S2.04 Exploitation d’une Base de Données

Aymen Alloune

Patrick Chen

Maurice Prieto

<!> Pour information en plus de ce fichier il y aura un autre fichier contenant tout le code de la base de données. <!>

***Sommaire:***

**I. Modélisation de données**

***I.1 Diagramme UML | PAGE 2***

***I.2 Script de création | PAGE 3***

**II: Visualisation de données**

***II.1: Définition d’un ensemble de données dérivées | PAGE 5***

***II.2 : Description de procédures et vues | PAGE 7***

*II.2.1 : Les vues | PAGE 5*

*II.2.2 : Les procédures | PAGE 11*

**III : Description de procédures et vues**

***III.1 : Définition des règles d’accès | PAGE 19***

***III.2 : Définition des règles d’accès | PAGE 20***

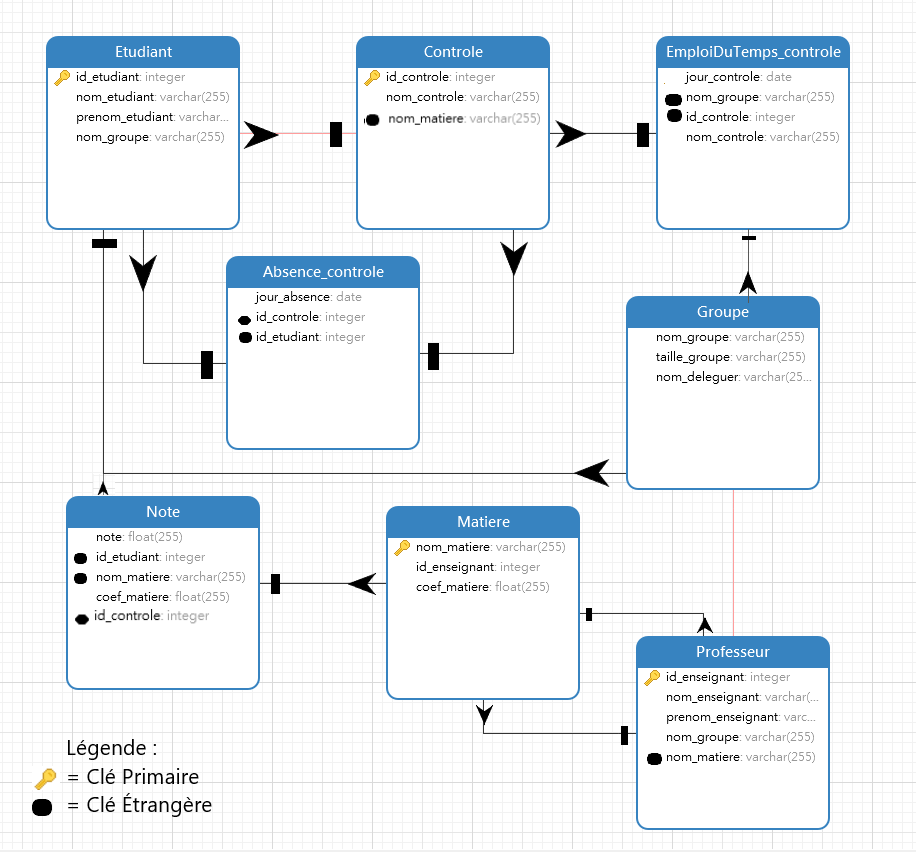
*III.2.1 : Procédures d’accès pour étudiant | PAGE 20*

*III.2.2 : Procédures d’accès pour professeur |PAGE23*

***Première Partie I : La modélisation de nos données***

***I.1 Diagramme UML :***

Juste en dessous on retrouve le diagramme UML de la base de données



***I.2 Script de création :***

Ensuite juste en dessous on retrouve toutes les tables de la base de données de gestion de note d’étudiant

CREATE TABLE Etudiant

(

id\_etudiant integer PRIMARY KEY,

nom\_etudiant varchar(10),

prenom\_etudiant varchar(10),

nom\_groupe varchar(10)

);

CREATE TABLE Matiere(

nom\_matiere varchar(10) PRIMARY KEY,

id\_enseignant integer,

coef\_matiere float

);

CREATE TABLE Professeur

(

id\_enseignant integer PRIMARY KEY,

nom\_enseignant varchar(10) ,

prenom\_enseignant varchar(10) ,

nom\_groupe varchar(10),

nom\_matiere varchar(10),

FOREIGN KEY (nom\_matiere) REFERENCES Matiere(nom\_matiere)

);

CREATE TABLE Controle (

id\_controle integer PRIMARY KEY,

nom\_controle varchar(20),

nom\_matiere varchar(20), /\*NEW\*/

FOREIGN KEY (nom\_matiere) REFERENCES Matiere(nom\_matiere)

);

CREATE TABLE Note

(

note float NOT NULL,

id\_etudiant integer,

nom\_matiere varchar(10) ,

coef\_matiere float,

id\_controle integer, /\*FOREIGN NEW\*/

FOREIGN KEY (id\_etudiant) REFERENCES Etudiant(id\_etudiant),

FOREIGN KEY (nom\_matiere) REFERENCES Matiere(nom\_matiere),

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

);

CREATE TABLE Groupe

(

nom\_groupe varchar(20) NOT NULL PRIMARY KEY ,

taille\_groupe integer,

nom\_deleguer varchar(20)

);

CREATE TABLE EDT\_controle

(

jour\_controle date NOT NULL,

id\_controle integer,

nom\_controle varchar(20),

nom\_groupe varchar(20),

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

FOREIGN KEY (nom\_groupe) REFERENCES Groupe(nom\_groupe)

);

CREATE TABLE Absence\_controle

(

jour\_absence date NOT NULL,

id\_controle integer,

id\_etudiant integer,

FOREIGN KEY (id\_etudiant) REFERENCES Etudiant(id\_etudiant),

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

);

***Deuxième partie II: Visualisation de données***

**II.1: Définition d’un ensemble de données dérivées**

Ici chaque donner a était créer dans un but précis pour qu’elle nous aide a essayer chaque vue et procédure qui arrivent par la suite, chaque table reçois des données dérivées.

INSERT INTO Etudiant values (5832,'Alloune','Aymen','Pegasus');

INSERT INTO Etudiant values (5733,'Chen','Patrick','Pegasus');

INSERT INTO Etudiant values (5266,'Prieto','Mauricio','Pegasus');

INSERT INTO Etudiant values (5183,'Dujardin','Jean','Draco');

INSERT INTO Etudiant values (5983,'Dupot','Jeanne','Draco');

INSERT INTO Etudiant values (5003,'Pierrot','Payera','Andromeda');

INSERT INTO Matiere values ('BD',8533,2.5);

INSERT INTO Matiere values ('Java',8749,4);

INSERT INTO Matiere values ('HTML',8652,3);

INSERT INTO Professeur values (8533,'Abir','Ab.','Pegasus');

INSERT INTO Professeur values (8749,'Azzag','Az.','Andromeda');

INSERT INTO Professeur values (8652,'Jean','Lassale','Draco');

INSERT INTO Controle values (1,'Test\_BD','BD');

INSERT INTO Controle values (2,'Test\_Java','Java');

INSERT INTO Controle values (3,'Test\_HTML','HTML');

INSERT INTO Note values (16.5,5832,'BD',2.5,1);

INSERT INTO Note values (17,5832,'Java',4,2);

INSERT INTO Note values (15,5832,'HTML',3,3);

INSERT INTO Note values (19.99,5733,'Java',4,2); /\* GROUPE PEGASUS \*/

INSERT INTO Note values (20,5266,'HTML',3,3);

INSERT INTO Note values (17,5183,'BD',2.5,1);

INSERT INTO Note values (17.75,5983,'BD',2.5,1); /\* GROUPE DRACO \*/

INSERT INTO Note values (15.5,5003,'HTML',3,3); /\* GROUPE ANDROMEDA \*/

INSERT INTO Note values (16.5,5003,'BD',2.5,1);

INSERT INTO Groupe values ('Pegasus',25,'Ony');

INSERT INTO Groupe values ('Draco',25,'Dujardin');

INSERT INTO Groupe values ('Andromeda',26,'Pierrot');

INSERT INTO EDT\_controle values ('2022-04-19',1,'Test\_BD', 'Pegasus');

INSERT INTO EDT\_controle values ('2022-05-10',2,'Test\_Java', 'Andromeda');

INSERT INTO EDT\_controle values ('2022-05-26',3,'Test\_HTML', 'Draco');

INSERT INTO Absence\_controle values ('2022-04-19',1,5832);

INSERT INTO Absence\_controle values ('2022-05-10',2,5832);

INSERT INTO Absence\_controle values ('2022-05-10',2,5733);

INSERT INTO Absence\_controle values ('2022-05-26',3,5266);

***II.2 : Description de procédures et vues***

En regroupant toutes les vues et procédures que nous allons présenter pour chaque table et données dérivées qu’il y aura, forcément on trouvera également un moyen d’accéder a ses données.

*II.2.1 : Les vues*

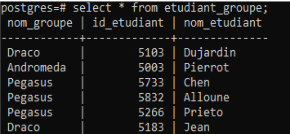
A/ Etudiant grouper par groupe

Cette vue nous donne accèes a chaque élèves ordonner par groupe

Create VIEW etudiant\_groupe as select distinct e.nom\_groupe,id\_etudiant,nom\_etudiant

from Etudiant e, Groupe g

where e.nom\_groupe=g.nom\_groupe;



B/ Moyenne des étudiants par matière

Une vue qui regroupe les étudiants par la moyenne qu’ils ont eu dans une matière.

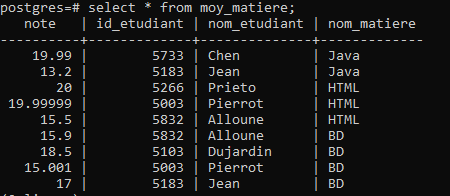
Create VIEW moy\_matiere as select \* from (

select avg(note)as Note,e.id\_etudiant,e.nom\_etudiant,nom\_matiere

from Note n , Etudiant e where n.id\_etudiant=e.id\_etudiant

group by nom\_matiere,e.id\_etudiant,e.nom\_etudiant order by Note DESC

) as s order by nom\_matiere DESC;



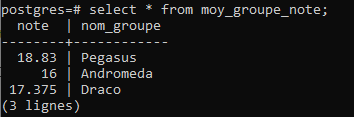
C/ Les moyennes des groupes

Ici on regroupe les moyenne par groupe

Create VIEW moy\_groupe\_note as select avg(note) as Note,nom\_groupe

from Note n, Etudiant e

where n.id\_etudiant=e.id\_etudiant group by nom\_groupe;



D/ Moyenne des délèguer

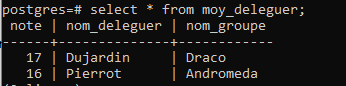
Cette vue nous montre les moyennes des délèguer

Create VIEW moy\_deleguer as select avg(note) as Note,nom\_deleguer,g.nom\_groupe

from Etudiant e , Note n , Groupe g

where g.nom\_deleguer=e.nom\_etudiant and e.id\_etudiant=n.id\_etudiant

group by nom\_deleguer,g.nom\_groupe order by Note DESC;



E/ Moyenne de groupe par controle

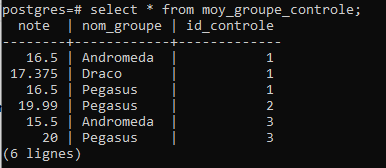
La prochaine vue regroupe les groupes ayant eu les meilleur notes , regrouper par contrôle fait.

Create view moy\_groupe\_controle as select avg(note) as Note,nom\_groupe,c.id\_controle

from Note n, Controle c,Etudiant e

where n.id\_controle=c.id\_controle and n.id\_etudiant=e.id\_etudiant

group by c.id\_controle,nom\_groupe order by c.id\_controle;



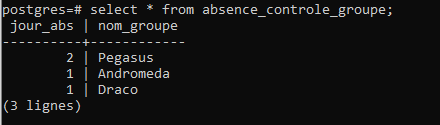
F/ Absence par contrôle des groupes

Pour celle-ci on va compter le nombre d’absent a chaque contrôle et les trier par groupe

CREATE VIEW absence\_controle\_groupe as SELECT count(jour\_absence) as Jour\_abs,nom\_groupe

from Absence\_controle a,Etudiant e

where a.id\_etudiant=e.id\_etudiant group by nom\_groupe order by Jour\_abs DESC;



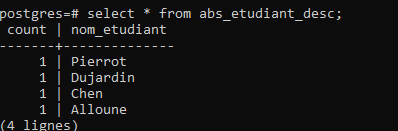
G/ Absence d’étudiant décroissant

Ici on va trier les absent a chaque contrôle et les trier par ordre décroissant et voir qui sont les élèves les plus absent.

CREATE VIEW abs\_etudiant\_desc as SELECT count(jour\_absence),nom\_etudiant

from Absence\_controle a , Etudiant e

where a.id\_etudiant=e.id\_etudiant group by nom\_etudiant order by nom\_etudiant DESC;



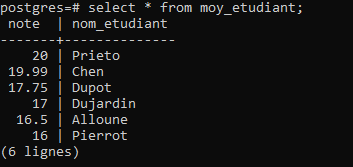
H/ Moyenne décroissante

Les moyennes de tous les étudiant trier dans l’ordre décroissant mettant en valeur les meilleur élèves.

CREATE VIEW moy\_etudiant as SELECT avg(note) as Note,nom\_etudiant

from Etudiant e , Note n

where e.id\_etudiant=n.id\_etudiant group by nom\_etudiant order by Note DESC;



I/ Les meilleurs professeurs

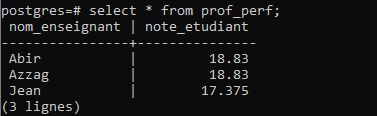
Enfin pour cette dernière vue on s’attaque à la table des professeurs, on va les trier selon les notes qu’on eut leurs élèves à chaque contrôle.

CREATE VIEW Prof\_Perf as SELECT nom\_enseignant,avg(note) as Note\_etudiant

from Etudiant e, Professeur p , Note n

where p.nom\_groupe=e.nom\_groupe and e.id\_etudiant=n.id\_etudiant

group by nom\_enseignant order by Note\_etudiant DESC;



***II.2.2 : Les procédures***

*A/ Première procédure,* cette procédure appeler Note\_CE qui signifie Note coefficients étudiants va nous permettre d’effectuer le calcul de coefficients avec d’une part les notes simples et de l’autre cotée les coefficients les deux donner en paramètre de la fonction, au final cela nous retournera la moyenne calculer grâce à la formule de calcul de coefficient.

CREATE or REPLACE function Note\_CE (in note float[],in coef float[])

RETURNS float

as

$$

DECLARE

note\_calculer float;

n int;

som\_notecoef float;

somme\_coef float;

BEGIN

som\_notecoef:=0;

somme\_coef:=0;

IF(array\_upper($1,1)!=array\_upper($2,1) ) THEN

note\_calculer:=0;

raise notice'<!> La taille des tableaux est pas pareil <!>';

return note\_calculer;

END IF;

/\* Remplissage de la variable som\_notecoef, en faisait la somme de chaque note multiplier par leur coefficients respectifs \*/

FOR n in 1 .. array\_upper($1,1) LOOP

som\_notecoef:=som\_notecoef+($1[n]\*$2[n]);

END LOOP;

/\* Remplissage de la variable somme\_coef , ici on fait juste la somme de tout les coefs \*/

FOR n in 1 .. array\_upper($2,1) LOOP

somme\_coef:=somme\_coef+coef[n];

END LOOP;

/\* enfin le calcul final on divise som\_notecoef par la somme des coefficients, somme\_coef\*/

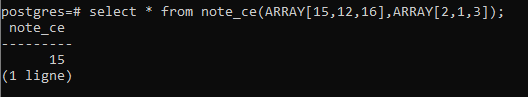
note\_calculer:=som\_notecoef/somme\_coef;

return note\_calculer;

END;

$$ language plpgsql;

Voici le test :



B/ *Seconde procédures ,* procédures appeler Moyenne calculer va faire appel a la procédure d’au dessus, elle va prendre en parametre simplement l’id d’un etudiant puis va créer deux tableaux de float, et va les remplir des notes de l’étudiant dans un premier tableaux puis des coefficients de ces même notes dans un second tableau, maintenant avec ces deux tableaux elle va appeler la fonction précèdente faire le calcul et retourner le résultat.

CREATE OR REPLACE function Moy\_calculer(in id int,out moyenne float)

returns float

as

$$

DECLARE

/\* Déclaration d'un curseur totalement lié pour une requête particulière \*/

curseur CURSOR FOR

select note,coef\_matiere from Note e, etudiant e2 where e.id\_etudiant=e2.id\_etudiant and e.id\_etudiant=$1 order by note DESC;

note\_array float[];

note\_coef float[];

tmp\_note float;

tmp\_coef float;

i int;

test\_etudiant int;

BEGIN

with test\_etudiant As(

select id\_etudiant from etudiant where id\_etudiant=$1

)

select count(\*) into test\_etudiant from test\_etudiant;

IF (test\_etudiant=0) THEN

raise notice '<!> Attention id\_etudiant ne correspond a personne <!>

<!> Une erreur se produit , modifier id\_etudiant <!>

';

END IF;

/\* initialisation du compteur i \*/

i:=1;

/\* On ouvre le curseur et on l'utilise pour remplir les tableaux note\_array(représente les note simple) et note\_coef \*/

OPEN curseur;

LOOP

FETCH curseur into tmp\_note,tmp\_coef;

EXIT WHEN NOT FOUND;

note\_array:=array\_append(note\_array,tmp\_note);

note\_coef:=array\_append(note\_coef,tmp\_coef);

i:=i+1;

END LOOP;

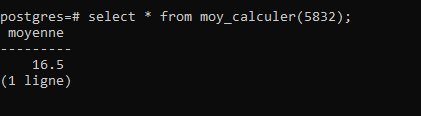
CLOSE curseur;

select \* into moyenne from Note\_CE(note\_array,note\_coef);

END;

$$ language plpgsql;

Voici le test:



C/ *troisième procédure,* Pour cette troisième procédure on va pouvoir avoir un œil sur les notes d’un étudiant avec un petit récapitulatif regroupant le minimum et maximum ainsi que le nombre de notes qu’il a et sa moyenne.

CREATE or REPLACE function apercu\_note

(int,out Nombre\_note float,out Note\_min float, out Moy\_note float, out Note\_max float)

returns setof record

as

$$

With note\_max As

(select note as max from note where id\_etudiant = $1 order by max DESC limit 1)

,note\_min as

(select note as min from note where id\_etudiant = $1 order by min ASC limit 1)

,note\_moy as

(select avg(note) as moy from note where id\_etudiant=$1)

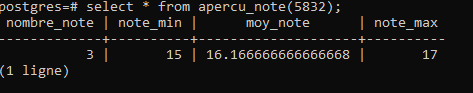
,nombre\_note as

(select count(note) as nbre from Note where id\_etudiant=$1)

SELECT nbre,min,moy,max from nombre\_note,note\_min,note\_moy,note\_max;

$$language sql;

Voici le test :



D/ *quatrième procédure,* de se coter-ci on va s’occuper de la tale groupe avec laquelle on va compare un nom de groupe donner en paramètre et retourner si ce groupe et le meilleur ou non ainsi que sa moyenne de groupe et la note du meilleur groupe.

Elle nous permet également d’avoir accès au note d’un étudiant.

CREATE OR REPLACE function meilleur\_groupe

(in varchar , out Resultat boolean,out Note\_Groupe float,out Note\_MeilleurGroupe float)

returns setof record

as

$$

DECLARE

classement varchar;

note\_m float;

BEGIN

/\* On utilise deux CTE pour les utiliser juste apres, une pour la variable de test , une pour la note\*/

WITH meilleur\_groupe as

(

select avg(note) as note,e.nom\_groupe

from Note n,Etudiant e,Groupe g

where n.id\_etudiant=e.id\_etudiant and e.nom\_groupe=g.nom\_groupe

group by e.nom\_groupe order by e.nom\_groupe DESC limit 1

)

,noteGroupe as (select avg(note) as note

from Note n,Etudiant e,Groupe g

where n.id\_etudiant=e.id\_etudiant and e.nom\_groupe=g.nom\_groupe and g.nom\_groupe=$1

order by note DESC limit 1)

SELECT nom\_groupe,m.note,n.note into classement,Note\_MeilleurGroupe,Note\_Groupe from meilleur\_groupe m ,noteGroupe n;

/\* Ici le test pour la variable OUT Resultat \*/

IF classement=$1 THEN Resultat:=true;

ELSE Resultat:=False;

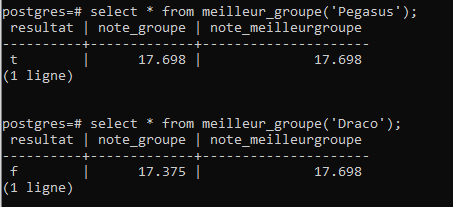
END IF;

return next;

END;

$$language plpgsql;

Voici le test :



E/ *cinquième procédure,* dans cette procédure on va pouvoir regrouper toutes les informations d’un contrôle précis donner grâce à l’id d’étudiant donner en paramètre.

CREATE or REPLACE function ControleInfo(in id int,out note float,out Nom\_Matiere varchar,out nom\_controle varchar,out Jour date,out nb\_Abscent int)

returns setof record

as

$$

select avg(note) as MoyenneNote,n.nom\_matiere as Nom\_Matiere,e.nom\_controle as NomControle,e.jour\_controle as Jour,count(a.id\_etudiant) as nb\_Abscent

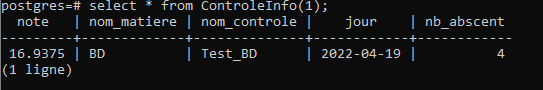
from controle c,Note n,EDT\_controle e,Absence\_controle a

where c.id\_controle=$1 and c.id\_controle=n.id\_controle and e.id\_controle=$1 and a.id\_controle=$1

group by n.Nom\_Matiere,NomControle,Jour order by MoyenneNote DESC;

$$ language sql;

Voici le test :



F/ *sixième procédure* , pour cette avant-dernière procédure on va se concentrer sur les professeur en donnant en paramètre le nom d’un matière on obtient le nom du professeur, le groupe que le professeur prend en charge ainsi que la moyenne de ce groupe dans sa matiere.

CREATE OR REPLACE function ProfesseurSearch(in NomMatiere varchar,out NomProf varchar,out MoyEleves float,out GroupeEleves varchar)

returns setof record

as

$$

DECLARE

test\_presence int;

BEGIN

/\* Ici test pour clarifier les erreurs humaines \*/

with ExisteOuPas as (

select count(nom\_matiere) from matiere where nom\_matiere=$1

)

select \* into test\_presence from ExisteOuPas;

IF (test\_presence=0) THEN

raise notice '<!> Le nom de la matiere existe pas <!>';

return;

END IF;

/\* On fait un select dans chaque variable OUT \*/

select nom\_enseignant,avg(note),p.nom\_groupe

into NomProf,MoyEleves,GroupeEleves

from Professeur p , Matiere m , Etudiant e, Note n

where m.nom\_matiere=p.nom\_matiere and m.nom\_matiere=$1 and e.id\_etudiant=n.id\_etudiant and p.nom\_groupe=e.nom\_groupe

group by nom\_enseignant,p.nom\_groupe ;

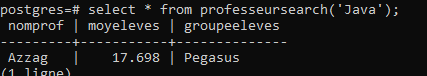
/\* Return next car c'est un setof record et qu'il faut retourner qu'une seule ligne on a donc pas besoin de boucle \*/

return next;

END;

$$language plpgsql;

Voici le test:



G/ *sixième procédure,* voici la dernière procédure , ici la spécificité est l’utilisation d’un curseur partiellement lié avec un argument celui donner en paramètre, on va avoir a la place d’un setof record , le nom de chaque étudiant renvoyer sous forme de notice et ne seulement retourner le nombre d’itération.

CREATE OR REPLACE function curseur\_controle(in nom\_controle varchar,out NombrePrésent int)

returns int

as

$$

DECLARE

/\* Utilisation d'un curseur partiellement lié \*/

curseur\_ctl CURSOR (nom\_ctl varchar) IS

select id\_etudiant,nom\_etudiant,e.nom\_groupe

from Etudiant e , Professeur p , Controle C , Matiere m

where e.nom\_groupe=p.nom\_groupe and m.id\_enseignant=p.id\_enseignant and m.nom\_matiere=c.nom\_matiere and c.nom\_controle=nom\_ctl

order by e.nom\_groupe;

IdEtudiant int;

NomEtudiant varchar;

i int;

BEGIN

i:=0;

OPEN curseur\_ctl($1);

/\* Première affichage pour rendre les notices compréhensible \*/

raise notice E'%\t% <-- Dans ce controle , voici la liste des eleves present : ',$1,' ' ;

LOOP

fetch curseur\_ctl into IdEtudiant,NomEtudiant;

EXIT WHEN NOT FOUND;

/\* Affichage de chaque élèves étant présent au controle donner en paramètre \*/

raise notice E’%\t%’,IdEtudiant,NomEtudiant;

i:=i+1;

END LOOP;

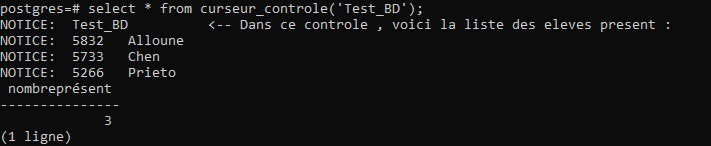
CLOSE curseur\_ctl;

NombrePrésent:=i;

END;

$$language plpgsql;

Voici le test :



***III : Restrictions d’accès aux données***

*III.1 : Définition des règles d’accèes*

En premier temps de cette troisième partie on va définir des règles d’accès :

Un étudiant ne peut accéder qu'a :

* Qu'a ses informations personnelles
* Qu'a ses absences,
* Qu'a son groupe,
* Qu'a ses notes,
* Qu'a ses matières,
* Qu'a ses noms d’enseignants et non pas à leur ID
* Qu'a ses contrôles qu'il a fait
* Qu'a son EDT ;

Un professeur ne peut accéder qu’à

* A accès a uniquement ses élèves
* A accès a uniquement sa matière
* A accès a uniquement ses informations de prof
* A accès à ses contrôles dans sa matière
* A accès uniquement au absence de ses contrôles
* A accès uniquement à son emploi du temps
* Au droit d'ajouter des notes

***III.2 : Procédures d’accès***

*III.2.1 : Procédures d’accès d’étudiant*

Voici ici les procédures permettant de créer ces règles

Pour les étudiants :

CREATE OR REPLACE FUNCTION MesInfos(out id\_etudiant int,out nom\_etudiant varchar,out prenom\_etudiant varchar,out nom\_groupe varchar)

returns setof record

as

$$

select id\_etudiant,nom\_etudiant,prenom\_etudiant,nom\_groupe

from Etudiant e

where e.nom\_etudiant=session\_user;

$$ language SQL

SECURITY DEFINER;

CREATE OR REPLACE FUNCTION MesAbsences(out jour\_abs date,out id\_controle int)

returns setof record

as

$$

select jour\_absence,id\_controle

from Absence\_controle a , Etudiant e

where a.id\_etudiant=e.id\_etudiant and e.nom\_etudiant=session\_user;

$$ language SQL

SECURITY DEFINER;

CREATE OR REPLACE FUNCTION MonGroupe(out nom\_groupe varchar,out taille\_groupe int,out nom\_deleguer varchar)

returns setof record

as

$$

Select g.nom\_groupe,taille\_groupe,nom\_deleguer

from Groupe g , etudiant e

where g.nom\_groupe=e.nom\_groupe and e.nom\_etudiant=session\_user;

$$ language sql

SECURITY DEFINER;

CREATE OR REPLACE FUNCTION MesNotes(out note float,out nom\_matiere varchar,out nom\_controle varchar,out coef\_matiere float,out id\_controle int)

returns setof record

as

$$

select n.note,n.nom\_matiere,c.nom\_controle,n.coef\_matiere,c.id\_controle

from Note n, Etudiant e, Controle c

where n.id\_etudiant=e.id\_etudiant and e.nom\_etudiant=session\_user;

$$ language sql

SECURITY DEFINER;

CREATE OR REPLACE FUNCTION MesMatieres(out nom\_matiere varchar,out nom\_enseignant varchar,out coef\_matiere float)

returns setof record

as

$$

select m.nom\_matiere,p.nom\_enseignant,m.coef\_matiere

from matiere m, Etudiant e,Professeur p

where m.id\_enseignant=p.id\_enseignant and p.nom\_groupe=e.nom\_groupe and e.nom\_etudiant=session\_user;

$$ language sql

SECURITY DEFINER;

CREATE OR REPLACE FUNCTION MonEDT(out JourControle date,out id\_controle int,out nom\_controle varchar,out jour\_absence date)

returns setof record

as

$$

SELECT jour\_controle,edt.id\_controle,nom\_controle,jour\_absence

from EDT\_controle edt ,Absence\_controle a, Etudiant e,Groupe g

where edt.nom\_groupe=g.nom\_groupe and g.nom\_groupe=e.nom\_groupe and e.nom\_etudiant=session\_user;

$$ language sql

SECURITY DEFINER;

*III.2.2 : Procédures d’accès pour professeur*

Et de ce côté les procédures pour professeur :

CREATE OR REPLACE FUNCTION MesInfos\_Prof(out id\_enseignant int,out nom\_enseignant varchar,out prenom\_enseignant varchar,out nom\_groupe varchar,out nom\_matiere varchar)

returns setof record

as

$$

select id\_enseignant,nom\_enseignant,prenom\_enseignant,nom\_groupe,nom\_matiere

from professeur p

where p.nom\_enseignant=session\_user;

$$ language SQL

SECURITY DEFINER;

CREATE OR REPLACE FUNCTION MesEleves(out id\_etudiant int,out nom\_etudiant varchar,out prenom\_etudiant varchar,out nom\_groupe varchar)

returns setof record

as

$$

select id\_etudiant,nom\_etudiant,prenom\_etudiant,e.nom\_groupe

from Etudiant e, Professeur p

where e.nom\_groupe=p.nom\_groupe and p.nom\_enseignant=session\_user;

$$language sql

SECURITY DEFINER;

CREATE OR REPLACE FUNCTION MesMatieres\_Prof(out MesMatieres varchar,out MesControles varchar,out coef\_matiere float)

returns setof record

as

$$

select m.nom\_matiere,c.nom\_controle,m.coef\_matiere

from Matiere m, Controle C, Professeur p

where m.id\_enseignant=p.id\_enseignant and p.nom\_enseignant=session\_user;

$$language sql

SECURITY DEFINER;

CREATE OR REPLACE FUNCTION MonEDT\_Prof(out JourControle date,out id\_controle int,out nom\_controle varchar,out NombreAbs int)

returns setof record

as

$$

select jour\_controle,edt.id\_controle,edt.nom\_controle,count(jour\_absence) as NombreAbs

from EDT\_controle edt,Absence\_controle a,Etudiant e ,Professeur p

where edt.nom\_groupe=p.nom\_groupe and a.id\_etudiant=e.id\_etudiant and e.nom\_groupe=p.nom\_groupe and p.nom\_enseignant=session\_user

group by jour\_controle,edt.id\_controle,edt.nom\_controle order by NombreAbs Desc;

$$language sql

SECURITY DEFINER;