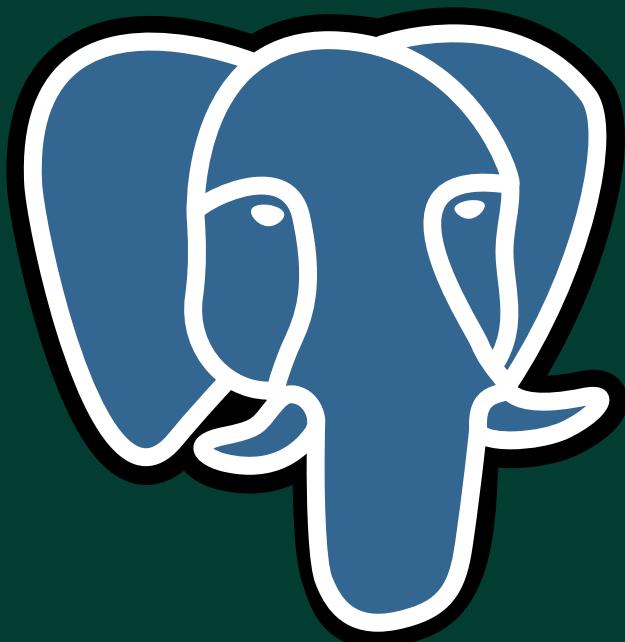


SAE S104

Bases de données et langage SQL

Création d'une base de données



PostgreSQL

Réalisé par

Thurairajasingam Kavusikan

Groupe Shango

BUT1 Informatique

Sommaire :

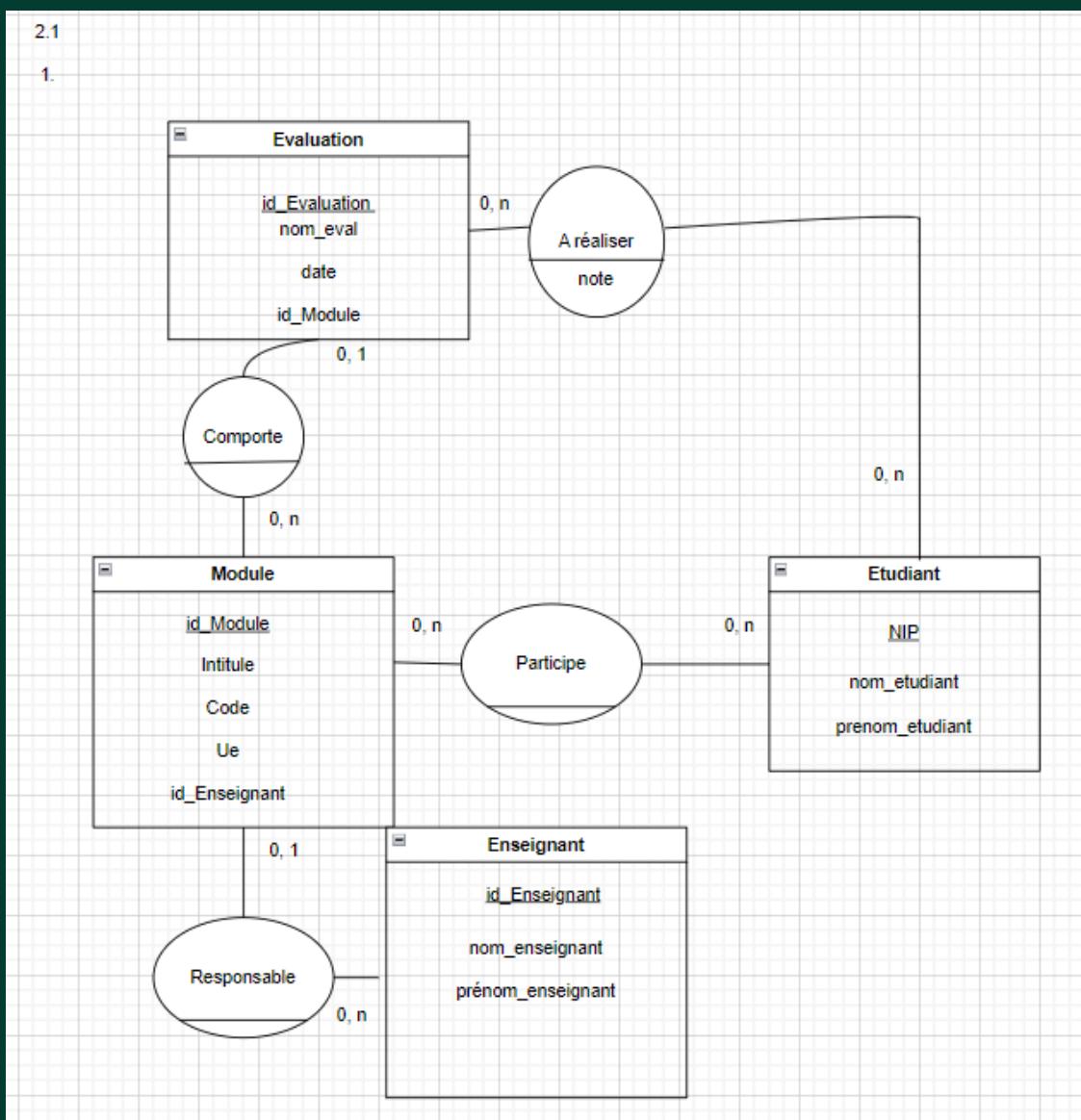
1-Modélisation et script de création
" sans AGL "

2-Modélisation et script de création
" avec AGL "

3-Peuplement des tables et requêtes

1- Modélisation et script de création " sans AGL "

1. Modèle entités-associations



2. Schéma relationnel

2.

Schéma relationnel :

- `Evaluation(id_Evaluation, nom_eval, date, id_Module)` `id_Module` fait référence à `Module`
- `Module(id_Module, Intitule, Code, Ue)` `id_EsEnseignant` fait référence à `Enseignant`
- `Enseignant(id_EsEnseignant, nom_enseignant, prénom_enseignant)`
- `Etudiant(NIP, nom_etudiant, prénom_etudiant)`
- `A realiser(id_Evaluation, NIP, note)` `id_Evaluation` fait référence à `Evaluation`, `NIP` fait référence à `Etudiant`
- `Participe(id_Module, NIP)` `id_Module` fait référence à `Module`, `NIP` fait référence à `Etudiant`

3 . Script SQL

```
CREATE TABLE Enseignant (
    id_Enseignant INTEGER PRIMARY KEY,
    nom_enseignant VARCHAR,
    prenom_enseignant VARCHAR
);
```

```
CREATE TABLE Etudiant (
    NIP    INTEGER PRIMARY KEY,
    nom_etudiant VARCHAR,
    prenom_etudiant VARCHAR
);
```

```
CREATE TABLE Module (
    id_Module INTEGER PRIMARY KEY,
    Intitule VARCHAR,
    Ue VARCHAR,
    Code VARCHAR,
    id_Enseignant INTEGER REFERENCES
    Enseignant(id_Enseignant)
);
```

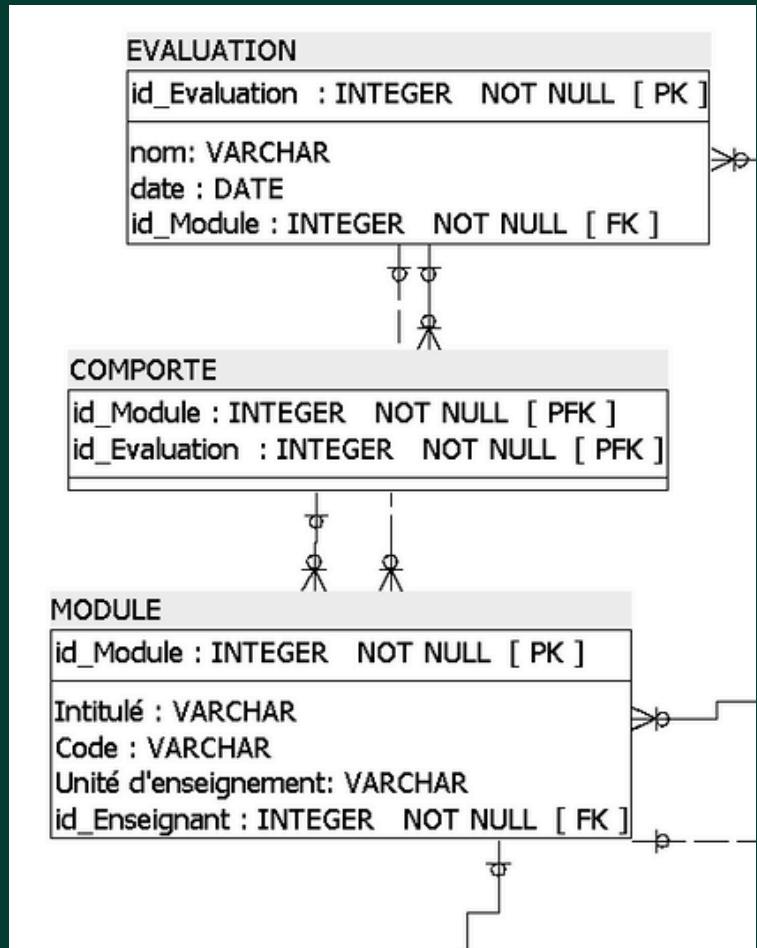
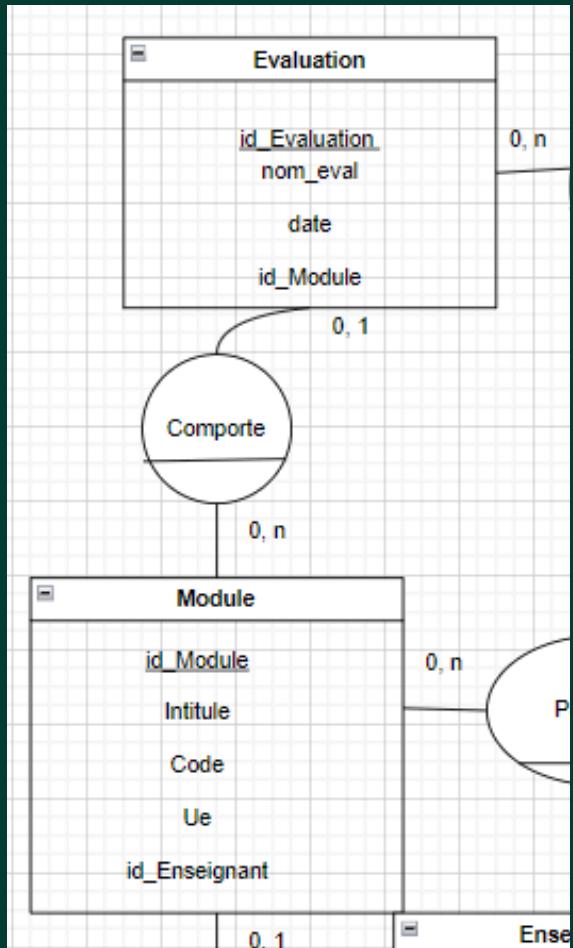
```
CREATE TABLE Evaluation (
    id_Evaluation SERIAL PRIMARY KEY,
    nom_eval VARCHAR,
    date DATE,
    id_Module INTEGER REFERENCES Module(id_Module)
);
```

```
CREATE TABLE A_realiser (
    id_Evaluation INTEGER REFERENCES
Evaluation(id_Evaluation),
    id_etudiant INTEGER REFERENCES Etudiant(NIP),
    note FLOAT,
    PRIMARY KEY (id_Evaluation,id_etudiant)
);
```

```
CREATE TABLE Participe (
    id_Module INTEGER REFERENCES Module(id_Module),
    id_etudiant INTEGER REFERENCES Etudiant(NIP),
    PRIMARY KEY (id_Module,id_etudiant)
);
```

2- Modélisation et script de création " avec AGL "

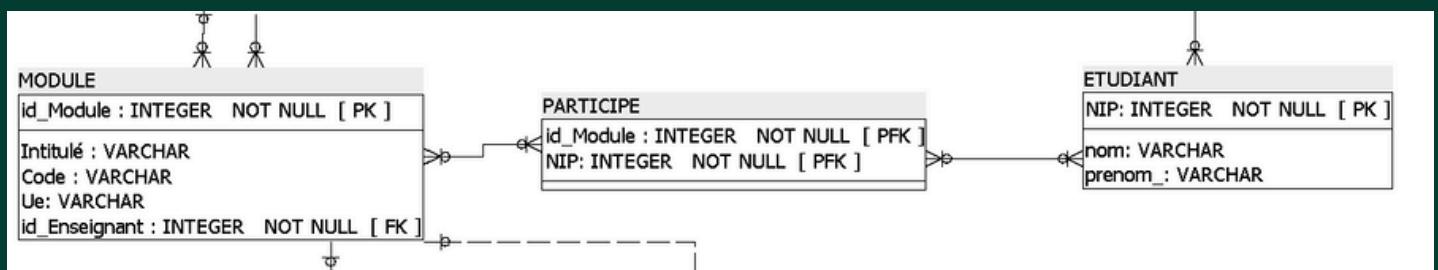
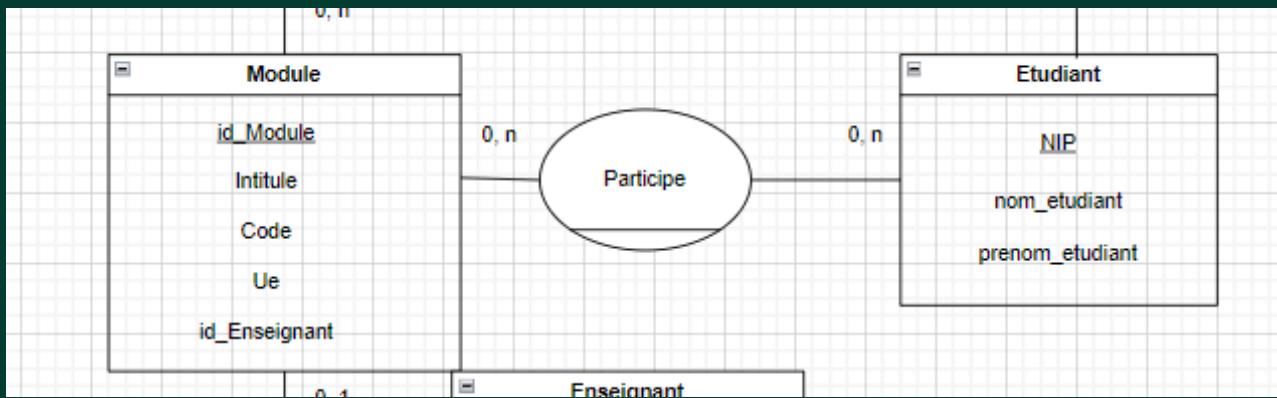
1. Illustration comparative association fonctionnelle



Nous pouvons voir que dans la représentation AGL l'association "COMPORTE" possède les PRIMARY FOREIGN KEY (id_Module,id_Evaluation) alors que, dans l'entités-associations sans AGL nous les voyons pas explicitement.

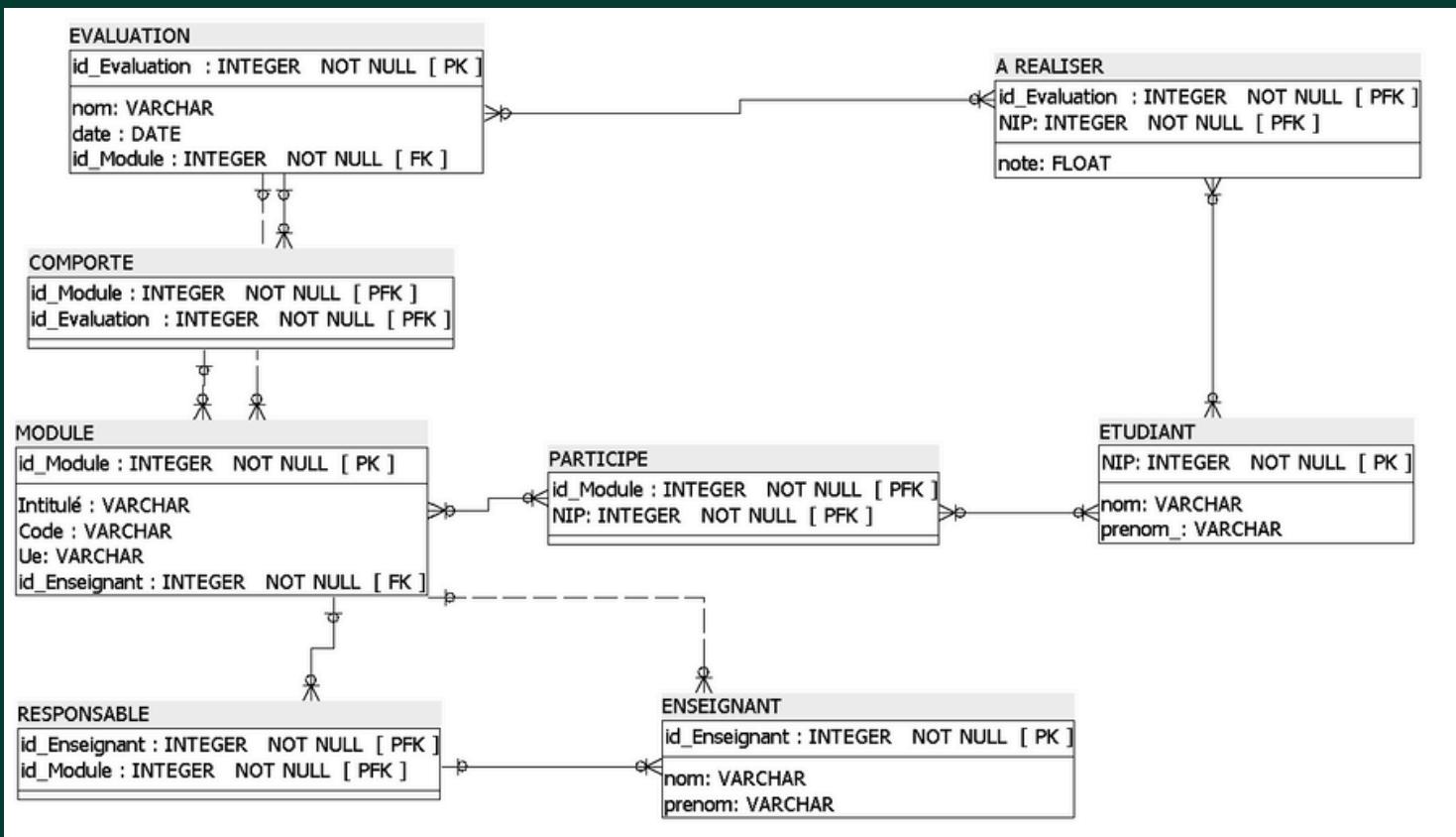
Nous pouvons constater qu'il y a des similitudes aussi les clés primaires, les cardinalités et les valeurs des tables.

2. Illustration comparative association maillée



Nous pouvons voir qu'il y a les mêmes valeurs dans les tables, les cardinalités sont représenter différemment, on peut voir spécifiquement les PRIMARY FOREIGN KEY de l'association "PARTICIPE" de l'AGL.

3. Modèle entités-associations réalisé avec AGL



4. Script SQL généré par l'AGL

```
CREATE TABLE public.ENSEIGNANT (
    id_Enseignant INTEGER NOT NULL,
    nom VARCHAR,
    prenom VARCHAR,
    CONSTRAINT id_enseignant PRIMARY KEY (id_Enseignant)
);

CREATE TABLE public.MODULE (
    id_Module INTEGER NOT NULL,
    Intitule VARCHAR,
    Code VARCHAR,
    Ue VARCHAR,
    id_Enseignant_ INTEGER NOT NULL,
    CONSTRAINT id_module PRIMARY KEY (id_Module)
);

CREATE TABLE public.RESPONSABLE (
    id_Enseignant_ INTEGER NOT NULL,
    id_Module_ INTEGER NOT NULL,
    CONSTRAINT id_moduleid_enseignant PRIMARY KEY (id_Enseignant,
    id_Module)
);

CREATE TABLE public.ETUDIANT (
    NIP INTEGER NOT NULL,
    nom VARCHAR,
    prenom VARCHAR,
    CONSTRAINT id_etudiant PRIMARY KEY (NIP)
);
```

4 .

```
CREATE TABLE public.PARTICIPE (
    id_Module INTEGER NOT NULL,
    NIP INTEGER NOT NULL,
    CONSTRAINT id_modulenip PRIMARY KEY (id_Module, NIP)
);
```

```
CREATE TABLE public.EVALUATION (
    id_Evaluation INTEGER NOT NULL,
    nom VARCHAR,
    date DATE,
    id_Module INTEGER NOT NULL,
    CONSTRAINT id_evaluation PRIMARY KEY (id_Evaluation)
);
```

```
CREATE TABLE public.COMPORTE (
    id_Module INTEGER NOT NULL,
    id_Evaluation INTEGER NOT NULL,
    CONSTRAINT id_evaluationid_module PRIMARY KEY (id_Module,
    id_Evaluation)
);
```

```
CREATE TABLE public.A_REALISER (
    id_Evaluation INTEGER NOT NULL,
    NIP INTEGER NOT NULL,
    note REAL,
    CONSTRAINT id_etudiantid_evaluation PRIMARY KEY
    (id_Evaluation, NIP)
);
```

4 .

```
ALTER TABLE public.RESPONSABLE ADD CONSTRAINT
enseignant_responsable_fk
FOREIGN KEY (id_Engseignant)
REFERENCES public.ENSEIGNANT (id_Engseignant)
ON DELETE NO ACTION
ON UPDATE NO ACTION
NOT DEFERRABLE;

ALTER TABLE public.MODULE ADD CONSTRAINT enseignant_module_fk
FOREIGN KEY (id_Engseignant)
REFERENCES public.ENSEIGNANT (id_Engseignant)
ON DELETE NO ACTION
ON UPDATE NO ACTION
NOT DEFERRABLE;

ALTER TABLE public.RESPONSABLE ADD CONSTRAINT
module_responsable_fk
FOREIGN KEY (id_Module)
REFERENCES public.MODULE (id_Module)
ON DELETE NO ACTION
ON UPDATE NO ACTION
NOT DEFERRABLE;

ALTER TABLE public.PARTICIPE ADD CONSTRAINT module_participe_fk
FOREIGN KEY (id_Module)
REFERENCES public.MODULE (id_Module)
ON DELETE NO ACTION
ON UPDATE NO ACTION
NOT DEFERRABLE;

ALTER TABLE public.COMPORTE ADD CONSTRAINT module_comporte_fk
FOREIGN KEY (id_Module)
REFERENCES public.MODULE (id_Module)
ON DELETE NO ACTION
ON UPDATE NO ACTION
NOT DEFERRABLE;
```

4 .

```
ALTER TABLE public.EVALUATION ADD CONSTRAINT module_evaluation_fk
FOREIGN KEY (id_Module)
REFERENCES public.MODULE (id_Module)
ON DELETE NO ACTION
ON UPDATE NO ACTION
NOT DEFERRABLE;

ALTER TABLE public.A_REALISER ADD CONSTRAINT etudiant_a_realiser_fk
FOREIGN KEY (NIP)
REFERENCES public.ETUDIANT (NIP)
ON DELETE NO ACTION
ON UPDATE NO ACTION
NOT DEFERRABLE;

ALTER TABLE public.PARTICIPE ADD CONSTRAINT etudiant_participe_fk
FOREIGN KEY (NIP)
REFERENCES public.ETUDIANT (NIP)
ON DELETE NO ACTION
ON UPDATE NO ACTION
NOT DEFERRABLE;

ALTER TABLE public.A_REALISER ADD CONSTRAINT evaluation_a_realiser_fk
FOREIGN KEY (id_Evaluation)
REFERENCES public.EVALUATION (id_Evaluation)
ON DELETE NO ACTION
ON UPDATE NO ACTION
NOT DEFERRABLE;

ALTER TABLE public.COMPORTE ADD CONSTRAINT evaluation_comporte_fk
FOREIGN KEY (id_Evaluation)
REFERENCES public.EVALUATION (id_Evaluation)
ON DELETE NO ACTION
ON UPDATE NO ACTION
NOT DEFERRABLE;
```

5 . D i f f e r e n c e e n t r e l e s s c r i p t s p r o d u i t s m a n u e l l e m e n t e t a u t o m a t i q u e m e n t .

Nous pouvons constater que les CREATE TABLE sont pratiquement les mêmes, sauf que dans le script généré par l'AGL les PRIMARY KEY sont nommés en tant que NOT NULL et les noms des tables sont précédé par le répertoire d'où ils ont été créées.

De plus, dans le script AGL il y a des ALTER TABLE ce qui permet de modifier une table existante, on peut par exemple ajouter, supprimer ou modifier une colonne.

3-Peuplement des tables et requêtes

1. Description des différents étapes du script de peuplement

Tout d'abord, nous avons créer la table scene afin de copier le fichier plat fourni dans nos tables à l'aide de notre script. Suite à cela nous avons insérer les éléments contenu dans le fichier dans les tables.

```
CREATE TABLE scene (
    id_enseignant INTEGER,
    nom_enseignant VARCHAR,
    prenom_enseignant VARCHAR,
    id_module INTEGER,
    code VARCHAR,
    ue VARCHAR,
    intitule_module VARCHAR,
    nom_evaluation VARCHAR,
    date_evaluation VARCHAR,
    note FLOAT,
    id_etudiant INTEGER,
    nom_etudiant VARCHAR,
    prenom_etudiant VARCHAR
);
copy scene
from '\Users\Keshiyan\Desktop\data.csv'
DELIMITER ';' CSV HEADER;
```

```
INSERT INTO Enseignant (
SELECT DISTINCT id_enseignant,
nom_enseignant, prenom_enseignant
FROM scene);

INSERT INTO Module (
SELECT DISTINCT id_Module, Code,
Intitule, Ue, id_Engseignant
FROM scene);

INSERT INTO Evaluation (
nom_eval, date, id_module) (SELECT DISTINCT
nom_evaluation, date_evaluation, id_module
FROM scene);

INSERT INTO Etudiant (
SELECT DISTINCT id_etudiant, nom_etudiant,
prenom_etudiant
FROM scene);

ALTER TABLE A_realiser
DROP constraint A_realiser_pkey cascade;

INSERT INTO A_realiser (
SELECT DISTINCT id_Evaluation, id_Etudiant,
note
FROM scene, Evaluation);
```

2. Présentation commentée de deux requêtes sur la base de données

```
SELECT id_enseignant,id_module  
FROM Module  
ORDER BY id_module;
```

J'ai choisis de présenter cette requête car elle nous permet de vérifier qu'il peut avoir qu'un seul responsable pour un module.

```
SELECT nom_enseignant  
FROM Enseignant  
WHERE prenom_enseignant = 'Frederic' ;
```

	nom_enseignant character varying	🔒
1	Larmonier	
2	Selosse	

J'ai choisis de présenter cette requête car on peut constater que il y a deux personne qui ont le même prénom.