

9

Manipulando Datos

Objetivos

Tras terminar esta lección seremos capaces de hacer lo siguiente:

Describir cada sentencia DML

- **Insertar filas en una tabla**
- **Actualizar las filas de una tabla**
- **Borrar filas de una tablas**
- **Controlar transacciones**

Lenguaje de Definición de Datos

- Una sentencia DML se ejecuta cuando:
 - Añadimos nuevas filas a la tabla
 - Se modifican filas existentes en la tabla.
 - Se borran filas existentes en la tabla
- Una **transacción** consiste en una colección de sentencias DML que forman una única unidad de trabajo.

Añadir una nueva fila a una tabla

50	DEVELOPMENT	
----	-------------	--

Nueva fila

DEPT

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

“...inserta una nueva fila en la tabla

DEPT ...”

DEPT

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON
50	DEVELOPMENT	

La sentencia INSERT

- Se añaden nuevas filas en la tabla mediante el uso de esta sentencia.

```
INSERT INTO  table [(column [, column...])]  
VALUES      (value [, value...]);
```

- Solo una fila se puede insertar a la vez mediante el uso de esta sentencia.

Insertando Filas Nuevas

- Se inserta una nueva fila conteniendo valores para cada columna.
- Se listan los valores en el orden de las columnas de la tabla.
- Se pueden listar las columnas opcionalmente en INSERT.

```
SQL> INSERT INTO      dept (deptno, dname, loc)
      2  VALUES      (50, 'DEVELOPMENT', 'DETROIT') ;
1 row created.
```

- Las cadenas de caracteres y las fechas se encierran entre comillas simples.

Inserción de filas con valores nulos

- **Implicitamente:** Se omite la columna en la lista de columnas.

```
SQL> INSERT INTO      dept (deptno, dname)
      2  VALUES        (60, 'MIS') ;
1 row created.
```

- **Explicitamente:** Se especifica NULL.

```
SQL> INSERT INTO      dept
      2  VALUES        (70, 'FINANCE', NULL) ;
1 row created.
```

Inserción de valores especiales

Ejemplo de uso de SYSDATE en INSERT

```
SQL> INSERT INTO      emp (empno, ename, job,  
  2                   mgr, hiredate, sal, comm,  
  3                   deptno)  
  4 VALUES           (7196, 'GREEN', 'SALESMAN',  
  5                   7782, SYSDATE, 2000, NULL,  
  6                   10) ;
```

1 row created.

Insertión de valores específicos de fecha

- Adición de un nuevo empleado.

```
SQL> INSERT INTO emp
  2  VALUES      (2296, 'AROMANO', 'SALESMAN', 7782,
  3                TO_DATE('FEB 3, 97', 'MON DD, YY'),
  4                1300, NULL, 10);
1 row created.
```

- Verificación de la adición.

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
-----	-----	-----	----	-----	-----	-----	-----
2296	AROMANO	SALESMAN	7782	03-FEB-97	1300		10

Copia de filas desde otra tabla

- Se escribe la sentencia SELECT de la subconsulta.

```
SQL> INSERT INTO managers(id, name, salary, hiredate)
2      SELECT empno, ename, sal, hiredate
3      FROM    emp
4      WHERE   job = 'MANAGER';
3 rows created.
```

- No se usa la cláusula VALUES.
- Debe coincidir el número de columnas de INSERT con las de la subconsulta.

Modificación de datos en una tabla

EMP

EMPNO	ENAME	JOB	...	DEPTNO
7839	KING	PRESIDENT		10
7698	BLAKE	MANAGER		30
7782	CLARK	MANAGER		10
7566	JONES	MANAGER		20
...				

“...actualiza una fila en la tabla EMP ...”



EMP

EMPNO	ENAME	JOB	...	DEPTNO
7839	KING	PRESIDENT		10
7698	BLAKE	MANAGER		30
7782	CLARK	MANAGER		20
7566	JONES	MANAGER		20
...				

La sentencia UPDATE

- Se modifican las filas existentes en la tabla mediante la sentencia UPDATE.

```
UPDATE      table  
SET         column = value [, column = value, ...]  
[WHERE      condition];
```

- Se puede modificar más de una fila a la vez.

Modificación de datos en una tabla

- Se especifican la filas a modificar cuando incluimos la cláusula WHERE.

```
SQL> UPDATE    emp
  2  SET        deptno = 20
  3  WHERE      empno = 7782;
1 row updated.
```

- Todas las filas de la tabla se modifican si omitimos la cláusula WHERE.

```
SQL> UPDATE    employee
  2  SET        deptno = 20;
14 rows updated.
```

Actualización con una Multiple-Column Subquery

Poner el trabajo y el departamento del empleado 7698 con los correspondientes valores del empleado 7499.

```
SQL> UPDATE emp
  2 SET      (job, deptno) =
  3          (SELECT job, deptno
  4             FROM   emp
  5             WHERE  empno = 7499)
  6 WHERE empno = 7698;
1 row updated.
```

Actualización de filas basada en otra tabla

Se usan subconsultas en la orden **UPDATE** para actualizar filas en una tabla basada en valores de otra tabla.

```
SQL> UPDATE   employee
  2  SET      deptno = (SELECT   deptno
  3                      FROM     emp
  4                      WHERE    empno = 7788)
  5  WHERE    job      = (SELECT   job
  6                      FROM     emp
  7                      WHERE    empno = 7788) ;
```

2 rows updated.

ACTUALIZACIÓN DE FILAS:

Error de integridad referencial

```
SQL> UPDATE    emp
      2  SET      deptno = 55
      3  WHERE    deptno = 10;
```

```
UPDATE emp
      *
```

ERROR at line 1:

ORA-02291: integrity constraint (USR.EMP_DEPTNO_FK)
violated - parent key not found

Department number 55 does not exist

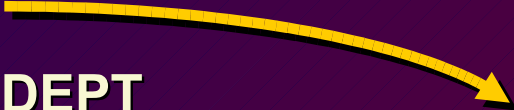
Borrar una fila de una tabla

DEPT

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON
50	DEVELOPMENT	DETROIT
60	MIS	
...		

“...borra una fila de la tabla DEPT ...”

DEPT



DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON
60	MIS	
...		

La sentencia DELETE

Podemos borrar filas de una tabla mediante el uso de la sentencia DELETE.

```
DELETE FROM      table  
[WHERE          condition] ;
```

Borrado de filas de una tabla

- Se borran filas específicas cuando incluimos la cláusula WHERE.

```
SQL> DELETE FROM      department
      2  WHERE          dname = 'DEVELOPMENT';
1 row deleted.
```

- Todas las filas de una tabla se borran si omitimos la cláusula WHERE .

```
SQL> DELETE FROM      department;
4 rows deleted.
```

Borrar filas basadas en otra tabla

Se usan subconsultas en la sentencia **DELETE** para borrar filas en una tabla basandonos en valores de otra tabla.

```
SQL> DELETE FROM      employee
      2  WHERE          deptno =
      3                  (SELECT      deptno
      4                        FROM      dept
      5                        WHERE      dname = 'SALES' ) ;
6 rows deleted.
```

BORRADO DE FILAS:

Error de integridad referencial

```
SQL> DELETE FROM dept
      2 WHERE deptno = 10;
```

```
DELETE FROM dept
          *
```

ERROR at line 1:

ORA-02292: integrity constraint (USR.EMP_DEPTNO_FK)
violated - child record found

*You cannot delete a row
that contains a primary key
that is used as a foreign key
in another table.*

Transacciones

Consisten en un grupo de sentencias que garantizan cambios consistentes en los datos. Pueden ser de los siguientes tipos:

- **Grupo de sentencias DML que el servidor ORACLE trata como una única unidad de trabajo.**
- **Una sentencia DDL**
- **Una sentencia DCL**

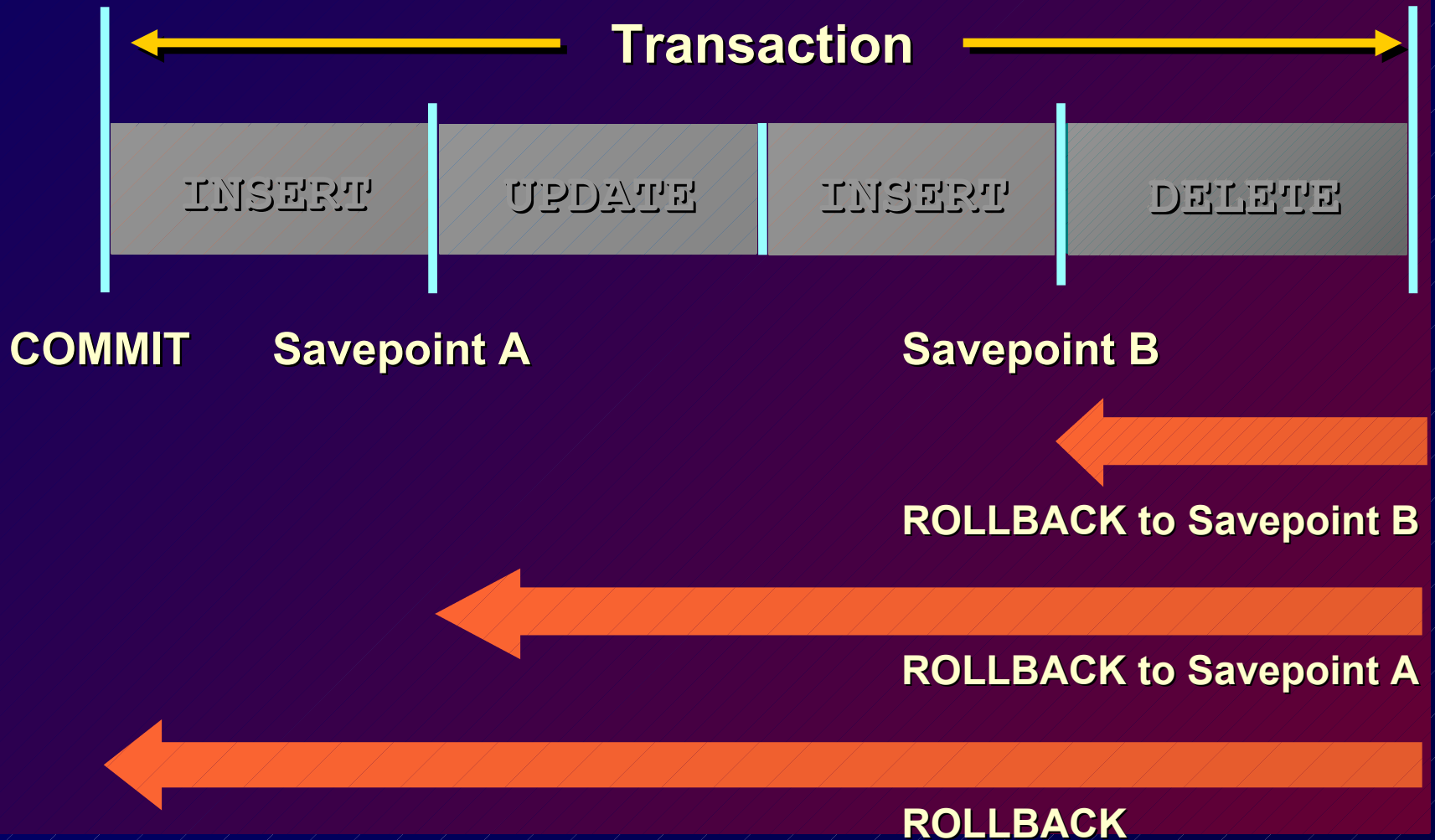
Transacciones

- **Comienzan cuando se encuentra la primera sentencia ejecutable SQL.**
- **Finalizan con uno de los siguientes eventos:**
 - **Se ejecuta COMMIT o ROLLBACK**
 - **Las sentencias DDL o DCL fuerzan a que se ejecute un COMMIT (commit implícito)**
 - **El usuario sale de SQL*PLUS**
 - **Fallos de sistema.**

Ventajas del uso de COMMIT y ROLLBACK

- **Aseguran la consistencia de los datos**
- **Permite ver los cambios en los datos antes de hacerlos definitivos.**
- **Agrupar lógicamente las operaciones efectuadas.**

Control de transacciones



Procesamiento Implícito de Transacciones

- Un commit automático se produce bajo una de las siguientes circunstancias:
 - Ejecución de una sentencia DDL
 - Ejecución de una sentencia DCL
 - Salida normal de SQL*Plus, sin ejecución explícita de COMMIT o ROLLBACK.
- Un rollback automático se produce por una terminación anormal de SQL*Plus o un fallo del sistema.

Estado de los datos antes de COMMIT o ROLLBACK

- El estado previo de los datos puede ser recuperado.
- El usuario en curso puede revisar el resultado de las operaciones DML operations usando la sentencia SELECT .
- Otros usuarios no pueden ver los resultados de las sentencias DML ejecutadas por el usuario en curso.
- Las filas afectadas están *bloqueadas*; otros usuarios no pueden cambiar los datos de las filas afectadas.

Estado de los datos tras COMMIT

- Los cambios en los datos se hacen permanentes en la base de datos.
- El estado previo de los datos se pierde definitivamente.
- Todos los usuarios pueden ver los resultados.
- El bloqueo sobre las filas afectadas se libera; las filas están disponibles para otros datos para ser manipuladas.
- Todos los savepoints son borrados.

Uso de Commit

- Efectuar los cambios.

```
SQL> UPDATE    emp
      2  SET      deptno = 10
      3  WHERE    empno = 7782;
1 row updated.
```

- Ejecución de los cambios.

```
SQL> COMMIT;
Commit complete.
```

Estado de los datos tras ROLLBACK

- Se descartan todos los cambios pendientes mediante el uso de ROLLBACK .
- Los cambios no se efectúan.
- Se restaura el estado previo de los datos.
- El bloqueo sobre las filas afectadas se libera.

```
SQL> DELETE FROM      employee;
```

```
14 rows deleted.
```

```
SQL> ROLLBACK;
```

```
Rollback complete.
```

Deshaciendo Cambios hasta un marcador

- Se crea un marcador en un transacción mediante el uso de **SAVEPOINT**.
- Se vuelve atrás hasta ese marcador mediante el uso de **ROLLBACK TO SAVEPOINT**.

```
SQL> UPDATE ...  
SQL> SAVEPOINT update_done;  
Savepoint created.  
SQL> INSERT ...  
SQL> ROLLBACK TO update_done;  
Rollback complete.
```

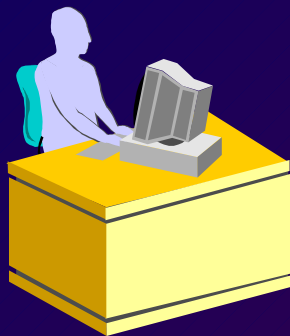
Statement-Level Rollback

- Si una sentencia DML falla durante la ejecución solo esa sentencia es “rolled back”.
- El servidor Oracle Server implementa un savepoint implícito.
- Se retienen todos los otros cambios.
- El usuario puede terminar explícitamente la transacción mediante la ejecución de COMMIT o ROLLBACK.

Consistencia de lectura (Read consistency)

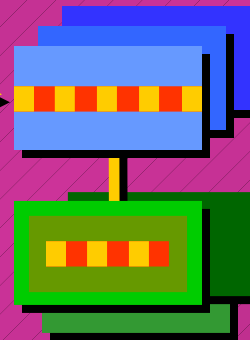
- **Garantiza una vista consistente de los dtos en todo momento.**
- **Los cambios efectuados por un usuario no entran en conflicto con los hechos por otro usuario.**
- **Se asegura que en los mismos datos cada usuario vea los datos como estaban tras el último COMMIT antes de que empezara una operación DML.**

Implementación of Read Consistency



User A

```
UPDATE emp  
SET sal = 2000  
WHERE ename =  
'SCOTT';
```



Data
blocks

Rollback
segments

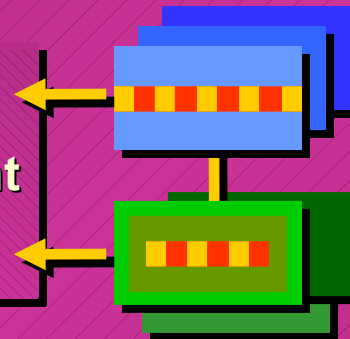


User B

```
SELECT *  
FROM emp;
```



Read
consistent
image



changed
and
unchanged
data

before
change
"old" data