

ARUMUGADAS Ravichandran  
Tlaloc  
BUT Informatique

Base de Données et langage SQL  
Situation d'Apprentissage et d'Évaluation (SAE 104)  
**Création d'une base de données : Notations**

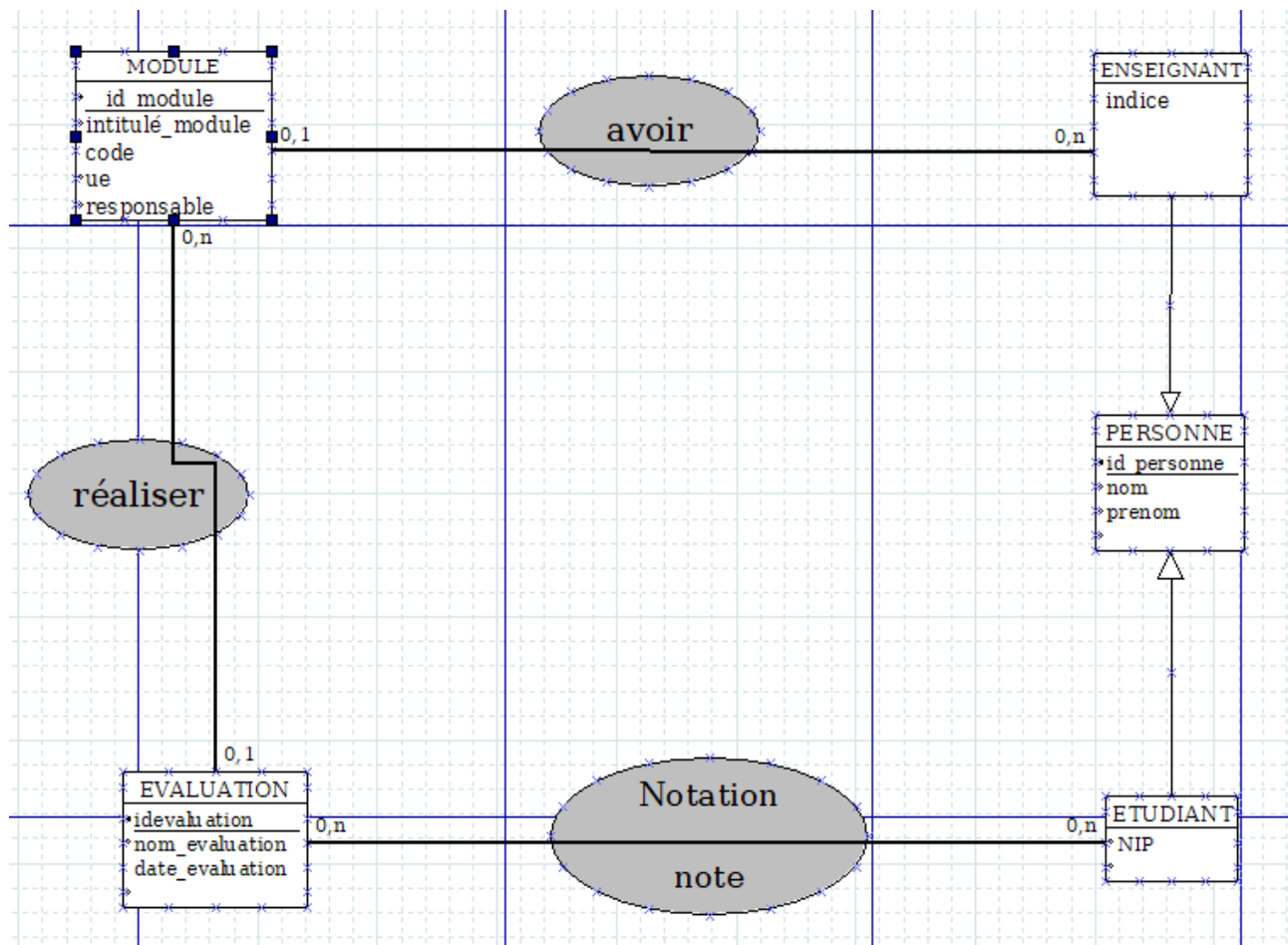
Tout d'abord, cette SAE a pour objectif de concevoir et alimenter une base de données afin de stocker toutes les notes des évaluations des étudiants d'une promotion.

## **SOMMAIRE**

- I - Modélisation et script de création « sans AGL »
- II - Modélisation et script de création «avec AGL »
- III - Peuplement des tables et requêtes

## - I - Modélisation et script de création « sans AGL »

Dans un premier temps, nous allons réaliser un modèle entités-associations de la base de données en respectant la syntaxe du cours. Voici le modèle entités-associations :



Un schéma de relation est composé par l'ensemble des schémas de relation en précisant le nom de la relation, la liste des attributs avec leur domaine. Voici le schéma relationnel de ce modèle :

```
MODULE(id_module, intitulé_module, code, ue, responsable, id_enseignant)
ENSEIGNANT(id_enseignant, indice)
PERSONNE(id_personne, nom, prenom)
ETUDIANT(id_etudiant, NIP)
NOTATION(idevaluation, id_etudiant, note)
EVALUATION(idevaluation, nom_evaluation, date_evaluation, id_module)
```

Voici le script sql afin de produire ces tables :

```
CREATE TABLE personne
(
id_personne INTEGER PRIMARY KEY,
nom VARCHAR NOT NULL,
prenom VARCHAR NOT NULL
);
```

```
CREATE TABLE enseignant
(
id_enseignant INTEGER PRIMARY KEY REFERENCES personne (id_personne)
);
```

```
CREATE TABLE etudiant
(
id_etudiant INTEGER PRIMARY KEY REFERENCES personne (id_personne)
nip INTEGER NOT NULL,
);
```

```
CREATE TABLE module
(
id_module INTEGER PRIMARY KEY,
id_enseignant INTEGER REFERENCES personne (id_personne),
intitule VARCHAR NOT NULL,
code VARCHAR NOT NULL,
ue VARCHAR NOT NULL
);
```

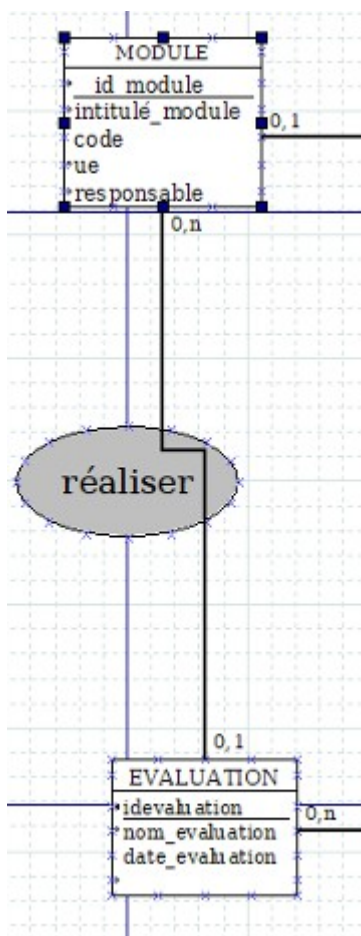
```
CREATE TABLE notation
(
id_etudiant INTEGER REFERENCES personne (id_personne),
id_evaluation INTEGER REFERENCES evaluation (id_evaluation),
note INTEGER NOT NULL,
PRIMARY KEY (id_etudiant, id_evaluation)
);
```

```
CREATE TABLE evaluation
(
idevaluation INTEGER PRIMARY KEY,
id_module INTEGER REFERENCES module (id_module),
nom_evaluation VARCHAR NOT NULL,
date DATE NOT NULL,
);
```

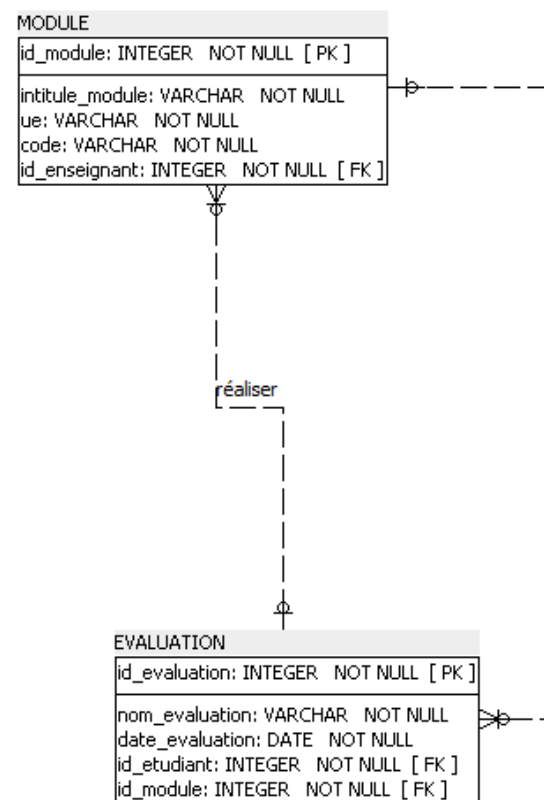
## - II - Modélisation et script de création «avec AGL »

Tout d'abord, nous allons faire des comparaisons entre les modèles entités-associations respectant la syntaxe du cours et les modèles entités-associations avec l'AGL SQL Power Architect.

Dans un premier temps, voici une illustration comparatives cours/ AGL commentée d'une association fonctionnel :



« Sans AGL »



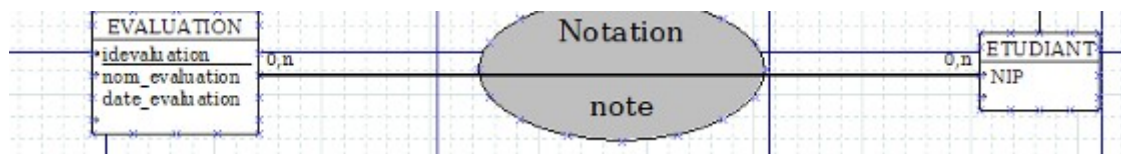
« Avec AGL »

Une association fonctionnelle se définit par le déplacement de la clé primaire côté n vers une clé étrangère côté 1.

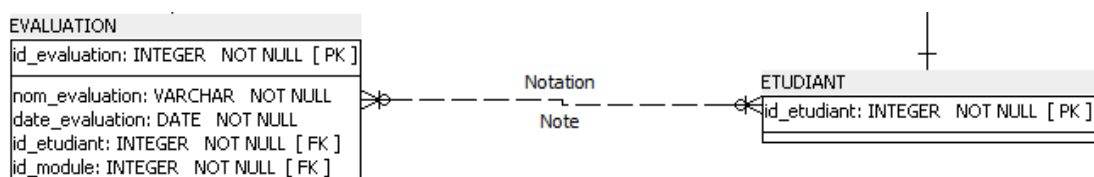
L'identifiant de ce type-association est constitué de l'identifiant du participant au niveau du côté de la cardinalité max de 1.

Nous remarquons qu'il y a plusieurs différences entre ces illustrations. Tout d'abord, nous avons vu en cours que les modèles entités-associations se font avec des liaisons (flèches) en représentant les cardinalités alors qu'avec l'AGL, les cardinalités ne sont pas représentées. D'ailleurs, avec l'AGL, les liaisons sont représentées avec des cercles et des traits. En plus de cela, nous pouvons remarquer que les types-associations ne sont pas représentés comme en cours (c'est-à-dire avec des cercles). Avec l'AGL, il faut juste écrire le nom du type association. Un autre fait qui est remarquable sur l'AGL, c'est le fait que les clés étrangères sont automatiquement mises dans les tables.

Voici une illustration comparative cours/ AGL commentée d'une association maillée:



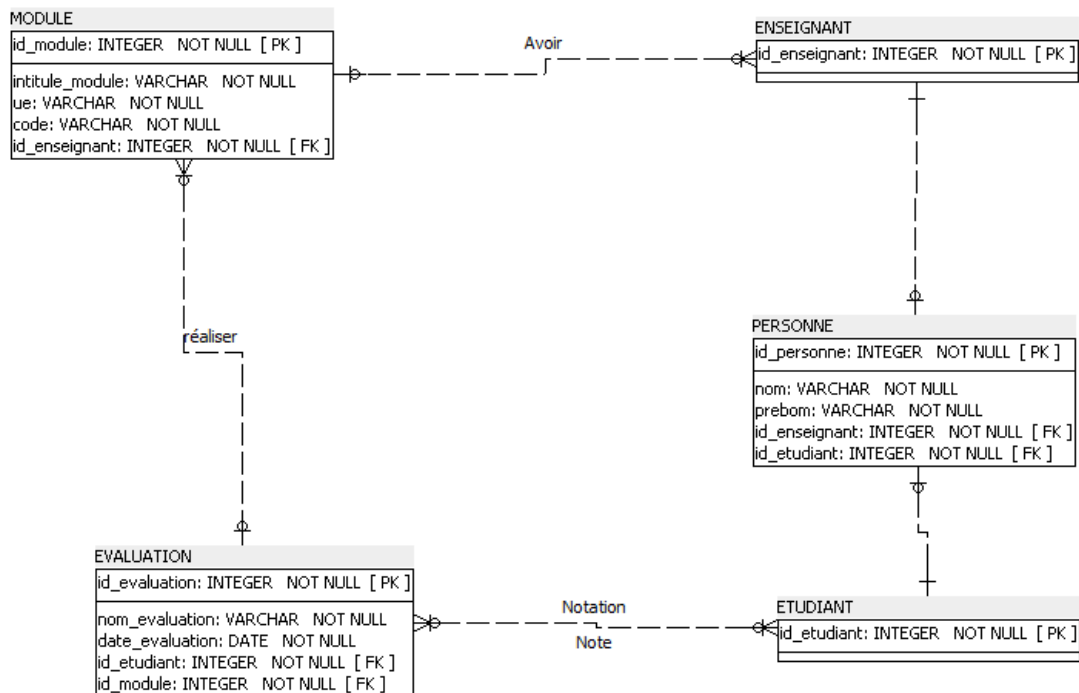
« Sans AGL »



« Avec AGL »

Il y a aussi plusieurs différences pour les associations maillées au niveau de la syntaxe du cours et l'AGL. Il y a en effet les mêmes différences que nous venons de voir précédemment. Nous savons que l'identifiant d'un type-association est composé des identifiants de ses participants. Cependant, dans l'AGL, il n'y a pas de type-association, ce qui fait que la clé étrangère se met automatiquement sur une table.

Voici un modèle entités-associations réalisé avec l'AGL :



Voici le script SQL de création des tables généré automatiquement par l'AGL SQL Power Architect :

```

CREATE TABLE ETUDIANT (
    id_etudiant INTEGER NOT NULL,
    CONSTRAINT id_etudiant PRIMARY KEY (id_etudiant)
);

CREATE TABLE ENSEIGNANT (
    id_enseignant INTEGER NOT NULL,
    CONSTRAINT idenseignant PRIMARY KEY (id_enseignant)
);

CREATE TABLE PERSONNE (
    id_personne INTEGER NOT NULL,
    nom VARCHAR NOT NULL,
    prenom VARCHAR NOT NULL,
    id_enseignant INTEGER NOT NULL,
    id_etudiant INTEGER NOT NULL,
    CONSTRAINT id_personne PRIMARY KEY (id_personne)
);

CREATE TABLE MODULE (

```

```
        id_module INTEGER NOT NULL,  
        intitule_module VARCHAR NOT NULL,  
        ue VARCHAR NOT NULL,  
        code VARCHAR NOT NULL,  
        id_enseignant INTEGER NOT NULL,  
        CONSTRAINT idmodule PRIMARY KEY (id_module)  
    );
```

```
CREATE TABLE EVALUATION (  
    id_evaluationl INTEGER NOT NULL,  
    nom_evaluation VARCHAR NOT NULL,  
    date_evaluation DATE NOT NULL,  
    id_etudiant INTEGER NOT NULL,  
    id_module INTEGER NOT NULL,  
    CONSTRAINT id_evaluation PRIMARY KEY (id_evaluationl)  
);
```

```
ALTER TABLE PERSONNE ADD CONSTRAINT ETUDIANT_PERSONNE_fk  
FOREIGN KEY (id_etudiant)  
REFERENCES ETUDIANT (id_etudiant)  
ON DELETE NO ACTION  
ON UPDATE NO ACTION  
NOT DEFERRABLE;
```

```
ALTER TABLE EVALUATION ADD CONSTRAINT ETUDIANT_EVALUATION_fk  
FOREIGN KEY (id_etudiant)  
REFERENCES ETUDIANT (id_etudiant)  
ON DELETE NO ACTION  
ON UPDATE NO ACTION  
NOT DEFERRABLE;
```

```
ALTER TABLE PERSONNE ADD CONSTRAINT ENSEIGNANT_PERSONNE_fk  
FOREIGN KEY (id_enseignant)  
REFERENCES ENSEIGNANT (id_enseignant)  
ON DELETE NO ACTION  
ON UPDATE NO ACTION  
NOT DEFERRABLE;
```

```
ALTER TABLE MODULE ADD CONSTRAINT ENSEIGNANT_MODULE_fk  
FOREIGN KEY (id_enseignant)  
REFERENCES ENSEIGNANT (id_enseignant)  
ON DELETE NO ACTION  
ON UPDATE NO ACTION  
NOT DEFERRABLE;
```

```
ALTER TABLE EVALUATION ADD CONSTRAINT MODULE_EVALUATION_fk  
FOREIGN KEY (id_module)  
REFERENCES MODULE (id_module)  
ON DELETE NO ACTION  
ON UPDATE NO ACTION  
NOT DEFERRABLE;
```



D'après ce script, nous pouvons apercevoir plusieurs différences avec les scripts produits manuellement.

En effet, nous pouvons remarquer que le script produit automatiquement est plus complet et plus long qu'un script écrit manuellement. Dans le script de l'AGL, il y a tout d'abord le script avec la commande « CREATE TABLE » et ensuite une commande « ALTER TABLE ». Cette commande « ALTER TABLE » permet de modifier une table qui existe déjà alors que dans un script écrit manuellement, nous nous consacrons essentiellement avec la commande « CREATE TABLE ». De plus, étant donné qu'il n'y a pas réellement de type-association dans l'AGL, les clés étrangères sont référencés dans les tables alors que manuellement, il y a par exemple la création d'une table unique.

Ainsi, nous pouvons alors dire que le script produit automatiquement est un avantage car le script est produit par le logiciel, nous avons juste à réaliser le modèle. Cependant, au niveau de la compréhension du script et du schéma, il y a sans doute quelques difficultés à comprendre par rapport le modèle et le script du cours.

### **- III - Peuplement des tables et requêtes**

Nous verrons maintenant les différentes étapes de mon script de peuplement.

- Dans un premier, nous allons créer une table data\_ qui a pour but de stocker toutes les données du fichier data.csv. Voici le script de cette table :

```
CREATE TABLE data_  
(  
    id_enseignant integer,  
    nom_enseignant character varying,  
    prenom_enseignant character varying,  
    id_module integer,  
    code character varying,  
    ue character varying,  
    intitule_module character varying,  
    nom_evaluation character varying,  
    date_evaluation date, note numeric,  
    id_etudiant integer,  
    nom_etudiant character varying,  
    prenom_etudiant character varying  
);
```

- Après cela, nous allons transférer toutes les données du fichier data.csv dans cette table. Et donc par exemple, tous les id, tous les prénoms de personnes seront stockés dans cette table. Voici le script qui permet de copier les données du fichier et ensuite transférer ces données dans cette table:

```
COPY data_ FROM 'C:\Program Files\PostgreSQL\15\scripts\data.csv' DELIMITER ';'
CSV Header;
```

- Maintenant, il s'agit de créer les tables.

Par exemple, nous pouvons commencer à créer la table étudiant :

```
CREATE TABLE etudiant(id_etudiant integer NOT NULL);
```

- Après cela, il faut insérer les données qui correspondent à cette table via la table data\_ qui a été créée précédemment :

```
INSERT INTO etudiant(id_etudiant) SELECT id_etudiant FROM data_;
```

- Il s'agit donc de copier toutes les colonnes de la table data\_ vers cette table étudiant en sélectionnant les informations correspondantes à cette table avec SELECT

Il faut effectuer de la même manière pour le reste des tables, ce qui nous donne :

```
CREATE TABLE enseignant(id_enseignant integer NOT NULL);
INSERT INTO enseignant(id_enseignant) SELECT id_enseignant FROM data_;
```

```
CREATE TABLE personne(id_personne integer, nom VARCHAR, prenom VARCHAR);
INSERT INTO personne(id_personne, nom, prenom) SELECT id_etudiant, nom_etudiant,
prenom_etudiant FROM data_;
```

```
INSERT INTO personne(id_personne, nom, prenom) SELECT id_enseignant,
nom_enseignant, prenom_enseignant FROM data_;
```

```
CREATE TABLE evaluation(nom_evaluation VARCHAR, date_evaluation DATE);
INSERT INTO evaluation(nom_evaluation, date_evaluation) SELECT nom_evaluation,
date_evaluation FROM data_;
```

```
CREATE TABLE module
(
  d_module integer NOT NULL,
  intitule_module character varying,
  code character varying,
  ue character varying,
  responsable character varying);
```

```
INSERT INTO module (id_module, intitule_module, code, ue, responsable) SELECT
id_module,intitule_module,code,ue,id_enseignant FROM data_;
```

- Après avoir créer tous ces tables, nous pouvons vérifier que notre base de données est bien exploitable.

Par exemple, voici deux requêtes sur cette base de donnée.



--

```
SELECT DISTINCT date_evaluation, intitule_module FROM evaluation, module WHERE
intitule_module='Initiation au développement' ORDER BY date_evaluation ;
```

	date_evaluation date	intitule_module character varying
1	2021-09-06	Initiation au développement
2	2021-09-13	Initiation au développement
3	2021-09-20	Initiation au développement
4	2021-09-29	Initiation au développement
5	2021-10-04	Initiation au développement
6	2021-10-05	Initiation au développement
7	2021-10-06	Initiation au développement
8	2021-10-12	Initiation au développement
9	2021-10-13	Initiation au développement
10	2021-10-20	Initiation au développement
11	2021-10-27	Initiation au développement
12	2021-11-01	Initiation au développement
13	2021-11-08	Initiation au développement
14	2021-11-10	Initiation au développement
15	2021-11-22	Initiation au développement
16	2021-11-23	Initiation au développement
17	2021-11-24	Initiation au développement
18	2021-12-08	Initiation au développement

Cette requête a pour but de retourner la date d'évaluation du module Initiation au développement de la date la plus ancienne à la plus récente en évitant les doublons

```
SELECT DISTINCT nom, prenom FROM personne JOIN etudiant ON  
personne.id_personne = etudiant.id_etudiant;
```

	nom character varying 	prenom character varying 
1	Maunoury	Pascal
2	Granato	Leonce
3	Bagur	Gerard
4	Caboche	Marc
5	El Alami	Lucien
6	Gerard	Lydie
7	Pietri	Georges
8	Carbonel	Nicole
9	Point	Yvon
10	Chassing	Alain
11	Grandchamp	Luc
12	Levasseur	Cyril
13	Onfroy	Ruth
14	Cagne	Georges
15	Auger	Philippe
16	Guyot	Thomas
17	Pons	Patricia
18	Nicaise	Rose
19	Chevret	Jacques

Cette requête a pour but de retourner les noms et les prénoms des étudiants en évitant les doublons (utilisation JOIN) ;

**Ainsi, voici mon rapport qui consiste a créer cette base de donnée. Nous avons donc vu différentes modélisations et pour finir la création de la base de donnée qui permet de stocker toutes les notes des évaluations des étudiants d'une promotion.**

