SGBD TP3

1. Création de la base de données

```
CREATE DATABASE FBJ;
use FBJ;
CREATE TABLE FOURNISSEUR(
codfrs Varchar(20) PRIMARY KEY,
nomfrs Varchar(50),
villefrs Varchar(50),
telfrs Varchar(8)
);
CREATE TABLE DIRECTEUR(
coddirecteur Varchar(20) PRIMARY KEY,
nomdirecteur Varchar(50)
);
CREATE TABLE PROJET(
codproj Varchar(20),
nomproj Varchar(50),
villeproj Varchar(50),
budjetproj INT,
coddirecteur Varchar(2),
PRIMARY KEY (codproj),
FOREIGN KEY (coddirecteur) REFERENCES
DIRECTEUR(coddirecteur)
);
CREATE TABLE PIECE(
```

```
codpiece Varchar(20) PRIMARY KEY,
nompiece Varchar(50),
couleurpiece Varchar(50),
poidspiece DECIMAL(10,2),
villepiece Varchar(50)
);
CREATE TABLE FPJ(
codfrs Varchar(20),
codpiece Varchar(2),
codproj Varchar(2),
qtelivree INT,
dateliv DATE,
FOREIGN KEY (codfrs) REFERENCES
FOURNISSEUR(codfrs),
FOREIGN KEY (codpiece) REFERENCES
PIECE(codpiece),
FOREIGN KEY (codproj) REFERENCES
PROJET(codproj)
);
```

2. Insertion des données

```
INSERT INTO
FOURNISSEUR(codfrs,nomfrs,villefrs,telfrs)
VALUES
('F1','Alterna','Tunis','71001001'),
('F2','Butagaz','Tunis','71123123'),
('F3','EDF','Nabeul','72234432');
```

```
INSERT INTO
DIRECTEUR(coddirecteur,nomdirecteur)
VALUES
('D1','Sami ktata'),
('D2','Anis ksontini'),
('D3','Taieb Falfel');

INSERT INTO
PROJET(codproj,nomproj,villeproj,budjetproj,c
oddirecteur)
VALUES
('P1','Production GAZ','Gabes',2590000,'D2'),
('P2',"Production
electricité",'Touzeur',2055000,'D1'),
('P3','Production
GAZ','Tabarka',2040000,'D3');


INSERT INTO
PIECE(codpiece,nompiece,couleurpiece,poidspie
ce,villepiece)
VALUES
('P1',"Cable
d'alimentation",'Bleu',200,'Sfax'),
('P2',"Appariel de
mesure",'Blanc',0.44,'Sfax'),
('P3',"Alimententateur",'Noir',1.50,'Sfax');

INSERT INTO
```

```
FPJ(codfrs,codpiece,codproj,qtelivree,dateliv
)
VALUES
('F1','P1','P1',20,'2019-10-10'),
('F1','P2','P1',15,'2019-07-19'),
('F1','P3','P3',13,'2019-11-13');
```

3.

1.

```
SELECT DISTINCT FPJ.codpiece
FROM FPJ
JOIN FOURNISSEUR
ON FPJ.codfrs = FOURNISSEUR.codfrs
JOIN PROJET
ON FPJ.codproj = PROJET.codproj
WHERE
FOURNISSEUR.villefrs = PROJET.villeproj;

Empty set (0,002 sec)
```

On regarde la table FPJ et on match le codfrs avec le fournisseur codfrs dans la table fournisseur
On ajoute le PROJET pour coresspondre le codproj avec le FPJ codproj

l'instruction WHERE nous permet de verifier que c'est les mêmes villes.
2)

```
SELECT DISTINCT FPJ.codproj
FROM FPJ
JOIN FOURNISSEUR
ON FPJ.codfrs = FOURNISSEUR.codfrs
JOIN PROJET
ON FPJ.codproj = PROJET.codproj
WHERE
FOURNISSEUR.villefrs != PROJET.villeproj;
+---------+
| codproj |
+---------+
| P1      |
| P3      |
+---------+
```

On change juste le SELECT pour le codproj et une inégalité dans le WHERE par rapport au requête précédente
3)

```
SELECT DISTINCT FPJ.codproj
FROM FPJ
WHERE FPJ.codfrs = 'F3';
```

```
Empty set (0,001 sec)
```

On a pas pieces fourni par F3

4.

```
INSERT INTO PROJET (codproj, nomproj,
villeproj, budjetproj, coddirecteur)
VALUES ('P4', "Generation d'electricité ",
'Paris', 3000000, 'D2');
+---------+-----------------------------+-----
------+------------+--------------+
| codproj | nomproj                     |
villeproj | budjetproj | coddirecteur |
+---------+-----------------------------+-----
------+------------+--------------+
| P1      | Production GAZ              |
Gabes     |    2590000 | D2             |
| P2      | Production electricité      |
Touzeur   |    2055000 | D1             |
| P3      | Production GAZ              |
Tabarka   |    2040000 | D3             |
| P4      | Generation d electricité    | Paris
|    3000000 | D2             |
+---------+-----------------------------+-----
------+------------+--------------
```

```
SELECT P.codproj
FROM PROJET as P
WHERE P.nomproj LIKE '_E%';
+---------+
| codproj |
+---------+
| P4      |
+---------+
```

Utilisation de LIKE et "_" pour avoir 1 lettre apres la première lettre
5)

```
SELECT P.codproj
FROM PROJET as P
JOIN DIRECTEUR as D
ON D.coddirecteur = P.coddirecteur
WHERE D.nomdirecteur LIKE '%A';
+---------+
| codproj |
+---------+
| P2      |
+---------+
```

6.

```
SELECT codpiece, COUNT(*) AS
nombre_de_livraison
FROM FPJ
GROUP BY codpiece;
+---------+----------------------+
| codpiece | nombre_de_livraison |
+---------+----------------------+
| P1       |                    1 |
| P2       |                    1 |
| P3       |                    1 |
+---------+----------------------+
```

Utilisation de count dans FPJ pour compter le nombre de livraison et afficher par codpiece
7)

```
SELECT COUNT(*) AS livraison
FROM FPJ
WHERE codfrs = 'F2'
AND dateliv BETWEEN '2019-01-01' AND '2020-
01-01';

+-----------+
| livraison  |
+-----------+
|          0 |
```

```
+------------+
```

Utilisation de BETWEEN dans dateliv

8)

```
SELECT codpiece, MAX(qtelivree) AS
quantite_max
FROM FPJ
WHERE codproj = 'P1';
+----------+--------------+
| codpiece | quantite_max |
+----------+--------------+
| P1       |           20 |
+----------+--------------+
```

Utilisation de la fonction agrée MAX dans qtelivree

9)

```
INSERT INTO FPJ (codfrs, codpiece, codproj,
qtelivree, dateliv)
VALUES ('F1', 'P1', 'P2', 10, '2020-01-15');

SELECT p.poidspiece
FROM PIECE p
JOIN FPJ f ON p.codpiece = f.codpiece
GROUP BY p.codpiece, p.poidspiece
```

```
HAVING COUNT(*) > 1
ORDER BY COUNT(*) DESC;
+------------+
| poidspiece |
+------------+
|     200.00 |
+------------+
```

on insert dans fpj une livraison supplémentaire
On utilise count(*) > 1 pour compter les pieces

10.

```
SELECT codpiece
FROM PIECE
WHERE codpiece NOT IN (
    SELECT DISTINCT FPJ.codpiece
    FROM FPJ
    JOIN PROJET ON FPJ.codproj =
PROJET.codproj
    WHERE PROJET.villeproj = 'PARIS'
);
+----------+
| codpiece |
+----------+
| P1       |
| P2       |
```

```
| P3         |
+------------+
```

On recupere avec une sous requete toutes les valeurs de fpj où le projet est locaclisé dans paris et on fais un NOT IN pour récuperer ceux qui ne sont pas dans paris

11.

```
INSERT INTO FPJ (codfrs, codpiece, codproj,
qtelivree, dateliv)
VALUES
('F1', 'P1', 'P1', 20, '2019-10-10'), ('F1',
'P2', 'P1', 15, '2019-07-19'), ('F1', 'P3',
'P1', 13, '2019-11-13');

SELECT p.codproj
FROM PROJET p
WHERE NOT EXISTS (
    SELECT pc.codpiece
    FROM PIECE pc
    WHERE NOT EXISTS (
        SELECT f.codpiece
        FROM FPJ f
        WHERE f.codpiece = pc.codpiece AND
f.codproj = p.codproj
    )
```

```
);
+---------+
| codproj |
+---------+
| P1      |
+---------+
```

Le NOT EXISTS à l'intérieur permet de confirmer que copiece existe dans FPJ pour le codproj
le NOT EXISTS à l'extérieur permet d'etre sur que toutes les pieves ont été délivré au projet
12)

```
UPDATE PROJET p
SET budjetproj = budjetproj + 1000
WHERE p.villeproj = 'PARIS'
AND (
    SELECT COUNT(*)
    FROM FPJ f
    JOIN FOURNISSEUR fr ON f.codfrs =
fr.codfrs
    WHERE f.codproj = p.codproj AND
fr.villefrs != 'PARIS'
) > 10;


Query OK, 0 rows affected (0,006 sec)
```

```
Rows matched: 0  Changed: 0  Warnings: 0
```

la sous requête compte le nombre de livraison pour chaque projet où la ville du fournisseur
n'est pas Paris
avec >10 on accepte les livraison > 10
avec set on augumente de 1000 le budget
13)

```
INSERT INTO PIECE (codpiece, nompiece,
couleurpiece, poidspiece, villepiece)
VALUES
('P4', 'Support métallique', 'Rouge', 2.50,
'Sfax'),
('P5', 'Tube d’échappement', 'Rouge', 1.75,
'Tunis');

UPDATE PIECE
SET couleurpiece = 'Orange'
WHERE couleurpiece = 'Rouge';

 SELECT * FROM PIECE;
 +---------+----------------------+--------
------+------------+------------+
| codpiece | nompiece                |
couleurpiece | poidspiece | villepiece |
```

```
+---------+----------------------+---------
-----+-----------+-----------+
| P1      | Cable dalimentation  | Bleu
|    200.00 | Sfax       |
| P2      | Appariel de mesure    | Blanc
|      0.44 | Sfax       |
| P3      | Alimententateur       | Noir
|      1.50 | Sfax       |
| P4      | Support métallique    | Orange
|      2.50 | Sfax       |
| P5      | Tube d’échappement    | Orange
|      1.75 | Tunis      |
+---------+----------------------+---------
-----+-----------+-----------+
```

On insere des pieces pour tester
puis on update avec le changement et on affiche.
14)

```
DELETE FROM PROJET
WHERE codproj NOT IN (
    SELECT DISTINCT codproj
    FROM FPJ
);
 SELECT * FROM PROJET;
+--------+-------------------------+--------
---+-----------+-------------+
```

```
| codproj | nomproj                    |
villeproj | budjetproj | coddirecteur |
+--------+-------------------------+--------
---+-----------+-------------+
| P1      | Production GAZ           | Gabes
|    2590000 | D2            |
| P2      | Production electricité   | Touzeur
|    2055000 | D1            |
| P3      | Production GAZ           | Tabarka
|    2040000 | D3            |
+--------+-------------------------+--------
---+-----------+-------------+
```

Suppression des projets qui ne sont pas dans les projets livrées donc FPJ
15)

```
INSERT INTO PIECE (codpiece, nompiece,
couleurpiece, poidspiece, villepiece)
VALUES
('P8', 'Rouge connecteur', 'Rouge', 3.00,
'Tunis'),
('P9', 'Support en acier', 'Rouge', 2.20,
'Gabes');
INSERT INTO FPJ (codfrs, codpiece, codproj,
qtelivree, dateliv)
```

```
VALUES
('F2', 'P8', 'P1', 25, '2023-01-15'),
('F3', 'P9', 'P2', 40, '2023-02-20');

UPDATE FPJ f
SET qtelivree = qtelivree * 1.10
WHERE f.codpiece IN (
    SELECT p.codpiece
    FROM PIECE p
    WHERE p.couleurpiece = 'Rouge'
);



SELECT *
FROM FPJ
WHERE codpiece IN (
    SELECT codpiece
    FROM PIECE
    WHERE couleurpiece = 'Rouge'
);
+--------+----------+---------+-----------+--
----------+
| codfrs | codpiece | codproj | qtelivree |
dateliv    |
+--------+----------+---------+-----------+--
----------+
| F2     | P8       | P1      |        28 |
2023-01-15 |
```

```
| F3     | P9        | P2       |          44 |
2023-02-20 |
+--------+-----------+----------+-------------+--
-----------+
```

La sous requete recupere toutes les pieces rouges et le where fait en sorte que seulement les livraison des pieces rouges sont mis à jour

Exercice 2 :

A)

```sql
CREATE DATABASE ImmeubleDB;
USE ImmeubleDB;


CREATE TABLE Immeuble (
    id INT PRIMARY KEY,
    adrNum VARCHAR(10),
    adrVoie VARCHAR(50),
    adrCodePostal VARCHAR(10),
    adrVille VARCHAR(50),
    fibreOptique BOOLEAN,
    parkingPrivatif BOOLEAN
);
```

```sql
CREATE TABLE Appartement (
    immeuble INT,
    num INT,
    description TEXT,
    loyer DECIMAL(10,2),
    superficie DECIMAL(10,2),
    terrasse BOOLEAN,
    classeConso VARCHAR(5),
    chauffage VARCHAR(50),
    placeParking BOOLEAN,
    prixParking DECIMAL(10,2),
    PRIMARY KEY (immeuble, num),
    FOREIGN KEY (immeuble) REFERENCES
Immeuble(id)
);


CREATE TABLE Piece (
    immeuble INT,
    appartement INT,
    num INT,
    superficie DECIMAL(10,2),
    fonction VARCHAR(50),
    PRIMARY KEY (immeuble, appartement, num),
    FOREIGN KEY (immeuble, appartement)
REFERENCES Appartement(immeuble, num)
);
```

```sql
CREATE TABLE Photo (
    num INT PRIMARY KEY,
    titre VARCHAR(100),
    description TEXT,
    uri VARCHAR(255),
    immeuble INT,
    appartement INT,
    FOREIGN KEY (immeuble, appartement)
REFERENCES Appartement(immeuble, num)
);
```

B)

```sql
DELIMITER //

CREATE TRIGGER TestPrixParking
BEFORE INSERT ON Appartement
FOR EACH ROW
BEGIN
    IF NEW.placeParking = FALSE AND
NEW.prixParking IS NOT NULL THEN
        SET NEW.prixParking = NULL;
    END IF;
END //

DELIMITER ;
```

```
INSERT INTO Immeuble (id, adrNum, adrVoie,
adrCodePostal, adrVille, fibreOptique,
parkingPrivatif)
VALUES (1, '10', 'Rue de Paris', '75001',
'Paris', TRUE, TRUE);


INSERT INTO Appartement (immeuble, num,
description, loyer, superficie, terrasse,
classeConso, chauffage, placeParking,
prixParking)
VALUES (1, 101, 'Appartement 101', 1200.00,
50.00, TRUE, 'A', 'Gaz', FALSE, 150.00);

+----------+-----+------------------+---------
+-----------+----------+------------+------
-----+--------------+-------------+
| immeuble | num | description      | loyer
| superficie | terrasse | classeConso |
chauffage | placeParking | prixParking |
+----------+-----+------------------+---------
+-----------+----------+------------+------
-----+--------------+-------------+
|        1 | 101 | Appartement 101 | 1200.00
|      50.00 |         1 | A            | Gaz
|            0 |         NULL |
```

```
+----------+-----+-----------------+---------
+-----------+---------+-------------+------
-----+--------------+-------------+

SELECT * FROM Appartement;
```

Le trigger vérifie et assure que si la placeparking est false alors il met prixparking a NULL
et On a bien NULL dans prixParking avec notre exemple.
C)

```
DELIMITER //

CREATE TRIGGER UpdateSuperfie
AFTER INSERT ON Piece
FOR EACH ROW
BEGIN
    UPDATE Appartement
    SET superficie = (
        SELECT SUM(superficie)
        FROM Piece
        WHERE immeuble = NEW.immeuble AND
appartement = NEW.appartement
    )
    WHERE immeuble = NEW.immeuble AND num =
NEW.appartement;
```

```
END //

DELIMITER ;
```

Chaque fois qu'une pièce est ajouté à un appartement on met à jour superfice
D)

```
DELIMITER //

CREATE TRIGGER UpdateSuperficieOnPieceDelete
AFTER DELETE ON Piece
FOR EACH ROW
BEGIN
    UPDATE Appartement
    SET superficie = (
        SELECT SUM(superficie)
        FROM Piece
        WHERE immeuble = OLD.immeuble AND
appartement = OLD.appartement
    )
    WHERE immeuble = OLD.immeuble AND num =
OLD.appartement;
END //

DELIMITER ;
```

On fait pariel pour les mis à jour de pièce ou suppresion

E)

```
DELIMITER //

CREATE TRIGGER CheckParkingetSuperficie
BEFORE INSERT ON Appartement
FOR EACH ROW
BEGIN
    IF (SELECT parkingPrivatif FROM Immeuble
WHERE id = NEW.immeuble) = FALSE THEN
        SET NEW.placeParking = FALSE;
        SET NEW.prixParking = NULL;
    END IF;


    SET NEW.superficie = 0;
END //

DELIMITER ;

DELIMITER //

CREATE TRIGGER EmpecheSuperficieModification
BEFORE UPDATE ON Appartement
FOR EACH ROW
BEGIN
```

```
    IF OLD.superficie != NEW.superficie THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = "Modifcation de la
superficie n'est pas autorisé!";
    END IF;
END //

DELIMITER ;
```

Test des triggers

```
INSERT INTO Immeuble (id, adrNum, adrVoie,
adrCodePostal, adrVille, fibreOptique,
parkingPrivatif)
VALUES (2, '20', 'Rue Lyon', '69001', 'Lyon',
TRUE, FALSE);

INSERT INTO Appartement (immeuble, num,
description, loyer, superficie, terrasse,
classeConso, chauffage, placeParking,
prixParking)
VALUES (2, 201, 'Appartement 201', 1500.00,
0.00, FALSE, 'B', 'Electrique', TRUE, NULL);

INSERT INTO Piece (immeuble, appartement,
num, superficie, fonction)
VALUES (2, 201, 1, 25.00, 'Salon'),
       (2, 201, 2, 30.00, 'Chambre');
```

```
-- AFFICHAGE CONSOLE--
INSERT INTO Piece (immeuble, appartement,
num, superficie, fonction)
    -> VALUES (2, 201, 1, 25.00, 'Salon'),
    ->        (2, 201, 2, 30.00, 'Chambre');
ERROR 1644 (45000): Modifcation de la
superficie n'est pas autorisé!'
--------------------------------


SELECT * FROM Appartement WHERE immeuble = 2
AND num = 201;
+----------+-----+------------------+---------
+-----------+----------+-------------+------
------+--------------+-------------+
| immeuble | num | description      | loyer
| superficie | terrasse | classeConso |
chauffage  | placeParking | prixParking |
+----------+-----+------------------+---------
+-----------+----------+-------------+------
------+--------------+-------------+
|        2 | 201 | Appartement 201 | 1500.00
|       0.00 |         0 | B             |
Electrique |             0 |         NULL |
+----------+-----+------------------+---------
+-----------+----------+-------------+------
------+--------------+-------------+
```

```
UPDATE Appartement
SET superficie = 100
WHERE immeuble = 2 AND num = 201;
->
ERROR 1644 (45000): Modifcation de la
superficie n'est pas autorisé!'
```