Manipulació de dades

NF2 - Llenguatge SQL: DDL

UF2 – Llenguatge SQL: DML i DDL

Laura Villalba Guerrero

Que veurem?

1. Introducció
2. Inserció de dades
    I. Omplir registres a partir de files d'una consulta
3. Actualització de registres
4. Esborrat de registres
5. Transaccions
    i. Commit
    ii. Commit explícit
    iii. Rollback
    iv. Savepoint
6. Resum

1. INTRODUCCIÓ

- Una de les part fonamentals de SQL és la manipulació de dades (DML).
- El DML ho formen les instruccions capaços de modificar les dades d'una base de dades.
- Al conjunt d'instruccions DML que s'executen de manera consecutiva se'ls anomena **transaccions.**
- Aquestes transaccions es poden acceptar o anul·lar, ja que la instrucció no s'executa fins no s'accepta amb la comanda **COMMIT.**

2. INSERCCIÓ DE DADES

- Afegir dades a una taula es realitza amb la comanda **INSERT**.
- **Sintaxis:**

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

- La taula representa la taula a la qual volem afegir el registre i els valors que segueixen a VALUES són els valors que donem als diferents camps del registre.
- Si no s'especifica la llista de camps, la llista de valors ha de seguir l'ordre de les columnes segons van ser creats (és l'ordre de columnes segons les retorna el comando DESCRIPTION).
- Els camps no emplenats explícitament amb l'ordre INSERT, s'emplenen amb el seu valor per defecte (DEFAULT) o bé amb NULL si no es va indicar valor algun.
- Si algun camp té restricció de tipus NOT NULL, ocorrerà un error si no emplenem el camp amb algun valor.

2. INSERCCIÓ DE DADES

- **Exemple**, suposem que tenim un taula clients amb els camps: dni, nom, cognom1, cognom2, localitat i adreça; suposem que aquest és l'ordre de creació dels camps d'aquesta taula i que la localitat té com a valor per defecte *Palència* i l'adreça no té valor per defecte. En aquest cas les següents instruccions són equivalents:

```
INSERT INTO clients VALUES ( '111111111','Pere', 'Vera', 'Crepo', DEFAULT, NULL);
ó
INSERT INTO clients (dni, nom, cognom1, cognom2) VALUES  '111111111','Pere', 'Vera', 'Crepo');
```

  - Són equivalents perquè a la segona instrucció els camps no indicats s'omplen amb el seu valor per defecte.

Omplir registres a partir de files d'una consulta

- Hi ha un tipus de consulta, anomenada d'**addició de dades**, que permet emplenar dades d'una taula copiant el resultat d'una consulta.
- Aquest farciment es basa en una consulta **SELECT** que posseirà les dades a afegir.

2. INSERCCIÓ DE DADES

Omplir registres a partir de files d'una consulta (II)

- **Sintaxis**

```
INSERT INTO nom_taula ( camp1, camp2, ...., campN)
SELECT CampCompatibleCamp1, CampCompatibleCamp2, ..., CampCompatibleCampN
FROM taula (s)
[... Altres clàusules del SELECT...]
```

- **Exemple**

```
INSERT INTO clients2015 (dni, nom, localitat, adreça)
   SELECT dni, nom, localitat, adreça
    FROM clients
    WHERE problemes=0;
```

3. ACTUALITZACIÓ DE REGISTRES

- La modificació de dades dels registres s'implementa amb la instrucció **UPDATE.**
- **Sintaxis**

```
UPDATE nom_taula
SET Columna1=valor1 [ , columna2=valor2 ... ]
 [WHERE  condicio]
```

- Es modifiquen les columnes implicades en l'apartat **SET** amb els valors indicats.
- La clàusula WHERE permet especificar quin registres hi seran modificats.
- **Exemples:**

```
UPDATE clients SET provincia='Ourense'
WHERE provincia='Ourense';

UPDATE productes SET preu=preu 1*16;

UPDATE partits SET data=NEXT_DAY (SYSDATE, 'Dimarts')
WHERE data=SYSDATE;

UPDATE empleats SET lloc_treball=(SELECT lloc_treball
                                  FROM empleats
                                  WHERE id_empleat=12)
 WHERE seccio=23;
```

4. ESBORRAT DE DADES

- L'esborrat de dades dels registres s'implementa amb la instrucció **DELETE.**
- **Sintaxis**

```
DELETE [FROM] nom_taula
 [WHERE  condicio]
```

- Elimina els registres de la taula que compleix la condició indicada.
- **Exemples:**

```
DELETE FROM clients SET
WHERE provincia='Ourense';

DELETE FROM empleats
WHERE id_empleat IN (SELECT id_empleat FROM errors_greus);
```

- L'esborrat de dades no pot provocar errades de integritat i que la opció d'integritat.

5. TRANSACCIONS

- Una transacció està formada per una sèrie d'instruccions DML.
- Una transacció comença amb la primera instrucció DML que s'executa i finalitza amb alguna d'aquestes circumstàncies:
  - Una operació **COMMIT** o **ROLLBACK**
  - Una instrucció DDL (com ALTER TABLE per exemple)
  - Una instrucció DCL (com GRANT)
  - L'usuari abandona la sessió
  - Caiguda del sistema
- Hi ha que tenir en compte que qualsevol instrucció DDL o DCL dóna lloc a un COMMIT implícit, és a dir totes les instruccions DML executades fins a aquest instant passen a ser definitives.

5. TRANSACCIONS (II)

COMMIT

- La instrucció COMMIT fa que els canvis realitzats per la transacció siguin definitius, irrevocables.
- Només s'ha d'utilitzar si estem d'acord amb els canvis, convé assegurar-se molt abans de realitzar el COMMIT ja que les instruccions executades poden afectar a milers de registres.
- A més el tancament correcte de la sessió dóna lloc a un COMMIT, encara que sempre convé executar explícitament aquesta instrucció a fi d'assegurar-nos del que fem.
- iSQL*Plus permet validar de manera automàtica les transaccions sense tenir que indicar-ho de forma explícita.
- Es fa servir la comanda **AUTOCOMMIT**. El valor d'aquest paràmetre es pot mostrar amb l'ordre **SHOW AUTOCOMMIT**;

5. TRANSACCIONS (III)

COMMIT IMPLÍCIT

- Les següents ordres SQL executen un COMMIT sense necessitat d'indicar-ho:

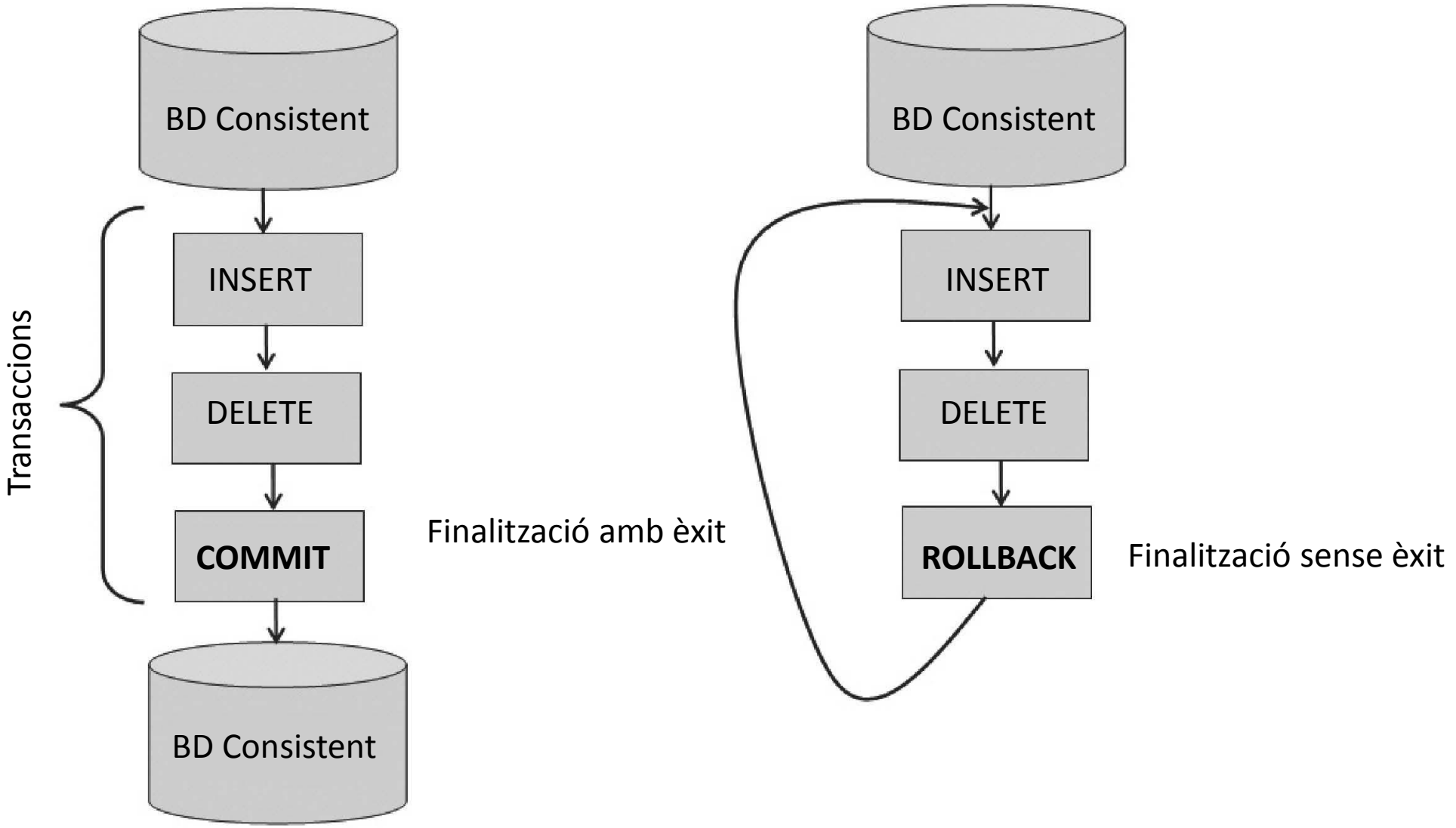
QUIT	DISCONNECT	CREATE VIEW	ALTER
EXIT	CREATE TABLE	DROP VIEW	REVOQUE
CONNECT	DROP TABLE	GRANT	AUDIT
			NOAUDIT

6. TRANSACCIONS (IV)

ROLLBACK

- Aquesta instrucció avorta la transacció, retorna a la situació de les taules de la base de dades des de l'últim COMMIT.
- Anul·la definitivament els canvis, per la qual cosa convé també assegurar-se d'aquesta operació.
- Un abandó de sessió incorrecte o un problema de comunicació o de caiguda del sistema donen lloc a un ROLLBACK implícit.

6. TRANSACCIONS (V)



5. TRANSACCIONS (VI)

SAVEPOINT

- Aquesta instrucció permet establir un punt de ruptura.
- El problema de la combinació ROLLBACK/COMMIT és que un COMMIT accepta tot i un ROLLBACK anul·la tot.
- SAVEPOINT permet assenyalar un punt intermedi entre l'inici de la transacció i la situació actual.
- **Sintaxis:**

```
….Instruccions DML …..
SAVEPOINT nom
…..Instruccions DML……
```

- Per retornar a un punt de ruptura concret es fa servir ROLLBACK TO SAVEPOINT seguit del nom donat el punt de ruptura.

6. RESUM

INSERT	Inserció d'una fila	**INSERT INTO** NomTaula [(columna [, columna] ...)] **VALUES** (valor [, valor] ...);
	Inserció multifila	**INSERT INTO** NomTaula1 [(columna [, columna] ...)] **SELECT** {columna [, columna] ... | *} **FROM** NomreTaula2 [CLÀUSULES DE SELECT];
UPDATE	Modificació files	**UPDATE** <NomTaula> **SET columna1 = valor1, ..., columnan = valorn** **WHERE** condició;
	Modificació de files amb SELECT	**UPDATE** <NomTaula> **SET** columna1 = valor1, columna2 = valor2, ... **WHERE** columna3=(**SELECT** ....); **UPDATE** <NomTaula> **SET** (columna1, columna2, ...)=(**SELECT** col1, col2, ...) **WHERE** condició; **UPDATE** <NomTaula> **SET** columna1 = (**SELECT** col1 ...), columna2 = (**SELECT** col2 ...) **WHERE** condició;

6. RESUM (II)

DELETE	Esborrar de files	**DELETE [FROM]** NomTaula **WHERE** condició;
TRANSACCIONS	Validar els canvis	**COMMIT;**
	Avortar transaccions	**ROLLBACK;**

Preguntes!!!!!