**SAE 204 : Exploitation d’une base de données**

HAMSEK | Fayçal | Zeus | BUT Info

Table des matières

[I- Mise en place de la base de données 2](#_Toc135585706)

[a. Cahier des charges 2](#_Toc135585707)

[b. Schéma Relationnel 3](#_Toc135585708)

[c. Script de création des tables 4](#_Toc135585709)

[d. Procédures stockées 6](#_Toc135585710)

[II- Visualisation des données 7](#_Toc135585711)

[a. Ensembles de données dérivées à visualiser 7](#_Toc135585712)

[b. Script des vues pour les données à visualiser 7](#_Toc135585713)

[III- Restriction d’accès aux données 8](#_Toc135585714)

[a. Définition des règles d’accès aux données : 8](#_Toc135585715)

[b. Script des règles d’accès aux données : 8](#_Toc135585716)

[IV- Conclusion 9](#_Toc135585717)

# Mise en place de la base de données

Cette première partie a pour but de mettre en place notre base de données pour mettre à bien notre projet. Pour cela nous allons faire un cahier des charges à respecter mais aussi un schéma relationnel duquel un script en découlera.

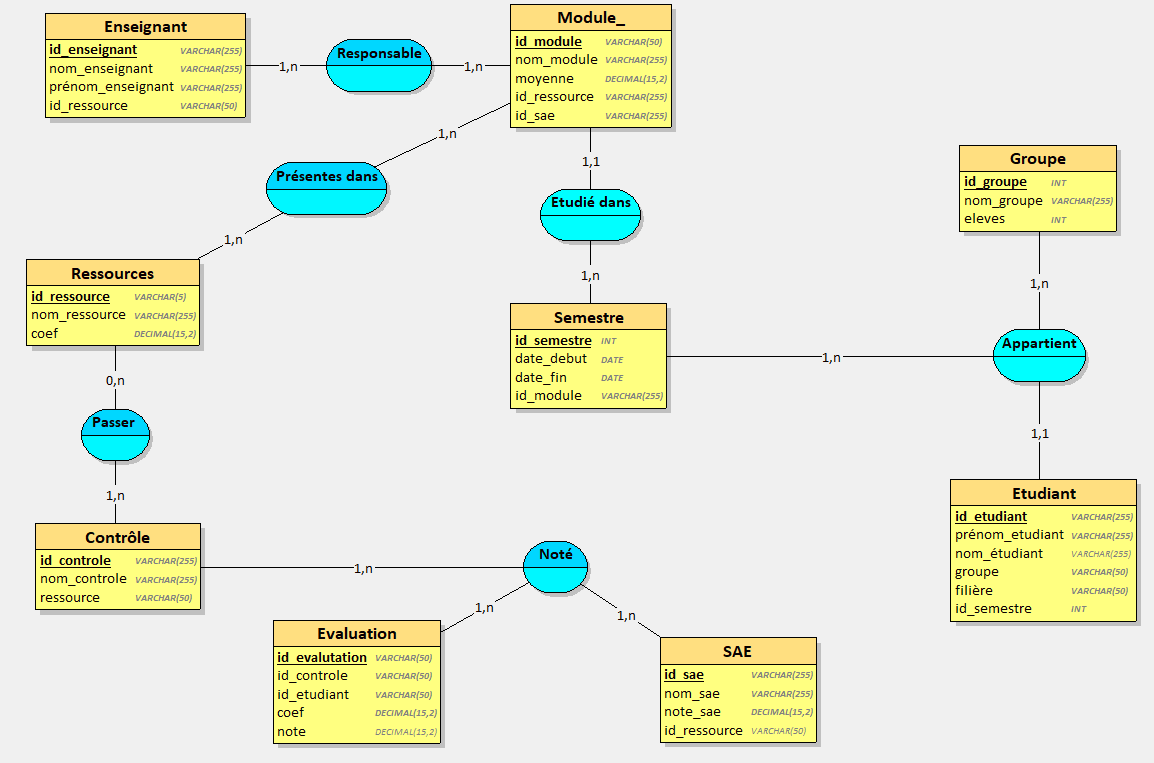
## Cahier des charges

La base de données doit contenir les tables suivantes :

* Enseignant : elle doit comporter les colonnes suivantes : id (clé primaire), nom, prénom, id\_ressource, semestre. La colonne id\_ressource est une clé étrangère qui référence la table Ressource.
* Etudiant : elle doit comporter les colonnes suivantes : id (clé primaire), nom, prénom, groupe, filière, id\_semestre. La colonne id\_semestre est une clé étrangère qui référence la table Semestre.
* Module : elle doit comporter les colonnes suivantes : id\_module (clé primaire), nom\_module, moyenne, id\_ressource, id\_sae. Les colonnes id\_ressource et id\_sae sont des clés étrangères qui référencent les tables Ressource et SAE respectivement.
* Evaluation : elle doit comporter les colonnes suivantes : id\_evaluation (clé primaire), id\_controle, id\_etudiant, coef, note.
* SAE : elle doit comporter les colonnes suivantes : id\_sae (clé primaire), nom\_SAE, id\_ressource, note. La colonne id\_ressource est une clé étrangère qui référence la table Module.
* Semestre : elle doit comporter les colonnes suivantes : id\_semestre (clé primaire), date\_debut, date\_fin, id\_module. La colonne id\_module est une clé étrangère qui référence la table Ressource.
* Groupe : elle doit comporter les colonnes suivantes : id\_groupe (clé primaire), nom\_groupe, eleves.
* Ressource : elle doit comporter les colonnes suivantes : id\_ressource (clé primaire), nom\_ressource, Coef.

Les clés étrangères doivent être correctement définies pour assurer l'intégrité de la base de données.

## Schéma Relationnel



## Script de création des tables

-- Création de la table Ressource

CREATE TABLE Ressource (

    id\_ressource VARCHAR(200) PRIMARY KEY,

    nom\_ressource VARCHAR(200),

    Coef DECIMAL

);

-- Création de la table SAE

CREATE TABLE SAE (

    id\_sae VARCHAR(200) PRIMARY KEY,

    nom\_SAE VARCHAR(200),

    id\_ressource VARCHAR(200),

    note DOUBLE PRECISION,

    FOREIGN KEY (id\_ressource) REFERENCES Ressource(id\_ressource)

);

-- Création de la table Module

CREATE TABLE Module1 (

    id\_module VARCHAR(200) PRIMARY KEY,

    nom\_module VARCHAR(200),

    moyenne DECIMAL,

    id\_ressource VARCHAR(200),

    id\_sae VARCHAR(200),

    FOREIGN KEY (id\_ressource) REFERENCES Ressource(id\_ressource),

    FOREIGN KEY (id\_sae) REFERENCES SAE(id\_sae)

);

-- Création de la table Semestre

CREATE TABLE Semestre (

    id\_semestre INT PRIMARY KEY,

    date\_debut DATE,

    date\_fin DATE,

    id\_module VARCHAR(200),

    FOREIGN KEY (id\_module) REFERENCES Module1(id\_module)

);

-- Création de la table Groupe

CREATE TABLE Groupe (

    id\_groupe INT PRIMARY KEY,

    nom\_groupe VARCHAR(200),

    eleves INT

);

-- Création de la table Enseignant

CREATE TABLE Enseignant (

    id\_enseignant SERIAL PRIMARY KEY,

    nom\_enseignant VARCHAR(200),

    prénom\_enseignant VARCHAR(200),

    id\_ressource VARCHAR(200),

    semestre INT,

    FOREIGN KEY (id\_ressource) REFERENCES Ressource(id\_ressource)

);

-- Création de la table Etudiant

CREATE TABLE Etudiant (

    id\_etudiant SERIAL PRIMARY KEY,

    nom\_etudiant VARCHAR(200),

    prénom\_etudiant VARCHAR(200),

    groupe VARCHAR(200),

    filière VARCHAR(200),

    id\_semestre INT,

    FOREIGN KEY (id\_semestre) REFERENCES Semestre(id\_semestre)

);

-- Création de la table Controle

CREATE TABLE Controle (

    id\_controle VARCHAR(200) PRIMARY KEY,

    id\_module VARCHAR(200),

    date\_controle DATE,

    FOREIGN KEY (id\_module) REFERENCES Module1(id\_module)

);

-- Création de la table Evaluation

CREATE TABLE Evaluation (

    id\_evaluation VARCHAR(200) PRIMARY KEY,

    id\_controle VARCHAR(200),

    id\_etudiant VARCHAR(200),

    coef DECIMAL,

    note DECIMAL

);

## Procédures stockées

-- retirer une note

CREATE OR REPLACE PROCEDURE retirer\_note(

    p\_id\_evaluation VARCHAR

)

AS $$

BEGIN

    DELETE FROM Evaluation

    WHERE id\_evaluation = p\_id\_evaluation;

END;

$$ LANGUAGE plpgsql;

--ajouter une SAE

CREATE OR REPLACE PROCEDURE ajouter\_sae(

    p\_id\_sae VARCHAR,

    p\_nom\_sae VARCHAR,

    p\_id\_ressource VARCHAR,

    p\_note DOUBLE PRECISION

)

AS $$

BEGIN

    INSERT INTO SAE (id\_sae, nom\_sae, id\_ressource, note)

    VALUES (p\_id\_sae, p\_nom\_sae, p\_id\_ressource, p\_note);

END;

$$ LANGUAGE plpgsql;

-- ajouter une note a un groupe

CREATE OR REPLACE FUNCTION ajouter\_note\_groupe(

    id\_groupe INT,

    id\_evaluation VARCHAR(200),

    id\_controle VARCHAR(200),

    id\_etudiant VARCHAR(200),

    coef DECIMAL,

    note DECIMAL

)

RETURNS VOID AS $$

BEGIN

    INSERT INTO Evaluation(id\_evaluation, id\_controle, id\_etudiant, coef, note)

    VALUES(id\_evaluation, id\_controle, id\_etudiant, coef, note);

    UPDATE Etudiant

    SET moyenne = (

        SELECT AVG(note \* coef)

        FROM Evaluation

        WHERE Evaluation.id\_etudiant = Etudiant.id\_etudiant

    )

    WHERE Etudiant.id\_groupe = id\_groupe;

END;

$$ LANGUAGE plpgsql;

# Visualisation des données

Dans cette deuxième partie nous allons faire des fonctions pour voir certaines données de notre base de données et nous allons donc utiliser VIEW et des procédures stockées.

## Ensembles de données dérivées à visualiser

* L’ensemble des professeurs
* Le relevé de note du groupe G1
* L’ensemble des élèves d’un groupe

## Script des vues pour les données à visualiser

--voir la liste des enseignants

CREATE VIEW Enseignants\_View AS

SELECT id\_enseignant, nom\_enseignant, prénom\_enseignant

FROM Enseignant;

 -- voir les notes du groupe G1

CREATE VIEW Notes\_Groupe\_View AS

SELECT Etudiant.nom\_etudiant, Etudiant.prénom\_etudiant, Evaluation.note

FROM Etudiant

JOIN Evaluation ON CAST(Etudiant.id\_etudiant AS VARCHAR) = Evaluation.id\_etudiant

WHERE Etudiant.groupe = 'G1';

-- voir les etudiants des groupes

CREATE VIEW Etudiant\_Groupe\_View AS

SELECT Etudiant.id\_etudiant, Etudiant.nom\_etudiant, Etudiant.prénom\_etudiant, Groupe.nom\_groupe, Groupe.filière

FROM Etudiant

INNER JOIN Groupe ON Etudiant.groupe = Groupe.nom\_groupe;

# Restriction d’accès aux données

Dans cette troisième et dernière partie nous allons fixer des règles à notre base de données car tout le monde ne peut pas la modifier et donc éviter des soucis de gestions. Nous allons faire différentes fonctions pour cela.

## Définition des règles d’accès aux données :

* Un enseignant ne peut consulter que les notes du module dont il est responsable
* Un enseignant peut supprimer une note d'un module dont il est responsable
* Un étudiant ne peut pas modifier les notes

## Script des règles d’accès aux données :

--un enseignant peut supprimer une note d'un module dont il est responsable

CREATE OR REPLACE FUNCTION supprimer\_note\_responsable(p\_enseignant\_id INTEGER, p\_module\_id VARCHAR(200), p\_etudiant\_id VARCHAR(200), p\_evaluation\_id VARCHAR(200))

RETURNS VOID

AS $$

BEGIN

    DELETE FROM Evaluation e

    USING Controle c

    INNER JOIN Module m ON c.id\_module = m.id\_module

    INNER JOIN Ressource r ON m.id\_ressource = r.id\_ressource

    WHERE e.id\_evaluation = p\_evaluation\_id

        AND e.id\_etudiant = p\_etudiant\_id

        AND c.id\_module = p\_module\_id

        AND r.id\_ressource = ANY (

            SELECT id\_ressource

            FROM Enseignant

            WHERE id\_enseignant = p\_enseignant\_id

        );

END;

$$ LANGUAGE plpgsql;

-- une étudiant ne peut pas modifier les notes

CREATE FUNCTION update\_evaluation\_forbidden() RETURNS trigger AS $$

BEGIN

    IF TG\_OP = 'UPDATE' THEN

        RAISE EXCEPTION 'Vous n''êtes pas autorisé à modifier les notes';

    END IF;

    RETURN NEW;

END;

$$ LANGUAGE plpgsql;

 -- il lui faut son trigger

 CREATE TRIGGER evaluation\_update\_forbidden

BEFORE UPDATE ON Evaluation

FOR EACH ROW

EXECUTE FUNCTION update\_evaluation\_forbidden();

-- un enseignent ne peut consulter que les notes de la ressource dont il est responsable

CREATE FUNCTION get\_notes\_responsable(p\_enseignant\_id INTEGER)

RETURNS TABLE (id\_evaluation VARCHAR(200), id\_controle VARCHAR(200), id\_etudiant VARCHAR(200), coef DECIMAL, note DECIMAL)

AS $$

BEGIN

    RETURN QUERY

    SELECT e.id\_evaluation, e.id\_controle, e.id\_etudiant, e.coef, e.note

    FROM Evaluation e

    INNER JOIN Controle c ON e.id\_controle = c.id\_controle

    INNER JOIN Module m ON c.id\_module = m.id\_module

    INNER JOIN Ressource r ON m.id\_ressource = r.id\_ressource

    WHERE r.id\_ressource = ANY (

        SELECT id\_ressource

        FROM Enseignant

        WHERE id = p\_enseignant\_id

    );

END;

$$ LANGUAGE plpgsql;

# Conclusion

Durant cette SAE j’ai vu de nombreux éléments que j’avais déjà utilisé sur la création et l’exploitation de base de données, les schémas relationnels, la réalisation de script. Mais j’ai pu m’initier à de nouveaux éléments et les mettre en place comme les procédures stockées, les vues et les règles de gestions. Ceci m’a permis d’acquérir des compétences mais d’en renforcer d’autres.