Generalitat de Catalunya
Departament d'Educació
Institut Thos i Codina

Llenguatge de Definició de Dades (DDL) amb MySQL

Domini Públic (CC0)

Índex

1. Què és el DDL **1**

2. Convencions de sintaxi i preparació de l'entorn **2**
2.1. Notació . . . 2
2.2. Normes per anomenar objectes . . . 2
2.3. Connexió al servidor . . . 2

3. Bases de dades: `CREATE`, `USE`, `DROP` **2**
3.1 Crear una base de dades . . . 3
3.2 Veure les bases de dades existents . . . 3
3.3 Seleccionar la base de dades de treball . . . 3
3.4 Esborrar una base de dades . . . 3

4. Tipus de dades **4**
4.1. Tipus alfanumèrics (*string*) . . . 4
4.2. Tipus numèrics . . . 5
4.3. Tipus per a moments temporals . . . 6
4.4. Altres tipus . . . 6

5. Crear taules: `CREATE TABLE` **7**
5.1. Sintaxi bàsica . . . 7
5.2. La primera taula . . . 7
5.3. Comprovar què hem creat . . . 8
5.4. Crear una taula a partir d'una altra . . . 8

6. Restriccions (*constraints*) **8**
6.1. `NOT NULL` . . . 9
6.2. `UNIQUE` . . . 9
6.3. `PRIMARY KEY` . . . 9
6.4. `FOREIGN KEY` . . . 10
6.5. `CHECK` . . . 10
6.6. `DEFAULT` . . . 11

7. Cas pràctic: la base de dades `institut` **11**
7.1. Taula `alumnes` . . . 12
7.2. Taula `moduls` . . . 12
7.3. Taula `matricules` . . . 13

8. Comprovem les restriccions **13**

9. Modificar taules: `ALTER TABLE` **15**
9.1. Afegir una columna: `ADD COLUMN` . . . 15
9.2. Modificar una columna: `MODIFY` i `CHANGE` . . . 15
9.3. Esborrar una columna: `DROP COLUMN` . . . 16
9.4. Afegir i esborrar restriccions . . . 16
9.5. Canviar el comptador d'`AUTO_INCREMENT` . . . 17
9.6. Canviar el nom d'una taula: `RENAME TABLE` . . . 17
9.7. Altres canvis habituals . . . 17

10. Índexs: `CREATE INDEX` 18

11. Vistes: `CREATE VIEW` 18

12. Esborrar dades i objectes: `TRUNCATE` i `DROP` 19
12.1. `TRUNCATE TABLE` . . . 19
12.2. `DROP TABLE` . . . 19
12.3. `DROP DATABASE` . . . 20

13. Errors habituals 20

14. Resum d'ordres DDL a MariaDB 21
Bases de dades . . . 21
Taules . . . 21
Restriccions . . . 22
Punts a tenir en compte . . . 22
Modificació de taules . . . 23
Índexs . . . 23
Vistes . . . 24
Buidar, eliminar i reanomenar . . . 24
Consulta de l'estructura . . . 24
Consulta i modificació de variables . . . 24

15. Exercici 25

16. Referències 25

Cicles formatius: Administració de Sistemes Informàtics en Xarxa (ASIX) / Desenvolupament d'aplicacions web (DAW)

Mòduls: 0372 -- Gestió de bases de dades / 0484 - Bases de dades

Aquest document explica, pas a pas, com **definir i modificar l'estructura d'una base de dades** amb el llenguatge SQL: crear bases de dades i taules, triar els tipus de dades, aplicar restriccions (claus primàries, foranes, etc.), modificar taules ja existents i esborrar objectes.

Està pensat per treballar amb **MySQL 8** (també funciona amb MariaDB 10.x; les diferències es marquen quan n'hi ha). Es recomana escriure i executar tots els exemples, sigui amb el client `mysql` o amb MySQL Workbench.

1. Què és el DDL

Les sentències que ofereix el llenguatge SQL s'acostumen a agrupar segons la tasca que realitzen:

Sigla	Nom	Per a què serveix	Sentències principals
DDL	*Data Definition Language* (LDD)	Definir els objectes de la base de dades: taules, camps, restriccions, índexs…	`CREATE`, `ALTER`, `DROP`, `TRUNCATE`, `RENAME`
DCL	*Data Control Language* (LCD)	Concedir i retirar permisos sobre els objectes	`GRANT`, `REVOKE`
QL	*Query Language* (LC)	Consultar les dades	`SELECT`
DML	*Data Manipulation Language* (LMD)	Actualitzar les dades (altes, baixes i modificacions)	`INSERT`, `UPDATE`, `DELETE`

NOTA

En alguns SGBD no es distingeix entre QL i DML, i només es parla de DML per a les consultes i les actualitzacions. És una simple classificació; el que importa és saber què fa cada sentència.

Aquest document se centra en el **DDL**: les sentències que *construeixen l'esquema*. Com a diferència clau amb el DML, recordeu que:

- El DDL treballa sobre l'**estructura** (la taula, la columna, la restricció).
- El DML treballa sobre el **contingut** (les files).
- A MySQL, cada sentència DDL fa un *commit* implícit: **no es pot desfer amb `ROLLBACK`**. Compte amb els `DROP`.

2. Convencions de sintaxi i preparació de l'entorn

2.1. Notació

A les descripcions de sintaxi s'utilitza la notació habitual:

Notació	Significat
`MAJÚSCULES`	Paraules clau de SQL (no cal escriure-les en majúscules, però és una bona pràctica)
`[ ... ]`	Element **opcional**
`a | b`	Elements **alternatius** (cal triar-ne un)
`nom_taula`, `columna`…	Noms que triem nosaltres

Totes les sentències SQL acaben amb punt i coma (`;`). Els comentaris es poden escriure amb `-- text` (cal un espai després dels dos guions), `# text` o `/* text */`.

2.2. Normes per anomenar objectes

- Utilitzeu noms **sense espais ni accents** (`data_naixement`, no `data de naixement`).
- Seguiu un criteri coherent: tot en minúscules i amb guió baix, per exemple.
- Si cal usar un nom reservat o amb caràcters especials, es pot encapsular amb accents greus: `` `ordre` ``. És millor evitar-ho.
- A Linux els noms de bases de dades i taules **distingeixen majúscules i minúscules** (perquè corresponen a directoris i fitxers); a Windows no. Per evitar problemes, useu sempre minúscules.

2.3. Connexió al servidor

Des de la línia d'ordres:

```
mysql -u root -p
```

Dins del client, ens demanarà la contrasenya i obtindrem un indicador `mysql>`. A MySQL Workbench n'hi ha prou amb crear una connexió i obrir una pestanya de consulta SQL.

3. Bases de dades: CREATE, USE, DROP

En MySQL, *base de dades* i *esquema* (`SCHEMA`) són sinònims.

3.1 Crear una base de dades

```
CREATE DATABASE [IF NOT EXISTS] nom_bd
    [CHARACTER SET joc_caracters]
    [COLLATE col·lacio];
```

Exemple: creem la base de dades que farem servir al llarg del document.

```
CREATE DATABASE IF NOT EXISTS institut
    CHARACTER SET utf8mb4
    COLLATE utf8mb4_unicode_ci;
```

- `IF NOT EXISTS` evita l'error si la base de dades ja existeix (només emet un avís).
- `CHARACTER SET utf8mb4` permet emmagatzemar qualsevol caràcter Unicode (accents, ç, ·, emojis…).
- `COLLATE` defineix com es comparen i s'ordenen els textos (per exemple, si a i à es consideren iguals o no).

3.2 Veure les bases de dades existents

```
SHOW DATABASES;
```

3.3 Seleccionar la base de dades de treball

Abans de crear taules, hem d'indicar a quina base de dades volem treballar:

```
USE institut;
```

Per saber quina base de dades tenim seleccionada:

```
SELECT DATABASE();
```

3.4 Esborrar una base de dades

```
DROP DATABASE [IF EXISTS] nom_bd;
```

AVÍS

`DROP DATABASE` elimina **totes les taules i totes les dades** de la base de dades, sense demanar confirmació i sense possibilitat de desfer-ho.

4. Tipus de dades

Cada columna d'una taula té un **tipus de dada** que defineix el domini de valors que pot contenir. Per exemple, una columna de tipus `DATE` no acceptarà el valor `'30 de febrer'`, ni el número `2`, ni la cadena `Hola`. Triar bé el tipus és fonamental: afecta la integritat de les dades, l'espai ocupat i el rendiment.

Cada SGBD té els seus propis tipus; aquí veiem els de MySQL, agrupats en quatre famílies: alfanumèrics, numèrics, temporals i altres.

4.1. Tipus alfanumèrics (*string*)

Tipus	Descripció	Longitud
`CHAR(n)`	Cadena de **longitud fixa**. Si el valor és més curt, s'omple amb espais fins a `n`.	0--255 caràcters
`VARCHAR(n)`	Cadena de **longitud variable**. Guarda exactament el que s'hi escriu. Cal indicar-ne sempre la llargada màxima.	fins a 65.535 caràcters (limitat per la mida de fila)
`BINARY(n)`	Com `CHAR`, però emmagatzema bytes.	0--255 bytes
`VARBINARY(n)`	Com `VARCHAR`, però emmagatzema bytes.	fins a 65.535 bytes
`TINYTEXT`, `TEXT`, `MEDIUMTEXT`, `LONGTEXT`	Textos llargs, guardats fora de la fila.	2^8, 2^{16}, 2^{24} i 2^{32} bytes (aprox.)
`TINYBLOB`, `BLOB`, `MEDIUMBLOB`, `LONGBLOB`	Dades binàries grans (imatges, fitxers…).	2^8, 2^{16}, 2^{24} i 2^{32} bytes (aprox.)
`ENUM('v1','v2',...)`	Un únic valor d'entre una llista tancada.	fins a 65.535 valors
`SET('v1','v2',...)`	Zero o més valors d'entre una llista tancada.	fins a 64 valors

`CHAR` contra `VARCHAR`. Comparem com es guarda cada valor en una columna de 4 caràcters:

Valor	`CHAR(4)`	`VARCHAR(4)`
`'ab'`	`'ab  '` (amb dos espais)	`'ab'`
`'abcd'`	`'abcd'`	`'abcd'`
`'abcdefgh'`	error (o es trunca, segons el mode SQL)	error (o es trunca, segons el mode SQL)

NOTA

El tipus alfanumèric més habitual per guardar *strings* és `VARCHAR`. Reserveu `CHAR` per a dades de longitud realment fixa: codis de cicle (`'SMX'`), DNI (`'12345678A'`), codis postals, etc.

Exemples de `ENUM` i `SET`:

```
sexe     ENUM('H','D','altre')
idiomes  SET('català','castellà','anglès','francès')
```

Una columna `SET` pot contenir, per exemple, `''`, `'català'` o `'català,anglès'`. L'ordre en què s'escriuen els elements no afecta el resultat.

4.2. Tipus numèrics

Enters

Tipus	Bytes	Mínim (amb signe)	Màxim (amb signe)	Màxim (sense signe)
`TINYINT`	1	-128	127	255
`SMALLINT`	2	-32.768	32.767	65.535
`MEDIUMIN`	3	-8.388.608	8.388.607	16.777.215
`INT` (o `INTEGER`)	4	-2.147.483.648	2.147.483.647	4.294.967.295
`BIGINT`	8	-9.223.372.036.854.775.808	9.223.372.036.854.775.807	18.446.744.073.709.551.615

NOTA

Per emmagatzemar enters, el més habitual és utilitzar `INT`. Si sabem que el rang és petit, un tipus més petit estalvia espai (per exemple, `TINYINT UNSIGNED` per a una edat).

Decimals de coma fixa: `DECIMAL(T,D)` / `NUMERIC(T,D)`

`DECIMAL` i `NUMERIC` són sinònims. `T` és el total de dígits i `D` el nombre de decimals. Per exemple, `DECIMAL(5,2)` admet valors des de -999,99 fins a 999,99. Si no s'indica res, `T` val 10 per defecte (màxim 65).

Els valors de coma fixa **mai s'arrodoneixen en silenci**: o són exactes o el SGBD dona error. Per això són els adequats per a **imports de diners**.

Decimals de coma flotant: `FLOAT`, `DOUBLE` (o `REAL`)

Emmagatzemen valors aproximats (4 bytes `FLOAT`, 8 bytes `DOUBLE`). Són útils per a magnituds científiques, però **no s'han d'usar per a diners** perquè introdueixen petits errors d'aproximació.

Altres tipus numèrics

- `BIT(M)`: bits, des d'1 (per defecte) fins a 64. Els literals s'escriuen `b'0101'`.
- `BOOL` o `BOOLEAN`: sinònim de `TINYINT(1)`. El valor 0 és fals; qualsevol altre valor és cert.

Modificadors numèrics

Es poden afegir a la definició d'una columna numèrica:

Modificador	Efecte
`UNSIGNED`	Només s'admeten valors no negatius (i el rang positiu es duplica).
`ZEROFILL`	S'afegeixen zeros a l'esquerra fins a completar l'amplada de visualització.
`AUTO_INCREMENT`	Quan s'insereix `NULL` (o 0, o no s'indica valor), s'hi guarda el valor més alt existent incrementat en 1. El primer valor és 1.

NOTA

La sintaxi `INT(4)` indica només l'**amplada de visualització** (amb `ZEROFILL`); no limita el valor que es pot guardar. A MySQL 8.0.17 i posteriors aquesta amplada està en desús (*deprecated*), igual que `FLOAT(E,D)`. Per això en aquest document escrivim simplement `INT`.

4.3. Tipus per a moments temporals

Tipus	Format	Rang
`DATE`	`'AAAA-MM-DD'`	de `'1000-01-01'` a `'9999-12-31'`
`DATETIME`	`'AAAA-MM-DD HH:MM:SS'`	de `'1000-01-01 00:00:00'` a `'9999-12-31 23:59:59'`
`TIMESTAMP`	`'AAAA-MM-DD HH:MM:SS'`	de `'1970-01-01 00:00:01'` a `'2038-01-19 03:14:07'` (UTC)
`TIME`	`'HH:MM:SS'`	de `'-838:59:59'` a `'838:59:59'` (també serveix per a diferències de temps)
`YEAR`	`AAAA`	de 1901 a 2155 (i 0000)

Punts a tenir en compte:

- `DATETIME` i `TIMESTAMP` tenen el mateix format, però `TIMESTAMP` és el tipus pensat per guardar **el moment en què s'ha afegit o modificat una fila**, gràcies a les clàusules `DEFAULT CURRENT_TIMESTAMP` i `ON UPDATE CURRENT_TIMESTAMP`.
- Escriviu sempre els anys amb **quatre dígits**. Si se n'utilitzen dos, MySQL interpreta 00--69 com a 2000--2069 i 70--99 com a 1970--1999.

```
-- Es completa en el moment de la inserció
data_alta        TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP

-- A més, s'actualitza cada vegada que es modifica la fila
darrera_mod      TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP
                 ON UPDATE CURRENT_TIMESTAMP
```

4.4. Altres tipus

MySQL també té tipus per a dades espacials (`POINT`, `LINESTRING`, `POLYGON`, `GEOMETRY`…), seguint el model geomètric de l'estàndard OpenGIS, i el tipus `JSON` per guardar documents JSON. Queden fora de l'abast d'aquest document.

5. Crear taules: CREATE TABLE

5.1. Sintaxi bàsica

```
CREATE TABLE [IF NOT EXISTS] nom_taula (
    columna1  tipus  [restriccions_de_columna],
    columna2  tipus  [restriccions_de_columna],
    ...
    [restriccions_de_taula]
) [ENGINE = InnoDB];
```

Cada columna s'ha de definir amb un **nom** i un **tipus**. Opcionalment es poden afegir restriccions. Al final de la llista es poden afegir les restriccions de taula (les que afecten més d'una columna, o les que volem anomenar explícitament).

5.2. La primera taula

Comencem per una taula senzilla amb els cicles formatius de l'institut:

```
USE institut;

CREATE TABLE cicles (
    codi          CHAR(4)           NOT NULL,
    nom           VARCHAR(100)      NOT NULL,
    durada_hores  SMALLINT UNSIGNED NOT NULL DEFAULT 2000,
    PRIMARY KEY (codi)
) ENGINE = InnoDB;
```

Anem per parts:

- `codi CHAR(4) NOT NULL`: codi de longitud fixa (`SMX`, `ASIX`). No pot ser nul.
- `nom VARCHAR(100) NOT NULL`: text de fins a 100 caràcters, obligatori.
- `durada_hores SMALLINT UNSIGNED NOT NULL DEFAULT 2000`: enter positiu de fins a 65.535; si no s'indica cap valor en inserir, en valdrà 2000.
- `PRIMARY KEY (codi)`: identifica cada fila de manera única (ho veurem a la secció 6).
- `ENGINE = InnoDB`: motor d'emmagatzematge. És el motor per defecte a MySQL 8 i el que admet **transaccions i claus foranes**.

5.3. Comprovar què hem creat

```
SHOW TABLES;

DESCRIBE cicles;          -- també: DESC cicles;  o  SHOW COLUMNS FROM
 ↪  cicles;

SHOW CREATE TABLE cicles; -- mostra la sentència completa que recrea
 ↪  la taula
```

La sortida de `DESCRIBE` mostra, per a cada columna, el tipus (`Type`), si admet nuls (`Null`), si forma part d'una clau (`Key`), el valor per defecte (`Default`) i informació addicional (`Extra`, com `auto_increment`).

5.4. Crear una taula a partir d'una altra

```
-- Còpia només de l'estructura (sense dades)
CREATE TABLE cicles_copia LIKE cicles;

-- Còpia de l'estructura i de les dades d'una consulta
CREATE TABLE cicles_resum AS
    SELECT codi, nom FROM cicles;
```

`CREATE TABLE ... LIKE` copia columnes, índexs i claus primàries/úniques. `CREATE TABLE ... AS SELECT` només copia columnes i dades, **però no les restriccions ni els índexs**. Esborrem aquestes taules de prova:

```
DROP TABLE cicles_copia, cicles_resum;
```

6. Restriccions (*constraints*)

Les restriccions són regles que el SGBD fa complir automàticament per garantir la **integritat** de les dades. Si una operació les incompleix, el SGBD la rebutja amb un error.

Restricció	Què garanteix
`NOT NULL`	La columna no pot tenir valors nuls.
`UNIQUE`	No hi ha dos valors repetits a la columna (o combinació de columnes).
`PRIMARY KEY`	Identifica cada fila de manera única: és `NOT NULL` + `UNIQUE`. Només n'hi ha una per taula.
`FOREIGN KEY`	El valor ha d'existir a la clau d'una altra taula (integritat referencial).
`CHECK`	El valor ha de complir una condició.
`DEFAULT`	Valor que s'assigna si no se n'indica cap.

Es poden definir de dues maneres:

- **A escala de columna**, just a continuació del tipus: `dni CHAR(9) NOT NULL UNIQUE`.

- **A escala de taula**, al final de la llista de columnes: obligatori quan afecta més d'una columna, i recomanat quan volem posar-hi **nom** amb `CONSTRAINT nom ...`. Així els missatges d'error són més clars i després es pot modificar o esborrar la restricció pel seu nom.

6.1. `NOT NULL`

Indica que la columna sempre ha de tenir valor.

```
nom  VARCHAR(50) NOT NULL
```

NOTA

`NULL` no és el mateix que zero ni que una cadena buida: significa *valor desconegut o inexistent*. Una columna que accepta nuls pot guardar `NULL`; per això `NOT NULL` és una de les restriccions més importants.

6.2. `UNIQUE`

Evita valors duplicats. A MySQL, una columna `UNIQUE` pot contenir diversos `NULL`.

```
-- A escala de columna
dni CHAR(9) NOT NULL UNIQUE

-- A escala de taula (amb nom, una o més columnes)
CONSTRAINT uq_alumnes_dni UNIQUE (dni)
CONSTRAINT uq_moduls_cicle_codi UNIQUE (cicle, codi)
```

Amb una restricció `UNIQUE` sobre dues columnes, el que no es pot repetir és **la combinació** (per exemple, el mateix codi de mòdul dins del mateix cicle).

6.3. `PRIMARY KEY`

La clau primària identifica cada fila. Ha de ser única i no nul·la, i només n'hi pot haver una per taula (però pot ser **composta** per diverses columnes).

```
-- Clau primària simple
id_alumne INT UNSIGNED NOT NULL AUTO_INCREMENT,
PRIMARY KEY (id_alumne)

-- Clau primària composta
PRIMARY KEY (id_alumne, id_modul, curs)
```

Se sol combinar amb `AUTO_INCREMENT` perquè el SGBD generi els identificadors automàticament.

CONSELL

`AUTO_INCREMENT` només es pot posar en una columna que sigui clau (o que tingui un índex), i només hi pot haver una columna `AUTO_INCREMENT` per taula.

6.4. `FOREIGN KEY`

La clau forana crea una **relació** entre dues taules: els valors de la columna de la taula *filla* han d'existir a la clau (primària o única) de la taula *mare*.

```
CONSTRAINT nom_fk
    FOREIGN KEY (columna_filla)
    REFERENCES taula_mare (columna_mare)
    [ON DELETE accio]
    [ON UPDATE accio]
```

Les **accions referencials** defineixen què passa quan s'esborra o es modifica una fila de la taula mare que té files filles:

Acció	Efecte
`RESTRICT` / `NO ACTION`	Impedeix l'operació (és el comportament per defecte).
`CASCADE`	Propaga l'operació a les files filles (les esborra o actualitza).
`SET NULL`	Posa a `NULL` la clau forana de les files filles (la columna ha d'admetre nuls).
`SET DEFAULT`	No és acceptada per InnoDB.

Requisits perquè una clau forana es pugui crear:

1. Les dues taules han d'usar el motor **InnoDB**.
2. Les columnes relacionades han de tenir **exactament el mateix tipus** (mateixa mida, mateix `UNSIGNED`, mateix joc de caràcters).
3. La columna referenciada ha de ser clau primària o tenir un índex únic.
4. La taula mare ha d'existir abans de crear la filla.

6.5. `CHECK`

Obliga que el valor compleixi una condició. A MySQL només es fa complir a partir de la versió **8.0.16** (en versions anteriors s'acceptava la sintaxi, però s'ignorava). MariaDB la fa complir des de la 10.2.1.

```
CONSTRAINT ck_matricules_nota CHECK (nota BETWEEN 0 AND 10)
CONSTRAINT ck_alumnes_email   CHECK (email IS NULL OR email LIKE
 ↪  '%_@_%')
```

6.6. DEFAULT

Defineix el valor que s'assigna quan en inserir una fila no s'indica res per a la columna.

```
durada_hores  SMALLINT UNSIGNED NOT NULL DEFAULT 2000
data_alta     TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP
estat         ENUM('actiu','baixa') NOT NULL DEFAULT 'actiu'
```

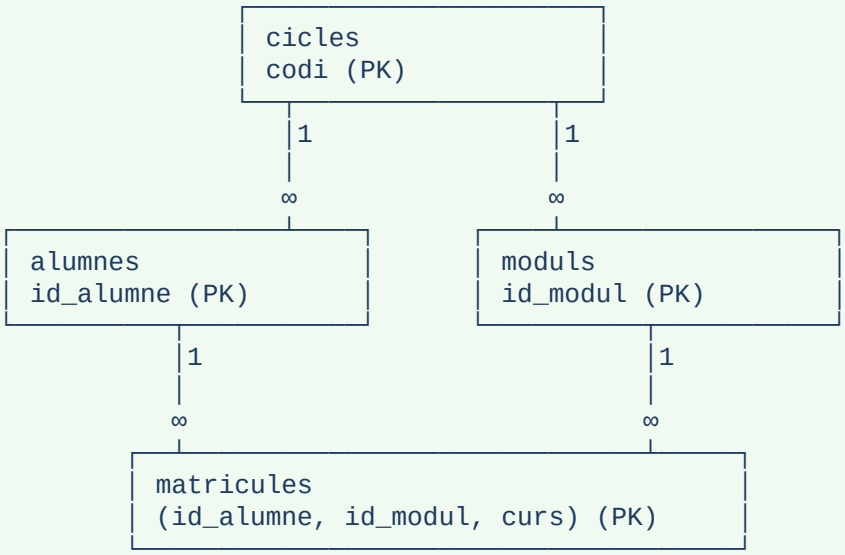
NOTA

A partir de MySQL 8.0.13 el valor per defecte també pot ser una expressió, escrita entre parèntesis: `DEFAULT (CURRENT_DATE)`.

7. Cas pràctic: la base de dades `institut`

Ja tenim la taula `cicles`. Completem l'esquema amb les taules `alumnes`, `moduls` i `matricules`:

- Un **cicle** té molts **alumnes** i molts **mòduls**.
- Un **alumne** es matricula de molts **mòduls** i un mòdul té molts alumnes: relació *molts a molts*, que es resol amb la taula intermèdia `matricules`.



7.1. Taula `alumnes`

```
CREATE TABLE alumnes (
    id_alumne       INT UNSIGNED  NOT NULL AUTO_INCREMENT,
    dni             CHAR(9)       NOT NULL,
    nom             VARCHAR(50)   NOT NULL,
    cognoms         VARCHAR(100)  NOT NULL,
    data_naixement  DATE,
    email           VARCHAR(120),
    cicle           CHAR(4)       NOT NULL,
    data_alta       TIMESTAMP     NOT NULL DEFAULT CURRENT_TIMESTAMP,
    PRIMARY KEY (id_alumne),
    CONSTRAINT uq_alumnes_dni   UNIQUE (dni),
    CONSTRAINT ck_alumnes_email CHECK (email IS NULL OR email LIKE
     ↪  '%_@_%'),
    CONSTRAINT fk_alumnes_cicles
        FOREIGN KEY (cicle) REFERENCES cicles (codi)
        ON UPDATE CASCADE
        ON DELETE RESTRICT
) ENGINE = InnoDB;
```

Decisions de disseny:

- `id_alumne` és una **clau substituta** autogenerada; el `dni` es manté com a `UNIQUE` perquè també ha de ser irrepetible.
- `data_naixement` i `email` admeten nuls (no són obligatoris).
- Si es canvia el codi d'un cicle (`ON UPDATE CASCADE`), s'actualitza automàticament als alumnes; però **no es pot esborrar** un cicle que tingui alumnes (`ON DELETE RESTRICT`).

7.2. Taula `moduls`

```
CREATE TABLE moduls (
    id_modul  INT UNSIGNED      NOT NULL AUTO_INCREMENT,
    cicle     CHAR(4)           NOT NULL,
    codi      CHAR(6)           NOT NULL,
    nom       VARCHAR(100)      NOT NULL,
    hores     SMALLINT UNSIGNED NOT NULL DEFAULT 99,
    PRIMARY KEY (id_modul),
    CONSTRAINT uq_moduls_cicle_codi UNIQUE (cicle, codi),
    CONSTRAINT fk_moduls_cicles
        FOREIGN KEY (cicle) REFERENCES cicles (codi)
        ON UPDATE CASCADE
        ON DELETE CASCADE
) ENGINE = InnoDB;
```

Aquí `ON DELETE CASCADE` fa que, si s'esborra un cicle, s'esborrin també els seus mòduls.

7.3. Taula `matricules`

```
CREATE TABLE matricules (
    id_alumne  INT UNSIGNED  NOT NULL,
    id_modul   INT UNSIGNED  NOT NULL,
    curs       YEAR          NOT NULL,
    nota       DECIMAL(4,2),
    PRIMARY KEY (id_alumne, id_modul, curs),
    CONSTRAINT ck_matricules_nota CHECK (nota BETWEEN 0 AND 10),
    CONSTRAINT fk_matricules_alumnes
        FOREIGN KEY (id_alumne) REFERENCES alumnes (id_alumne)
        ON DELETE CASCADE,
    CONSTRAINT fk_matricules_moduls
        FOREIGN KEY (id_modul) REFERENCES moduls (id_modul)
        ON DELETE RESTRICT
) ENGINE = InnoDB;
```

- La **clau primària composta** (`id_alumne, id_modul, curs`) permet que un alumne repeteixi un mòdul en un altre curs, però no matricular-se'n dues vegades en el mateix.
- `nota DECIMAL(4,2)` admet valors com `7.50` o `10.00`. Pot ser `NULL` mentre no s'ha avaluat.

Verifiquem l'esquema:

```
SHOW TABLES;
DESCRIBE alumnes;
SHOW CREATE TABLE matricules;
```

8. Comprovem les restriccions

Per veure que les restriccions funcionen, inserim algunes dades (els `INSERT` són DML, però ens serveixen per provar el DDL que hem escrit).

```
INSERT INTO cicles (codi, nom) VALUES
    ('SMX',  'Sistemes Microinformàtics i Xarxes'),
    ('ASIX', 'Administració de Sistemes Informàtics en Xarxa');

-- Observeu que durada_hores ha agafat el valor per defecte
SELECT * FROM cicles;

INSERT INTO alumnes (dni, nom, cognoms, data_naixement, email, cicle)
 ↪  VALUES
    ('12345678A', 'Marta', 'Soler Puig', '2007-03-14',
     ↪  'marta@exemple.cat', 'SMX');

-- data_alta s'ha omplert sola gràcies al DEFAULT CURRENT_TIMESTAMP
SELECT id_alumne, nom, cicle, data_alta FROM alumnes;
```

```
INSERT INTO moduls (cicle, codi, nom, hores) VALUES
    ('SMX', 'M02', 'Sistemes operatius monolloc', 165);

INSERT INTO matricules (id_alumne, id_modul, curs, nota) VALUES (1, 1,
 ↪  2026, 7.50);
```

Ara provem d'incomplir cada restricció. **Tots aquests `INSERT` han de fallar:**

```
-- PRIMARY KEY duplicada (error 1062)
INSERT INTO cicles (codi, nom) VALUES ('SMX', 'Un altre nom');

-- UNIQUE: DNI repetit (error 1062)
INSERT INTO alumnes (dni, nom, cognoms, cicle)
    VALUES ('12345678A', 'Pau', 'Vila Roca', 'SMX');

-- NOT NULL: falta el cognom (error 1364: el camp no té valor per
 ↪  defecte)
INSERT INTO alumnes (dni, nom, cicle)
    VALUES ('87654321B', 'Pau', 'SMX');

-- FOREIGN KEY: el cicle 'DAW' no existeix (error 1452)
INSERT INTO alumnes (dni, nom, cognoms, cicle)
    VALUES ('87654321B', 'Pau', 'Vila Roca', 'DAW');

-- CHECK: l'email no té arrova (error 3819 a MySQL; 4025 a MariaDB)
INSERT INTO alumnes (dni, nom, cognoms, email, cicle)
    VALUES ('87654321B', 'Pau', 'Vila Roca', 'pau.exemple.cat',
     ↪  'SMX');

-- CHECK: nota fora de rang (error 3819)
INSERT INTO matricules (id_alumne, id_modul, curs, nota) VALUES (1, 1,
 ↪  2027, 11);

-- ON DELETE RESTRICT: no es pot esborrar un cicle amb alumnes (error
 ↪  1451)
DELETE FROM cicles WHERE codi = 'SMX';
```

Finalment, comprovem el comportament de les accions referencials:

```
-- ON UPDATE CASCADE: el canvi del codi del cicle es propaga als
 ↪  alumnes i als mòduls
UPDATE cicles SET codi = 'SMIX' WHERE codi = 'SMX';
SELECT id_alumne, nom, cicle FROM alumnes;
SELECT id_modul, cicle, codi FROM moduls;

-- Ho tornem a deixar com estava
UPDATE cicles SET codi = 'SMX' WHERE codi = 'SMIX';
```

NOTA

Els errors es poden provocar des del client interactiu. Si executeu un script amb `mysql < fitxer.sql`, l'execució s'atura en el primer error (afegiu `--force` per continuar).

9. Modificar taules: `ALTER TABLE`

Un cop creada una taula, el seu disseny pot canviar: apareixen nous requisits, es detecten errors… `ALTER TABLE` permet **modificar l'estructura sense haver de recrear la taula**.

```
ALTER TABLE nom_taula
    accio [, accio ...];
```

Es poden encadenar diverses accions separades per comes en una sola sentència.

9.1. Afegir una columna: `ADD COLUMN`

```
-- Al final de la taula
ALTER TABLE alumnes ADD COLUMN telefon VARCHAR(15);

-- En una posició concreta
ALTER TABLE alumnes ADD COLUMN poblacio VARCHAR(60) AFTER email;

-- Diverses columnes alhora
ALTER TABLE alumnes
    ADD COLUMN observacions TEXT,
    ADD COLUMN actiu BOOLEAN NOT NULL DEFAULT TRUE;
```

Si afegim una columna `NOT NULL` a una taula que ja té files, les files existents rebran el valor per defecte (o el valor implícit del tipus, si no n'hi ha cap).

9.2. Modificar una columna: `MODIFY` i `CHANGE`

- `MODIFY` canvia **el tipus i les propietats** d'una columna (però no el nom).
- `CHANGE` canvia **el nom** i, a la vegada, la definició (cal tornar a escriure la definició completa).

```
-- Ampliem la longitud del telèfon i el fem obligatori amb un valor
 ↪  per defecte.
-- Abans, cal substituir els nuls que ja hi hagi: si no, MODIFY
 ↪  donaria error.
UPDATE alumnes SET telefon = '' WHERE telefon IS NULL;
ALTER TABLE alumnes MODIFY COLUMN telefon VARCHAR(20) NOT NULL DEFAULT
 ↪  '';

-- Canviem el nom de la columna 'poblacio' per 'municipi'
ALTER TABLE alumnes CHANGE COLUMN poblacio municipi VARCHAR(80);

-- Només canviar el valor per defecte
ALTER TABLE alumnes ALTER COLUMN actiu SET DEFAULT FALSE;
ALTER TABLE alumnes ALTER COLUMN actiu DROP DEFAULT;
```

AVÍS

Convertir en `NOT NULL` una columna que ja conté `NULL` falla (error 1265 o 1138, segons la versió). Cal omplir abans aquests valors amb un `UPDATE`, com a l'exemple anterior.

AVÍS

Quan es redueix la mida d'una columna (per exemple, de `VARCHAR(100)` a `VARCHAR(10)`) o es canvia el tipus, les dades que no hi càpiguen poden perdre's o provocar un error. Feu sempre una **còpia de seguretat** abans.

CONSELL

MySQL 8 també permet `ALTER TABLE ... RENAME COLUMN antic TO nou`, que només canvia el nom, sense repetir el tipus.

```
ALTER TABLE alumnes RENAME COLUMN municipi TO poblacio;
```

9.3. Esborrar una columna: `DROP COLUMN`

```
ALTER TABLE alumnes DROP COLUMN observacions;
ALTER TABLE alumnes DROP COLUMN actiu, DROP COLUMN poblacio, DROP
 ↪  COLUMN telefon;
```

AVÍS

Esborrar una columna elimina també totes les dades que contenia, de manera irreversible. No es pot esborrar una columna que formi part d'una clau forana mentre la restricció existeixi.

9.4. Afegir i esborrar restriccions

```
-- UNIQUE
ALTER TABLE alumnes ADD CONSTRAINT uq_alumnes_email UNIQUE (email);
ALTER TABLE alumnes DROP INDEX uq_alumnes_email;

-- CHECK
ALTER TABLE matricules
    ADD CONSTRAINT ck_matricules_curs CHECK (curs >= 2000);
ALTER TABLE matricules DROP CONSTRAINT ck_matricules_curs;

-- FOREIGN KEY: l'esborrem i la tornem a crear amb una altra acció
ALTER TABLE matricules DROP FOREIGN KEY fk_matricules_moduls;
ALTER TABLE matricules
    ADD CONSTRAINT fk_matricules_moduls
        FOREIGN KEY (id_modul) REFERENCES moduls (id_modul)
        ON DELETE CASCADE;
```

```
-- PRIMARY KEY (només si la taula no en té)
-- ALTER TABLE nom_taula ADD PRIMARY KEY (columna);
-- ALTER TABLE nom_taula DROP PRIMARY KEY;
```

Fixeu-vos que el nom de la restricció és imprescindible per poder-la esborrar. Si no li hem posat nom, MySQL n'hi assigna un (`alumnes_chk_1`, `alumnes_ibfk_1`...) que podem consultar amb `SHOW CREATE TABLE`.

NOTA

`DROP CONSTRAINT nom` per a `CHECK` i `UNIQUE` funciona a MySQL des de la 8.0.19 i a MariaDB. A versions anteriors de MySQL s'utilitzava `DROP CHECK nom`.

9.5. Canviar el comptador d'`AUTO_INCREMENT`

```
ALTER TABLE alumnes AUTO_INCREMENT = 1000;
```

El següent alumne que s'insereixi tindrà l'`id_alumne` 1000 (o el primer valor lliure més gran).

9.6. Canviar el nom d'una taula: `RENAME TABLE`

```
RENAME TABLE moduls TO moduls_formatius;
RENAME TABLE moduls_formatius TO moduls;

-- Equivalent
ALTER TABLE moduls RENAME TO moduls_formatius;
ALTER TABLE moduls_formatius RENAME TO moduls;
```

Les claus foranes que apunten a la taula s'actualitzen automàticament.

9.7. Altres canvis habituals

```
-- Canviar el motor d'emmagatzematge
ALTER TABLE cicles ENGINE = InnoDB;

-- Canviar el joc de caràcters de la taula
ALTER TABLE cicles CONVERT TO CHARACTER SET utf8mb4 COLLATE
 ↪  utf8mb4_unicode_ci;
```

10. Índexs: `CREATE INDEX`

Un **índex** és una estructura que accelera les cerques i les ordenacions sobre una columna. Funciona com l'índex d'un llibre: en comptes de llegir tota la taula, el SGBD va directament a les files que interessen.

MySQL crea automàticament un índex per a cada `PRIMARY KEY`, `UNIQUE` i `FOREIGN KEY`. Per a altres columnes que consultem sovint amb `WHERE`, `ORDER BY` o `JOIN`, els podem crear a mà:

```
CREATE [UNIQUE] INDEX nom_index ON nom_taula (columna1 [, columna2
 ↪  ...]);
```

```
-- Índex sobre una columna
CREATE INDEX idx_alumnes_cognoms ON alumnes (cognoms);

-- Índex compost (útil per a cerques per cognoms i nom)
CREATE INDEX idx_alumnes_cognoms_nom ON alumnes (cognoms, nom);

-- Veure els índexs de la taula
SHOW INDEX FROM alumnes;

-- Esborrar un índex
DROP INDEX idx_alumnes_cognoms ON alumnes;
DROP INDEX idx_alumnes_cognoms_nom ON alumnes;
```

NOTA

Els índexs no són gratis: ocupen espai i alenteixen les insercions, modificacions i esborrats, perquè s'han d'actualitzar. Creeu-ne només per a les columnes que realment es consulten sovint.

11. Vistes: `CREATE VIEW`

Una **vista** és una consulta desada amb un nom, que es pot utilitzar com si fos una taula. No emmagatzema dades pròpies: cada vegada que s'hi accedeix, s'executa la consulta.

```
CREATE VIEW v_alumnes_smx AS
    SELECT id_alumne, nom, cognoms, email
    FROM alumnes
    WHERE cicle = 'SMX';

SELECT * FROM v_alumnes_smx;

-- Modificar la definició
CREATE OR REPLACE VIEW v_alumnes_smx AS
    SELECT id_alumne, nom, cognoms
    FROM alumnes
    WHERE cicle = 'SMX';
```

```
-- Esborrar la vista
DROP VIEW v_alumnes_smx;
```

Les vistes serveixen per **simplificar consultes complexes** i per **restringir l'accés** a determinades columnes o files.

12. Esborrar dades i objectes: `TRUNCATE` i `DROP`

12.1. `TRUNCATE TABLE`

Buida una taula **conservant-ne l'estructura**. És molt més ràpid que `DELETE FROM taula` i, a més, reinicia el comptador `AUTO_INCREMENT`.

```
TRUNCATE TABLE nom_taula;
```

Diferències amb `DELETE`:

	`DELETE FROM t`	`TRUNCATE TABLE t`
Tipus de sentència	DML	DDL
Es pot desfer amb `ROLLBACK`	Sí (dins d'una transacció)	No
Admet `WHERE`	Sí	No
Reinicia `AUTO_INCREMENT`	No	Sí
Taules referenciades per claus foranes	Permès (si no vulnera les restriccions)	**Error** mentre hi hagi claus foranes que hi apuntin

12.2. `DROP TABLE`

Elimina la taula completa: estructura, dades, índexs i restriccions.

```
DROP TABLE [IF EXISTS] taula1 [, taula2 ...];
```

Si altres taules tenen claus foranes que apunten a la que volem esborrar, **cal esborrar primer les taules filles** (o eliminar abans les restriccions). L'ordre invers al de creació sempre funciona:

```
DROP TABLE matricules;
DROP TABLE moduls;
DROP TABLE alumnes;
DROP TABLE cicles;
```

També es pot fer en una sola sentència; MySQL ordena les dependències per nosaltres si les taules s'esborren juntes:

```
DROP TABLE IF EXISTS matricules, moduls, alumnes, cicles;
```

AVÍS

Existeix la variable `SET FOREIGN_KEY_CHECKS = 0;` que desactiva la comprovació de claus foranes i permet esborrar taules en qualsevol ordre. Usa-la només en situacions controlades i torneu-la a activar amb `SET FOREIGN_KEY_CHECKS = 1;`. Si us oblideu de fer-ho, la base de dades pot quedar inconsistent.

12.3. DROP DATABASE

Per acabar la pràctica, esborrem la base de dades de treball:

```
DROP DATABASE IF EXISTS institut;
```

13. Errors habituals

Error	Missatge típic	Causa i solució
1005 / 3780	`Can't create table ... (errno: 150) "Foreign key constraint is incorrectly formed"` / `Referencing column and referenced column are incompatible`	Les columnes de la clau forana i de la clau referenciada no tenen exactament el mateix tipus (per exemple, `INT` contra `INT UNSIGNED`), o la taula mare no existeix o no és InnoDB.
1050	`Table 'x' already exists`	La taula ja existeix. Useu `IF NOT EXISTS` o esborreu-la abans.
1051	`Unknown table 'x'`	Esteu esborrant una taula que no existeix. Useu `DROP TABLE IF EXISTS`.
1046	`No database selected`	Cal executar `USE nom_bd;` abans d'operar amb taules.
1062	`Duplicate entry 'x' for key ...`	Es vulnera una `PRIMARY KEY` o `UNIQUE`.
1064	`You have an error in your SQL syntax`	Error de sintaxi: reviseu comes, parèntesis i paraules clau a prop del punt indicat.
1068	`Multiple primary key defined`	Heu definit més d'una clau primària a la mateixa taula.
1091	`Can't DROP 'x'; check that column/key exists`	La columna, índex o restricció que voleu esborrar no existeix (comproveu el nom amb `SHOW CREATE TABLE`).
1451	`Cannot delete or update a parent row: a foreign key constraint fails`	Esteu esborrant o modificant una fila mare que té files filles.
1452	`Cannot add or update a child row: a foreign key constraint fails`	La clau forana apunta a un valor que no existeix a la taula mare.
3819 (MySQL) / 4025 (MariaDB)	`Check constraint 'x' is violated` / `CONSTRAINT x failed for ...`	El valor incompleix una restricció `CHECK`.

14. Resum d'ordres DDL a MariaDB

En MariaDB, **cada ordre DDL fa un *commit* implícit** i no es pot desfer amb `ROLLBACK`. Les restriccions amb claus foranes només funcionen amb motors que les suporten (**InnoDB**, el per defecte).

Bases de dades

Una base de dades és un conjunt organitzat de dades relacionades entre si, emmagatzemades de manera estructurada perquè es puguin consultar, modificar i gestionar de manera eficient, segura i consistent.

```
CREATE DATABASE IF NOT EXISTS institut
  CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci;
ALTER DATABASE institut CHARACTER SET utf8mb4 COLLATE
 ↪  utf8mb4_unicode_ci;
DROP DATABASE IF EXISTS institut;
USE institut;
```

`SCHEMA` és sinònim de `DATABASE` a MariaDB.

Taules

Una taula és l'objecte bàsic on una base de dades relacional emmagatzema les dades, organitzades en files i columnes, com un full de càlcul amb regles estrictes.

```
CREATE TABLE IF NOT EXISTS grup (
    id    INT UNSIGNED AUTO_INCREMENT,
    nom   VARCHAR(20) NOT NULL,
    cicle ENUM('SMX','ASIX') NOT NULL DEFAULT 'SMX',
    CONSTRAINT pk_grup PRIMARY KEY (id),
    CONSTRAINT uq_grup_nom UNIQUE (nom)
) ENGINE=InnoDB;

CREATE TABLE alumne (
    id        INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
    nom       VARCHAR(50)  NOT NULL,
    email     VARCHAR(100),
    nota      DECIMAL(4,2) DEFAULT NULL,
    data_alta DATE         NOT NULL DEFAULT (CURRENT_DATE),
    grup_id   INT UNSIGNED,
    CONSTRAINT uq_alumne_email UNIQUE (email),
    CONSTRAINT ck_alumne_nota  CHECK (nota BETWEEN 0 AND 10),
    CONSTRAINT fk_alumne_grup  FOREIGN KEY (grup_id)
        REFERENCES grup(id)
        ON DELETE SET NULL
        ON UPDATE CASCADE
) ENGINE=InnoDB;
```

Altres variants útils:

```
CREATE TABLE copia LIKE alumne;                          -- només
 ↪  estructura
CREATE TABLE resum AS SELECT id, nom FROM alumne;        -- estructura
 ↪  + dades
CREATE OR REPLACE TABLE alumne (...);                   -- específic de
 ↪  MariaDB
CREATE TEMPORARY TABLE tmp (...);
```

Restriccions

Una restricció és una regla definida sobre una taula que el SGBD verifica automàticament cada vegada que s'insereix, modifica o elimina una dada. Si l'operació la incompleix, MariaDB la rebutja amb un error. Serveixen per garantir la integritat i la qualitat de les dades, sense dependre de l'aplicació.

La integritat referencial és la regla que garanteix que cada valor d'una clau forana existeix a la taula que referencia (o és NULL). És a dir, no pot haver-hi una referència a una cosa que no existeix.

Restricció	Què garanteix	Notes a MariaDB
`NOT NULL`	El camp no pot ser nul	Es defineix a la columna
`DEFAULT`	Valor per defecte	Admet expressions des de 10.2.1: `DEFAULT (CURRENT_DATE)`
`PRIMARY KEY`	Identifica la fila: únic + `NOT NULL`	Només una per taula; pot ser composta
`UNIQUE`	Valors no repetits	Admet múltiples `NULL`; crea un índex
`FOREIGN KEY`	Integritat referencial	Només InnoDB; la columna ha d'estar indexada
`CHECK`	Condició sobre els valors	**Aplicada des de MariaDB 10.2.1**; abans s'ignorava
`AUTO_INCREMENT`	Número seqüencial	No és una constraint, però va lligat a la PK

Accions de clau forana (`ON DELETE` / `ON UPDATE`): `RESTRICT`, `NO ACTION`, `CASCADE`, `SET NULL`.

Claus compostes i checks a escala de taula:

```
PRIMARY KEY (alumne_id, modul_id)
CONSTRAINT ck_dates CHECK (data_fi > data_inici)
```

Punts a tenir en compte

- Els noms de constraints en MariaDB són per taula; convé usar prefixos (`pk_`, `fk_`, `uq_`, `ck_`).
- Si crees una FK sense nom, MariaDB en genera un automàticament, cosa que dificulta el `DROP FOREIGN KEY` posterior.
- Les columnes d'una FK han de tenir **el mateix tipus exacte** (inclòs `UNSIGNED`) que la columna referenciada.

Modificació de taules

```
-- Columnes
ALTER TABLE alumne ADD COLUMN telefon VARCHAR(15) AFTER email;
ALTER TABLE alumne MODIFY COLUMN nom VARCHAR(80) NOT NULL;
ALTER TABLE alumne CHANGE COLUMN telefon mobil VARCHAR(20);   --
 ↪  canvia nom i tipus
ALTER TABLE alumne RENAME COLUMN mobil TO tel;               -- 10.5.2+
ALTER TABLE alumne ALTER COLUMN nota SET DEFAULT 0;
ALTER TABLE alumne ALTER COLUMN nota DROP DEFAULT;
ALTER TABLE alumne DROP COLUMN tel;

-- Afegir restriccions
ALTER TABLE alumne ADD PRIMARY KEY (id);
ALTER TABLE alumne ADD CONSTRAINT uq_email UNIQUE (email);
ALTER TABLE alumne ADD CONSTRAINT ck_nota CHECK (nota BETWEEN 0 AND
 ↪  10);
ALTER TABLE alumne ADD CONSTRAINT fk_grup
    FOREIGN KEY (grup_id) REFERENCES grup(id) ON DELETE CASCADE;

-- Eliminar restriccions (cada tipus té la seva sintaxi)
ALTER TABLE alumne DROP PRIMARY KEY;
ALTER TABLE alumne DROP INDEX uq_email;          -- UNIQUE
ALTER TABLE alumne DROP FOREIGN KEY fk_grup;     -- no esborra l'índex
 ↪  associat
ALTER TABLE alumne DROP CONSTRAINT ck_nota;      -- CHECK

-- Taula
ALTER TABLE alumne ENGINE=InnoDB;
ALTER TABLE alumne RENAME TO estudiant;
ALTER TABLE alumne AUTO_INCREMENT = 1000;
```

Índexs

Un índex és una estructura de dades auxiliar que la base de dades manté al costat d'una taula per localitzar files ràpidament sense haver de llegir-la sencera. Funciona com l'índex alfabètic d'un llibre: en lloc de passar full per full, vas directament a la pàgina on és el que busques.

```
CREATE INDEX idx_nom ON alumne(nom);
CREATE UNIQUE INDEX idx_email ON alumne(email);
CREATE FULLTEXT INDEX idx_ft ON alumne(nom);
DROP INDEX idx_nom ON alumne;
```

Vistes

Una vista és una consulta SELECT desada amb un nom, que es comporta com una taula virtual. No emmagatzema dades pròpies: cada vegada que la consultes, MariaDB executa la consulta subjacent sobre les taules reals.

```
CREATE OR REPLACE VIEW v_aprovats AS SELECT * FROM alumne WHERE nota
 ↪  >= 5;
DROP VIEW IF EXISTS v_aprovats;
```

Buidar, eliminar i reanomenar

```
TRUNCATE TABLE alumne;                          -- buida i reinicia
 ↪  AUTO_INCREMENT
DROP TABLE IF EXISTS alumne, grup;              -- es poden eliminar-ne
 ↪  diverses
RENAME TABLE alumne TO estudiant, grup TO classe;
```

`TRUNCATE` falla si la taula és referenciada per una FK. Cal desactivar-la temporalment amb `SET FOREIGN_KEY_CHECKS = 0;` (i tornar-la a activar) o buidar primer les taules filles.

Consulta de l'estructura

```
SHOW DATABASES;
SHOW TABLES;
DESCRIBE alumne;                  -- o SHOW COLUMNS FROM alumne
SHOW CREATE TABLE alumne\G        -- DDL complet, amb constraints
SHOW INDEX FROM alumne;
SELECT * FROM information_schema.TABLE_CONSTRAINTS
 WHERE TABLE_SCHEMA = 'institut';
```

Consulta i modificació de variables

```
SHOW VARIABLES;                   -- totes (sessió)
SHOW GLOBAL VARIABLES;            -- valors globals
SHOW VARIABLES LIKE 'innodb%';    -- filtra per patró
SET foreign_key_checks = 0;
```

15. Exercici

Exercici IES: enunciat, model ER, model relacional i model físic

16. Referències

- Obiols Llopart, C.; Guixà Miranda, I. (adaptació de continguts). *Llenguatge SQL. Consultes*. Mòdul *Gestió de bases de dades*, unitat 4. Fitxer `fp_asx_m02_u4`. S'ha consultat sobretot el capítol 1 (tipus de sentències SQL i tipus de dades de MySQL).
- W3Schools. *MySQL Tutorial* (seccions *Database*, *Constraints*, *Create/Drop Table*, *Alter Table*, *Create Index*, *Auto Increment* i *Views*): https://www.w3schools.com/mysql/
- Documentació oficial de MySQL 8.0: *Data Definition Statements* (https://dev.mysql.com/doc/refman/8.0/en/sql-data-definition-statements.html) i *Data Types* (https://dev.mysql.com/doc/refman/8.0/en/data-types.html).

Versions d'aquest document

- HTML
- PDF
- ODT
- MD

Domini Públic (CC0)