
SAE BASES DE DONNÉES ET LANGAGE SQL

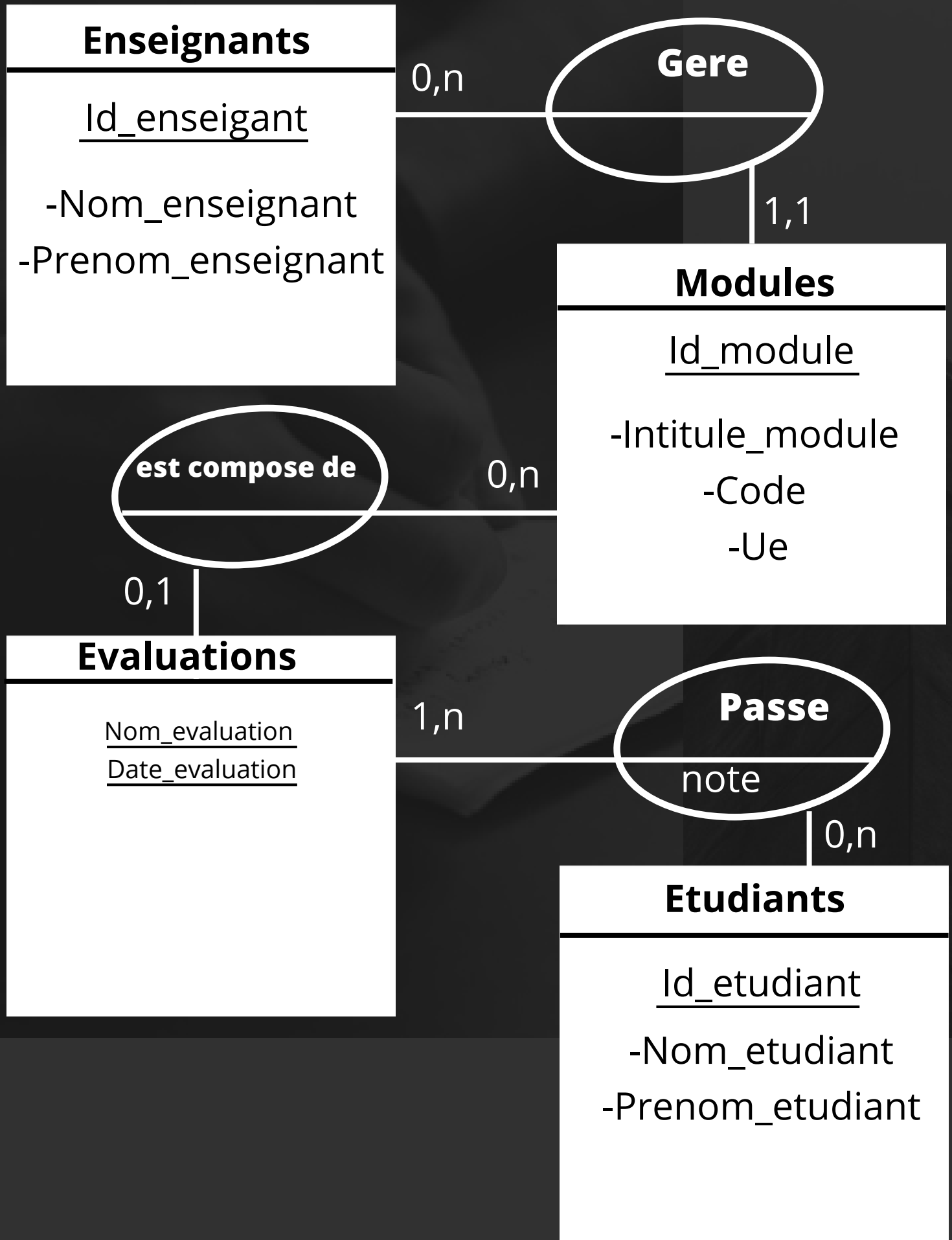
15/01/2023



S104

- 1 Modélisation et script de création
<sans AGL> P1 --> P3
- 2 Modélisation et script de création
<avec AGL> P4 --> P8
- 3 Peuplement des tables de la base
de données P9 --> P12

1) Modèle entités-associations:




2) SCHÉMA RELATIONNEL

- Enseignants (id_enseignant, nom_enseignant, prenom_enseignant)
- Modules (id_module, intitule_module, code, ue, id_enseignant) ou id_enseignant est une clef étrangère qui fait référence a Enseignants.
- Evaluations (nom_evaluation, date_evaluation, id_module) ou id_module est une clef étrangère qui fait référence a Modules.
- PASSE (nom_evaluation,date_evaluation,id_etudiant, note) ou id_module et id_etudiant sont des clefs étrangère qui font respectivement référence a Evaluations et Etudiants.
- Etudiants (id_etudiant, nom_etudiant, prenom_etudiant)

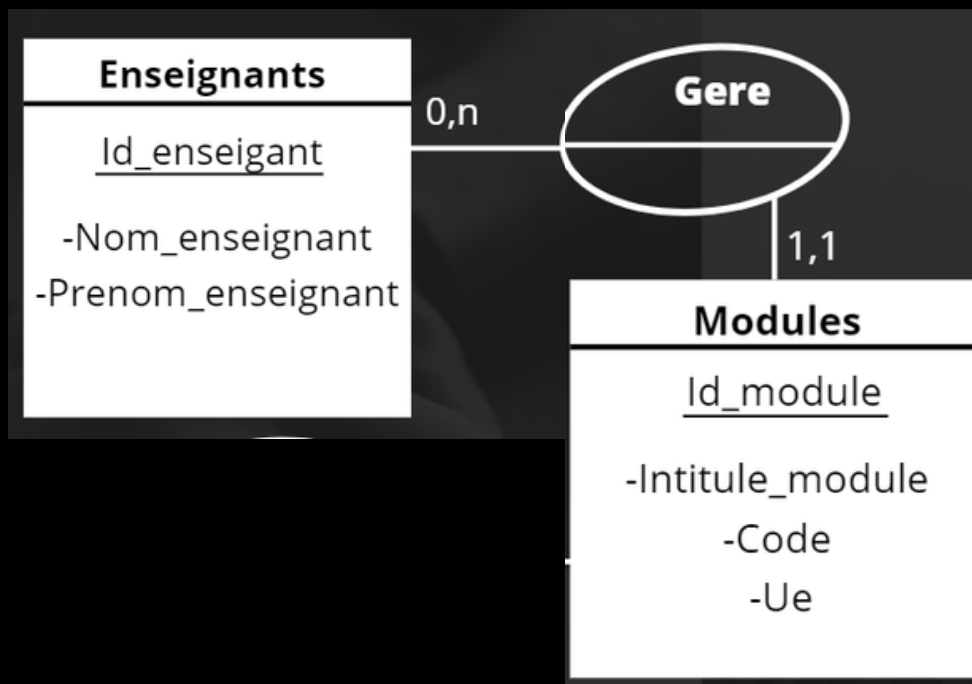
3) Script SQL de création des tables

```
CREATE TABLE Enseignants(  
    id_enseignant INTEGER PRIMARY KEY,  
    nom_enseignant VARCHAR,  
    prenom_enseignant VARCHAR  
);  
CREATE TABLE Modules(  
    id_module INTEGER PRIMARY KEY,  
    intitule_module VARCHAR,  
    ue VARCHAR,  
    code CHAR,  
    id_enseignant integer REFERENCES Enseignants(id_enseignant)  
);  
CREATE TABLE Evaluations(  
    nom_evaluation VARCHAR,  
    date_evaluation DATE,  
    id_module INTEGER REFERENCES Modules(id_module),  
    PRIMARY KEY (nom_evaluation,date_evaluation)  
);  
CREATE TABLE Etudiants(  
    id_etudiant INTEGER PRIMARY KEY,  
    nom_etudiant VARCHAR,  
    prenom_etudiant VARCHAR  
);  
CREATE TABLE PASSE(  
    id_etudiant INTEGER REFERENCES Etudiants (id_etudiant),  
    nom_evaluation VARCHAR,  
    date_evaluation DATE,  
    note FLOAT,  
    PRIMARY KEY (nom_evaluation,date_evaluation,id_etudiant),  
    FOREIGN KEY (nom_evaluation,date_evaluation) REFERENCES evaluations  
(nom_evaluation,date_evaluation)  
);
```



1) Illustrations comparatives cours/AGL commentée d'une association fonctionnelle

Illustrations (cours) d'une association fonctionnelle :



Illustrations (AGL) d'une association fonctionnelle :

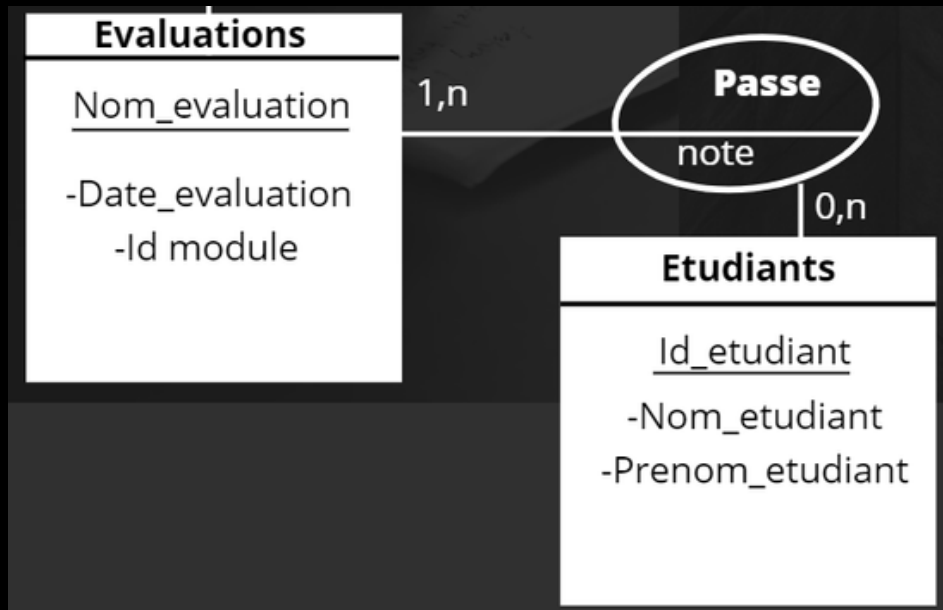


Commentaire:

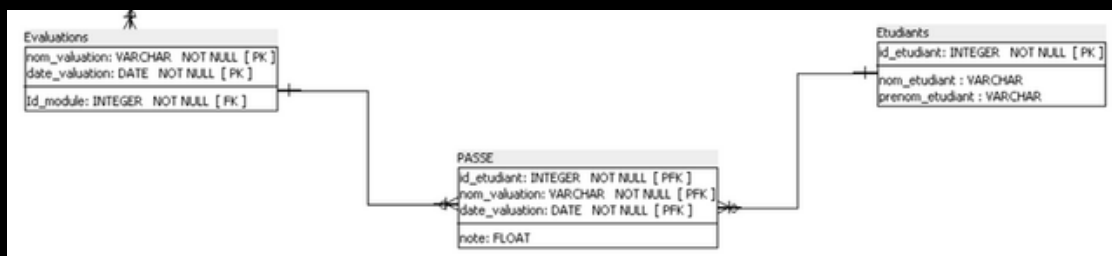
Pour commencer, on peut dire que les deux illustrations nous guide pratiquement vers une même idée mais avec une légère différences. En effet, en ce qui concerne les cardinalités et les types associations, ils sont ici présentes seulement dans l'illustration du cours et pas dans celle de l'AGL. De plus, les noms des colonnes sont seulement indiqués dans l'illustration du cours, or, dans l'illustration de l'AGL on précise les types de chaque colonnes et d'autres informations complémentaires aussi . Et finalement, concernant les clefs étrangères et primaires, ils y sont (les clefs étrangères) mais implicitement dans l'illustration du cours alors qu'ils sont précisés dans l'illustration d'AGL avec le mot "FK". Et les clefs primaires sont soulignés dans l'illustration du cours alors qu'ils sont précisés par le mot "PK" dans l'illustration d'AGL.

2) Illustrations comparatives cours/AGL commentée d'une association maillée

Illustrations (cours) d'une association maillée :



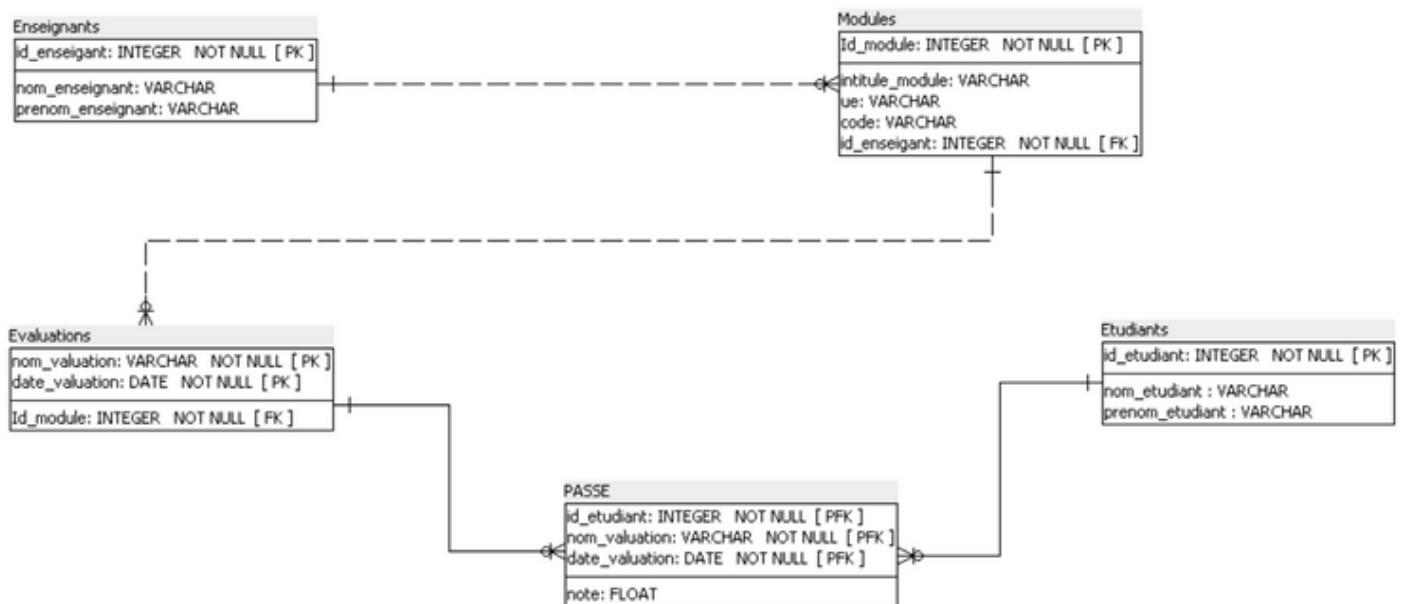
Illustrations (AGL) d'une association maillée:



Commentaire:

Pour commencer, on peut dire que les deux illustrations nous guide pratiquement vers une même idée mais avec une légère différences. En effet, en ce qui concerne les cardinalités, ils sont ici présentes seulement dans l'illustration du cours et pas dans celle de l'AGL. Aussi, le type association est présent dans l'illustration du cours mais transformé en type entité dans l'illustration de l'AGL. De plus, les noms des colonnes sont seulement indiqués dans l'illustration du cours, or, dans l'illustration de l'AGL on précise les types de chaque colonnes et d'autres informations complémentaires aussi . Et finalement, concernant les clefs étrangères et primaires, ils y sont (les clefs étrangères) mais implicitement dans l'illustration du cours alors qu'ils sont précisés dans l'illustration d'AGL avec le mot "FK". Et les clefs primaires sont soulignés dans l'illustration du cours alors qu'ils sont précisés par le mot "PK" dans l'illustration d'AGL.

3) Le modèle entités-associations réalisé avec l'AGL



4) Le script SQL par AGL

```
CREATE SEQUENCE etudiants_id_etudiant_seq;
```

```
CREATE TABLE Etudiants (
    id_etudiant INTEGER NOT NULL DEFAULT nextval('etudiants_id_etudiant_seq'),
    nom_etudiant VARCHAR,
    prenom_etudiant VARCHAR,
    CONSTRAINT id_etudiant PRIMARY KEY (id_etudiant)
);
```

```
ALTER SEQUENCE etudiants_id_etudiant_seq OWNED BY Etudiants.id_etudiant;
```

```
CREATE SEQUENCE enseignants_id_enseignant_seq;
```

```
CREATE TABLE Enseignants (
    id_enseignant INTEGER NOT NULL DEFAULT nextval('enseignants_id_enseignant_seq'),
    nom_enseignant VARCHAR,
    prenom_enseignant VARCHAR,
    CONSTRAINT id_enseignant PRIMARY KEY (id_enseignant)
);
```

```
ALTER SEQUENCE enseignants_id_enseignant_seq OWNED BY Enseignants.id_enseignant;
```

```
CREATE SEQUENCE modules_id_module_seq;
```

```
CREATE TABLE Modules (
    id_module INTEGER NOT NULL DEFAULT nextval('modules_id_module_seq'),
    intitule_module VARCHAR,
    ue VARCHAR,
    code VARCHAR,
    id_enseignant INTEGER NOT NULL,
    CONSTRAINT id_module PRIMARY KEY (id_module)
);
```

```
ALTER SEQUENCE modules_id_module_seq OWNED BY Modules.id_module;
```

```
CREATE TABLE Evaluations (
    nom_valuation VARCHAR NOT NULL,
    date_valuation DATE NOT NULL,
    id_module INTEGER NOT NULL,
    CONSTRAINT nom_valuation PRIMARY KEY (nom_valuation, date_valuation)
);
```

```
CREATE TABLE PASSE (
    id_etudiant INTEGER NOT NULL,
    nom_valuation VARCHAR NOT NULL,
    date_valuation DATE NOT NULL,
    note REAL,
    CONSTRAINT nom_evaluation_id_etudiant PRIMARY KEY (id_etudiant, nom_valuation, date_valuation)
);
```

```
ALTER TABLE PASSE ADD CONSTRAINT etudiants_passe_fk
FOREIGN KEY (id_etudiant)
REFERENCES Etudiants (id_etudiant)
ON DELETE NO ACTION
ON UPDATE NO ACTION
NOT DEFERRABLE;
```

```
ALTER TABLE Modules ADD CONSTRAINT enseignants_modules_fk
FOREIGN KEY (id_enseignant)
REFERENCES Enseignants (id_enseignant)
ON DELETE NO ACTION
ON UPDATE NO ACTION
NOT DEFERRABLE;
```

```
ALTER TABLE Evaluations ADD CONSTRAINT modules_evaluations_fk
FOREIGN KEY (id_module)
REFERENCES Modules (id_module)
ON DELETE NO ACTION
ON UPDATE NO ACTION
NOT DEFERRABLE;
```

```
ALTER TABLE PASSE ADD CONSTRAINT evaluations_passe_fk
FOREIGN KEY (nom_valuation, date_valuation)
REFERENCES Evaluations (nom_valuation, date_valuation)
ON DELETE NO ACTION
ON UPDATE NO ACTION
NOT DEFERRABLE;
```

5) La difference entre les deux script SQL

Dans cette partie, on analyseras et on indiqueras les différences entre le script produit manuellement et celui produit par l'AGL. Tout d'abord, on peut remarquer que le script proposé par AGL est beaucoup plus long que celui que j'ai réalisé manuellement. Il contient plusieurs instructions qu'on a pas vu en cours comme par exemple CREATE SEQUENCE, qui permet de créer un objet séquence et spécifie ses propriétés, ALTER SEQUENCE, qui permet de modifier les paramètres d'un générateur de séquence, etc... . De plus, dans toute les colonnes du script produit par l'AGL, on remarque qu'il est précisé l'attribut "not nul" ce qui signifie que la colonne ne peut pas être vide. Ensuite , les clef primaire sont déclarée avec l'attribut CONSTRAINT , ce qui permet de poser une contrainte sur une colonne d'une table.

PARTIE 3 :

1) DESCRIPTION COMMENTÉE DES DIFFÉRENTES ÉTAPES DE VOTRE SCRIPT DE PEUPLEMENT

```
CREATE TABLE data (  
  id_enseignant INTEGER ,  
  nom_enseignant VARCHAR,  
  prenom_enseignant VARCHAR,  
  id_module INTEGER,  
  ue VARCHAR,  
  code VARCHAR,  
  intitule_module VARCHAR,  
  nom_evaluation VARCHAR,  
  date_evaluation DATE,  
  note FLOAT,  
  id_etudiant INTEGER,  
  nom_etudiant VARCHAR,  
  prenom_etudiant VARCHAR  
);  
\copy data FROM 'C:\Users\Public\sae104_data.csv\data.csv' DELIMITER ';' CSV HEADER ;  
SELECT * FROM data ;  
INSERT INTO enseignants SELECT DISTINCT id_enseignant,nom_enseignant,prenom_enseignant  
FROM data;  
SELECT * FROM enseignants ;  
INSERT INTO modules SELECT DISTINCT id_module,ue,code,intitule_module,id_enseignant  
FROM data;  
SELECT * FROM modules ;  
INSERT INTO evaluations SELECT DISTINCT nom_evaluation,date_evaluation,id_modules FROM  
data;  
SELECT * FROM evaluations ;  
INSERT INTO etudiants SELECT DISTINCT id_etudiant,nom_etudiant,prenom_etudiant FROM  
data;  
SELECT * FROM etudiant ;  
INSERT INTO passe SELECT DISTINCT nom_evaluation,date_evaluation,note,id_etudiant FROM  
data;  
SELECT * FROM passe ;
```

1) DESCRIPTION COMMENTÉE DES DIFFÉRENTES ÉTAPES DE VOTRE SCRIPT DE PEUPLEMENT

Commentaire:

D'emblée, en ce qui concerne ce script de peuplement, je l'ai réalisé en trois grandes parties.

Tout d'abord, je suis passé par une table intermédiaire ,qui est data, que j'ai crée qui me permettra par la suite l'insertion des différentes données dans mes tables.

Ensuite, j'ai inséré tout les données du fichier csv que vous nous avez fournit dans ma table data et j'ai bien vérifier aussi par la suite que ce dernier contient bien tout les données du fichier.

Et finalement, j'ai inséré les données de chaque table à partir de notre table data et après chaque insertion je vérifie bien que mes données ont bien été insérer correctement.

2) PRÉSENTATION COMMENTÉE DE DEUX REQUÊTES INTÉRESSANTES SUR LA BASE DE DONNÉES

Nous nous pencherons d'abord sur la moyenne, qui est le fait de faire la somme des résultats, divisée par le nombre d'étudiants ayant fait l'examen. . Afin de calculer la moyenne dans SQL, nous devons utiliser la fonction `avg(name_of_the_colone)` dans notre requête. L'exemple suivant montre son usage :

```
sae=# SELECT avg(nom_de_la_colonne) FROM nom_de_la_table ;
```

Par conséquent, nous devons utiliser la fonction « SELECT » dans laquelle nous allons indiquer la colonne sur laquelle on veut appliquer la « avg ». il reste à confirmer la table dans laquelle se trouve la colonne avec l'indication "FROM".

Il en résultera ce qui suit :

```
sae=# SELECT avg(note) FROM passe ;
      avg
-----
 9.588844544023978
(1 ligne)
```

Personnellement, j'ai choisit cette première requête SQL parce que je trouve que cette dernière est très utile surtout avec des bases de données plus grandes et elle répond efficacement aux besoins de l'utilisateur et dans le cas de notre SAE, elle nous ai utile pour avoir la moyenne général du groupe par exemple.

2) PRÉSENTATION COMMENTÉE DE DEUX REQUETES INTÉRESSANTES SUR LA BASE DE DONNÉES

Et maintenant, nous allons voir une fonction qui nous permet de calculer un écart-type, qui représente le degré de dispersion des résultats autour de la moyenne. Plus les résultats sont largement distribués autour de la moyenne, plus l'écart-type est élevé et plus l'examen est discriminant. Nous allons exécuter le calcul de l'écart-type d'une colonne. Afin de le calculer dans SQL, nous devons utiliser la fonction `stddev(name_of_the_colone)` dans notre requête. L'exemple suivant montre son usage :

```
sae=# SELECT stddev(nom_de_la_colonne) FROM nom_de_la_table ;
```

Par conséquent, nous devons utiliser la fonction « SELECT » dans laquelle nous allons indiquer la colonne sur laquelle on veut appliquer la « stddev », il reste à confirmer la table dans laquelle se trouve la colonne avec l'indication "FROM".

Il en résultera ce qui suit :

```
sae=# SELECT stddev(note) FROM passe ;
      stddev
-----
 5.176560058899757
(1 ligne)
```

Personnellement, j'ai choisit cette deuxième requête SQL parce que je trouve que cette dernière est également très utile et très intéressantes surtout avec des bases de données plus grandes celles des entreprises par exemple lors de leurs études statistiques etc... Et elle répond aussi bien efficacement aux besoins de l'utilisateur et dans le cas de notre SAE, elle nous ai utile pour avoir des informations qui peuvent être utiles pour chaque enseignants sur sa matières, ses contrôles, ses groupes par exemple, etc...