Generalitat de Catalunya
Departament d'Educació
Institut Thos i Codina

Mòdul 0484 - Bases de dades

Domini Públic (CC0)

Índex

Base de dades d'exemple 1

1. Emmagatzematge de la informació **1**
1.1. Fitxers 1
1.2. Bases de dades: conceptes i tipus 1
1.3. Sistemes gestors de bases de dades (SGBD) 2
1.4. Bases de dades distribuïdes i fragmentació 2
1.5. Legislació sobre protecció de dades 3
1.6. Big Data i intel·ligència de negocis 3

2. Bases de dades relacionals: creació i estructura (DDL i DCL) **3**
2.1. Model relacional i terminologia 3
2.2. Instal·lació i connexió 4
2.3. Tipus de dades 4
2.4. Creació de taules i restriccions 4
2.5. El valor NULL 6
2.6. Modificar i esborrar estructures 6
2.7. Índexs 6
2.8. Vistes 7
2.9. Usuaris i privilegis (DCL) 7
2.10. Eines gràfiques i assistents 8

3. Realització de consultes (DQL) **8**
3.1. Estructura general 8
3.2. Projecció, selecció i ordenació 8
3.3. Operadors 8
3.4. Funcions habituals 9
3.5. Consultes de resum i agrupament 9
3.6. Composicions internes (`INNER JOIN`) 9
3.7. Composicions externes 10
3.8. Subconsultes 10
3.9. Combinació de múltiples seleccions 11
3.10. Funcions de finestra (ampliació) 11
3.11. Optimització de consultes 11

4. Tractament de dades (DML i transaccions) **12**
4.1. Inserció 12
4.2. Modificació i supressió 12
4.3. Integritat referencial 13
4.4. Transaccions 13
4.5. Concurrència, bloquejos i nivells d'aïllament 14

5. Programació de bases de dades **15**
5.1. Automatització de tasques i guions 15
5.2. Variables 15
5.3. Funcions emmagatzemades 15
5.4. Estructures de control de flux 16
5.5. Procediments emmagatzemats 16
5.6. Excepcions 17

5.7. Cursors 17
5.8. Disparadors (*triggers*) 18
5.9. Esdeveniments programats 19

6. Diagrames Entitat/Relació i normalització **19**
6.1. El model E/R 19
6.2. Model E/R ampliat 20
6.3. Pas de E/R a model relacional 20
6.4. Restriccions semàntiques 20
6.5. Normalització 20

7. Bases de dades no relacionals **21**
7.1. Característiques 21
7.2. Tipus principals 21
7.3. Elements d'una base documental (MongoDB) 22
7.4. Instal·lació i connexió (MongoDB) 22
7.5. Operacions CRUD 22
7.6. Agregació i índexs 23
7.7. Modelatge: incrustar o referenciar? 23
7.8. Eines d'administració 24

8. Recursos recomanats **24**

Cicle formatiu de grau superior: Desenvolupament d'aplicacions web (DAW)

Durada: 198 hores (132 al centre + 66 d'estada a l'empresa)

- SGBD de referència: **MariaDB / MySQL** (relacional) i **MongoDB** (no relacional)

Base de dades d'exemple

Al llarg d'aquest document fem servir una base de dades d'una botiga en línia senzilla, `botiga`:

Taula	Descripció
`client`	Persones que fan comandes
`categoria`	Categories de producte
`producte`	Articles a la venda
`comanda`	Capçalera de la comanda
`linia_comanda`	Productes inclosos en cada comanda

1. Emmagatzematge de la informació

1.1. Fitxers

Abans de les bases de dades, la informació es guardava en fitxers. Segons l'organització:

Tipus	Descripció	Avantatge	Inconvenient
Pla (seqüencial)	Registres un darrere l'altre (CSV, TXT)	Simple, portable	Cercar és lent (cal llegir-ho tot)
Indexat	Fitxer de dades + fitxer d'índex	Cerca ràpida per clau	Cal mantenir l'índex
Accés directe (hash)	La posició es calcula a partir de la clau	Accés molt ràpid per clau	Mal ús de l'espai, cerques per rang difícils

Problemes dels fitxers: **redundància**, **inconsistència**, **dependència dels programes** respecte al format, manca de **concurrència** i de **seguretat**. Les bases de dades neixen per resoldre'ls.

1.2. Bases de dades: conceptes i tipus

Una **base de dades** és un conjunt organitzat de dades relacionades entre si, emmagatzemades amb un mínim de redundància i accessibles per a diversos usuaris i aplicacions.

Segons el model de dades:

- **Relacional**: taules, files i columnes (MySQL, PostgreSQL, Oracle, SQL Server).
- **Jeràrquic** i **en xarxa**: models històrics.
- **Orientat a objectes**.
- **No relacional (NoSQL)**: clau-valor, documental, columnar, de grafs.

Segons la ubicació de la informació:

- **Centralitzada**: tot en un sol servidor.
- **Distribuïda**: repartida en diversos nodes connectats per xarxa.

1.3. Sistemes gestors de bases de dades (SGBD)

Un **SGBD** és el programari que permet crear, mantenir i accedir a les bases de dades.

Funcions principals:

- **Definició** de dades (DDL).
- **Manipulació** de dades (DML).
- **Control**: seguretat, integritat, concurrència, còpies i recuperació.

Components típics: gestor d'emmagatzematge, processador de consultes (parser, optimitzador, executor), gestor de transaccions, gestor de bloquejos, catàleg o diccionari de dades.

Classificació: per model (relacional, NoSQL), per llicència (lliure o propietari), per ubicació (centralitzat o distribuït), per nombre d'usuaris (monousuari o multiusuari).

1.4. Bases de dades distribuïdes i fragmentació

Una base de dades distribuïda reparteix les dades entre diversos nodes però es presenta a l'usuari com una sola.

Avantatges: disponibilitat, escalabilitat, proximitat de les dades als usuaris. **Inconvenients:** complexitat, coherència entre rèpliques, cost de les consultes entre nodes.

Tècniques de fragmentació:

- **Horitzontal**: es reparteixen files (p. ex. clients d'Europa en un node i d'Amèrica en un altre).
- **Vertical**: es reparteixen columnes (dades d'ús freqüent separades de les pesades, com fotografies).
- **Mixta**: combinació de les dues.

Regles perquè la fragmentació sigui correcta: **completesa** (no es perd cap dada), **reconstrucció** (es pot recompondre l'original) i **disjunció** (sense duplicats innecessaris).

1.5. Legislació sobre protecció de dades

- **RGPD** (Reglament (UE) 2016/679) i **LOPDGDD** (Llei orgànica 3/2018) a Espanya.
- Conceptes: dada personal, responsable i encarregat del tractament, consentiment, minimització de dades, limitació de la finalitat.
- **Drets ARSULIPO**: accés, rectificació, supressió, limitació, portabilitat i oposició.
- Mesures tècniques que un administrador de bases de dades ha de conèixer: control d'accés per privilegis, xifratge, còpies de seguretat, registre d'accessos (auditoria), pseudonimització.
- Autoritat de control: **AEPD** i, a Catalunya, l'**Autoritat Catalana de Protecció de Dades (APDCAT)**.

1.6. Big Data i intel·ligència de negocis

- **Big Data**: volums de dades que superen les capacitats de les eines tradicionals. Les **5 V**: Volum, Velocitat, Varietat, Veracitat i Valor.
- **Business Intelligence (BI)**: processos i eines que transformen dades en informació per a la presa de decisions. Conceptes clau: **data warehouse**, **ETL** (extreure, transformar, carregar), **OLAP** i quadres de comandament.
- Diferència entre **OLTP** (transaccions del dia a dia) i **OLAP** (anàlisi).

2. Bases de dades relacionals: creació i estructura (DDL i DCL)

2.1. Model relacional i terminologia

Terme relacional	Equivalent informal
Relació	Taula
Tupla	Fila / registre
Atribut	Columna / camp
Domini	Conjunt de valors permesos
Grau	Nombre de columnes
Cardinalitat	Nombre de files

Claus:

- **Clau candidata**: conjunt mínim d'atributs que identifica una fila.
- **Clau primària (PK)**: la candidata triada. No admet `NULL` ni repeticions.
- **Clau alternativa**: candidata no triada com a primària.
- **Clau forana (FK)**: atributs que referencien la clau primària d'una altra taula.

2.2. Instal·lació i connexió

```
sudo apt install mariadb-server
sudo mysql_secure_installation
sudo mysql -u root -p
```

```
CREATE DATABASE botiga CHARACTER SET utf8mb4 COLLATE
 ↪  utf8mb4_catalan_ci;
USE botiga;
```

> Fes servir sempre `utf8mb4`: és el joc de caràcters complet d'UTF-8. El `utf8` de MySQL només cobreix 3 bytes per caràcter.

2.3. Tipus de dades

Categoria	Tipus habituals
Enters	`TINYINT`, `SMALLINT`, `INT`, `BIGINT`
Decimals exactes	`DECIMAL(p,s)` (imports, preus)
Decimals aproximats	`FLOAT`, `DOUBLE`
Text	`CHAR(n)`, `VARCHAR(n)`, `TEXT`
Data i hora	`DATE`, `TIME`, `DATETIME`, `TIMESTAMP`
Lògic	`BOOLEAN` (sinònim de `TINYINT(1)`)
Enumerats	`ENUM('a','b')`
Binaris / JSON	`BLOB`, `JSON`

Criteris de selecció: la mida mínima suficient, precisió exacta per als diners (`DECIMAL`, **mai** `FLOAT`), longituds raonables en `VARCHAR`.

2.4. Creació de taules i restriccions

```
CREATE TABLE categoria (
    id_categoria INT AUTO_INCREMENT PRIMARY KEY,
    nom          VARCHAR(50) NOT NULL UNIQUE
);

CREATE TABLE client (
    id_client   INT AUTO_INCREMENT PRIMARY KEY,
    nom         VARCHAR(60)  NOT NULL,
    cognoms     VARCHAR(100) NOT NULL,
    correu      VARCHAR(120) NOT NULL UNIQUE,
    telefon     VARCHAR(15),
    data_alta   DATE         NOT NULL DEFAULT (CURRENT_DATE),
    CONSTRAINT ck_correu CHECK (correu LIKE '%@%')
);

CREATE TABLE producte (
    id_producte  INT AUTO_INCREMENT PRIMARY KEY,
    nom          VARCHAR(100)  NOT NULL,
```

```
    preu         DECIMAL(8,2)  NOT NULL,
    estoc        INT           NOT NULL DEFAULT 0,
    id_categoria INT           NOT NULL,
    CONSTRAINT ck_preu  CHECK (preu >= 0),
    CONSTRAINT ck_estoc CHECK (estoc >= 0),
    CONSTRAINT fk_prod_cat FOREIGN KEY (id_categoria)
        REFERENCES categoria (id_categoria)
        ON UPDATE CASCADE
        ON DELETE RESTRICT
);

CREATE TABLE comanda (
    id_comanda INT AUTO_INCREMENT PRIMARY KEY,
    id_client  INT NOT NULL,
    data       DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
    estat      ENUM('pendent','enviada','entregada','cancel·lada') NOT
    ↪  NULL DEFAULT 'pendent',
    CONSTRAINT fk_com_cli FOREIGN KEY (id_client)
        REFERENCES client (id_client)
        ON DELETE RESTRICT
);

CREATE TABLE linia_comanda (
    id_comanda  INT,
    id_producte INT,
    quantitat   INT           NOT NULL,
    preu_unitat DECIMAL(8,2)  NOT NULL,
    PRIMARY KEY (id_comanda, id_producte),
    CONSTRAINT ck_quant CHECK (quantitat > 0),
    CONSTRAINT fk_lin_com FOREIGN KEY (id_comanda)
        REFERENCES comanda (id_comanda) ON DELETE CASCADE,
    CONSTRAINT fk_lin_pro FOREIGN KEY (id_producte)
        REFERENCES producte (id_producte)
);
```

Restriccions de validació: `NOT NULL`, `UNIQUE`, `PRIMARY KEY`, `FOREIGN KEY`, `CHECK`, `DEFAULT`.

Accions referencials de les claus foranes:

Acció	Efecte
`RESTRICT` / `NO ACTION`	Rebutja l'operació si hi ha files filles
`CASCADE`	Propaga l'esborrat o la modificació a les files filles
`SET NULL`	Posa `NULL` a la clau forana (la columna ha d'admetre-ho)
`SET DEFAULT`	Valor per defecte (no suportat per InnoDB)

2.5. El valor NULL

`NULL` significa «desconegut» o «no aplicable»; **no és** zero ni cadena buida.

```
SELECT * FROM client WHERE telefon = NULL;   -- MAI funciona (retorna
 ↪  0 files)
SELECT * FROM client WHERE telefon IS NULL;  -- correcte
SELECT COALESCE(telefon, 'sense telèfon') FROM client;
```

Qualsevol comparació amb `NULL` dona `UNKNOWN`. Les funcions d'agregació (excepte `COUNT(*)`) ignoren els `NULL`.

2.6. Modificar i esborrar estructures

```
ALTER TABLE client ADD COLUMN poblacio VARCHAR(60);
ALTER TABLE client MODIFY telefon VARCHAR(20);
ALTER TABLE client DROP COLUMN poblacio;
ALTER TABLE producte ADD CONSTRAINT uq_prod_nom UNIQUE (nom);
DROP TABLE IF EXISTS prova;
TRUNCATE TABLE prova;       -- buida la taula (no es pot desfer amb
 ↪  ROLLBACK)
```

2.7. Índexs

Un **índex** és una estructura auxiliar (normalment un arbre B+) que accelera les cerques a costa d'espai i de lentitud en les escriptures.

```
CREATE INDEX idx_client_cognoms ON client (cognoms);
CREATE UNIQUE INDEX idx_client_correu ON client (correu);
CREATE INDEX idx_com_client_data ON comanda (id_client, data);  --
 ↪  compost
SHOW INDEX FROM client;
DROP INDEX idx_client_cognoms ON client;
```

Criteris: indexar columnes que apareixen sovint a `WHERE`, `JOIN` i `ORDER BY`; evitar indexar columnes amb pocs valors diferents o taules molt petites. Les PK i UNIQUE ja creen un índex.

2.8. Vistes

Una **vista** és una consulta desada amb nom que es comporta com una taula virtual.

```
CREATE VIEW v_resum_comanda AS
SELECT c.id_comanda, cl.nom, cl.cognoms, c.data,
       SUM(l.quantitat * l.preu_unitat) AS total
FROM comanda c
JOIN client cl        ON cl.id_client = c.id_client
JOIN linia_comanda l  ON l.id_comanda = c.id_comanda
GROUP BY c.id_comanda, cl.nom, cl.cognoms, c.data;

SELECT * FROM v_resum_comanda WHERE total > 100;
DROP VIEW v_resum_comanda;
```

Usos: simplificar consultes, **restringir l'accés** a columnes o files, mantenir compatibilitat si canvia l'esquema.

2.9. Usuaris i privilegis (DCL)

```
CREATE USER 'web'@'localhost' IDENTIFIED BY 'ContrasenyaForta!2026';
CREATE USER 'informes'@'%'     IDENTIFIED BY 'AltraContrasenya!2026';

GRANT SELECT, INSERT, UPDATE, DELETE ON botiga.* TO 'web'@'localhost';
GRANT SELECT ON botiga.v_resum_comanda TO 'informes'@'%';

SHOW GRANTS FOR 'web'@'localhost';
REVOKE DELETE ON botiga.* FROM 'web'@'localhost';
DROP USER 'informes'@'%';
FLUSH PRIVILEGES;
```

Principi del mínim privilegi: cada usuari ha de tenir només els permisos imprescindibles. Una aplicació web **mai** s'ha de connectar com a `root`.

Rols (MariaDB 10.0.5+):

```
CREATE ROLE editor;
GRANT SELECT, INSERT, UPDATE ON botiga.* TO editor;
GRANT editor TO 'web'@'localhost';
```

2.10. Eines gràfiques i assistents

`DBeaver`, `MySQL Workbench`, `phpMyAdmin`, `HeidiSQL`. Són útils per explorar i dissenyar, però cal saber fer el mateix amb sentències SQL: són reproduïbles, versionables i automatitzables.

3. Realització de consultes (DQL)

3.1. Estructura general

```
SELECT   [DISTINCT] columnes | expressions
FROM     taula
[JOIN    ...]
[WHERE   condició]
[GROUP BY columnes]
[HAVING  condició sobre grups]
[ORDER BY columnes [ASC|DESC]]
[LIMIT   n [OFFSET m]];
```

Ordre d'execució lògic: `FROM` → `WHERE` → `GROUP BY` → `HAVING` → `SELECT` → `ORDER BY` → `LIMIT`. Això explica per què no es pot fer servir un àlies del `SELECT` dins del `WHERE`.

3.2. Projecció, selecció i ordenació

```
SELECT nom, preu FROM producte;                          -- projecció
SELECT DISTINCT estat FROM comanda;                      -- sense
 ↪  repeticions
SELECT nom, preu * 1.21 AS preu_amb_iva FROM producte; -- expressió i
 ↪  àlies

SELECT * FROM producte WHERE preu > 20;                  -- selecció
SELECT * FROM producte ORDER BY preu DESC, nom ASC;
SELECT * FROM producte ORDER BY preu DESC LIMIT 5;     -- els 5 més
 ↪  cars
```

3.3. Operadors

Tipus	Operadors
Comparació	`=`, `<>` o `!=`, `<`, `>`, `<=`, `>=`
Rang i llista	`BETWEEN a AND b`, `IN (...)`
Patrons	`LIKE` (`%` = qualsevol cadena, `_` = un caràcter), `REGEXP`
Nuls	`IS NULL`, `IS NOT NULL`
Lògics	`AND`, `OR`, `NOT` (precedència: `NOT` > `AND` > `OR`)

```
SELECT * FROM client WHERE cognoms LIKE 'Gar%';
SELECT * FROM producte WHERE preu BETWEEN 10 AND 50 AND estoc > 0;
SELECT * FROM comanda WHERE estat IN ('pendent','enviada');
```

Usa parèntesis quan barregis `AND` i `OR`: `WHERE a AND (b OR c)`.

3.4. Funcions habituals

```
SELECT UPPER(nom), LENGTH(nom), CONCAT(nom,' ',cognoms) FROM client;
SELECT ROUND(preu, 1), CEIL(preu), FLOOR(preu) FROM producte;
SELECT NOW(), CURDATE(), YEAR(data), DATEDIFF(CURDATE(), data_alta)
 ↪  FROM client;
SELECT IF(estoc = 0, 'Esgotat', 'Disponible') FROM producte;
SELECT CASE WHEN preu < 10 THEN 'barat'
            WHEN preu < 50 THEN 'mitjà'
            ELSE 'car' END AS gamma
FROM producte;
```

3.5. Consultes de resum i agrupament

Funcions d'agregació: `COUNT`, `SUM`, `AVG`, `MIN`, `MAX`.

```
SELECT COUNT(*) AS total_clients FROM client;
SELECT COUNT(telefon) FROM client;               -- ignora NULL
SELECT AVG(preu), MIN(preu), MAX(preu) FROM producte;

SELECT id_categoria, COUNT(*) AS num_productes, AVG(preu) AS
 ↪  preu_mitja
FROM producte
GROUP BY id_categoria
HAVING COUNT(*) >= 3
ORDER BY preu_mitja DESC;
```

`WHERE` filtra files abans d'agrupar; `HAVING` filtra grups després d'agrupar.

3.6. Composicions internes (`INNER JOIN`)

Retornen només les files que tenen correspondència a les dues taules.

```
SELECT p.nom AS producte, c.nom AS categoria, p.preu
FROM producte p
INNER JOIN categoria c ON c.id_categoria = p.id_categoria;

-- Tres taules
SELECT cl.nom, cl.cognoms, co.id_comanda, p.nom AS producte,
 ↪  l.quantitat
```

```
FROM client cl
JOIN comanda co       ON co.id_client   = cl.id_client
JOIN linia_comanda l  ON l.id_comanda   = co.id_comanda
JOIN producte p       ON p.id_producte  = l.id_producte;
```

3.7. Composicions externes

```
-- Tots els clients, tinguin o no comandes
SELECT cl.nom, co.id_comanda
FROM client cl
LEFT JOIN comanda co ON co.id_client = cl.id_client;

-- Clients que mai han comprat
SELECT cl.*
FROM client cl
LEFT JOIN comanda co ON co.id_client = cl.id_client
WHERE co.id_comanda IS NULL;
```

`RIGHT JOIN` és el simètric. MySQL/MariaDB no tenen `FULL OUTER JOIN`: es simula amb `UNION` d'un `LEFT` i un `RIGHT`.

També existeix el `CROSS JOIN` (producte cartesià) i l'**autocomposició** (una taula amb ella mateixa mitjançant àlies diferents).

3.8. Subconsultes

```
-- Escalar: productes més cars que la mitjana
SELECT nom, preu FROM producte
WHERE preu > (SELECT AVG(preu) FROM producte);

-- Amb IN
SELECT nom FROM client
WHERE id_client IN (SELECT id_client FROM comanda WHERE estat =
 ↪  'pendent');

-- Amb EXISTS (sovint més eficient)
SELECT nom FROM client cl
WHERE EXISTS (SELECT 1 FROM comanda co WHERE co.id_client =
 ↪  cl.id_client);

-- Correlacionada: producte més car de cada categoria
SELECT p.nom, p.preu, p.id_categoria
FROM producte p
WHERE p.preu = (SELECT MAX(preu) FROM producte WHERE id_categoria =
 ↪  p.id_categoria);

-- A FROM (taula derivada)
SELECT t.id_categoria, t.mitjana
```

```
FROM (SELECT id_categoria, AVG(preu) AS mitjana
      FROM producte GROUP BY id_categoria) t
WHERE t.mitjana > 25;
```

Alerta: `NOT IN` amb una subconsulta que retorni `NULL` no retorna cap fila. Prefereix `NOT EXISTS`.

3.9. Combinació de múltiples seleccions

```
SELECT correu FROM client
UNION            -- sense duplicats (UNION ALL els conserva)
SELECT correu FROM proveidor;
```

`INTERSECT` i `EXCEPT` estan disponibles en MariaDB 10.3+ (en MySQL, `INTERSECT`/`EXCEPT` des de la 8.0.31).

Les consultes combinades han de tenir el **mateix nombre de columnes** i tipus compatibles.

3.10. Funcions de finestra (ampliació)

```
SELECT nom, preu,
       RANK()       OVER (ORDER BY preu DESC) AS posicio,
       AVG(preu)    OVER (PARTITION BY id_categoria) AS
        ↪  mitjana_categoria
FROM producte;
```

3.11. Optimització de consultes

- Fes servir **`EXPLAIN`** per veure el pla d'execució.
- Evita `SELECT *`: demana només les columnes necessàries.
- Crea **índexs** en columnes de `WHERE`, `JOIN` i `ORDER BY`.
- No apliquis funcions sobre columnes indexades al `WHERE` (`WHERE YEAR(data) = 2026` no usa l'índex; millor `WHERE data >= '2026-01-01' AND data < '2027-01-01'`).
- Substitueix subconsultes correlacionades per `JOIN` quan sigui possible.
- Un `LIKE '%text'` (comodí al principi) no aprofita l'índex.
- Limita els resultats amb `LIMIT`.

```
EXPLAIN SELECT * FROM client WHERE cognoms = 'Garcia';
```

Mira les columnes `type` (evita `ALL`), `key` (índex usat) i `rows` (estimació de files llegides).

4. Tractament de dades (DML i transaccions)

4.1. Inserció

```
INSERT INTO categoria (nom) VALUES ('Informàtica');

INSERT INTO client (nom, cognoms, correu, telefon)
VALUES ('Anna', 'Puig Serra', 'anna@exemple.cat', '600111222'),
       ('Jordi', 'Vila Roca', 'jordi@exemple.cat', NULL);

-- Inserir el resultat d'una consulta
INSERT INTO client_arxiu (id_client, nom, cognoms)
SELECT id_client, nom, cognoms FROM client WHERE data_alta <
 ↪  '2020-01-01';

-- Inserir o actualitzar si ja existeix
INSERT INTO producte (id_producte, nom, preu, id_categoria, estoc)
VALUES (1, 'Ratolí', 12.50, 1, 30)
ON DUPLICATE KEY UPDATE preu = VALUES(preu), estoc = VALUES(estoc);
```

4.2. Modificació i supressió

```
UPDATE producte SET preu = preu * 1.05 WHERE id_categoria = 1;
UPDATE client SET telefon = NULL WHERE telefon = '';

DELETE FROM comanda WHERE estat = 'cancel·lada' AND data <
 ↪  '2025-01-01';
```

> **Sempre escriu el `WHERE` primer**, o comprova amb un `SELECT` equivalent abans d'executar `UPDATE` o `DELETE`. Sense `WHERE` s'afecten **totes** les files.

Subconsultes i composicions en les sentències d'edició:

```
UPDATE producte
SET estoc = 0
WHERE id_producte IN (SELECT id_producte FROM producte_descatalogat);

UPDATE producte p
JOIN categoria c ON c.id_categoria = p.id_categoria
SET p.preu = p.preu * 0.9
WHERE c.nom = 'Oferta';

DELETE co FROM comanda co
LEFT JOIN client cl ON cl.id_client = co.id_client
WHERE cl.id_client IS NULL;
```

4.3. Integritat referencial

Amb les claus foranes (InnoDB), el SGBD impedeix deixar files òrfenes:

```
DELETE FROM categoria WHERE id_categoria = 1;
-- ERROR 1451: Cannot delete or update a parent row: a foreign key
 ↪  constraint fails
```

El comportament es controla amb `ON DELETE` / `ON UPDATE` (vegeu l'apartat 2.4).

4.4. Transaccions

Una **transacció** és un conjunt d'operacions que s'executen com una unitat indivisible. Compleix les propietats **ACID**:

Propietat	Significat
Atomicitat	Tot o res
Consistència	Passa d'un estat vàlid a un altre estat vàlid
Insolament (*isolation*)	Les transaccions concurrents no s'interfereixen
Durabilitat	Un cop confirmada, perdura encara que hi hagi una caiguda

```
START TRANSACTION;

INSERT INTO comanda (id_client) VALUES (1);
SET @id = LAST_INSERT_ID();

INSERT INTO linia_comanda (id_comanda, id_producte, quantitat,
 ↪  preu_unitat)
VALUES (@id, 3, 2, 19.90);

UPDATE producte SET estoc = estoc - 2 WHERE id_producte = 3;

COMMIT;     -- confirma
-- ROLLBACK; -- desfà tot
```

Anul·lació parcial amb **punts de salvaguarda**:

```
START TRANSACTION;
UPDATE producte SET estoc = estoc - 1 WHERE id_producte = 3;
SAVEPOINT despres_estoc;
UPDATE client SET telefon = '000' WHERE id_client = 1;
ROLLBACK TO SAVEPOINT despres_estoc;   -- només desfà l'últim UPDATE
COMMIT;
```

Notes:

- Només els motors transaccionals (**InnoDB**) suporten transaccions; MyISAM no.
- Per defecte `autocommit = 1`: cada sentència és una transacció pròpia. Consulta-ho amb `SELECT @@autocommit;`.
- Les sentències DDL (`CREATE`, `ALTER`, `DROP`, `TRUNCATE`) fan un **commit implícit**.

4.5. Concurrència, bloquejos i nivells d'aïllament

Quan diverses transaccions treballen a la vegada poden aparèixer anomalies:

Anomalia	Descripció
Lectura bruta (*dirty read*)	Llegir dades no confirmades d'una altra transacció
Lectura no repetible	Dues lectures de la mateixa fila donen valors diferents
Lectura fantasma	Una segona consulta troba files noves

Nivells d'aïllament (estàndard SQL):

Nivell	Lectura bruta	No repetible	Fantasma
`READ UNCOMMITTED`	Possible	Possible	Possible
`READ COMMITTED`	No	Possible	Possible
`REPEATABLE READ` (per defecte a InnoDB)	No	No	En gran part evitades (*next-key locks*)
`SERIALIZABLE`	No	No	No

```
SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;
SELECT @@transaction_isolation;
```

Bloquejos explícits:

```
START TRANSACTION;
SELECT estoc FROM producte WHERE id_producte = 3 FOR UPDATE;  --
 ↪  bloqueig d'escriptura sobre la fila
UPDATE producte SET estoc = estoc - 1 WHERE id_producte = 3;
COMMIT;
```

- **Bloquejos compartits (S)** per llegir; **exclusius (X)** per escriure.
- **Interbloqueig** (*deadlock*): dues transaccions s'esperen mútuament. InnoDB en detecta i avorta una; l'aplicació ha de saber reintentar. Es redueixen accedint sempre als recursos en el mateix ordre i fent transaccions curtes.

Pràctica a l'aula: obre dues sessions de client i reprodueix cada anomalia canviant el nivell d'aïllament.

5. Programació de bases de dades

5.1. Automatització de tasques i guions

Maneres d'automatitzar: guions SQL, procediments i funcions emmagatzemats, disparadors, esdeveniments programats, `cron` des del sistema operatiu.

Execució de guions:

```
mysql -u root -p botiga < esquema.sql
mysql -u root -p -e "SELECT COUNT(*) FROM botiga.client;"
```

```
SOURCE /home/ramon/scripts/dades.sql;
```

Editors: DBeaver, Workbench, VS Code amb extensions SQL, `vim`/`nano`.

5.2. Variables

```
-- Variables d'usuari (de sessió)
SET @iva = 0.21;
SELECT nom, preu * (1 + @iva) FROM producte;

-- Variables del sistema
SHOW VARIABLES LIKE 'autocommit';
SELECT @@version, @@sql_mode;
SET SESSION sql_mode = 'STRICT_ALL_TABLES';

-- Variables locals (només dins de rutines)
DECLARE v_total DECIMAL(10,2) DEFAULT 0;
```

5.3. Funcions emmagatzemades

Retornen **un valor** i es poden fer servir dins d'una sentència `SELECT`.

```
DELIMITER //

CREATE FUNCTION preu_amb_iva(p_preu DECIMAL(8,2), p_iva DECIMAL(4,2))
RETURNS DECIMAL(8,2)
DETERMINISTIC
BEGIN
    RETURN ROUND(p_preu * (1 + p_iva / 100), 2);
END //

DELIMITER ;

SELECT nom, preu_amb_iva(preu, 21) FROM producte;
```

`DELIMITER` canvia el terminador perquè el client no talli la definició al primer `;`.

5.4. Estructures de control de flux

```
DELIMITER //
CREATE FUNCTION categoria_client(p_id INT) RETURNS VARCHAR(10)
READS SQL DATA
BEGIN
    DECLARE v_total DECIMAL(10,2);

    SELECT COALESCE(SUM(l.quantitat * l.preu_unitat), 0) INTO v_total
    FROM comanda c JOIN linia_comanda l ON l.id_comanda = c.id_comanda
    WHERE c.id_client = p_id;

    IF v_total >= 1000 THEN
        RETURN 'or';
    ELSEIF v_total >= 300 THEN
        RETURN 'plata';
    ELSE
        RETURN 'bronze';
    END IF;
END //
DELIMITER ;
```

Altres estructures: `CASE ... WHEN ... END CASE`, bucles `WHILE ... DO ... END WHILE`, `REPEAT ... UNTIL ... END REPEAT`, `LOOP ... LEAVE etiqueta`.

5.5. Procediments emmagatzemats

No retornen valor directament; admeten paràmetres `IN`, `OUT` i `INOUT` i s'invoquen amb `CALL`.

```
DELIMITER //

CREATE PROCEDURE crear_comanda(
    IN  p_client   INT,
    IN  p_producte INT,
    IN  p_quant    INT,
    OUT p_id_comanda INT
)
BEGIN
    DECLARE v_preu DECIMAL(8,2);
    DECLARE v_estoc INT;

    DECLARE EXIT HANDLER FOR SQLEXCEPTION
    BEGIN
        ROLLBACK;
        RESIGNAL;
    END;

    START TRANSACTION;

    SELECT preu, estoc INTO v_preu, v_estoc
    FROM producte WHERE id_producte = p_producte FOR UPDATE;
```

```
    IF v_estoc < p_quant THEN
       SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Estoc insuficient';
    END IF;

    INSERT INTO comanda (id_client) VALUES (p_client);
    SET p_id_comanda = LAST_INSERT_ID();

    INSERT INTO linia_comanda VALUES (p_id_comanda, p_producte,
     ↪  p_quant, v_preu);
    UPDATE producte SET estoc = estoc - p_quant WHERE id_producte =
     ↪  p_producte;

    COMMIT;
END //

DELIMITER ;

CALL crear_comanda(1, 3, 2, @nova);
SELECT @nova;
```

Gestió: `SHOW PROCEDURE STATUS WHERE Db='botiga';`, `SHOW CREATE PROCEDURE crear_comanda;`, `DROP PROCEDURE crear_comanda;`.

5.6. Excepcions

- **`DECLARE ... HANDLER`** captura errors: `EXIT` (surt del bloc) o `CONTINUE` (segueix).
- Condicions: `SQLEXCEPTION`, `SQLWARNING`, `NOT FOUND`, codi d'error concret (`1062` = duplicat), o `SQLSTATE`.
- **`SIGNAL`** llança un error propi; **`RESIGNAL`** el rellança.

```
DECLARE CONTINUE HANDLER FOR 1062
    SET @duplicat = 1;
```

5.7. Cursors

Permeten recórrer fila a fila el resultat d'una consulta.

```
DELIMITER //
CREATE PROCEDURE total_estoc(OUT p_total INT)
BEGIN
    DECLARE v_estoc INT;
    DECLARE v_fi BOOLEAN DEFAULT FALSE;
    DECLARE cur CURSOR FOR SELECT estoc FROM producte;
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET v_fi = TRUE;

    SET p_total = 0;
    OPEN cur;
```

```
    bucle: LOOP
        FETCH cur INTO v_estoc;
        IF v_fi THEN LEAVE bucle; END IF;
        SET p_total = p_total + v_estoc;
    END LOOP;
    CLOSE cur;
END //
DELIMITER ;
```

Els cursors són lents comparats amb una sola sentència SQL: fes-los servir només quan no hi hagi alternativa basada en conjunts (`SELECT SUM(estoc) FROM producte` seria molt millor aquí).

5.8. Disparadors (*triggers*)

S'executen automàticament abans o després d'un `INSERT`, `UPDATE` o `DELETE`.

```
CREATE TABLE auditoria_preu (
    id          INT AUTO_INCREMENT PRIMARY KEY,
    id_producte INT,
    preu_antic  DECIMAL(8,2),
    preu_nou    DECIMAL(8,2),
    usuari      VARCHAR(80),
    moment      DATETIME DEFAULT CURRENT_TIMESTAMP
);

DELIMITER //
CREATE TRIGGER trg_preu_audit
AFTER UPDATE ON producte
FOR EACH ROW
BEGIN
    IF OLD.preu <> NEW.preu THEN
        INSERT INTO auditoria_preu (id_producte, preu_antic, preu_nou,
        ↪  usuari)
        VALUES (OLD.id_producte, OLD.preu, NEW.preu, CURRENT_USER());
    END IF;
END //
DELIMITER ;
```

`OLD` i `NEW` donen accés als valors anterior i posterior. Un `BEFORE` trigger pot validar o modificar `NEW`.

5.9. Esdeveniments programats

```
SET GLOBAL event_scheduler = ON;

CREATE EVENT ev_neteja_cancelades
ON SCHEDULE EVERY 1 DAY STARTS '2026-11-06 03:00:00'
DO
    DELETE FROM comanda
    WHERE estat = 'cancel·lada' AND data < NOW() - INTERVAL 1 YEAR;
```

6. Diagrames Entitat/Relació i normalització

6.1. El model E/R

- **Entitat**: objecte del món real del qual interessa guardar informació (*Client*, *Producte*).
  - **Entitat feble**: no té identificació pròpia; depèn d'una entitat forta (p. ex. *Línia de comanda* depèn de *Comanda*).
- **Atribut**: propietat d'una entitat. Pot ser **identificador**, **simple/compost**, **monovalent/multivalent**, **derivat**.
- **Relació**: associació entre entitats (*Client* **fa** *Comanda*).
- **Cardinalitat** (mínim, màxim): `(0,1)`, `(1,1)`, `(0,N)`, `(1,N)`.
- **Grau**: nombre d'entitats que participen (binària, ternària…).

Tipus de relacions per cardinalitat màxima: **1:1**, **1:N**, **N:M**.

Exemple de lectura: *«Un client pot fer zero o moltes comandes; una comanda és feta per un i només un client.»* `Client (1,1) ------ fa ------ (0,N) Comanda` segons la notació de Chen que s'ha fet servir.

Eines gràfiques: **draw.io**, **MySQL Workbench** (EER), **dbdiagram.io**, **Mermaid**.

Exemple en Mermaid:

```
erDiagram
    CLIENT   ||--o{ COMANDA        : fa
    COMANDA  ||--|{ LINIA_COMANDA  : conté
    PRODUCTE ||--o{ LINIA_COMANDA  : "apareix a"
    CATEGORIA ||--o{ PRODUCTE      : classifica
```

6.2. Model E/R ampliat

- **Generalització / especialització** (herència): *Persona* → *Client*, *Empleat*. Es classifica com a **total/parcial** i **disjunta/solapada**.
- **Agregació**: una relació es tracta com una entitat per poder-la relacionar amb una altra.

6.3. Pas de E/R a model relacional

Element E/R	Transformació
Entitat forta	Taula amb la seva PK
Entitat feble	Taula amb PK = PK de l'entitat forta + discriminant
Relació **1:N**	La **FK** va a la banda **N**
Relació **N:M**	**Nova taula** amb les PK de les dues entitats com a FK (i PK compostes) més els atributs de la relació
Relació **1:1**	FK (única) en una de les dues taules, o fusionar-les
Atribut multivalent	Nova taula amb FK a l'entitat
Generalització	(a) una taula per entitat, (b) només subtipus, o (c) una sola taula amb tipus

En l'exemple, la relació N:M entre *Comanda* i *Producte* esdevé la taula `linia_comanda`, que a més conté `quantitat` i `preu_unitat`.

6.4. Restriccions semàntiques

No tot es pot expressar amb l'esquema. Exemples de restriccions que cal **documentar** i implementar amb disparadors, `CHECK` o a l'aplicació:

- «Un client no pot tenir més de 5 comandes pendents.»
- «El preu d'una línia no pot diferir del preu del producte en més d'un 50 %.»
- «Una comanda enviada no es pot modificar.»

6.5. Normalització

Procés per eliminar redundàncies i anomalies d'inserció, esborrat i modificació.

Dependència funcional: `A → B` significa que el valor d'A determina el de B.

Forma normal	Condició
1FN	Tots els atributs són atòmics; no hi ha grups repetitius
2FN	1FN + cap atribut no clau depèn només d'una **part** de la clau (clau composta)
3FN	2FN + cap atribut no clau depèn d'un altre atribut no clau (**dependència transitiva**)
FNBC	Tot determinant és clau candidata

Exemple pas a pas. Taula inicial sense normalitzar:

id_comanda	client	correu_client	productes
1	Anna Puig	anna@exemple.cat	Ratolí, Teclat

1. **1FN**: `productes` no és atòmic → una fila per producte de la comanda.
2. **2FN**: amb clau `(id_comanda, producte)`, `client` i `correu_client` depenen només d'`id_comanda` → separar *Comanda* i *Línia*.
3. **3FN**: `correu_client` depèn de `client`, que no és clau → crear la taula *Client*.

Resultat: l'esquema `client` / `comanda` / `linia_comanda` / `producte` que hem fet servir.

> **Desnormalització controlada:** a vegades es dupliquen dades per rendiment (p. ex. `preu_unitat` a `linia_comanda`, que guarda el preu en el moment de la compra). Cal fer-ho conscientment i documentar-ho.

7. Bases de dades no relacionals

7.1. Característiques

- **Sense esquema fix** (o esquema flexible).
- **Escalabilitat horitzontal** (afegir nodes) en lloc de vertical.
- Sovint prioritzen disponibilitat i rendiment per sobre de la consistència immediata (**teorema CAP**: només es poden garantir dues de tres propietats entre Consistència, Disponibilitat i tolerància a Particions).
- Model de consistència **BASE** (*Basically Available, Soft state, Eventually consistent*) davant d'ACID.
- Habitualment no suporten `JOIN` ni transaccions multidocument tan completes com el SQL.

7.2. Tipus principals

Tipus	Estructura	Exemples	Usos típics
Clau-valor	Parelles clau → valor	Redis, DynamoDB	Caché, sessions, carretons
Documental	Documents JSON/BSON	MongoDB, CouchDB	Catàlegs, CMS, aplicacions web
Columnar	Famílies de columnes	Cassandra, HBase	Sèries temporals, grans volums
Grafs	Nodes i arestes	Neo4j	Xarxes socials, recomanacions

7.3. Elements d'una base documental (MongoDB)

Relacional	MongoDB
Base de dades	Base de dades
Taula	**Col·lecció**
Fila	**Document**
Columna	**Camp**
`JOIN`	`$lookup` o documents incrustats
Clau primària	`_id`

7.4. Instal·lació i connexió (MongoDB)

```
# Via contenidor
docker run -d --name mongo -p 27017:27017 mongo:7
docker exec -it mongo mongosh
```

```
use botiga
show collections
```

7.5. Operacions CRUD

```
// Create
db.producte.insertOne({
  nom: "Ratolí òptic",
  preu: 12.5,
  estoc: 30,
  categoria: "Informàtica",
  etiquetes: ["perifèric", "usb"],
  especificacions: { dpi: 1600, sensors: 1 }
})

db.producte.insertMany([
  { nom: "Teclat", preu: 25, estoc: 10, categoria: "Informàtica" },
  { nom: "Monitor", preu: 180, estoc: 4, categoria: "Informàtica" }
])

// Read
db.producte.find()
db.producte.find({ preu: { $gt: 20 } }, { nom: 1, preu: 1, _id: 0 })
db.producte.find({ etiquetes: "usb" })
db.producte.find({ $or: [{ preu: { $lt: 15 } }, { estoc: 0 }]
 ↪  }).sort({ preu: -1 }).limit(5)

// Update
db.producte.updateOne({ nom: "Teclat" }, { $set: { preu: 27 }, $inc: {
 ↪  estoc: -1 } })
```

```
db.producte.updateMany({ categoria: "Informàtica" }, { $mul: { preu:
 ↪  1.05 } })

// Delete
db.producte.deleteOne({ nom: "Monitor" })
db.producte.deleteMany({ estoc: 0 })
```

Operadors habituals: `$eq`, `$gt`, `$gte`, `$lt`, `$lte`, `$in`, `$nin`, `$and`, `$or`, `$not`, `$exists`, `$regex`.

7.6. Agregació i índexs

```
db.producte.aggregate([
  { $match: { estoc: { $gt: 0 } } },
  { $group: { _id: "$categoria", mitjana: { $avg: "$preu" }, total: {
   ↪  $sum: 1 } } },
  { $sort: { mitjana: -1 } }
])

db.producte.createIndex({ nom: 1 })
db.producte.createIndex({ categoria: 1, preu: -1 })
db.producte.find({ nom: "Teclat" }).explain("executionStats")
```

7.7. Modelatge: incrustar o referenciar?

- **Incrustar** (documents dins de documents): quan les dades es llegeixen sempre juntes i la relació és 1:pocs. Lectures ràpides, però duplicació.
- **Referenciar** (guardar l'`_id` d'un altre document): quan la relació és 1:molts o N:M, o les dades canvien sovint.

```
// Comanda amb línies incrustades
{
  _id: ObjectId("..."),
  client: { id: 1, nom: "Anna Puig" },
  data: ISODate("2026-11-05T10:00:00Z"),
  linies: [
    { producte: "Ratolí òptic", quantitat: 2, preu: 12.5 },
    { producte: "Teclat",       quantitat: 1, preu: 25 }
  ]
}
```

7.8. Eines d'administració

`mongosh`, **MongoDB Compass** (interfície gràfica), `mongodump` / `mongorestore` (còpies), `mongoimport` / `mongoexport` (importació/exportació JSON/CSV).

8. Recursos recomanats

- Documentació oficial de **MariaDB**: https://mariadb.com/kb/en/documentation/
- Documentació oficial de **MongoDB**: https://www.mongodb.com/docs/
- **SQLBolt** i **SQLZoo** per practicar SQL en línia.
- **dbdiagram.io** i **draw.io** per als diagrames E/R.
- **AEPD** i **APDCAT** per a la normativa de protecció de dades.

Versions d'aquest document

- HTML
- PDF
- ODT
- MD

Domini Públic (CC0)