# Mòdul 0372 – Gestió de bases de dades

# Índex

**Cicle formatiu:** Administració de Sistemes Informàtics en Xarxa (ASIX)

**Durada:** 198 h (132 h centre + 66 h empresa)

**SGBD de referència:** MariaDB

# 1. Sistemes d'emmagatzematge de la informació

## 1.1. Fitxers

Abans de les bases de dades modernes, la informació es guardava en **fitxers**. Existeixen diversos tipus:

| Tipus | Descripció |
|---|---|
| **Plans (seqüencials)** | Els registres s'emmagatzemen un darrere l'altre. L'accés és lent perquè cal llegir des del principi. |
| **Indexats** | Disposen d'un índex que permet localitzar registres sense recórrer tot el fitxer. |
| **Accés directe** | Es calcula la posició física del registre mitjançant una funció hash. L'accés és molt ràpid. |

**Problemes dels sistemes de fitxers tradicionals:**

- **Redundància:** la mateixa informació es repeteix en múltiples fitxers.
- **Inconsistència:** si es modifica un fitxer i no els altres, les dades no coincideixen.
- **Dificultat d'accés:** cal programar cada consulta específicament.
- **Aïllament de dades:** els formats heterogenis dificulten compartir informació.
- **Problemes de seguretat i concurrència:** difícil controlar qui accedeix i quan.

## 1.2. Bases de dades: conceptes, usos i tipus

Una **base de dades (BD)** és una col·lecció de dades relacionades entre si, organitzades de manera que es puguin emmagatzemar, recuperar i modificar de forma eficient.

**Tipus segons el model de dades**

| Model | Descripció | Exemples |
|---|---|---|
| **Jeràrquic** | Les dades s'organitzen en forma d'arbre. Cada registre fill té un únic pare. | IMS d'IBM |
| **En xarxa** | Extensió del jeràrquic; un fill pot tenir múltiples pares. | IDMS |
| **Relacional** | Les dades s'organitzen en taules (relacions). És el model més estès. | MariaDB, PostgreSQL, Oracle |
| **Orientat a objectes** | Les dades es representen com a objectes, amb atributs i mètodes. | db4o |
| **NoSQL** | Dissenys no relacionals: documents, clau-valor, grafs, columnar. | MongoDB, Redis, Cassandra |

**Tipus segons la ubicació de la informació**

- **Centralitzades:** totes les dades resideixen en un únic servidor.
- **Distribuïdes:** les dades es reparteixen entre múltiples nodes interconnectats.
- **En núvol:** allotjades en infraestructures de cloud computing (Amazon RDS, Google Cloud SQL…).

## 1.3. Sistemes gestors de bases de dades (SGBD)

Un **SGBD** (en anglès *DBMS*, *Database Management System*) és el programari que permet crear, mantenir i controlar l'accés a una base de dades.

**Funcions principals**

- **Definició de dades:** permet definir l'estructura (esquema) de la BD mitjançant el DDL.
- **Manipulació de dades:** permet inserir, consultar, modificar i eliminar dades (DML).
- **Control d'accés:** gestiona els permisos d'usuaris i rols.
- **Integritat:** garanteix que les dades compleixin les regles definides.
- **Concurrència:** permet que múltiples usuaris accedeixin simultàniament sense conflictes.
- **Recuperació:** restaura la BD a un estat consistent en cas d'error.

**Components d'un SGBD**

- **Motor de la base de dades:** nucli que processa les consultes.
- **Diccionari de dades (catàleg):** emmagatzema metadades sobre l'estructura de la BD.
- **Intèrpret del llenguatge de consulta:** processa sentències SQL.
- **Gestor de transaccions:** garanteix les propietats: atomicitat, consistència, aïllament i durabilitat.
- **Gestor d'emmagatzematge:** controla l'accés als fitxers físics.
- **Gestor de concurrència:** coordina l'accés simultani.
- **Gestor de recuperació:** gestiona còpies de seguretat i restauració.

**Classificació dels SGBD**

- **Relacionals (RDBMS):** MariaDB, MySQL, PostgreSQL, Oracle, SQL Server.
- **NoSQL:** MongoDB (documents), Redis (clau-valor), Neo4j (grafs), Cassandra (columnar).
- **NewSQL:** CockroachDB, TiDB (relacionals amb escalabilitat horitzontal).
- **Propietaris vs. lliures:** Oracle i SQL Server (propietaris); MariaDB, PostgreSQL (lliures).

# 2. Disseny lògic de bases de dades

## 2.1. El model de dades

El **model de dades** és un conjunt de conceptes i notacions per descriure les dades d'un sistema d'informació. El disseny passa per tres nivells:

1. **Nivell conceptual:** model entitat-relació (E/R), independent del SGBD.
2. **Nivell lògic:** model relacional, adaptat al tipus de SGBD.
3. **Nivell físic:** implementació concreta sobre un SGBD determinat.

## 2.2. El diagrama Entitat-Relació (E/R)

El **diagrama E/R** és una representació gràfica del model conceptual que descriu les entitats del sistema i les relacions entre elles.

### Elements principals

| Element | Símbol | Descripció |
|---|---|---|
| **Entitat** | Rectangle | Objecte del món real sobre el qual volem guardar informació. Ex: `CLIENT`, `PRODUCTE`. |
| **Atribut** | Ellipse | Propietat o característica d'una entitat. Ex: `nom`, `preu`. |
| **Atribut clau** | Ellipse subratllad | Atribut que identifica unívocament cada instància. Ex: `DNI`, `codi_producte`. |
| **Relació** | Rombe | Associació entre dues o més entitats. Ex: `COMPRA`. |
| **Atribut multivalu** | Ellipse doble | Pot tenir múltiples valors. Ex: `telèfon`. |
| **Atribut derivat** | Ellipse discontínu | Es calcula a partir d'altres atributs. Ex: `edat` (de `data_naixement`). |

### Cardinalitat

Indica el nombre d'instàncies d'una entitat que es poden associar a instàncies d'una altra:

- **1:1** (un a un): un empleat té un únic despatx.
- **1:N** (un a molts): un client pot fer moltes comandes.
- **N:M** (molts a molts): un alumne pot matricular-se de moltes assignatures, i una assignatura pot tenir molts alumnes.

### Participació (modalitat)

- **Total (obligatòria):** tota instància de l'entitat participa en la relació (doble línia).
- **Parcial (opcional):** no totes les instàncies han de participar (línia simple).

### Entitats dèbils

Una **entitat dèbil** no té prou atributs per identificar-se per si sola; depèn d'una entitat forta.
Ex: `LÍNIA_COMANDA` depèn de `COMANDA`.
Es representa amb rectangle doble; la relació identificadora, amb rombe doble.

## 2.3. El model E/R ampliat

El model E/R ampliat afegeix:

- **Generalització/Especialització (herència):** una entitat superclasse es descompon en subclasses que hereten els seus atributs. Ex: `PERSONA` → `EMPLEAT`, `CLIENT`.
  - **Disjunta:** una instància pertany a una única subclasse.
  - **Encavalcada:** una instància pot pertànyer a múltiples subclasses.
- **Agregació:** una relació es tracta com si fos una entitat per poder relacionar-la amb altres.

## 2.4. El model relacional

### Terminologia

| Terme relacional | Equivalent informal |
|---|---|
| Relació | Taula |
| Tupla | Fila / registre |
| Atribut | Columna / camp |
| Domini | Conjunt de valors possibles d'un atribut |
| Grau | Nombre de columnes d'una relació |
| Cardinalitat | Nombre de files d'una relació |

### Característiques d'una relació

- No hi ha tuples duplicades.
- L'ordre de les tuples és irrellevant.
- L'ordre dels atributs és irrellevant.
- Els valors dels atributs són atòmics (primera forma normal).

### Claus

- **Superclau:** qualsevol conjunt d'atributs que identifica unívocament una tupla.
- **Clau candidata:** superclau mínima (no es pot eliminar cap atribut sense perdre la unicitat).
- **Clau primària (PK):** clau candidata escollida per identificar les tuples. No pot ser `NULL`.
- **Clau forana (FK):** atribut(s) d'una relació que referencien la PK d'una altra relació.

### Regles d'integritat

- **Integritat d'entitat:** cap atribut de la PK pot tenir valor `NULL`.
- **Integritat referencial:** si una FK té valor, ha d'existir una tupla corresponent a la taula referenciada.

## 2.5. Pas del diagrama E/R al model relacional

| Cas E/R | Transformació |
|---|---|
| Entitat forta | Una taula amb tots els seus atributs; la clau E/R és la PK. |
| Entitat dèbil | Una taula que inclou la PK de l'entitat forta com a part de la seva PK (clau composta). |
| Relació 1:1 | S'incorpora la PK d'una entitat com a FK a l'altra (es prefereix el costat de participació total). |
| Relació 1:N | La PK del costat “1” s'afegeix com a FK al costat “N”. |
| Relació N:M | Es crea una nova taula de relació amb les PK de les dues entitats com a PK composta. |
| Atribut multivaluat | Es crea una nova taula amb la PK de l'entitat i el valor de l'atribut. |
| Especialitza | Opció 1: una taula per a la superclasse i una per a cada subclasse (FK a superclasse). Opció 2: una taula per a cada subclasse amb tots els atributs. Opció 3: una sola taula amb un atribut discriminador. |

## 2.6. Normalització

La **normalització** és el procés de descomposició de les relacions per eliminar redundàncies i anomalies d'inserció, modificació i eliminació.

### Dependències funcionals

- **Dependència funcional (DF):** $A \rightarrow B$ significa que cada valor d'$A$ determina un únic valor de $B$.
- **DF parcial:** $B$ depèn d'una part d'una clau composta.
- **DF transitiva:** $A \rightarrow B$ i $B \rightarrow C$, per tant $A \rightarrow C$ (on $B$ no és clau).

### Formes normals

#### 1FN (Primera Forma Normal)

- Tots els atributs han de ser atòmics (valors indivisibles).
- No hi ha grups repetits.

#### 2FN (Segona Forma Normal)

- Compleix la 1FN.
- Tots els atributs no clau depenen **totalment** de la PK (no hi ha dependències parcials).

- S'aplica quan la PK és composta.

```
Exemple:
Taula incorrecta: COMANDA_PRODUCTE(cod_comanda, cod_producte,
↪  quantitat, nom_producte)
  nom_producte depèn només de cod_producte (DF parcial)

Solució:
  COMANDA_PRODUCTE(cod_comanda, cod_producte, quantitat)
  PRODUCTE(cod_producte, nom_producte)
```

**3FN (Tercera Forma Normal)**

- Compleix la 2FN.
- No hi ha **dependències transitives**: cap atribut no clau depèn d'un altre atribut no clau.

```
Exemple:
Taula incorrecta: EMPLEAT(cod_empleat, nom, cod_departament,
↪  nom_departament)
  nom_departament depèn de cod_departament (DF transitiva)

Solució:
  EMPLEAT(cod_empleat, nom, cod_departament)
  DEPARTAMENT(cod_departament, nom_departament)
```

**FNBC (Forma Normal de Boyce-Codd)**

- Versió reforçada de la 3FN: per a tota DF $A \rightarrow B$, A ha de ser superclau.

# 3. Disseny físic de bases de dades

## 3.1. Eines gràfiques

MariaDB pot gestionar-se amb eines gràfiques com:

- **phpMyAdmin:** eina web molt estesa, adequada per a gestió bàsica.
- **DBeaver:** client multiplataforma, molt complet.
- **MySQL Workbench:** compatible amb MariaDB, permet disseny E/R visual.

## 3.2. El llenguatge de definició de dades (DDL)

El **DDL** (*Data Definition Language*) permet crear i modificar l'estructura de la BD.

### Crear i seleccionar una base de dades

```
-- Crear una base de dades
CREATE DATABASE institut
    CHARACTER SET utf8mb4
    COLLATE utf8mb4_unicode_ci;

-- Seleccionar la base de dades activa
USE institut;

-- Eliminar una base de dades
DROP DATABASE IF EXISTS institut;
```

### Crear taules

```
CREATE TABLE alumne (
    id_alumne    INT             NOT NULL AUTO_INCREMENT,
    nom          VARCHAR(50)     NOT NULL,
    cognoms      VARCHAR(100)    NOT NULL,
    data_naix    DATE,
    email        VARCHAR(100)    UNIQUE,
    CONSTRAINT pk_alumne PRIMARY KEY (id_alumne)
);
```

### Modificar taules

```
-- Afegir una columna
ALTER TABLE alumne ADD COLUMN telefon VARCHAR(15);

-- Modificar el tipus d'una columna
ALTER TABLE alumne MODIFY COLUMN telefon VARCHAR(20);

-- Eliminar una columna
ALTER TABLE alumne DROP COLUMN telefon;

-- Renomenar una taula
ALTER TABLE alumne RENAME TO estudiant;
```

### Eliminar taules

```
DROP TABLE IF EXISTS alumne;

-- Buidar una taula (elimina totes les files, però manté l'estructura)
TRUNCATE TABLE alumne;
```

## 3.3. Tipus de dades a MariaDB

### Numèrics

| Tipus | Bytes | Rang (amb signe) |
|---|---|---|
| `TINYINT` | 1 | -128 a 127 |
| `SMALLINT` | 2 | -32.768 a 32.767 |
| `MEDIUMINT` | 3 | -8.388.608 a 8.388.607 |
| `INT` / `INTEGER` | 4 | -2.147.483.648 a 2.147.483.647 |
| `BIGINT` | 8 | $\pm 9{,}2 \times 10^{18}$ |
| `DECIMAL(p,s)` | Variable | Precisió exacta; útil per a diners |
| `FLOAT` | 4 | Punt flotant, precisió simple |
| `DOUBLE` | 8 | Punt flotant, precisió doble |

### Cadenes de text

| Tipus | Descripció |
|---|---|
| `CHAR(n)` | Longitud fixa de n caràcters (màx. 255). |
| `VARCHAR(n)` | Longitud variable fins a n caràcters (màx. 65.535). |
| `TEXT` | Text llarg (fins a 65.535 bytes). |
| `MEDIUMTEXT` | Fins a 16 MB. |
| `LONGTEXT` | Fins a 4 GB. |

### Data i hora

| Tipus | Format | Descripció |
|---|---|---|
| `DATE` | `AAAA-MM-DD` | Només data. |
| `TIME` | `HH:MM:SS` | Només hora. |
| `DATETIME` | `AAAA-MM-DD HH:MM:SS` | Data i hora. No ajusta zona horària. |
| `TIMESTAMP` | `AAAA-MM-DD HH:MM:SS` | Com DATETIME, però s'ajusta a la zona horària del servidor. |
| `YEAR` | `AAAA` | Any de 4 dígits. |

### Altres

| Tipus | Descripció |
|---|---|
| `BOOLEAN` / `TINYINT(1)` | 0 (fals) o 1 (veritat). |
| `ENUM('v1','v2',...)` | Un valor d'una llista predefinida. |
| `SET('v1','v2',...)` | Zero o més valors d'una llista predefinida. |
| `BLOB` | Dades binàries (imatges, fitxers). |
| `JSON` | Dades en format JSON (MariaDB 10.2+). |

## 3.4. Restriccions (constraints)

Les restriccions garanteixen la integritat de les dades.

```
CREATE TABLE matricula (
    id_matricula    INT             NOT NULL AUTO_INCREMENT,
    id_alumne       INT             NOT NULL,
    id_modul        INT             NOT NULL,
    curs            YEAR            NOT NULL DEFAULT (YEAR(CURDATE())),
    nota_final      DECIMAL(4,2)    CHECK (nota_final BETWEEN 0 AND
 ↪  10),

    CONSTRAINT pk_matricula     PRIMARY KEY (id_matricula),
    CONSTRAINT uq_mat_alu_mod   UNIQUE (id_alumne, id_modul, curs),
    CONSTRAINT fk_mat_alumne    FOREIGN KEY (id_alumne)
                                REFERENCES alumne(id_alumne)
                                ON DELETE RESTRICT
                                ON UPDATE CASCADE,
    CONSTRAINT fk_mat_modul     FOREIGN KEY (id_modul)
                                REFERENCES modul(id_modul)
                                ON DELETE RESTRICT
                                ON UPDATE CASCADE
);
```

### Polítiques d'integritat referencial

| Acció | Descripció |
|---|---|
| `RESTRICT` | No permet eliminar/modificar la fila pare si té files filles. |
| `CASCADE` | Propaga l'eliminació/modificació a les files filles. |
| `SET NULL` | Posa `NULL` a la FK de les files filles. |
| `NO ACTION` | Similar a `RESTRICT` (comportament per defecte). |
| `SET DEFAULT` | Posa el valor per defecte a la FK (poc suportat). |

# 4. Realització de consultes

## 4.1. La sentència SELECT

```
SELECT [DISTINCT] llista_columnes | *
FROM   taula [AS àlies]
[JOIN  ...]
[WHERE condició]
[GROUP BY columnes]
[HAVING condició_grups]
[ORDER BY columnes [ASC|DESC]]
[LIMIT n [OFFSET m]];
```

## 4.2. Consultes simples

```
-- Totes les columnes
SELECT * FROM alumne;

-- Columnes específiques amb àlies
SELECT nom, cognoms AS 'Cognoms complets' FROM alumne;

-- Eliminar duplicats
SELECT DISTINCT curs FROM matricula;

-- Ordenació
SELECT nom, cognoms FROM alumne ORDER BY cognoms ASC, nom ASC;

-- Limitar resultats
SELECT * FROM alumne ORDER BY cognoms LIMIT 10 OFFSET 20;
```

#### Condicions (WHERE)

```
-- Comparació
SELECT * FROM alumne WHERE id_alumne = 5;

-- Rang
SELECT * FROM matricula WHERE nota_final BETWEEN 5 AND 7;

-- Llista de valors
SELECT * FROM modul WHERE curs IN (1, 2);

-- Patrons de text
SELECT * FROM alumne WHERE cognoms LIKE 'Garcia%';
SELECT * FROM alumne WHERE email LIKE '%@gmail.com';

-- Valors nuls
SELECT * FROM alumne WHERE email IS NULL;
```

```
SELECT * FROM alumne WHERE email IS NOT NULL;

-- Operadors lògics
SELECT * FROM matricula WHERE nota_final >= 5 AND curs = 2024;
SELECT * FROM alumne WHERE nom = 'Pere' OR nom = 'Joan';
SELECT * FROM alumne WHERE NOT (nom = 'Pere');
```

## 4.3. Consultes de resum i agrupament

### Funcions d'agregació

| Funció | Descripció |
|---|---|
| `COUNT(*)` | Compta el nombre de files. |
| `COUNT(col)` | Compta els valors no nuls d'una columna. |
| `SUM(col)` | Suma els valors. |
| `AVG(col)` | Calcula la mitjana. |
| `MAX(col)` | Valor màxim. |
| `MIN(col)` | Valor mínim. |

```
-- Nombre total d'alumnes
SELECT COUNT(*) AS total_alumnes FROM alumne;

-- Nota mitjana per mòdul
SELECT id_modul, AVG(nota_final) AS nota_mitjana
FROM   matricula
WHERE  nota_final IS NOT NULL
GROUP  BY id_modul;

-- Nombre d'alumnes aprovats per mòdul (nota >= 5)
SELECT id_modul, COUNT(*) AS aprovats
FROM   matricula
WHERE  nota_final >= 5
GROUP  BY id_modul
HAVING COUNT(*) > 10
ORDER  BY aprovats DESC;
```

> **Nota:** `WHERE` filtra **files individuals** abans de l'agrupament; `HAVING` filtra **grups** després de l'agrupament.

## 4.4. Composicions (JOIN)

### INNER JOIN

Retorna les files que tenen correspondència en les dues taules.

```
SELECT a.nom, a.cognoms, m.curs, mo.nom_modul, m.nota_final
FROM   matricula m
       INNER JOIN alumne a  ON m.id_alumne = a.id_alumne
       INNER JOIN modul  mo ON m.id_modul  = mo.id_modul
WHERE  m.curs = 2024
ORDER  BY a.cognoms;
```

### LEFT JOIN

Retorna **totes** les files de la taula esquerra, i les coincidents de la dreta (o `NULL` si no n'hi ha).

```
-- Alumnes amb o sense matrícules
SELECT a.nom, a.cognoms, COUNT(m.id_matricula) AS num_matricules
FROM   alumne a
       LEFT JOIN matricula m ON a.id_alumne = m.id_alumne
GROUP  BY a.id_alumne, a.nom, a.cognoms;
```

### RIGHT JOIN

Igual que el LEFT JOIN però amb la taula dreta com a base.

```
SELECT mo.nom_modul, COUNT(m.id_matricula) AS num_alumnes
FROM   matricula m
       RIGHT JOIN modul mo ON m.id_modul = mo.id_modul
GROUP  BY mo.id_modul, mo.nom_modul;
```

### Producte cartesià (CROSS JOIN)

Combina cada fila d'una taula amb totes les files de l'altra.

```
SELECT a.nom, mo.nom_modul
FROM   alumne a CROSS JOIN modul mo;
```

**Auto-unió (self-join)**

```
-- Empleats i el nom del seu supervisor
SELECT e.nom AS empleat, s.nom AS supervisor
FROM   empleat e
       LEFT JOIN empleat s ON e.id_supervisor = s.id_empleat;
```

## 4.5. Subconsultes

Una subconsulta és una SELECT dins d'una altra sentència SQL.

```
-- Alumnes amb nota superior a la mitjana del seu mòdul
SELECT a.nom, a.cognoms, m.nota_final
FROM   matricula m
       INNER JOIN alumne a ON m.id_alumne = a.id_alumne
WHERE  m.nota_final > (
           SELECT AVG(nota_final)
           FROM   matricula
           WHERE  id_modul = m.id_modul
       );

-- Alumnes que mai han suspès (amb NOT IN)
SELECT nom, cognoms
FROM   alumne
WHERE  id_alumne NOT IN (
           SELECT DISTINCT id_alumne
           FROM   matricula
           WHERE  nota_final < 5
       );

-- Mòduls amb més alumnes que la mitjana (amb EXISTS)
SELECT mo.nom_modul
FROM   modul mo
WHERE  EXISTS (
           SELECT 1
           FROM   matricula m
           WHERE  m.id_modul = mo.id_modul
           GROUP  BY m.id_modul
           HAVING COUNT(*) > (SELECT AVG(c) FROM (
               SELECT COUNT(*) AS c FROM matricula GROUP BY id_modul
           ) AS sub)
       );
```

### 4.6. Unions de consultes

```
-- UNION: elimina duplicats
SELECT nom FROM alumne
UNION
SELECT nom FROM professor;

-- UNION ALL: manté duplicats
SELECT nom FROM alumne
UNION ALL
SELECT nom FROM professor;
```

## 5. Edició de les dades

### 5.1. INSERT

```
-- Inserir una fila especificant columnes
INSERT INTO alumne (nom, cognoms, data_naix, email)
VALUES ('Maria', 'Puig Soler', '2005-03-15',
 ↪  'maria.puig@exemple.cat');

-- Inserir múltiples files
INSERT INTO modul (nom_modul, hores, curs) VALUES
    ('Gestió de bases de dades', 165, 1),
    ('Administració de sistemes', 198, 1),
    ('Xarxes locals', 165, 1);

-- Inserir a partir d'una consulta
INSERT INTO historial_matricula (id_alumne, id_modul, nota_final,
 ↪  any_hist)
SELECT id_alumne, id_modul, nota_final, YEAR(NOW())
FROM   matricula
WHERE  nota_final IS NOT NULL;
```

### 5.2. UPDATE

```
-- Actualitzar una fila concreta
UPDATE alumne
SET    email = 'maria.puig.nou@exemple.cat'
WHERE  id_alumne = 12;

-- Actualitzar múltiples files
UPDATE matricula
SET    nota_final = nota_final + 0.5
WHERE  id_modul = 3 AND curs = 2024 AND nota_final IS NOT NULL;

-- Actualitzar amb subconsulta
```

```
UPDATE alumne
SET    actiu = FALSE
WHERE  id_alumne NOT IN (
           SELECT DISTINCT id_alumne
           FROM   matricula
           WHERE  curs = YEAR(NOW())
       );
```

**PRECAUCIÓ**

**Sempre** afegiu clàusula `WHERE` per evitar modificar totes les files per error.

## 5.3. DELETE

```
-- Eliminar una fila concreta
DELETE FROM alumne WHERE id_alumne = 12;

-- Eliminar amb condició
DELETE FROM matricula WHERE curs < 2020;

-- Eliminar amb subconsulta
DELETE FROM matricula
WHERE  id_alumne IN (
           SELECT id_alumne FROM alumne WHERE actiu = FALSE
       );
```

## 5.4. Transaccions

Una **transacció** és una seqüència d'operacions que s'executen com una unitat indivisible. Han de complir les propietats:

| Propietat | Descripció |
|---|---|
| **Atomicitat** | Totes les operacions s'executen o cap. |
| **Consistència** | La BD passa d'un estat consistent a un altre. |
| **Aïllament** | Les transaccions concurrents no s'interfereixen. |
| **Durabilitat** | Els canvis confirmats persisteixen malgrat fallades. |

```
-- Inici de la transacció
START TRANSACTION;

-- Operacions
UPDATE compte SET saldo = saldo - 500 WHERE id_compte = 101;
UPDATE compte SET saldo = saldo + 500 WHERE id_compte = 205;

-- Si tot ha anat bé: confirmar
COMMIT;
```

```
-- Si hi ha hagut un error: desfer
ROLLBACK;
```

### Punts de restauració (SAVEPOINT)

```
START TRANSACTION;

INSERT INTO comanda (id_client, data) VALUES (5, NOW());
SAVEPOINT sp_comanda;

INSERT INTO linia_comanda (id_comanda, id_producte, quantitat) VALUES
 ↪  (LAST_INSERT_ID(), 12, 3);

-- Si falla la línia, desfer només fins al savepoint
ROLLBACK TO sp_comanda;

-- Si tot va bé
COMMIT;
```

### Nivells d'aïllament

```
-- Veure el nivell actual
SELECT @@tx_isolation;

-- Canviar el nivell per a la sessió
SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;
```

| Nivell | Lectures brutes | Lectures no repetibles | Lectures fantasma |
|---|---|---|---|
| `READ UNCOMMITTED` | Sí | Sí | Sí |
| `READ COMMITTED` | No | Sí | Sí |
| `REPEATABLE READ` (per defecte) | No | No | Sí |
| `SERIALIZABLE` | No | No | No |

## 5.5. Polítiques de bloqueig

MariaDB implementa **bloqueig a escala de fila** (amb InnoDB) per gestionar l'accés concurrent.

```
-- Bloqueig compartit (lectura): múltiples transaccions poden llegir
 ↪  simultàniament
SELECT * FROM alumne WHERE id_alumne = 5 LOCK IN SHARE MODE;

-- Bloqueig exclusiu (escriptura): cap altra transacció pot llegir ni
 ↪  escriure
```

```
SELECT * FROM alumne WHERE id_alumne = 5 FOR UPDATE;
```

# 6. Construcció de guions

## 6.1. Introducció al llenguatge procedimental de MariaDB

MariaDB permet escriure codi procedimental per automatitzar tasques complexes a través de **procediments emmagatzemats**, **funcions** i **disparadors**.

## 6.2. Tipus de dades, variables i identificadors

```
DELIMITER //

CREATE PROCEDURE exemple_variables()
BEGIN
    -- Declaració de variables locals
    DECLARE v_nom       VARCHAR(50);
    DECLARE v_total     INT DEFAULT 0;
    DECLARE v_nota      DECIMAL(4,2);
    DECLARE v_data      DATE;

    -- Assignació de valors
    SET v_nom = 'Joan';
    SET v_total = v_total + 1;

    -- Assignació amb SELECT...INTO
    SELECT COUNT(*) INTO v_total FROM alumne;

    SELECT CONCAT('Total alumnes: ', v_total) AS resultat;
END //

DELIMITER ;

CALL exemple_variables();
```

## 6.3. Operadors i estructures de control

### Estructura IF

```
DELIMITER //

CREATE PROCEDURE qualificacio(IN p_nota DECIMAL(4,2), OUT p_qualif
 ↪  VARCHAR(20))
BEGIN
    IF p_nota >= 9 THEN
        SET p_qualif = 'Excel·lent';
    ELSEIF p_nota >= 7 THEN
        SET p_qualif = 'Notable';
    ELSEIF p_nota >= 5 THEN
        SET p_qualif = 'Aprovat';
    ELSE
        SET p_qualif = 'Suspès';
    END IF;
END //

DELIMITER ;

CALL qualificacio(7.5, @resultat);
SELECT @resultat;
```

### Estructura CASE

```
DELIMITER //

CREATE PROCEDURE torn_horari(IN p_hora INT, OUT p_torn VARCHAR(10))
BEGIN
    CASE
        WHEN p_hora BETWEEN 8  AND 14 THEN SET p_torn = 'Matí';
        WHEN p_hora BETWEEN 15 AND 21 THEN SET p_torn = 'Tarda';
        ELSE                               SET p_torn = 'Nit';
    END CASE;
END //

DELIMITER ;
```

### Estructura WHILE

```
DELIMITER //

CREATE PROCEDURE inserir_alumnes_prova(IN p_n INT)
BEGIN
    DECLARE i INT DEFAULT 1;
    WHILE i <= p_n DO
        INSERT INTO alumne (nom, cognoms)
```

```
        VALUES (CONCAT('Nom_', i), CONCAT('Cognom_', i));
        SET i = i + 1;
    END WHILE;
END //

DELIMITER ;

CALL inserir_alumnes_prova(10);
```

**Estructura REPEAT**

```
DELIMITER //

CREATE PROCEDURE exemple_repeat()
BEGIN
    DECLARE v_count INT DEFAULT 0;
    REPEAT
        SET v_count = v_count + 1;
    UNTIL v_count >= 5
    END REPEAT;
    SELECT v_count;
END //

DELIMITER ;
```

**Cursors**

```
DELIMITER //

CREATE PROCEDURE llistar_alumnes_aprovats(IN p_id_modul INT)
BEGIN
    DECLARE v_nom       VARCHAR(50);
    DECLARE v_cognoms   VARCHAR(100);
    DECLARE v_nota      DECIMAL(4,2);
    DECLARE fi          BOOLEAN DEFAULT FALSE;

    DECLARE cur CURSOR FOR
        SELECT a.nom, a.cognoms, m.nota_final
        FROM   matricula m
               INNER JOIN alumne a ON m.id_alumne = a.id_alumne
        WHERE  m.id_modul = p_id_modul AND m.nota_final >= 5;

    DECLARE CONTINUE HANDLER FOR NOT FOUND SET fi = TRUE;

    OPEN cur;
    bucle: LOOP
        FETCH cur INTO v_nom, v_cognoms, v_nota;
        IF fi THEN LEAVE bucle; END IF;
```

```
        SELECT CONCAT(v_nom, ' ', v_cognoms, ' - ', v_nota) AS
 ↪  alumne_aprovat;
    END LOOP bucle;
    CLOSE cur;
END //

DELIMITER ;
```

# 7. Gestió de la seguretat de les dades

## 7.1. Còpies de seguretat

### mysqldump (còpia lògica)

```
# Còpia completa d'una base de dades
mysqldump -u root -p institut > /backup/institut_$(date +%Y%m%d).sql

# Còpia de múltiples bases de dades
mysqldump -u root -p --databases institut rrhh > /backup/multi_$(date
 ↪  +%Y%m%d).sql

# Còpia de totes les bases de dades
mysqldump -u root -p --all-databases > /backup/total_$(date
 ↪  +%Y%m%d).sql

# Còpia d'una taula concreta
mysqldump -u root -p institut alumne > /backup/alumne_$(date
 ↪  +%Y%m%d).sql

# Opcions útils:
# --single-transaction   → còpia consistent sense bloquejar (InnoDB)
# --routines             → inclou procediments i funcions
# --triggers             → inclou disparadors
# --events               → inclou esdeveniments planificats
mysqldump -u root -p --single-transaction --routines --triggers
 ↪  institut \
    > /backup/institut_complet_$(date +%Y%m%d).sql
```

### Restauració

```
# Restaurar una còpia
mysql -u root -p institut < /backup/institut_20241015.sql

# Crear la BD prèviament si no existeix
mysql -u root -p -e "CREATE DATABASE IF NOT EXISTS institut;"
mysql -u root -p institut < /backup/institut_20241015.sql
```

**mariabackup (còpia física, per a entorns de producció)**

```
# Còpia completa
mariabackup --backup --target-dir=/backup/full --user=root
 ↪  --password=secret

# Preparar la còpia per a la restauració
mariabackup --prepare --target-dir=/backup/full

# Restaurar
systemctl stop mariadb
mariabackup --copy-back --target-dir=/backup/full
chown -R mysql:mysql /var/lib/mysql
systemctl start mariadb
```

## 7.2. Importació i exportació de dades

**Exportar a CSV**

```
SELECT id_alumne, nom, cognoms, email
FROM   alumne
INTO OUTFILE '/tmp/alumnes.csv'
    FIELDS TERMINATED BY ','
    OPTIONALLY ENCLOSED BY '"'
    LINES TERMINATED BY '\n';
```

**Importar des de CSV**

```
LOAD DATA INFILE '/tmp/alumnes.csv'
INTO TABLE alumne
    FIELDS TERMINATED BY ','
    OPTIONALLY ENCLOSED BY '"'
    LINES TERMINATED BY '\n'
    (nom, cognoms, email);
```

**Exportar/importar amb mysqldump i format XML**

```
# Exportar com a XML
mysql -u root -p --xml -e "SELECT * FROM alumne" institut > alumnes.xml
```

## 7.3. Fitxers de registre (logs)

MariaDB disposa de diversos fitxers de registre configurables a `/etc/mysql/mariadb.conf.d/50-server.cnf`:

```
[mysqld]
# Registre d'errors (sempre actiu)
log_error = /var/log/mysql/error.log

# Registre general de consultes
general_log        = 1
general_log_file   = /var/log/mysql/general.log

# Registre de consultes lentes
slow_query_log     = 1
slow_query_log_file = /var/log/mysql/slow.log
long_query_time    = 2   # en segons

# Registre binari (per a replicació i recuperació puntual)
log_bin            = /var/log/mysql/mariadb-bin
binlog_format      = ROW
expire_logs_days   = 7
```

### Consultar els logs des de SQL

```
-- Variables relacionades amb els logs
SHOW VARIABLES LIKE '%log%';

-- Fitxers de log binari existents
SHOW BINARY LOGS;

-- Contingut d'un log binari
SHOW BINLOG EVENTS IN 'mariadb-bin.000001';
```

## 7.4. Transferència de dades entre sistemes gestors

Per migrar dades entre sistemes gestors heterogenis (p. ex. de PostgreSQL a MariaDB):

1. **Exportar** les dades de l'origen en format CSV o SQL genèric.
2. **Adaptar** els tipus de dades i la sintaxi SQL si cal.
3. **Importar** al destí amb `LOAD DATA INFILE` o `mysql < fitxer.sql`.

Eines específiques: **pgloader** (migra des de PostgreSQL, SQLite, CSV directament a MariaDB), **DBeaver** (permet transferència visual entre SGBD).

```
# Exemple amb pgloader (des de PostgreSQL a MariaDB)
pgloader postgres://usuari:pass@localhost/bd_origen \
         mysql://root:pass@localhost/bd_destino
```

**Versions d'aquest document**

- HTML - 0372.html
- PDF - 0372.pdf
- ODT - 0372.odt
- MD - 0372.md