Bases de Données et langage SQL

Situation d'Apprentissage et d'Evaluation (SAE)S104

Création d'une base de données

Nom *: Hatem*

Prénom *: Kenza*

Groupe *: Zeus*

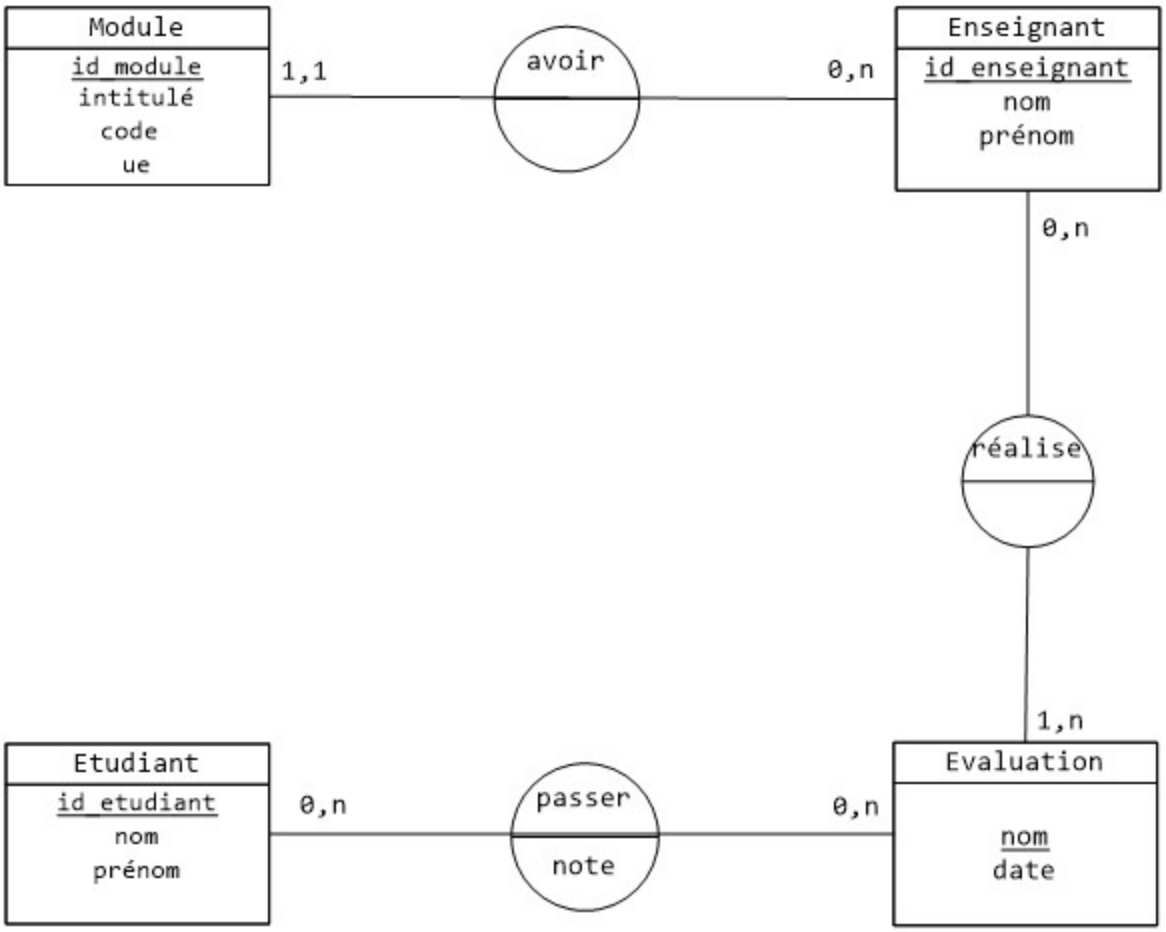
Chargé de TD *: M. Ellouze*

Promotion *: 1ère année BUT informatique*

Nom de la SAE *: S104 Base de Données*

SOMMAIRE :

2.1.1 Modèle entités-associations respectant la syntaxe du cours :



2.1.2.

Schéma Relationnel:

Enseignant (id_enseignant, nom, prenom)

Module (id_module, id_Enseignant, intitulé, code, ue) id_Enseignant qui fait référence à la table Enseignant.

Evaluation (nom, date, id_module) où id_module fait référence à la table Module.

Realisation(id_Enseignant,nom_evaluation) où id_Enseignant et nom_evaluation font référence aux tables Enseignant et Evaluation.

Etudiant (id_etudiant, nom, prenom)

Passation (id_Etudiant, nom_evaluation, note) où id_Etudiant et nom_evaluation font référence aux tables Etudiant et Evaluation.

2.1.3

Script SQL de création des tables :

```
CREATE TABLE Enseignant (
      id_enseignant INTEGER PRIMARY KEY ,
      nom_enseignant VARCHAR NOT NULL,
      prenom_enseignant VARCHAR NOT NULL) ;


CREATE TABLE Module (
      id_module INTEGER PRIMARY KEY ,
      id_enseignant INTEGER REFERENCES Enseignant ON DELETE SET NULL,
      code VARCHAR ,
      ue VARCHAR ,
      intitule_module VARCHAR);


CREATE TABLE Evaluation (
      nom_evaluation VARCHAR PRIMARY KEY,
      id_module INTEGER REFERENCES Module ON DELETE SET NULL ,
      date_evaluation DATE NOT NULL ) ;


CREATE TABLE Realisation (
```

```
      id_enseignant INTEGER REFERENCES Enseignant,
      nom_evaluation VARCHAR REFERENCES Evaluation ,
      PRIMARY KEY (id_enseignant, nom_evaluation)) ;
CREATE TABLE Etudiant (
      id_etudiant INTEGER PRIMARY KEY ,
      nom_etudiant VARCHAR NOT NULL ,
      prenom_etudiant VARCHAR NOT NULL ) ;
CREATE TABLE Passation (
      id_etudiant INTEGER REFERENCES Etudiant ON DELETE SET NULL ,
      nom_evaluation VARCHAR REFERENCES Evaluation (nom_evaluation) ON
DELETE SET NULL ,
      PRIMARY KEY ( id_etudiant , nom_evaluation) ,
      note REAL CHECK ( note > 0 )) ;
```

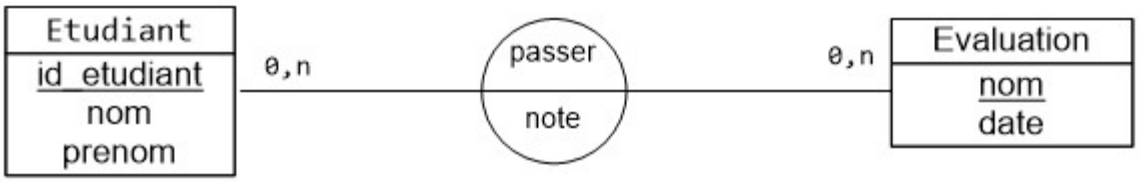
Liste des relations générés par postgreSQl : (\d)

```
            List of relations
 Schema |    Name     | Type  |  Owner
--------+-------------+-------+----------
 public | enseignant  | table | postgres
 public | etudiant    | table | postgres
 public | evaluation  | table | postgres
 public | module      | table | postgres
 public | passation   | table | postgres
 public | realisation | table | postgres
(6 rows)
```

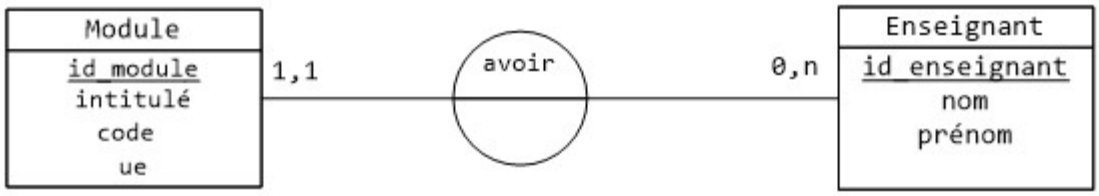
2.2.

Les deux associations maillé et fonctionnelle du cours :

Une association maillé du cours :



Une association fonctionnelle du cours :



Les deux associations générées par l'AGL :

Illustration d'une association maillé généré par AGL :

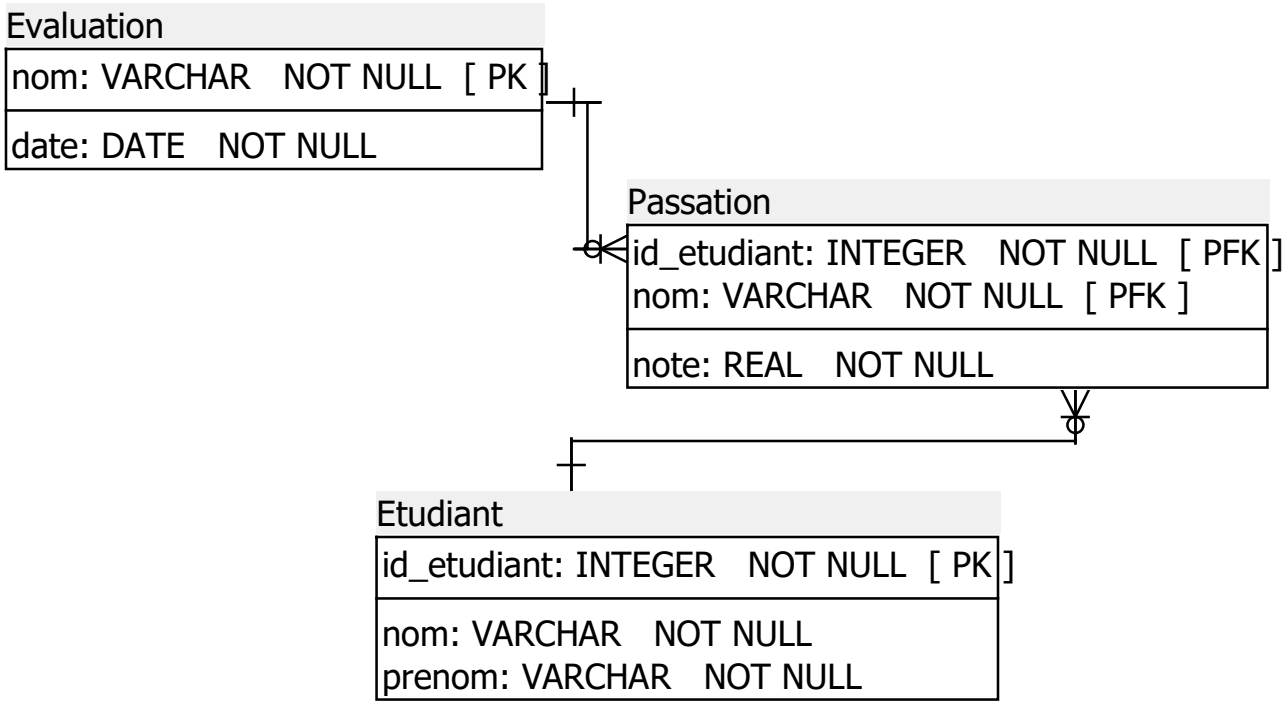
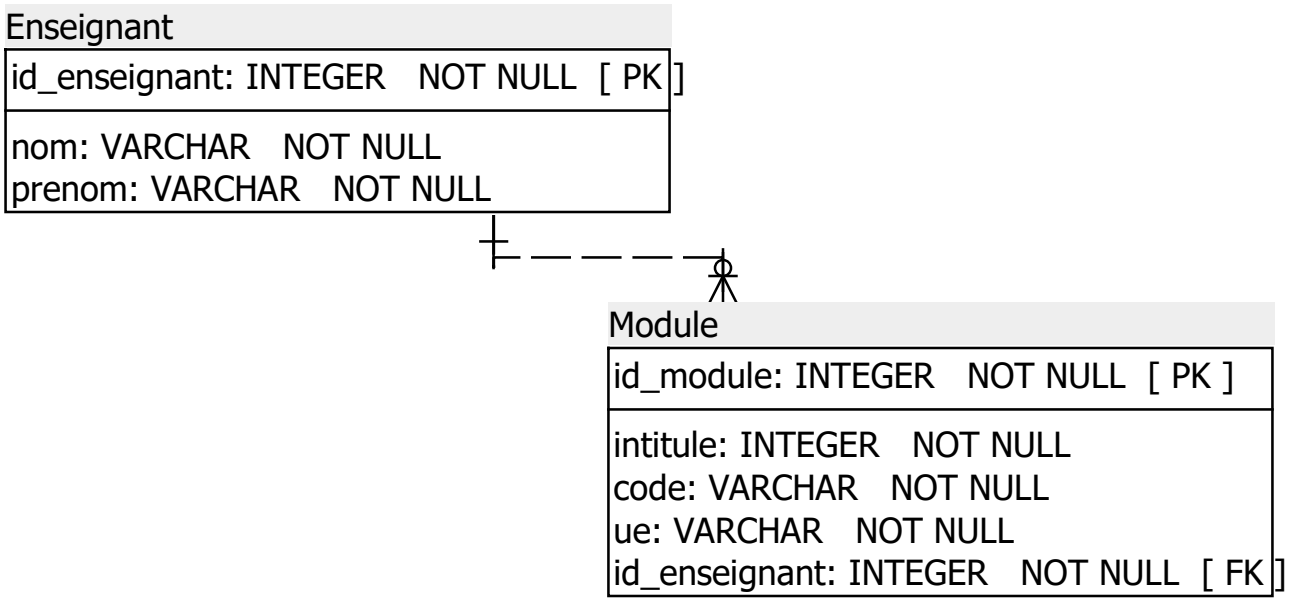


Illustration d'une association fonctionnelle généré par AGL :



2.2.1 Comparaison entre l'illustration d'une association maillé en cours et en AGL :

Une association maillé en cours :

- 2 type-entité (Etudiant , Evaluation) reliés par un type-association (passer) portant un attribut note.
- L'entité Etudiant contient 3 attributs : id_etudiant (clé primaire), nom , prenom.
- L'entité Evaluation contient 2 attributs : nom (clé primaire), date, un attribut id module qui apparait par sur le schéma qui fait référence à la table Module.
- Le schéma en plus il contient les cardinalités sur chaque liaison avec le type-association (0,n) en précisant le nombre minimal et maximal d'intervention d'une entité dans quelconque association.
- L'association « passer » contient un attribut note qui dépend des deux entités Etudiant et Evaluation.
- Les clés primaires sont désignés par un soulignement.
- Les types des attributs n'est pas précisés.

Une association maillé généré par l'AGL :

- Présentation des types entités et associations de la même façon (un tableau) en contenant les attributs.
- Présence des clés étrangères dans les tableaux : Les clés id_etudiant et nom dans l'association « Passation ».
- Les types des attributs (les éléments du tableau) sont bien précisés.
- Les clés primaires sont désignés de la présence d'un crochet ([PK] signifiant PRIMARY KEY).
- Les clés étrangères sont désignés de la présence d'un crochet ([PFK] signifiant PRIMARY FOREIGN KEY).
- Les clés primaires et les clés étrangères sont classés dans une case pour les distinguer des autres attributs.

2.2.2 Comparaison entre l'illustration d'une association fonctionnelle en cours et en AGL :

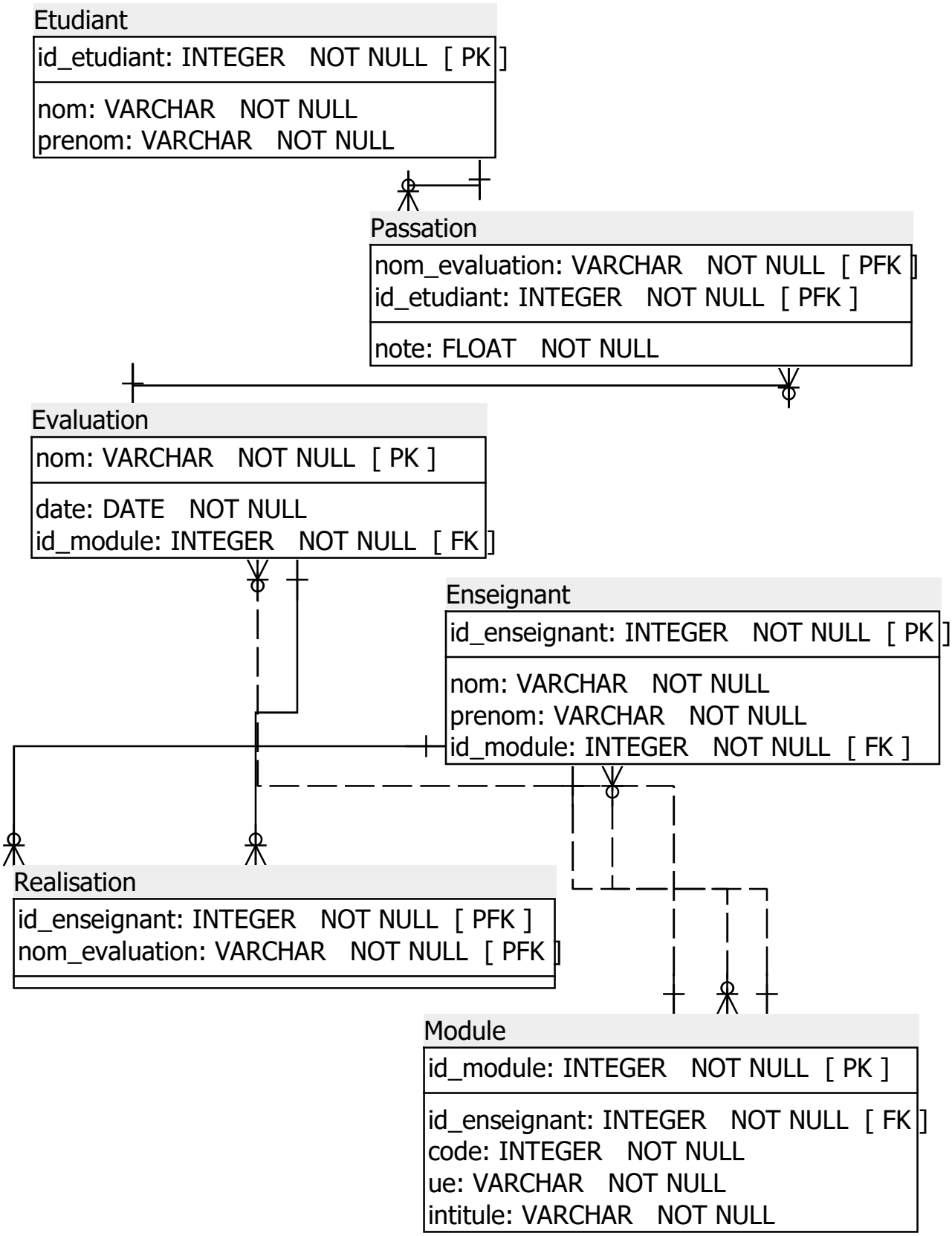
Une association fonctionnelle en cours :

- 2 type-entité (Enseignant , Module) reliés par un type-association (avoir).
- L'entité Enseignant contient 3 attributs : id_enseignant (clé primaire), nom , prenom.
- L'entité Module contient 4 attributs : id_module (clé primaire),intitulé, code, ue + une clé étrangère id enseignant qui apparait pas sur le schéma mais qui fait référence au tableau enseignant.
- Le schéma en plus il contient les cardinalités sur un côté de la une liaison avec le type-association (0,n) et de l'autre côté (1,1).
- L'association « avoir » ne contient pas d'attribut contrairement à l'association « passer ».
- Les clés primaires sont désignés par un soulignement.
- Les types des attributs n'est pas précisés.

Une association fonctionnelle généré par l'AGL :

- Présentation des types entités dans un tableau en contenant les attributs.
- Le type-association « avoir » n'est pas présenté dans le schéma contrairement à la présentation vu en cours.
- Présence des clés étrangères dans les tableaux : La clé id_enseignant dans le tableau « Module ».
- Les types des attributs (les éléments du tableau) sont bien précisés.
- Les clés primaires sont désignés de la présence d'un crochet ([PK] signifiant PRIMARY KEY).
- Les clés étrangères sont désignés de la présence d'un crochet ([FK] signifiant FOREIGN KEY).
- Les clés primaires et les clés étrangères sont classés dans une case pour les distinguer des autres attributs.

2.2.3. Modèle entités-associations réalisé avec l'AGL :



2.2.4.

Script SQL de création de tables généré automatiquement par l'AGL :

```
CREATE TABLE Enseignant (
id_enseignant INT NOT NULL,
nom VARCHAR NOT NULL,
prenom VARCHAR NOT NULL,
id_module INT NOT NULL,
PRIMARY KEY (id_enseignant)
);

CREATE TABLE Etudiant (
id_etudiant INT NOT NULL,
nom VARCHAR NOT NULL,
prenom VARCHAR NOT NULL,
PRIMARY KEY (id_etudiant)
);

CREATE TABLE Evaluation (
nom VARCHAR NOT NULL,
date_1 DATE NOT NULL,
id_module INT NOT NULL,
PRIMARY KEY (nom,date_1)
);


CREATE TABLE Passation (
nom_evaluation VARCHAR NOT NULL,
id_etudiant INT NOT NULL,
note DOUBLE PRECISIONS NOT NULL,
PRIMARY KEY (nom_evaluation, id_etudiant)
);
```

```
CREATE TABLE Realisation (
id_enseignant INT NOT NULL,
nom_evaluation VARCHAR NOT NULL,
PRIMARY KEY (id_enseignant, nom_evaluation)
);


CREATE TABLE Module (
id_module INT NOT NULL,
id_enseignant INT NOT NULL,
code INT NOT NULL,
ue VARCHAR NOT NULL,
intitule VARCHAR NOT NULL,
PRIMARY KEY (id_module)
);

ALTER TABLE Evaluation ADD CONSTRAINT module_evaluation_fk
FOREIGN KEY (id_module)
REFERENCES Enseignant (id_module)
ON DELETE NO ACTION
ON UPDATE NO ACTION;

ALTER TABLE Enseignant ADD CONSTRAINT module_enseignant_fk
FOREIGN KEY (id_module)
REFERENCES Enseignant (id_module)
ON DELETE NO ACTION
ON UPDATE NO ACTION;

ALTER TABLE Realisation ADD CONSTRAINT enseignant_realisation_fk
FOREIGN KEY (id_enseignant)
REFERENCES Enseignant (id_enseignant)
ON DELETE NO ACTION
```

```
ON UPDATE NO ACTION;

ALTER TABLE Module ADD CONSTRAINT enseignant_module_fk
FOREIGN KEY (id_enseignant)
REFERENCES Enseignant (id_enseignant)
ON DELETE NO ACTION
ON UPDATE NO ACTION;

ALTER TABLE Passation ADD CONSTRAINT etudiant_passation_fk
FOREIGN KEY (id_etudiant)
REFERENCES Etudiant (id_etudiant)
ON DELETE NO ACTION
ON UPDATE NO ACTION;

ALTER TABLE Evaluation ADD CONSTRAINT evaluation_evaluation_fk
FOREIGN KEY (Parent_id_evaluation)
REFERENCES Evaluation (id_evaluation)
ON DELETE NO ACTION
ON UPDATE NO ACTION;

ALTER TABLE Realisation ADD CONSTRAINT evaluation_realisation_fk
FOREIGN KEY (nom_evaluation)
REFERENCES Evaluation (nom)
ON DELETE NO ACTION
ON UPDATE NO ACTION;

ALTER TABLE Passation ADD CONSTRAINT evaluation_passation_fk
FOREIGN KEY (nom_evaluation)
REFERENCES Evaluation (nom)
ON DELETE NO ACTION
ON UPDATE NO ACTION;
```

2.2.5 Discussion sur les différences entre les scripts produits manuellement et automatiquement :

La clé primaire :

Manuellement : En ajoutant PRIMARY KEY comme spécificité à l'attribut <nom_de_attribut> <type> PRIMARY KEY

(exemple : id_enseignant INTEGER PRIMARY KEY)

Automatiquement : En ajouter une ligne après la création de tous les attributs du tableau en écrivant PRIMARY KEY (<cle_primaire_1, cle_primaire_2 ..>)

(exemple : PRIMARY KEY (id_enseignant),même quand il y a une seule clé primaire,

Or manuellement on l'applique juste quand il existe plusieurs clés primaires.

Les clés étrangères :

Manuellement : On l'ajoute lors de la création du tableau en précisant le tableau de référence (REFERENCES <nom_du_tableau_de_reference>) et ajoutant soit ON DELETE SET NULL où ON DELETE SET CASCADE.

Automatiquement : Est ajoutée à la fin du script en modifiant le tableau (ALTER TABLE <nom_du_tableau> ADD CONSTRAINT <tableau.de.reference_tableau_fk>

FOREIGN KEY <la_cle_dans_le_nouveau_tableau>

REFERENCES <tableau_de_reference> <la_cle_dans_le_tableau_de_reference>

ON DELETE NO ACTION

ON UPDATE NO ACTION)

Le type de l'attribut entier (par exemple l'identifiant) :

Manuellement : le type est INTEGER.

Automatiquement : le type est INT.

Le type NOT NULL :

Manuellement : En l'ajoutant manuellement à chaque attribut.

Automatiquement : le type NOT NULL s'ajoute automatiquement à chaque attribut.

2.3.1.

Description des différentes étapes du peuplement des tables :

On a deux manières de faire :

1- En séparant le fichier csv en plusieurs fichiers selon les colonnes des tables déjà crées et copier les données du fichier dans la table :

La création des fichiers csv pour chaque table sera en respectant la chronologie des colonnes du tableau déjà créé sur notre base de données et en les séparant par un point-virgule, et ça serait appliqué sur toutes les tables.

Exemple (pour la table enseignant) :

145;Heron;Anne

161;Coignard;Charles

…

=> (id_enseignant ;nom_enseignant ;prenom_enseignant)

La table Enseignant :

```
COPY enseignant (id_enseignant,nom_enseignant,prenom_enseignant) FROM 'enseignant.csv' DELIMITER';' CSV ;
```

Résultat généré par postgreSQL de la table Enseignant:

```
SELECT * FROM Enseignant ;
id_enseignant | nom_enseignant | prenom_enseignant
--------------+----------------+------------------
          153 | Caplot         | Prosper
          148 | Sabatier       | Michele
          158 | Larmonier      | Frederic
          159 | Carrere        | Mohamed
          156 | Lusseau        | Patrice
          146 | Denis          | Olivier
          147 | Selosse        | Frederic
          152 | Grosperrin     | Yvon
          154 | Rotsztein      | Nicolas
          150 | Gervais        | Vincent
          160 | Montier        | Joao
          151 | Leroy          | Vincent
          149 | Martos         | Marcelle
```

```
          157 | Donizeau       | Leon
          155 | Grandin        | Antoine
          144 | Helin          | Mohamed
          161 | Coignard       | Charles
          145 | Heron          | Anne
(18 rows)
```

La table Module :

- Peupler la table Module :

```
COPY module (id_module,id_enseignant,code,ue,intitule_module) FROM 'module.csv' DELIMITER';' CSV ;
```

Résultat généré par postgreSQL de la table Module :

```
SELECT * FROM Module ;
id_module | id_enseignant |                          intitule_module                          | code |
ue
-----------+---------------+-------------------------------------------------------------------+------
+------
        15 |           158 | Introduction aux systèmes d’exploitation et à leur fonctionnement | R104 |
UE12
        10 |           153 | Installation d'un poste pour le développement                     | S103 |
UE13
        17 |           160 | Économie durable et numérique                                     | R109 |
UE12
         6 |           149 | Bases de la communication                                         | R111 |
UE12
         4 |           147 | Comparaison d'approches algorithmiques                            | S102 |
UE13
         2 |           145 | Initiation au développement                                       | R101 |
UE12
        16 |           159 | Introduction aux bases de données et SQL                          | R105 |
UE12
         5 |           148 | Introduction à l'architecture des ordinateurs                     | R103 |
UE12
        12 |           155 | Projet professionnel et personnel                                 | R112 |
UE12
         1 |           144 | Création d'une base de données                                    | S104 |
UE13
        18 |           161 | Outils mathématiques fondamentaux                                 | R107 |
UE12
        13 |           156 | Recueil de besoins                                                | S105 |
UE13
         8 |           151 | Gestion de projet & des organisations                             | R108 |
UE12
```

```
       14 |          157 | Mathématiques discrètes                                    | R106 |
UE12
        9 |          152 | Découverte de l'environnement économique et écologique     | S106 |
UE13
       11 |          154 | Implémentation d'un besoin client                          | S101 |
UE13
        3 |          146 | Développement d'interfaces web                             | R102 |
UE12
        7 |          150 | Anglais technique                                          | R110 |
UE12
(18 rows)
```

La table Evaluation :

- ***Peupler la table Evaluation :***

1- Tout d'abord en changeant le paramètre de l'heure automatique par postgresql :

SET datestyle to SQL,DMY;

Automatiquement c'est réglé à MDY (month , day ,year)

Sur le fichier le format est de DMY (day , month , year)

Pour voir les paramètres enregistrés à propos du style de la date :

SHOW datestyle ;

Résultat :

```
 DateStyle
-----------
 SQL, DMY
```

2- Copier les données contenues dans le fichier 'evaluation.csv' qui a été rempli en suivant la chronologies des colonnes du tableau :

```
COPY evaluation (nom_evaluation,id_module,date_evaluation) FROM
'evaluation.csv' DELIMITER';' CSV ;
```

Résultat généré par postgreSQL de la table Evaluation :

```
                nom_evaluation                | id_module | date_evaluation
----------------------------------------------+-----------+----------------
 Final exam                                   |         7 | 2022-01-20
 CV (andromeda Cygnus)                        |         6 | 2022-01-22
 Devoir écrit Draco                           |         6 | 2022-01-11
```

Revue de presse (Draco)	6	2022-01-11
Note globale	9	2022-01-16
évaluation Installation	10	2021-12-13
Controle moyen 1	2	2021-09-29
Minicontrole 4	2	2021-10-12
Participation	14	2021-09-06
Contrôle 1	15	2021-10-27
Contrôle 2	3	2022-01-20
Controle moyen 2 - Exercice 3	11	2021-10-27
Conception d'un article (Draco)	6	2022-01-11
Controle moyen 2 - Bonus minicontrole 4	2	2021-10-27
Evaluation code	4	2022-01-17
Note du contrôle continu	17	2021-11-01
retour matériel	10	2021-12-13
Contrôle long	16	2022-01-05
Contrôle loqique mathématique	14	2021-12-08
Oral presentation	7	2022-01-20
QCM	5	2022-01-21
Contrôle court	3	2021-12-15
Oral Tous : Prise de parole en public, Discours ou Implication ou Captation	6	2021-11-23
Contrôle 4	2	2022-01-21
compte-rendu	10	2021-12-13
Exercice du contrôle 4	4	2022-01-21
Contrôle 2	18	2022-01-12
Mini-test Prédicats	14	2021-11-22
Examen final	17	2021-11-10
Evaluation phase 2	1	2021-09-06
Contrôle 1	18	2021-11-24
Projet	11	2021-11-10
Examen final	8	2022-01-20
Tests unitaires - Controle moyen 1	11	2021-09-29
Mini-test 2 Relations	14	2021-10-04
Contrôle Ensembles et relations	14	2021-10-20
contrôle court	2	2021-12-08

La table Realisation :

- Peupler la table Realisation :

```
COPY realisation (id_enseignant,nom_evaluation) FROM 'realisation.csv'
DELIMITER';' CSV ;
```

Résultat généré par postgreSQL de la table Realisation:

(seulement quelques lignes)

```
SELECT * FROM Réalisation ;

id_enseignant |                              nom_evaluation
--------------+--------------------------------------------------------------------------------
          144 | Evaluation phase 2
          145 | contrôle court
          158 | Contrôle 1
          157 | Contrôle loqique mathématique
          157 | Contrôle Ensembles et relations
          149 | Ecrit Phoenix/Pegasus Recueil du besoin
          144 | Evaluation phase 1
          149 | Conception d'un article (Draco)
          149 | Revue de presse (Draco)
          145 | Minicontrole 1
          157 | Mini-test Prédicats
          150 | Final exam
          144 | Evaluation phase 3
          153 | QCM
          145 | Contrôle 4
```

La table Etudiant :

- <u>Peupler la table Etudiant :</u>

```
COPY etudiant (id_etudiant,nom_etudiant,prenom_etudiant) FROM
'etudiant.csv' DELIMITER';' CSV ;
```

Résultat généré par postgreSQL de la table Etudiant:

(seulement quelques résultats)

```
SELECT *FROM Etudiant ;

id_etudiant |  nom_etudiant   | prenom_etudiant
------------+-----------------+-----------------
        110 | Frejafon        | Antonio
        118 | Pietri          | Georges
         44 | Ravailler       | Felix
```

89	Boumahdi	Paul
40	Bolot	Eve
108	Gesret	Tiphanie
107	Parra	Christine
43	Bagur	Gerard
98	Etcheverry	Therese
130	Miller	Antonio
30	Desjardin	Yvonne
131	Delayen	Cindy
133	Dufour	Paul
21	Sow	Nadine
15	Belkadi	Thierry

…

(137 rows)

La table Passation :

- Peupler la table Passation :

```
COPY passation (id_etudiant ,nom_evaluation ,note) FROM 'passation.csv'
DELIMITER';' CSV ;
```

Résultat généré par postgreSQL de la table Passation:

```
SELECT * FROM Passation ;
```

id_etudiant	nom_evaluation	note
34	Mini-test Prédicats	10
134	Contrôle 1	7
15	Mini-test 2 Relations	9.75
3	Minicontrole 4 - Exercice 2	3.5
25	Evaluation phase 2	8.3
52	Examen final	17.5
37	Contrôle loqique mathématique	11.5
62	Contrôle 2	13.67
56	Evaluation code	7.3
34	Controle moyen 2 - Bonus minicontrole 4	3
126	Mini-test 3 Logique	8.5
5	Note de contrôle continue	16
34	Contrôle court	14
10	Note globale	16

18	Mini-test 1 Ensembles	10
87	Contrôle court	16
12	Contrôle Ensembles et relations	17.826086
19	Minicontrole 3 - Exercice 3 + bonus Mini4	1.5
114	Exercice du contrôle 4	1.5
19	Contrôle 1	11
3	Evaluation phase 1	5

2- La deuxième manière de faire est :

De créer une table élémentaire contenant toutes les données contenues dans le fichier csv

C’est une tableau complètement exploitable

Mais pour copier les données sur chaque table faudrait copier de la table élémentaire les colonnes de la table qu’on souhaiterait compléter.

2.3.2.Présentation commentée de deux requêtes intéressantes sur la base de données :

Première requête :

Afficher les étudiants(nom et prénom) et le nombre d'évaluations passées en Maths discrètes et la note minimale et la note maximale obtenues pour chaque étudiant ainsi que la moyenne du module , les résultats seront affichés de la meilleure moyenne à la mauvaise.

Solution de la requête :

```
hatem=>
SELECT nom_etudiant ,prenom_etudiant ,
COUNT(nom_evaluation) AS nb_evaluation_passees ,
MIN(note) , MAX(note) , AVG(note) AS moyenne
FROM Etudiant INNER JOIN Passation USING(id_etudiant)
INNER JOIN Evaluation USING (nom_evaluation)
INNER JOIN Module USING (id_module)
WHERE intitule_module= 'Mathématiques discrètes'
GROUP BY id_etudiant , nom_etudiant , prenom_etudiant
ORDER BY moyenne DESC ;
```

Résultat de la requête :

```
  nom_etudiant   | prenom_etudiant | nb_evaluation_passees | min  |    max    |      moyenne
-----------------+-----------------+-----------------------+------+-----------+------------
--------
 Chevret         | Jacques         |                     5 |   10 |      19.5 | 15.530434799194335
 Frejafon        | Antonio         |                     7 |  9.3 |      22.1 | 15.285714421953474
 Bagur           | Gerard          |                     7 |   10 |      21.5 | 15.285714285714286
 Lourenco        | Roland          |                     7 |    7 |        22 | 14.795031138828822
 Taris           | Juana           |                     7 |    9 |        22 |  14.77950314113072
 Moreau          | Dragan          |                     7 |    8 |      20.6 | 14.507143020629883
 Roussel         | Aurelie         |                     7 |  9.5 |      21.5 |  14.45714282989502
 Morain          | Adrien          |                     7 |    7 |        20 | 14.385714394705635
 Freard          | Daniel          |                     7 |    7 |        23 | 14.298136574881417
 Badji           | Elodie          |                     7 |  8.5 |      20.5 | 14.171428680419922
 Guidi           | Nicolas         |                     7 |  8.5 |        20 |                 14
```

Gesret	Tiphanie	7	8	23	13.726708003452845
Guiard	Alphonse	7	8.5	18.5	13.714285714285714
Ravailler	Felix	7	6	20.2	13.707142966134208
Auger	Philippe	7	7.5	21	13.689440863473076
Chaumaz	Herve	7	7	19.5	13.52267074584961
Delmotte	Coralie	7	8.5	18	13.52173914228167
Guyot	Thomas	6	7	19	13.424999872843424
Guilloton	Benoit	7	7	19	13.385714394705635
Bacquey	Luc	7	6.5	20.5	13.296583720615931
Massone	Henriette	7	7	20.2	13.285714285714286
Brunet	Laurent	7	6.5	19.1	13.26428576878139
Boumahdi	Paul	6	8.75	20	13.191666762034098

…

Desjardin	Yvonne	2	1	5.5	3.25
Riviere	Emmanuel	1	1.5	1.5	1.5
Legrand	Damien	1	1	1	1

(137 rows)

Deuxième requête :

Afficher le module, son responsable(nom et prénom de l'enseignant) , la meilleure et la mauvaise note obtenus et ainsi la moyenne du module de toute la promo et afficher le résultat en ordre croissant selon la moyenne.

La moyenne est affichée en arrondi au nombre entier le plus proche.

Solution de la requête :

hatem=>

```
SELECT intitule_module , nom_enseignant, prenom_enseignant
MIN(note) AS min_promo , MAX(note) AS max_promo ,
CAST(AVG(note) as integer) AS moyenne_promo
FROM Enseignant INNER JOIN Module USING(id_enseignant)
INNER JOIN Réalisation USING(id_enseignant)
INNER JOIN Passation USING(nom_evaluation)
GROUP BY intitule_module , id_enseignant , nom_enseignant , prenom_enseignant
ORDER BY moyenne_promo ASC ;
```

Résultat de la requête :

intitule_module	nom_enseignant	prenom_enseignant	min_promo	max_promo	moyenne_promo
Comparaison d'approches algorithmiques	Selosse	Frederic	0.5	8	4
Implémentation d'un besoin client	Rotsztein	Nicolas	0.25	19.6	6
Initiation au développement	Heron	Anne	0.25	20	6
Création d'une base de données	Helin	Mohamed	0.5	16.5	7
Introduction à l'architecture des ordinateurs	Sabatier	Michele	0.25	16.5	9
Outils mathématiques fondamentaux	Coignard	Charles	1	20	10
Introduction aux systèmes d’exploitation et à leur fonctionnement	Larmonier	Frederic	0.75	20	11
Installation d'un poste pour le développement	Caplot	Prosper	1	20	11
Mathématiques discrètes	Donizeau	Leon	1	23	11
Anglais technique	Gervais	Vincent	1	18	11
Introduction aux bases de données et SQL	Carrere	Mohamed	0.5	19	11
Économie durable et numérique	Montier	Joao	2.5	18	12
Recueil de besoins	Lusseau	Patrice	1.8	20	12
Bases de la communication	Martos	Marcelle	2	18	13
Gestion de projet & des organisations	Leroy	Vincent	5	18	13
Développement d'interfaces web	Denis	Olivier	3	23.75	13
Projet professionnel et personnel	Grandin	Antoine	7	18	14
Découverte de l'environnement économique et écologique	Grosperrin	Yvon	3.5	18.5	14

(18 rows)