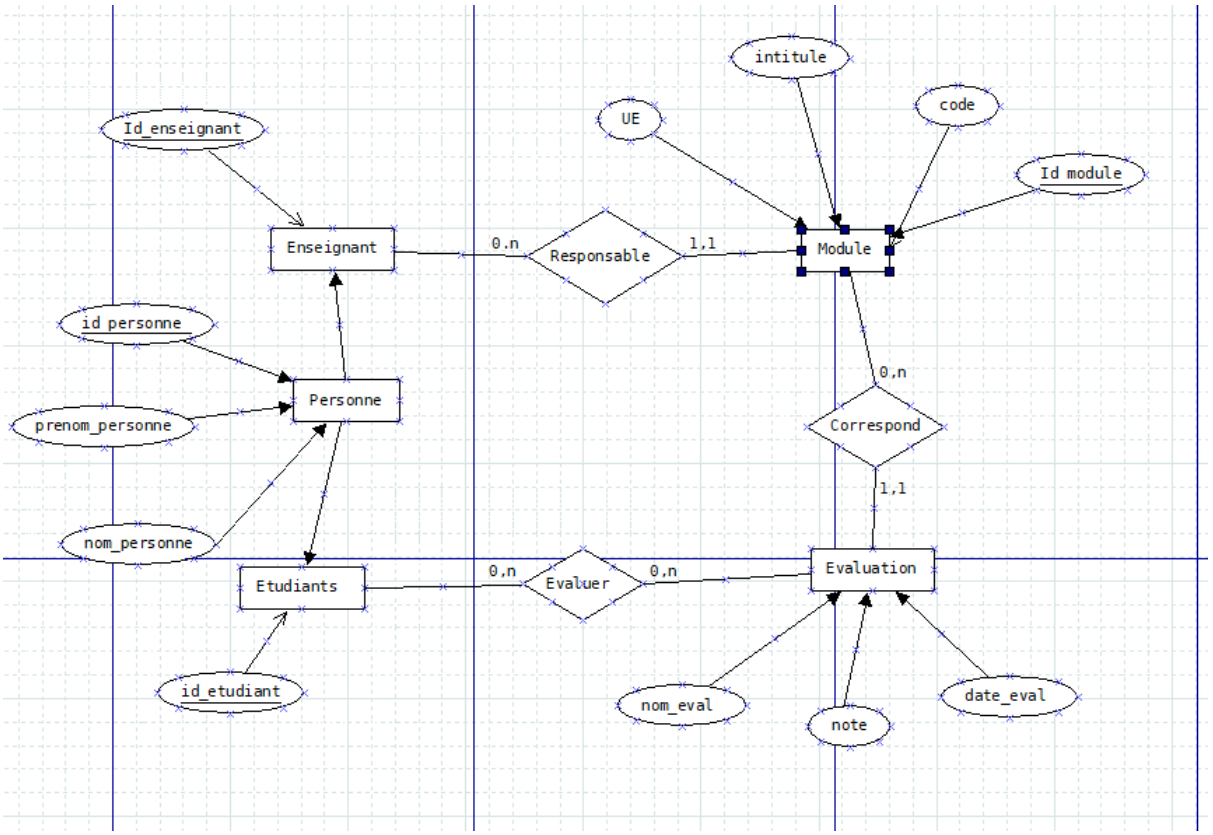
RAPPORT SAE S104

Oki Samy Group Tlaloc

Sommaire

1] Modélisation et script de création « sans AGL »

1) Réprésation du model entitées-association



2) Schéma relationnel

persone(id_personne, nom_personne, prenom_personne,id_etudiant, id_personne)
etudiant(id_etudiant)
enseignant(id_enseignant)
module(id_module, UE, intitule,code, id_enseignant)
evaluation(nom_eval, date_eval, id_module)
est_evaluer(id_etudiant)

3) Script SQL de création des tables

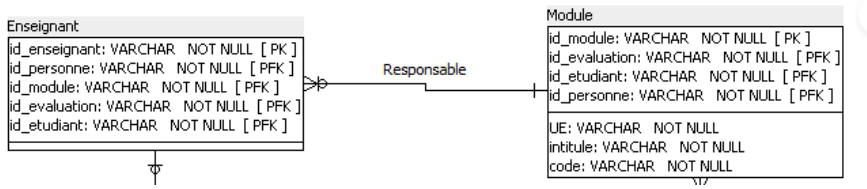
DROP TABLE enseignant;

```
DROP TABLE personne;
DROP TABLE etudiant;
DROP TABLE evaluation;
DROP TABLE module;

CREATE TABLE personne ( id_personne INTEGER PRIMARY KEY,
nom_personne VARCHAR NOT NULL, prénom_personne VARCHAR NOT
NULL);

CREATE TABLE enseignant ( id_enseignant INTEGER REFERENCES
personne (id_personne));

CREATE TABLE etudiant (id_etudiant INTEGER REFERENCES personne
(id_personne));

CREATE TABLE evaluation (nom_evaluation VARCHAR NOT NULL,
date_evaluation DATE NOT NULL);

CREATE TABLE module (id_module INTEGER PRIMARY KEY,
unite_enseignement VARCHAR NOT NULL, intitule_evaluation VARCHAR
NOT NULL);
```

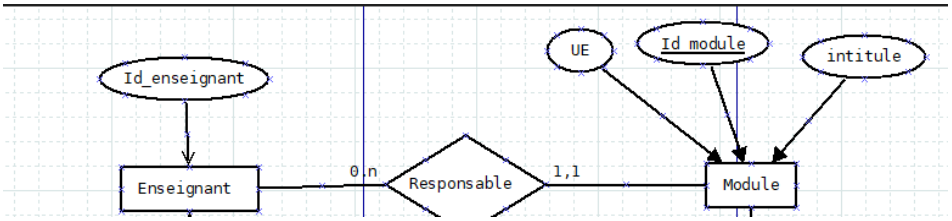
2] Modélisation et script de création « avec AGL »

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

Association fonctionnelle sur AGL :



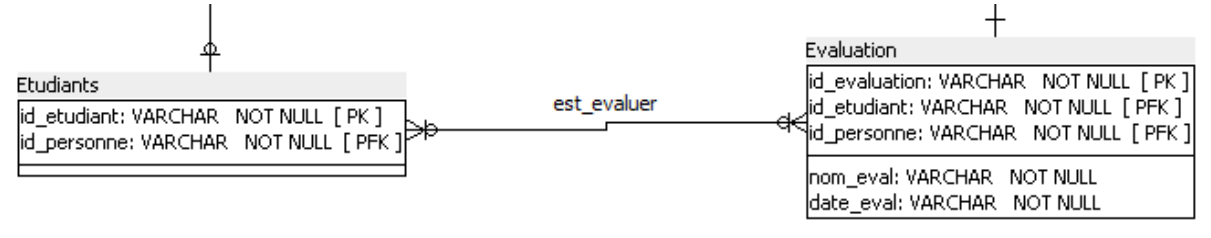
Association fonctionnelle du cour (faite sur DIA):



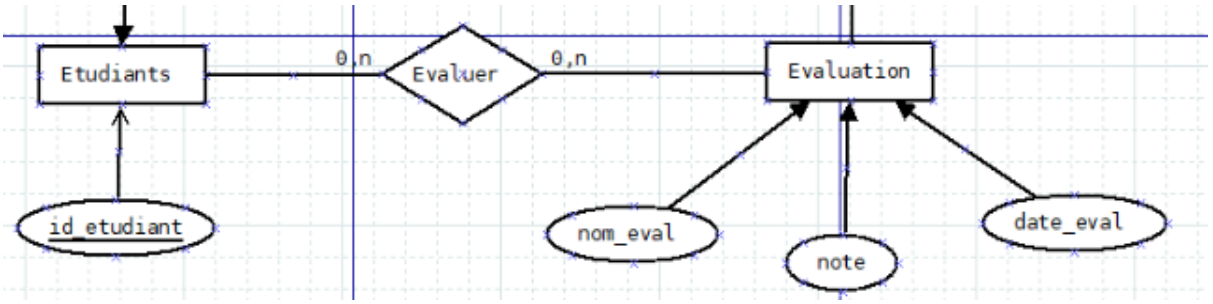
Une association fonctionnelle est une association où les cardinalité sont égal à 0,n et 1,1.
Premiérement on peux constater que la représentation DIA est plus schématiser et compréhensif que sur AGL. En effet l'association fonctionnelle faite sur DIA comporte des éléments plus lisible telle que la cardinalité, les attributs et les clés primaire de chaque table, contrairement sur AGL où les éléments cité sont afficher de façon plus laborieux . Mais AGL posséde un avantage car contrairement sur DIA , lors d'une assocition les clés étrangéres sont répertorier dans les tables, permettant de faire les liens entre les entités. De plus sur AGL pour chaque attribut le type d'attribut et la condition et répertorier contrairement sur l'association précédente.

2) Illustration comparatives cours/AGL commentée d'une assocition maillée

Association maillée sur AGL :

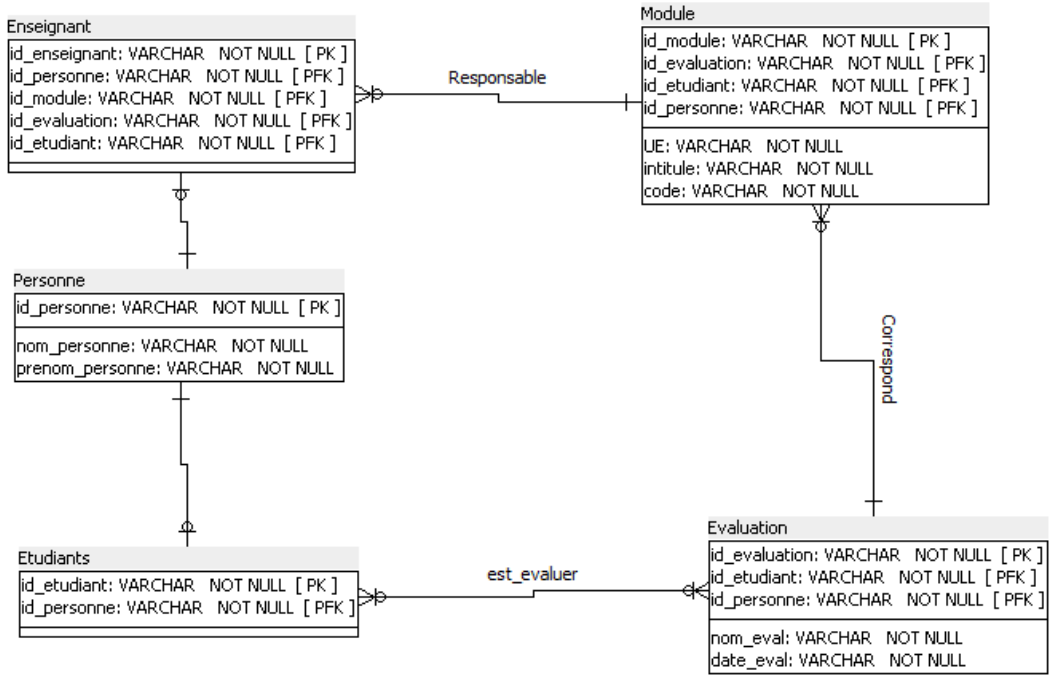


Association maillée du cour (faite sur DIA):



Une association maillée est une association où les cardinalité sont égal à 0,n.
Comme pour l'association fonctionnelle AGL est un logiciel où les informations sont assez compliqué à assimilé contrairement à l'association faite sur DIA mais ces dernières reste très complète. En effet nous avons toujours pour chaque attribut le type d'attribut et la condition et répertorier contrairement sur l'association précédente et lors d'une assocition les clés étrangéres sont répertorier dans les tables. Mais sur DIA les cardinalités sont plus lisible que sur AGL.

3) Modèle entités association réaliser avec l'AGL



4) Script SQL de création des tables généré automatique par l'AGL

```
CREATE TABLE public.Personne (
        id_personne VARCHAR NOT NULL,
        nom_personne VARCHAR NOT NULL,
        prenom_personne VARCHAR NOT NULL,
        CONSTRAINT id_personne PRIMARY KEY (id_personne)
);


CREATE TABLE public.Etudiants (
        id_etudiant VARCHAR NOT NULL,
        id_personne VARCHAR NOT NULL,
        CONSTRAINT id_etudiant PRIMARY KEY (id_etudiant, id_personne)
);


CREATE TABLE public.Evaluation (
        id_evaluation VARCHAR NOT NULL,
        id_etudiant VARCHAR NOT NULL,
        id_personne VARCHAR NOT NULL,
        nom_eval VARCHAR NOT NULL,
        date_eval VARCHAR NOT NULL,
```

```
	CONSTRAINT id_evaluation_ PRIMARY KEY (id_evaluation, id_etudiant,
id_personne)
);


CREATE TABLE public.Module (
	id_module VARCHAR NOT NULL,
	id_evaluation VARCHAR NOT NULL,
	id_etudiant VARCHAR NOT NULL,
	id_personne VARCHAR NOT NULL,
	UE VARCHAR NOT NULL,
	intitule VARCHAR NOT NULL,
	code VARCHAR NOT NULL,
	CONSTRAINT id_module PRIMARY KEY (id_module, id_evaluation,
id_etudiant, id_personne)
);


CREATE TABLE public.Enseignant (
	id_enseignant VARCHAR NOT NULL,
	id_personne VARCHAR NOT NULL,
	id_module VARCHAR NOT NULL,
	id_evaluation VARCHAR NOT NULL,
	id_etudiant VARCHAR NOT NULL,
	CONSTRAINT id_enseignant_ PRIMARY KEY (id_enseignant,
id_personne, id_module, id_evaluation, id_etudiant)
);


ALTER TABLE public.Enseignant ADD CONSTRAINT personne_enseignant_fk
FOREIGN KEY (id_personne)
REFERENCES public.Personne (id_personne)
ON DELETE NO ACTION
ON UPDATE NO ACTION
NOT DEFERRABLE;

ALTER TABLE public.Etudiants ADD CONSTRAINT personne_etudiants_fk
FOREIGN KEY (id_personne)
REFERENCES public.Personne (id_personne)
ON DELETE NO ACTION
ON UPDATE NO ACTION
NOT DEFERRABLE;

ALTER TABLE public.Evaluation ADD CONSTRAINT evaluer
```

```
FOREIGN KEY (id_personne, id_etudiant)
REFERENCES public.id_etudiant (id_personne, id_etudiant)
ON DELETE NO ACTION
ON UPDATE NO ACTION
NOT DEFERRABLE;

ALTER TABLE public.Module ADD CONSTRAINT correspond
FOREIGN KEY (id_etudiant, id_personne, id_evaluation)
REFERENCES public.id_evaluation (id_etudiant, id_personne, id_evaluation)
ON DELETE NO ACTION
ON UPDATE NO ACTION
NOT DEFERRABLE;

ALTER TABLE public.Enseignant ADD CONSTRAINT responsable
FOREIGN KEY (id_module, id_personne, id_evaluation, id_etudiant)
REFERENCES public.Moduke (id_module, id_personne, id_evaluation, id_etudiant)
ON DELETE NO ACTION
ON UPDATE NO ACTION
NOT DEFERRABLE;
```

5) Discussion sur les différences entre les scripts ^roduits manuellement et automatique

Tout abord nous pouvons constater que le script générer par l'AGL est beaucoup plus détailler que celui fait manuellement. On peut constater que dans le script fait manuellement ne comporte pas le code de création des types associations. On peut noter que le script générer par l'AGL est plus sûre et plus fiable car les chances de se tromper sur le script générer manuellement sont plus forte.

3 / Peuplement des tables et requêtes

1) Description commentée des différentes étapes de votre script de peuplement.

J’ai réaliser ma base de donnée sur le logiciel postgress, pour créer mes tables j’ai lancer l’option “create script” où j’ai rentrer le script pour créer mes tables ,ensuite pour extraire les données du fichier csv je réalise la commande suivante

```
COPY Temp
FROM 'C:\Users\Samy\Desktop\SAE104_data.csv\data.csv'
DELIMITER ';'
CSV Header;
```

Cette commande permet de copier le fichier csv afin de remplir plus facilement chaque table de ma base de donnée.
Ensuite il me reste plus cas utiliser la table Temp pour remplir les tables à l’aide des commandes suivantes :

```
INSERT INTO enseignant (SELECT DISTINCT
id_enseignant,nom_enseignant,prenom_enseignant FROM Temp);

INSERT INTO etudiant (SELECT DISTINCT
id_etudiant,nom_etudiant,prenom_etudiant FROM Temp);

INSERT INTO evaluation (SELECT DISTINCT
nom_evaluation,date_evaluation,note FROM Temp);

INSERT INTO module (SELECT DISTINCT
id_module,intitule_module,ue,code FROM Temp);
DROP TABLE Temp;
```

Désormais les tables sont remplis.

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

Tout abord la requête SELECT * FROM Etudiant permet d’afficher la tables Etudiant de la base de données afin de constater s’il n’y a pas de faute dans cette table.

Et pour finir la requête SELECT nom_evaluation, date_evaluation FROM evaluation, module WHERE ue='UE13' qui permet de connaitre le nom et la date des évaluations de l’unité d’enseignement UE13