



INF3710 –Bases de données

Hiver 2024

TP No. 1

Groupe 03



Soumis à

Samedi 20 Janvier 2024

1) Création d'une base de données

1. Créez la base de données nommée « BD_TP1 » avec une requête SQL. **0.25pt**
Quelle commande avez-vous utilisé?

L'utilisation de l'interface graphique n'est pas permise pour cette commande.

1/12/2024		501 msec
1:26:00 PM		
Date	Rows affected	Duration
Copy	Copy to Query Editor	
CREATE DATABASE BD_TP1;		
Messages		
Query returned successfully in 501 msec.		

2. Donnez la requête SQL nécessaire pour la création d'une table nommée *Tache*, cette table contient un numéro de tache, l'identifiant de l'employé qui effectue la tâche, l'identifiant du projet auquel la tache appartient, la description de la tache (ne peut pas être nulle) et la date limite pour cette tâche. **0.50pt**

SQL :

```
CREATE TABLE Tache (  
    noTache SERIAL PRIMARY KEY,  
    idEmploye INT REFERENCES Employe(idEmploye),  
    idProjet INT REFERENCES Projet(idProjet),  
    descriptionTache VARCHAR(255) NOT NULL,  
    dateLimite DATE  
);
```

2) Création des tables

3. En se basant sur le script dans le fichier « BD-TP1-schema », fourni sur Moodle, et en ajoutant votre nouvelle requête de la question 2, créez les 5 tables (*EmployeProjet*, *Departement*, *Adresse*, *Employe*, *Tache et Projet*) en exécutant les commandes

CREATE

TABLE.

Justifiez l'ordre de création. **0.50pt**

Réponse : Les tables *Departement* et *Projet* ont aucune référence à d'autres tables, donc elles peuvent être créées en premier. Ensuite, la table *Employe* devrait être créée, puisqu'elle est référencée par les trois autres tables (*EmployeProjet*, *Adresse* et *Tache*). Finalement, on crée les tables *EmployeProjet*, *Adresse* et *Tache*, dans n'importe quel ordre, puisqu'elles ne se référencent pas entre-elles.

SQL:

```
CREATE TABLE Departement (  
    idDepartement SERIAL PRIMARY KEY,  
    nomDepartement VARCHAR(255) NOT NULL  
);  
CREATE TABLE Projet (  
    idProjet SERIAL PRIMARY KEY,  
    nomProjet VARCHAR(255) NOT NULL  
);  
CREATE TABLE Employe (  
    idEmploye SERIAL PRIMARY KEY,  
    idDepartement INT REFERENCES Departement(idDepartement),  
    nomEmploye VARCHAR(255) NOT NULL,  
    salaire DECIMAL(10, 2) NOT NULL  
);  
CREATE TABLE EmployeProjet (  
    idEmploye INT REFERENCES Employe(idEmploye),  
    idProjet INT REFERENCES Projet(idProjet),  
    PRIMARY KEY (idEmploye, idProjet)  
);  
CREATE TABLE Adresse (  
    idAdresse SERIAL PRIMARY KEY,  
    idEmploye INT REFERENCES Employe(idEmploye),  
    rue VARCHAR(255) NOT NULL,  
    ville VARCHAR(255) NOT NULL,  
    etat VARCHAR(255),  
    codePostal VARCHAR(10) NOT NULL  
);  
CREATE TABLE Tache (  
    noTache SERIAL PRIMARY KEY,  
    idEmploye INT REFERENCES Employe(idEmploye),
```

```
idProjet INT REFERENCES Projet(idProjet),
descriptionTache VARCHAR(255) NOT NULL,
dateLimite DATE
);
Résultat :
```

Messages

Query returned successfully in 103 msec.

4. Si vous essayez de créer une deuxième fois la table *Departement* avec la commande suivante

```
CREATE TABLE Departement (
  idDepartement SERIAL PRIMARY KEY,
  nomDepartement VARCHAR(255) NOT NULL
);
```

Que se passe-t-il?

Quel est la cause à votre avis?

Quelle mot clef manque-t-il? **0.50pt**

Réponse : On obtient une erreur, puisque la table département existe déjà. Pour régler cette erreur, il suffit d'ajouter le mot clef IF NOT EXISTS à la commande. La nouvelle commande est la suivante :

SQL :

```
CREATE TABLE IF NOT EXISTS Departement (
  idDepartement SERIAL PRIMARY KEY,
  nomDepartement VARCHAR(255) NOT NULL
);
```

Résultat :

1/12/2024 3:44:59 PM	68 msec
Date	Rows affected

Copy Copy to Query Editor

```
CREATE TABLE Departement (
  idDepartement SERIAL PRIMARY KEY,
  nomDepartement VARCHAR(255) NOT NULL
);
```

Messages

ERROR: relation "departement" already exists

SQL state: 42P07

1/12/2024 3:16:08 PM	48 msec
Date	Rows affected

Copy Copy to Query Editor

```
CREATE TABLE IF NOT EXISTS Departement (
  idDepartement SERIAL PRIMARY KEY,
  nomDepartement VARCHAR(255) NOT NULL
);
```

Messages

Query returned successfully in 48 msec.

3) Peuplement des tables

5. En se basant toujours sur la base de données « BD-TP1 », créez les requêtes SQLs nécessaires pour peupler les tables dans l'ordre (*Insérez au moins 2 ensembles de données dans chaque table*). **0.50pt**

Departement :

SQL :

```
INSERT INTO Departement VALUES (DEFAULT, 'gigl');
```

```
INSERT INTO Departement VALUES (DEFAULT, 'chimique');
```

Résultat :

Messages

```
Successfully run. Total query runtime: 144 msec. 2  
rows affected.
```

Projet :

SQL :

```
INSERT INTO Projet VALUES (DEFAULT, 'Proj1');
```

```
INSERT INTO Projet VALUES (DEFAULT, 'Proj2');
```

Résultat :

Messages

```
Successfully run. Total query runtime: 68 msec. 2 rows  
affected.
```

Employe :

SQL :

```
INSERT INTO Employe VALUES (DEFAULT, 1, 'Omar', 100000.01);
```

```
INSERT INTO Employe VALUES (DEFAULT, 2, 'Alex', 200000.02);
```

Résultat :

Messages

```
Successfully run. Total query runtime: 100 msec. 2  
rows affected.
```

EmployeProjet:

SQL :

```
INSERT INTO EmployeProjet VALUES (1, 1);
```

```
INSERT INTO EmployeProjet VALUES (2, 2);
```

Résultat :

Messages

```
Successfully run. Total query runtime: 67 msec. 2 rows  
affected.
```

Adresse :

SQL :

```
INSERT INTO Adresse VALUES (DEFAULT, 1, 'stecath', 'mtl', 'QC', 'H2M9IS');
```

```
INSERT INTO Adresse VALUES (DEFAULT, 2, 'byward', 'ott', 'ON', 'H4S7SA');
```

Résultat :

Messages

Successfully run. Total query runtime: 67 msec. 2 rows affected.

Tache :

SQL :

```
INSERT INTO Tache VALUES (DEFAULT, 1, 1, 'code', '2023-12-24');
```

```
INSERT INTO Tache VALUES (DEFAULT, 2, 2, 'compil', '2023-12-25');
```

Résultat :

Messages

Successfully run. Total query runtime: 74 msec. 2 rows affected.

6. Expérimentez la commande SQL :




```
SELECT * FROM Employe;
```

A quoi sert cette commande ? **0.10pt**

Réponse:

La commande est utilisée pour récupérer toutes les colonnes de toutes les lignes de la table "Employe" et l'afficher en format tableau.

Résultat :

Data Output Messages Explain × Notifications			
			
	idprojet [PK] integer 	nomprojet character varying (255) 	
1	1	Proj1	
2	2	Proj2	

7. Fournissez la requête utilisée pour détruire la table « *EmployeProjet* » et celle pour détruire la table « *Employe* ». Si les requêtes SQL pour supprimer ces deux tables sont différentes, expliquez pourquoi. Chaque opération doit utiliser une seule requête seulement. **0.50pt**

Réponse:

Pour détruire la table *EmployeProjet*, on utilise la commande `DROP TABLE EmployeProjet`. Pour détruire la table *Employe*, on doit rajouter le mot clef `CASCADE`, puisque d'autres tables dépendent de celle-ci. La nouvelle commande est : `DROP TABLE Employe CASCADE`.

SQL/Résultat:

1/12/2024 2:38:04 PM	54 msec
Date	Rows affected
	Duration

Copy	Copy to Query Editor
------	----------------------

```
DROP TABLE EmployeProjet;
```

Messages

Query returned successfully in 54 msec.

1/12/2024 2:38:23 PM	136 msec
Date	Rows affected
	Duration

Copy	Copy to Query Editor
------	----------------------

```
DROP TABLE Employe CASCADE;
```

Messages

Query returned successfully in 136 msec.