SAE 2.04
Exploitation d’un base de donnée

Antoine David, Zhou William
Groupe : Tlaloc

I. Cahier des charges

Problème :
Une école veut mettre en place une base de données de gestion des notes des étudiants en BUT.

Contexte :
La gestion des données seront les relevés de notes, bilans,etc… Il faudra d'abord créer une base de données regroupant tous les étudiants, enseignants, module, matière, contrôle et notes. A partir de cela, on aura besoin d'une visualisation de données, c'est-à-dire de pouvoir voir le relevé des notes de chaque étudiant, de pouvoir voir le relevé de note d'un groupe, etc… Et enfin de restreindre l'accès aux notes c'est-à-dire que chaque étudiant a le droit de voir que leurs notes et les enseignants peuvent quant à eux modifier les notes que de leur module, etc…

Objectif :
Réaliser une base de données d'une école supérieure afin que les étudiants puissent consulter leur note.
Réaliser une base de données afin que les enseignants puissent modifier ou ajouter des notes de leur module.

Dates :
Début : 06/02/2023
Fin : 23/05/2023 à 12h00

Limites :
Ne pas dépasser la limite de la mémoire vive allouée.
Ne pas saturer la base de données.

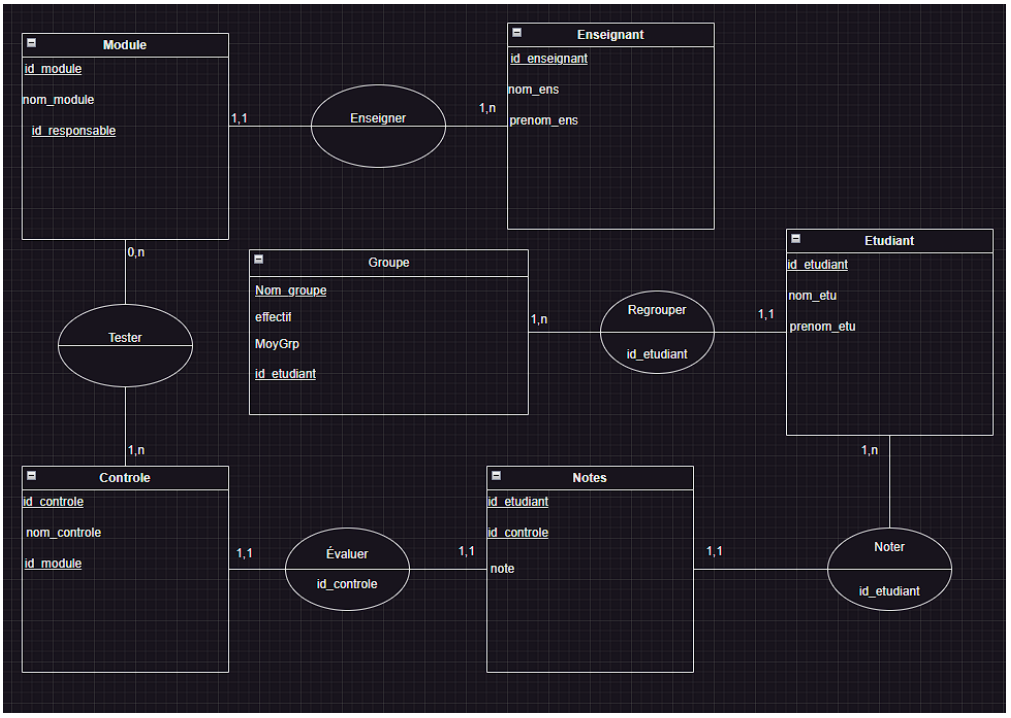
Exigences fonctionnelles :
- Accès à son compte étudiants et ses notes
- Accès à son compte enseignants et ses modules

Exigences non-fonctionnelles :

- Contraintes : Prix de la base de données, ressources humaines et matériels.
- Conformité : respect des normes réglementaires
- Maintenabilité : pouvoir faire des maintenances, régler les erreurs.
- Performance : 10000 utilisateurs possible dans la base de données.
- Portabilité : l'utilisateur peut se connecter sur toutes les plateformes et navigateurs.
- Fiabilité : La base de données est sécurisée, elle est rapidement remise en état de marche en cas de panne et est capable de résister aux attaques.
- Sécurité : protection de la vie privée de l'utilisateur.
- Utilisation : facile d'utilisation

II. Modélisation de Données

1. Base de données uml



2. Script de création de base de données

```
create table Etudiant ( id_etudiant serial primary key,nom_etu varchar(50), prenom_etu varchar(50));

create table Enseignant ( id_enseignant serial primary key, nom_ens varchar(50), prenom_ens varchar(50));

create table Module ( id_module int primary key, nom_module varchar(50), id_responsable int references Enseignant(id_enseignant));

create table Controle ( id_controle serial primary key, nom_controle varchar(50), id_module int references Module(id_module));

create table Notes ( id_etudiant int references Etudiant(id_etudiant), id_controle int references Controle(id_controle),
                    note decimal(4,2), primary key (id_etudiant, id_controle));

create table Groupe ( Nom_grp varchar(10) primary key, effectif int, MoyGrp int, id_etudiant int references Etudiant(id_etudiant));
```

Table Etudiant :

id_etudiant	nom_etu	prenom_etu
[PK] integer	character varying (50)	character varying (50)

Table Enseignant :

id_enseignant	nom_ens	prenom_ens
[PK] integer	character varying (50)	character varying (50)

Table Module :

id_module	nom_module	id_responsable
[PK] integer	character varying (50)	integer

Table Contrôle :

id_controle	nom_controle	id_module
[PK] integer	character varying (50)	integer

Table Notes :

id_etudiant	id_controle	note
[PK] integer	[PK] integer	numeric (4,2)

Table Groupe :

nom_grp	effectif	moygrp	id_etudiant
[PK] character varying (10)	integer	integer	integer

III. Visualisation de Données

1. Définir un ensemble de données dérivées à visualiser :

relevé de notes de chaque étudiants :

```
create function moy_etudiant(in id_etudiant int, out moy int)
    returns int
as
$$
declare
moy int;
begin
select avg(n.notes)
into moy
from note n
where n.id_etudiant = id_etudiant;

end;
$$ language plpgsql;
```

relevé de note d'un groupe :

```
create view moy_groupe as
    select e.id_etudiant, g.Nom_grp, avg(note) as moyenne
    from Etudiant e, Controle c, Notes n, Groupe g
        where c.id_controle = n.id_controle
        and n. id_etudiant = e. id_etudiant
        and e.id_etudiant = g.id_etudiant
    group by e. id_etudiant, Nom_Grp;
```

relevé de note d'un groupe d'une matière :

```
create view moyen_mat_group as
    select e.id_etudiant, nom_etu, prenom_etu, m.id_module, g.Nom_grp, avg(note) as moyenne
        from Etudiant e, Module m, Controle c, Notes n, Groupe g
            where m. id_module = c.id_module
                and c. id_controle = n. id_controle
                and n. id_etudiant = e. id_etudiant
                and e.id_etudiant = g.id_etudiant
        group by e.id_etudiant, nom_etu, prenom_etu, m. id_module, g. Nom_Grp;
```

IV. Restrictions d’accès aux Données

1. Définir des règles d’accès aux données.

L’enseignant ne peut modifier que les notes des modules dont il est responsable :

```
--un Enseignant ne peut consulter seulement les notes des modules dont il est responsable
CREATE or Replace function note_mod_by_ens(in id_enseignant int,in modules varchar) returns numeric

as



$$
    select note from Notes Natural join Controle Natural join module Natural join Enseignant
    where id_enseignant = $1 and nom_module = $2;


$$ language SQL

--Un Enseignant ne peut consulter seulement les notes des Élève qui sont inscrit aux modules de l'enseignant
```

un étudiant ne peut consulter que ses propres notes :

```
create or replace function consulter_notes_etudiant(pid_etudiant INT)
  returns table (id_etudiant INT, notes NUMERIC)
as $$
begin
  return query
  select n.id_etudiant, n.notes
  from note n
  where n.id_etudiant = p_id_etudiant;

  return;
end;
$$ language plpgsql
    security definer;
```

un enseignant saisit les notes de ses contrôles :

```
create or replace function saisir_note_ens(id_ens int,id_etudiant int, id_controle int, note int)
    returns void as
$$
begin
if exists (select $1 from controle c
    inner join module m on c.id_module = m.id_module where c.id_controle = id_controle
    and m.id_responsable = id_ens) then
        insert into notes(id_etudiant,id_controle,note)
        values(id_etudiant,id_controle,note);
end if;
end;
$$ language plpgsql
    security definer;
```