazareno
CONSEJERÍA DE EDUCACIÓN

Raúl Ruiz Padilla
rruizp@gmail.com
Enero 2012

1. Inserción de registros.


Si queremos añadir datos individuales a una tabla debemos emplear la orden INSERT...VALUES.

```
INSERT INTO nombre_tabla [(column1 [, (...)]]  
VALUES (valor1 [, valor2]...);
```

Si no se especifican columnas, se entiende que serán todas las columnas de la tabla.

Los valores que se darán a cada columna deben tener el mismo tipo de dato con que se definió la columna.

Si una columna no está en la lista recibirá NULL. Si al crear la tabla la habíamos definido como NOT NULL, la orden INSERT fallará.



1. Inserción de registros. Ejemplos.

```
INSERT INTO profesores (apellidos, especialidad, cod_centro)  
VALUES ('Quiroga Martín, Ma Isabel', 'INFORMATICA', 45) ;
```

```
INSERT INTO profesores (apellidos, especialidad, cod_centro)  
VALUES('Seco Jiménez, Ernesto', 'LENGUA');  
Error..
```

```
INSERT INTO profesores  
VALUES(22, 23444800, 'Gonzalez Sevilla, Miguel A.', 'HISTORIA');
```



1. Inserción de registros. Consultas de datos anexados.

En ocasiones, interesa introducir en una tabla un conjunto de datos provenientes de otra fuente. En ese caso, es necesario realizar una consulta de datos anexados. El formato es el siguiente:

```
INSERT INTO nombre_tabla [(column1 [, (...)])]  
SELECT...
```

Se insertarán en la tabla tantos datos como filas devuelva la consulta. Veamos un ejemplo:

```
INSERT INTO emp_bcn  
SELECT *  
FROM emp  
WHERE deptno IN (SELECT deptno  
                  FROM dept  
                  WHERE loc = 'BARCELONA');
```



1. Inserción de registros. Consultas de datos anexados.

También es posible insertar un registro individual combinando información que ya conocemos con otra que hay que extraer de la base de datos. Veamos un ejemplo de esto:

Insertar un empleado con código 1111 y nombre GARCIA en el departamento en el que trabaja PEPE con el mismo sueldo de éste. La fecha de ingreso será la del sistema.

```
INSERT INTO emp (empno, ename, hiredate, sal, deptno)
SELECT 1111, 'GARCIA', sysdate, sal, deptno
FROM emp
WHERE ename = 'PEPE';
```



1. Inserción de registros. Consultas de datos anexados.

También es posible insertar un registro individual combinando información que ya conocemos con otra que hay que extraer de la base de datos. Veamos un ejemplo de esto:

Insertar un empleado con código 1111 y nombre GARCIA en el departamento en el que trabaja PEPE con el mismo sueldo de éste. La fecha de ingreso será la del sistema.

```
INSERT INTO emp (empno, ename, hiredate, sal, deptno)
SELECT 1111, 'GARCIA', sysdate, sal, deptno
FROM emp
WHERE ename = 'PEPE';
```

Actividades:

- Dadas las tablas, ALUM y NUEVOS, inserta en ALUM los nuevos alumnos.
- Inserta en EMP, el empleado 2000, SAAVEDRA, fecha de alta la de sistema, el salario el 20% mas que MILLER y el resto de los datos los de este.

2. Modificación de datos.

Para modificar el valor de los datos existentes en la tabla se emplea la orden UPDATE. La sintaxis es la siguiente:

```
UPDATE nombre_tabla  
SET column1 = valor1, ..column = valorn  
WHERE condicion;
```

Si se omite la cláusula WHERE se actualizarán todas las filas.

Ejemplo:

```
UPDATE emp  
SET sal = sal + 100, comm = NVL(comm, 0) + 10  
WHERE empno = 7369;
```



2. Modificación de datos. Consultas de actualización.

En una orden UPDATE podemos usar una consulta para obtener el valor que se va a poner en una columna.

En este caso, debe devolver una fila y el mismo nº de columnas y tipo de datos que las que hay entre paréntesis en el SET:

```
UPDATE emp  
SET sal = (SELECT max(sal) FROM emp)  
WHERE empno = 7082;
```

O bien podemos usarla para determinar qué filas se van a actualizar:

```
UPDATE emp  
SET sal = 1000  
WHERE deptno = (SELECT deptno FROM dept WHERE loc =  
'LORA');
```




2. Modificación de datos. Consultas de actualización.

También pueden combinarse ambos formatos como en el siguiente ejemplo:

Haz que los empleados del departamento 'ACCOUNTING' tengan el nombre en minúsculas y ganen el doble que 'JAMES':

```
UPDATE emp
SET ename = LOWER(ename),
    sal =      (SELECT sal * 2
                FROM emp
                WHERE ename = 'JAMES')
WHERE deptno = (SELECT deptno
                FROM dept
                WHERE dname = 'ACCOUNTING');
```



3. Borrado de datos. Consultas de eliminación.

Si deseamos eliminar uno o varios registros de una tabla, debemos emplear la orden DELETE. Su sintaxis es la siguiente:

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

Se borran los registros que cumplan la condición, en la que puede incluirse una sentencia SELECT si es necesario. A esto se le llama consulta de eliminación.

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

Ejemplo:

```
DELETE EMP  
WHERE DEPTNO = 20;
```



4. Control de transacciones en SQL.

Transacción: Secuencia de una o más ordenes SQL, que juntas forman una unidad de trabajo.

ROLLBACK, aborta la transacción, volviendo las tablas a la situación del último COMMIT.

ROLLBACK automático, se produce cuando haya algún fallo del sistema y no hayamos validado el trabajo hasta entonces.



4. Control de transacciones en SQL.

COMMIT, para validar las operaciones DML que hayamos realizado en la BD.

AUTOCOMMIT, parámetro que en SQL*Plus e iSQL*Plus, sirve para validar automáticamente las transacciones en la BD, siempre que esté en ON.

Por defecto está en OFF, por tanto INSERT, UPDATE y DELETE no se ejecutan automáticamente, hasta que no hagamos COMMIT

Para saber si está activado:

SHOW AUTOCOMMIT

Para activarlo:

SET AUTOCOMMIT ON



4. Control de transacciones en SQL.

Hay algunas órdenes SQL que fuerzan COMMIT implícito:

QUIT

EXIT

CONNECT

DISCONNECT

CREATE TABLE

DROP TABLE

CREATE VIEW

DROP VIEW

GRANT

ALTER

REVOKE

AUDIT

NOAUDIT

