Índice

1. INTRODUCCIÓN

Hay tres formas de actualizar los datos de una tabla:

1-Si se tiene que **añadir una nueva fila** a una tabla se utilizará el comando INSERT.
2.-Si se tiene que **borrar una fila** en una tabla se utilizará el comando DELETE.
3.-Si se tiene que **modificar alguna columna** de una fila se utilizará el comando UPDATE.

Los comandos anteriores se conocen a menudo como comandos de manipulación de datos o sentencias del lenguaje de manipulación de datos (DML).

2 INSERCIÓN DE FILAS

Para añadir nuevas filas a tablas ya existentes se utiliza el comando INSERT:

INSERT INTO <nombre_de_tabla>

[(<nombre_de_columna1>, <nombre_de_columna2>, ...)]

VALUES (<valor1>, <valor2>, ...);

2-1 Inserción de todas las columnas de una fila

Cuando se vayan a insertar valores para todas las columnas de una fila se puede utilizar el formato simplificado:

INSERT INTO <nombre_de_tabla>
VALUES (<valor1>, <valor2>, ...);

El orden de los valores dentro del paréntesis debe coincidir exactamente con el orden de las columnas tal y como fueron definidas al crear la tabla.

A cada columna se le deberá dar un valor, se puede utilizar la palabra clave **NULL** para indicar que esa columna tiene un valor nulo.

Ejemplo:

```
INSERT INTO employee values
(5000, 'GARCIA', 'MARIA', 'R', 667, 7839, sysdate, 900, null, 20);
```

```
1 filas insertadas.
```

Ahora podemos comprobar la inserción en la tabla:

```
SELECT * FROM employee
WHERE employee_id=5000;
```

	EMPLOYEE_ID	LAST_NAME	FIRST_NAME	MIDDLE_INITIAL	JOB_ID	MANAGER_ID	HIRE_DATE	SALARY	COMMISSION	DEPARTMENT_ID
1	5000	GARCIA	MARIA	R	667	7839	04/01/17	900	(null)	20

2-2 Inserción parcial de columnas

Si se quiere insertar valores sólo en parte de las columnas, habrá que indicar en la sentecia INSERT cuales son las columnas en las que vamos a insertar valores.

INSERT INTO <nombre_de_tabla>

[(<nombre_de_columna1>, <nombre_de_columna2>, ...)]

VALUES (<valor1>, <valor2>, ...);

El orden de los valores debe coincidir con el orden de las columnas que se ha especificado en la lista de columnas en las que se van a introducir datos.

Pero el orden no está determinado por el orden de las columnas dentro de la tabla.

Si una columna está definida como no nula hay que insertar obligatoriamente un valor en ella porque sino, al ejecutar el comando, se producirá un error.

Ejemplo:

```
INSERT INTO employee (employee_id, last_name, salary, department_id)  VALUES
(1657, 'SANZ', 3000,13);
```

```
1 filas insertadas.
```

Ahora podemos comprobar la inserción en la tabla:

```
SELECT * FROM employee
WHERE employee_id = 1657;
```

	EMPLOYEE_ID	LAST_NAME	FIRST_NAME	MIDDLE_INITIAL	JOB_ID	MANAGER_ID	HIRE_DATE	SALARY	COMMISSION	DEPARTMENT_ID
1	1657	SANZ	(null)	(null)	(null)	(null)	(null)	3000	(null)	13

2-3 Inserción de datos **desde otra tabla**

Otra posibilidad de crear filas es insertar los datos desde otra u otras tablas:

INSERT INTO <nombre_de_tabla>

[(<nombre_de_columna1>, <nombre_de_columna2>, ...)]

SELECT <lista_de_columnas_y_expresiones>

FROM <lista_de_tablas>

WHERE <criterios_de_selección>;

Se insertarán los datos de las tablas que figuren en la lista de tablas y que cumplan los criterios de selección.

El orden de las columnas en la claúsula SELECT debe coincidir con el orden de las columnas de la claúsula INSERT.

Se pueden incluir en la lista de selección expresiones aritméticas, valores reservados del sistema (SYSDATE) o funciones.

Ejemplo:

Supongamos que creamos una tabla con las primas de los empleados que llamamos 'prima' y que contiene las siguientes columnas: num_emp (número de empleado), cuantía (cantidad de la prima), funcion, fecha_prima.

```
CREATE TABLE prima (
        num_emp number (4),
        cuantia number (7,2),
        funcion number (3),
        fecha_prima date
);
```

```
table PRIMA creado.
```

Queremos insertar en esta tabla todos los empleados de la tabla employee cuya función sea 669 y 671 y asignarles una prima del 20% del sueldo.

```
INSERT INTO prima (num_emp, cuantia, funcion, fecha_prima)
SELECT employee_id, salary * 0.2, job_id, sysdate
FROM employee
WHERE job_id IN ( 669, 671);
```

```
11 filas insertadas.
```

Comprobamos que se han insertado las 11 filas

```
SELECT *
FROM prima
```

	NUM_EMP	CUANTIA	FUNCION	FECHA_PRIMA
1	7505	570	671	04/01/17
2	7506	550	671	04/01/17
3	7507	440	671	04/01/17
4	7566	595	671	04/01/17
5	7569	600	671	04/01/17
6	7698	570	671	04/01/17
7	7782	490	671	04/01/17
8	7788	600	669	04/01/17
9	7799	600	669	04/01/17
10	7902	600	669	04/01/17
11	7916	575	669	04/01/17

2.4 Inserción de fechas

Si estamos insertando datos en una columna de tipo fecha tendremos que introducir los datos con el formato por defecto de las fechas DD-MES-AA. Si la fecha tiene otro formato se deberá utilizar la función **TO_DATE** para indicarlo.

La empleada HIDALGO ingresó el 12/23/88

```
INSERT INTO employee  VALUES
(1456,'HIDALGO', 'MARIA', NULL,668,7839, TO_DATE ('12/23/88','MM/DD/YY'), 7000, NULL, 30);
```

```
1 filas insertadas.
```

Ahora podemos comprobar la inserción en la tabla:

```
SELECT * FROM employee
WHERE employee_id =1456;
```

	EMPLOYEE_ID	LAST_NAME	FIRST_NAME	MIDDLE_INITIAL	JOB_ID	MANAGER_ID	HIRE_DATE	SALARY	COMMISSION	DEPARTMENT_ID
1	1456	HIDALGO	MARIA	(null)	668	7839	23/12/88	7000	(null)	30

Si ahora queremos dar de alta al empleado PEREZ que ingresó el 4 de abril de 1979 a las 11:15.

```
INSERT INTO employee  VALUES
(1001, 'PEREZ', 'LUIS', NULL, 670, 1456, TO_DATE('04-04-79 11:15','DD-MM-YY HH:MI'), 8000, NULL, 30);
```

```
1 filas insertadas.
```

Ahora podemos comprobar la inserción en la tabla:

```
SELECT  employee_id, last_name,  job_id, TO_CHAR(hire_date, 'DD-MM-YY HH:MI'), salary
FROM employee
WHERE employee_id = 1001;
```

	EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
1	1001	PEREZ	670	04-04-79 11:15

Para insertar datos de tipo fecha también podemos utilizar la palabra reservada SYSDATE para insertar la fecha actual.

Consideraciones al insertar filas

Para insertar filas en una tabla ésta deberá existir.

Se utilizará el segundo formato cuando no haya que insertar en todas las columnas o cuando se quiera cambiar el orden de las columnas.

Las columnas de la tabla que no aparezcan en el comando INSERT tomarán un valor nulo.

Los valores insertados deben corresponderse con el tipo de datos de la columna.

Los valores de tipo CHAR y DATE se deben especificar entre comillas simples.

Si se utiliza el tercer formato la sentencia SELECT tiene que tener la sintáxis de consulta SQL, pero no se puede incluir la claúsula ORDER BY.

3. MODIFICACION DE FILAS

El comando de manipulación de datos UPDATE cambia los datos de una tabla dándoles nuevos valores. Pueden cambiarse todos los valores de una columna de todas las filas de la tabla o especificar las filas que deben ser cambiadas.

```
UPDATE  <nombre_de_la_tabla>
SET  columna = <nuevo_valor> [, columna = <nuevo_valor>]
[ WHERE  <criterio_de_selección_de_registro> ];
```

Se ha decidido que el empleado GARCIA pase al departamento 40.

```
UPDATE  employee
SET department_id = 40
WHERE last_name = 'GARCIA';
```

```
1 filas actualizadas.
```

```
SELECT *
FROM employee
WHERE last_name = 'GARCIA';
```

	EMPLOYEE_ID	LAST_NAME	FIRST_NAME	MIDDLE_INITIAL	JOB_ID	MANAGER_ID	HIRE_DATE	SALARY	COMMISSION	DEPARTMENT_ID
1	5000	GARCIA	MARIA	R	667	7839	04/01/17	900	(null)	40

El empleado SANZ pasa a tener la función 670 y va a tener una comisión de 1000.

```
UPDATE  employee
SET job_id = 670 , commission = 1000
WHERE last_name = 'SANZ';
```

```
1 filas actualizadas.
```

Comprobamos que se han actualizado los campos con la siguiente consulta

```
SELECT *
FROM employee
WHERE last_name='SANZ';
```

	EMPLOYEE_ID	LAST_NAME	FIRST_NAME	MIDDLE_INITIAL	JOB_ID	MANAGER_ID	HIRE_DATE	SALARY	COMMISSION	DEPARTMENT_ID
1	1657	SANZ	(null)	(null)	670	(null)	(null)	3000	1000	13

Se ha decidido aumentar las comisiones un 20%.

```
SELECT commission
FROM employee
WHERE commission IS NOT NULL;
```

	COMMISSION
1	300
2	500
3	300
4	1200
5	300
6	900
7	1400
8	1000
9	800
10	0
11	1000

```
UPDATE  employee
SET commission = commission * 1.2
WHERE commission IS NOT NULL;
```

```
11 filas actualizadas.
```

```
SELECT commission
FROM employee
WHERE commission IS NOT NULL;
```

	COMMISSION
1	360
2	600
3	360
4	1410
5	360
6	1080
7	1680
8	1200
9	960
10	0
11	1200

El nuevo valor de la columna se puede obtener mediante una consulta anidada.

UPDATE <nombre_de_la_tabla>
SET columna = (SELECT columna y/o expresión FROM ...)
[**WHERE** <criterio_de_selección_de_registro>];

Ejemplo:

El presidente debe ganar igual que el empleado que más gane:

```
UPDATE  employee
SET salary = ( SELECT max(salary)
                FROM employee
                WHERE job_id != (SELECT job_id
                                  FROM job
                                  WHERE function= 'PRESIDENT'))
WHERE job_id = (SELECT job_id
```

```
                    FROM job
                    WHERE function= 'PRESIDENT');
```

```
1 filas actualizadas.
```

Para comprobar

```
SELECT *
FROM EMPLOYEE
WHERE job_id = (SELECT job_id
                FROM job
                WHERE function= 'PRESIDENT');
```

	EMPLOYEE_ID	LAST_NAME	FIRST_NAME	MIDDLE_INITIAL	JOB_ID	MANAGER_ID	HIRE_DATE	SALARY	COMMISSION	DEPARTMENT_ID
1	7839	KING	FRANCIS	A	672	(null)	17/11/85	8000	(null)	10

En los criterios de selección pueden aparecer también consultas anidadas.

Se ha decidido subir el sueldo un 35% a los empleados que tengan un sueldo inferior al del empleado SANZ:

Para ello comprobamos primero lo que cobra el empleado SANZ

```
SELECT last_name, salary
FROM employee
WHERE salary = ( SELECT salary
                 FROM employee
                 WHERE last_name = 'SANZ');
```

	LAST_NAME	SALARY
1	ALBERTS	3000
2	SCOTT	3000
3	FISHER	3000
4	FORD	3000
5	SANZ	3000

Ahora vemos lo que cobran los empleados con un sueldo inferior a SANZ y lo que cobrarían cuando le subimos un 35%

```
SELECT last_name, salary, salary*1.35
FROM employee
WHERE salary < ANY ( SELECT salary
                     FROM employee
                     WHERE last_name = 'SANZ')
ORDER BY salary DESC
```

	LAST_NAME	SALARY	SALARY*1.35
1	JONES	2975	4016,25
2	ROBERTS	2875	3881,25
3	BLAKE	2850	3847,5
4	DOYLE	2850	3847,5
5	DENNIS	2750	3712,5
6	CLARK	2450	3307,5
7	BAKER	2200	2970
8	SOMMERS	1850	2497,5
9	LEWIS	1800	2430
10	ALLEN	1600	2160
11	TURNER	1500	2025
12	WEST	1500	2025
13	MILLER	1300	1755
14	ROSS	1300	1755
15	MARTIN	1250	1687,5
16	SHAW	1250	1687,5
17	WARD	1250	1687,5
18	PETERS	1250	1687,5
19	DUNCAN	1250	1687,5
20	LANGE	1250	1687,5
21	PORTER	1250	1687,5
22	ADAMS	1100	1485
23	JAMES	950	1282,5
24	GARCIA	900	1215
25	SMITH	800	1080
26	DOUGLAS	800	1080
27	MURRAY	750	1012,5
28	JENSEN	750	1012,5

Al actualizar los 6 primeros empleados no aparecerán porque se pasan de lo que cobra SANZ que cobra 3000. La segunda columna ha de coincidir con el salario actualizado

```
UPDATE  employee
SET salary = salary * 1.35
WHERE salary < ANY ( SELECT salary
                     FROM employee
                     WHERE last_name = 'SANZ');
```

```
28 filas actualizadas.
```

```
SELECT last_name, salary
FROM employee
WHERE salary < ANY ( SELECT salary
                     FROM employee
                     WHERE last_name = 'SANZ')
ORDER BY salary DESC;
```

	LAST_NAME	SALARY
1	BAKER	2970
2	SOMERS	2497,5
3	LEWIS	2430
4	ALLEN	2160
5	WEST	2025
6	TURNER	2025
7	MILLER	1755
8	ROSS	1755
9	WARD	1687,5
10	PETERS	1687,5
11	MARTIN	1687,5
12	LANGE	1687,5
13	DUNCAN	1687,5
14	SHAW	1687,5
15	PORTER	1687,5
16	ADAMS	1485
17	JAMES	1282,5
18	GARCIA	1215
19	DOUGLAS	1080
20	SMITH	1080
21	MURRAY	1012,5
22	JENSEN	1012,5

Consideraciones al modificar filas

Si se omite la claúsula WHERE se modificarán todas las filas de la tabla, pero si se especifica, se modificarán sólo las filas que cumplan la condición.

El nuevo valor del dato se puede obtener mediante expresiones, constantes o subconsultas.

Si el nuevo valor de la columna se obtiene a través de una subconsulta, esta subconsulta deberá recuperar tantas columnas como se estén modificando.

```
UPDATE  employee
```

```
SET (salary,commission) = (SELECT max(salary),max(commission)
                           FROM employee
                           WHERE job_id = (SELECT job_id
                                           FROM job
                                           WHERE function ='SALESPERSON'));
```

```
36 filas actualizadas.
```

```
SELECT salary, commission
FROM employee;
```

	SALARY	COMMISSION
1	8000	1680
2	8000	1680
3	8000	1680
4	8000	1680
5	8000	1680
6	8000	1680
7	8000	1680
8	8000	1680
9	8000	1680
10	8000	1680
11	8000	1680
12	8000	1680
13	8000	1680
14	8000	1680
15	8000	1680
16	8000	1680
17	8000	1680
18	8000	1680
19	8000	1680
20	8000	1680
21	8000	1680
22	8000	1680
23	8000	1680
24	8000	1680
25	8000	1680
26	8000	1680
27	8000	1680
28	8000	1680

4. BORRADO DE FILAS

Los usuarios pueden borrar filas de las tablas utilizando el comando DELETE.

DELETE [FROM] <nombre_de_la_tabla>
[**WHERE** <criterio_de_selección>];

Si un comando DELETE no contiene la claúsula WHERE se borrarán todas las filas de una tabla.

La definición de la tabla seguirá almacenada en la base de datos y se podrán insertar nuevas filas con el comando INSERT.
Las filas no se pueden borrar parcialmente, siempre lo son en su totalidad.

El empleado SANZ ha decidido abandonar la empresa:

```
DELETE employee
WHERE  last_name = 'SANZ';
```

```
1  filas  eliminado
```

Se va a rescindir el contrato a todos los empleados del departamento de investigación.

Comprobamos que hay empleados en el departamento de investigación

```
SELECT *
FROM employee
WHERE  department_id IN ( SELECT department_id
                          FROM department
                          WHERE  name = 'RESEARCH');
```

	EMPLOYEE_ID	LAST_NAME	FIRST_NAME	MIDDLE_INITIAL	JOB_ID	MANAGER_ID	HIRE_DATE	SALARY	COMMISSION	DEPARTMENT_ID
1	7919	DOUGLAS	MICHAEL	A	667	7799	04/01/87	800	(null)	12
2	7916	ROBERTS	GRACE	M	669	7569	04/01/87	2875	(null)	12
3	7799	FISHER	MATTHEW	G	669	7569	12/12/86	3000	(null)	12
4	7569	ALBERTS	CHRIS	L	671	7839	06/04/85	3000	(null)	12
5	7902	FORD	JENNIFER	D	669	7566	03/12/85	3000	(null)	20
6	7566	JONES	TERRY	M	671	7839	02/04/85	2975	(null)	20
7	7369	SMITH	JOHN	Q	667	7902	17/12/84	800	(null)	20
8	7876	ADAMS	DIANE	G	667	7788	12/01/87	1100	(null)	20
9	7788	SCOTT	DONALD	T	669	7566	09/12/86	3000	(null)	20

Borramos ahora los empleados del departamento de investigación

```
DELETE employee
WHERE  department_id IN ( SELECT department_id
                          FROM department
                           WHERE  name = 'RESEARCH');
```

```
9 filas eliminado
```

Comprobamos ahora que se han borrado

```
SELECT *
FROM employee
WHERE  department_id IN ( SELECT department_id
                    FROM department
                    WHERE  name = 'RESEARCH');
```

EMPLOY...	LAST_N...	FIRST_...	MIDDLE...	JOB_ID	MANAG...	HIRE_D...	SALARY	COMMIS...	DEPART...

Vemos que no hay ninguno

5.PROCESO DE TRANSACCIONES

8. MANIPULACION DE DATOS

ORACLE permite que los cambios en la base de datos se puedan agrupar en transacciones para mantener la consistencia e integridad de los datos.

Sólo se producirán los cambios contenidos en una transacción si se producen todos. Si no se producen todos no se producirá ninguno.

Para controlar explícitamente el proceso de transacciones se dispone de los siguientes comandos SQL:

COMMIT [WORK];	Señala el final de una transacción y el principio de otra para indicarle al sistema que se deben validar los cambios que se produjeron desde el principio de la transacción que se da por concluida y **BIEN.**
ROLLBACK [WORK];	Señala el final de una transacción y el principio de otra para indicarle al sistema que se restaure el estado de la base de datos tal y como estaba al comenzar la transacción, **HEMOS HECHO ALGO MAL.**

SQL*PLUS dispone de un comando para controlar las validaciones de forma automática:

SHOW AUTOCOMMIT	Nos muestra si los cambios van a validarse automáticamente después de cada operación de manipulación de datos (IMM) o si es el usuario el que llevará el control de las transacciones con los comandos anteriores (OFF).
SET AUTOCOMMIT [ON/OFF]	Sirve para especificar la opción deseada.

Todos los cambios que se han producido en la base de datos desde la última validación o restauración se validan con un COMMIT y se restaura la situación anterior con un ROLLBACK.

Antes de que se valide cualquier cambio éste sólo será visible para el usuario que lo ha hecho (consistencia en la lectura).

Todos los cambios que formen una unidad lógica deben ir incluidos dentro de una transacción y validarse o restaurarse todos a la vez.

Si la opción AUTOCOMMIT está activa todos los cambios se validan automáticamente cuando son ejecutados.

Los comandos del lenguaje de definición de datos llevan implícita una validación.

La desconexión de ORACLE lleva implícita una validación.

Si se produce una caída del sistema o una terminación anormal de una aplicación se llevará a cabo una restauración automática.

6.EJERCICIOS PROPUESTOS

1.-Insertar un nuevo departamento alumnos con el código 44 sin localidad.

2.-Insertar un empleado nuevo llamado JUAN GARCIA que va a trabajar en las mismas condiciones que ALICE y va a ser su código + 1.

3.-Al final no se incorpora JUAN GARCÍA, sino que se incorpora PEPE PEREZ con las mismas condiciones. Modificar el registro.

4.- Subir el sueldo un 10% a los empleados del departamento 20 y un 5% al resto

5.-Poner una comisión del 10% del salario a los empleados cuya función sean analistas:

6.- Poner a SHAPE UP con el máximo crédito

7.- Modificar el nombre de todos los departamentos anteponiendo su código de departamento – y el resto del nombre.

8.- Borrar todos los productos cuya fecha de comienzo sea enero de 1989

9.- poner una comisión del 10% del salario a aquellos empleados del departamento 20 que no tienen comision

10.- Modificar, del departamento 20, aquellos empleados que tienen sueldo maximo, poniendole como comision el 50% del salario.

11.- Pon una comisión del 20% del salario al empleado que tenga mayor comisión.

12. Aumentar el salario un10% mas la comisión a los empleados que trabajan en el departamento 30 y tienen un salario inferior a la media salarial de su departamento, estos empleados han de tener como función de trabajo analista.

13.- El producto que menos se ha vendido ha de pasar a llamarse Patatas fritas.

14.- Dar de alta dos clientes cuyo vendedor asociado sea el 7499 y tengan como

crédito limite 6000.

15.- A los empleados con menos antigüedad en la empresa, considerando menos antigüedad a todos los que entraron el ultimo año que se contrató, les vamos a poner como fecha de entrada en la empresa 01/01/2006, el salario de DOYLE y como jefe a JOHN.

16.- Insertar un empleado de apellido Fernández con nº de identificación 6000, la fecha de alta será la de hoy el salario el de GREGORY mas el 20% y el resto de los datos los mismos que los de GREGORY.

17.- Modificar el nº de departamento de Fernández, el nº de departamento será el departamento donde hay más empleados cuya función sea Clerk.

18.- Borrar todos los departamentos de la tabla departamentos para los cuales no existen empleados en la tabla empleados.

19.- Borrar aquellos clientes que no han realizado compras.

20.- Borrar todos los empleados cuyo jefe es DOYLE.

21.- Borrar el departamento 43

22.- hacer rollbac. Insertar los siguientes empleados en la tabla employee:

Maria camacho de codigo 8000 realiza la funcion 669 tiene como jefe al empleado 7820, la fecha de entrada en la empresa es la de hoy, su salario es 1300, la comision 50, pertenece al departamento 20 y su middle_initial es Q.

Bernabé Gonzalez de codigo 9000 realiza la funcion 670 tiene como jefe al empleado 7820 la fecha de entrada en la empresa es la de hoy su salario es 6000, la comision 600 pertenece al departamento 43 y su middle initial es S.

Comprobar que el ejercicio anterior sigue siendo válido. Es decir que sigue borrando el departamento 43.