1ere année en BUT informatique a l'iut de sorbonne paris nord
J-B Clément - Université, 93430 Villetaneuse

SAE 2.04 : Exploitation d'une base de données

Rapport

Réalise par : Oumsid Hiba, Bouamlat Nour El Houda et Medani Lina

Enseignant : Mr. Abir Hocine

Rendu le : 23/05/2023

Cahier des charges

Au cours de notre première année en BUT informatique, nous avons étudié la création et la manipulation de bases de données à l'aide de requêtes SQL sur PostgreSQL. Nous avons acquis les compétences nécessaires pour concevoir des bases de données, définir leurs structures et effectuer des opérations telles que l'insertion, la mise à jour, la suppression etc.

Ainsi que nous avons appris l'importance de protéger les informations sensibles stockées dans une base de données. Cette SAE a pour objectif :

1. L'étude d'un modèle de données pour mettre en place une base de données de gestion des notes des étudiants en BUT

2. L'étude et la mise en œuvre de la gestion des données dérivées : relevé de notes, bilans…

3. L'étude et la mise en œuvre des restrictions d'accès à ces données : étudiant, enseignant, responsable de matière, etc. Il sera organisé en 3 parties successives comme décrit dans ce qui suit.

Tables des matières

I. Modélisation de Données

1. Modèle de la base de données

La situation de modélisation de la base de données est la suivante **:**

- Une table **Module** qui contient :
  - **Id_Module** : identifiant unique du module de type Integer qui est aussi une clé primaire.
  - **Code** : chaine de caractère qui contient le code du module.
  - **UE** : chaine de caractère qui contient l'unité d'enseignement à laquelle le module appartient.
  - **Intitulé** : chaine de caractère qui contient le nom du module.

- Une table **Enseignant** qui contient :
  - **Id_enseignant** : identifiant unique de l'enseignant de type Integer qui est aussi une clé primaire.
  - **Nom_enseignant** : une chaîne de caractères contenant le nom du professeur.
  - **Prénom_enseignant** : une chaîne de caractères contenant le prénom de l'enseignant.
  - **Id_Module** : une clé étrangère qui fait référence à la table Module.

- Une table **Etudiant** qui contient :
  - **Id_étudiant** : identifiant unique de l'étudiant de type Integer qui est aussi une clé primaire.
  - **Nom_étudiant** : une chaîne de caractères contenant nom de l'étudiant.
  - **Prénom_étudiant** : une chaîne de caractères contenant prénom de l'étudiant.
  - **Groupe** : une chaîne de caractères contenant le groupe de chaque étudiant.

- Une table **Evaluation** qui contient :
  - **Id_Evaluation** : identifiant unique de l'évaluation de type Integer qui est aussi la première partie de la clé primaire.
  - **Id_module** : clé étrangère qui fait référence au module auquel l'évaluation est associée.
  - **Nom_évaluation** : une chaîne de caractères contenant le nom de l'évaluation.
  - **Date** : date de l'évaluation de type Date.

- Une table **Résultat** qui contient :
  - **Id_etudiant** : la première partie de la clé primaire, elle fait référence à l'étudiant qui a passé l'évaluation.
  - **Id_Evaluation** : la deuxième partie de la clé primaire, elle fait référence à l'évaluation à laquelle la note est attribuée.
  - **Note** : note obtenue par l'étudiant à l'évaluation.

a. Règles de gestion de la base de données

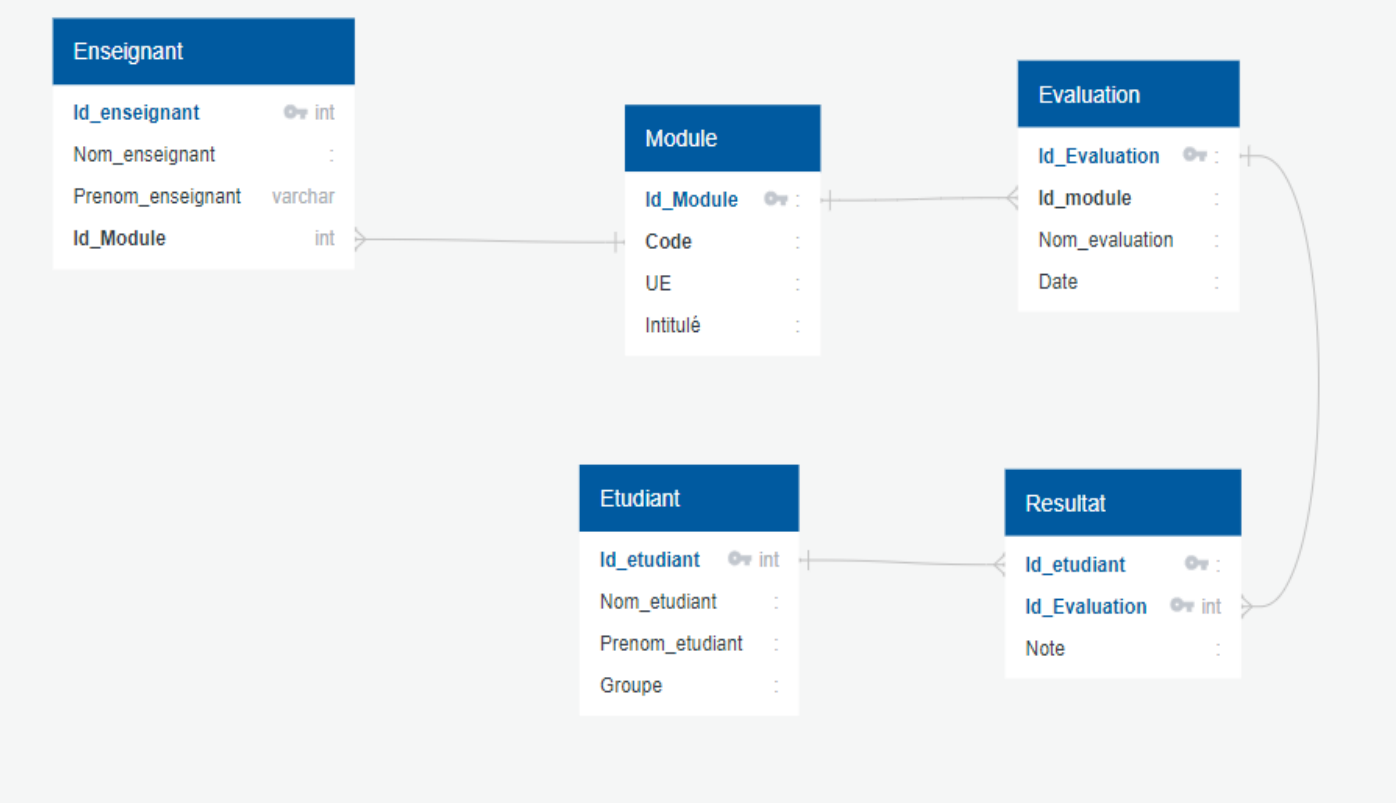
Notre base de données possède trois groupes d'utilisateurs : **SuperAdmin**, **étudiant** et **enseignant**. Les règles d'accès à cette BD sont définies en ce qui suit :

- Le groupe **SuperAdmin** a tous les droits sur toute la BD.

- Le groupe **enseignant** à tous les droits sur la table **Résultat**, il peut accéder aux autres tables mais il ne peut pas modifier.
- Le groupe **étudiant** n'a aucun droit sur les tables **Enseignant** et **Module**, or il peut voir ses notes dans la table **Résultat** et ses informations sur la table **Etudiant.**

b. Schéma de la base de données

Le schéma de notre base de données est le suivant :



c. Script de la base de données

- La création des tables :

```
-- Création de la table Enseignant
CREATE TABLE Enseignant (
      Id_enseignant INTEGER PRIMARY KEY,
      Nom_enseignant VARCHAR,
      Prénom_enseignant VARCHAR ?
      FOREIGN KEY (Id_Module) REFERENCES Module(Id_Module)
);
-- Création de la table Etudiant
CREATE TABLE Etudiant (
  Id_etudiant INTEGER PRIMARY KEY,
  Nom_etudiant VARCHAR,
  Prénom_etudiant VARCHAR );
```

```sql
-- Création de la table Module
CREATE TABLE Module (
     Id_Module INTEGER PRIMARY KEY,
     code VARCHAR,
     intitulé VARCHAR
);

-- Création de la table Evaluation
CREATE TABLE Evaluation (
  Id_evaluation INTEGER PRIMARY KEY,
  Nom_evaluation VARCHAR,
  Date DATE,
  Id_Module INTEGER,
  FOREIGN KEY (Id_Module) REFERENCES Module(Id_Module)
);
-- Création de la table resultat
CREATE TABLE resultat (
  Id_etudiant INTEGER,
  Id_evaluation INTEGER,
  Note DECIMAL,
  PRIMARY KEY (Id_etudiant, Id_evaluation),
  FOREIGN KEY (Id_etudiant) REFERENCES Etudiant (Id_etudiant),
  FOREIGN KEY (Id_evaluation) REFERENCES Evaluation (Id_evaluation)
);
```

- L’insertion des données :

```sql
INSERT INTO Enseignant (Id_enseignant, Nom_enseignant, Prenom_enseignant,
Id_Module)
VALUES (1, 'Abir', 'Hocine',1),
       (2, 'Kritsikis', 'Evaggelos',2),
       (3, 'Azzag', 'Hanene',2),
       (4, 'Martinez', 'Antony',4),
       (5, 'Bosc', 'Marcel',3),
       (6, 'Capdeville', 'Valérie',4);
INSERT INTO Module (Id_Module, Intitule, code)
VALUES (1, 'SQL', 'R2.06'),
       (2, 'Java', 'R2.01'),
       (3, 'Js', 'R2.02'),
       (4, 'Anglais', 'R2.12');

INSERT INTO Etudiant (Id_etudiant, Nom_etudiant, Prenom_etudiant, Groupe)
VALUES (1, 'Oumsid', 'Hiba', 'A'),
       (2, 'Bouamlat', 'Nour', 'A'),
       (3, 'Medani', 'Lina', 'A'),
       (4, 'Kim', 'William', 'B'),
       (5, 'Dupont', 'Céline', 'B'),
```

```
       (6, 'Antoine', 'Sophie', 'B');

INSERT INTO Evaluation (Id_evaluation, Id_Module, Nom_evaluation, Date)
VALUES (1, 1, 'contrôle', '2023-02-20'),
       (2, 2, 'Examen', '2023-04-23'),
       (3, 3, 'contrôle', '2023-04-23'),
       (4, 4, 'contrôle', '2023-03-15');

INSERT INTO Resultat (Id_etudiant, Id_evaluation, Note)
VALUES (1, 1, 15),
       (2, 1, 10),
       (3, 1, 12),
       (4, 1, 12),
       (5, 1, 10),
       (6, 1, 9),
       (6, 2, 6),
       (5, 2, 14),
       (4, 2, 10),
       (3, 2, 12),
       (2, 2, 13),
       (1, 2, 13),
       (1, 3, 13),
       (2, 3, 12),
       (3, 3, 15),
       (4, 3, 10),
       (5, 3, 11),
       (6, 3, 12),
       (6, 4, 9),
       (5, 4, 5),
       (4, 4, 12),
       (3, 4, 11),
       (2, 4, 12),
       (1, 4, 11);
```

Le résultat après une requête select donne l'affichage suivant :

```
SELECT * FROM Etudiant;
```

```
Id_etudiant | Nom_etudiant | Prenom_étudiant | Groupe
-------------------------------------------------
1           | Oumsid       | Hiba            | A
2           | Bouamlat     | Nour            | A
3           | Medani       | Lina            | A
4           | Kim          | William         | B
5           | Dupont       | Céline          | B
6           | Antoine      | Sophie          | B
```

```sql
SELECT * FROM Enseignant;
```

Id_enseignant	Nom_enseignant	Prenom_enseignant	Id_Module
1	Abir	Hocine	1
2	Kritsikis	Evaggelos	2
3	Azzag	Hanene	2
4	Martinez	Antony	4
5	Bosc	Marcel	3
6	Capdeville	Valérie	4

```sql
SELECT * FROM Module;
```

Id_Module	Intitule	code
1	SQL	R2.06
2	Java	R2.01
3	Js	R2.02
4	Anglais	R2.12

```sql
SELECT * FROM Resultat;
```

Id_etudiant	Id_Evaluation	Note
1	1	15
2	1	10
3	1	12
4	1	12
5	1	10
6	1	9
6	2	6
5	2	14
4	2	10
3	2	12
2	2	13
1	2	13
1	3	13
2	3	12
3	3	15

```
3          | 3          | 15
4          | 3          | 10
5          | 3          | 11
6          | 3          | 12
6          | 4          | 9
5          | 4          | 5
4          | 4          | 12
3          | 4          | 11
2          | 4          | 12
1          | 4          | 11
```

II. Visualisation de données

1-Relevé de notes des étudiants

```
CREATE OR REPLACE FUNCTION Releve_Etudiant(IN id_etudiant INT, OUT
nom_etudiant VARCHAR, OUT prenom_etudiant VARCHAR, OUT  groupe VARCHAR, OUT
id_module INT, OUT id_evaluation INT, OUT nomevaluation VARCHAR, OUT note
FLOAT) RETURNS SETOF RECORD AS
$$
SELECT Etudiant.Nométudiant, Etudiant.Groupe, Etudiant.Prénom_étudiant,
Evaluation.Id_evaluation, Module.IdModule, Evaluation.Noméval, resultat.Note
FROM Etudiant
JOIN resultat ON Etudiant.Id_étudiant = resultat.Id_Etudiant
JOIN Evaluation ON resultat.Id_Evaluation = Evaluation.Id_evaluation
JOIN Module ON Evaluation.Id_Module = Module.IdModule
WHERE Etudiant.Idétudiant = $1
ORDER BY Evaluation.Id_evaluation;
$$ LANGUAGE SQL;
```

Avec la fonction Releve_Etudiant, nous pouvons obtenir toutes les notes de l'étudiant en question juste en précisant son numéro d'étudiant lorsqu'on utilise cette fonction.

Si on essaie avec cette requête :

```
SELECT * FROM Releve_Etudiant(2);
```

Le résultat de cette fonction sera:

nom_etudiant	groupe	prenom_etudiant	id_module	id_evaluation	nomevaluation	note
Bouamlat	A	Nour	2	1	contrôle	10
Bouamlat	A	Nour	2	2	Examen	13
Bouamlat	A	Nour	2	3	contrôle	12
Bouamlat	A	Nour	2	4	contrôle	10

2- Relevé de notes du groupe :

```
CREATE OR REPLACE FUNCTION Releve_Groupe(IN groupe VARCHAR, OUT nom_etudiant
VARCHAR, OUT prenom_etudiant VARCHAR, OUT groupe_etudiant VARCHAR, OUT
id_module INT, OUT id_evaluation INT, OUT nom_evaluation VARCHAR, OUT note
FLOAT) RETURNS SETOF RECORD AS
$$
SELECT Etudiant.Nom_etudiant, Etudiant.Prenom_etudiant, Etudiant.Groupe,
Module.Id_Module, Evaluation.Id_evaluation, Evaluation.Nom_evaluation,
Resultat.Note
FROM Etudiant
JOIN Resultat ON Etudiant.Id_etudiant = Resultat.Id_etudiant
JOIN Evaluation ON Resultat.Id_evaluation = Evaluation.Id_evaluation
JOIN Module ON Evaluation.Id_module = Module.Id_module
WHERE Etudiant.Groupe = $1
ORDER BY Evaluation.Id_evaluation;
$$ LANGUAGE SQL;
```

Cette fonction nous donne toutes les informations sur tous les étudiants d'un groupe, on obtient donc presque les mêmes informations que le relevé de note individuelle mais avec tous ceux du groupe.

Si on fait cet exemple :

```
SELECT * FROM Releve_Groupe('A');
```

Le résultat sera le suivant :

nom_etudiant	prenom_etudiant	groupe_etudiant	id_module	id_evaluation	nom_evaluation	note
Oumsid	Hiba	A	1	1	contrôle	15
Oumsid	Hiba	A	1	2	Examen	13
Oumsid	Hiba	A	1	3	contrôle	13
Oumsid	Hiba	A	1	4	contrôle	11
Bouamlat	Nour	A	2	1	contrôle	10
Bouamlat	Nour	A	2	2	Examen	14
Bouamlat	Nour	A	2	3	contrôle	12
Bouamlat	Nour	A	2	4	contrôle	10
Medani	Lina	A	3	1	contrôle	12
Medani	Lina	A	3	2	Examen	11
Medani	Lina	A	3	3	contrôle	13
Medani	Lina	A	3	4	contrôle	12

3- Moyenne de l'étudiant :

```
CREATE OR REPLACE FUNCTION Moyenne_Etudiant(IN id_etudiant INT, OUT
id_etudiant INT, OUT id_Module VARCHAR, OUT moyenne FLOAT)
RETURNS SETOF RECORD AS
$$
SELECT Resultat.id_Etudiant, Evaluation.id_Module, AVG(Resultat.note)
FROM Resultat
JOIN Evaluation ON Resultat.id_Evaluation = Evaluation.id_Evaluation
WHERE Resultat.id_Etudiant = $1
GROUP BY Resultat.id_Etudiant, Evaluation.id_Module;
$$ LANGUAGE SQL;
```

Avec cette fonction on peut obtenir la moyenne par module d'un étudiant seulement avec son numéro étudiant, on aura accès à son id, sa moyenne et le module dont il a cette moyenne.

Si on teste avec cet exemple :

```
SELECT * FROM Moyenne_Etudiant(1);
```

Le résultat sera :

id_etudiant	id_Module	moyenne
1	1	11.83
1	2	12.17
1	3	10.67
1	4	12.00

4-La moyenne du groupe :

```
CREATE OR REPLACE FUNCTION Moyenne_Groupe(IN groupe VARCHAR, OUT Groupe
VARCHAR, OUT  idModule VARCHAR, OUT  Moyenne FLOAT)
RETURNS SETOF RECORD AS
$$
```

```
SELECT Etudiant.Groupe, AVG(Resultat.Note), Evaluation.id_Module
FROM Resultat
LEFT JOIN Evaluation ON Resultat.id_Evaluation = Evaluation.id_Evaluation
LEFT JOIN Etudiant ON Resultat.id_Etudiant = Etudiant.Id_etudiant
WHERE Etudiant.Groupe = $1
GROUP BY Evaluation.id_Module, Etudiant.Groupe;
$$ LANGUAGE SQL;
```

Avec cette fonction on peut obtenir la moyenne par module d'un groupe seulement avec le nom du groupe, on aura accès au nom du groupe, la moyenne et le module dont le groupe a cette moyenne.

Voici un exemple :

```
SELECT * FROM Moyenne_Groupe('A');
```

Le résultat est :

Groupe	IdModule	Moyenne
A	1	11.3333
A	2	10.6667
A	3	12.0
A	4	10.1667

III. Restrictions d'accès aux données

1- Définir les règles d'accès aux données

```
2- CREATE USER '12214687' WITH PASSWORD '65981';
3- CREATE USER '12214685' WITH PASSWORD '43290';
4- CREATE USER CHEF WITH PASSWORD '1234568';
5-
6- CREATE GROUP professeur WITH USER '12214698';
7- CREATE GROUP etude WITH USER '12214687', '12214685';
8-
9- GRANT SELECT ON Enseignant TO professeur;
10-GRANT SELECT ON Etudiant TO professeur;
11-GRANT SELECT, UPDATE, INSERT ON Module TO professeur;
12-GRANT SELECT, UPDATE, INSERT ON Evaluation TO professeur;
13-GRANT SELECT, UPDATE, INSERT ON Resultat TO professeur;
```

```
14-
15-GRANT SELECT ON Module TO etude;
16-GRANT SELECT ON Evaluation TO etude;
17-GRANT SELECT ON Resultat TO etude;
18-
19-GRANT ALL PRIVILEGES ON Enseignant TO CHEF;
20-GRANT ALL PRIVILEGES ON Etudiant TO CHEF;
21-GRANT ALL PRIVILEGES ON Module TO CHEF;
22-GRANT ALL PRIVILEGES ON Evaluation TO CHEF;
23-GRANT ALL PRIVILEGES ON Resultat TO CHEF;
```

Ce script nous permet de créer les différents utilisateurs et de les ajouter dans deux groupes différents, le premier est professeur celui-ci permet d'avoir différents accès aux tables notamment comme ceci :

● Les enseignants peuvent sélectionner toutes les tables (SELECT) et ils ont quelques droits en plus sur les tables Module, Evaluation et Resultat (UPDATE et INSERT).

Le second rôle est etude ceci donne certaines restrictions aux étudiants sur les autres tables comme ceci :

● Les étudiants peuvent sélectionner seulement les tables evaluation, module et resultat (SELECT) afin de voir leur propre note, ils n'ont aucun accès sur les tables enseignant et etudiant.

Enfin nous disposons d'un super utilisateur CHEF qui a accès à toutes les tables de la base de données, il peut modifier, supprimer ou créer des tables.

2- Changer la note :

```
CREATE OR REPLACE FUNCTION Changer_Note(
    in_Note FLOAT,
    in_idEt INT,
    in_IdMat VARCHAR,
    in_NomControle VARCHAR
) RETURNS FLOAT AS
$$
DECLARE
    updated_note FLOAT;
BEGIN
    UPDATE controle
    SET Note = in_Note
    WHERE in_idEt = controle.IdEtudiant
        AND controle.IdProfesseur::VARCHAR(15) = session_user
        AND controle.IdControle = in_IdMat
        AND controle.Nom = in_NomControle
```

```
    RETURNING Note INTO updated_note;

    RETURN updated_note;
END;
$$
LANGUAGE plpgsql
SECURITY DEFINER;
```

Cette fonction retournera la nouvelle note de l'étudiant qui est prise en paramètres.

Par exemple si on fait

```
SELECT Changer_Note(18, 1, '1', 'controle');
```

Il affichera 18.

3-Procédure pour voir le résultat d'un étudiant

```
CREATE OR REPLACE PROCEDURE Voir_Notes_Etudiant(in_IdEtudiant INT)
AS $$
BEGIN
    SELECT e.Nom_étudiant, e.Prénom_étudiant, ev.Nom_évaluation, r.Note
    FROM Etudiant e
    JOIN Résultat r ON e.Id_étudiant = r.Id_etudiant
    JOIN Evaluation ev ON r.Id_Evaluation = ev.Id_Evaluation
    WHERE e.Id_étudiant = in_IdEtudiant;
END;
$$
LANGUAGE plpgsql;
```

Si on essaie cet exemple :

```
SELECT Voir_Notes_Etudiant(1);
```

Il affiche ce résultat :

```
 Nom_étudiant | Prénom_étudiant | Nom_évaluation | Note
--------------+-----------------+----------------+------
 Oumsid       | Hiba            | contrôle       | 15
 Oumsid       | Hiba            | Examen         | 13
 Oumsid       | Hiba            | contrôle       | 13
 Oumsid       | Hiba            | contrôle       | 11
```