S2.04 Exploitation d'une Base de Données

THURAIRAJASINGAMM Kavusikan
RIZAOGLU Fulya
CAI Kemo
Groupe : Shango
Professeur chargé de TD : ABIR Hocine

IUT de Villetaneuse
Université Sorbonne Paris Nord

SOMMAIRE :

I : Introduction :

- l'étude d'un modèle de données pour mettre en place une base de données de gestion des notes des étudiants en BUT.
- L'étude et la mise en œuvre de la gestion des données dérivées : relevé de notes, bilans, etc.
- L'étude et la mise en œuvre des restrictions d'accès à ces données : étudiant, enseignant, responsable de matière, etc.

II : Modélisation de données :

- Établir un cahier des charges : vous pouvez consulter la notice "INFO Référentiel Informatique V15ACD.pdf" (joint à ce projet) pour plus d'informations.
- Étudier un modèle de données et réaliser une Base de Données à partir de ce modèle.
- Définir les règles de gestion de ces données et leurs mises en œuvre par des procédures stockées.
- Fournir un script de création de la base de données :
    - Modèle de données
    - Script du modèle de création de la base de données

III : Visualisation de Données :

- Définir un ensemble de données dérivées à visualiser.
- Décrire des procédures, vues ou vues matérialisées pour accéder à ces données.

IV : Restrictions d'accès aux données :

- Définir des règles d'accès aux données.
- Décrire des procédures ou vues pour mettre en œuvre ces règles.

PARTIE II : Cahier des charges :

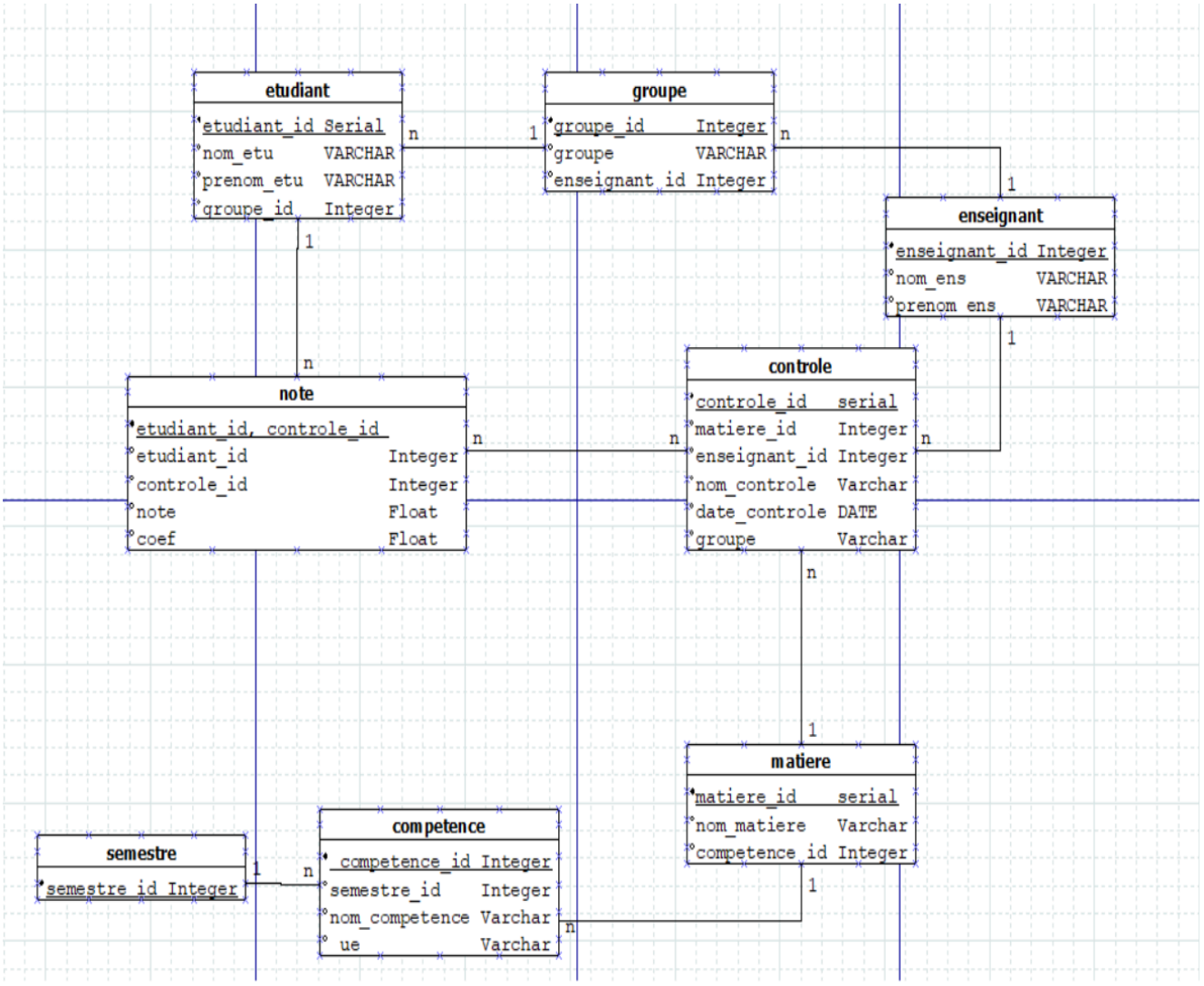
Données de l'énoncé : Mise en place d'une base de données de gestion des notes des étudiants en BUT.

Spécifications extrapolées de ces données :

- Le relevé de notes
- Des bilans sur les notes
- Gérer les restrictions d'accès

PARTIE II : Création de la base de données :

Modèle des données :



Script du modèle de création de la base de données :

```
CREATE DATABASE BUT ;
```

```
CREATE TABLE etudiant
        (
                etudiant_id serial
                        Primary Key,
                nom_etu VARCHAR(50),
                prenom_etu VARCHAR(50),
                groupe_id INTEGER
                        references groupe(groupe_id)
        );
```

```
CREATE TABLE enseignant
        (
                enseignant_id INTEGER
                        Primary Key,
                nom_ens VARCHAR(50),
                prenom_ens VARCHAR(50)
        );
```

```
CREATE TABLE semestre
        (
                semestre_id INTEGER
                        Primary Key
        );
```

```
CREATE TABLE competence
        (
                competence_id INTEGER
                        Primary Key,
                semestre_id INTEGER
                        references semestre(semestre_id),
                nom_competence VARCHAR(50),
                ue VARCHAR(50)
        );
```

```
CREATE TABLE matiere
        (
                matiere_id serial
                        Primary Key,
                nom_matiere VARCHAR(50),
                competence_id INTEGER
                        references competence(competence_id)
        );
```

```
CREATE TABLE controle
        (
                controle_id serial
                        Primary Key,
                matiere_id INTEGER
                        references matiere(matiere_id),
                enseignant_id INTEGER
                        references enseignant(enseignant_id),
                nom_controle VARCHAR(50),
                date_controle DATE,
                groupe VARCHAR(50)
        );
```

```
CREATE TABLE note
        (
                etudiant_id INTEGER
                        references etudiant(etudiant_id),
                controle_id INTEGER
                        references controle(controle_id),
                note FLOAT,
                coef FLOAT,
                Primary Key (etudiant_id,controle_id)
        );
```

```
CREATE TABLE groupe
        (
                groupe_id INTEGER
                        Primary Key,
                groupe VARCHAR(50),
                enseignant_id INTEGER
                        references enseignant(enseignant_id)
        );
```

III : Données dérivées à visualiser :

Exemple de relevé des notes de chaque étudiant :

exemple avec un étudiant dont sont identifiant étudiant est 1 dans la base de données du BUT :

```
SELECT * FROM note WHERE etudiant_id=1;

+------------+-------------+------+------+
| etudiant_id| controle_id | note | coef |
+------------+-------------+------+------+
|     1      |      1      |  14  |  0.5 |
|     1      |      2      |  11  |   2  |
|     1      |      3      |  08  |   1  |
|     1      |      4      |  16  |  0.5 |
|     1      |      5      |  20  |   1  |
|     1      |      6      |  12  |   4  |
+------------+-------------+------+------+
(6 rows)
```

III : Description des procédures :

Exemple de description des procédures par les vues :

```
CREATE VIEW vue_etudiant AS
SELECT semestre.semestre_id, etudiant.nom_etu, etudiant.prenom_etu,
competence.nom_competence, matiere.nom_matiere, controle.nom_controle,
controle.date_controle, note.note, note.coef
FROM note
JOIN etudiant ON note.etudiant_id = etudiant.etudiant_id
JOIN controle ON note.controle_id = controle.controle_id
JOIN matiere ON controle.matiere_id = matiere.matiere_id
JOIN competence ON matiere.competence_id = competence.competence_id
JOIN semestre ON competence.semestre_id = semestre.semestre_id
WHERE etudiant.prenom_etu::text = SESSION_USER;
```

exemple de description des procédures :

```
CREATE OR REPLACE FUNCTION moy_matiere(nom_matiere VARCHAR(50))
RETURNS TABLE (matiere VARCHAR(50), moy_matiere FLOAT) AS
$$
DECLARE
   cur CURSOR FOR
      SELECT m.nom_matiere, AVG(n.note * n.coef) AS moy_matiere
      FROM matiere m
      INNER JOIN controle co ON m.matiere_id = co.matiere_id
      INNER JOIN note n ON co.controle_id = n.controle_id
      WHERE m.nom_matiere = nom_matiere
      GROUP BY m.nom_matiere;
BEGIN
   OPEN cur;
   FETCH NEXT FROM cur INTO matiere, moy_matiere;
   WHILE FOUND LOOP
      RETURN NEXT;
      FETCH NEXT FROM cur INTO matiere, moy_matiere;
   END LOOP;
   CLOSE cur;
END;
$$
LANGUAGE plpgsql;


CREATE OR REPLACE FUNCTION moy_competence(nom_competence
VARCHAR(50))
RETURNS TABLE (semestre_id INTEGER, competence_id INTEGER,
moy_competence FLOAT) AS
$$
BEGIN
   RETURN QUERY
   SELECT c.semestre_id, c.competence_id, AVG(n.note * n.coef) AS
moy_competence
   FROM competence c
   INNER JOIN matiere m ON c.competence_id = m.competence_id
   INNER JOIN controle co ON m.matiere_id = co.matiere_id
   INNER JOIN note n ON co.controle_id = n.controle_id
   WHERE c.nom_competence = nom_competence
   GROUP BY c.semestre_id, c.competence_id;
END;
$$
LANGUAGE plpgsql;
```

```
CREATE OR REPLACE FUNCTION moy_semestre(semestre_id INTEGER)
RETURNS TABLE (id_semestre INTEGER, moy_semestre FLOAT) AS
$$
BEGIN
   RETURN QUERY
   SELECT c.semestre_id, AVG(n.note * n.coef) AS moy_semestre
   FROM controle c
   INNER JOIN note n ON c.controle_id = n.controle_id
   WHERE c.semestre_id = semestre_id
   GROUP BY c.semestre_id;
END;
$$
LANGUAGE plpgsql;
```

IV : Définition des règles d'accès aux données :

```
CREATE ROLE etudiant ;
```

```
GRANT SELECT ON vue_etudiant TO etudiant ;
```

```
ALTER USER kavusikan SET ROLE etudiant ;
```

Une fois qu'un utilisateur est sur sa session d'utilisateur, il ne pourra exécuter que les commandes qui lui seront autorisées.

exemple d'un étudiant sur sa session, où il a seulement accès à ses notes :

```
SELECT * FROM note ;

+------------+-------------+------+------+
| etudiant_id| controle_id | note | coef |
+------------+-------------+------+------+
|      5     |      1      |  19  |  0.5 |
|      5     |      2      |  10  |   2  |
|      5     |      3      |  17  |   1  |
|      5     |      4      |  20  |  0.5 |
|      5     |      5      |  15  |   1  |
|      5     |      6      |  14  |   4  |
+------------+-------------+------+------+
(6 rows)
```

IV : Descriptions des procédures :

```
CREATE OR REPLACE FUNCTION enseignant_res_trig()
RETURNS TRIGGER AS
$$
BEGIN
   IF EXISTS (
      SELECT 1 FROM enseignant WHERE nom_ens = CURRENT_USER
   ) AND TG_OP = 'INSERT' THEN
      INSERT INTO note (etudiant_id, controle_id, note, coef)
      VALUES (NEW.etudiant_id, NEW.controle_id, NEW.note, NEW.coef);
      RETURN NEW;
   END IF;
END;
$$
LANGUAGE plpgsql;

CREATE TRIGGER enseignant_res_trig
BEFORE INSERT ON note
FOR EACH ROW
EXECUTE FUNCTION enseignant_res_trig();
```