



Unidad 10

Manipulación de datos

CONCEPTOS

- Insertar, modificar y eliminar filas de una tabla: INSERT, UPDATE, DELETE
- Transacción: COMMIT, ROLLBACK

1. INSERCIÓN DE DATOS

- INSERT

Con esta orden se añaden filas de datos en una tabla.

- Es posible introducir los valores directamente en la sentencia o introducirlos a partir de la información existente en la BD mediante una SELECT

Formato

**INSERT INTO *Nombretabla* [(columna [,columna]....)]
VALUES (*valor* [,valor]....);**

- [(columna [,columna]....)] representa la columna o columnas donde se van a introducir valores. Si las columnas no se especifican, se consideran por defecto todas las columnas
- (*valor* [,valor]....) representa los valores que se van a dar a las columnas. Si no se da la lista de columnas, se han de introducir valores en todas las columnas. Se deben corresponder los tipos de datos. Las columnas que no se encuentren en la lista recibirán el valor de NULL

Ejemplo1

]

Nombre de Columna	Tipo de Dato	Nulo	V
JOB_ID	VARCHAR2(10)	No	-
JOB_TITLE	VARCHAR2(35)	No	-
MIN_SALARY	NUMBER(6,0)	Yes	-
MAX_SALARY	NUMBER(6,0)	Yes	-

```
INSERT INTO JOBS (JOB_ID, JOB_TITLE, MIN_SALARY)
VALUES ('REPRE', 'Representante', 1800);
```

1 fila(s) insertada(s).

Ejemplo1

AC_ACCOUNT	Public Accountant	4200	9000
SA_MAN	Sales Manager	10000	20000
SA_REP	Sales Representative	6000	12000
PU_MAN	Purchasing Manager	8000	15000
PU_CLERK	Purchasing Clerk	2500	5500
ST_MAN	Stock Manager	5500	8500
ST_CLERK	Stock Clerk	2000	5000
SH_CLERK	Shipping Clerk	2500	5500
IT_PROG	Programmer	4000	10000
MK_MAN	Marketing Manager	9000	15000
MK_REP	Marketing Representative	4000	9000
HR_REP	Human Resources Representative	4000	9000
PR_REP	Public Relations Representative	4500	10500
REPRE	Representante	1800	-

- Las columnas a las que damos valores se identifican por su nombre

***INSERT INTO JOBS (JOB_ID, JOB_TITLE, MIN_SALARY)
VALUES ('REPRE', 'Representante', 1800);***

- La asociación columna-valor es por posición
JOB_ID -> 'REPRE', JOB_TITLE → Representante'
- Los valores que se dan a las columnas deben coincidir con el tipo de dato de la columna.
- Los valores constantes de tipo carácter o fecha deben ir entre comillas simples.
- Las columnas para las que no damos valores, aparecen como NULL

Ejemplo2

Nombre de Columna	Tipo de Dato	Nulo	V
JOB_ID	VARCHAR2(10)	No	-
JOB_TITLE	VARCHAR2(35)	No	-
MIN_SALARY	NUMBER(6,0)	Yes	-
MAX_SALARY	NUMBER(6,0)	Yes	-

```
INSERT INTO JOBS (JOB_ID, MIN_SALARY)  
VALUES ('CLA_VE', 2000);
```

```
ORA-01400: no se puede realizar una inserción NULL en ("HR"."JOBS"."JOB_TITLE")
```

Si en la definición de la columna, la especificamos como NOT NULL, no podemos insertar una fila con valor nulo en esa columna

Ejemplo3

```
INSERT INTO JOBS  
VALUES ('COM', 'Comercial',1500,2000);
```

Si damos un valor para cada una de las columnas, no es necesario indicar el nombre de las columnas.

INSERT CON SELECT

- Hasta ahora solo hemos insertado una fila, pero si a INSERT añadimos una consulta, se añaden tantas filas como devuelva la consulta. Si no se especifican los nombres de columna en INSERT, por defecto se consideran todas las columnas de la tabla

```
INSERT INTO Nombretabla1 [(columna [,columna]...)]  
  SELECT {columna [,columna] ... | *}  
  FROM Nombretabla2 [cláusulas de select];
```

Ejemplo 4

- Primero creamos la tabla test

```
create table test(id number(6) primary key,  
name varchar2(20), salary number(8,2))
```

- Insertamos en esta tabla el identificador de empleado, el apellido y el salario de los empleados del departamento 30

```
insert into test
```

```
select employee_id, last_name, salary  
from employees  
where department_id=30
```

Ejemplo5

```
insert into test (id, name)  
select employee_id, first_name  
from employees  
where job_id='SA_MAN'
```

En este caso es necesario escribir los nombres de columna id y name

Ejemplo 6

- Insertar en la tabla JOBS el identificador REP, el nombre de trabajo, Repartidor y el salario máximo y mínimo, el mismo que el del puesto de trabajo Sales Manager.

insert into jobs

select 'REP', 'Repartidor', min_salary, max_salary
from jobs

where job_title='Sales Manager'

Ejemplo 7

- Podemos utilizar funciones a la hora de insertar valores

JOB_HISTORY

]

Nombre de Columna	Tipo de Dato	Nulo
EMPLOYEE_ID	NUMBER(6,0)	No
START_DATE	DATE	No
END_DATE	DATE	No
JOB_ID	VARCHAR2(10)	No
DEPARTMENT_ID	NUMBER(4,0)	Yes

insert into job_history
values (110,'1/11/2001', sysdate, 'SA_MAN', 90)

2. MODIFICACIÓN DE DATOS

Para actualizar los valores de las columnas de una o varias filas utilizamos la orden:

UPDATE nombretabla

SET col1=valor1, col2=valor2,...

WHERE condición

Si se omite WHERE, la actualización afectará a todas las filas de la tabla.



Ejemplo8

ID	NAME	SALARY
114	Raphaely	11000
115	Khoo	3100
116	Baida	2900
117	Tobias	2800
118	Himuro	2600
119	Colmenares	2500
145	John	-
146	Karen	-
147	Alberto	-
148	Gerald	-



UPDATE test

SET name ='David' , salary=2000

WHERE id=148

Después de la actualización...

ID	NAME	SALARY
114	Raphaely	11000
115	Khoo	3100
116	Baida	2900
117	Tobias	2800
118	Himuro	2600
119	Colmenares	2500
145	John	-
146	Karen	-
147	Alberto	-
148	David	2000



UPDATE Con SELECT

- Podemos incluir una subconsulta en una sentencia UPDATE. La SELECT puede estar contenida en la cláusula WHERE o puede formar parte de SET.
- Cuando la subconsulta (SELECT) forma parte de SET debe seleccionar una única fila y el mismo número de columnas, que las que hay entre paréntesis al lado de SET

UPDATE Con SELECT

Formato1:

**UPDATE nombretabla
SET (col1,col2,...)=(SELECT
col1,col2,...)
WHERE condición**

UPDATE Con SELECT

Formato2:

UPDATE nombretabla

**SET column1=(SELECT col1...),
column2 =(SELECT col2...), ...**

WHERE condición

UPDATE Con SELECT

Formato3:

UPDATE nombretabla

**SET columna1=valor1,
columna2=valor2 ...**

WHERE condición=(select...)

UPDATE TEST

SET (name, salary)=(select first_name, salary
from employees where employee_id=148)

WHERE id=114.

En este caso modifica el nombre y salario del
id 114 con el nombre y salario del empleado
148



Ejemplo10

UPDATE TEST

```
SET (name, salary)=(select first_name, salary  
from employees where employee_id=148)
```

▪

En este caso modifica el nombre y salario
todos los empleados con el nombre y salario
del empleado 148

3. ELIMINACIÓN DE DATOS

DELETE FROM NombreTabla
WHERE condicion

Si no especificamos la cláusula WHERE,
borraremos TODAS las filas de la tabla.

Ejemplo11

```
DELETE FROM test  
WHERE name='Robert'
```

Borra todas las filas de la tabla test en las que el nombre es Robert.

Ejemplo12

```
DELETE FROM test  
WHERE salary > (select avg(salary) from  
employees where department_id=110)
```

Borra todas las filas de la tabla test en las que el salario sea mayor que la media del salario de los empleados del departamento 110



Ejemplo13

**DELETE FROM departments
WHERE department_id=50**

Explicar Describir SQL Guardado Historial

ORA-02292: restricción de integridad (HR.EMP_DEPT_FK) violada - registro secundario encontrado

No permite borrar ya que hay una clave ajena en la tabla employees que hace referencia al departamento 50 (por defecto, si no se especifica nada, la opción de borrado es restrict)

4. Truncar una tabla

- Quita todas las filas de una tabla sin registrar las eliminaciones individuales de filas. Desde un punto de vista funcional, TRUNCATE TABLE es equivalente a la instrucción DELETE sin una cláusula WHERE; no obstante, TRUNCATE TABLE es más rápida y utiliza menos recursos de registros de transacciones y de sistema.

Ejemplo13

■ TRUNCATE TABLE TEST

Borra las filas de la tabla test, **no borra la tabla.**

TRANSACCIONES

- Una transacción es un grupo de acciones que hacen transformaciones (inserción, modificación o eliminación de datos) en las tablas preservando la consistencia de los datos.

TRANSACCIONES

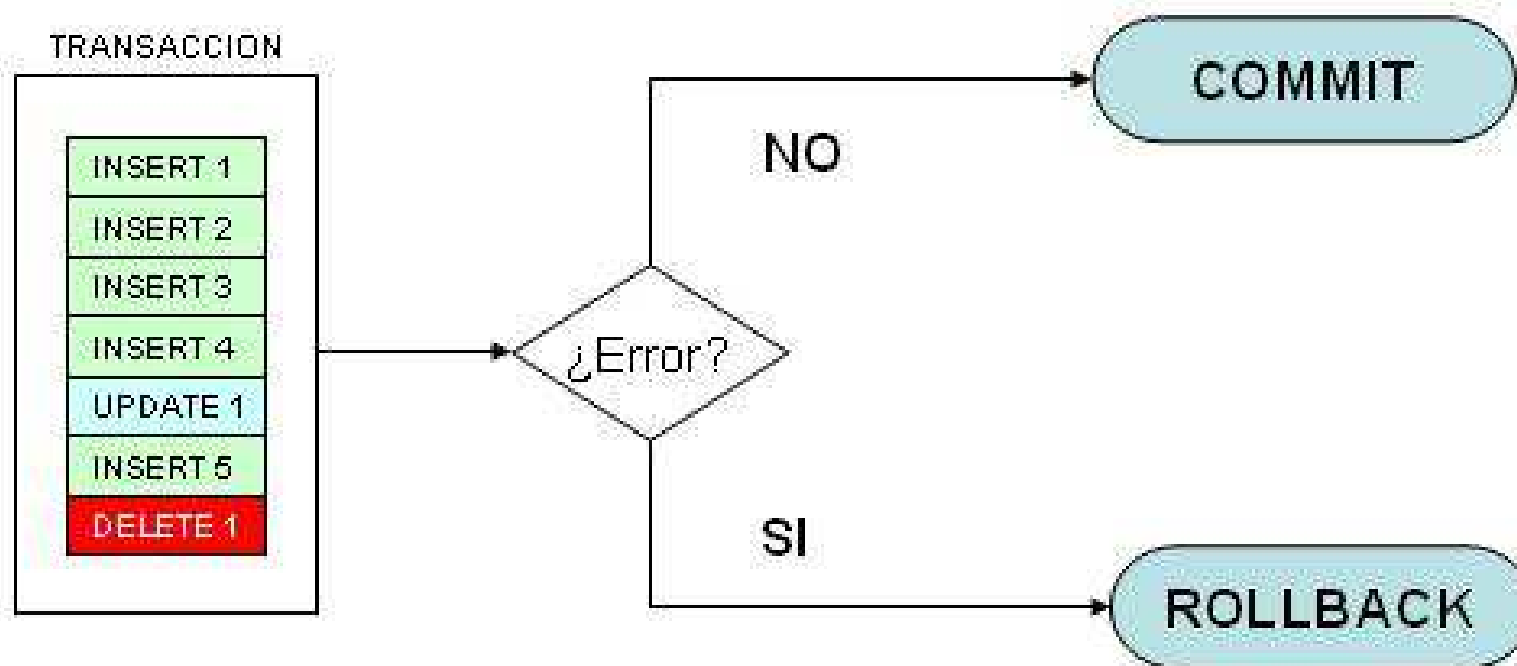
- Es decir, una transacción es una o varias sentencias SQL que se ejecutan en una base de datos como una única operación, confirmándose o deshaciéndose en grupo.
- No todas las operaciones SQL son transaccionales. Sólo son transaccionales las operaciones correspondiente al DML, es decir, sentencias SELECT, INSERT, UPDATE y

DELETE

COMMIT Y ROLLBACK

- Para confirmar una transacción se utiliza la sentencia **COMMIT**. Cuando realizamos **COMMIT** los cambios se escriben en la BD
- Para deshacer una transacción se utiliza la sentencia **ROLLBACK**. Cuando realizamos **ROLLBACK** se deshacen todas las modificaciones realizadas por la transacción en la BD, quedando ésta en el mismo estado que antes de iniciarse la transacción.

FLUJO DE UNA TRANSACCIÓN



Ejemplo

- Un ejemplo son las transferencias bancarias. Para realizar una transferencia de dinero entre dos cuentas debemos descontar el dinero de una cuenta, realizar el ingreso en la otra cuenta y grabar las operaciones y movimientos necesarios, actualizar los saldos
- Si en alguno de estos puntos se produce un fallo en el sistema podríamos hacer descontado el dinero de una de las cuentas y no haberlo ingresado en la otra. Por lo tanto, todas estas operaciones deben ser correctas o fallar todas. En estos casos, al confirmar la transacción (COMMIT) o al deshacerla (ROLLBACK) garantizamos que todos los datos quedan en un estado consistente

TRANSACCIONES

- En una transacción los datos modificados no son visibles por el resto de usuarios hasta que se confirme la transacción.
- Mientras se procesa una transacción, ningún otro usuario puede realizar cambios sobre los datos afectados por ella.

El siguiente ejemplo muestra una supuesta transacción bancaria:



```
DECLARE
    importe NUMBER;
    ctaOrigen VARCHAR2(23);
    ctaDestino VARCHAR2(23);
BEGIN
    importe := 100;
    ctaOrigen := '2530 10 2000 1234567890';
    ctaDestino := '2532 10 2010 0987654321';
    UPDATE CUENTAS SET SALDO = SALDO - importe
    WHERE CUENTA = ctaOrigen;
    UPDATE CUENTAS SET SALDO = SALDO + importe
    WHERE CUENTA = ctaDestino;
    INSERT INTO MOVIMIENTOS
    (CUENTA_ORIGEN, CUENTA_DESTINO, IMPORTE, FECHA_MOVIMIENTO)
    VALUES
    (ctaOrigen, ctaDestino, importe*(-1), SYSDATE);
    INSERT INTO MOVIMIENTOS
    (CUENTA_ORIGEN, CUENTA_DESTINO, IMPORTE, FECHA_MOVIMIENTO)
    VALUES
    (ctaDestino, ctaOrigen, importe, SYSDATE);
    COMMIT;
EXCEPTION
    WHEN OTHERS THEN
        dbms_output.put_line('Error en la transaccion: '||SQLERRM);
        dbms_output.put_line('Se deshacen las modificaciones');
        ROLLBACK;
END;
```

Si alguna de las tablas afectadas por la transacción tiene triggers, las operaciones que realiza el trigger están dentro del ámbito de la transacción, y son confirmadas o deshechas conjuntamente con la transacción.

VALIDACIÓN AUTOMÁTICA

Podemos validar automáticamente las transacciones sin tener que indicarlo de forma explícita, es decir, sin tener que hacer un COMMIT.

Desde el entorno gráfico de OracleXE, podemos marcar la casilla *Confirmación automática*

ORACLE® Database Express Edition

Usuario: HR

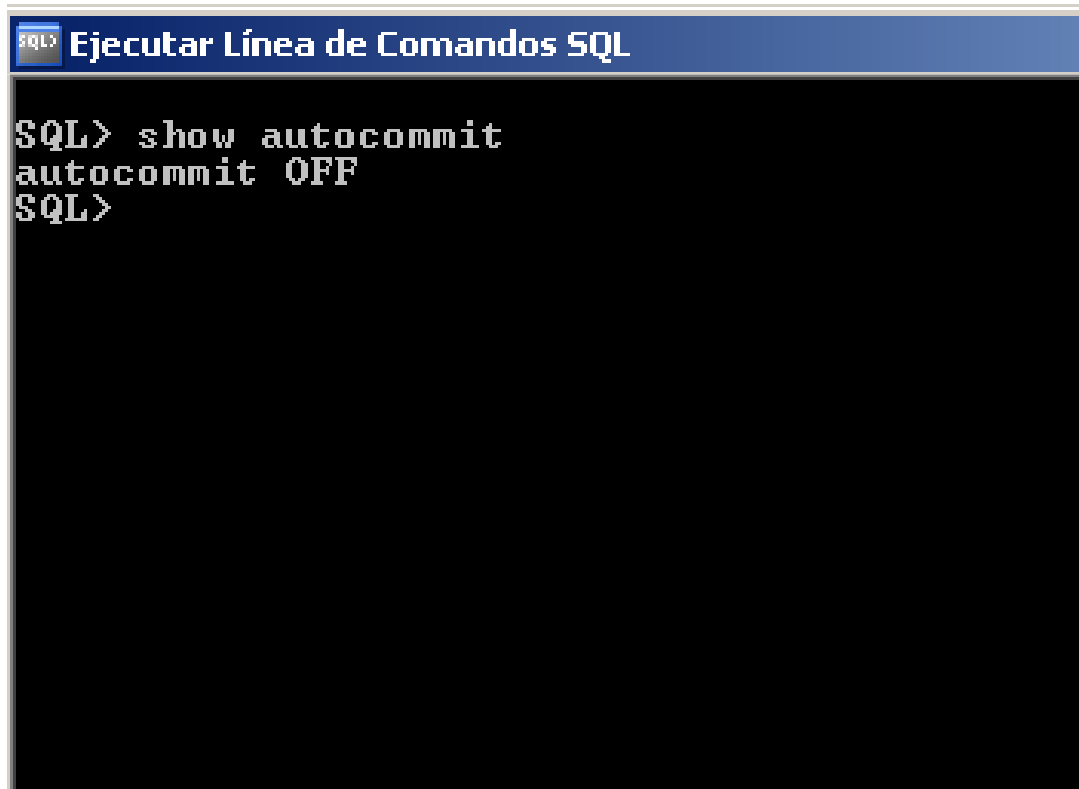
Inicio > SQL > Comandos SQL

☐ Confirmación Automática Mostrar 10 ▼

```
delete from jobs
```

VALIDACIÓN AUTOMÁTICA

Desde SQL*Plus, utilizaremos el parámetro AUTOCOMMIT. Para ver su valor



```
SQL> show autocommit
autocommit OFF
SQL>
```


VALIDACIÓN AUTOMÁTICA

OFF es el valor por omisión, de modo que las transacciones Insert, update, delete, no son definitivas hasta que no hagamos COMMIT, Para activar el AUTOCOMMIT

SQL> Ejecutar Línea de Comandos SQL

```
SQL> set autocommit on;  
SQL> show autocommit  
autocommit IMMEDIATE  
SQL>
```

COMMIT

- Hay varias órdenes SQL que fuerzan a que se ejecute un COMMIT sin necesidad de indicarlo
-
- | | | |
|--------------|--------------|------------|
| ■ QUIT | CREATE TABLE | DROP TABLE |
| ■ EXIT | CREATE VIEW | DROP VIEW |
| ■ CONNECT | GRANT | ALTER |
| ■ DISCONNECT | REVOKE | AUDIT |
| | | NOAUDIT |

ROLLBACK

La orden ROLLBACK, aborta la transacción o cambio, volviendo a la situación de las tablas de la BD desde el último COMMIT.

Notar que si AUTOCOMMIT o la confirmación automática está activada, de nada servirá hacer un ROLLBACK. Es decir para que podamos aplicar ROLLBACK, deberemos tener la confirmación automática desactivada.

ROLLBACK automático

- Si después de haber realizado cambios en nuestras tablas, se produce un fallo del sistema (se va la luz) y no hemos validado el trabajo, Oracle hace un ROLLBACK automático sobre cualquier trabajo no validado. Tendremos que repetir el trabajo desde el último COMMIT

Test ACID en Oracle

- Las bases de datos transaccionales deben cumplir el test ACID. Esto es que debe cumplir los siguientes principios:

ATOMICIDAD

- El principio de atomicidad dice que para que una transacción sea completa, se deben completar todas las partes de la transacción o ninguna de ella. Por ejemplo, si transfieren fondos de una cuenta bancaria a otra, la transacción puede fallar, pero no deben restarse los fondos de una cuenta si no se ha sumado a la otra y al revés.
- La BD debe garantizar ambos procesos o ninguno de ellos. Si alguno de los procesos va mal por cualquier motivo la base de datos debe garantizar que lo que haya podido hacer se puede dar marcha atrás.

CONSISTENCIA

- El principio de consistencia dice que el resultado de un query debe ser consistente con el estado de la base de datos en el momento que el query comenzó. Si se lanza una query sobre una tabla y antes de que termine se actualizan registros en esta tabla, ¿qué valores debe incluir la query: los antiguos o los nuevos?.
- El principio de consistencia dice que la query no debe incluir los valores que fueron insertados o borrados o modificados después de que la query iniciara.

ISOLACIÓN (aislamiento)

- El principio de aislamiento dice que una transacción incompleta (que no se ha hecho commit) debe ser invisible para el resto del mundo.
- Mientras la transacción está en curso, solo la sesión que está ejecutando la transacción tiene permiso de ver los cambios. Esto se debe a que 1º tiene que tener lugar toda la transacción y hasta que esto no se produzca los demás usuarios no deben ver los cambios. Verán versiones anteriores de los datos hasta que la transacción se haya completado.

Bloqueos

- ORACLE controla el aislamiento mediante el uso de bloqueos, hay dos tipos compartidos y exclusivos

BLOQUEOS COMPARTIDOS

Los bloqueos compartidos se adquieren cuando se realiza una sentencia SELECT:

Oracle bloquea la tabla para asegurarse que nadie modifica su estructura, pero no realiza ningún bloqueo sobre los registros que estan consultando

Los bloqueos no previenen a otros usuarios a la hora de realizar consultas o modificaciones, solo sobre cambios en su estructura con:

ALTER TABLE

DROP TABLE

Varios usuarios pueden realizar bloqueos de este tipo sobre lo mismos datos o estructuras

BLOQUEOS EXCLUSIVOS

Los bloqueos exclusivos se realizan cuando utilizamos sentencias DML:

Se aplican a todos los registros afectados por sentencias DML

Los bloqueos exclusivos previenen a otros usuarios de bloquear de forma exclusiva los datos utilizados como parte de una transacción, hasta que realicemos un COMMIT o ROLLBACK

Esto previene a dos usuarios de intentar actualizar los mismos datos al mismo tiempo

Cuando un usuario intenta actualizar datos bloqueados por otro usuario, éste debe esperar hasta que finalice el bloqueo

Oracle permite realizar un bloqueo manualmente con la sentencia:

LOCK TABLE

DURABILIDAD

- El principio de durabilidad implica que una vez que se ha realizado un commit la transacción no se debe perder. Hasta que se haga commit nadie puede ver la transacción, pero una vez hecho commit todo el mundo debe poder verla y la base de datos debe garantizar que el dato no se pierda.