Berdah

Clément

Zeus

SAÉ 2.04. Exploitation d’une base de données.

1. **Introduction.**
2. **Modélisation de Données.**
3. **Visualisation de Données.**
4. **Restriction d’accès aux Données.**

### Introduction.

Ce projet 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, etc.
3. L’étude et la mise 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.

### **II. Modélisation de Données**

**Cahier des charges :**

Dans cette base de données représentant le système de gestion des notes des étudiants du BUT, on y trouve plusieurs tables.

Il faut donc tous les éléments qui correspondent à ce système. On a donc cette liste de tables :

* Groupes

Cette table contient l’id du groupe, son nom, ainsi que le nombre d’élèves qui en font partie.

* Étudiant

Pour celle-ci, on a l’id, nom et prénom de l’étudiant ainsi que l’id du groupe auquel il appartient.

* Semestre

Celle-ci contient l’id du semestre, sa date de début et fin.

* Ressources

Dans cette table on y écrit l’id et le nom de la ressource ainsi que le coefficient de la ressource et l’id du semestre auquel il appartient.

* Contrôle

Pour celui-ci, on a l’id, le nom du contrôle, mais aussi la ressource concernée.

* Enseignant

Ici, on a l’id, nom et prénom de l’enseignant ainsi que la matière qu’il enseigne.

* Notes

Ici, on a l’id de la note, l’étudiant et le contrôle concerné ainsi que le coefficient et la note.

* SAÉ

Pour les SAÉ, on a l’id, son code, son nom ainsi que l’id du contrôle référent et la note.

* Module

Celle-ci va comporter l’id et le nom du module, la moyenne des notes du module, la SAÉ du module ainsi que son code.

Voici un schéma relationnel illustrant ceci :

Groupes(id\_groupe INT PRIMARY KEY, nom\_groupe VARCHAR(255), nb\_eleves INT);

Etudiant(id\_etudiant INT PRIMARY KEY, nom\_etudiant VARCHAR(255), prenom\_etudiant (VARCHAR 255), groupe FOREIGN KEY (groupe) REFERENCES Groupes(id\_groupe));

Semestre(id\_semestre INT PRIMARY KEY, date\_debut DATE, date\_fin DATE);

Ressources(id\_ressource VARCHAR(5) PRIMARY KEY, nom\_ressource VARCHAR(255), coef FLOAT, semestre FOREIGN KEY (semestre) REFERENCES Semestre(id\_semestre));

Controle (id\_controle INT PRIMARY KEY, nom\_controle VARCHAR(255), ressource\_concernee FOREIGN KEY references Ressource(id\_ressource));

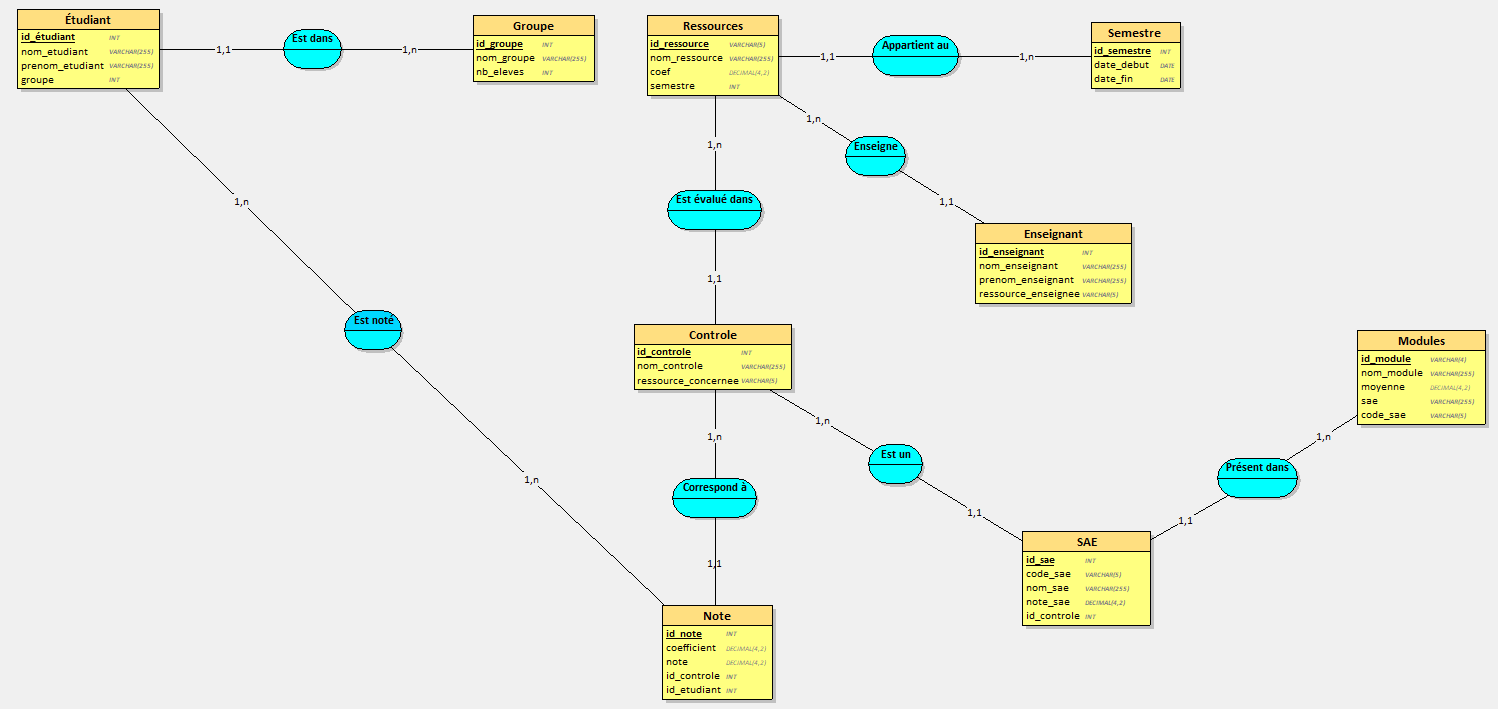
Enseignant(id\_enseignant INT PRIMARY KEY, nom\_enseignant VARCHAR(255), prenom\_enseignant VARCHAR(255), ressource\_enseignee FOREIGN KEY REFERENCES Ressources(id\_ressource));

Note(id\_note INT PRIMARY KEY, id\_etudiant FOREIGN KEY references etudiant(id\_etudiant), id\_controle FOREIGN KEY REFERENCES controle(id\_controle), coefficient FLOAT, note FLOAT);

SAE(id\_sae INT PRIMARY KEY, code\_sae VARCHAR(5) UNIQUE , nom\_sae VARCHAR(255), note\_sae FLOAT, id\_controle FOREIGN KEY REFERENCES controle(id\_controle));

Modules (id\_module VARCHAR(4) PRIMARY KEY, nom\_module VARCHAR(255), moyenne FLOAT,sae VARCHAR(255), code\_sae FOREIGN KEY REFERENCES sae(code\_sae));

**Réalisation du modèle de la Base de Données :**

****

**Règles de gestion des données et mises en oeuvres par des procédures stockées :**

* **Ajouter un enseignant.**

**→ Procédure :**

CREATE FUNCTION ajouter\_enseignant(id\_enseignant INT, nom\_enseignant VARCHAR(255), prenom\_enseignant VARCHAR(255), ressource\_enseignee VARCHAR(5))

RETURNS void AS $$

BEGIN

INSERT INTO Enseignant (id\_enseignant, nom\_enseignant, prenom\_enseignant, ressource\_enseignee) VALUES (id\_enseignant, nom\_enseignant, prenom\_enseignant, ressource\_enseignee);

END;

$$ LANGUAGE plpgsql;

SELECT ajouter\_enseignant(3, Buscaldi, 'Davide', 'R.206');

* **Chaque ID d’enseignant est unique**.

CREATE OR REPLACE FUNCTION verif\_id\_enseignant() RETURNS TRIGGER AS $$

BEGIN

IF EXISTS(SELECT 1 FROM enseignant WHERE id\_enseignant = NEW.id\_enseignant) THEN

RAISE EXCEPTION 'Cet ID enseignant existe déjà.';

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER tr\_verif\_id\_enseignant

BEFORE INSERT ON enseignant

FOR EACH ROW

EXECUTE FUNCTION verif\_id\_enseignant();

SELECT ajouter\_enseignant(3, 'Ellouze', 'Slim', 'R2.01');

error : “Cet ID enseignant existe déjà.”

**Réalisation du script SQL :**

CREATE TABLE Groupes (

id\_groupe INT PRIMARY KEY,

nom\_groupe VARCHAR(255),

nb\_eleves INT

);

CREATE TABLE etudiant (

id\_etudiant INT PRIMARY KEY,

nom\_etudiant VARCHAR(255) NOT NULL,

prenom\_etudiant VARCHAR(255) NOT NULL,

groupe INTEGER NOT NULL,

FOREIGN KEY (groupe) REFERENCES groupes(id\_groupe)

);

CREATE TABLE Semestre (

id\_semestre INT PRIMARY KEY,

date\_debut DATE,

date\_fin DATE

);

CREATE TABLE ressource (

id\_ressource VARCHAR(5) PRIMARY KEY,

nom\_ressource VARCHAR(255),

coef INT,

semestre INT,

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

);

CREATE TABLE Controle (

id\_controle INT PRIMARY KEY,

nom\_controle VARCHAR(255),

ressource\_concernee VARCHAR(5),

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

);

CREATE TABLE enseignant (

id\_enseignant INT PRIMARY KEY,

nom\_enseignant VARCHAR(255),

prenom\_enseignant VARCHAR(255),

ressource\_enseignee VARCHAR(5)

);

CREATE TABLE note (

id\_note INT PRIMARY KEY,

id\_etudiant INT REFERENCES etudiant(id\_etudiant),

id\_controle INT REFERENCES controle(id\_controle),

coefficient INT,

note FLOAT

);

CREATE TABLE SAE (

id\_sae INT PRIMARY KEY,

code\_sae VARCHAR(5) UNIQUE,

nom\_sae VARCHAR(255),

id\_controle INT,

note\_sae FLOAT,

FOREIGN KEY (id\_controle) REFERENCES Controle(id\_controle)

);

CREATE TABLE module (

id\_module VARCHAR(4) PRIMARY KEY,

nom\_module VARCHAR(255),

moyenne INT,

code\_sae VARCHAR(5),

sae VARCHAR(255),

FOREIGN KEY (code\_sae) REFERENCES sae(code\_sae)

);

### **III. Visualisation de données.**

* **Voici une procédure VUE permettant de voir les enseignants, ainsi que la ressource qu’ils enseignent.**

CREATE VIEW vue\_enseignants\_ressources AS

SELECT enseignant.nom\_enseignant, enseignant.prenom\_enseignant, ressource.nom\_ressource

FROM enseignant

INNER JOIN ressource ON enseignant.ressource\_enseignee = ressource.id\_ressource;

* **Voici une procédure VUE qui affiche chaque contrôle ainsi que la ressource auquel il appartient.**

CREATE VIEW vue\_controles\_ressources AS

SELECT Controle.nom\_controle, ressource.nom\_ressource

FROM Controle

INNER JOIN ressource ON Controle.ressource\_concernee = ressource.id\_ressource;

* **Voici une procédure VUE qui affiche le nom, prénom, groupe ainsi que la moyenne de tous les élèves ayant une moyenne supérieure à 10.**

CREATE OR REPLACE VIEW vue\_moyenne\_sup\_10 AS

SELECT e.nom\_etudiant, e.prenom\_etudiant, g.nom\_groupe, AVG(n.note) as moyenne

FROM etudiant e

JOIN note n ON e.id\_etudiant = n.id\_etudiant

JOIN groupe g ON e.groupe = g.id\_groupe

GROUP BY e.id\_etudiant, g.nom\_groupe

HAVING AVG(n.note) > 10;

* **Voici une procédure VUE qui affiche le nombre d’élèves d’un groupe.**

CREATE OR REPLACE VIEW vue\_nb\_etudiants\_par\_groupe\_semestre AS

SELECT g.nom\_groupe, COUNT(e.id\_etudiant) as nb\_etudiants

FROM etudiant e

JOIN groupe g ON e.groupe = g.id\_groupe

JOIN semestre s ON s.id\_semestre = <ID\_SEMESTRE>

WHERE EXISTS (

SELECT \*

FROM ressource r

JOIN controle c ON r.id\_ressource = c.ressource\_concernee

JOIN note n ON n.id\_controle = c.id\_controle

WHERE n.id\_etudiant = e.id\_etudiant

AND r.semestre = s.id\_semestre

)

GROUP BY g.nom\_groupe;

### **III. Restriction d’accès aux données.**

* **Procédure permettant de récupérer les notes d’un étudiant spécifique.**

CREATE PROCEDURE note\_etudiant

@etudiant INT

AS

BEGIN

IF EXISTS (

SELECT 1

FROM note

WHERE id\_etudiant = @etudiant

)

BEGIN

SELECT id\_note, id\_etudiant, id\_controle, coefficient, note

FROM note

WHERE id\_etudiant = @etudiant;

END

END;

**Commande pour éxecuter :**

EXEC note\_etudiant @etudiant = id\_etudiant ;

* **Procédure permettant à un enseignant de mettre une note à un élève sur un contrôle qui est présent dans la ressource qu’il enseigne.**

CREATE PROCEDURE mettre\_note

@enseignant INT,

@etudiant INT,

@controle INT,

@coef INT,

@note FLOAT

AS

BEGIN

IF EXISTS (

SELECT 1

FROM enseignant e

INNER JOIN controle c ON e.ressource\_enseignee = c.ressource\_concernee

WHERE e.id\_enseignant = @enseignant AND c.id\_controle = @controle

)

BEGIN

INSERT INTO note (id\_etudiant, id\_controle, coefficient, note)

VALUES (@etudiant, @controle, @coef, @note);

PRINT 'Note ajoutée';

END

END;

**Commande pour éxecuter :**

EXEC InsertTeacherNote @teacherId = <ID de l'enseignant>, @studentId = <ID de l'étudiant>,

@controlId = <ID du contrôle>, @coefficient = <coefficient>, @note = <note>;

* **Procédure qui restreint un élève à mettre des notes.**

CREATE PROCEDURE restreindre\_eleve

@etudiant INT

AS

BEGIN

IF EXISTS (

SELECT 1

FROM etudiant

WHERE id\_etudiant = @etudiant

)

BEGIN

PRINT 'Les élèves ne sont pas autorisés à mettre des notes.';

END

END;

**Commande pour éxectuer :**

EXEC RestrictStudentGrade @etudiant = id\_etudiant;