Grau en Enginyeria Informàtica

Sistemes de Gestió de Bases de Dades

Pràctica Final Recuperació

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

6 de febrer 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 necessàries 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 i finalment a Oracle, 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.

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. *Neteja de les dades*

2.1 Conversió dels fitxers del format ODS a CSV

En primer lloc, consultant la documentació de MySQL, ens adonem que aquest sistema de gestió de base de dades no pot importar dades des d'un fitxer ODS de manera directa. És per això que s'han convertit aquests fitxers a un format CSV, que sí que accepta el gestor. Podeu consultar aquests scripts a la secció 9.

2.2 Format de les dades

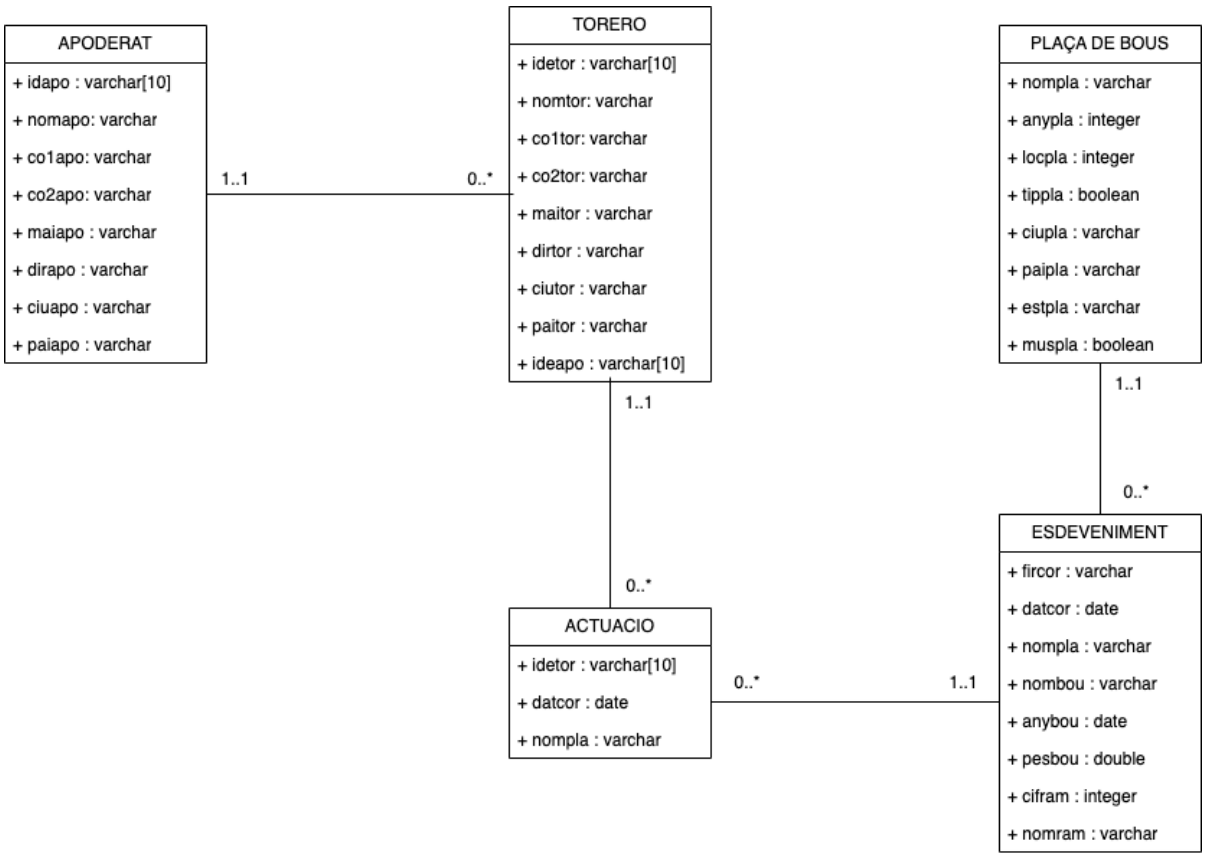
En segon lloc, tenim que diversos arxius contenen elements encapsulats amb dobles cometes i altres que no, columnes totalment buides o inclús formats de dates incorrectes (possiblement a causa de la conversió a CSV). Veure també els scripts a la secció 9

Són aquests els motius de la implementació d'aquesta neteja prèvia i, amb l'objectiu de fer una correcta importació posterior.

3. *Anàlisi i modelat de les dades*

3.1 Model conceptual

Aquest model reflecteix l'estructura de dades dels fitxers d'on s'importaran les dades i que serà l'estructura de la base de dades IMPBD. Aquesta realitzarà la funció de *data lake* (llac de dades en català).



3.1.1 Aclariments i suposicions

Aquest model representa les diferents taules i relacions de la base de dades d'importació d'aquest projecte

En primer lloc, ens trobem les taules `APODERAT` i `TORERO` que tenen una relació `1..1` a `0..*` el que ens suggereix que un apoderat pot tenir diversos toreros assignats (això es dona, per exemple, perquè un apoderat té una escola i, per tant, mentoritza diversos toreros) i un torero única i exclusivament pot tenir un apoderat associat (persona de total confiança). En segon lloc, trobem la relació `TORERO` a `ACTUACIO` amb la mateixa cardinalitat que la relació anterior, on un torero pot participar en diferents actuacions, però una actuació només pot tenir un torero assignat.

Seguidament, ens trobem amb la relació `ACTUACIO` a `ESDEVENIMENT` on també es repeteix la mateixa cardinalitat i bàsicament una actuació pot formar part únicament d'un esdeveniment, però un esdeveniment pot donar lloc a diferents actuacions. Finalment, la relació `PLAÇA DE BOUS` amb `ESDEVENIMENT` que ens indica que una plaça de bous pot albergar diferents esdeveniments, però que un esdeveniment només pot estar associat a una plaça concreta.

D'aquesta forma, aquesta base de dades ens servirà per al ja comentat i mancarà de *constraints* com el de la clau forana, ja que no necessitem establir-lo en el nostre sistema degut a la funcionalitat que durà a terme aquesta base de dades; importar informació per després migrar-la a una altra base de dades normalitzada. Això ho veurem a les següents seccions.

3.2 Model relacional

A continuació les diferents passes per dur a terme la correcta normalització de la nostra base de dades:

3.2.1 Taules

APODERAT(idapo, nomapo, co1apo, co2apo, maiapo, dirapo, ciuapo, paiapo)

TORERO(idetor, nomtor, co1tor, co2tor, maitor, dirtor, ciutor, paitor, ideapo)

ACTUACIO(idetor, datcor, nompla)

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

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

3.2.2 Normalització

PERSONA(nom, cog, mai, dir)

APODERAT(ideapo)

TORERO(idetor, ideapo)

ACTUACIO(idetor, datcor, nompla)

ESDEVENIMENT(fircor, datcor, nompla)

RAMADERIA(cifram, nomram)

BOU(nombou, anybou, pesbou)

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

UBICACIO(ciu, pai)

3.2.3 Elecció de claus primàries

D'aquí en endavant es marcaran les claus primàries en negreta.

PERSONA(**ideper**, nom, cog, mai, dir)

APODERAT(**ideapo**)

TORERO(**idetor**, ideapo)

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

ESDEVENIMENT(**idesd**, fircor, datcor, nompla)

RAMADERIA(**cifram**, nomram)

BOU(**idbou**, nombou, anybou, pesbou)

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

UBICACIO(**ciu, pai**)

3.2.4 Relacional

D'aquí en endavant es marcaran les claus foranes en cursiva.

PERSONA(**ideper**, nom, cog, mai, dir, *ciu, pai*)
{(ciu, pai) FK de la taula UBICACIO}

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, *idesd*)
{idetor FK de la taula TORERO}
{idesd FK de la taula ESDEVENIMENT}

ESDEVENIMENT(**idesd**, fircor, datcor, *nompla*, *idbou*)
{idbou FK de la taula BOU}
{nompla FK de la taula PLAÇA DE BOUS}

RAMADERIA(**cifram**, nomram)

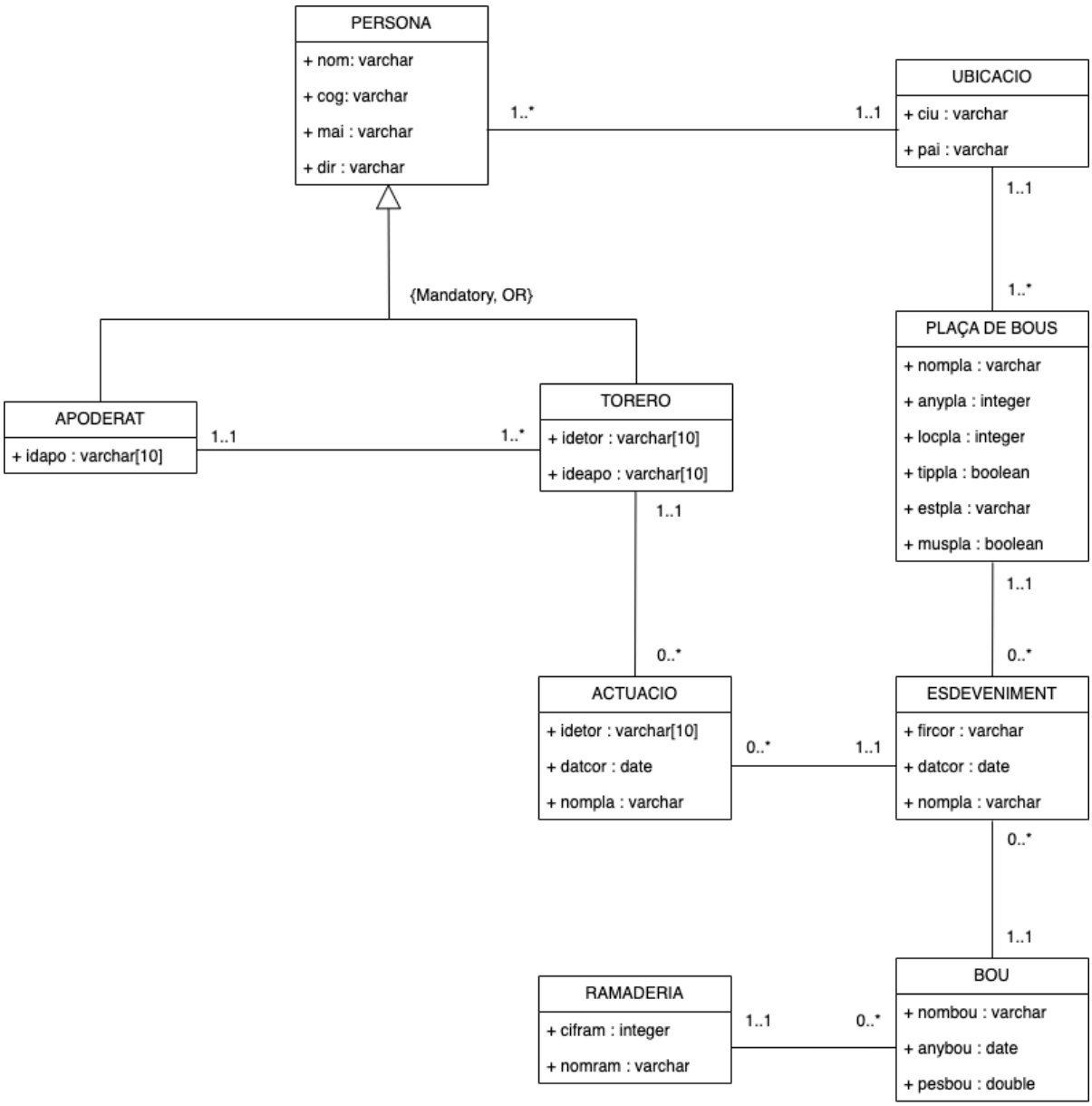
BOU(**idbou**, nombou, anybou, pesbou, *cifram*)
{cifram FK de la taula RAMADERIA}

PLAÇA DE BOUS(**nompla**, anypla, locpla, tippla, *ciupla, paipla*, estpla, muspla)
{(ciupla, paipla) FK de la taula UBICACIO}

UBICACIO(**ciu, pai**)

3.3 Model conceptual normalitzat

A continuació una imatge de com quedaria el model de dades normalitzat de la nostra base de dades:



3.3.1 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 `ESDEVENIMENT`, on una actuació només pot estar assignada a esdeveniment i que un esdeveniment pot tenir definides de zero a moltes actuacions; per poder tenir en compte les noves actuacions que encara no han estat dutes a terme a un esdeveniment. 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. Després la taula `BOU` que representa el bou que participa en l'esdeveniment concret. En aquest cas tindrem una relació `0..*` a `1..1`, ja que a un mateix esdeveniment només participarà un bou, però un bou podria participar en zero (en cas de no haver participat en cap) o més d'un; ja que de vegades pot passar. Després, a la taula `RAMADERIA` ens trobem una relació també `0..*` a `1..1` que reflecteix com la ramaderia pot tenir en criança diversos bous, però un bou només pot formar d'una ramaderia.

D'altra banda, tenim la taula `PLAÇA DE BOUS`, la qual està relacionada amb `ESDEVENIMENT` amb una relació `0..*` a `1..1`,és a dir, que una plaça de bous pot albergar diferents esdeveniments, però un esdeveniment només es pot dur a terme a una plaça. Finalment, tenim la taula `UBICACIÓ` que està relacionada tant amb la taula `PLAÇA DE TOROS` com amb `PERSONA`. Ambdues relacions es componen d'una cardinalitat `1..*` a `1..1`, el que ens indica que una persona o una plaça només podran tenir assignat una única ubicació (país i ciutat), però que una ubicació pot albergar diferents persones com places.

4. *Implementació amb MySQL*

4.1 Creació de l'espai d'emmagatzemament

Per a la creació de l'espai d'emmagatzemament de 5GB i fita màxima de 10GB, haurem d'utilitzar un `TABLESPACE` amb diversos paràmetres.

```
CREATE TABLESPACE tb_1
ADD DATAFILE 'tb_1.ibd'
ENGINE = InnoDB
INITIAL_SIZE = 5G
MAX_SIZE = 10G;
```

Aquest serà creat al directori per defecte que té MySQL a macOS `/usr/local/var/mysql`.

4.2 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 IF NOT EXISTS IMPBD
CHARACTER SET = 'utf8';

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

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

```
CREATE USER 'impbd'@'localhost' IDENTIFIED BY 'admin';
GRANT CREATE, INSERT, UPDATE, DELETE, REFERENCES 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:

```
1    USE IMPBD;
```

Després executarem el script SQL que crearà les respectives taules que va adjunt a aquest treball i està anomenat com `Lluis_Barca_bd_corregudes_bous_mysql.sql`.

4.2.1 Descripció dels *constraints*, *checks* i *triggers* implementats

- `PERSONA`
  - **Constraints**
    - **PRIMARY KEY**: `ideper` serà la clau primària per identificar a totes les persones. Aquesta clau realment serà l'identificador o del torero o de l'apoderat.
    - **NOT NULL**: Pràcticament tots els atributs no poden ser NULL, però `mai` i `dir` sí que ho poden ser, ja que tenim qualque persona que no té aquestes dades.
- `APODERAT`
  - **Constraints**
    - **PRIMARY KEY**: `ideapo` serà la clau primària per identificar a una persona que és apoderat.
    - FOREIGN KEY: `ideapo` serà també la clau forana a la taula `PERSONA`.
    - **NOT NULL**: Aquesta taula només té una columna i es tracta de la clau primària, per tant, no pot ser NULL mai.
- `TORERO`
  - **Constraints**
    - **PRIMARY KEY**: `idetor` serà la clau primària per identificar a una persona que és torero.
    - **FOREIG KEY**:
      - `ideapo` serà la clau forana de la taula `APODERAT`.
      - `idetor` també serà clau forana de la taula `PERSONA`.
    - **NOT NULL**: Totes les columnes seran NOT NULL, ja que són o claus primàries o foranes.

- `ACTUACIO`
  - **Constraints**
    - **PRIMARY KEY**: `idact` serà la clau primària que ens permetrà identificar les diferents actuacions. Aquesta s'autoincrementa cada vegada que s'insereix una actuació.
    - **FOREIGN KEY**:
      - `idetor` serà clau forana a la taula `TORERO`.
      - `idesd` serà clau forana a la taula `ESDEVENIMENT`.
    - **NOT NULL**: Cap columna pot ser NULL, ja que contenen informació necessària.
- `ESDEVENIMENT`
  - **Constraints**
    - **PRIMARY KEY**: `idesd` serà la clau primària que ens permetrà identificar els diferents esdeveniments. Aquesta s'autoincrementa cada vegada que s'insereix un esdeveniment.
    - **FOREIGN KEY**:
      - `idbou` serà la clau forana a la taula `BOU`.
      - `nompla` serà la clau forana a la taula `PLAÇA_DE_BOUS`.
    - **NOT NULL**: Cap columna pot ser NULL, ja que contenen informació necessària.
- `RAMADERIA`
  - **Constraints**
    - **PRIMARY KEY**: `cifram` és el codi que identifica cada ramaderia faria de clau primària; vindria a ser una espècie de NIF.
    - **NOT NULL**: Cap columna pot ser NULL, ja que contenen informació necessària.
- `BOU`
  - **Constraints**
    - **PRIMARY KEY**: `idbou` és el codi que identifica cada bou faria de clau primària i s'incrementa de forma automàtica.
    - **FOREIGN KEY**: `cifram` és la clau forana a la taula `RAMADERIA`.
    - **NOT NULL**: Cap columna pot ser NULL, ja que contenen informació necessària.
- `PLAÇA_DE_BOUS`
  - **Constraints**
    - **PRIMARY KEY**: `nompla` és el nom de la plaça que seria la nostra clau primària, ja que no volem tenir places duplicades.
    - **FOREIGN KEY**: La tupla `ciupla, paipla` és la clau forana a la taula `UBICACIO`.

  - **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.

- `UBICACIO`
  - **Constraints**
    * **PRIMARY KEY**: La dupla `ciu, pai` efectuarien la funció de clau primària.
    * **NOT NULL**: Cap columna pot ser NULL, ja que contenen informació necessària.

4.3 Importació de les dades a IMPBD

En primer lloc, s'ha fet una feina prèvia d'exportació dels fitxers en format ODS al format CSV, amb la finalitat que es puguin importar les dades amb la comanda `LOAD DATA LOCAL INFILE`. Després, per poder inserir dades de forma "automàtica", 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:

- `APODERAT`

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

- TORERO

```
LOAD DATA LOCAL INFILE
    ↪ '/Users/luisbarcap/Desktop/sgbd_recu_final/csv/TOREROS.csv'
INTO TABLE TORERO
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;
```

- PLAÇA DE BOUS

```
LOAD DATA LOCAL INFILE
    ↪ '/Users/luisbarcap/Desktop/sgbd_recu_final/csv/PLACES_BOUS.csv'
INTO TABLE PLAÇA_DE_BOUS
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;
```

- ESDEVENIMENT

```
LOAD DATA LOCAL INFILE
    ↪ '/Users/luisbarcap/Desktop/sgbd_recu_final/csv/esdeveniments.csv'
INTO TABLE ESDEVENIMENT
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n';
```

- ACTUACIO

```
LOAD DATA LOCAL INFILE
    ↪ '/Users/luisbarcap/Desktop/sgbd_recu_final/csv/ACTUACIONS.csv'
INTO TABLE ACTUACIO
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;
```

4.4 Creació de la base de dades DEFBD i l'usuari MIGBD

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

```
CREATE DATABASE IF NOT EXISTS DEFBD
CHARACTER SET = 'utf8';
```

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 codi SQL que adjunt a la secció 8.

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

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:

```
-- Taula UBICACIO
INSERT INTO DEFBD.UBICACIO (ciu, pai)
SELECT DISTINCT ciuapo, paiapo FROM IMPBD.APODERAT
UNION
SELECT DISTINCT ciutor, paitor FROM IMPBD.TORERO
UNION
SELECT DISTINCT ciupla, paipla FROM IMPBD.PLAÇA_DE_BOUS;

-- Taula PERSONA
DELIMITER $$

CREATE PROCEDURE migracio_persona()
BEGIN
    DECLARE finished INTEGER DEFAULT 0;
    DECLARE newID VARCHAR(10);
```

```
        DECLARE exist INT;
        DECLARE baseID INT DEFAULT 100000000;
        DECLARE suffix CHAR(1);
        DECLARE nom VARCHAR(255);
        DECLARE cog VARCHAR(255);
        DECLARE mai VARCHAR(255);
        DECLARE dir VARCHAR(255);
        DECLARE ciu VARCHAR(255);
        DECLARE pai VARCHAR(255);
        DECLARE cur CURSOR FOR
            SELECT nomapo, CONCAT_WS(' ', co1apo, co2apo) AS cog, maiapo,
            ↪  dirapo, ciuapo, paiapo FROM IMPBD.APODERAT
            UNION ALL
            SELECT nomtor, CONCAT_WS(' ', co1tor, co2tor) AS cog, maitor,
            ↪  dirtor, ciutor, paitor FROM IMPBD.TORERO;
        DECLARE CONTINUE HANDLER FOR NOT FOUND SET finished = 1;

        OPEN cur;

        read_loop: LOOP
            FETCH cur INTO nom, cog, mai, dir, ciu, pai;
            IF finished = 1 THEN
                LEAVE read_loop;
            END IF;

            SET exist = 1;
            WHILE exist = 1 DO
                SET baseID = baseID + 1;
                SET suffix = CHAR(65 + FLOOR(RAND() * 26));
                SET newID = CONCAT(CAST(baseID AS CHAR(9)), suffix);

                SELECT COUNT(*) INTO exist FROM DEFBD.PERSONA WHERE ideper =
                ↪  newID;

                IF exist = 0 THEN
                    INSERT INTO DEFBD.PERSONA (ideper, nom, cog, mai, dir, ciu,
                    ↪  pai)
                    VALUES (newID, nom, cog, mai, dir, ciu, pai);
                    SET exist = 0;
                END IF;
            END WHILE;
        END LOOP read_loop;

        CLOSE cur;
```

```
    END$$

    DELIMITER ;

    -- Taula APODERAT
    DELIMITER $$

    CREATE PROCEDURE migracio_apoderat()
    BEGIN
        DECLARE apodID VARCHAR(10);
        DECLARE personID VARCHAR(10);
        DECLARE nom VARCHAR(255);
        DECLARE co1 VARCHAR(255);
        DECLARE co2 VARCHAR(255);
        DECLARE mai VARCHAR(255);
        DECLARE dir VARCHAR(255);
        DECLARE ciu VARCHAR(255);
        DECLARE pai VARCHAR(255);
        DECLARE finished INT DEFAULT 0;
        DECLARE cur CURSOR FOR SELECT ideapo, nomapo, co1apo, co2apo, maiapo,
        ↪  dirapo, ciuapo, paiapo FROM IMPBD.APODERAT;
        DECLARE CONTINUE HANDLER FOR NOT FOUND SET finished = 1;

        OPEN cur;

        my_loop: LOOP
            FETCH cur INTO apodID, nom, co1, co2, mai, dir, ciu, pai;
            IF finished THEN
                LEAVE my_loop;
            END IF;

            SELECT ideper INTO personID FROM DEFBD.PERSONA
            WHERE nom = nom AND cog = CONCAT_WS(co1, ' ', co2) AND mai = mai AND
            ↪  dir = dir AND ciu = ciu AND pai = pai
            LIMIT 1;

            IF personID IS NOT NULL THEN
                INSERT INTO DEFBD.APODERAT (ideapo) VALUES (personID);
            END IF;
        END LOOP;

        CLOSE cur;
    END$$

```

```
98  DELIMITER ;
99
100 -- Taula TORERO
101 DELIMITER $$
102
103 CREATE PROCEDURE migracio_torero()
104 BEGIN
105     DECLARE torID VARCHAR(10);
106     DECLARE personID VARCHAR(10);
107     DECLARE apoderatID VARCHAR(10);
108     DECLARE nom VARCHAR(255);
109     DECLARE co1 VARCHAR(255);
110     DECLARE co2 VARCHAR(255);
111     DECLARE mai VARCHAR(255);
112     DECLARE dir VARCHAR(255);
113     DECLARE ciu VARCHAR(255);
114     DECLARE pai VARCHAR(255);
115     DECLARE finished INT DEFAULT 0;
116     DECLARE cur CURSOR FOR SELECT idetor, nomtor, co1tor, co2tor, maitor,
        ↪  dirtor, ciutor, paitor, ideapo FROM IMPBD.TORERO;
117     DECLARE CONTINUE HANDLER FOR NOT FOUND SET finished = 1;
118
119     OPEN cur;
120
121     torero_loop: LOOP
122         FETCH cur INTO torID, nom, co1, co2, mai, dir, ciu, pai, apoderatID;
123         IF finished THEN
124             LEAVE torero_loop;
125         END IF;
126
127         SELECT ideper INTO personID FROM DEFBD.PERSONA
128         WHERE nom = nom AND cog = CONCAT_WS(' ', co1, co2) AND mai = mai AND
            ↪  dir = dir AND ciu = ciu AND pai = pai
129         LIMIT 1;
130
131         SELECT ideapo INTO apoderatID FROM DEFBD.APODERAT
132         JOIN DEFBD.PERSONA ON DEFBD.PERSONA.ideper = DEFBD.APODERAT.ideapo
133         JOIN IMPBD.APODERAT ON IMPBD.APODERAT.ideapo = apoderatID
134         WHERE IMPBD.APODERAT.nomapo = nom AND IMPBD.APODERAT.ciuapo = ciu
            ↪  AND IMPBD.APODERAT.paiapo = pai
135         LIMIT 1;
136
137         IF personID IS NOT NULL THEN
138             INSERT INTO DEFBD.TORERO (idetor, ideapo)
```

```
                VALUES (personID, apoderatID);
            END IF;
        END LOOP;

        CLOSE cur;
    END$$

    DELIMITER ;

    -- Taula PLAÇA DE BOUS
    INSERT INTO DEFBD.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;

    -- Taula RAMADERIA
    INSERT INTO DEFBD.RAMADERIA (cifram, nomram)
    SELECT DISTINCT cifram, nomram
    FROM IMPBD.ESDEVENIMENT;

    -- Taula BOU
    INSERT INTO DEFBD.BOU (nombou, anybou, pesbou, cifram)
    SELECT nombou, anybou, pesbou, cifram
    FROM IMPBD.ESDEVENIMENT;

    -- Taula ESDEVENIMENT
    INSERT INTO DEFBD.ESDEVENIMENT (fircor, datcor, nompla, idbou)
    SELECT
        e.fircor,
        e.datcor,
        e.nompla,
        (SELECT b.idbou
         FROM DEFBD.BOU b
         WHERE b.nombou = e.nombou AND b.anybou = e.anybou AND b.pesbou =
         ↪  e.pesbou
         LIMIT 1)
    FROM IMPBD.ESDEVENIMENT e;

    -- Taula ACTUACIO
    INSERT INTO DEFBD.ACTUACIO (idetor, datcor, idesd)
    SELECT
        t.idetor,
        a.datcor,
        e.idesd
```

```
    FROM
        IMPBD.ACTUACIO a
        JOIN DEFBD.TORERO t ON a.idetor = t.idetor
        JOIN DEFBD.ESDEVENIMENT e ON a.datcor = e.datcor AND a.nompla =
        ↪  e.nompla;
```

4.6 Consulta SQL

A continuació s'implementarà la consulta per a la actualitzacio de les dates de naixement dels bous.

4.6.1 Definicio de la consulta

```
UPDATE BOU b
JOIN ESDEVENIMENT e ON b.idbou = e.idbou
SET b.anybou = DATE_SUB(e.datcor, INTERVAL 2.5 YEAR)
WHERE DATE_SUB(e.datcor, INTERVAL 2.5 YEAR) > b.anybou;
```

4.6.2 Millores de rendiment

A continuació podem analitzar el pla d'execució de la consulta per intentar optimizar-la:

```
EXPLAIN UPDATE BOU b
JOIN ESDEVENIMENT e ON b.idbou = e.idbou
SET b.anybou = DATE_SUB(e.datcor, INTERVAL 2.5 YEAR)
WHERE DATE_SUB(e.datcor, INTERVAL 2.5 YEAR) > b.anybou;
```

```
mysql> EXPLAIN UPDATE BOU b
    ->     JOIN ESDEVENIMENT e ON b.idbou = e.idbou
    ->     SET b.anybou = DATE_SUB(e.datcor, INTERVAL 2.5 YEAR)
    ->     WHERE DATE_SUB(e.datcor, INTERVAL 2.5 YEAR) > b.anybou;
+----+-------------+-------+------------+--------+---------------+---------+---------+---------------+------+----------+-------------+
| id | select_type | table | partitions | type   | possible_keys | key     | key_len | ref           | rows | filtered | Extra       |
+----+-------------+-------+------------+--------+---------------+---------+---------+---------------+------+----------+-------------+
|  1 | SIMPLE      | e     | NULL       | ALL    | idbou         | NULL    | NULL    | NULL          |    1 |   100.00 | NULL        |
|  1 | UPDATE      | b     | NULL       | eq_ref | PRIMARY       | PRIMARY | 4       | defbd.e.idbou |    1 |    33.33 | Using where |
+----+-------------+-------+------------+--------+---------------+---------+---------+---------------+------+----------+-------------+
2 rows in set, 1 warning (0.00 sec)
```

En aquest cas afegirem dos índex i una condició extra per millorar el filtratge:

```
1  ALTER TABLE IMPBD.ESDEVENIMENT ADD INDEX idx_datcor (datcor);
2  ALTER TABLE DEFBD.BOU ADD INDEX idx_anybou (anybou);
3
4  EXPLAIN UPDATE BOU b
5  JOIN ESDEVENIMENT e ON b.idbou = e.idbou
6  SET b.anybou = DATE_SUB(e.datcor, INTERVAL 2.5 YEAR)
7  WHERE e.datcor > fecha_inicioAND e.datcor <= fecha_final
8  AND b.anybou < DATE_SUB(e.datcor, INTERVAL 2.5 YEAR);
```

```
mysql> EXPLAIN UPDATE BOU b
    ->     JOIN ESDEVENIMENT e ON b.idbou = e.idbou
    ->     SET b.anybou = DATE_SUB(e.datcor, INTERVAL 2.5 YEAR)
    ->     WHERE e.datcor > @fecha_inicio AND e.datcor <= @fecha_final
    ->     AND b.anybou < DATE_SUB(e.datcor, INTERVAL 2.5 YEAR);
+----+-------------+-------+------------+--------+-------------------+---------+---------+---------------+------+----------+-------------+
| id | select_type | table | partitions | type   | possible_keys     | key     | key_len | ref           | rows | filtered | Extra       |
+----+-------------+-------+------------+--------+-------------------+---------+---------+---------------+------+----------+-------------+
|  1 | SIMPLE      | e     | NULL       | ALL    | idbou             | NULL    | NULL    | NULL          |    1 |   100.00 | Using where |
|  1 | UPDATE      | b     | NULL       | eq_ref | PRIMARY,idx_anybou | PRIMARY | 4       | defbd.e.idbou |    1 |    33.33 | Using where |
+----+-------------+-------+------------+--------+-------------------+---------+---------+---------------+------+----------+-------------+
2 rows in set, 1 warning (0.01 sec)
```

5. *Migració a PostgreSQL*

5.1 Configuració del FDW

5.1.1 Configuració a PostgreSQL

Instal·lar l'extensió (si no està instal·lada):

```
CREATE EXTENSION IF NOT EXISTS postgres_fdw;
```

Crear un servidor FDW que apunti al servidor MySQL:

```
CREATE SERVER mysql_server FOREIGN DATA WRAPPER postgres_fdw OPTIONS (host
↪  'localhost', port '3306', dbname 'defbd');
```

Establir l'usuari i contrasenya per a la connexió:

```
CREATE USER mapping
FOR postgres
SERVER mysql_server
OPTIONS (user 'mysql_user', password 'admin');
```

5.2 Creació de l'espai d'emmagatzemament

Aquest `TABLESPACE` quedarà emmagatzemat a la ruta indicada, dins el directori específic per emmagatzemar aquest tipus d'objectes.

```
CREATE TABLESPACE data
OWNER postgres
LOCATION '/usr/local/var/postgresql@14/pg_tblspc';
```

5.3 Creació de la base de dades FET i l'esquema Temp

Creem la base de dades assignant-li el `TABLESPACE` creat anteriorment:

```
CREATE DATABASE fet
WITH OWNER = postgres
TABLESPACE data;
```

A continuació l'esquema `temp`:

```
CREATE SCHEMA temp AUTHORIZATION postgres;
```

5.4 Gestió d'usuaris i inserció de dades a FET

Per a gestionar aquesta base de dades, crearem un usuari amb uns privilegis específics que podrà crear, inserir i consultar les taules d'aquesta base de dades.

```
CREATE USER ufdw WITH PASSWORD 'admin';

GRANT CREATE, INSERT, SELECT ON ALL TABLES IN SCHEMA temp TO ufdw;
```

5.5 Consultes SQL

5.5.1 Mitjana de pes dels bous per any en què es celebra la seva lidia

```
SELECT AVG(pesbou), EXTRACT(YEAR FROM datcor) AS any_lidia
FROM Temp.bous
GROUP BY any_lidia;
```

5.5.2 Bous s'han lidiat a l'estat espanyol

```
SELECT COUNT(*)
FROM Temp.bous;
```

6. *Migració a Oracle*

Per a la migració a Oracle, es farà servir un contenidor de Docker amb la imatge d'Oracle en la seva versió 19.3.0 Standard 2 amb l'arquitectura *Single Instance*. Per tant, s'ha de tenir instal·lat Docker prèviament. En aquest cas, s'ha instal·lat la versió d'escriptori per a macOS que inclou la interfície gràfica per manejar Docker.

6.1 Configuració del gestor

En primer lloc, haurem de clonar el repositori oficial d'Oracle, on està la imatge desitjada.

```
1    git clone https://github.com/oracle/docker-images.git
```

A continuació baixarem la versió d'Oracle indicada anteriorment, des de la pàgina web oficial, i l'inserirem dins el directori `/19.3.0` de la imatge clonada.

Ara executarem el programa corresponent per construir el contenidor:

```
1    sudo ./buildContainerImage.sh -v 19.3.0 -s
```

Hem de treure la memòria virtual total al meu sistema (macOS):

```
1    sysctl vm.swapusage
```

Que ens retorna el valor de 2048M. Per tant, els valors de:

- SGA = 2048 * 0.5 = 1024 MB
- PGA = 2048 * 0.15 = 307 MB

Per tant, la comanda quedarà tal que així:

```
sudo docker run -d --name oracle_bd \
-p 1521:1521 -p 5500:5500 \
-e ORACLE_SID=FET \
-e ORACLE_PDB=FETPDB1 \
-e INIT_SGA_SIZE=1024 \
-e INIT_PGA_SIZE=307 \
-e ENABLE_ARCHIVING=true \
oracle/database:19.3.0-se2
```

6.2 Creació de l'usuari Utest i gestió dels seus privilegis

A continuació creem l'usuari amb els privilegis corresponents per gestionar la base de dades:

```
CREATE USER utest IDENTIFIED BY admin;
GRANT CREATE SESSION, CREATE TABLE, INSERT, SELECT ON utest.* TO utest;
```

6.3 Migració de dades

Aquesta part, com només necessitem unes quantes taules, ho farem mitjançant l'exportació a arxius CSV de la base de dades de MySQL. Aquests arxius després els importarem a la base de dades d'oracle.

6.3.1 Exportació de dades

A continuació els diferents codis SQL per a cada una de les taules de la base de dades de MySQL per a efectuar la correcta exportacio:

```
SELECT * INTO OUTFILE
↪  '/Users/luisbarcap/Desktop/sgbd_recu_final/csv_oracle/UBICACIO.csv'
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY '\n'
FROM UBICACIO;

SELECT * INTO OUTFILE
↪  '/Users/luisbarcap/Desktop/sgbd_recu_final/csv_oracle/PLAÇA_DE_BOUS.csv'
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"'
```

```
    LINES TERMINATED BY '\n'
    FROM PLAÇA_DE_BOUS;

    SELECT * INTO OUTFILE
    ↪  '/Users/luisbarcap/Desktop/sgbd_recu_final/csv_oracle/ESDEVENIMENT.csv'
    FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"'
    LINES TERMINATED BY '\n'
    FROM ESDEVENIMENT;

    SELECT * INTO OUTFILE
    ↪  '/Users/luisbarcap/Desktop/sgbd_recu_final/csv_oracle/BOU.csv'
    FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"'
    LINES TERMINATED BY '\n'
    FROM BOU;
```

6.3.2 Creació de taules

La creació de taules s'ha dut a terme com és habitual i trobarem el codi implementat a la secció 8.

6.3.3 Importació de dades

Per tant, s'haurà de crear un arxiu per a cada fitxer del qual vulguem importar les dades. A continuació el codi corresponent a cada un dels fitxers per a la seva corresponents taula:

```
    LOAD DATA
    INFILE '/Users/luisbarcap/Desktop/sgbd_recu_final/csv_oracle/UBICACIO.csv'
    INTO TABLE UBICACIO
    FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"'
    (ciu, pai)

    LOAD DATA
    INFILE
    ↪  '/Users/luisbarcap/Desktop/sgbd_recu_final/csv_oracle/PLAÇA_DE_BOUS.csv'
    INTO TABLE PLAÇA_DE_BOUS
    FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"'
    (nompla, anypla, locpla, tippla, ciupla, paipla, estpla, muspla)

    LOAD DATA
    INFILE '/Users/luisbarcap/Desktop/sgbd_recu_final/csv_oracle/BOU.csv'
    INTO TABLE BOU
    FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"'
    (idbou, nombou, anybou, pesbou, cifram)
```

```
18
19      LOAD DATA
20      INFILE
        ↪  '/Users/luisbarcap/Desktop/sgbd_recu_final/csv_oracle/ESDEVENIMENT.csv'
21      INTO TABLE ESDEVENIMENT
22      FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"'
23      (idesd, fircor, datcor, nompla, idbou)
```

A continuació utilitzem SQL Loader que ens permetrà amb la següent comanda, importar cada un dels fitxers:

```
1       sqlldr USERID=utest/admin@oracle_db CONTROL=nom_fitxer.ctl
        ↪  LOG=nom_fitxer.log
```

El `nom_fitxer` fa referència a cada un dels diferents CSV.

6.4 Consulta SQL

6.4.1 Definició de la consulta

Nom i ubicació de la plaça de bous, juntament amb el nombre de bous que s'han torejat al llarg de tota la història en ella.

```
1       SELECT
2           pb.nompla AS "Nom de la Plaça de Bous",
3           u.ciu AS "Ciutat",
4           u.pai AS "País",
5           COUNT(b.idbou) AS "Nombre de Bous Torejats"
6       FROM PLAÇA_DE_BOUS pb
7       JOIN UBICACIO u ON pb.ciu = u.ciu AND pb.pai = u.pai
8       JOIN ESDEVENIMENT e ON pb.nompla = e.nompla
9       JOIN BOU b ON e.idbou = b.idbou
10      GROUP BY
11          pb.nompla, u.ciu, u.pai;
```

7. *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 PostgreSQL 2023, *Manual de PostgreSQL 14* Recuperat de https://www.postgresql.org/docs/14/index.html

[3] Oracle Database container images, 2023, *Installation, configuration, and environment setup* Recuperat de https://github.com/oracle/docker-images/blob/main/OracleDatabase/SingleInstance/README.md

8. *Annex I: Codi SQL*

8.1 Codi de MySQL

Important llegir els aclariments sobre algunes decisions al final del codi

```sql
CREATE TABLE UBICACIO (
    ciu VARCHAR(255) NOT NULL,
    pai VARCHAR(255) NOT NULL,
    PRIMARY KEY (ciu, pai)
);

CREATE TABLE PERSONA (
    ideper VARCHAR(10) NOT NULL,
    nom VARCHAR(255) NOT NULL,
    cog VARCHAR(255) NOT NULL,
    mai VARCHAR(255),
    dir VARCHAR(255),
    ciu VARCHAR(255) NOT NULL,
    pai VARCHAR(255) NOT NULL,
    PRIMARY KEY (ideper),
    FOREIGN KEY (ciu, pai) REFERENCES UBICACIO(ciu, pai)
);

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

CREATE TABLE TORERO (
    idetor VARCHAR(10) NOT NULL,
    ideapo VARCHAR(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 NOT NULL,
    locpla INT NOT NULL,
    tippla BOOLEAN NOT NULL,
    ciupla VARCHAR(255) NOT NULL,
    paipla VARCHAR(255) NOT NULL,
    estpla TEXT,
    muspla BOOLEAN NOT NULL,
    PRIMARY KEY (nompla),
    FOREIGN KEY (ciupla, paipla) REFERENCES UBICACIO(ciu, pai),
    CHECK (tippla IN (True, False)),
    CHECK (muspla IN (True, False))
);

CREATE TABLE RAMADERIA (
    cifram INT NOT NULL,
    nomram VARCHAR(255) NOT NULL,
    PRIMARY KEY (cifram)
);

CREATE TABLE BOU (
    idbou INT NOT NULL AUTO_INCREMENT,
    nombou VARCHAR(255) NOT NULL,
    anybou DATE NOT NULL,
    pesbou NUMERIC(5, 2) NOT NULL,
    cifram INT NOT NULL,
    PRIMARY KEY (idbou),
    FOREIGN KEY (cifram) REFERENCES RAMADERIA(cifram)
);

CREATE TABLE ESDEVENIMENT (
    idesd INT NOT NULL AUTO_INCREMENT,
    fircor VARCHAR(255) NOT NULL,
    datcor DATE NOT NULL,
    nompla VARCHAR(255) NOT NULL,
    idbou INT NOT NULL,
    PRIMARY KEY (idesd),
    FOREIGN KEY (idbou) REFERENCES BOU(idbou),
    FOREIGN KEY (nompla) REFERENCES PLAÇA_DE_BOUS(nompla)
);

```

```sql
CREATE TABLE ACTUACIO (
    idact INT NOT NULL AUTO_INCREMENT,
    idetor VARCHAR(10) NOT NULL,
    datcor DATE NOT NULL,
    idesd INT NOT NULL,
    PRIMARY KEY (idact),
    FOREIGN KEY (idetor) REFERENCES TORERO(idetor),
    FOREIGN KEY (idesd) REFERENCES ESDEVENIMENT(idesd)
);
```

8.2 Aclariments sobre el codi SQL

La majoria de les columnes amb el tipus de dades `VARCHAR` tenen una longitud màxima de 255 bytes. Això és així perquè aquest tipus d'estructura de dades a MySQL utilitzen la memòria per emmagatzemar les dades en funció del que hagin d'emmagatzemar. Però, utilitzen un byte extra per emmagatzemar la longitud màxima, es dona el cas que 255 és el màxim per a 1 byte (segons la documentació oficial) i, per tant, tindria el mateix impacte un `VARCHAR(20)` que un de `VARCHAR(255)`. D'aquesta forma, en les columnes on no s'especifica una llargària màxima, oferim un rang bastant ampli per si un cas.

8.3 Codi d'Oracle

```
CREATE TABLE UBICACIO (
    ciu VARCHAR2(255) NOT NULL,
    pai VARCHAR2(255) NOT NULL,
    CONSTRAINT pk_ubicacio PRIMARY KEY (ciu, pai)
);

CREATE TABLE PLAÇA_DE_BOUS (
    nompla VARCHAR2(255) NOT NULL,
    anypla DATE NOT NULL,
    locpla NUMBER NOT NULL,
    tippla NUMBER(1) NOT NULL,
    ciupla VARCHAR2(255) NOT NULL,
    paipla VARCHAR2(255) NOT NULL,
    estpla CLOB,
    muspla NUMBER(1) NOT NULL,
    CONSTRAINT pk_plaça_de_bous PRIMARY KEY (nompla),
    CONSTRAINT fk_plaça_de_bous_ubicacio FOREIGN KEY (ciupla, paipla)
    ↪ REFERENCES UBICACIO(ciu, pai),
    CONSTRAINT ck_plaça_de_bous_tippla CHECK (tippla IN (0, 1)),
    CONSTRAINT ck_plaça_de_bous_muspla CHECK (muspla IN (0, 1))
);

CREATE TABLE BOU (
    idbou NUMBER NOT NULL,
    nombou VARCHAR2(255) NOT NULL,
    anybou DATE NOT NULL,
    pesbou NUMBER(7, 2) NOT NULL,
    cifram NUMBER NOT NULL,
    CONSTRAINT pk_bou PRIMARY KEY (idbou),
    CONSTRAINT fk_bou_ramaderia FOREIGN KEY (cifram) REFERENCES
    ↪ RAMADERIA(cifram)
);

CREATE TABLE ESDEVENIMENT (
    idesd NUMBER NOT NULL,
    fircor VARCHAR2(255) NOT NULL,
    datcor DATE NOT NULL,
    nompla VARCHAR2(255) NOT NULL,
    idbou NUMBER NOT NULL,
    CONSTRAINT pk_esdeveniment PRIMARY KEY (idesd),
    CONSTRAINT fk_esdeveniment_bou FOREIGN KEY (idbou) REFERENCES
    ↪ BOU(idbou),
```

```
40          CONSTRAINT fk_esdeveniment_plaça_de_bous FOREIGN KEY (nompla) REFERENCES
            ↪  PLAÇA_DE_BOUS(nompla)
41      );
```

8.4 Aclariments sobre el codi SQL

Només s'han implementat les taules necessàries per fer la consulta demanada, tal com s'especifica als requisits del treball.

Cal afegir que, a Oracle per exemple, no existeixen els valors `AUTO_INCREMENT` i, per tant, s'haurien de crear seqüències i posteriorment *triggers* que fessin aquesta funció. Però per temes de simplicitat, no s'ha dut a terme perquè no s'utilitzarien.

9. *Annex II: Scripts de neteja*

- ACTUACIONS

```
import pandas as pd

def convert_ods_csv(input_file, output_file):
    # Llegim el fitxer ODS
    data = pd.read_excel(input_file, engine="odf")

    # Salvem el fitxer com a CSV
    data.to_csv(output_file, index=False)

    # Eliminem les columnes que no ens interessen
    data = pd.read_csv(output_file)
    data["DATCOR"] = data["DATCOR"].str[:10] # Només ens quedam amb la data
    data.to_csv(output_file, index=False)

if __name__ == "__main__":
    convert_ods_csv("./ACTUACIONS.ods", "./csv/ACTUACIONS.csv")
```

- APODERATS

```
    import pandas as pd

    def convert_ods_csv(input_file, output_file):
        # Llegim el fitxer ODS
        data = pd.read_excel(input_file, engine="odf")

        # Salvem el fitxer com a CSV
        data.to_csv(output_file, index=False)

        # Eliminem les columnes que no ens interessen
        data = pd.read_csv(output_file)
```

```
            data = data.iloc[:, :-2]

            # Eliminem les cometes dobles i els espais en blanc de la columna DIRAPO
            data["DIRAPO"] = data["DIRAPO"].str.strip().str.replace('"', '')
            data.to_csv(output_file, index=False)

        if __name__ == "__main__":
            convert_ods_csv("./APODERATS.ods", "./csv/APODERATS.csv")
```

- PLACES BOUS

```
        import pandas as pd

        def convert_ods_csv(input_file, output_file):
            # Llegim el fitxer ODS
            data = pd.read_excel(input_file, engine="odf")

            # Salvem el fitxer com a CSV
            data.to_csv(output_file, index=False)

            # Eliminem l'hora de la columna ANYPLA
            data = pd.read_csv(output_file)
            data["ANYPLA"] = data["ANYPLA"].str[:10] # Mantenim només la data
            data.to_csv(output_file, index=False)

        if __name__ == "__main__":
            convert_ods_csv("./PLACES_BOUS.ods", "./csv/PLACES_BOUS.csv")
```

- TORERO

```
        import pandas as pd

        def convert_ods_csv(input_file, output_file):
            # Llegim el fitxer ODS
            data = pd.read_excel(input_file, engine="odf")

            # Salvem el fitxer com a CSV
            data.to_csv(output_file, index=False)
```

```
9
10              # Eliminem les dues últimes columnes buides
11              data = pd.read_csv(output_file)
12              data = data.iloc[:, :-2]
13
14              # Eliiminam les cometes dobles i els espais en blanc de la
                ↪  columna DIRTOR
15              data["DIRTOR"] = data["DIRTOR"].str.strip().str.replace('"',
                ↪  '')
16              data.to_csv(output_file, index=False)
17
18          if __name__ == "__main__":
19              convert_ods_csv("./TOREROS.ods", "./csv/TOREROS.csv")
```