

ANANDASANGAR

Agash

BUT1

Tlaloc

## **RAPPORT SAE 1.04 BASE DE DONNÉES**

# **SOMMAIRE :**

## 1.Modélisation et script de création « sans AGL » :(page 3-4)

1. Modèle entités-associations respectant la syntaxe du cours.
2. Schéma relationnel.
3. Script SQL de création des tables.

## 2.Modélisation et script de création « avec AGL » :(page 5-7)

1. Illustrations comparatives cours/ AGL commentée d'une association fonctionnelle.
2. Illustrations comparatives cours/ AGL commentée d'une association maillée.
3. Modèle entités-associations réalisé avec l'AGL.
4. Script SQL de création des tables généré automatiquement par l'AGL.
5. Discussion sur les différences entre les scripts produits manuellement et automatiquement.

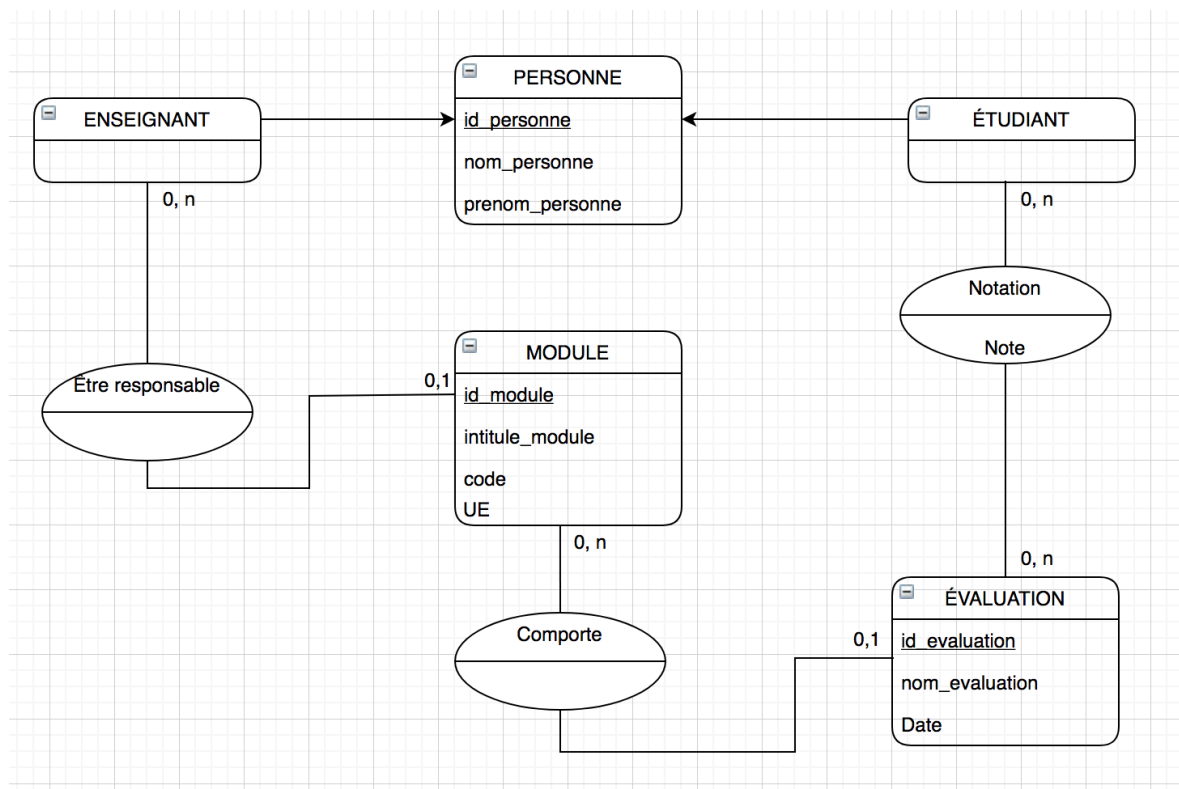
## 3.Peuplement des tables et requêtes:(page 7-9)

1. Description commentée des différentes étapes de mon script de peuplement.
2. Présentation commentée de deux requêtes intéressantes sur la base de données.

Pour cette première partie, je vais concevoir le modèle entités-associations de notre sujet puis après je vais écrire un script SQL pour la création de chaque table dans la base de donnée.

## 1. Modélisation et script de création « sans AGL »

### 1. Modèle entités-associations



### 2. Schéma relationnel du modèle entités-associations

PERSONNE(id\_personne, nom\_personne, prenom\_personne)  
 ETUDIANT (id\_etudiant) où id\_etudiant fait référence à PERSONNE  
 ENSEIGNANT (id\_enseignant) où id\_enseignant fait référence à PERSONNE  
 MODULE (id\_module, intituléModule, code, ue, id\_enseignant) où id\_enseignant fait référence à ENSEIGNANT  
 EVALUATION (id\_evaluation, nom\_evaluation, date\_evaluation, id\_module) où id\_module fait référence à MODULE  
 NOTATION (id\_etudiant, id\_evaluation, note) où id\_etudiant, id\_evaluation fait référence à ETUDIANT et EVALUATION

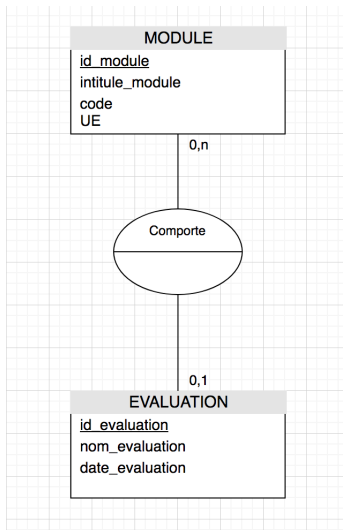
### 3. Script SQL de création des tables

```
1  DROP TABLE notation;
2  DROP TABLE etudiant;
3  DROP TABLE evaluation;
4  DROP TABLE module;
5  DROP TABLE enseignant;
6  DROP TABLE personne;
7
8  ▼ CREATE TABLE personne (
9      id_personne INT PRIMARY KEY,
10     nom_personne VARCHAR NOT NULL,
11     prenom_personne VARCHAR NOT NULL
12 );
13
14
15 ▼ CREATE TABLE etudiant (
16     id_etudiant INT REFERENCES personne (id_personne) ON DELETE CASCADE,
17     PRIMARY KEY (id_etudiant)
18 );
19
20 ▼ CREATE TABLE enseignant (
21     id_enseignant INT REFERENCES personne (id_personne) ON DELETE CASCADE,
22     PRIMARY KEY (id_enseignant)
23 );
24
25 ▼ CREATE TABLE module (
26     id_module INT PRIMARY KEY,
27     code VARCHAR NOT NULL,
28     ue VARCHAR NOT NULL,
29     intitule_module VARCHAR NOT NULL,
30     id_enseignant INT REFERENCES personne (id_personne) ON DELETE CASCADE
31 );
32
33 ▼ CREATE TABLE evaluation (
34     id_evaluation INT PRIMARY KEY,
35     nom_evaluation VARCHAR NOT NULL,
36     date_evaluation VARCHAR NOT NULL,
37     id_module INT REFERENCES module (id_module) ON DELETE CASCADE
38 );
39
40 ▼ CREATE TABLE notation (
41     id_etudiant INT REFERENCES personne (id_personne) ON DELETE CASCADE,
42     id_evaluation INT REFERENCES evaluation (id_evaluation) ON DELETE CASCADE,
43     PRIMARY KEY (id_etudiant, id_evaluation),
44     note REAL CHECK (note > 0 OR note IS NULL)
45 );
46
```

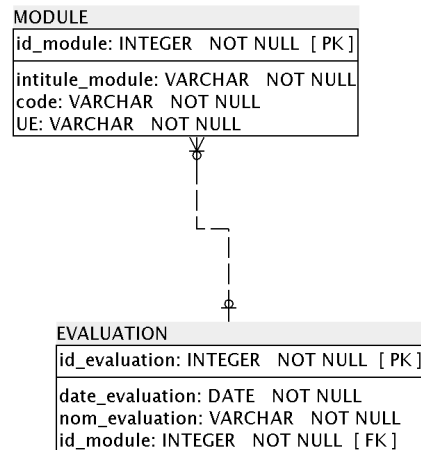
Donc, pour cette première partie je vous ai montré le modèle entités-associations et le script SQL que j'ai fait manuellement. Pour la suite, je vais comparer mon devoir avec celui généré avec SQL power architect.

## 2. Modélisation et script de création « avec AGL »

### 1. Illustration comparative cours/AGL commentée d'une association fonctionnelle



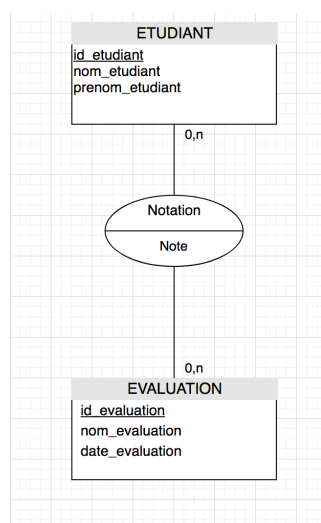
cours



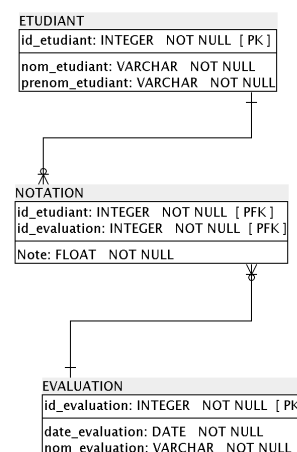
AGL

La différence entre AGL et ce que nous avons vu en cours c'est qu'il n'y a pas d'association dans l'AGL et on peut voir que les cardinalités sont représentées sous forme de symbole. Dans l'association fonctionnelle du cours, on ne voit pas la clé étrangère de la table "Module" dans évaluation.

### 2. Illustration comparative cours/AGL commentée d'une association maillée



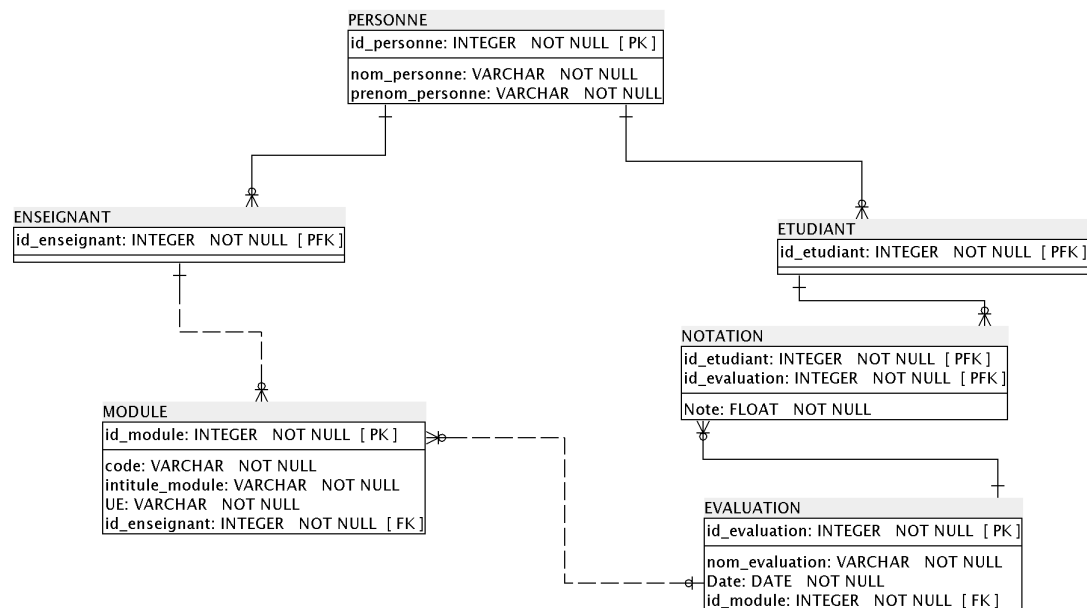
cours



AGL

Les différences entre AGL et cours c'est comme pour l'association fonctionnelle, il n'y pas d'association dans AGL, les cardinalités sont sous forme de symbole. Dans l'AGL, nous avons remplacé l'association "Notation" en table et ses clés primaires sont les clés étrangères "id\_etudiant" d'Etudiant et "id\_evaluation" d'Evaluation. Et dans cours, on ne voit pas ces clés étrangères dans l'association "Notation".

### 3. Modèle entités-associations réalisé avec l'AGL



### 4. Script SQL de création des tables généré par l'AGL

```

1 CREATE TABLE public.PERSONNE (
2     id_personne INTEGER NOT NULL,
3     nom_personne VARCHAR NOT NULL,
4     prenom_personne VARCHAR NOT NULL,
5     CONSTRAINT id_personne PRIMARY KEY (id_personne)
6 );
7
8
9
10 CREATE TABLE public.ETUDIANT (
11     id_etudiant INTEGER NOT NULL,
12     CONSTRAINT id_etudiant PRIMARY KEY (id_etudiant)
13 );
14
15
16 CREATE TABLE public.ENSEIGNANT (
17     id_enseignant INTEGER NOT NULL,
18     CONSTRAINT id_enseignant PRIMARY KEY (id_enseignant)
19 );
20
21
22 CREATE TABLE public.MODULE (
23     id_module INTEGER NOT NULL,
24     code VARCHAR NOT NULL,
25     intitule_module VARCHAR NOT NULL,
26     UE VARCHAR NOT NULL,
27     id_enseignant INTEGER NOT NULL,
28     CONSTRAINT id_module PRIMARY KEY (id_module)
29 );
30
31
32 CREATE TABLE public.EVALUATION (
33     id_evaluation INTEGER NOT NULL,
34     nom_evaluation VARCHAR NOT NULL,
35     Date DATE NOT NULL,
36     id_module INTEGER NOT NULL,
37     CONSTRAINT id_evaluation PRIMARY KEY (id_evaluation)
38 );
39
40
41 CREATE TABLE public.NOTATION (
42     id_etudiant INTEGER NOT NULL,
43     id_evaluation INTEGER NOT NULL,
44     Note REAL NOT NULL,
45     CONSTRAINT id_notation PRIMARY KEY (id_etudiant, id_evaluation)
46 );
47
  
```

```

48
49 ALTER TABLE public.ENSEIGNANT ADD CONSTRAINT personne_enseignant_fk
50 FOREIGN KEY (id_enseignant)
51 REFERENCES public.PERSONNE (id_personne)
52 ON DELETE CASCADE
53 ON UPDATE CASCADE
54 NOT DEFERRABLE;
55
56 ALTER TABLE public.ETUDIANT ADD CONSTRAINT personne_etudiant_fk
57 FOREIGN KEY (id_etudiant)
58 REFERENCES public.PERSONNE (id_personne)
59 ON DELETE CASCADE
60 ON UPDATE CASCADE
61 NOT DEFERRABLE;
62
63 ALTER TABLE public.NOTATION ADD CONSTRAINT etudiant_notation_fk
64 FOREIGN KEY (id_etudiant)
65 REFERENCES public.ETUDIANT (id_etudiant)
66 ON DELETE NO ACTION
67 ON UPDATE NO ACTION
68 NOT DEFERRABLE;
69
70 ALTER TABLE public.MODULE ADD CONSTRAINT enseignant_module_fk
71 FOREIGN KEY (id_enseignant)
72 REFERENCES public.ENSEIGNANT (id_enseignant)
73 ON DELETE CASCADE
74 ON UPDATE CASCADE
75 NOT DEFERRABLE;
76
77 ALTER TABLE public.EVALUATION ADD CONSTRAINT module_evaluation_fk
78 FOREIGN KEY (id_module)
79 REFERENCES public.MODULE (id_module)
80 ON DELETE CASCADE
81 ON UPDATE CASCADE
82 NOT DEFERRABLE;
83
84 ALTER TABLE public.NOTATION ADD CONSTRAINT evaluation_notation_fk
85 FOREIGN KEY (id_evaluation)
86 REFERENCES public.EVALUATION (id_evaluation)
87 ON DELETE NO ACTION
88 ON UPDATE NO ACTION
89 NOT DEFERRABLE;
  
```

## 5. Discussion sur les différences entre les scripts produits manuellement et automatiquement

La grosse différence entre les scripts produits manuellement et automatiquement c'est que dans le script généré par AGL, on a ALTER TABLE pour chaque table du modèle entités-associations. Pour chaque ALTER TABLE, on trouve les clés étrangères de chaque table, par exemple dans la table "Module", on trouve id\_enseignant en clé étrangère. Dans le script AGL, nous n'avons pas les DROP TABLE au début du fichier.

Du coup pour cette dernière partie pour cette SAE, je vais vous montrer comment j'ai fait pour remplir toutes mes tables.

### 3.Peuplement des tables et requêtes:

#### 1. Description commentée des différentes étapes de mon script de peuplement.

Pour commencer, j'ai dû créer une nouvelle table "temp" dans laquelle je vais stocker toutes les données du fichier csv.

Pour ça j'ai écrit une nouvelle script pour la création de la table "temp" et pour remplir la table :

```
1  DROP TABLE temp;
2
3  CREATE TABLE temp (
4  id_enseignant INT,
5  nom_enseignant VARCHAR NOT NULL,
6  prenom_enseignant VARCHAR NOT NULL,
7  id_module INT,
8  code VARCHAR NOT NULL,
9  UE VARCHAR NOT NULL,
10 intitule_module VARCHAR NOT NULL,
11 nom_evaluation VARCHAR NOT NULL,
12 date_evaluation VARCHAR,
13 note REAL CHECK (note > 0 OR note IS NULL),
14 id_etudiant INT,
15 nom_etudiant VARCHAR NOT NULL,
16 prenom_etudiant VARCHAR NOT NULL
17 );
18
19 COPY temp FROM '/Users/agash/Desktop/data.csv' DELIMITER ';' CSV Header;
20
```

Puis par la suite j'ai ajouté le script SQL que j'ai écrite dans la première partie avec quelques modifications, j'ai enlevé la table "Personne" car ça me posait des problèmes et j'ai enlevé "Notation" et j'ai ajouté "Note" dans "Evaluation".

J'ai aussi enlevé les clés primaires et étrangères car ça m'affiche des erreurs de type "cette clé existe déjà".

```
1  DROP TABLE etudiant;
2  DROP TABLE evaluation;
3  DROP TABLE module;
4  DROP TABLE enseignant;
5
6
7  CREATE TABLE etudiant (
8      id_etudiant INT,
9      nom_etudiant VARCHAR NOT NULL,
10     prenom_etudiant VARCHAR NOT NULL
11 );
12
13 CREATE TABLE enseignant (
14     id_enseignant INT,
15     nom_enseignant VARCHAR NOT NULL,
16     prenom_enseignant VARCHAR NOT NULL
17 );
18
19 CREATE TABLE module (
20     id_module INT,
21     code VARCHAR NOT NULL,
22     UE VARCHAR NOT NULL,
23     intitule_module VARCHAR NOT NULL
24 );
25
26 CREATE TABLE evaluation (
27     nom_evaluation VARCHAR NOT NULL,
28     date_evaluation VARCHAR NOT NULL,
29     note REAL CHECK (note > 0 OR note IS NULL)
30 );
```

Puis j'ajoute la commande INSERT INTO table (SELECT attr1, attr2 ... FROM temp); après chaque création de table.

```
28 CREATE TABLE etudiant (
29     id_etudiant INT,
30     nom_etudiant VARCHAR NOT NULL,
31     prenom_etudiant VARCHAR NOT NULL
32 );
33
34 INSERT INTO etudiant (SELECT id_etudiant, nom_etudiant, prenom_etudiant FROM temp);
35
36 CREATE TABLE enseignant (
37     id_enseignant INT,
38     nom_enseignant VARCHAR NOT NULL,
39     prenom_enseignant VARCHAR NOT NULL
40 );
41
42 INSERT INTO enseignant (SELECT id_enseignant, nom_enseignant, prenom_enseignant FROM temp);
43
44 CREATE TABLE module (
45     id_module INT,
46     code VARCHAR NOT NULL,
47     UE VARCHAR NOT NULL,
48     intitule_module VARCHAR NOT NULL
49 );
50
51 INSERT INTO module (SELECT id_module, code, UE, intitule_module FROM temp);
52
53 CREATE TABLE evaluation (
54     nom_evaluation VARCHAR NOT NULL,
55     date_evaluation VARCHAR NOT NULL,
56     note REAL CHECK (note > 0 OR note IS NULL)
57 );
58
59 INSERT INTO evaluation (SELECT nom_evaluation, date_evaluation, note FROM temp);
```



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

```
SELECT DISTINCT nom_etudiant, prenom_etudiant FROM etudiant ORDER  
BY id_etudiant ASC;
```

Cette requête va nous lister de façon distincte les noms et prénoms de chaque étudiant présent dans la promotion.

```
SELECT DISTINCT intitule_module FROM module;
```

Cette requête va lister de façon distincte les noms des intitulés modules de la promotion.