



Unidad Didáctica 5:

LENGUAJE DE MANIPULACIÓN  
DE DATOS: DML

## 1. Introducción

El lenguaje de manipulación de datos (DML) es una pieza central de SQL. El DML (**Data Manipulation Language**) lo forman las instrucciones capaces de modificar los datos de las tablas: agregar filas nuevas a una tabla, modificar las filas existentes de una tabla o eliminar filas existentes de una tabla.

Al conjunto de instrucciones DML que se ejecutan consecutivamente, se las llama **transacciones** y se pueden anular todas ellas o aceptar, ya que una instrucción DML no es realmente efectuada hasta que no se acepta (**COMMIT**).

## 2. Inserción de filas en una tabla

### Adición de una Fila Nueva a una Tabla

70	Public Relations	100	1700
----	------------------	-----	------

Fila  
nueva

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
50	Shipping	124	1500
60	IT	103	1400
80	Sales	149	2500
90	Executive	100	1700
110	Accounting	205	1700
190	Contracting		1700

...inserte una fila  
nueva en la tabla  
DEPARTMENTS...

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
50	Shipping	124	1500
60	IT	103	1400
80	Sales	149	2500
90	Executive	100	1700
110	Accounting	205	1700
190	Contracting		1700
70	Public Relations	100	1700

La adición de datos a una tabla se realiza mediante la instrucción **INSERT**. Su sintaxis fundamental es:

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

La **tabla** representa la tabla a la que queremos añadir la fila y los valores que siguen a **VALUES** son los valores que damos a las distintas columnas de la tabla. Si no se especifica la lista de campos, la lista de valores debe seguir el orden de las columnas según fueron creados (es el orden de columnas según las devuelve el comando **DESCRIBE**).

La lista de campos a rellenar se indica si no queremos rellenar todos los campos. Los campos no rellenados explícitamente con la orden **INSERT**, se rellenan con su valor por defecto (**DEFAULT**) o bien con **NULL** si no se indicó valor alguno. Si no hay ningún valor por defecto definido para la columna, se inserta en su lugar un valor **NULL**. Si algún campo tiene restricción de obligatoriedad (**NOT NULL**), ocurrirá un error si no rellenamos el campo con algún valor.

#### Ejemplo:

La forma de insertar a un cliente en la tabla **clientes** de la base de datos es:

```
INSERT INTO clientes  
VALUES (10, 'ECIGSA', '37.248.573-C', 'ARAGON 242',  
'Barcelona', DEFAULT);
```

o bien:

```
INSERT INTO clientes(nif, nombre_cli, codigo_cli, telefono,  
direccion, ciudad)  
VALUES ('37.248.573-C', 'ECIGSA', 10, DEFAULT, 'ARAGON 242',  
'Barcelona');
```

Con esta sintaxis sólo se inserta una fila cada vez.

Los valores de tipo carácter y de fecha se deben escribir entre comillas simples. Los valores numéricos no se recomienda escribirlos entre comillas porque puede tener lugar una conversión implícita para los valores numéricos asignados a columnas de tipo **NUMBER**.

A la hora de insertar valores nulos tenemos dos métodos: omitir la columna en la lista de columnas o especificar la palabra clave **NULL** en la lista de **VALUES**. Para cadenas de caracteres y fechas podemos especificar la cadena vacía (' ') en la lista de **VALUES**.

También podemos utilizar funciones para introducir valores especiales en la tabla: la función **SYSDATE**, para registrar la fecha y hora actual o la función **USER** para registrar el nombre del usuario actual.

## 2.1. Insertar filas desde otra tabla

Podemos utilizar la sentencia `INSERT` para agregar filas a una tabla desde otras tablas existentes.

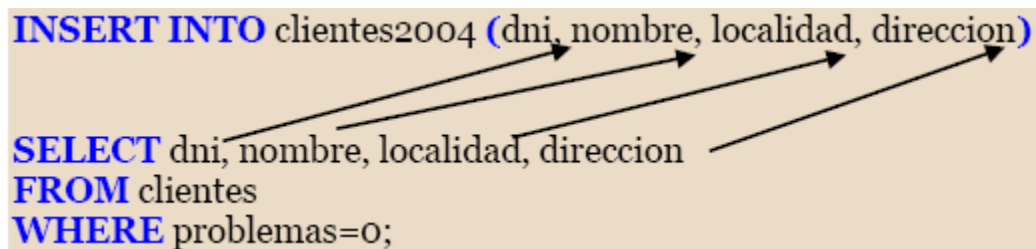
La sintaxis es :

```
INSERT INTO tabla [(columna1[,columna2 ...])  
subquery;
```

donde:

*subquery* es la subconsulta que devuelve filas a la tabla.

El numero de columnas y sus tipos de dato en la lista de columnas de la cláusula `INSERT` debe coincidir con el número de valores y sus tipos de dato en la subconsulta.



```
INSERT INTO clientes2004 (dni, nombre, localidad, direccion)  
  
SELECT dni, nombre, localidad, direccion  
FROM clientes  
WHERE problemas=0;
```

### 3. Modificación de filas de una tabla

Si quisiéramos modificar los valores de algunas filas de una tabla, tendríamos que utilizar la sentencia **UPDATE**. A continuación presentamos su formato:

```
UPDATE tabla  
SET columna = {expresión|DEFAULT|NULL}  
[, columna = {expr|DEFAULT|NULL} ...]  
WHERE condiciones;
```

## Cambio de los Datos de una Tabla

### EMPLOYEES

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	HIRE_DATE	JOB_ID	SALARY	DEPARTMENT_ID	COMMISSION_F
100	Steven	King	SKING	17-JUN-87	AD_PRES	24000	90	
101	Neena	Kochhar	NKOCHHAR	21-SEP-89	AD_VP	17000	90	
102	Lex	De Haan	LDEHAAN	13-JAN-93	AD_VP	17000	90	
103	Alexander	Hunold	AHUNOLD	03-JAN-90	IT_PROG	9000	60	
104	Bruce	Ernst	BERNST	21-MAY-91	IT_PROG	6000	60	
107	Diana	Lorentz	DLORENTZ	07-FEB-99	IT_PROG	4200	60	
124	Kevin	Mourgos	KMOURGOS	16-NOV-99	ST_MAN	5800	50	

Actualice las filas de la tabla **EMPLOYEES**.

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	HIRE_DATE	JOB_ID	SALARY	DEPARTMENT_ID	COMMISSIO
100	Steven	King	SKING	17-JUN-87	AD_PRES	24000	90	
101	Neena	Kochhar	NKOCHHAR	21-SEP-89	AD_VP	17000	90	
102	Lex	De Haan	LDEHAAN	13-JAN-93	AD_VP	17000	90	
103	Alexander	Hunold	AHUNOLD	03-JAN-90	IT_PROG	9000	30	
104	Bruce	Ernst	BERNST	21-MAY-91	IT_PROG	6000	30	
107	Diana	Lorentz	DLORENTZ	07-FEB-99	IT_PROG	4200	30	
124	Kevin	Mourgos	KMOURGOS	16-NOV-99	ST_MAN	5800	50	

Con esta sentencia podemos modificar más de una fila si es necesario. En general, utilizar la clave primaria para identificar una sola fila. Si usamos otras columnas se pueden actualizar varias filas de forma inesperada.

Si se omite la cláusula **WHERE** se modifican todas las filas.

**Ejemplo:**

Supongamos que queremos incrementar el sueldo de todos los empleados del proyecto 2 en 1.000 euros. La modificación a ejecutar sería:

```
UPDATE empleados
SET sueldo = sueldo + 1000
WHERE num_proyec = 2;
```

Notemos que el proyecto número 2 podría tener a más de un empleado asignado y, por lo tanto, se modificaría la columna sueldo, de más de una fila con una sola sentencia.

### 3.1. Actualización de varias columnas con subconsultas

Podemos actualizar varias columnas en la cláusula **SET** de una sentencia **UPDATE** escribiendo varias subconsultas.

**Ejemplo:**

Actualizar el oficio y el salario del empleado 114 para que coincida con el del empleado 205.

```
UPDATE EMPLE
SET OFICIO =(SELECT OFICIO
              FROM EMPLE
              WHERE EMP_NO=205),
      SALARIO =(SELECT SALARIO
              FROM EMPLE
              WHERE EMP_NO=205)
WHERE EMP_NO = 114;
```

## 4. Borrado de filas de una tabla

Para borrar valores de algunas filas de una tabla podemos utilizar la sentencia **DELETE**. Su formato es el siguiente:

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

En cambio, si lo que quisiéramos conseguir es borrar todas las filas de una tabla, entonces sólo tendríamos que poner la sentencia **DELETE FROM**, sin **WHERE**.

## Eliminación de una Fila de una Tabla

### DEPARTMENTS

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
30	Purchasing		
100	Finance		
50	Shipping	124	1500
60	IT	103	1400

### Suprima una fila de la tabla DEPARTMENTS

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
30	Purchasing		
50	Shipping	124	1500
60	IT	103	1400

**Ejemplo:**

Podemos dejar la tabla proyectos sin ninguna fila:

```
DELETE FROM proyectos;
```

En nuestra base de datos, borrar los proyectos del cliente 2 se haría de la forma que mostramos a continuación:

```
DELETE FROM proyectos  
WHERE codigo_cliente = 2;
```

## 5. Transacciones de Base de Datos



Oracle Server asegura la consistencia de datos basándose en **transacciones**. Las transacciones le ofrecen más flexibilidad y control al cambiar datos y aseguran una consistencia de datos en caso de fallo del proceso de usuario o fallo del sistema.

Las transacciones constan de sentencias DML que constituyen un cambio consistente en los datos. Por ejemplo, una transferencia de fondos entre dos cuentas debe incluir el débito a una cuenta y el cargo a otra por el mismo importe. Las dos acciones se deben realizar correcta o incorrectamente al mismo tiempo, de forma que el crédito no se debe validar sin el débito.

### Tipos de Transacción

Tipo	Descripción
Lenguaje de manipulación de datos (DML)	Consta de un número variable de sentencias DML que Oracle Server trata como entidad única o como unidad de trabajo lógica.
Lenguaje de definición de datos (DDL)	Consta de una sola sentencia DDL.
Lenguaje de control de datos (DCL)	Consta de una sola sentencia DCL.

## 5.1. ¿Cuándo Empieza y Termina una Transacción?

Una transacción comienza cuando se encuentra la primera sentencia DML y termina cuando se produce una de las siguientes opciones:

- Se emite una sentencia **COMMIT** o **ROLLBACK**.
- Se emite una sentencia DDL, como por ejemplo, **CREATE**.
- Se emite una sentencia DCL.
- El usuario sale de SQL\*Plus.
- Se produce un fallo de una máquina o en el sistema.

Cuando termina una transacción, la siguiente sentencia SQL ejecutable inicia automáticamente la siguiente transacción.

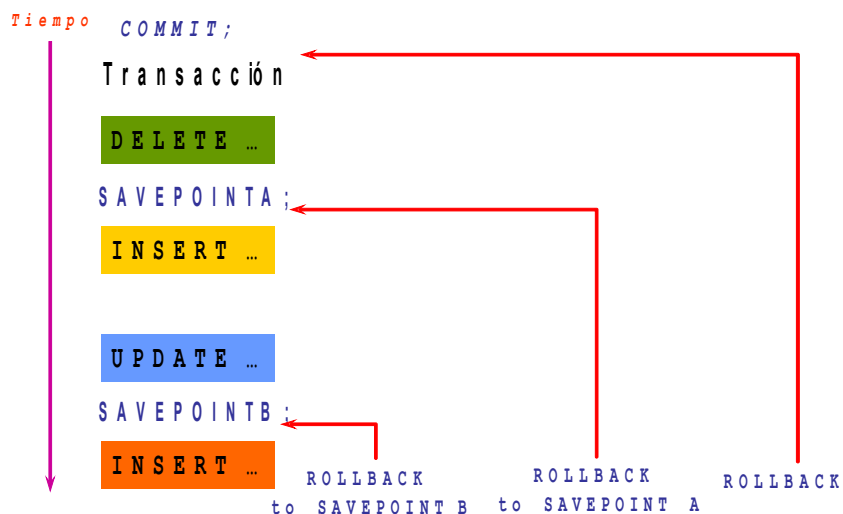
Se valida automáticamente una sentencia DDL o una sentencia DCL y, por lo tanto, finaliza implícitamente una transacción.

## 5.2. Sentencias Explícitas de Control de Transacciones

Puede controlar la lógica de las transacciones utilizando las sentencias **COMMIT**, **SAVEPOINT** y **ROLLBACK**.

Sentencia	Descripción
<b>COMMIT</b>	Termina la transacción actual haciendo permanentes todos los cambios de datos pendientes.
<code>SAVEPOINT name</code>	Marca un punto de grabación dentro de la transacción actual.
<b>ROLLBACK</b>	<b>ROLLBACK</b> termina la transacción actual desechando todos los cambios de datos pendientes.
<code>ROLLBACK TO SAVEPOINT name</code>	<code>ROLLBACK TO SAVEPOINT</code> realiza rollback de la transacción actual al punto de grabación especificado, desechando así los cambios o los puntos de grabación creados después del punto de grabación al que está realizando rollback. Si omite la cláusula <code>TO SAVEPOINT</code> , la sentencia <code>ROLLBACK</code> realiza rollback de toda la transacción. Como los puntos de grabación son lógicos, no hay ninguna forma de enumerar los puntos de grabación que ha creado.

## Control de Transacciones



### 5.3. Procesamiento Implícito de Transacciones

La siguiente tabla muestra cuando se produce un procesamiento implícito de una transacción, ya sea una validación automática (COMMIT) o un ROLLBACK automático:

Estado	Circunstancias
Validación automática	Se emite una sentencia DDL o DCL. Salida normal de SQL*Plus, sin emitir explícitamente comandos COMMIT o ROLLBACK.
Rollback automático	Terminación anormal de SQL*Plus o fallo del sistema.

En SQL\*Plus está disponible un tercer comando. El comando **AUTO COMMIT** se puede activar o desactivar. Si está *activado*, cada sentencia DML individual se valida en cuanto se ejecuta. No puede realizar rollback de los cambios. Si está *desactivado*, la sentencia COMMIT todavía se puede emitir explícitamente. Además, la sentencia COMMIT se emite cuando se emite una sentencia DDL o al salir de SQL\*Plus.

## Fallos del Sistema:

Cuando se interrumpe una transacción por un fallo del sistema, se realiza rollback automáticamente a toda la transacción. Esto evita que el error produzca cambios no deseados en los datos y devuelve las tablas al estado que tenían en el momento de la última validación. De esta forma, Oracle Server protege la integridad de las tablas.

Con SQL\*Plus, se consigue salir normalmente escribiendo el comando EXIT en el prompt. Cerrar la ventana se interpreta como una salida anormal.

## 5.4. Validación de Cambios

Todo cambio de datos realizado durante la transacción es temporal hasta que se valida la transacción.

El estado de los datos antes de que se emitan sentencias COMMIT o ROLLBACK:

- Las operaciones de manipulación de datos afectan principalmente al buffer de la base de datos; por lo tanto, el estado anterior de los datos se puede recuperar.
- El usuario actual puede revisar los resultados de las operaciones de manipulación de datos consultando las tablas.
- Otros usuarios no pueden ver los resultados de las operaciones de manipulación de datos realizados por el usuario actual. Oracle Server establece la consistencia de lectura para asegurar que cada usuario ve los datos como eran en la última validación.
- Las filas afectadas se bloquean; otros usuarios no pueden cambiar los datos de las filas afectadas.

Podemos hacer que todos los cambios pendientes sean permanentes utilizando la sentencia COMMIT. Tras una sentencia COMMIT:

- Los cambios de datos se escriben en la base de datos.
- El estado anterior de los datos se pierde permanentemente.
- Todos los usuarios pueden ver los resultados de la transacción.
- Los bloqueos de las filas afectadas se liberan; las filas están ahora disponibles para que otros usuarios realicen nuevos cambios de datos.
- Todos los puntos de grabación se borran.

Podemos deshacer todos los cambios pendientes utilizando la sentencia `ROLLBACK`. Tras una sentencia `ROLLBACK`:

- Los cambios de datos se deshacen.
- El estado anterior de los datos se restaura.
- Los bloqueos de las filas afectadas se liberan.

**Ejemplo:**

Al intentar eliminar un registro de la tabla `TEST`, puede vaciar accidentalmente la tabla. Puede corregir el error, volver a emitir la sentencia correcta y hacer que el cambio de datos sea permanente.

```
DELETE FROM test;  
25 rows deleted.
```

```
ROLLBACK;  
Rollback complete.
```

```
DELETE FROM test  
WHERE id = 100;  
1 row deleted.
```

```
SELECT *  
FROM test  
WHERE id = 100;  
No rows selected.
```

```
COMMIT;  
Commit complete.
```

## 6. Consistencia de Lectura

Los usuarios de la base de datos acceden a ésta de dos formas:

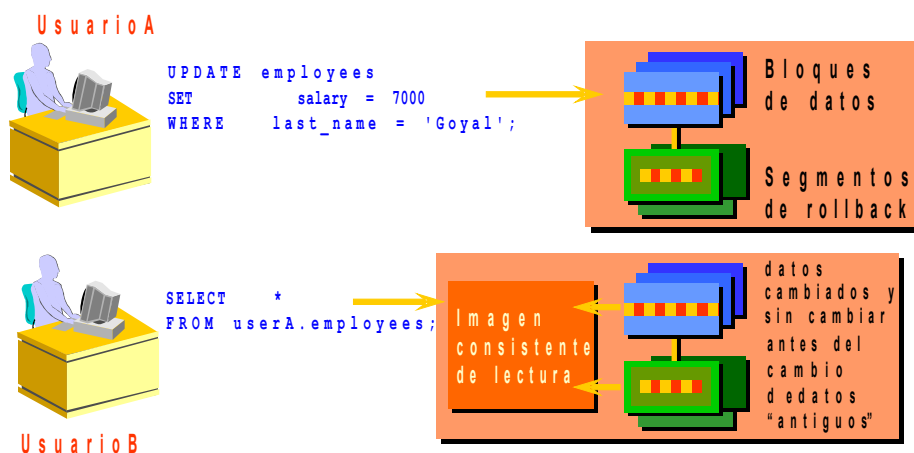
- Operaciones de lectura (sentencia `SELECT`).
- Operaciones de escritura (sentencias `INSERT`, `UPDATE`, `DELETE`).

Necesitamos consistencia de lectura para que:

- El escritor y el lector de la base de datos se aseguren una visualización consistente de los datos.
- Los lectores no visualicen los datos que están en proceso de cambio.
- Los escritores se aseguren de que los cambios en la base de datos se realizan de forma consistente.
- Los cambios realizados por un escritor no interrumpen a los que está haciendo otro escritor ni entren en conflicto con ellos.

El objetivo de la consistencia de lectura es asegurar que cada usuario ve los datos tal como eran en la última validación, antes de que comenzara una operación DML.

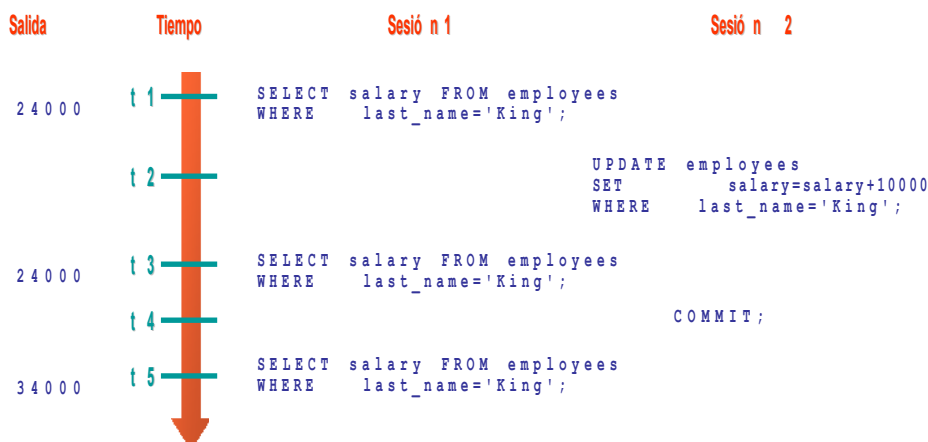
### Implementación de Consistencia de Lectura



La consistencia de lectura es una implementación automática. Mantiene una copia parcial de la base de datos en segmentos deshacer.

- Cuando se realiza una operación de inserción, actualización o supresión en la base de datos, Oracle Server realiza una copia de los datos antes de que se cambien y los escribe en un *segmento deshacer*.
- Todos los lectores, excepto el que emitió el cambio, seguirán viendo la base de datos como existía antes de que comenzaran los cambios; ven la "instantánea" del segmento de rollback de los datos.
- Antes de que se validen los cambios en la base de datos, sólo el usuario que está modificando los datos ve la base de datos con las modificaciones; los demás ven la instantánea en el segmento deshacer. Esto garantiza que los lectores de los datos lean datos consistentes que no están sufriendo ningún cambio actualmente.
- Cuando se valida una sentencia DML, el cambio realizado en la base de datos se hace visible para cualquier persona que ejecute una sentencia SELECT. El espacio ocupado por los datos *antiguos* en el archivo del segmento deshacer se libera para su nuevo uso.

## Ejemplo de Consistencia de Lectura



Si se realiza rollback de la transacción, los cambios se deshacen:

- La versión original (más antigua) de los datos del segmento deshacer se vuelve a escribir en la tabla.
- Todos los usuarios ven la base de datos como existía antes de que comenzara la transacción.

## 7. ¿Qué son los Bloqueos?

Los bloqueos son mecanismos que evitan una interacción destructiva entre las transacciones que acceden al mismo recurso, ya sea un objeto de usuario (como tablas o filas) o un objeto de sistema no visible a los usuarios (como estructuras de datos compartidos y filas del diccionario de datos).

### **Cómo Bloquea Datos la Base de Datos Oracle**

El bloqueo de Oracle se realiza automáticamente y no requiere acción del usuario. El bloqueo implícito se produce para las sentencias SQL según sea necesario, dependiendo de la acción solicitada. Este bloqueo se produce para todas las sentencias SQL excepto SELECT.

Los usuarios también pueden bloquear los datos manualmente, lo que se denomina bloqueo explícito.

### **Bloqueo DML**

Al realizar operaciones del lenguaje de manipulación de datos (DML), Oracle Server proporciona simultaneidad de datos a través del bloqueo DML, que se produce a dos niveles:

- Un bloqueo compartido se obtiene automáticamente a nivel de tabla durante operaciones DML: Con el modo de bloqueo compartido, varias transacciones pueden adquirir bloqueos compartidos sobre el mismo recurso.
- Un bloqueo exclusivo se adquiere automáticamente para cada fila modificada por una sentencia DML. Los bloqueos exclusivos evitan que otras transacciones cambien la fila hasta que la transacción se valida o se realiza rollback. Este bloqueo asegura que ningún otro usuario puede modificar la misma fila al mismo tiempo ni sobrescribir los cambios que aún no han sido validados por otro usuario.
- Los bloqueos DDL se producen al modificar un objeto de base de datos como una tabla.