
Grau en Enginyeria Informàtica

Sistemes de Gestió de Bases de Dades

# Pràctica Final

Lluis Barca Pons
lluis.barca1@estudiant.uib.es

22 de gener de 2024

# *Índex*

# 1. *Introducció*

La Reial Federació Taurina d'Espanya, amb una història rica i complexa, ha acumulat un vast llegat de dades des de la seva fundació el 1949. Des de l'assignatura de Sistemes de Gestió de Bases de Dades se'ns ha proposat fer aquest treball que fa èmfasi tant el disseny, la manipulació i migració de dades en un entorn semirealista. Comencem amb la construcció d'un model de dades robust, dissenyat per reflectir la xarxa de relacions entre toreros, apoderats, actuacions, esdeveniments i les places de toros que són l'escenari d'aquest art cultural. S'ha prestat especial atenció a les restriccions, verificacions i disparadors necessaris per preservar la integritat de les dades, aconseguint així la normalització desitjada sense sacrificar la flexibilitat operativa.

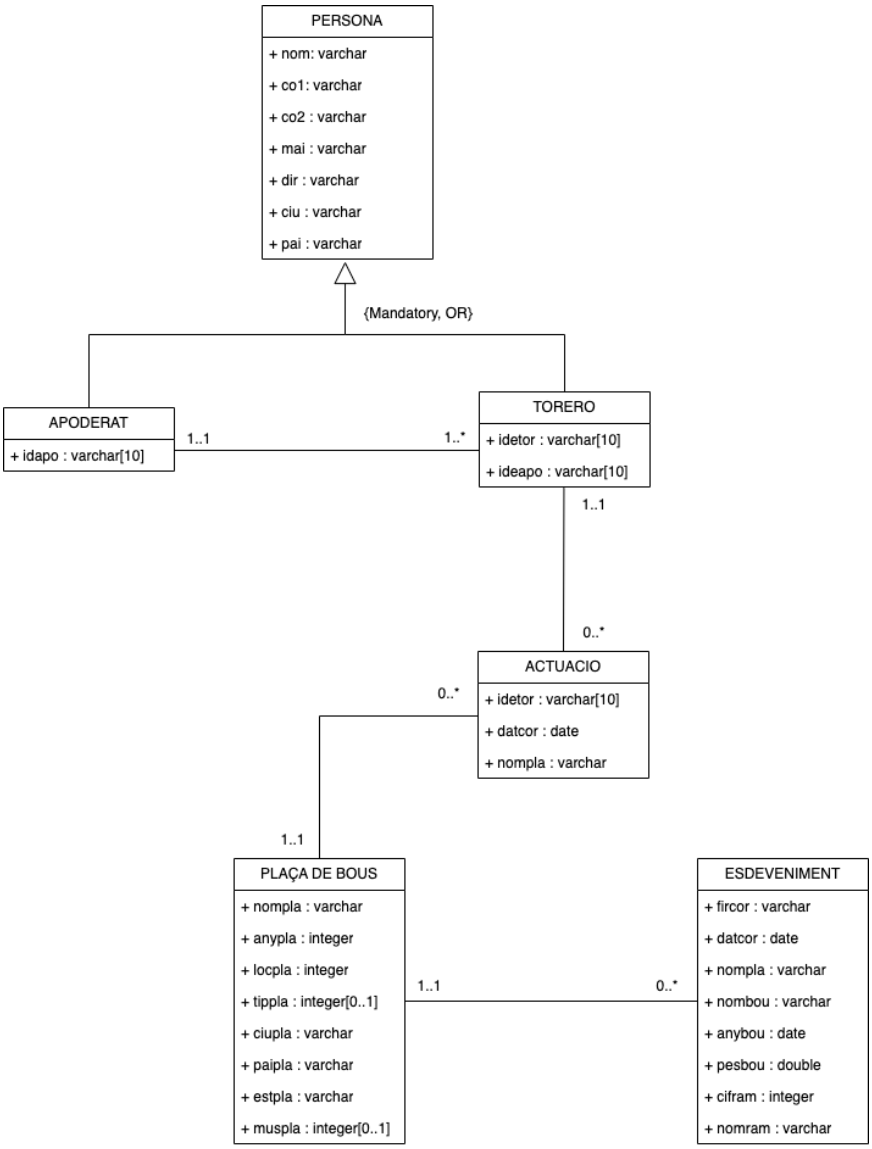
Amb una implementació meticulosa cap a MySQL, i posteriorment la migració cap a PostgreSQL, hem traslladat sàviament la riquesa de la història taurina a una infraestructura digital moderna. La migració no només garanteix la preservació de les dades, sinó que també obre noves possibilitats per anàlisi i accessibilitat. Durant aquest procés, s'han establert usuaris especialitzats i privilegis necessaris per facilitar la gestió eficient i segura de les dades.

S'han implementat consultes SQL específiques per extreure'n coneixements significatius, com la distribució de seients per plaça i país, i hem pres mesures proactives per optimitzar el rendiment de les consultes, garantint respostes ràpides i eficients. Aquest document reflecteix tota la feina duta a terme.

En resum, aquest treball no sols documenta la transició de registres històrics a un sistema gestionable i consultable, sinó que posa en practica els diversos coneixements adquirits durant tot el curs.

# 2. *Anàlisi i modelat de les dades*

## 2.1 Model conceptual



### 2.1.1 Aclariments i suposicions

En el model conceptual podem observar, en primer lloc, una herència anomenada `PERSONA`, que ens permet encapsular tots aquells atributs que engloben un objecte persona. En aquest cas es tractaria tant de l'apoderat com el torero, ambdós amb pràcticament els mateixos atributs, a excepció dels seus identificadors. Aquesta herència tindrà una restricció `Mandatory, OR`, la qual assegura que, una persona, pot ser tant Torero com Apoderat en qualque moment de la seva vida. Això s'ha implementat així perquè habitualment els apoderats solen ser persones amb experiència dins el sector de la tauromàquia i, per tant, poden haver estat toreros. Amb aquesta implementació, aconseguim una reducció de la redundància de dades a la nostra base de dades.

Seguidament, ens trobem les taules `APODERAT` i `TORERO`, les quals tenen una relació `1..1` i `1..*`. És a dir, un torero podrà tenir un únic apoderat, però un apoderat podrà tenir diversos toreros; perquè els apoderats viuen pràcticament de mentoritzar a aquests toreros i inclús tenen escoles dedicades exclusivament. A continuació ens trobem amb la taula `ACTUACIO`, que està relacionada amb la taula `TORERO` amb una relació `1..*` a `0..*`, la qual cosa significa que un torero pot haver participat a zero actuacions o a moltes i que una actuació pot tenir com a protagonista un únic torero; en cas d'estar torejant dos toreros en una mateixa actuació, tindríem dues actuacions al mateix moment (la qual cosa és molt poc probable).

A continuació trobem una relació entre les taules `ACTUACIO` i `PLAÇA DE BOUS`, on una actuació només pot estar assignada a una plaça de bous i que una plaça de bous pot tenir definides de zero a moltes actuacions; per poder tenir en compte les noves places de bous que mai han tingut una actuació. Després tenim que la taula `PLAÇA DE BOUS` està relacionada amb la taula `ESDEVENIMENT` per una relació `0..*` i `1..1`. És a dir, una plaça de bous pot tenir de zero a molts esdeveniments i un esdeveniment pot tenir únicament una plaça de bous associada.

## 2.2 Model relacional

### 2.2.1 Taules

PERSONA(nom, co1, co2, mai, dir, ciu, pai)

APODERAT(ideapo)

TORERO(idetor, ideapo)

ACTUACIO(idetor, datcor, nompla)

PLAÇA DE BOUS(nompla, anypla, locpla, tippla, ciupla, paipla, estpla, muspla)

ESDEVENIMENT(fircor, datcor, nompla, nombou, anybou, pesbou, cifram, nomram)

### 2.2.2 Elecció de claus primàries

PERSONA(**ideper**, nom, co1, co2, mai, dir, ciu, pai)

APODERAT(**ideapo**)

TORERO(**idetor**, ideapo)

ACTUACIO(**idact**, *idetor*, datcor, *nompla*)

PLAÇA DE BOUS(**nompla**, anypla, locpla, tippla, ciupla, paipla, estpla, muspla)

ESDEVENIMENT(**idesd**, fircor, datcor, nompla, nombou, anybou, pesbou, cifram, nomram)

### 2.2.3 Relacional

PERSONA(**ideper**, nom, co1, co2, mai, dir, ciu, pai)

APODERAT(***ideapo***)
{ideapo FK de la taula PERSONA}

TORERO(***idetor***, *ideapo*)
{ideapo FK de la taula APODERAT}
{idetor FK de la taula PERSONA}

ACTUACIO(**idact**, *idetor*, datcor, *nompla*)
{idetor FK de la taula TORERO}
{nompla FK de la taula PLAÇA DE BOUS}

PLAÇA DE BOUS(**nompla**, anypla, locpla, tippla, ciupla, paipla, estpla, muspla)

ESDEVENIMENT(**idesd**, fircor, datcor, nompla, nombou, anybou, pesbou, cifram, nomram)

### 2.2.4 Normalització

- `PERSONA`
  - Clau primària: **ideper**.
  - Tots els camps són atòmics, per la qual cosa és a Primera Forma Normal (1NF).
  - Atès que la clau primària és única i no hi ha camps no claus dependents d'una part d'una clau primària (ja que no és composta), la taula també és a Segona Forma Normal (2NF).

  - Sense dependències transitives aparents, podem considerar que és a Tercera Forma Normal (3NF).

- `APODERAT`
  - Clau primària: **ideapo**.
  - Només conté la clau primària, de manera que automàticament compleix amb 1NF, 2NF i 3NF.

- `TORERO`
  - Clau primària: **idetor**. Clau forana: **ideapo**.
  - Compleix 1NF, ja que tots els camps són atòmics.
  - Compleix 2NF i 3NF, ja que no hi ha camps no clau i l'única clau forana està directament relacionada amb la clau primària.

- `ACTUACIÓ`
  - Clau primària: **idact**. Claus foranes: **idetor, nompla**.
  - Compleix amb 1NF.
  - Atès que la clau primària és única i no hi ha camps no claus dependents d'una part d'una clau primària, compleix amb 2NF.
  - Compleix amb 3NF, ja que no hi ha dependències transitives entre les claus i els camps no clau.

- `PLAÇA DE BOUS`
  - Clau primària: **nompla**.
  - Compleix 1NF.
  - Com que és la clau primària única i no hi ha camps no clau que depenguin d'una part d'una clau primària, compleix 2NF.
  - Compleix amb 3NF, ja que no hi ha dependències transitives.

- `ESDEVENIMENT`
  - Clau primària: **idesd**.
  - Compleix la 1NF perquè tots els camps són atòmics.
  - Com que no hi ha camps no clau que depenguin d'una part d'una clau primària, compleix la 2NF.
  - Compleix la 3NF, ja que no hi ha dependències transitives entre la clau primària i els altres camps.

En resum, totes les taules proporcionades estan a la Tercera Forma Normal (3NF). Aquesta conclusió es basa en la informació proporcionada i assumeix que no hi ha dependències funcionals addicionals no especificades. També es podria arribar a 4NF o 5NF, però atès el disseny de la base de dades, no seria el més òptim.

# 3. *Implementació amb MySQL*

## 3.1 Creació de l'espai d'emmagatzemament

### 3.1.1 Modificació del paràmetre `innodb_data_file_path`

Per a la creació de l'espai d'emmagatzemament de, concretament 2GB, haurem de modificar l'arxiu de configuració de MySQL on estableix l'espai atorgat per a cada base de dades.

```
/usr/local/etc/my.cnf
```

Farem una còpia d'aquest fitxer, per si un cas necessitem restablir la configuració:

```
sudo cp /usr/local/etc/my.cnf /usr/local/etc/my.cnf.backup

sudo nano /usr/local/etc/my.cnf
```

Dins aquest fitxer hem afegit la línia següent `innodb_data_file_path = ibdata1:10M:autoextend:max:2G` que ens permet configurar un màxim de 2GB per a qualsevol base de dades creada.

```
SHOW VARIABLES LIKE 'innodb_data_file_path';
```

```
mysql> SHOW VARIABLES LIKE 'innodb_data_file_path';
+-----------------------+-------------------------------+
| Variable_name         | Value                         |
+-----------------------+-------------------------------+
| innodb_data_file_path | ibdata1:10M:autoextend:max:2G |
+-----------------------+-------------------------------+
1 row in set (0.01 sec)
```

### 3.1.2 Creació del TABLESPACE

Com ja s'ha mencionat, a MySQL no podem crear un TABLESPACE amb una mida màxima per defecte. És per això que s'ha realitzat la modificació anterior de l'arxiu de configuració de MySQL.

Llavors ara podem crear el TABLESPACE que, com a màxim, tindrà una mida de 2GB, ja que la mateixa aplicació de MySQL no deixarà una grandària major. Aquest l'emmagatzemem al directori creat a continuació:

```
sudo mkdir /var/lib/mysql/my_tablespace
```

I a continuació el TABLESPACE:

```
CREATE TABLESPACE tb1
ADD DATAFILE 'tb1.ibd'
ENGINE = InnoDB;
```

Especificant l'*engine* podem assegurar-nos que utilitzarà l'innodb per crear el TABLESPACE.

## 3.2 Creació de la base de dades i l'usuari IMPBD

Per crear la base de dades a MySQL Server haurem d'executar el següent codi SQL:

```
CREATE DATABASE IMPBD;

# Podem comprovar la correcta creació amb la comanda:
SHOW DATABASES;
```

```
mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| IMPBD              |
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.00 sec)
```

A continuació creem l'usuari IMPBD i li donem permisos:

```
CREATE USER 'IMPBD'@'localhost' IDENTIFIED BY 'admin';
GRANT CREATE, INSERT, UPDATE, DELETE, REFERENCES, FILE ON IMPBD.* TO
↪  'IMPBD'@'localhost';
FLUSH PRIVILEGES;
```

Ara, haurem d'iniciar `mysql` des de l'usuari creat i crear les respectives taules de la base de dades, segons els fitxers `.csv` aportats per la Real Federació Taurina d'Espanya.

Primer haurem de seleccionar la base de dades:

```
USE IMPBD;
```

Després executarem el script SQL que crearà les respectives taules (vegeu a la secció 6).

### 3.2.1 Descripció dels *constraints*, *checks* i *triggers* implementats

- `PERSONA`
  - **Constraints**
    - **ideper**: Clau primària per identificar a totes les persones. Aquesta clau realment serà l'identificador o del torero o de l'apoderat.
- `APODERAT`
  - **Constraints**
    - **ideapo**: Clau primària per identificar a una persona que és apoderat.
- `TORERO`
  - **Constraints**
    - **idetor**: Clau primària per identificar a una persona que és torero.
- `ACTUACIO`
  - **Constraints**
    - **idact**: Clau primària que ens permetrà identificar les diferents actuacions. Aquesta s'autoincrementa cada vegada que s'insereix una actuació. D'aquesta forma la gestió d'aquesta clau és molt senzilla.
    - **idetor**: Clau forana que apunta a la classe `TORERO`.
    - **nompla**: Clau forana que apunta a la classe `PLAÇA DE BOUS`.

– **Triggers**
Aquest *trigger* s'assegura que només s'insereixin actuacions amb toreros i places de toros existents i en dates futures.

```
DELIMITER $$
CREATE TRIGGER TRG_ACTUACIO_INSERT
BEFORE INSERT ON ACTUACIO
FOR EACH ROW
BEGIN
    IF NEW.datcor < CURDATE() THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'No es pot
        ↪ insertar una actuació amb una data anterior a avui.';
    END IF;
    IF NOT EXISTS (SELECT * FROM TORERO WHERE idetor = NEW.idetor)
    ↪ THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'No es pot
        ↪ insertar una actuació amb un torero inexistent.';
    END IF;
    IF NOT EXISTS (SELECT * FROM PLAÇA_DE_BOUS WHERE nompla =
    ↪ NEW.nompla) THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'No es pot
        ↪ insertar una actuació en una plaça de bous
        ↪ inexistent.';
    END IF;
END$$
DELIMITER ;
```

- PLAÇA_DE_BOUS
  – **Constraints**
    * **nompla**: El nom de la plaça seria la nostra clau primària, ja que no volem tenir places duplicades.
  – **Checks**
    * **tippla**: Verifica que el valor d'aquesta columna sigui 0 o 1. Aquests valors ens informen de si la plaça de bous és fixa o mòbil.
    * **muspla**: Verifica que el valor d'aquesta columna sigui 0 o 1. Aquests valors ens informen de si la plaça de bous té o no museu.
  – **Triggers**
  Aquest *trigger* s'encarrega de verificar que cada plaça de bous inserida, tingui com a capacitat màxima, l'establida per llei.

```
DELIMITER $$
CREATE TRIGGER TRG_PLAÇA_DE_BOUS_INSERT
BEFORE INSERT ON PLAÇA_DE_BOUS
FOR EACH ROW
BEGIN
    IF NEW.locpla > 10000 THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'No poden haver
        ↪ places de bous amb una capacitat major a 10.000
        ↪ seients.';
    END IF;
END$$
DELIMITER ;
```

- ESDEVENIMENT
  - **Constraints**
    - **idesd**: Clau primària que ens permetrà identificar els diferents esdeveniments. Aquesta s'autoincrementa cada vegada que s'insereix un esdeveniment. D'aquesta forma la gestió d'aquesta clau és molt senzilla.
  - **Triggers**
    Aquest *trigger* s'assegura que tots els esdeveniments registrats, es facin amb una data futura a l'actual data d'inserció. D'aquesta manera es garanteix la integritat de les dades.

```
DELIMITER $$
CREATE TRIGGER TRG_ESDEVENIMENT_INSERT
BEFORE INSERT ON ESDEVENIMENT
FOR EACH ROW
BEGIN
    IF NEW.datcor < CURDATE() THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'No es pot
        ↪ insertar un esdeveniment amb una data anterior a
        ↪ avui.';
    END IF;
END$$
DELIMITER ;
```

## 3.3 Importació de les dades a IMPBD

En primer lloc, per poder inserir dades de forma "automàtica" utilitzant la comanda `LOAD DATA LOCAL INFILE`, que ens permet importar dades des d'un arxiu CSV, s'ha de modificar l'arxiu de configuració de MySQL. Concretament s'ha d'afegir la següent línia:

```
local_infile = 1
```

La qual activa la funció perquè el sistema de gestió de base de dades permeti la inserció de dades a través de la comanda descrita anteriorment. A continuació s'ha d'entrar a la terminal de MySQL amb l'usuari corresponent, però marcant el paràmetre `local-infile` a 1.

```
mysql --local-infile=1 -u IMPBD -p
```

A partir d'aquí, ja podem utilitzar els següents scripts que s'han implementat per a la importació total de les dades:

- `PERSONA`

  Per aquesta taula, s'ha hagut de fusionar els dos arxius CSV de toreros i apoderats amb un script programat en Python que trobareu a l'annex (secció 7).

  Ara ja, amb el nou PERSONA.csv, podem realitzar la inserció de dades:

  ```
  LOAD DATA LOCAL INFILE
  ↪  '/Users/luisbarcap/Desktop/sgbd_final/PERSONAS.csv'
  INTO TABLE PERSONA
  FIELDS TERMINATED BY ','
  ENCLOSED BY '"'
  LINES TERMINATED BY '\n'
  IGNORE 1 ROWS;
  ```

- `APODERAT`

  Per aquesta taula, s'ha hagut de programar un script, que trobareu a l'annex (secció 7), que genera un nou arxiu CSV amb l'atribut `ideapo` únicament; ja que serà l'únic que necessitem per importar a la nostra taula `APODERAT`.

Ara ja, amb el nou APODERATS_edit.csv, podem realitzar la inserció de dades:

```
LOAD DATA LOCAL INFILE
↪ '/Users/luisbarcap/Desktop/sgbd_final/APODERATS_edit.csv'
INTO TABLE APODERAT
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;
```

- `TORERO`

  Per aquesta taula, s'ha hagut de programar un script, que trobareu a l'annex (secció 7), que genera un nou arxiu CSV amb els atributs `idetor, ideapo` únicament; ja que seran els únics que necessitem per importar a la nostra taula `TORERO`.

  Ara ja, amb el nou TOREROS_edit.csv, podem realitzar la inserció de dades:

```
LOAD DATA INFILE
↪ '/Users/luisbarcap/Desktop/sgbd_final/TOREROS_edit.csv'
INTO TABLE TORERO
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;
```

- `ACTUACIO`

```
LOAD DATA INFILE
↪ '/Users/luisbarcap/Desktop/sgbd_final/ACTUACIONS.csv'
INTO TABLE ACTUACIO
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS
(idetor, datcor, nompla);
```

- ESDEVENIMENT

```
LOAD DATA INFILE
↪  '/Users/luisbarcap/Desktop/sgbd_final/esdeveniments.csv'
INTO TABLE ESDEVENIMENT
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS
(fircor, datcor, nompla, nombou, anybou, pesbou, cifram, nomram);
```

## 3.4 Creació de la base de dades DEFBD i l'usuari MIGBD

Crearem la base de dades utilitzant l'usuari `root`:

```
CREATE DATABASE DEFBD;
```

I li assignarem els permisos pertinents a l'usuari MIGBD perquè pugui interactuar amb aquesta:

```
CREATE USER 'MIGBD'@'localhost' IDENTIFIED BY 'admin';
GRANT CREATE, SELECT, INSERT, REFERENCES ON DEFBD.* TO 'MIGBD'@'localhost';
FLUSH PRIVILEGES;
```

Llavors amb aquest usuari, podem inserir les taules a la base de dades DEFBD, utilitzant el mateix codi SQL que s'ha utilitzat a la secció 3.2.

## 3.5 Traspàs de la informació de les taules creades amb l'usuari IMPBD a l'usuari MIGBD

### 3.5.1 Modificació de privilegis

Donem el privilegi de `SELECT` a l'usuari MIGBD sobre la base de dades IMPBD perquè pugui consultar-la:

```
GRANT SELECT ON IMPBD.* TO 'MIGBD'@'localhost';
FLUSH PRIVILEGES;
```

### 3.5.2 Transferència de les dades

Per a efectuar la transferència de les dades d'IMPBD a DEFBD, primer haurem d'accedir a MySQL amb l'usuari MIGBD i a continuació especificar la base de dades on importarem les dades:

```
USE DEFBD;
```

Ara ja podem executar els diferents *inserts* que efectuaran la migració:

```
INSERT INTO PERSONA (ideper, nom, co1, co2, mai, dir, ciu, pai)
SELECT ideper, nom, co1, co2, mai, dir, ciu, pai
FROM IMPBD.PERSONA;

INSERT INTO APODERAT (ideapo)
SELECT ideapo
FROM IMPBD.APODERAT;

INSERT INTO TORERO (idetor, ideapo)
SELECT idetor, ideapo
FROM IMPBD.TORERO;

INSERT INTO PLAÇA_DE_BOUS (nompla, anypla, locpla, tippla, ciupla, paipla,
↪  estpla, muspla)
SELECT nompla, anypla, locpla, tippla, ciupla, paipla, estpla, muspla
FROM IMPBD.PLAÇA_DE_BOUS;

INSERT INTO ACTUACIO (idetor, datcor, nompla)
SELECT idetor, datcor, nompla
FROM IMPBD.ACTUACIO;

INSERT INTO ESDEVENIMENT (fircor, datcor, nompla, nombou, anybou, pesbou,
↪  cifram, nomram)
SELECT fircor, datcor, nompla, nombou, anybou, pesbou, cifram, nomram
FROM IMPBD.ESDEVENIMENT;
```

## 3.6 Actualització dels seients màxims a les places de bous

Per a definir el màxim de places de bous, es pot executar la següent comanda SQL, la qual modificarà totes les places de bous que excedeixen de 10.000 places.

```sql
UPDATE PLAÇA_DE_BOUS
SET locpla = 10000
WHERE locpla > 10000;
```

```
mysql> UPDATE PLAÇA_DE_BOUS
    ->     SET locpla = 10000
    ->     WHERE locpla > 10000;
Query OK, 33 rows affected (0.02 sec)
Rows matched: 33  Changed: 33  Warnings: 0
```

A més, també s'ha implementat un *trigger* per a quan es faci una inserció d'una nova plaça de bous, es comprovi que compleix aquesta normativa, tal com veiem a la secció 3.2.1.

## 3.7 Millores de rendiment

Analitzarem l'execució de tres consultes que podrien ser bastant freqüents en aquesta base de dades i mirarem com podem millorar (si podem) el seu rendiment. Cal mencionar que les dades (com alguns id, dates, etc) que s'agafaran per aquests exemples són totalment aleatòries, única i exclusivament per analitzar l'execució de les consultes.

1. **Actuacions d'un torero en un període de temps concret**

```sql
SELECT A.idetor, A.datcor, A.nompla
FROM ACTUACIO A
JOIN TORERO T ON A.idetor = T.idetor
WHERE T.idetor = '94961045W'
  AND A.datcor BETWEEN '1801-01-02' AND '1801-12-18';
```

Utilitzant la comanda `EXPLAIN` podem veure el pla d'execució de la consulta:

```
mysql> EXPLAIN SELECT A.idetor, A.datcor, A.nompla
    ->         FROM ACTUACIO A
    ->         JOIN TORERO T ON A.idetor = T.idetor
    ->         WHERE T.idetor = '94961045W'
    ->           AND A.datcor BETWEEN '1801-01-02' AND '1801-12-18';
+----+-------------+-------+------------+-------+---------------+---------+---------+-------+------+----------+-----------------------------------+
| id | select_type | table | partitions | type  | possible_keys | key     | key_len | ref   | rows | filtered | Extra                             |
+----+-------------+-------+------------+-------+---------------+---------+---------+-------+------+----------+-----------------------------------+
|  1 | SIMPLE      | T     | NULL       | const | PRIMARY       | PRIMARY | 40      | const |    1 |   100.00 | Using index                       |
|  1 | SIMPLE      | A     | NULL       | ref   | idetor        | idetor  | 40      | const |   33 |    11.11 | Using index condition; Using where |
+----+-------------+-------+------------+-------+---------------+---------+---------+-------+------+----------+-----------------------------------+
2 rows in set, 1 warning (0.00 sec)
```

- La taula `TORERO` utilitza una cerca per clau primària, la qual cosa és eficient (indicat per "Using inde" a la columna "Extra"). Això vol dir que MySQL pot trobar ràpidament la fila corresponent utilitzant l'índex de la clau primària.

- La taula `ACTUACIO` utilitza una operació de referència (indicat per "ref" a la columna "type") per unir-se a la taula `TORERO`, cosa que també és força eficient. A més, s'utilitza una condició d'índex i una clàusula `WHERE` per filtrar els resultats.

El fet que les dues taules mostrin "Using index" suggereix que ja s'està fent ús d'índexs, el qual és una cosa bastant bona. El nombre de files que s'estan llegint de la taula `APODERAT` és 33, cosa que sembla raonable si aquest és el nombre total de files que corresponen a les condicions del `WHERE`.

En resum, la consulta ja sembla optimitzada en termes d'ús d'índexs

2. **Nom i identificador d'un torero donat el seu apoderat**

```sql
SELECT T.idetor, P.nom
FROM TORERO T
JOIN APODERAT A ON T.ideapo = A.ideapo
JOIN PERSONA P ON T.idetor = P.ideper
WHERE P.nom = 'Gaston';
```

**Temps de resposta: 0.09s**
**Files afectades: 475**

Utilitzant la comanda `EXPLAIN` podem veure el pla d'execució de la consulta:

```
mysql> EXPLAIN SELECT T.idetor, P.nom
    ->         FROM TORERO T
    ->         JOIN APODERAT A ON T.ideapo = A.ideapo
    ->         JOIN PERSONA P ON T.idetor = P.ideper
    ->         WHERE P.nom = 'Gaston';
+----+-------------+-------+------------+--------+---------------+---------+---------+---------------+--------+----------+-------------+
| id | select_type | table | partitions | type   | possible_keys | key     | key_len | ref           | rows   | filtered | Extra       |
+----+-------------+-------+------------+--------+---------------+---------+---------+---------------+--------+----------+-------------+
|  1 | SIMPLE      | P     | NULL       | ALL    | PRIMARY       | NULL    | NULL    | NULL          | 235986 |    10.00 | Using where |
|  1 | SIMPLE      | T     | NULL       | eq_ref | PRIMARY,ideapo | PRIMARY | 40      | defbd.P.ideper |      1 |   100.00 | NULL        |
|  1 | SIMPLE      | A     | NULL       | eq_ref | PRIMARY       | PRIMARY | 40      | defbd.T.ideapo |      1 |   100.00 | Using index |
+----+-------------+-------+------------+--------+---------------+---------+---------+---------------+--------+----------+-------------+
3 rows in set, 1 warning (0.01 sec)
```

- La taula `PERSONA` està fent un *full table scan*, indicat pel tipus "ALL" a la columna "type" i "Using where" a la columna "Extra". Això vol dir que s'està examinant cada fila de la taula per trobar coincidències, el que és costós en termes de rendiment, especialment perquè hi ha 235986 files per revisar.
- Les taules `TORERO` i `APODERAT` estan utilitzant correctament els índexs, com es veu a "Using index" i el tipus "eq_ref", que és un dels tipus de *join* més eficients a MySQL.

Llavors podem crear un índex sobre la columna "nom" de la taula `PERSONA` per intentar millorar el rendiment:

```sql
CREATE INDEX idx_nom_persona ON PERSONA(nom);
```

I efectivament reduïm en un 1/9 el temps de resposta:

**Temps de resposta: 0.01s**
**Files afectades: 475**

3. **Nombre d'esdeveniments a cada plaça de bous, ordenat alfabèticament**

```sql
SELECT P.nompla, COUNT(E.nompla) AS NumeroDeEventos
FROM PLAÇA_DE_BOUS P
LEFT JOIN ESDEVENIMENT E ON P.nompla = E.nompla
GROUP BY P.nompla
ORDER BY P.nompla ASC;
```

**Temps de resposta: 9.03s**
**Files afectades: 266**

Utilitzant la comanda `EXPLAIN` podem veure el pla d'execució de la consulta:

```
mysql> EXPLAIN SELECT P.nompla, COUNT(E.nompla) AS NumeroDeEventos
    ->         FROM PLAÇA_DE_BOUS P
    ->         LEFT JOIN ESDEVENIMENT E ON P.nompla = E.nompla
    ->         GROUP BY P.nompla
    ->         ORDER BY P.nompla ASC;
+----+-------------+-------+------------+-------+---------------+---------+---------+------+---------+----------+----------------------------------------------+
| id | select_type | table | partitions | type  | possible_keys | key     | key_len | ref  | rows    | filtered | Extra                                        |
+----+-------------+-------+------------+-------+---------------+---------+---------+------+---------+----------+----------------------------------------------+
|  1 | SIMPLE      | P     | NULL       | index | PRIMARY       | PRIMARY | 1022    | NULL |     266 |   100.00 | Using index; Using temporary; Using filesort |
|  1 | SIMPLE      | E     | NULL       | ALL   | NULL          | NULL    | NULL    | NULL | 5034602 |   100.00 | Using where; Using join buffer (hash join)   |
+----+-------------+-------+------------+-------+---------------+---------+---------+------+---------+----------+----------------------------------------------+
2 rows in set, 1 warning (0.01 sec)
```

- La taula `PLAÇA_DE_BOUS` utilitza un índex, la qual cosa és positiva. No obstant això, també indica que s'utilitza una taula temporal i un ordenament de fitxers ("Using temporary; Using filesort"), cosa que sol passar quan es realitzen operacions de `GROUP BY` i `ORDER BY`. Això pot ser ineficient, especialment si el conjunt de resultats és gran.
- La taula `ESDEVENIMENT` realitza un *full table scan*, indicat pel tipus "ALL" a la columna "type". Això és costós en termes de rendiment, ja que implica que MySQL està examinant cada fila de la taula per trobar coincidències.

Llavors podem crear un índex sobre la columna "nompla" de la taula `ESDEVENIMENT` per intentar millorar el rendiment:

```sql
CREATE INDEX idx_nompla_esdeveniment ON ESDEVENIMENT(nompla);
```

I efectivament reduïm en un 1/3 el temps de resposta:

**Temps de resposta: 3.08s**
**Files afectades: 266**

# 4. *Migració a PostgreSQL*

Per a accedir com a usuari `root` al sistema de gestió de base de dades de PostgreSQL, utilitzarem l'usuari `postgres`.

```
psql postgres
```

## 4.1 Creació de l'espai d'emmagatzemament

Creem un `TABLESPACE` a una ubicació escollida prèviament:

```
CREATE TABLESPACE PracticaFinal OWNER postgres LOCATION
↪  'Users/luisbarcap/PostgreSQL/sgbd';
```

## 4.2 Creació de la base de dades MIGBD, esquema i usuari

Creem la base de dades i especifiquem que s'emmagatzemi al `TABLESPACE` creat anteriorment:

```
CREATE DATABASE migbd TABLESPACE PracticaFinal;
```

Ara ens podem connectar a la base de dades i crear l'esquema:

```
\connect migbd
CREATE SCHEMA sgbd;
```

Ara llavors podem crear l'usuari i donar-li els permisos requerits:

```
CREATE USER migbd WITH PASSWORD 'admin';
GRANT SELECT, INSERT ON ALL TABLES IN SCHEMA sgbd TO migbd;
GRANT CREATE ON SCHEMA sgbd TO migbd;
```

## 4.3 Creació de la base de dades DEFBD, esquema i usuari FED

Per a després de la migració, mourem les dades a una nova base de dades que anomenarem DEFBD, on tindrem la nostra base de dades totalment migrada a PostgreSQL.

```
CREATE DATABASE defbd;
```

Creem l'esquema demanat:

```
\connect defbd
CREATE SCHEMA fed;
```

Donem privilegis a l'usuari `migbd` perquè pugui manipular la base de dades sense problemes:

```
GRANT ALL PRIVILEGES ON SCHEMA fed TO migbd;
```

## 4.4 Creació de les taules amb l'usuari MIGBD

Hem d'entrar a la consola de PostgreSQL amb la comanda:

```
psql migbd
```

Ara podem inserir les taules, equivalents a les inserides a la secció 3.2 i que podem consultar a la secció 6.

## 4.5 Migració a PostgreSQL utilitzant `pg_loader`

### 4.5.1 Creació de l'usuari MIGRATOR

Se'ns demana crear un nou usuari, amb els permisos corresponents, encarregat de la migració de la base de dades que prové del sistema de gestió de base de dades MySQL.

```
CREATE USER migrator WITH PASSWORD 'admin';
GRANT INSERT ON ALL TABLES IN SCHEMA sgbd TO migrator;
GRANT CREATE, ALTER ON SCHEMA sgbd TO migbd;
```

### 4.5.2 Instal·lació i configuració de pg_loader

Instal·lació de pg_loader, concretament la versió 3.6.9 per a macOS.

```
brew install pgloader
```

A continuació l'arxiu de configuració .load:

```
LOAD DATABASE
    FROM mysql://MIGBD:adminlocalhost/DEFBD
    INTO postgresql://migrator:adminlocalhost/migbd

BEFORE LOAD DO
$$ ALTER DATABASE migbd SET search_path TO sgbd, public; $$;
```

### 4.5.3 Migració des de terminal

Per a dur a terme la correcta migració, s'haurà de tenir tres terminals.

1. **Terminal MySQL**:

   En aquesta terminal tindrem executat els serveis de MySQL i la sessió oberta amb l'usuari corresponent.

2. **Terminal PostgreSQL**:

   En aquesta terminal tindrem executat els serveis de PostgreSQL i la sessió oberta amb l'usuari corresponent.

3. **Terminal pg_loader**:

   En aquesta terminal haurem d'executar la comanda:

   ```
   pgloader migracio.load
   ```

   Des del directori on tinguem aquest document emmagatzemat.

Finalment tindríem la migració completa satisfactòriament.

## 4.6 Migració de dades de MIGBD a DEFBD

Prèviament, haurem creat les respectives taules a DEFBD iguals que les implementades a MIGBD, tal com podem observar a la secció 6.

```
INSERT INTO defbd.fed.PERSONA (ideper, nom, co1, co2, mai, dir, ciu, pai)
SELECT columnas FROM migbd.sgbd.PERSONA;

INSERT INTO defbd.fed.APODERAT (ideapo)
SELECT columnas FROM migbd.sgbd.APODERAT;

INSERT INTO defbd.fed.TORERO (idetor, ideapo)
SELECT columnas FROM migbd.sgbd.TORERO;

INSERT INTO defbd.fed.PLAÇA_DE_BOUS (nompla, anypla, locpla, tippla, ciupla,
↪  paipla, estpla, muspla)
SELECT columnas FROM migbd.sgbd.PLAÇA_DE_BOUS;

INSERT INTO defbd.fed.ACTUACIO (idetor, datcor, nompla)
SELECT columnas FROM migbd.sgbd.ACTUACIO;

INSERT INTO defbd.fed.ESDEVENIMENT (fircor, datcor, nompla, nombou, anybou,
↪  pesbou, cifram, nomram)
SELECT columnas FROM migbd.sgbd.ESDEVENIMENT;
```

## 4.7 Sentències SQL

### 4.7.1 Consulta I: Extreure la mitjana de seients de les places de bous per pais

```
1    SELECT paipla, AVG(locpla)
2    FROM PLAÇA_DE_BOUS
3    GROUP BY paipla;
```

El resultat ens dona les següents mitjanes:

| paipla | AVG(locpla) |
|---|---|
| Ecuador | 5234 |
| España | 4218 |
| México | 6585 |
| Francia | 4000 |
| Portugal | 6000 |
| Colombia | 4388 |
| Marruecos | 5500 |
| Venezuela | 8200 |
| Sahara Occidental | 4000 |
| Perú | 6273 |
| Angola | 1000 |
| Uruguay | 10000 |
| Mozambique | 8000 |
| Argelia | 4600 |
| Costa Rica | 5250 |

Cal mencionar que s'ha arrodonit a la baixa, ja que no poden haver-hi decimals quan parlem de seients.

### 4.7.2 Consulta II: Extreure quants seients disponibles hi ha a totes les places de bous de Portugal

```
1    SELECT SUM(locpla)
2    FROM PLAÇA_DE_BOUS
3    WHERE paipla = 'Portugal';
```

El resultat dona que trobem **30.000** seients entre totes les places de bous de Portugal.

# 5. *Conclusions*

Aquest treball ha evidenciat la interacció entre la teoria i la pràctica en el domini de la gestió de bases de dades, cosa que conjuntament amb l'enfrontament amb desafiaments reals i la resolució de problemes inesperats han afavorit una comprensió profunda dels sistemes de gestió de bases de dades. La necessitat d'adaptabilitat i un enfocament analític per superar obstacles s'ha convertit en una pedra angular del procés d'aprenentatge.

Durant el desenvolupament del projecte, s'han identificat i superat múltiples reptes, des de qüestions de normalització de dades fins a l'optimització de consultes SQL. La implementació d'un model de dades en un escenari real ha proporcionat una bona oportunitat per aplicar els coneixements adquirits i ha demostrat ser una experiència enriquidora.

En resum, els objectius establerts a l'inici han estat assolits amb èxit, confirmant la relació entre la teoria apresa i la seva aplicació pràctica. El treball no només compleix amb les expectatives acadèmiques sinó que també serveix com a testimoni del valor de l'experiència pràctica adquirida. Els resultats obtinguts i les habilitats desenvolupades formen ara part de la meva preparació professional, sent testimonis del meu creixement i evolució en l'àmbit de la informàtica i la gestió de dades.

# *Bibliografia*


[1] Documentació de MySQL, 2023, *Manual de MySQL 8.3* Recuperat de `https://dev.mysql.com/doc/refman/8.3/en/`

[2] Documentació de MySQL, 2023, *Motor d'emmagatzemament InnoDB*, Recuperat de `https://dev.mysql.com/doc/refman/8.0/en/innodb-storage-engine.html`

[3] Documentació de pg_loader, 2023 *Migració de MySQL a PostgreSQL* Recuperat de `https://pgloader.readthedocs.io/en/latest/tutorial/tutorial.html#migrating-from-mysql-to-postgresql`

# 6. *Annex I: Codi SQL*

## 6.1 Codi de MySQL

```sql
CREATE TABLE PERSONA (
    ideper CHAR(10) NOT NULL,
    nom VARCHAR(255),
    co1 VARCHAR(255),
    co2 VARCHAR(255),
    mai VARCHAR(255),
    dir VARCHAR(255),
    ciu VARCHAR(255),
    pai VARCHAR(255),
    PRIMARY KEY (ideper)
) TABLESPACE tb1;

CREATE TABLE APODERAT (
    ideapo CHAR(10) NOT NULL,
    PRIMARY KEY (ideapo),
    FOREIGN KEY (ideapo) REFERENCES PERSONA(ideper)
) TABLESPACE tb1;

CREATE TABLE TORERO (
    idetor CHAR(10) NOT NULL,
    ideapo CHAR(10) NOT NULL,
    PRIMARY KEY (idetor),
    FOREIGN KEY (ideapo) REFERENCES APODERAT(ideapo),
    FOREIGN KEY (idetor) REFERENCES PERSONA(ideper)
) TABLESPACE tb1;

CREATE TABLE PLAÇA_DE_BOUS (
    nompla VARCHAR(255) NOT NULL,
    anypla DATE,
    locpla INT,
    tippla INT,
```

```
        ciupla VARCHAR(255),
        paipla VARCHAR(255),
        estpla TEXT NULL,
        muspla INT,
        PRIMARY KEY (nompla),
        CHECK (tippla IN (0, 1)),
        CHECK (muspla IN (0, 1))
    ) TABLESPACE tb1;

    CREATE TABLE ACTUACIO (
        idact INT NOT NULL AUTO_INCREMENT,
        idetor CHAR(10) NOT NULL,
        datcor DATE,
        nompla VARCHAR(255) NOT NULL,
        PRIMARY KEY (idact),
        FOREIGN KEY (idetor) REFERENCES TORERO(idetor),
        FOREIGN KEY (nompla) REFERENCES PLAÇA_DE_BOUS(nompla)
    ) TABLESPACE tb1;

    CREATE TABLE ESDEVENIMENT (
        idesd INT NOT NULL AUTO_INCREMENT,
        fircor VARCHAR(255),
        datcor DATE,
        nompla VARCHAR(255),
        nombou VARCHAR(255),
        anybou DATE,
        pesbou FLOAT(5, 2),
        cifram INT,
        nomram VARCHAR(255),
        PRIMARY KEY (idesd)
    ) TABLESPACE tb1;
```

## 6.2 Codi de PostgreSQL

```
SET search_path TO SGBD;

CREATE TABLE PERSONA (
    ideper CHAR(10) NOT NULL,
    nom VARCHAR(255),
    co1 VARCHAR(255),
    co2 VARCHAR(255),
    mai VARCHAR(255),
    dir VARCHAR(255),
    ciu VARCHAR(255),
    pai VARCHAR(255),
    PRIMARY KEY (ideper)
);

CREATE TABLE APODERAT (
    ideapo CHAR(10) NOT NULL,
    PRIMARY KEY (ideapo),
    FOREIGN KEY (ideapo) REFERENCES PERSONA(ideper)
);

CREATE TABLE TORERO (
    idetor CHAR(10) NOT NULL,
    ideapo CHAR(10) NOT NULL,
    PRIMARY KEY (idetor),
    FOREIGN KEY (ideapo) REFERENCES APODERAT(ideapo),
    FOREIGN KEY (idetor) REFERENCES PERSONA(ideper)
);

CREATE TABLE PLAÇA_DE_BOUS (
    nompla VARCHAR(255) NOT NULL,
    anypla DATE,
    locpla INT,
    tippla INT,
    ciupla VARCHAR(255),
    paipla VARCHAR(255),
    estpla TEXT NULL,
    muspla INT,
    PRIMARY KEY (nompla),
    CHECK (tippla IN (0, 1)),
    CHECK (muspla IN (0, 1))
);
```

```
42
43  CREATE TABLE ACTUACIO (
44      idact SERIAL,
45      idetor CHAR(10) NOT NULL,
46      datcor DATE,
47      nompla VARCHAR(255) NOT NULL,
48      PRIMARY KEY (idact),
49      FOREIGN KEY (idetor) REFERENCES TORERO(idetor),
50      FOREIGN KEY (nompla) REFERENCES PLAÇA_DE_BOUS(nompla)
51  );
52
53  CREATE TABLE ESDEVENIMENT (
54      idesd SERIAL,
55      fircor VARCHAR(255),
56      datcor DATE,
57      nompla VARCHAR(255),
58      nombou VARCHAR(255),
59      anybou DATE,
60      pesbou NUMERIC(5, 2),
61      cifram INT,
62      nomram VARCHAR(255),
63      PRIMARY KEY (idesd)
64  );
```

# 7. *Annex II: Codi Python*

## 7.1 Script per a la fusió de les taules TOREROS i APODERATS en PERSONAS

```python
import csv

def merge_csv(file1, file2, output_file):
    # Llegir es contingut del primer arxiu CSV
    with open(file1, 'r', encoding='utf-8') as csv_file1:
        reader1 = csv.reader(csv_file1)
        data1 = list(reader1)

        data1 = data1[1:] # Eliminar la capçalera

    # Llegir es contingut del segon arxiu CSV
    with open(file2, 'r', encoding='utf-8') as csv_file2:
        reader2 = csv.reader(csv_file2)
        data2 = list(reader2)

        data2 = data2[1:] # Eliminar la capçalera

    # Fusionar les dades dels dos arxius
    merged_data = data1 + data2

    # Escriure les dades fusionades a un nou arxiu i nombrar les columnes
    with open(output_file, 'w', encoding='utf-8') as csv_file:
        writer = csv.writer(csv_file)
        writer.writerow(['idper', 'nom', 'co1', 'co2', 'mai', 'dir', 'ciu',
        ↪  'pai'])
        writer.writerows(merged_data)

if __name__ == '__main__':
    merge_csv('TOREROS.csv', 'APODERATS.csv', 'PERSONAS.csv')
```

## 7.2 Script per a editar l'arxiu APODERATS

```
import csv

def edit_apoderats(output_file):
    # Llegir el contingut del arxiu CSV
    with open('../APODERATS.csv', 'r', encoding='utf-8') as csv_file:
        reader = csv.reader(csv_file)
        data = list(reader)

        # Copiar la capçalera de la primera columna
        header = [data[0][0]]

        data = data[1:] # Eliminar la capçalera

        # Guardar la primera columna del fitxer
        new_data = list()
        for row in data:
            new_data.append([row[0]])

        # Guardar les dades
        with open(output_file, 'w', encoding='utf-8') as csv_file:
            writer = csv.writer(csv_file)
            writer.writerow(header)
            writer.writerows(new_data)

if __name__ == '__main__':
    edit_apoderats('../APODERATS_edit.csv')
```

## 7.3 Script per a editar l'arxiu TOREROS

```python
import csv

def edit_toreros(output_file):
    # Llegir el contingut del arxiu CSV
    with open('../TOREROS.csv', 'r', encoding='utf-8') as csv_file:
        reader = csv.reader(csv_file)
        data = list(reader)

        # Copiar la capçalera de la primera i última columna
        header = [data[0][0]]
        header.append(data[0][-1])

        data = data[1:] # Eliminar la capçalera

        # Guardar la primera i última columna del fitxer
        new_data = list()
        for row in data:
            new_data.append([row[0]])
            new_data[-1].append(row[-1])

        # Guardar les dades
        with open(output_file, 'w', encoding='utf-8') as csv_file:
            writer = csv.writer(csv_file)
            writer.writerow(header)
            writer.writerows(new_data)

if __name__ == '__main__':
    edit_toreros('../TOREROS_edit.csv')
```