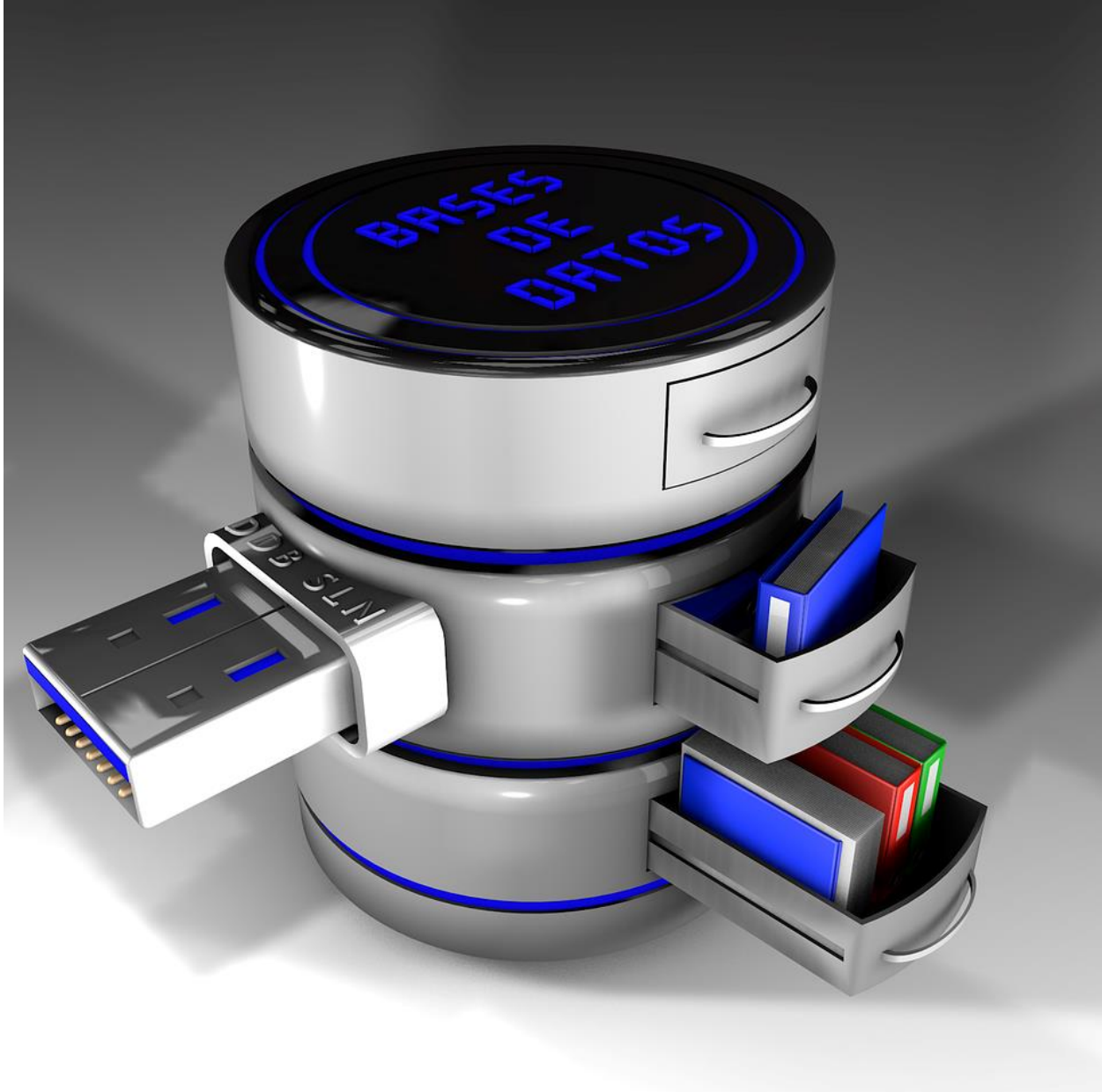
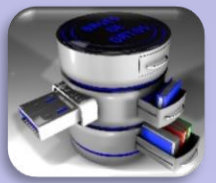
UNITAT DIDÀCTICA 12

SQL. Llenguatge DML



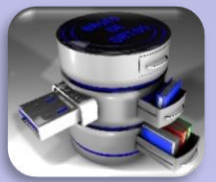
Índex



Sentència INSERT

El llenguatge de manipulació de dades (DML) és una part fonamentalment de SQL. Per a agregar, actualitzar o suprimir les dades de la base de dades, execute una sentència DML. La recopilació de sentències DML que formen una unitat lògica de treball es denomina **transacció**.

Pense en una base de dades bancària. Quan un client del banc transfereix diners del seu compte d'estalvi a un compte corrent, la transacció pot constar de les següents tres accions diferents: reduir el compte d'estalvi, augmentar el compte corrent i registrar la transacció en el diari de transaccions. El servidor d'Oracle ha de garantir que s'executen les tres sentències SQL per a mantindre el balanç correcte dels comptes. Si alguna cosa impedeix que una de les sentències de la transacció s'execute, les altres sentències de la transacció s'han de desfer.



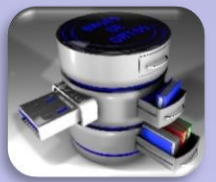
Sentència INSERT

Pots agregar noves files a una taula emetent la sentència INSERT.

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

Amb aquesta instrucción només s'insertarà una fila de dades.

Com que pot afegir una nova fila que continga els valors de cada columna, no és necessària la llista de columnes en la clàusula INSERT. No obstant això, si no utilitza la llista de columnes, els valors s'han de mostrar segons l'ordre per defecte de les columnes en la taula i s'ha de proporcionar un valor per a cada columna.



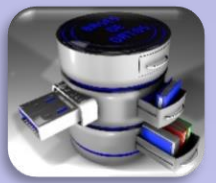
Sentència INSERT

Perquè estiga tot més clar, és preferible que s'utilitze la llista de columnes en la clàusula INSERT. Recorda que els valors de caràcters i data s'utilitzen cometes simples; no obstant això, no es recomana utilitzar-les en els valors numèrics.

```
DESC dept80;
```

```
Nombre      Nulo     Tipo
----------- -------- ------------
LAST_NAME   NOT NULL VARCHAR2(30)
FIRST_NAME           VARCHAR2(20)
SALARY               NUMBER(8,2)
ANSAL                NUMBER
```

Es recomana l'ús de la llista de columnes perquè fa la sentència INSERT més llegible i fiable o menys procliu a errors.

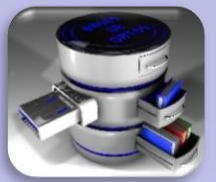


Sentència INSERT

Perquè estiga tot més clar, és preferible que s'utilitze la llista de columnes en la clàusula INSERT. Recorda que els valors de caràcters i data s'utilitzen cometes simples; no obstant això, no es recomana utilitzar-les en els valors numèrics.

```
INSERT INTO dept80 (last_name, first_name, salary, ansal)
VALUES ('Garcia','Ana', 1580, 22000);
```

LAST_NAME	FIRST_NAME	SALARY	ANSAL
Garcia	Ana	1580	22000
Kumar	Sundita	6100	73200



Sentència INSERT

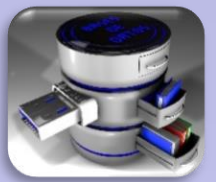
```
INSERT INTO dept80 (last_name, first_name, salary)
VALUES ('Giménez','Juan', 1300);
```

Quan s'ometen alguns valors, cal assegurar-se que aqueixa columna permet inserir valors nuls

LAST_NAME	FIRST_NAME	SALARY	ANSAL
Giménez	Juan	1300	(null)
Garcia	Ana	1580	22000

```
INSERT INTO dept80
VALUES ('Pérez','María', 1700, 24000);
```

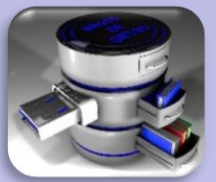
LAST_NAME	FIRST_NAME	SALARY	ANSAL
Giménez	Juan	1300	(null)
Garcia	Ana	1580	22000
Pérez	María	1700	24000



Sentència INSERT

Els errors comuns que es produeixen durant l'entrada de l'usuari es comproven en el següent ordre:

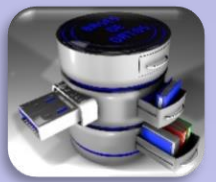
- ✓ Valor obligatori que falta per a una columna NOT NULL
- ✓ Valor duplicat que viola qualsevol restricció de clau única o primària
- ✓ Qualsevol valor que viole una restricció CHECK
- ✓ Manteniment de la integritat referencial per a la restricció de clau aliena
- ✓ No coincidències de tipus de dada o valors massa amples per a la columna



Sentència INSERT

Quan s'introdueixen les dades és molt important tindre en compte les restriccions de cadascun dels atributs.

En el cas de les claus alienes cal tindre en compte que SEMPRE fan referència a una dada que existeix en una altra taula, per tant, quan introduïm una dada en una columna amb restricció FOREING KEY la dada ha d'existir prèviament en l'atribut de la taula referenciada, en cas contrari es produirà una violació de la restricció.

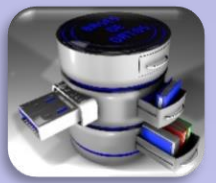


Sentència INSERT

Pot utilitzar funcions per a introduir valors especials en la taula, com per exemple SYSDATE.

```
INSERT INTO emp_copy (cod_employee, first_name, last_name, email, hire_date, job_id)
VALUES (86, 'María','López', 'maria.lopez@empresa.es',SYSDATE, 'RH_CAP');
```

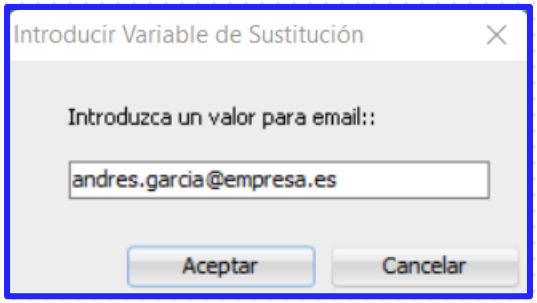
COD_EMPLOYEE	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID
86	María	López	maria.lopez@empresa.es	(null)	30/04/22	RH_CAP



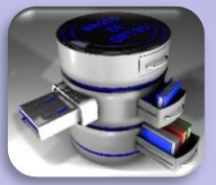
Sentència INSERT

També es pot utilitzar la variable de substitució per a sol·licitar a l'usuari inserir les dades mitjançant les finestres emergents

```
INSERT INTO emp_copy (cod_employee, first_name, last_name, email, hire_date, job_id)
VALUES (87, 'Andrés','García', '&email',SYSDATE, 'RH_CAP');
```



COD_EMPLOYEE	FIRST_NAME	LAST_NAME	EMAIL
86	María	López	maria.lopez@empresa.es
87	Andrés	García	andres.garcia@empresa.es



Sentència INSERT

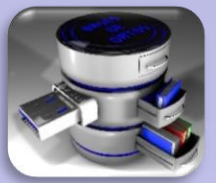
També es poden inserir valors a través d'una subconsulta

Has de tindre en compte que:

- No utilitza la clàusula VALUES
- Cal fer coincidir el nombre de columnes de INSERT i el de SELECT
- Inserida totes les files que retorna la subconsulta

```
INSERT INTO dept80
SELECT last_name, first_name, salary, salary*12, department_id
FROM employees
WHERE department_id = 50;
```

LAST_NAME	FIRST_NAME	SALARY	ANSAL	DEPARTMENT_ID
Johnson	Charles	6200	74400	80
Weiss	Matthew	8000	96000	50
Fripp	Adam	8200	98400	50



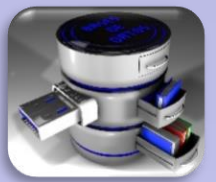
Sentència UPDATE

Quan ja tenim les dades en la taula, és possible que vulguem modificar algun d'ells. Per a aqueixa acció utilitzarem la sentència UPDATE.

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

Si s'especifica la clàusula WHERE es modificaran totes les files que complisquen la condició.

Si no s'especifica, es modificaran totes les files de la taula.



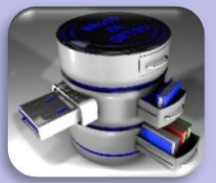
Sentència UPDATE

```
SELECT cod_employee, last_name, first_name, salary
FROM emp_copy;
```

COD_EMPLOYEE	LAST_NAME	FIRST_NAME	SALARY
100	King	Steven	24000

```
UPDATE emp_copy
SET salary = 26000
WHERE cod employee = 100;
```

COD_EMPLOYEE	LAST_NAME	FIRST_NAME	SALARY
100	King	Steven	26000



Sentència UPDATE

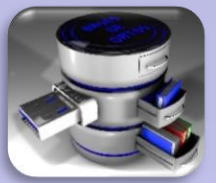
```
SELECT *
FROM dept80;
```

LAST_NAME	FIRST_NAME	SALARY	ANSAL	DEPARTMENT_ID
Hutton	Alyssa	8800	105600	80
Taylor	Jonathon	8600	103200	80
Livingston	Jack	8400	100800	80
Johnson	Charles	6200	74400	80
Weiss	Matthew	8000	96000	50
Fripp	Adam	8200	98400	50
Kaufling	Payam	7900	94800	50

```
UPDATE emp_copy
SET department_id=80;
```

```
79 filas actualizadas.
```

LAST_NAME	FIRST_NAME	SALARY	ANSAL	DEPARTMENT_ID
Hutton	Alyssa	8800	105600	80
Taylor	Jonathon	8600	103200	80
Livingston	Jack	8400	100800	80
Johnson	Charles	6200	74400	80
Weiss	Matthew	8000	96000	80
Fripp	Adam	8200	98400	80
Kaufling	Payam	7900	94800	80



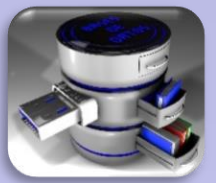
Sentència UPDATE

És possible actualitzar columnes en la clàusula SET d'una sentència UPDATE mitjançant l'escriptura de subconsultas.

LAST_NAME	FIRST_NAME	SALARY	ANSAL
Russell	John	14000	168000
Partners	Karen	13500	162000
Errazuriz	Alberto	12000	144000

```
UPDATE dept80
SET salary= (SELECT ROUND(AVG(salary))
             FROM emp_copy
             WHERE department_id=80),
    ansal = (SELECT ROUND(AVG(salary))*12
             FROM emp_copy
             WHERE department_id=80);
```

LAST_NAME	FIRST_NAME	SALARY	ANSAL
Russell	John	8956	107472
Partners	Karen	8956	107472
Errazuriz	Alberto	8956	107472



Sentència UPDATE

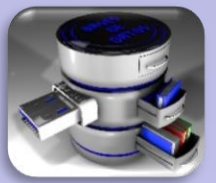
També poden utilitzar-se subconsultes en les clàusula WHERE

LAST_NAME	FIRST_NAME	SALARY	ANSAL	DEPARTMENT_ID
Olson	TJ	2100	25200	80
Philtanker	Hazel	2200	26400	80
Markle	Steven	2200	26400	80
Gee	Ki	2400	28800	80

```
UPDATE dept80
SET salary= (SELECT ROUND(AVG(salary))
             FROM emp_copy
             WHERE department_id=80),
    ansal = (SELECT ROUND(AVG(salary))*12
             FROM emp_copy
             WHERE department_id=80)
WHERE salary<(SELECT ROUND(AVG(salary))
             FROM emp_copy
             WHERE department_id=80);
```

```
62 filas actualizadas.
```

LAST_NAME	FIRST_NAME	SALARY	ANSAL
Olsen	Christopher	8956	107472
Cambrault	Nanette	8956	107472
Tuvault	Oliver	8956	107472



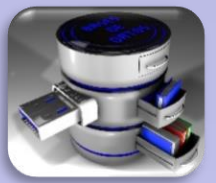
Sentència DELETE

Si volem eliminar alguna fila de dades, utilitzarem la sentència DELETE.

Si s'especifica la clàusula WHERE s'eliminaran totes les files que complisquen la condició.

Si no s'especifica, s'eliminaran totes les files de la taula.

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

Sentència DELETE

```
SELECT cod_employee, last_name, first_name,department_id
FROM emp_copy;
```

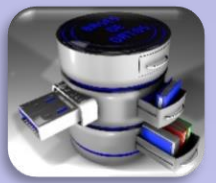
COD_EMPLOYEE	LAST_NAME	FIRST_NAME	DEPARTMENT_ID
100	King	Steven	90
101	Kochhar	Neena	90
102	De Haan	Lex	90
103	Hunold	Alexander	60

```
DELETE FROM emp_copy
WHERE department_id=90;
```

3 filas eliminado

```
SELECT cod_employee, last_name, first_name,department_id
FROM emp_copy
ORDER BY 4 desc;
```

COD_EMPLOYEE	LAST_NAME	FIRST_NAME	DEPARTMENT_ID
108	Greenberg	Nancy	100
113	Popp	Luis	100
145	Russell	John	80



Sentència DELETE

```
DELETE FROM emp_copy;
```

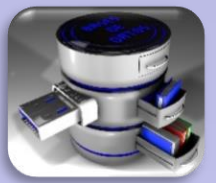
109 filas eliminado

```
SELECT *
FROM emp_copy;
```

COD_EMP...	FIRST_NA...	LAST_NAME

```
DELETE FROM emp_copy
WHERE department_id = (SELECT department_id
                       FROM emp_copy
                       WHERE last_name LIKE 'Lee' AND first_name LIKE 'David');
```

34 filas eliminado



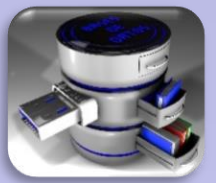
Sentència TRUNCATE

Una forma més eficaç de buidar una taula és amb la sentència TRUNCATE. Pots utilitzar la sentència TRUNCATE per a eliminar fàcilment totes les files d'una taula. L'eliminació de files amb les sentencia TRUNCATE és més ràpida que l'eliminació amb la sentència DELETE perquè TRUNCATE és una sentència de llenguatge de definició de dades (DDL) i no genera cap informació que puga utilitzar-se després en rollback.

```
TRUNCATE TABLE table_name;
```

```
TRUNCATE TABLE emp_copy;
```

```
Table EMP_COPY truncado.
```

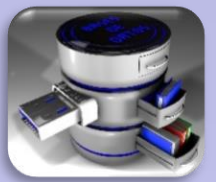



Control de transaccions de bases de dades

El control de transaccions de bases de dades es realitza mitjançant COMMIT, ROLLBACK i SAVEPOINT.

El servidor d'Oracle garanteix la consistència de les dades basada en **transaccions**. Les transaccions proporcionen més flexibilitat i control en canviar les dades i garanteixen la consistència de les dades en cas de fallada de procés d'usuari o del sistema.

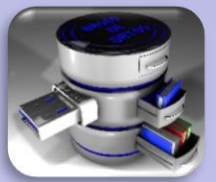
Les transaccions consten de sentències DML que suposen un canvi en les dades. Per exemple, una transferència de fons entre dos comptes ha d'incloure un dèbit en un compte i un crèdit en una altra per la mateixa quantitat. Totes dues accions han de ser correctes o incorrectes per igual; el crèdit no s'ha de confirmar sense el dèbit



Control de transaccions de bases de dades

Una transacció de bases de dades consisteix en una de les següents opcions:

- ✓ Diverses sentències DML (INSERT, UPDATE, DELETE) que constitueixen un canvi consistent de les dades.
- ✓ Una sentència DDL (CREATE TABLE, ALTER TABLE, DROP TABLE)
- ✓ Una sentència DCL



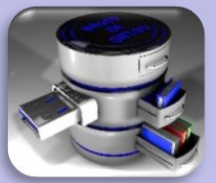
Control de transaccions de bases de dades

Inici i fi d'una transacció

Comença quan s'execute la primera sentència SQL – DML (INSERT, UPDATE, DELETE)

Acaba amb un dels següents esdeveniments:

- S'emet una sentència COMMIT o ROLLBACK
- S'executa una sentència DDL o DCL
- L'usuari ix de l'aplicació
- El sistema falla



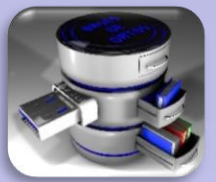
Control de transaccions de bases de dades

Avantatges de COMMIT i ROLLBAKC

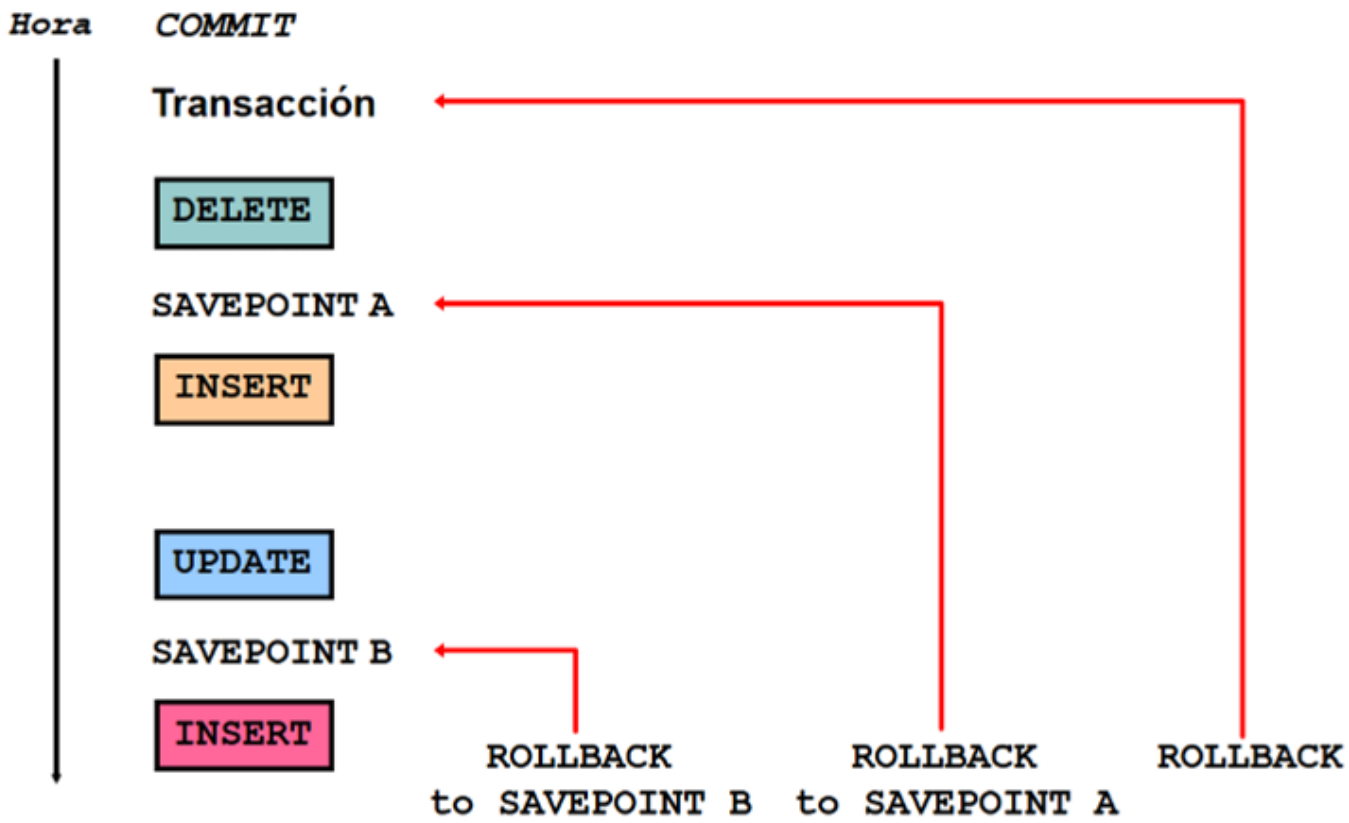
COMIMIT s'utilitza per a confirmar els canvis d'una transacció. ROLLBACK s'utilitza per a desfer la transacció.

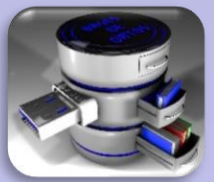
Amb les sentències COMMIT i ROLLBACK es pot:

- Garantir la consistència
- Visualitzar una presentació preliminar dels canvis de les dades abans de fer-los permanents



Control de transaccions de bases de dades





Control de transaccions de bases de dades

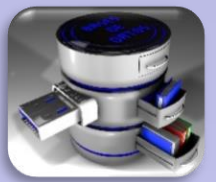
Pot controlar la lògica de les transaccions mitjançant les sentències COMMIT, SAVEPOINT i ROLLBACK.

COMMIT: Finalitza la transacció actual convertint tots els canvis de dades pendents en permanents.

SAVEPOINT: marca un punt d'enregistrament de la transacció actual.

ROLLBACK: finalitza la transacció actual descartant tots els canvis de dades pendents.

ROLLBACK TO SAVEPOINT: realitza un rollback de la transacció actual en el punt d'enregistrament especificat, descartant d'aquesta manera la possibilitat de fer canvis i punts d'enregistrament creats després del punt d'enregistrament en el qual està realitzant el rollback. Si s'omet la clàusula TO SAVEPOINT, la sentència ROLLBACK realitza un rollback de tota la transacció, no hi ha manera de mostrar el que ha creat.

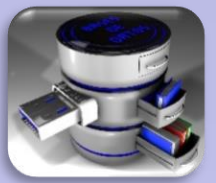


Control de transaccions de bases de dades

Pot crear un marcador en la transacció actual mitjançant la sentència SAVEPOINT, que divideix la transacció en seccions més xicotetes. Pot descartar els canvis pendents fins a aquest marcador amb la sentència ROLLBACK TO SAVEPOINT.

Si crea un segon punt d'enregistrament amb el mateix nom que un punt d'enregistrament anterior, aquest se suprimeix.

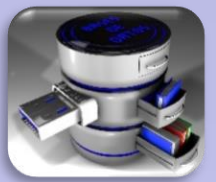
```
UPDATE...
SAVEPOINT update_done;
SAVEPOINT update_done succeeded.
INSERT...
ROLLBACK TO update_done;
ROLLBACK TO succeeded.
```

Control de transaccions de bases de dades

L'estat de les dades abans d'emetre les sentències COMMIT o ROLLBACK es descriu a continuació:

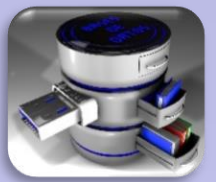
- ✓ Es pot recuperar l'estat anterior de les dades.
- ✓ L'usuari actual pot revisar els resultats de les operacions de manipulació de dades mitjançant la consulta de les taules.
- ✓ Els altres usuaris no poden veure els resultats de les operacions de manipulació realitzades per l'usuari actual. El servidor d'Oracle estableix la consistència de lectura per a garantir que tots els usuaris veuen les dades tal com estaven en el moment de l'última confirmació.
- ✓ Les files afectades estan bloquejades; altres usuaris no poden canviar les dades de les files afectades.



Control de transaccions de bases de dades

L'estat de les dades després de la sentència COMMIT és el següent:

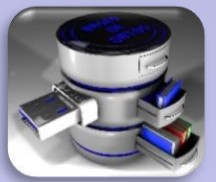
- ✓ Els canvis de dades s'escriuen en la base de dades.
- ✓ L'estat anterior de les dades ja no està disponible amb les consultes SQL normals.
- ✓ Tots els usuaris poden veure els resultats de la transacció.
- ✓ Els bloquejos de les files afectades s'alliberen i les files queden ara disponibles perquè altres usuaris facen nous canvis en les dades.
- ✓ S'esborren tots els punts d'enregistrament.



Control de transaccions de bases de dades

Per a descartar tots els canvis pendents, utilitzem la sentència ROLLBACK obtenint els següents resultats:

- ✓ Es desfan els canvis de dades.
- ✓ Es restaura l'estat anterior de les dades.
- ✓ S'alliberen els bloquejos de les files afectades.



Control de transaccions de bases de dades

En intentar eliminar un registre de la taula TEST, pot esborrar accidentalment la taula. Pot corregir l'error, tornar a emetre la sentència correcta i fer permanents els canvis de dades

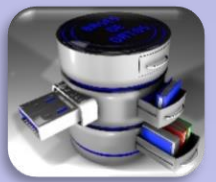
```
DELETE FROM test;
25,000 rows deleted.

ROLLBACK ;
Rollback complete.

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

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

COMMIT;
Commit complete.
```

Activitat

Utilitzant les taules curse i persona que apareixen en les diapositives del tema 11, afig almenys dos cursos i tres persones, de tal manera que una persona estiga matriculada en un curs i les altres dues persones estiguen matriculades en l'altre curs.

Recorda que hauràs de tindre en compte les restriccions de VNN i UNI.

Utilitza aquestes taules per a modificar les dades inserides i fins i tot elimina alguna de les files.