Universitat de les Illes Balears
Grau d'Enginyeria Informàtica
21751 - Sistemes de Gestió de Bases de Dades

Pràctica Final

Michele Vincenzo Gentile

Course 2024-2025

Continguts

Referència creuada de l'activitat

Capítol	Pregunta	Secció corresponent	Fitxer
Design	-	*Capítols 1 i 2*	-
MySQL	A	*Section 3.1.1*	01_raw_import_setup.sql
	B	*Section 3.1.2*	
	C	*Section 3.1.3*	
	D	*Section 3.1.4*	
	E	*Section 3.2.1*	02_gestmat_setup.sql
	F	*Section 3.2.2*	
	G	*Section 3.2.3*	
	H	*Section 3.2.4*	03_raw_to_gestmat.sql
	I	*Section 3.3*	04_questions.sql
	J	*Section 3.4*	05_optimizations.sql
PostgreSQL	1	*Section 4.1.1*	06_gestmat_setup.sql
	2	*Section 4.1.2*	
	3	*Section 4.1.3*	
	4	*Section 4.2*	-
	5	*Section 4.3*	08_prod_setup.sql
	6	*Section 4.3.1*	
	7	*Section 4.3.2*	
	8	*Section 4.3.3*	09_old_to_prod.sql
	9	*Section 4.3.3*	
	10	*Section 4.4*	10_questions.sql
	12	*Section 4.5*	11_optimizations.sql
Comparació	11	*Capítol 5*	-

Capítol 1

Anàlisi del disseny

1.1 Anàlisi de fitxers (.csv)

Llistat_CENTRES.csv (Centres)

1. Clau primària: 'CODI'
2. Atributs:
    - 'CODI' (Codi) - Identificador únic per a cada centre
    - 'DENOMINACIÓ GENÈRICA' (Denominació genèrica)
    - 'NOM' (Nom)
    - 'CORREU ELECTRÒNIC 1' (Correu electrònic 1)
    - 'CORREU ELECTRÒNIC 2' (Correu electrònic 2)
    - 'PÀGINA WEB' (Pàgina web)
    - 'TITULAR' (Titular)
    - 'NIF' (NIF)
    - 'LOCALITAT' (Localitat)
    - 'MUNICIPI' (Municipi)
    - 'ADREÇA' (Adreça)
    - 'CP' (Codi postal)
    - 'ILLA' (Illa)
    - 'TELEF1' (Telèfon 1)

estudiants.csv (Estudiants)

1. Clau primària: 'dni'
2. Atributs:
    - 'dni' - Número d'identificació nacional
    - 'nom' (Nom)
    - 'primer_cognom' (Primer cognom)
    - 'segon_cognom' (Segon cognom)
    - 'correu_electronic' (Correu electrònic)
    - 'codi_postal_i_districte' (Codi postal i districte)
    - 'comunitat_autonoma' (Comunitat autònoma)
    - 'municipi' (Municipi)

matricules_202425.csv (Matrícules)

1. Clau primària: Composta
2. Atributs:
    - ‘dni‘ - DNI de l’estudiant
    - ‘tipus_ensenyament‘ (Tipus d’ensenyament)
    - ‘modalitat‘ (Modalitat)
    - ‘curs‘ (Curs)
    - ‘nom_assignatura‘ (Nom de l’assignatura)
    - ‘grup_de_classe‘ (Grup de classe)
    - ‘codi_centre‘ (Codi del centre)

1.2 Definició de relacions

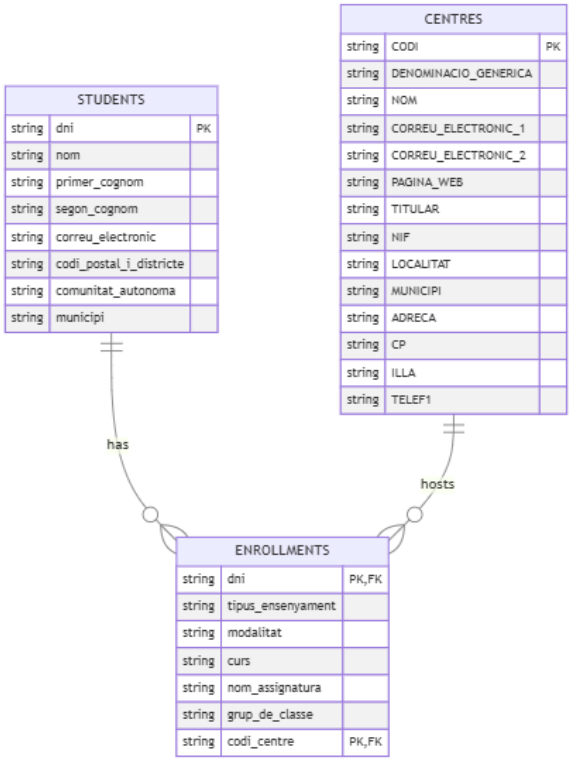
1. Relació Estudiant-Matrícula
    - Un a molts: un estudiant pot tenir múltiples matrícules
    - Obligatòria per a Matrícula (no pot existir sense un estudiant)
    - Opcional per a Estudiant (pot existir sense matrícules)
2. Relació Centre-Matrícula
    - Un a molts: un centre pot tenir múltiples matrícules
    - Obligatòria per a Matrícula (no pot existir sense un centre)
    - Opcional per a Centre (pot existir sense matrícules)

1.3 Diagrama ER Actual

Codi Mermaid per generar el diagrama ER:

```
erDiagram
    CENTRES {
        string CODI PK
        string DENOMINACIO_GENERICA
        string NOM
        string CORREU_ELECTRONIC_1
        string CORREU_ELECTRONIC_2
        string PAGINA_WEB
        string TITULAR
        string NIF
        string LOCALITAT
        string MUNICIPI
        string ADRECA
        string CP
        string ILLA
        string TELEF1
    }

    STUDENTS {
        string dni PK
```

```
    string nom
    string primer_cognom
    string segon_cognom
    string correu_electronic
    string codi_postal_i_districte
    string comunitat_autonoma
    string municipi
}

ENROLLMENTS {
    string dni PK, FK
    string tipus_ensenyament PK
    string modalitat PK
    string curs PK
    string nom_assignatura PK
    string grup_de_classe PK
    string codi_centre PK, FK
}

STUDENTS ||--o{ ENROLLMENTS : "té"
CENTRES ||--o{ ENROLLMENTS : "acull"
```



Capítol 2

Normalització de Bases de Dades

2.1 Problemes de normalització actuals

1. A la taula CENTRES:
   (a) Les dades de localització (‘MUNICIPI‘, ‘CP‘, ‘ILLA‘) representen una possible dependència multivaluada
   (b) En alguns casos un ‘TITULAR‘ és propietari de múltiples centres
   (c) Diferents centres tenen la mateixa ‘DENOMINACIO GENERICA‘
2. A la taula STUDENTS:
   (a) Les dades de localització (‘comunitat_autonoma‘, ‘municipi‘, ‘codi_postal_i_districte‘) representen una possible dependència multivaluada
3. A la taula ENROLLMENTS:
   (a) La informació acadèmica (‘tipus_ensenyament‘, ‘modalitat‘, ‘curs‘, ‘nom_assignatura‘) mostra possibles dependències parcials

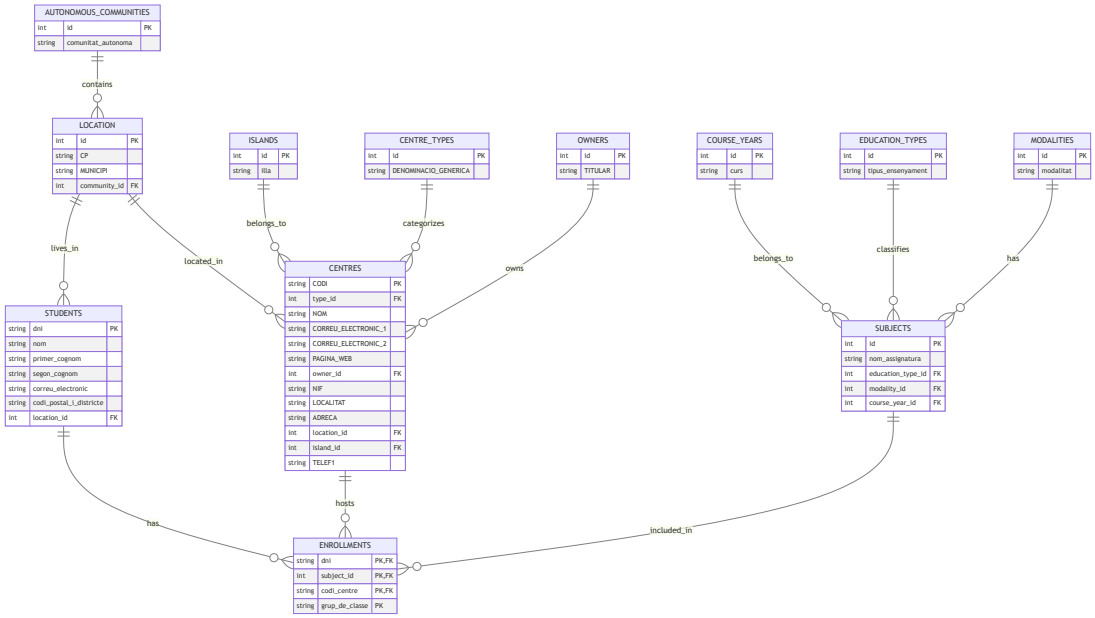
2.2 Diagrama ER Normalitzat

Codi Mermaid per generar el diagrama ER:

```
erDiagram
    AUTONOMOUS_COMMUNITIES {
        int id PK
        string comunitat_autonoma
    }

    LOCATION {
        int id PK
        string CP
        string MUNICIPI
        int community_id FK
    }

    ISLANDS {
        int id PK
        string illa
    }
```

```
COURSE_YEARS {
    int id PK
    string curs
}

EDUCATION_TYPES {
    int id PK
    string tipus_ensenyament
}

MODALITIES {
    int id PK
    string modalitat
}

CENTRE_TYPES {
    int id PK
    string DENOMINACIO_GENERICA
}

OWNERS {
    int id PK
    string TITULAR
}

CENTRES {
    string CODI PK
    int type_id FK
    string NOM
    string CORREU_ELECTRONIC_1
    string CORREU_ELECTRONIC_2
    string PAGINA_WEB
    int owner_id FK
    string NIF
    string LOCALITAT
    string ADRECA
    int location_id FK
    int island_id FK
    string TELEF1
}

STUDENTS {
    string dni PK
    string nom
    string primer_cognom
    string segon_cognom
    string correu_electronic
    string codi_postal_i_districte
    int location_id FK
}

SUBJECTS {
    int id PK
```

```
    string nom_assignatura
    int education_type_id FK
    int modality_id FK
    int course_year_id FK
}

ENROLLMENTS {
    string dni PK, FK
    int subject_id PK, FK
    string codi_centre PK, FK
    string grup_de_classe PK
}

AUTONOMOUS_COMMUNITIES ||--o{ LOCATION : "conté"
LOCATION ||--o{ CENTRES : "ubicat_a"
LOCATION ||--o{ STUDENTS : "viu_a"
ISLANDS ||--o{ CENTRES : "pertany_a"
CENTRE_TYPES ||--o{ CENTRES : "categoritza"
OWNERS ||--o{ CENTRES : "és_propietat_de"
COURSE_YEARS ||--o{ SUBJECTS : "correspon_a"
EDUCATION_TYPES ||--o{ SUBJECTS : "classifica"
MODALITIES ||--o{ SUBJECTS : "té"
STUDENTS ||--o{ ENROLLMENTS : "té"
CENTRES ||--o{ ENROLLMENTS : "acull"
SUBJECTS ||--o{ ENROLLMENTS : "inclou"
```


AUTONOMOUS_COMMUNITIES
int id PK
string comunitat_autonoma
contains
LOCATION
int id PK
string CP
string MUNICIPI
int community_id FK
ISLANDS
int id PK
string illa
CENTRE_TYPES
int id PK
string DENOMINACIO_GENERICA
OWNERS
int id PK
string TITULAR
COURSE_YEARS
int id PK
string curs
EDUCATION_TYPES
int id PK
string tipus_ensenyament
MODALITIES
int id PK
string modalitat
belongs_to
categorizes
lives_in
located_in
owns
belongs_to
classifies
has
CENTRES
string CODI PK
int type_id FK
string NOM
string CORREU_ELECTRONIC_1
string CORREU_ELECTRONIC_2
string PAGINA_WEB
int owner_id FK
string NIF
string LOCALITAT
string ADRECA
int location_id FK
int island_id FK
string TELEF1
STUDENTS
string dni PK
string nom
string primer_cognom
string segon_cognom
string correu_electronic
string codi_postal_i_districte
int location_id FK
SUBJECTS
int id PK
string nom_assignatura
int education_type_id FK
int modality_id FK
int course_year_id FK
hosts
has
included_in
ENROLLMENTS
string dni PK,FK
int subject_id PK,FK
string codi_centre PK,FK
string grup_de_classe PK

Capítol 3

Implementació de MySQL

3.1 RAW_IMPORT

3.1.1 Crear DB

A MySQL no és possible establir un espai de taules per defecte. Totes les taules creades en RAW_IMPORT hauran d'especificar l'espai de taules explícitament.

```
-- Crear l'espai de taules DADES_TEMPORALS
CREATE TABLESPACE DADES_TEMPORALS
    ADD DATAFILE 'DADES_TEMPORALS.ibd'
    ENGINE = InnoDB;

-- Crear la base de dades RAW_IMPORT
CREATE DATABASE RAW_IMPORT
    DEFAULT CHARACTER SET utf8mb4
    DEFAULT COLLATE utf8mb4_unicode_ci;
```

3.1.2 Crear l'usuari IMPORTADOR_1

```
-- Crear l'usuari IMPORTADOR_1
CREATE USER 'IMPORTADOR_1'@'localhost' IDENTIFIED BY 'password';

-- Atorgar privilegis per crear i inserir dades a la base de dades RAW_IMPORT
GRANT CREATE, INSERT ON RAW_IMPORT.* TO 'IMPORTADOR_1'@'localhost';

-- Atorgar privilegis per a arxius per importar des dels fitxers csv.
GRANT FILE ON *.* TO 'IMPORTADOR_1'@'localhost';

-- Aplicar els privilegis
FLUSH PRIVILEGES;
```

3.1.3 Crear taules

Primer hem d'accedir com a l'usuari nou que hem creat:

```
mysql> exit
C:\Program Files\MySQL\MySQL Server 8.0\bin>mysql -u IMPORTADOR_1 -p
```

Després podem procedir:

```sql
USE RAW_IMPORT;

CREATE TABLE centers (
    CODI VARCHAR(20) PRIMARY KEY,
    DENOMINACIO_GENERICA VARCHAR(100),
    NOM VARCHAR(256),
    CORREU_ELECTRONIC_1 VARCHAR(100),
    CORREU_ELECTRONIC_2 VARCHAR(100),
    PAGINA_WEB VARCHAR(255),
    TITULAR VARCHAR(100),
    NIF VARCHAR(20),
    LOCALITAT VARCHAR(100),
    MUNICIPI VARCHAR(100),
    ADRECA VARCHAR(255),
    CP VARCHAR(10),
    ILLA VARCHAR(100),
    TELEF1 VARCHAR(20)
) TABLESPACE DADES_TEMPORALS;

CREATE TABLE students (
    dni VARCHAR(20) PRIMARY KEY,
    nom VARCHAR(100),
    primer_cognom VARCHAR(100),
    segon_cognom VARCHAR(100),
    correu_electronic VARCHAR(100),
    codi_postal_i_districte VARCHAR(50),
    comunitat_autonoma VARCHAR(100),
    municipi VARCHAR(100)
) TABLESPACE DADES_TEMPORALS;

CREATE TABLE enrollments (
    dni VARCHAR(20),
    tipus_ensenyament VARCHAR(50),
    modalitat VARCHAR(50),
    curs VARCHAR(50),
    nom_assignatura VARCHAR(100),
    grup_de_classe VARCHAR(50),
    codi_centre VARCHAR(20),
    PRIMARY KEY (dni, tipus_ensenyament, modalitat, curs, nom_assignatura,
        grup_de_classe, codi_centre)
) TABLESPACE DADES_TEMPORALS;
```

3.1.4 Importar dades

```sql
LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/centres.csv'
INTO TABLE centers
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;

LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/estudiants.csv'
INTO TABLE students
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;
```

```
14
15 LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/matricules_202425.
       csv'
16 IGNORE INTO TABLE enrollments -- Alguns duplicats existeixen en aquest csv, amb l'
       opci  IGNORE els ignorem.
17 FIELDS TERMINATED BY ','
18 ENCLOSED BY '"'
19 LINES TERMINATED BY '\n'
20 IGNORE 1 ROWS;
```

3.2 GESTMAT

3.2.1 Crear DB

A MySQL no és possible establir un espai de taules per defecte. Totes les taules creades en GESTMAT hauran d'especificar l'espai de taules explícitament.

```
1 -- Crear l'espai de taules PANDORA
2 CREATE TABLESPACE PANDORA
3     ADD DATAFILE 'PANDORA.ibd'
4     ENGINE = InnoDB;
5
6 -- Crear la base de dades GESTMAT
7 CREATE DATABASE GESTMAT
8     DEFAULT CHARACTER SET utf8mb4
9     DEFAULT COLLATE utf8mb4_unicode_ci;
```

3.2.2 Crear l'usuari TRANSFORMADOR_1

```
1 -- Crear l'usuari TRANSFORMADOR_1
2 CREATE USER 'TRANSFORMADOR_1'@'localhost' IDENTIFIED BY 'password';
3
4 -- Atorgar privilegis per crear i inserir dades a la base de dades GESTMAT
5 GRANT CREATE, INSERT, SELECT, REFERENCES, INDEX, CREATE VIEW ON GESTMAT.* TO '
      TRANSFORMADOR_1'@'localhost';
6 GRANT SELECT ON RAW_IMPORT.* TO 'TRANSFORMADOR_1'@'localhost';
7
8 -- Aplicar els privilegis
9 FLUSH PRIVILEGES;
```

3.2.3 Crear taules

Primer hem d'accedir com a l'usuari nou que hem creat:

```
1 mysql> exit
2 C:\Program Files\MySQL\MySQL Server 8.0\bin>mysql -u TRANSFORMADOR_1 -p
```

Després podem procedir:

```
1 USE GESTMAT;
2
3 CREATE TABLE AUTONOMOUS_COMMUNITIES (
4     id INT AUTO_INCREMENT PRIMARY KEY,
5     comunitat_autonoma VARCHAR(104) NOT NULL UNIQUE
6 ) TABLESPACE PANDORA;
7
```

```sql
CREATE TABLE LOCATION (
    id INT AUTO_INCREMENT PRIMARY KEY,
    CP VARCHAR(8) NOT NULL,
    MUNICIPI VARCHAR(104) NOT NULL,
    community_id INT NOT NULL,
    FOREIGN KEY (community_id) REFERENCES AUTONOMOUS_COMMUNITIES(id),
    UNIQUE(CP, MUNICIPI, community_id)
) TABLESPACE PANDORA;

CREATE TABLE ISLANDS (
    id INT AUTO_INCREMENT PRIMARY KEY,
    illa VARCHAR(104) NOT NULL UNIQUE
) TABLESPACE PANDORA;

CREATE TABLE COURSE_YEARS (
    id INT AUTO_INCREMENT PRIMARY KEY,
    curs VARCHAR(48) NOT NULL UNIQUE
) TABLESPACE PANDORA;

CREATE TABLE EDUCATION_TYPES (
    id INT AUTO_INCREMENT PRIMARY KEY,
    tipus_ensenyament VARCHAR(56) NOT NULL UNIQUE
) TABLESPACE PANDORA;

CREATE TABLE MODALITIES (
    id INT AUTO_INCREMENT PRIMARY KEY,
    modalitat VARCHAR(56) NOT NULL UNIQUE
) TABLESPACE PANDORA;

CREATE TABLE CENTRE_TYPES (
    id INT AUTO_INCREMENT PRIMARY KEY,
    DENOMINACIO_GENERICA VARCHAR(104) NOT NULL UNIQUE
) TABLESPACE PANDORA;

CREATE TABLE OWNERS (
    id INT AUTO_INCREMENT PRIMARY KEY,
    TITULAR VARCHAR(256) NOT NULL UNIQUE
) TABLESPACE PANDORA;

CREATE TABLE CENTRES (
    CODI VARCHAR(24) PRIMARY KEY,
    type_id INT NOT NULL,
    NOM VARCHAR(256) NOT NULL,
    CORREU_ELECTRONIC_1 VARCHAR(256),
    CORREU_ELECTRONIC_2 VARCHAR(256),
    PAGINA_WEB VARCHAR(256),
    owner_id INT NOT NULL,
    NIF VARCHAR(24) NOT NULL,
    LOCALITAT VARCHAR(256) NOT NULL,
    ADRECA VARCHAR(256) NOT NULL,
    location_id INT NOT NULL,
    island_id INT NOT NULL,
    TELEF1 VARCHAR(16),
    FOREIGN KEY (type_id) REFERENCES CENTRE_TYPES(id),
    FOREIGN KEY (owner_id) REFERENCES OWNERS(id),
    FOREIGN KEY (location_id) REFERENCES LOCATION(id),
    FOREIGN KEY (island_id) REFERENCES ISLANDS(id)
) TABLESPACE PANDORA;

```

```sql
CREATE TABLE STUDENTS (
    dni VARCHAR(24) PRIMARY KEY,
    nom VARCHAR(256) NOT NULL,
    primer_cognom VARCHAR(256) NOT NULL,
    segon_cognom VARCHAR(256),
    correu_electronic VARCHAR(256) NOT NULL, -- Hauria de ser nic per per les dades proporcionades hi ha correus duplicats
    districte VARCHAR(24) NOT NULL,
    location_id INT NOT NULL,
    FOREIGN KEY (location_id) REFERENCES LOCATION(id)
) TABLESPACE PANDORA;

CREATE TABLE SUBJECTS (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nom_assignatura VARCHAR(256) NOT NULL,
    education_type_id INT NOT NULL,
    modality_id INT NOT NULL,
    course_year_id INT NOT NULL,
    FOREIGN KEY (education_type_id) REFERENCES EDUCATION_TYPES(id),
    FOREIGN KEY (modality_id) REFERENCES MODALITIES(id),
    FOREIGN KEY (course_year_id) REFERENCES COURSE_YEARS(id),
    UNIQUE(nom_assignatura, education_type_id, modality_id, course_year_id)
) TABLESPACE PANDORA;

CREATE TABLE ENROLLMENTS (
    dni VARCHAR(24),
    subject_id INT,
    codi_centre VARCHAR(24),
    grup_de_classe VARCHAR(56),
    PRIMARY KEY (dni, subject_id, codi_centre, grup_de_classe),
    FOREIGN KEY (dni) REFERENCES STUDENTS(dni),
    FOREIGN KEY (subject_id) REFERENCES SUBJECTS(id),
    FOREIGN KEY (codi_centre) REFERENCES CENTRES(CODI)
) TABLESPACE PANDORA;
```

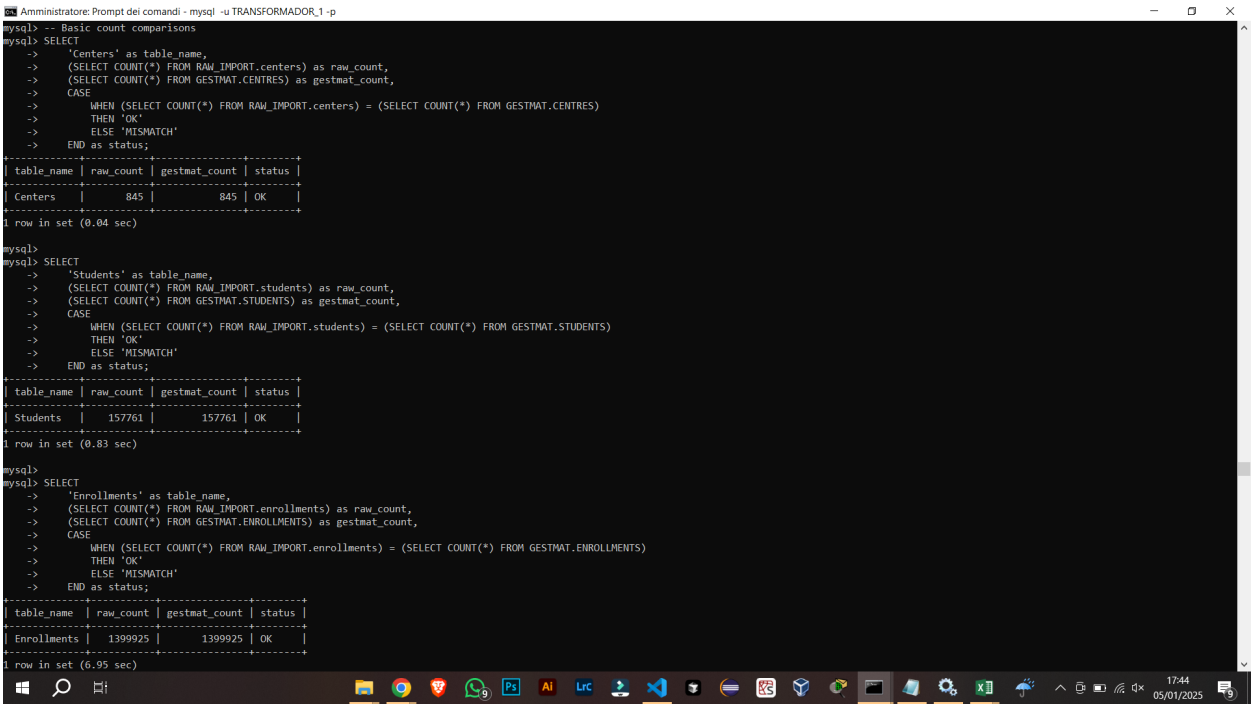
3.2.4 Transferir dades

```sql
-- Primer, omplim totes les taules de consulta
-- Comunitats aut nomes (Els centres no especifiquen la comunitat per tots s n a les Illes Balears)
INSERT INTO AUTONOMOUS_COMMUNITIES (comunitat_autonoma)
VALUES ('Illes Balears');

-- Importem les comunitats niques dels estudiants
INSERT IGNORE INTO AUTONOMOUS_COMMUNITIES (comunitat_autonoma)
SELECT DISTINCT comunitat_autonoma
FROM RAW_IMPORT.students
WHERE comunitat_autonoma IS NOT NULL;

-- Omplim ISLANDS des dels centres
INSERT INTO ISLANDS (illa)
SELECT DISTINCT ILLA
FROM RAW_IMPORT.centers
WHERE ILLA IS NOT NULL;

-- Omplim CENTRE_TYPES des dels centres
INSERT INTO CENTRE_TYPES (DENOMINACIO_GENERICA)
```

```sql
SELECT DISTINCT DENOMINACIO_GENERICA
FROM RAW_IMPORT.centers
WHERE DENOMINACIO_GENERICA IS NOT NULL;

-- Omplim OWNERS des dels centres
INSERT INTO OWNERS (TITULAR)
SELECT DISTINCT TITULAR
FROM RAW_IMPORT.centers
WHERE TITULAR IS NOT NULL;

-- Omplim EDUCATION_TYPES des de les inscripcions
INSERT INTO EDUCATION_TYPES (tipus_ensenyament)
SELECT DISTINCT tipus_ensenyament
FROM RAW_IMPORT.enrollments
WHERE tipus_ensenyament IS NOT NULL;

-- Omplim MODALITIES des de les inscripcions
INSERT INTO MODALITIES (modalitat)
SELECT DISTINCT modalitat
FROM RAW_IMPORT.enrollments
WHERE modalitat IS NOT NULL;

-- Omplim COURSE_YEARS des de les inscripcions
INSERT INTO COURSE_YEARS (curs)
SELECT DISTINCT curs
FROM RAW_IMPORT.enrollments
WHERE curs IS NOT NULL;

-- Omplim la taula LOCATION
-- Primer des dels centres
INSERT INTO LOCATION (CP, MUNICIPI, community_id)
SELECT DISTINCT c.CP, c.MUNICIPI, ac.id
FROM RAW_IMPORT.centers c
CROSS JOIN AUTONOMOUS_COMMUNITIES ac
WHERE ac.comunitat_autonoma = 'Illes Balears'
AND c.CP IS NOT NULL
AND c.MUNICIPI IS NOT NULL;

-- Despr s des dels estudiants
INSERT IGNORE INTO LOCATION (CP, MUNICIPI, community_id)
SELECT DISTINCT
    SUBSTRING(s.codi_postal_i_districte, 1, 5),
    s.municipi,
    ac.id
FROM RAW_IMPORT.students s
JOIN AUTONOMOUS_COMMUNITIES ac ON ac.comunitat_autonoma = s.comunitat_autonoma
WHERE s.codi_postal_i_districte IS NOT NULL
AND s.municipi IS NOT NULL;

-- Omplim CENTRES
INSERT INTO CENTRES
SELECT
    TRIM(LEADING '0' FROM c.CODI) as CODI,
    ct.id as type_id,
    c.NOM,
    c.CORREU_ELECTRONIC_1,
    c.CORREU_ELECTRONIC_2,
    c.PAGINA_WEB,
    o.id as owner_id,
```

```
    c.NIF,
    c.LOCALITAT,
    c.ADRECA,
    l.id as location_id,
    i.id as island_id,
    c.TELEF1
FROM RAW_IMPORT.centers c
JOIN CENTRE_TYPES ct ON ct.DENOMINACIO_GENERICA = c.DENOMINACIO_GENERICA
JOIN OWNERS o ON o.TITULAR = c.TITULAR
JOIN LOCATION l ON l.CP = c.CP AND l.MUNICIPI = c.MUNICIPI
JOIN ISLANDS i ON i.illa = c.ILLA;

-- Omplim STUDENTS
INSERT INTO STUDENTS
SELECT
    s.dni,
    s.nom,
    s.primer_cognom,
    s.segon_cognom,
    s.correu_electronic,
    SUBSTRING(s.codi_postal_i_districte, 6, 2) as districte,
    l.id as location_id
FROM RAW_IMPORT.students s
JOIN LOCATION l ON l.CP = SUBSTRING(s.codi_postal_i_districte, 1, 5)
    AND l.MUNICIPI = s.municipi;

-- Omplim SUBJECTS
INSERT INTO SUBJECTS (nom_assignatura, education_type_id, modality_id,
    course_year_id)
SELECT DISTINCT
    e.nom_assignatura,
    et.id as education_type_id,
    m.id as modality_id,
    cy.id as course_year_id
FROM RAW_IMPORT.enrollments e
JOIN EDUCATION_TYPES et ON et.tipus_ensenyament = e.tipus_ensenyament
JOIN MODALITIES m ON m.modalitat = e.modalitat
JOIN COURSE_YEARS cy ON cy.curs = e.curs;

-- Finalment, omplim ENROLLMENTS
INSERT INTO ENROLLMENTS
SELECT
    e.dni,
    s.id as subject_id,
    e.codi_centre,
    e.grup_de_classe
FROM RAW_IMPORT.enrollments e
JOIN SUBJECTS s ON s.nom_assignatura = e.nom_assignatura
JOIN EDUCATION_TYPES et ON et.tipus_ensenyament = e.tipus_ensenyament
    AND et.id = s.education_type_id
JOIN MODALITIES m ON m.modalitat = e.modalitat
    AND m.id = s.modality_id
JOIN COURSE_YEARS cy ON cy.curs = e.curs
    AND cy.id = s.course_year_id;
```

Ara comprovem que totes les dades s'han transferit correctament:



3.3 Consultes

a

Quins centres tenen usuaris inscrits en cursos de Formació Professional i quants?

```sql
SELECT
    c.CODI,
    c.NOM as centre_name,
    COUNT(DISTINCT e.dni) as num_students
FROM CENTRES c
JOIN ENROLLMENTS e ON c.CODI = e.codi_centre
JOIN SUBJECTS s ON e.subject_id = s.id
JOIN EDUCATION_TYPES et ON s.education_type_id = et.id
WHERE et.tipus_ensenyament = 'FPA'
GROUP BY c.CODI, c.NOM
ORDER BY num_students DESC;
```

```
212 files en conjunt (2.95 seg)
```

b

Mostra els centres més saturats. Demostra-ho amb dades.

```sql
SELECT
    c.CODI,
    c.NOM as centre_name,
    COUNT(DISTINCT e.dni) as total_students,
    COUNT(e.subject_id) as total_enrollments,
    COUNT(e.subject_id) / COUNT(DISTINCT e.dni) as avg_subjects_per_student
FROM CENTRES c
JOIN ENROLLMENTS e ON c.CODI = e.codi_centre
GROUP BY c.CODI
```

```
ORDER BY total_students DESC, total_enrollments DESC
LIMIT 10;
```

10 files en conjunt (35.99 seg)

c

Quines assignatures tenen menys de 4 usuaris inscrits? També extreu les dades del centre on s'ensenya l'assignatura.

```
SELECT
    s.nom_assignatura,
    c.CODI as centre_code,
    c.NOM as centre_name,
    COUNT(DISTINCT e.dni) as num_students
FROM SUBJECTS s
JOIN ENROLLMENTS e ON s.id = e.subject_id
JOIN CENTRES c ON e.codi_centre = c.CODI
GROUP BY s.id, s.nom_assignatura, c.CODI, c.NOM
HAVING num_students < 4
ORDER BY num_students, s.nom_assignatura;
```

13 files en conjunt (5 min 2.88 seg)

d

Quins estudiants no estan inscrits en cap assignatura?

```
SELECT
    s.dni,
    s.nom,
    s.primer_cognom,
    s.segon_cognom,
    s.correu_electronic,
    l.MUNICIPI,
    l.CP,
    s.districte
FROM STUDENTS s
LEFT JOIN ENROLLMENTS e ON s.dni = e.dni
LEFT JOIN LOCATION l ON s.location_id = l.id
WHERE e.dni IS NULL
ORDER BY s.primer_cognom, s.nom;
```

3977 files en conjunt (38.17 seg)

e

Basat en la consulta anterior, extreu el total de nombre d'estudiants no inscrits.

```
SELECT COUNT(*) as total_unenrolled_students
FROM STUDENTS s
LEFT JOIN ENROLLMENTS e ON s.dni = e.dni
WHERE e.dni IS NULL;
```

1 file en conjunt (16.27 seg)

f

Quin centre hauria d'assistir cada estudiant no matriculat? La premissa es basarà en la proximitat del centre al lloc de residència de l'estudiant.

```sql
WITH UnenrolledStudents AS (
    SELECT
        s.dni,
        s.nom,
        s.primer_cognom,
        s.segon_cognom,
        l.MUNICIPI as student_municipi,
        l.CP as student_cp
    FROM STUDENTS s
    LEFT JOIN ENROLLMENTS e ON s.dni = e.dni
    LEFT JOIN LOCATION l ON s.location_id = l.id
    WHERE e.dni IS NULL
),
RankedCenters AS (
    SELECT
        us.dni,
        us.nom,
        us.primer_cognom,
        us.segon_cognom,
        us.student_municipi,
        us.student_cp,
        c.CODI as centre_code,
        c.NOM as centre_name,
        c.LOCALITAT as centre_locality,
        l.CP as centre_cp,
        ROW_NUMBER() OVER (
            PARTITION BY us.dni
            ORDER BY
                CASE WHEN us.student_municipi = l.municipi THEN 1 ELSE 2 END,  -- Prioritat: mateix municipi
                ABS(CAST(us.student_cp AS SIGNED) - CAST(l.CP AS SIGNED))      -- Fallback: codi postal m s proper
        ) as row_num
    FROM UnenrolledStudents us
    CROSS JOIN CENTRES c
    JOIN LOCATION l ON c.location_id = l.id
)
SELECT
    rc.dni,
    rc.nom,
    rc.primer_cognom,
    rc.segon_cognom,
    rc.student_municipi,
    rc.student_cp,
    rc.centre_code,
    rc.centre_name,
    rc.centre_locality,
    rc.centre_cp
FROM RankedCenters rc
WHERE rc.row_num = 1; -- Selecciona nom s el centre amb millor classificaci per a cada estudiant
```

3977 files en el conjunt (23 min 22.67 sec)

3.4 Optimitzacions

a

Consulta A: Centres amb matrícules en Formació Professional

Problemes actuals de rendiment:

1. Múltiples JOINs sense indexació adequada

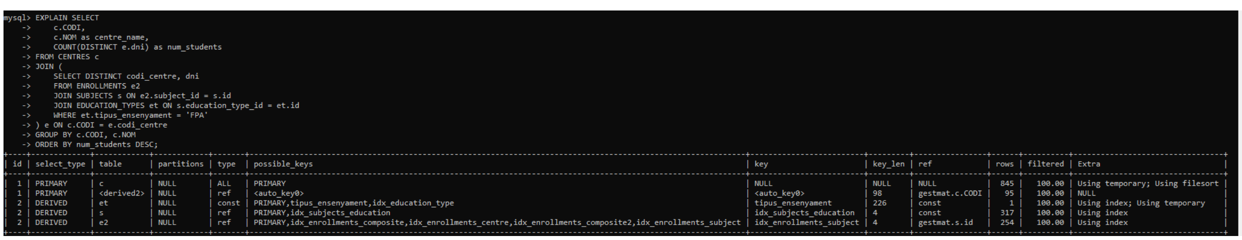
Millores:

```sql
CREATE INDEX idx_education_type ON EDUCATION_TYPES(tipus_ensenyament);
CREATE INDEX idx_enrollments_composite ON ENROLLMENTS(codi_centre, dni);
CREATE INDEX idx_subjects_education ON SUBJECTS(education_type_id);
```

És important també reescriure la consulta per aprofitar completament aquests índexs.

```sql
SELECT
    c.CODI,
    c.NOM as centre_name,
    COUNT(DISTINCT e.dni) as num_students
FROM CENTRES c
JOIN (
    SELECT DISTINCT codi_centre, dni
    FROM ENROLLMENTS e2
    JOIN SUBJECTS s ON e2.subject_id = s.id
    JOIN EDUCATION_TYPES et ON s.education_type_id = et.id
    WHERE et.tipus_ensenyament = 'FPA'
) e ON c.CODI = e.codi_centre
GROUP BY c.CODI, c.NOM
ORDER BY num_students DESC;
```

Comprovem la correcta creació dels índexs amb un explain:



```
mysql> EXPLAIN SELECT
    ->     c.CODI,
    ->     c.NOM as centre_name,
    ->     COUNT(DISTINCT e.dni) as num_students
    -> FROM CENTRES c
    -> JOIN (
    ->     SELECT DISTINCT codi_centre, dni
    ->     FROM ENROLLMENTS e2
    ->     JOIN SUBJECTS s ON e2.subject_id = s.id
    ->     JOIN EDUCATION_TYPES et ON s.education_type_id = et.id
    ->     WHERE et.tipus_ensenyament = 'FPA'
    -> ) e ON c.CODI = e.codi_centre
    -> GROUP BY c.CODI, c.NOM
    -> ORDER BY num_students DESC;
```

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	PRIMARY	c	NULL	ALL	PRIMARY	NULL	NULL	NULL	845	100.00	Using temporary; Using filesort
1	PRIMARY	<derived2>	NULL	ref	<auto_key0>	<auto_key0>	98	gestmat.c.CODI	95	100.00	NULL
2	DERIVED	et	NULL	const	PRIMARY,tipus_ensenyament,idx_education_type	tipus_ensenyament	226	const	1	100.00	Using index; Using temporary
2	DERIVED	s	NULL	ref	PRIMARY,idx_subjects_education	idx_subjects_education	4	const	317	100.00	Using index
2	DERIVED	e2	NULL	ref	PRIMARY,idx_enrollments_composite,idx_enrollments_centre,idx_enrollments_composite2,idx_enrollments_subject	idx_enrollments_subject	4	gestmat.s.id	254	100.00	Using index

b

Consulta B: Centres saturats

Problemes actuals de rendiment:

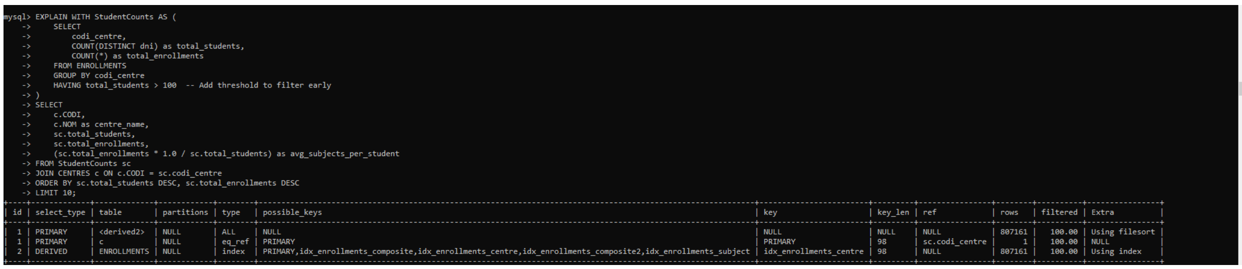
1. Operacions de comptatge en grans resultats de JOIN
2. No es limita el conjunt de dades inicial abans de l'agrupació

Millores:

```sql
CREATE INDEX idx_enrollments_centre ON ENROLLMENTS(codi_centre);
CREATE INDEX idx_enrollments_composite2 ON ENROLLMENTS(codi_centre, dni,
    subject_id);

-- Consulta optimitzada:
WITH StudentCounts AS (
    SELECT
```

```sql
        codi_centre,
        COUNT(DISTINCT dni) as total_students,
        COUNT(*) as total_enrollments
    FROM ENROLLMENTS
    GROUP BY codi_centre
    HAVING total_students > 100  -- Afegir un llindar per filtrar aviat
)
SELECT
    c.CODI,
    c.NOM as centre_name,
    sc.total_students,
    sc.total_enrollments,
    (sc.total_enrollments * 1.0 / sc.total_students) as avg_subjects_per_student
FROM StudentCounts sc
JOIN CENTRES c ON c.CODI = sc.codi_centre
ORDER BY sc.total_students DESC, sc.total_enrollments DESC
LIMIT 10;
```

Comprovem la correcta creació dels índexs amb un explain:



```
mysql> EXPLAIN WITH StudentCounts AS (
    ->     SELECT
    ->         codi_centre,
    ->         COUNT(DISTINCT dni) as total_students,
    ->         COUNT(*) as total_enrollments
    ->     FROM ENROLLMENTS
    ->     GROUP BY codi_centre
    ->     HAVING total_students > 100  -- Add threshold to filter early
    -> )
    -> SELECT
    ->     c.CODI,
    ->     c.NOM as centre_name,
    ->     sc.total_students,
    ->     sc.total_enrollments,
    ->     (sc.total_enrollments * 1.0 / sc.total_students) as avg_subjects_per_student
    -> FROM StudentCounts sc
    -> JOIN CENTRES c ON c.CODI = sc.codi_centre
    -> ORDER BY sc.total_students DESC, sc.total_enrollments DESC
    -> LIMIT 10;
```

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	PRIMARY	<derived2>	NULL	ALL	NULL	NULL	NULL	NULL	807161	100.00	Using filesort
1	PRIMARY	c	NULL	eq_ref	PRIMARY	PRIMARY	98	sc.codi_centre	1	100.00	NULL
2	DERIVED	ENROLLMENTS	NULL	index	PRIMARY,idx_enrollments_composite,idx_enrollments_centre,idx_enrollments_composite2,idx_enrollments_subject	idx_enrollments_centre	98	NULL	807161	100.00	Using index

c

Consulta C: Assignatures amb menys de 4 usuaris
Problemes actuals de rendiment:

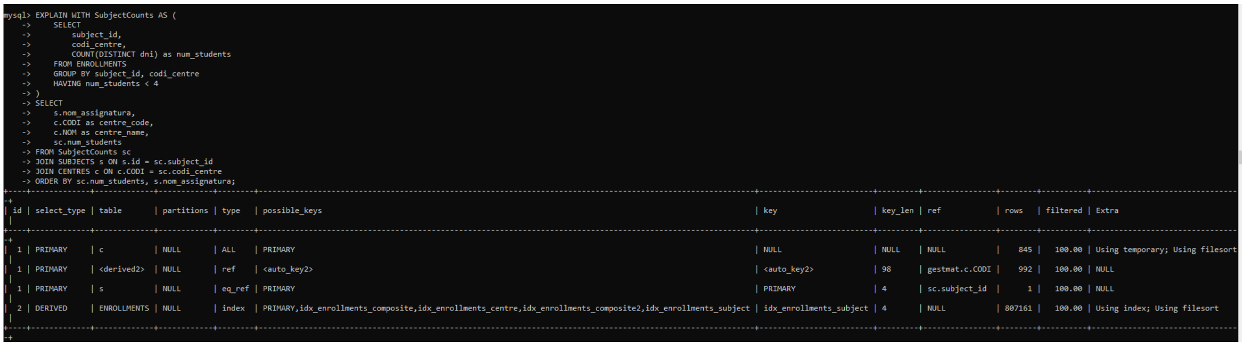
1. Múltiples JOINs abans de l'agrupació
2. No es fa filtratge previ

Millores:

```sql
CREATE INDEX idx_enrollments_subject ON ENROLLMENTS(subject_id);

-- Consulta optimitzada:
WITH SubjectCounts AS (
    SELECT
        subject_id,
        codi_centre,
        COUNT(DISTINCT dni) as num_students
    FROM ENROLLMENTS
    GROUP BY subject_id, codi_centre
    HAVING num_students < 4
)
SELECT
    s.nom_assignatura,
    c.CODI as centre_code,
    c.NOM as centre_name,
    sc.num_students
FROM SubjectCounts sc
JOIN SUBJECTS s ON s.id = sc.subject_id
```

```sql
JOIN CENTRES c ON c.CODI = sc.codi_centre
ORDER BY sc.num_students, s.nom_assignatura;
```

Comprovem la correcta creació dels índexs amb un explain:



```
mysql> EXPLAIN WITH SubjectCounts AS (
    ->     SELECT
    ->         subject_id,
    ->         codi_centre,
    ->         COUNT(DISTINCT dni) as num_students
    ->     FROM ENROLLMENTS
    ->     GROUP BY subject_id, codi_centre
    ->     HAVING num_students < 4
    -> )
    -> SELECT
    ->     s.nom_assignatura,
    ->     c.CODI as centre_code,
    ->     c.NOM as centre_name,
    ->     sc.num_students
    -> FROM SubjectCounts sc
    -> JOIN SUBJECTS s ON s.id = sc.subject_id
    -> JOIN CENTRES c ON c.CODI = sc.codi_centre
    -> ORDER BY sc.num_students, s.nom_assignatura;
```

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	PRIMARY	c	NULL	ALL	PRIMARY	NULL	NULL	NULL	845	100.00	Using temporary; Using filesort
1	PRIMARY	<derived2>	NULL	ref	<auto_key2>	<auto_key2>	98	gestmat.c.CODI	992	100.00	NULL
1	PRIMARY	s	NULL	eq_ref	PRIMARY	PRIMARY	4	sc.subject_id	1	100.00	NULL
2	DERIVED	ENROLLMENTS	NULL	index	PRIMARY,idx_enrollments_composite,idx_enrollments_centre,idx_enrollments_composite2,idx_enrollments_subject	idx_enrollments_subject	4	NULL	807161	100.00	Using index; Using filesort

d & e

Consultes D i E: Estudiants no matriculats

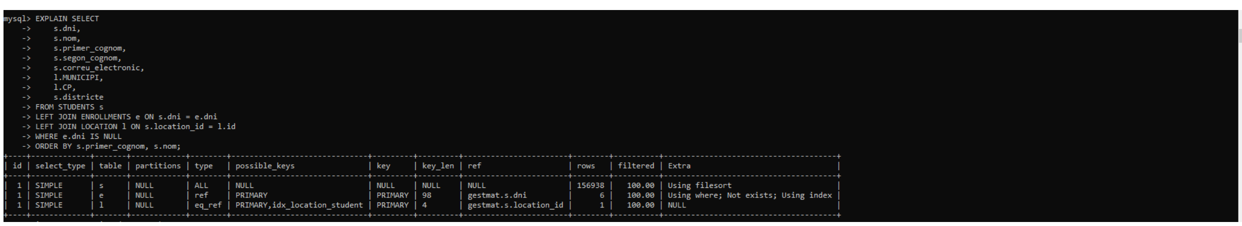
Problemes actuals de rendiment:

1. LEFT JOIN sense indexació adequada

Millores:

```sql
CREATE INDEX idx_students_enrollment ON STUDENTS(dni);
CREATE INDEX idx_location_student ON LOCATION(id);
```

Comprovem la correcta creació dels índexs amb un explain:



```
mysql> EXPLAIN SELECT
    ->     s.dni,
    ->     s.nom,
    ->     s.primer_cognom,
    ->     s.segon_cognom,
    ->     s.correu_electronic,
    ->     l.MUNICIPI,
    ->     l.CP,
    ->     s.districte
    -> FROM STUDENTS s
    -> LEFT JOIN ENROLLMENTS e ON s.dni = e.dni
    -> LEFT JOIN LOCATION l ON s.location_id = l.id
    -> WHERE e.dni IS NULL
    -> ORDER BY s.primer_cognom, s.nom;
```

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	s	NULL	ALL	NULL	NULL	NULL	NULL	156938	100.00	Using filesort
1	SIMPLE	e	NULL	ref	PRIMARY	PRIMARY	98	gestmat.s.dni	6	100.00	Using where; Not exists; Using index
1	SIMPLE	l	NULL	eq_ref	PRIMARY,idx_location_student	PRIMARY	4	gestmat.s.location_id	1	100.00	NULL

f

Consulta F: Recomanacions de centres

Problemes actuals de rendiment:

1. El CROSS JOIN crea un conjunt de resultats enorme
2. Càlculs complexos en ORDER BY
3. Conversió de cadenes a números en l'ordenació

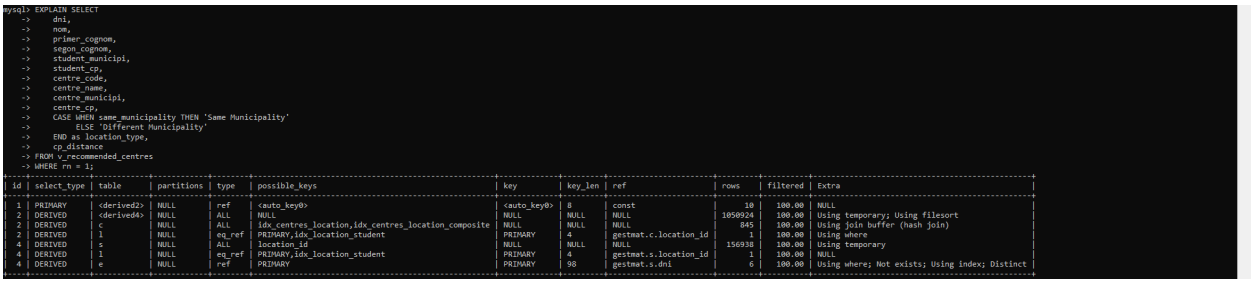
Millores:

```sql
CREATE INDEX idx_location_municipi ON LOCATION(MUNICIPI);
CREATE INDEX idx_location_cp ON LOCATION(CP);
CREATE INDEX idx_centres_location ON CENTRES(location_id);
CREATE INDEX idx_location_municipi_cp ON LOCATION(MUNICIPI, CP);
CREATE INDEX idx_centres_location_composite ON CENTRES(location_id, CODI);
CREATE INDEX idx_enrollments_student ON ENROLLMENTS(dni);


```

```sql
-- Vista base per a estudiants no matriculats amb les seves dades de localitzaci
CREATE VIEW v_unenrolled_students AS
SELECT DISTINCT
    s.dni,
    s.nom,
    s.primer_cognom,
    s.segon_cognom,
    l.MUNICIPI,
    l.CP,
    CAST(REPLACE(l.CP, ' ', '') AS SIGNED) as cp_num
FROM STUDENTS s
LEFT JOIN ENROLLMENTS e ON s.dni = e.dni
JOIN LOCATION l ON s.location_id = l.id
WHERE e.dni IS NULL;

-- Vista base per a centres amb les seves dades de localitzaci
CREATE VIEW v_centre_locations AS
SELECT
    c.CODI,
    c.NOM,
    l.MUNICIPI,
    l.CP,
    CAST(REPLACE(l.CP, ' ', '') AS SIGNED) as cp_num
FROM CENTRES c
JOIN LOCATION l ON c.location_id = l.id;

-- Vista que cont  totes les possibles coincid ncies d'estudiants i centres amb
    les seves dist ncies
CREATE VIEW v_student_centre_distances AS
SELECT
    us.dni,
    us.nom,
    us.primer_cognom,
    us.segon_cognom,
    us.MUNICIPI as student_municipi,
    us.CP as student_cp,
    cl.CODI as centre_code,
    cl.NOM as centre_name,
    cl.MUNICIPI as centre_municipi,
    cl.CP as centre_cp,
    (us.MUNICIPI = cl.MUNICIPI) as same_municipality,
    ABS(us.cp_num - cl.cp_num) as cp_distance
FROM v_unenrolled_students us
CROSS JOIN v_centre_locations cl
WHERE ABS(us.cp_num - cl.cp_num) <= 5000;  -- L mit de dist ncia per reduir les
    files

-- Vista amb els resultats finals classificats
CREATE VIEW v_recommended_centres AS
SELECT
    *,
    ROW_NUMBER() OVER (
        PARTITION BY dni
        ORDER BY
            same_municipality DESC,
            cp_distance
    ) as rn
FROM v_student_centre_distances;

```

```sql
-- Consulta final senzilla
SELECT
    dni,
    nom,
    primer_cognom,
    segon_cognom,
    student_municipi,
    student_cp,
    centre_code,
    centre_name,
    centre_municipi,
    centre_cp,
    CASE WHEN same_municipality THEN 'Same Municipality'
         ELSE 'Different Municipality'
    END as location_type,
    cp_distance
FROM v_recommended_centres
WHERE rn = 1;
```

Comprovem la correcta creació dels índexs amb un explain:



```
mysql> EXPLAIN SELECT
    ->     dni,
    ->     nom,
    ->     primer_cognom,
    ->     segon_cognom,
    ->     student_municipi,
    ->     student_cp,
    ->     centre_code,
    ->     centre_name,
    ->     centre_municipi,
    ->     centre_cp,
    ->     CASE WHEN same_municipality THEN 'Same Municipality'
    ->          ELSE 'Different Municipality'
    ->     END as location_type,
    ->     cp_distance
    -> FROM v_recommended_centres
    -> WHERE rn = 1;
+----+-------------+------------+------------+--------+-----------------------------------------------------+-------------+---------+-------------------------+---------+----------+-----------------------------------------------+
| id | select_type | table      | partitions | type   | possible_keys                                       | key         | key_len | ref                     | rows    | filtered | Extra                                         |
+----+-------------+------------+------------+--------+-----------------------------------------------------+-------------+---------+-------------------------+---------+----------+-----------------------------------------------+
|  1 | PRIMARY     | <derived2> | NULL       | ref    | <auto_key0>                                         | <auto_key0> | 8       | const                   |      10 |   100.00 | NULL                                          |
|  2 | DERIVED     | <derived4> | NULL       | ALL    | NULL                                                | NULL        | NULL    | NULL                    | 1050924 |   100.00 | Using temporary; Using filesort               |
|  2 | DERIVED     | c          | NULL       | ALL    | idx_centres_location,idx_centres_location_composite | NULL        | NULL    | NULL                    |     845 |   100.00 | Using join buffer (hash join)                 |
|  2 | DERIVED     | l          | NULL       | eq_ref | PRIMARY,idx_location_student                        | PRIMARY     | 4       | gestmat.c.location_id   |       1 |   100.00 | Using where                                   |
|  4 | DERIVED     | s          | NULL       | ALL    | location_id                                         | NULL        | NULL    | NULL                    |  156938 |   100.00 | Using temporary                               |
|  4 | DERIVED     | l          | NULL       | eq_ref | PRIMARY,idx_location_student                        | PRIMARY     | 4       | gestmat.s.location_id   |       1 |   100.00 | NULL                                          |
|  4 | DERIVED     | e          | NULL       | ref    | PRIMARY                                             | PRIMARY     | 98      | gestmat.s.dni           |       6 |   100.00 | Using where; Not exists; Using index; Distinct |
+----+-------------+------------+------------+--------+-----------------------------------------------------+-------------+---------+-------------------------+---------+----------+-----------------------------------------------+
```

3.4.1 Demostració d'Optimitzacions

Consulta	Temps Exec. Antic	Temps Exec. Nou	Millora en Temps		Millora en Velocitat
a	2.95 sec	0.66 sec	-2.29 sec	77.63%	4.47x
b	35.99 sec	2.32 sec	-33.67 sec	93.55%	15.51x
c	5 min 2.88 sec	15.81 sec	-287.07 sec	94.78%	19.15x
d	38.17 sec	14.65 sec	-23.52 sec	61.61%	2.61x
e	16.27 sec	3.86 sec	-12.41 sec	76.28%	4.21x
f	23 min 22.67 sec	2 min 47.71 sec	-1234.96 sec	88.05%	8.36x

Capítol 4

Implementació de PostgreSQL

4.1 GESTMAT

4.1.1 Crear DB

```
-- Crear el tablespace PANDORA
CREATE TABLESPACE PANDORA
    LOCATION '/var/lib/postgresql/data/pandora';

-- Crear la base de dades GESTMAT utilitzant el tablespace PANDORA
CREATE DATABASE GESTMAT
    TABLESPACE PANDORA
    ENCODING 'UTF8';
```

4.1.2 OLDGESTMAT

```
-- Connectar-se a la base de dades GESTMAT
\c GESTMAT

-- Crear l'esquema OLDGESTMAT
CREATE SCHEMA OLDGESTMAT;
```

4.1.3 Crear l'usuari UDATAMOVEMENT

```
-- Crear l'usuari UDATAMOVEMENT
CREATE USER UDATAMOVEMENT WITH PASSWORD 'password';

-- Concedir privilegis necessaris a l'usuari UDATAMOVEMENT sobre l'esquema
    OLDGESTMAT
GRANT USAGE ON SCHEMA OLDGESTMAT TO UDATAMOVEMENT;
GRANT CREATE ON SCHEMA OLDGESTMAT TO UDATAMOVEMENT;
GRANT INSERT ON ALL TABLES IN SCHEMA OLDGESTMAT TO UDATAMOVEMENT;

-- Permetre que UDATAMOVEMENT pugui crear noves taules en l'esquema OLDGESTMAT
ALTER DEFAULT PRIVILEGES IN SCHEMA OLDGESTMAT
GRANT INSERT ON TABLES TO UDATAMOVEMENT;

-- Aplicar els privilegis
GRANT CONNECT ON DATABASE GESTMAT TO UDATAMOVEMENT;
```

4.2 PG_LOADER

En el meu cas, utilitzo MySQL al meu PC amb Windows i PostgreSQL en una màquina virtual Ubuntu que s'executa a Oracle VirtualBox en el mateix PC. Per aquesta raó, la migració ha de realitzar-se d'una manera diferent de la normal. [1]

4.2.1 Instal·lar `pgloader` a la VM Ubuntu

Normalment, el descarregaries així:

```
sudo apt-get update
sudo apt-get install pgloader
```

Lamentablement, la versió disponible a apt de pgloader no està actualitzada. Després de cercar a diversos fòrums, la única solució sembla ser compilar pgloader des del codi font:

```
apt remove pgloader -y
git clone https://github.com/dimitri/pgloader.git
apt-get install sbcl unzip libsqlite3-dev make curl gawk freetds-dev libzip-dev
cd pgloader
make pgloader
./build/bin/pgloader --help
```

4.2.2 Configurar MySQL per a Accés Remot

A Windows (MySQL):

1. Edita el fitxer de configuració de MySQL (`my.ini`), que es troba a:

   ```
   C:\ProgramData\MySQL\MySQL Server 8.0\my.ini
   ```

2. A la secció `[mysqld]` afegeix:

   ```
   bind-address = 0.0.0.0
   default_authentication_plugin=mysql_native_password
   ```

3. Reinicia el servidor MySQL des dels serveis de Windows.

Crear usuari MySQL amb accés remot:

```
CREATE USER 'migration_user'@'%' IDENTIFIED BY 'password';
GRANT ALL PRIVILEGES ON GESTMAT.* TO 'migration_user'@'%';
FLUSH PRIVILEGES;
ALTER USER 'migration_user'@'%' IDENTIFIED WITH mysql_native_password BY 'password
    '; -- Necessari perqu  pgloader no suporta l'acc s amb els m todes d'
    autenticaci  m s nous
```

[1] *El resultat final mostrat aquí prové de diverses proves i integracions que vaig haver de fer per a la meva situació particular. He intentat reportar tot en aquest document, però és possible que hagi oblidat alguna cosa. De totes maneres, crec que l'important és que al final vaig aconseguir fer-ho funcionar creuant i integrant diversos comentaris de StackOverflow, problemes a GitHub i documentacions que vaig trobar en línia per entendre com la meva situació era diferent d'una més comuna.*

4.2.3 Configuració de Xarxa

1. Troba l'adreça IP de Windows:
   - Obre CMD i executa: `ipconfig`
   - Anota l'adreça IPv4.
2. Prova la connexió des de la VM Ubuntu:

   ```
   ping <windows_ip_address>
   ```

3. Assegura't que l'accés estigui permès executant la següent ordre a PowerShell de Windows:

   ```
   New-NetFirewallRule -DisplayName "MySQL" -Direction Inbound -LocalPort 3306
   -Protocol TCP -Action Allow
   ```

4.2.4 Crear el fitxer de comandes `pgloader`

Crea un fitxer anomenat `migration.load` a Ubuntu amb:

```
LOAD DATABASE
     FROM mysql://migration_user:password@<windows_ip_address>:3306/GESTMAT
     INTO postgresql://UDATAMOVEMENT:password@localhost/gestmat
     SET search_path TO 'oldgestmat'
;
```

4.2.5 Executar la Migració

Executa la migració:

```
./build/bin/pgloader migration.load
```

```
2025-01-12T18:24:58.152002+01:00 LOG report summary reset
                   table name     errors       rows      bytes      total time
-----------------------------  ---------  ---------  ---------  -------------
               fetch meta data          0         65                    0.179s
                Create Schemas          0          0                    0.000s
              Create SQL Types          0          0                    0.012s
                 Create tables          0         24                    0.077s
                Set Table OIDs          0         12                    0.010s
-----------------------------  ---------  ---------  ---------  -------------
           gestmat.enrollments          0    1399925    32.7 MB         18.341s
              gestmat.subjects          0       2942   133.0 kB          3.950s
              gestmat.students          0     157761    10.0 MB          7.309s
                gestmat.owners          0        262     8.9 kB          4.514s
         gestmat.centre_types          0         57     0.5 kB          5.050s
              gestmat.islands          0          4     0.0 kB          5.519s
           gestmat.modalities          0          3     0.0 kB          5.935s
              gestmat.centres          0        845   120.4 kB          2.907s
             gestmat.location          0        219     4.8 kB          3.686s
       gestmat.education_types          0          4     0.0 kB          4.321s
          gestmat.course_years          0          4     0.0 kB          4.512s
gestmat.autonomous_communities          0          1     0.0 kB          4.941s
-----------------------------  ---------  ---------  ---------  -------------
       COPY Threads Completion          0          4                   18.314s
                Create Indexes          0         41                 1m20.733s
        Index Build Completion          0         41                   20.057s
               Reset Sequences          0          9                    0.132s
                  Primary Keys          0         12                    0.019s
          Create Foreign Keys          0         12                    2.676s
              Create Triggers          0          0                    0.000s
              Set Search Path          0          1                    0.002s
             Install Comments          0          0                    0.000s
-----------------------------  ---------  ---------  ---------  -------------
             Total import time          ✓    1562027    43.0 MB         2m1.933s
```

4.2.6 Verificar la Migració

Des del resum ja podem comprovar que el nombre de files coincideix. Ara comprovem que les taules es van crear efectivament a l'esquema OLDGESTMAT. Connecta't a PostgreSQL i comprova les dades:

```
\c GESTMAT
SET search_path TO OLDGESTMAT;
\dt
```

```
gestmat=# set search_path to oldgestmat;
SET
gestmat=# \dt
                  Lista delle relazioni
   Schema   |          Nome           |  Tipo   | Proprietario
------------+-------------------------+---------+--------------
 oldgestmat | autonomous_communities | tabella | postgres
 oldgestmat | centre_types           | tabella | postgres
 oldgestmat | centres                | tabella | postgres
 oldgestmat | course_years           | tabella | postgres
 oldgestmat | education_types        | tabella | postgres
 oldgestmat | enrollments            | tabella | postgres
 oldgestmat | islands                | tabella | postgres
 oldgestmat | location               | tabella | postgres
 oldgestmat | modalities             | tabella | postgres
 oldgestmat | owners                 | tabella | postgres
 oldgestmat | students               | tabella | postgres
 oldgestmat | subjects               | tabella | postgres
(12 righe)
```

4.3 PROD

Crear l'esquema PROD a la base de dades GESTMAT

```
\c GESTMAT
CREATE SCHEMA PROD;
```

4.3.1 Crear l'usuari UCONSELLERIA

```
CREATE USER UCONSELLERIA WITH PASSWORD 'password';

-- Fer que UCONSELLERIA sigui el propietari de la base de dades GESTMAT
ALTER DATABASE GESTMAT OWNER TO UCONSELLERIA;

-- Concedir tots els privilegis sobre els esquemes existents
GRANT ALL PRIVILEGES ON SCHEMA OLDGESTMAT TO UCONSELLERIA;
GRANT ALL PRIVILEGES ON SCHEMA PROD TO UCONSELLERIA;
GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA OLDGESTMAT TO UCONSELLERIA;
GRANT ALL PRIVILEGES ON ALL SEQUENCES IN SCHEMA OLDGESTMAT TO UCONSELLERIA;
GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA PROD TO UCONSELLERIA;
GRANT ALL PRIVILEGES ON ALL SEQUENCES IN SCHEMA PROD TO UCONSELLERIA;
```

Ara connecta't com a UCONSELLERIA:

```
postgres=# exit
mivige@mivige-VirtualBox:~$ psql -U uconselleria -d gestmat
```

4.3.2 Crear la DB

Crear totes les taules a l'esquema PROD canviant petits detalls de la implementació de MySQL, com per exemple
AUTO_INCREMENT a SERIAL o INT a INTEGER:

```
SET search_path TO PROD;

CREATE TABLE AUTONOMOUS_COMMUNITIES (
    id SERIAL PRIMARY KEY,
    comunitat_autonoma VARCHAR(104) NOT NULL UNIQUE
);

CREATE TABLE LOCATION (
    id SERIAL PRIMARY KEY,
    CP VARCHAR(8) NOT NULL,
    MUNICIPI VARCHAR(104) NOT NULL,
    community_id INTEGER NOT NULL,
    FOREIGN KEY (community_id) REFERENCES AUTONOMOUS_COMMUNITIES(id),
    UNIQUE(CP, MUNICIPI, community_id)
);

CREATE TABLE ISLANDS (
    id SERIAL PRIMARY KEY,
    illa VARCHAR(104) NOT NULL UNIQUE
);

CREATE TABLE COURSE_YEARS (
    id SERIAL PRIMARY KEY,
    curs VARCHAR(48) NOT NULL UNIQUE
);

CREATE TABLE EDUCATION_TYPES (
    id SERIAL PRIMARY KEY,
    tipus_ensenyament VARCHAR(56) NOT NULL UNIQUE
);

CREATE TABLE MODALITIES (
    id SERIAL PRIMARY KEY,
    modalitat VARCHAR(56) NOT NULL UNIQUE
);

CREATE TABLE CENTRE_TYPES (
    id SERIAL PRIMARY KEY,
    DENOMINACIO_GENERICA VARCHAR(104) NOT NULL UNIQUE
);

CREATE TABLE OWNERS (
    id SERIAL PRIMARY KEY,
    TITULAR VARCHAR(256) NOT NULL UNIQUE
);

CREATE TABLE CENTRES (
    CODI VARCHAR(24) PRIMARY KEY,
    type_id INTEGER NOT NULL,
    NOM VARCHAR(256) NOT NULL,
    CORREU_ELECTRONIC_1 VARCHAR(256),
    CORREU_ELECTRONIC_2 VARCHAR(256),
    PAGINA_WEB VARCHAR(256),
```

```
    owner_id INTEGER NOT NULL,
    NIF VARCHAR(24) NOT NULL,
    LOCALITAT VARCHAR(256) NOT NULL,
    ADRECA VARCHAR(256) NOT NULL,
    location_id INTEGER NOT NULL,
    island_id INTEGER NOT NULL,
    TELEF1 VARCHAR(16),
    FOREIGN KEY (type_id) REFERENCES CENTRE_TYPES(id),
    FOREIGN KEY (owner_id) REFERENCES OWNERS(id),
    FOREIGN KEY (location_id) REFERENCES LOCATION(id),
    FOREIGN KEY (island_id) REFERENCES ISLANDS(id)
);

CREATE TABLE STUDENTS (
    dni VARCHAR(24) PRIMARY KEY,
    nom VARCHAR(256) NOT NULL,
    primer_cognom VARCHAR(256) NOT NULL,
    segon_cognom VARCHAR(256),
    correu_electronic VARCHAR(256) NOT NULL,  -- Hauria de ser  nic , per  per
        les dades proporcionades hi ha correus duplicats
    districte VARCHAR(24) NOT NULL,
    location_id INTEGER NOT NULL,
    FOREIGN KEY (location_id) REFERENCES LOCATION(id)
);

CREATE TABLE SUBJECTS (
    id SERIAL PRIMARY KEY,
    nom_assignatura VARCHAR(256) NOT NULL,
    education_type_id INTEGER NOT NULL,
    modality_id INTEGER NOT NULL,
    course_year_id INTEGER NOT NULL,
    FOREIGN KEY (education_type_id) REFERENCES EDUCATION_TYPES(id),
    FOREIGN KEY (modality_id) REFERENCES MODALITIES(id),
    FOREIGN KEY (course_year_id) REFERENCES COURSE_YEARS(id),
    UNIQUE(nom_assignatura, education_type_id, modality_id, course_year_id)
);

CREATE TABLE ENROLLMENTS (
    dni VARCHAR(24),
    subject_id INTEGER,
    codi_centre VARCHAR(24),
    grup_de_classe VARCHAR(56),
    PRIMARY KEY (dni, subject_id, codi_centre, grup_de_classe),
    FOREIGN KEY (dni) REFERENCES STUDENTS(dni),
    FOREIGN KEY (subject_id) REFERENCES SUBJECTS(id),
    FOREIGN KEY (codi_centre) REFERENCES CENTRES(CODI)
);
```

4.3.3 Moviment de dades

Migració de dades des de OLDGESTMAT a l'esquema PROD. Primer, connecta't com a UCONSELLERIA:

```
postgres=# exit
mivige@mivige-VirtualBox:~$ psql -U uconselleria -d gestmat
```

Després inicia la migració de cada taula en l'ordre correcte per mantenir la integritat referencial:

```
SET search_path TO PROD, OLDGESTMAT;

```

```sql
INSERT INTO PROD.AUTONOMOUS_COMMUNITIES (id, comunitat_autonoma)
SELECT id, comunitat_autonoma FROM OLDGESTMAT.AUTONOMOUS_COMMUNITIES;

INSERT INTO PROD.ISLANDS (id, illa)
SELECT id, illa FROM OLDGESTMAT.ISLANDS;

INSERT INTO PROD.COURSE_YEARS (id, curs)
SELECT id, curs FROM OLDGESTMAT.COURSE_YEARS;

INSERT INTO PROD.EDUCATION_TYPES (id, tipus_ensenyament)
SELECT id, tipus_ensenyament FROM OLDGESTMAT.EDUCATION_TYPES;

INSERT INTO PROD.MODALITIES (id, modalitat)
SELECT id, modalitat FROM OLDGESTMAT.MODALITIES;

INSERT INTO PROD.CENTRE_TYPES (id, DENOMINACIO_GENERICA)
SELECT id, DENOMINACIO_GENERICA FROM OLDGESTMAT.CENTRE_TYPES;

INSERT INTO PROD.OWNERS (id, TITULAR)
SELECT id, TITULAR FROM OLDGESTMAT.OWNERS;

INSERT INTO PROD.LOCATION (id, CP, MUNICIPI, community_id)
SELECT id, CP, MUNICIPI, community_id FROM OLDGESTMAT.LOCATION;

INSERT INTO PROD.CENTRES (
    CODI, type_id, NOM, CORREU_ELECTRONIC_1, CORREU_ELECTRONIC_2,
    PAGINA_WEB, owner_id, NIF, LOCALITAT, ADRECA,
    location_id, island_id, TELEF1
)
SELECT
    CODI, type_id, NOM, CORREU_ELECTRONIC_1, CORREU_ELECTRONIC_2,
    PAGINA_WEB, owner_id, NIF, LOCALITAT, ADRECA,
    location_id, island_id, TELEF1
FROM OLDGESTMAT.CENTRES;

INSERT INTO PROD.STUDENTS (
    dni, nom, primer_cognom, segon_cognom,
    correu_electronic, districte, location_id
)
SELECT
    dni, nom, primer_cognom, segon_cognom,
    correu_electronic, districte, location_id
FROM OLDGESTMAT.STUDENTS;

INSERT INTO PROD.SUBJECTS (
    id, nom_assignatura, education_type_id,
    modality_id, course_year_id
)
SELECT
    id, nom_assignatura, education_type_id,
    modality_id, course_year_id
FROM OLDGESTMAT.SUBJECTS;

INSERT INTO PROD.ENROLLMENTS (
    dni, subject_id, codi_centre, grup_de_classe
)
SELECT
    dni, subject_id, codi_centre, grup_de_classe
FROM OLDGESTMAT.ENROLLMENTS;
```

Integritat de dades

Abans de fer res, verifiquem la consistència de les dades entre els esquemes. Per fer-ho, vaig crear una funció de verificació integral que comprova totes les taules:

```
CREATE OR REPLACE FUNCTION verify_table_counts() RETURNS TABLE (
    table_name text,
    oldgestmat_count bigint,
    prod_count bigint,
    is_equal boolean
) AS $$
BEGIN
    RETURN QUERY

    -- Verificaci d'AUTONOMOUS_COMMUNITIES
    SELECT
        'AUTONOMOUS_COMMUNITIES'::text,
        (SELECT count(*) FROM OLDGESTMAT.AUTONOMOUS_COMMUNITIES),
        (SELECT count(*) FROM PROD.AUTONOMOUS_COMMUNITIES),
        (SELECT count(*) FROM OLDGESTMAT.AUTONOMOUS_COMMUNITIES) =
        (SELECT count(*) FROM PROD.AUTONOMOUS_COMMUNITIES)

    UNION ALL

    -- Verificaci de LOCATION
    SELECT
        'LOCATION'::text,
        (SELECT count(*) FROM OLDGESTMAT.LOCATION),
        (SELECT count(*) FROM PROD.LOCATION),
        (SELECT count(*) FROM OLDGESTMAT.LOCATION) =
        (SELECT count(*) FROM PROD.LOCATION)

    UNION ALL

    -- Verificaci de ISLANDS
    SELECT
        'ISLANDS'::text,
        (SELECT count(*) FROM OLDGESTMAT.ISLANDS),
        (SELECT count(*) FROM PROD.ISLANDS),
        (SELECT count(*) FROM OLDGESTMAT.ISLANDS) =
        (SELECT count(*) FROM PROD.ISLANDS)

    UNION ALL

    -- Verificaci de COURSE_YEARS
    SELECT
        'COURSE_YEARS'::text,
        (SELECT count(*) FROM OLDGESTMAT.COURSE_YEARS),
        (SELECT count(*) FROM PROD.COURSE_YEARS),
        (SELECT count(*) FROM OLDGESTMAT.COURSE_YEARS) =
        (SELECT count(*) FROM PROD.COURSE_YEARS)

    UNION ALL

    -- Verificaci de EDUCATION_TYPES
    SELECT
        'EDUCATION_TYPES'::text,
```

```
53          (SELECT count(*) FROM OLDGESTMAT.EDUCATION_TYPES),
54          (SELECT count(*) FROM PROD.EDUCATION_TYPES),
55          (SELECT count(*) FROM OLDGESTMAT.EDUCATION_TYPES) =
56          (SELECT count(*) FROM PROD.EDUCATION_TYPES)
57
58      UNION ALL
59
60      -- Verificaci  de MODALITIES
61      SELECT
62          'MODALITIES'::text,
63          (SELECT count(*) FROM OLDGESTMAT.MODALITIES),
64          (SELECT count(*) FROM PROD.MODALITIES),
65          (SELECT count(*) FROM OLDGESTMAT.MODALITIES) =
66          (SELECT count(*) FROM PROD.MODALITIES)
67
68      UNION ALL
69
70      -- Verificaci  de CENTRE_TYPES
71      SELECT
72          'CENTRE_TYPES'::text,
73          (SELECT count(*) FROM OLDGESTMAT.CENTRE_TYPES),
74          (SELECT count(*) FROM PROD.CENTRE_TYPES),
75          (SELECT count(*) FROM OLDGESTMAT.CENTRE_TYPES) =
76          (SELECT count(*) FROM PROD.CENTRE_TYPES)
77
78      UNION ALL
79
80      -- Verificaci  de OWNERS
81      SELECT
82          'OWNERS'::text,
83          (SELECT count(*) FROM OLDGESTMAT.OWNERS),
84          (SELECT count(*) FROM PROD.OWNERS),
85          (SELECT count(*) FROM OLDGESTMAT.OWNERS) =
86          (SELECT count(*) FROM PROD.OWNERS)
87
88      UNION ALL
89
90      -- Verificaci  de CENTRES
91      SELECT
92          'CENTRES'::text,
93          (SELECT count(*) FROM OLDGESTMAT.CENTRES),
94          (SELECT count(*) FROM PROD.CENTRES),
95          (SELECT count(*) FROM OLDGESTMAT.CENTRES) =
96          (SELECT count(*) FROM PROD.CENTRES)
97
98      UNION ALL
99
100     -- Verificaci  de STUDENTS
101     SELECT
102         'STUDENTS'::text,
103         (SELECT count(*) FROM OLDGESTMAT.STUDENTS),
104         (SELECT count(*) FROM PROD.STUDENTS),
105         (SELECT count(*) FROM OLDGESTMAT.STUDENTS) =
106         (SELECT count(*) FROM PROD.STUDENTS)
107
108     UNION ALL
109
110     -- Verificaci  de SUBJECTS
111     SELECT
```

```
        'SUBJECTS'::text,
        (SELECT count(*) FROM OLDGESTMAT.SUBJECTS),
        (SELECT count(*) FROM PROD.SUBJECTS),
        (SELECT count(*) FROM OLDGESTMAT.SUBJECTS) =
        (SELECT count(*) FROM PROD.SUBJECTS)

    UNION ALL

    -- Verificaci de ENROLLMENTS
    SELECT
        'ENROLLMENTS'::text,
        (SELECT count(*) FROM OLDGESTMAT.ENROLLMENTS),
        (SELECT count(*) FROM PROD.ENROLLMENTS),
        (SELECT count(*) FROM OLDGESTMAT.ENROLLMENTS) =
        (SELECT count(*) FROM PROD.ENROLLMENTS);
END;
$$ LANGUAGE plpgsql;
```

```
gestmat=> SELECT
    table_name,
    oldgestmat_count,
    prod_count,
    CASE
        WHEN is_equal THEN 'OK'
        ELSE 'MISMATCH!'
    END as status
FROM verify_table_counts()
ORDER BY table_name;
       table_name       | oldgestmat_count | prod_count | status
------------------------+------------------+------------+--------
 AUTONOMOUS_COMMUNITIES |                1 |          1 | OK
 CENTRES                |              845 |        845 | OK
 CENTRE_TYPES           |               57 |         57 | OK
 COURSE_YEARS           |                4 |          4 | OK
 EDUCATION_TYPES        |                4 |          4 | OK
 ENROLLMENTS            |          1399925 |    1399925 | OK
 ISLANDS                |                4 |          4 | OK
 LOCATION               |              219 |        219 | OK
 MODALITIES             |                3 |          3 | OK
 OWNERS                 |              262 |        262 | OK
 STUDENTS               |           157761 |     157761 | OK
 SUBJECTS               |             2942 |       2942 | OK
(12 righe)
```

Ara podem estar segurs que totes les dades s'han transferit correctament, així que podem començar la neteja:

- Eliminar la funció de verificació

```
DROP FUNCTION verify_table_counts();
```

a Eliminar OLDGESTMAT

```
DROP SCHEMA OLDGESTMAT CASCADE;
ALTER DATABASE GESTMAT SET search_path TO prod, public;
```

b Eliminar l'usuari UDATAMOVEMENT. És necessari entrar com a administrador:

```
gestmat=> exit
mivige@mivige-VirtualBox:~$ psql -U postgres
```

```
REVOKE ALL ON DATABASE GESTMAT FROM UDATAMOVEMENT;
DROP USER UDATAMOVEMENT;
```

4.4 Consultes

Ara adaptem les consultes de MySQL.

a

No cal cap adaptació.

```
SELECT
    c.CODI,
    c.NOM as centre_name,
    COUNT(DISTINCT e.dni) as num_students
FROM CENTRES c
JOIN ENROLLMENTS e ON c.CODI = e.codi_centre
JOIN SUBJECTS s ON e.subject_id = s.id
JOIN EDUCATION_TYPES et ON s.education_type_id = et.id
WHERE et.tipus_ensenyament = 'FPA'
GROUP BY c.CODI, c.NOM
ORDER BY num_students DESC;
```

```
Time: 795.678 ms
```

b

Com la primera, no cal cap canvi.

```
SELECT
    c.CODI,
    c.NOM as centre_name,
    COUNT(DISTINCT e.dni) as total_students,
    COUNT(e.subject_id) as total_enrollments,
    ROUND(COUNT(e.subject_id)::numeric / NULLIF(COUNT(DISTINCT e.dni), 0), 2) as
        avg_subjects_per_student
FROM CENTRES c
JOIN ENROLLMENTS e ON c.CODI = e.codi_centre
GROUP BY c.CODI, c.NOM
ORDER BY total_students DESC, total_enrollments DESC
LIMIT 10;
```

```
Time: 5609.533 ms
```

c

No cal cap canvi.

```sql
SELECT
    s.nom_assignatura,
    c.CODI as centre_code,
    c.NOM as centre_name,
    COUNT(DISTINCT e.dni) as num_students
FROM SUBJECTS s
JOIN ENROLLMENTS e ON s.id = e.subject_id
JOIN CENTRES c ON e.codi_centre = c.CODI
GROUP BY s.id, s.nom_assignatura, c.CODI, c.NOM
HAVING COUNT(DISTINCT e.dni) < 4
ORDER BY num_students, s.nom_assignatura;
```

```
Time: 4282.806 ms
```

d

Res a canviar.

```sql
SELECT
    s.dni,
    s.nom,
    s.primer_cognom,
    s.segon_cognom,
    s.correu_electronic,
    l.MUNICIPI,
    l.CP,
    s.districte
FROM STUDENTS s
LEFT JOIN ENROLLMENTS e ON s.dni = e.dni
LEFT JOIN LOCATION l ON s.location_id = l.id
WHERE e.dni IS NULL
ORDER BY s.primer_cognom, s.nom;
```

```
Time: 715.837 ms
```

e

Res a ajustar.

```sql
SELECT COUNT(*) as total_unenrolled_students
FROM STUDENTS s
LEFT JOIN ENROLLMENTS e ON s.dni = e.dni
WHERE e.dni IS NULL;
```

```
Time: 542.708 ms
```

f

L'únic ajust a fer és quan es fa el cast, substituint 'AS SIGNED' per 'AS INTEGER'.

```sql
WITH UnenrolledStudents AS (
    SELECT
        s.dni,
        s.nom,
```

```
        s.primer_cognom,
        s.segon_cognom,
        l.MUNICIPI as student_municipi,
        l.CP as student_cp
    FROM STUDENTS s
    LEFT JOIN ENROLLMENTS e ON s.dni = e.dni
    LEFT JOIN LOCATION l ON s.location_id = l.id
    WHERE e.dni IS NULL
),
RankedCenters AS (
    SELECT
        us.dni,
        us.nom,
        us.primer_cognom,
        us.segon_cognom,
        us.student_municipi,
        us.student_cp,
        c.CODI as centre_code,
        c.NOM as centre_name,
        c.LOCALITAT as centre_locality,
        l.CP as centre_cp,
        ROW_NUMBER() OVER (
            PARTITION BY us.dni
            ORDER BY
                CASE WHEN us.student_municipi = l.municipi THEN 1 ELSE 2 END,   -- Prioritat: Mismunicipi
                ABS(CAST(us.student_cp AS INTEGER) - CAST(l.CP AS INTEGER))     -- Recolzament: Codi postal m s proper
        ) as row_num
    FROM UnenrolledStudents us
    CROSS JOIN CENTRES c
    JOIN LOCATION l ON c.location_id = l.id
)
SELECT
    rc.dni,
    rc.nom,
    rc.primer_cognom,
    rc.segon_cognom,
    rc.student_municipi,
    rc.student_cp,
    rc.centre_code,
    rc.centre_name,
    rc.centre_locality,
    rc.centre_cp
FROM RankedCenters rc
WHERE rc.row_num = 1
ORDER BY rc.primer_cognom, rc.nom;
```

```
Time: 8968.675 ms
```

4.5 Optimitzacions

Una cosa important a considerar és el fet que pgloader també va transferir els índexs creats per optimitzar les consultes a MySQL, així que aquestes consultes ja estan bastant optimitzades. PostgreSQL implementa mètodes d'optimització específics:

f, d i e

Podem començar creant vistes per a l'última consulta (la més llarga actualment), ja que no van ser transferides per pgloader i amb vistes materialitzades a PostgreSQL podem obtenir grans millores.

```sql
CREATE MATERIALIZED VIEW mv_unenrolled_students AS
SELECT DISTINCT
    s.dni,
    s.nom,
    s.primer_cognom,
    s.segon_cognom,
    l.MUNICIPI,
    l.CP,
    CAST(REPLACE(l.CP, ' ', '') AS INTEGER) as cp_num
FROM STUDENTS s
LEFT JOIN ENROLLMENTS e ON s.dni = e.dni
JOIN LOCATION l ON s.location_id = l.id
WHERE e.dni IS NULL
WITH DATA;

CREATE MATERIALIZED VIEW mv_centre_locations AS
SELECT
    c.CODI,
    c.NOM,
    l.MUNICIPI,
    l.CP,
    CAST(REPLACE(l.CP, ' ', '') AS INTEGER) as cp_num
FROM CENTRES c
JOIN LOCATION l ON c.location_id = l.id
WITH DATA;
```

Per aprofitar completament les vistes materialitzades també podem crear índexs sobre elles.

```sql
CREATE INDEX idx_mv_unenrolled_cp_num ON mv_unenrolled_students(cp_num);
CREATE INDEX idx_mv_centre_cp_num ON mv_centre_locations(cp_num);
```

Ara podem millorar l'estructura de la consulta i utilitzar les vistes.

```sql
SET max_parallel_workers_per_gather = 4;

WITH RECURSIVE closest_centers AS MATERIALIZED (
    SELECT
        us.dni,
        us.nom,
        us.primer_cognom,
        us.segon_cognom,
        us.MUNICIPI as student_municipi,
        us.CP as student_cp,
        cl.CODI as centre_code,
        cl.NOM as centre_name,
        cl.MUNICIPI as centre_municipi,
        cl.CP as centre_cp,
        us.MUNICIPI = cl.MUNICIPI as same_municipality,
        ABS(us.cp_num - cl.cp_num) as cp_distance,
        ROW_NUMBER() OVER (
            PARTITION BY us.dni
            ORDER BY
                (us.MUNICIPI = cl.MUNICIPI) DESC,
                ABS(us.cp_num - cl.cp_num)
        ) as rn
    FROM mv_unenrolled_students us
```

```sql
    CROSS JOIN LATERAL (
        SELECT *
        FROM mv_centre_locations cl
        WHERE ABS(us.cp_num - cl.cp_num) < 5000
        ORDER BY (us.MUNICIPI = cl.MUNICIPI) DESC,
                ABS(us.cp_num - cl.cp_num)
        LIMIT 1
    ) cl
)
SELECT
    dni,
    nom,
    primer_cognom,
    segon_cognom,
    student_municipi,
    student_cp,
    centre_code,
    centre_name,
    centre_municipi,
    centre_cp,
    CASE
        WHEN same_municipality THEN 'Mismunicipi'
        ELSE 'Diferent municipi'
    END as location_type,
    cp_distance
FROM closest_centers;
```

```
        FROM mv_centre_locations cl
        WHERE ABS(us.cp_num - cl.cp_num) < 5000
        ORDER BY (us.MUNICIPI = cl.MUNICIPI) DESC,
                ABS(us.cp_num - cl.cp_num)
        LIMIT 1
    ) cl
)
SELECT
    dni,
    nom,
    primer_cognom,
    segon_cognom,
    student_municipi,
    student_cp,
    centre_code,
    centre_name,
    centre_municipi,
    centre_cp,
    CASE
        WHEN same_municipality THEN 'Same Municipality'
        ELSE 'Different Municipality'
    END as location_type,
FROM closest_centers;
                                              QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------
 CTE Scan on closest_centers  (cost=14261.34..14340.88 rows=3977 width=2752)
   CTE closest_centers
     ->  WindowAgg  (cost=14142.03..14261.34 rows=3977 width=108)
           ->  Sort  (cost=14142.03..14151.97 rows=3977 width=100)
                 Sort Key: us.dni, (((us.municipi)::text = (cl.municipi)::text)) DESC, (abs((us.cp_num - cl.cp_num)))
                 ->  Nested Loop  (cost=34.32..13904.26 rows=3977 width=100)
                       ->  Seq Scan on mv_unenrolled_students us  (cost=0.00..82.77 rows=3977 width=52)
                       ->  Memoize  (cost=34.32..34.34 rows=1 width=51)
                             Cache Key: us.municipi, us.cp_num, us.cp_num
                             Cache Mode: binary
                             ->  Subquery Scan on cl  (cost=34.31..34.33 rows=1 width=51)
                                   ->  Limit  (cost=34.31..34.32 rows=1 width=56)
                                         ->  Sort  (cost=34.31..35.02 rows=282 width=56)
                                               Sort Key: (((us.municipi)::text = (cl_1.municipi)::text)) DESC, (abs((us.cp_num - cl_1.cp_num)))
                                               ->  Seq Scan on mv_centre_locations cl_1  (cost=0.00..32.90 rows=282 width=56)
                                                     Filter: (abs((us.cp_num - cp_num)) < 5000)
(16 righe)
```

Òbviament, les vistes creades han permès millorar també les consultes d i e mitjançant el seu ús.

```sql
-- d. Quins estudiants no estan matriculats a cap assignatura?
SELECT
    s.dni,
    s.nom,
    s.primer_cognom,
    s.segon_cognom,
    s.MUNICIPI,
    s.CP,
    s.cp_num
FROM mv_unenrolled_students s
ORDER BY s.primer_cognom, s.nom;

-- e. A partir de la consulta anterior, obtenir el total.
SELECT COUNT(*) as total_unenrolled_students
FROM mv_unenrolled_students;
```

```
gestmat=> EXPLAIN SELECT
    s.dni,
    s.nom,
    s.primer_cognom,
    s.segon_cognom,
    s.MUNICIPI,
    s.CP,
    s.cp_num
FROM mv_unenrolled_students s
ORDER BY s.primer_cognom, s.nom;
                                QUERY PLAN
---------------------------------------------------------------------------
 Sort  (cost=320.54..330.49 rows=3977 width=52)
   Sort Key: primer_cognom, nom
   ->  Seq Scan on mv_unenrolled_students s  (cost=0.00..82.77 rows=3977 width=52)
(3 righe)

Tempo: 8,361 ms
gestmat=> EXPLAIN SELECT COUNT(*) as total_unenrolled_students
FROM mv_unenrolled_students;
                                QUERY PLAN
---------------------------------------------------------------------------
 Aggregate  (cost=92.71..92.72 rows=1 width=8)
   ->  Seq Scan on mv_unenrolled_students  (cost=0.00..82.77 rows=3977 width=0)
(2 righe)
```

b

Ara podem millorar la segona consulta, crearem una vista que representi la saturació dels centres.

```sql
CREATE MATERIALIZED VIEW mv_centre_saturation AS
SELECT
    c.CODI,
    c.NOM as centre_name,
    COUNT(DISTINCT e.dni) as total_students,
    COUNT(e.subject_id) as total_enrollments,
    ROUND(COUNT(e.subject_id)::numeric / NULLIF(COUNT(DISTINCT e.dni), 0), 2) as
        avg_subjects_per_student
```

```sql
FROM CENTRES c
JOIN ENROLLMENTS e ON c.CODI = e.codi_centre
GROUP BY c.CODI, c.NOM
WITH DATA;

CREATE INDEX idx_mv_centre_saturation_students ON mv_centre_saturation(
    total_students DESC);
CREATE INDEX idx_mv_centre_saturation_enrollments ON mv_centre_saturation(
    total_enrollments DESC);
```

La consulta resultants serà:

```sql
SELECT * FROM mv_centre_saturation
ORDER BY total_students DESC, total_enrollments DESC
LIMIT 10;
```

```
gestmat=> EXPLAIN SELECT * FROM mv_centre_saturation
ORDER BY total_students DESC, total_enrollments DESC
LIMIT 10;
                                                                  QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------------------------------
 Limit  (cost=0.31..1.97 rows=10 width=53)
   ->  Incremental Sort  (cost=0.31..35.48 rows=212 width=53)
         Sort Key: total_students DESC, total_enrollments DESC
         Presorted Key: total_students
         ->  Index Scan using idx_mv_centre_saturation_students on mv_centre_saturation  (cost=0.14..27.30 rows=212 width=53)
(5 righe)

Tempo: 0,604 ms
```

c

En aquest punt, per millorar la tercera consulta, podem crear una vista que representi les matriculacions de cada assignatura. Posteriorment, en la consulta seleccionarem les assignatures amb menys de 4 estudiants, això per poder utilitzar la vista en futures consultes sobre temes similars.

```sql
CREATE MATERIALIZED VIEW mv_enrollment_subjects AS
SELECT
    s.id,
    s.nom_assignatura,
    c.CODI as centre_code,
    c.NOM as centre_name,
    COUNT(DISTINCT e.dni) as num_students
FROM SUBJECTS s
JOIN ENROLLMENTS e ON s.id = e.subject_id
JOIN CENTRES c ON e.codi_centre = c.CODI
GROUP BY s.id, s.nom_assignatura, c.CODI, c.NOM
WITH DATA;

CREATE INDEX idx_mv_enrollment_students ON mv_enrollment_subjects(num_students);
CREATE INDEX idx_mv_enrollment_subject ON mv_enrollment_subjects(nom_assignatura);
```

Com s'ha dit, la consulta resultants serà:

```sql
SELECT * FROM mv_enrollment_subjects
WHERE num_students < 4
ORDER BY num_students, nom_assignatura;
```

a

Per a la primera consulta, podem reutilitzar la vista creada per a la consulta *c* amb algunes additions.

```sql
SELECT
    c.centre_code,
    c.centre_name,
    SUM(c.num_students) as num_students
FROM mv_enrollment_subjects c
JOIN SUBJECTS s ON c.id = s.id
JOIN EDUCATION_TYPES et ON s.education_type_id = et.id
WHERE et.tipus_ensenyament = 'FPA'
GROUP BY c.centre_code, c.centre_name
ORDER BY num_students DESC;
```

```
gestmat=> EXPLAIN SELECT * FROM mv_enrollment_subjects
WHERE num_students < 4
ORDER BY num_students, nom_assignatura;
                                   QUERY PLAN
--------------------------------------------------------------------------------
 Sort  (cost=1103.35..1103.38 rows=12 width=78)
   Sort Key: num_students, nom_assignatura
   ->  Seq Scan on mv_enrollment_subjects  (cost=0.00..1103.14 rows=12 width=78)
         Filter: (num_students < 4)
(4 righe)

Tempo: 0,496 ms
gestmat=> EXPLAIN SELECT
    c.centre_code,
    c.centre_name,
    SUM(c.num_students) as num_students
FROM mv_enrollment_subjects c
JOIN SUBJECTS s ON c.id = s.id
JOIN EDUCATION_TYPES et ON s.education_type_id = et.id
WHERE et.tipus_ensenyament = 'FPA'
GROUP BY c.centre_code, c.centre_name
ORDER BY num_students DESC;
                                                                        QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------------------------------------
 Sort  (cost=1240.43..1240.64 rows=84 width=63)
   Sort Key: (sum(c.num_students)) DESC
   ->  GroupAggregate  (cost=1235.86..1237.75 rows=84 width=63)
         Group Key: c.centre_code, c.centre_name
         ->  Sort  (cost=1235.86..1236.07 rows=84 width=39)
               Sort Key: c.centre_code, c.centre_name
               ->  Hash Join  (cost=76.46..1233.17 rows=84 width=39)
                     Hash Cond: (c.id = s.id)
                     ->  Seq Scan on mv_enrollment_subjects c  (cost=0.00..997.71 rows=42171 width=43)
                     ->  Hash  (cost=76.39..76.39 rows=6 width=4)
                           ->  Hash Join  (cost=8.18..76.39 rows=6 width=4)
                                 Hash Cond: (s.education_type_id = et.id)
                                 ->  Seq Scan on subjects s  (cost=0.00..60.42 rows=2942 width=8)
                                 ->  Hash  (cost=8.17..8.17 rows=1 width=4)
                                       ->  Index Scan using education_types_tipus_ensenyament_key on education_types et  (cost=0.15..8.17 rows=1 width=4)
                                             Index Cond: ((tipus_ensenyament)::text = 'FPA'::text)
(16 righe)

Tempo: 1,045 ms
```

Refrescant les vistes

Les vistes materialitzades necessiten ser refrescades de tant en tant. Per a aquest propòsit, he creat una funció.

```sql
CREATE OR REPLACE FUNCTION refresh_materialized_views()
RETURNS void AS $$
BEGIN
    REFRESH MATERIALIZED VIEW CONCURRENTLY mv_unenrolled_students;
    REFRESH MATERIALIZED VIEW CONCURRENTLY mv_centre_locations;
    REFRESH MATERIALIZED VIEW CONCURRENTLY mv_centre_saturation;
    REFRESH MATERIALIZED VIEW CONCURRENTLY mv_enrollment_subjects;
END;
$$ LANGUAGE plpgsql;
```

4.5.1 Demostració d'Optimitzacions

Consulta	Temps d'Execució Antic	Temps d'Execució Nou	Millora del Temps		Millora de la Velocitat
a	0.795 seg	0.034 seg	-0.761 seg	95.73%	23.38x
b	5.609 seg	0.0007 seg	-5.6083 seg	99.99%	8012.26x
c	4.282 seg	0.001 seg	-4.281 seg	99.98%	4282x
d	0.715 seg	0.030 seg	-0.685 seg	95.80%	23.83x
e	0.542 seg	0.0006 seg	-0.5414 seg	99.89%	903.33x
f	8.968 seg	0.100 seg	-8.868 seg	98.88%	89.68x

Capítol 5

Comparació entre implementacions

5.1 Introducció

Aquesta secció analitza les diferències en la resolució de consultes i el rendiment entre MySQL i PostgreSQL. L'objectiu és avaluar l'execució de les consultes implementades per comprendre els avantatges i les limitacions de cada sistema de gestió de bases de dades.

5.2 Resolució de consultes: Característiques clau

MySQL i PostgreSQL difereixen significativament en el seu disseny i enfocament per resoldre consultes:

- **MySQL:** Conegut per la seva simplicitat i velocitat en l'execució de consultes senzilles, aprofitant una optimització agressiva de la memòria i el motor *InnoDB*, que utilitza arbres B+ per a l'indexació.
- **PostgreSQL:** Excel·leix en el compliment dels estàndards SQL i una arquitectura avançada per a consultes complexes, incloent-hi l'optimització de consultes basada en costos i el suport per a consultes paral·leles.

5.3 Comparació de rendiment: Consultes SQL

Execució a MySQL Per optimitzar el rendiment a MySQL, es van crear diversos índexs, com ara:

- Índex compost sobre `ENROLLMENTS(codi_centre, dni)`.
- Índex sobre `SUBJECTS(education_type_id)`.

Execució a PostgreSQL PostgreSQL va utilitzar funcions avançades com:

- Optimitzador de consultes basat en costos per a una selecció eficient dels plans.
- Ús d'índexs GIN i BRIN per a tipus de dades de text i rang, respectivament.

5.3.1 Comparació de temps

Comencem comparant els temps d'execució de cada consulta, prenent els valors després de les optimitzacions:

Consulta	Temps Exec. MySQL	Temps Exec. PostgreSQL	Diferència de Temps		Dif. de Velocitat
a	0.66 s	0.034 s	-0.626 s	94.85%	19.41x
b	2.32 s	0.0007 s	-2.319 s	99.97%	3314.29x
c	15.81 s	0.001 s	-15.809 s	99.99%	15810x
d	14.65 s	0.030 s	-14.62 s	99.80%	488.33x
e	3.86 s	0.0006 s	-3.8594 s	99.98%	6433.33x
f	2 min 47.71 s	0.100 s	-167.61 s	99.94%	1677.1x

És evident que PostgreSQL és molt més ràpid i ofereix moltes opcions diferents per optimitzar la base de dades i les consultes.

5.3.2 Diferències clau en el rendiment

Les diferències clau observades són:

1. **Eficiència dels índexs:** MySQL excel·leix en consultes amb índexs senzills, mentre que PostgreSQL utilitza eficientment índexs compostos i plans optimitzats.
2. **Execució paral·lela:** PostgreSQL admet l'execució paral·lela de consultes, oferint millor rendiment en sistemes multicore per a operacions complexes.
3. **Flexibilitat:** PostgreSQL admet funcions avançades com les vistes materialitzades, facilitant l'optimització de consultes en alguns casos.

5.4 Conclusions

L'elecció d'un sistema de gestió de bases de dades depèn dels requisits específics del sistema:

- MySQL és ideal per a sistemes amb càrregues de treball més senzilles i operacions predominantment de lectura.
- PostgreSQL és més adequat per a bases de dades amb consultes complexes i la necessitat d'una escalabilitat robusta.

Ambdós sistemes poden assolir un rendiment òptim amb la configuració i optimització adequades de les consultes.