

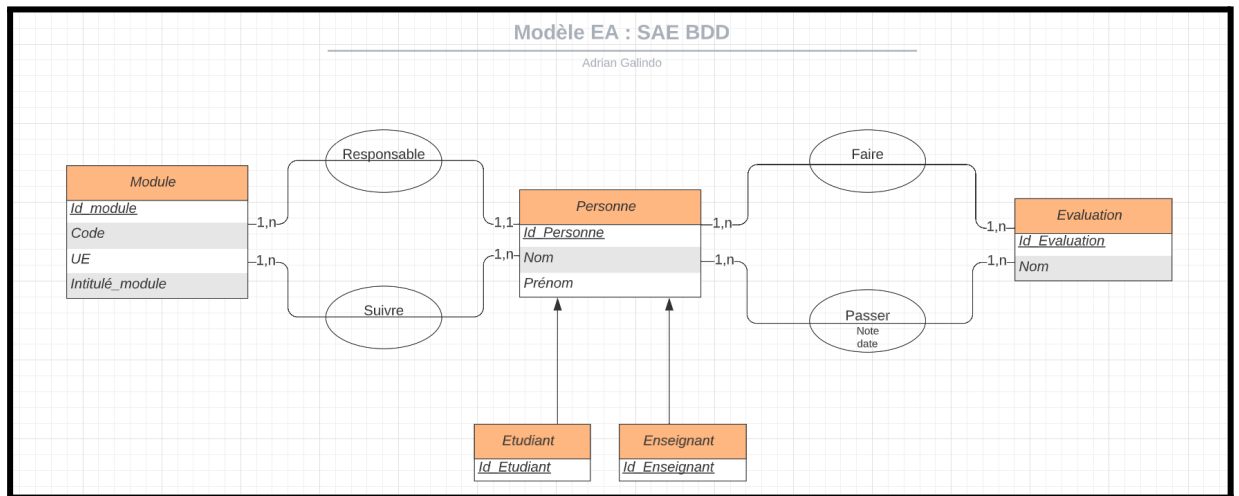
**Bases de données et langage SQL**  
**Situation d'apprentissage et d'Évaluation (SAE) S104**  
**Création d'une base de données : Notations**

**Sommaire**

1. Modélisation et script de création “ sans AGL ”
  - a. Modèle entités-associations.
  - b. Schéma relationnel.
  - c. Script SQL.
2. Modélisation et script de création “ avec ALG ”
  - a. Illustrations comparatives cours/AGL commentées d'une association fonctionnelle.
  - b. Illustrations comparatives cours/AGL commentées d'une association maillée.
  - c. Modèle entités-associations réalisé avec l'AGL.
  - d. Script SQL de création des tables généré automatiquement par l'AGL
  - e. Discussion sur les différences entre les scripts produits manuellement et automatiquement.
3. Peuplement des tables
  - a. Description commentée des différentes étapes de votre script de peuplement.
  - b. Présentation de deux requêtes intéressantes sur la base de données.

## 1. Modélisation et script de création “ sans AGL “

### a. Modèle entités-associations.



### b. Schéma relationnel.

- **Personne** (Id personne, nom, prénom)
- **Etudiant** (Id Etudiant) où Id\_Etudiant est une clé étrangère qui fait référence au schéma de relation **Personne**
- **Enseignant** (Id Enseignant) où Id\_Enseignant est une clé étrangère qui fait référence au schéma de relation **Personne**
- **Module** (Id Module, intitulé\_module, code, UE, id Enseignant) où id\_Enseignant est le responsable du module, id\_Enseignant fait référence au schéma **Personne**
- **Faire** (Id Evaluation, Id personne) où Id\_Evaluation, Id personne font référence à **Evaluation** et **Personne**
- **Passer** (Id Evaluation, Id personne, date, note) où Id\_Evaluation, Id personne font référence à **Evaluation** et **Personne**
- **Responsable** (Id module, Id personne) où Id module, Id personne font référence à **Module** et **Personne**
- **Suivre** (Id module, Id personne) où Id module, Id personne font référence à **Module** et **Personne**
- **Evaluation** (Id\_Evaluation, nom)

### c. Script SQL.

```

CREATE TABLE enseignant
(
    id_enseignant INTEGER REFERENCES personne(id_personne),
);
CREATE TABLE etudiant
(
    id_etudiant INTEGER REFERENCES personne(id_personne),
);
CREATE TABLE passer
(

```

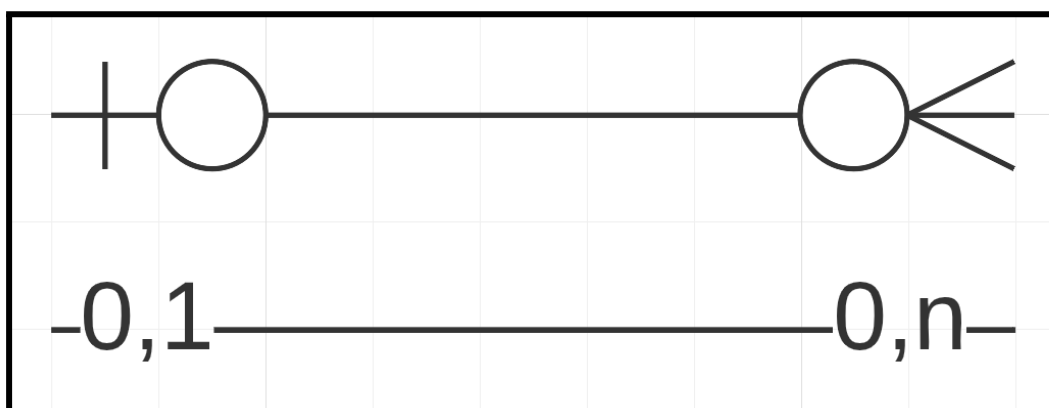
```

    id_evaluation INTEGER REFERENCES evaluation(id_evaluation),
    id_etudiant INTEGER REFERENCES personne(id_etudiant),
    jour DATE,
    note INTEGER,
    PRIMARY KEY (id_evaluation, id_etudiant)
) ;
CREATE TABLE personne
(
    id_personne INTEGER PRIMARY KEY
    nom VARCHAR,
    prenom VARCHAR
) ;
CREATE TABLE evaluation
(
    id_evaluation INTEGER KEY PRIMARY,
    nom VARCHAR
) ;
CREATE TABLE module
(
    id_module INTEGER KEY PRIMARY,
    Code VARCHAR,
    UE VARCHAR,
    Intitule_module VARCHAR,
    id_responsable INTEGER REFERENCES personne(id_enseignant)
) ;

```

## 2. Modélisation et script de création “avec AGL”

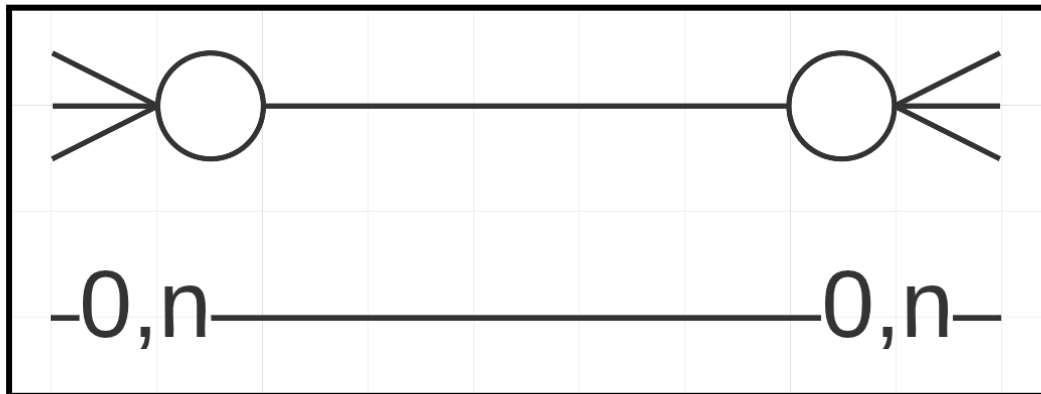
### d. Illustrations comparatives cours/AGL commentées d’une association fonctionnelle.



Comme vous pouvez le constater, la représentation des cardinalités entre deux types d'entités change, dans l'image en dessus nous avons que la première ligne est la représentation utilisée dans les logiciels, tandis que la

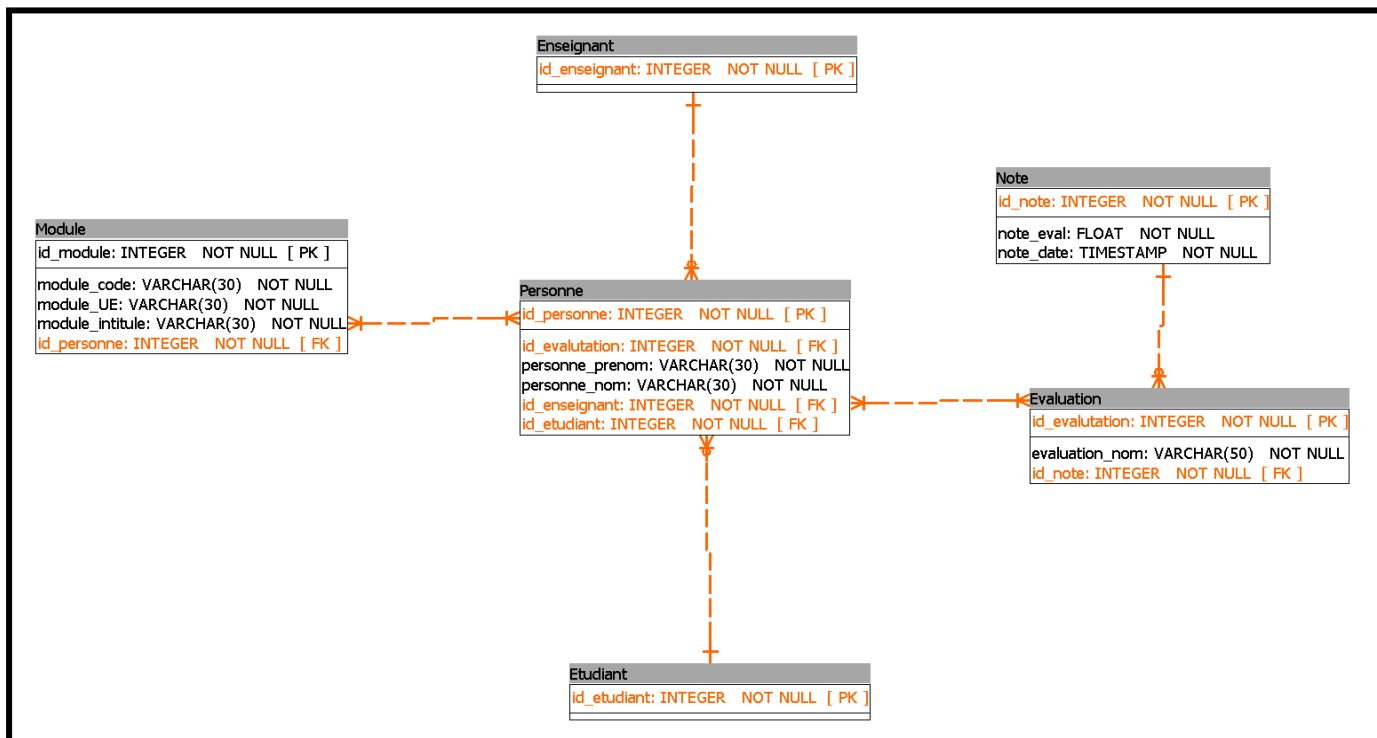
deuxième est celui que nous avons appris en cours. Nous pouvons voir que la cardinalité minimum est représentée à l'intérieur et le maximum à l'extérieur. Le rond représente le 0, la barre le 1 et plusieurs traits équivalent à n.

**e. Illustrations comparatives cours/AGL commentées d'une association maillée.**



Ici, c'est le même principe que celui d'avant, la seule différence sont les cardinalités.

**f. Modèle entités-associations réalisé avec l'AGL**



**g. Scripte SQL de création des tables généré automatiquement par l'AGL**

```
CREATE SEQUENCE note_id_note_seq_1;
```

```
CREATE TABLE Note (
    id_note INTEGER NOT NULL DEFAULT
nextval('note_id_note_seq_1'),
    note_eval REAL NOT NULL,
    note_date TIMESTAMP NOT NULL,
    CONSTRAINT id_note PRIMARY KEY (id_note)
);

ALTER SEQUENCE note_id_note_seq_1 OWNED BY Note.id_note;

CREATE SEQUENCE evaluation_id_evaluation_seq_1;

CREATE TABLE Evaluation (
    id_evaluation INTEGER NOT NULL DEFAULT
nextval('evaluation_id_evaluation_seq_1'),
    evaluation_nom VARCHAR(50) NOT NULL,
    id_note INTEGER NOT NULL,
    CONSTRAINT id_evaluation PRIMARY KEY
(id_evaluation)
);

ALTER SEQUENCE evaluation_id_evaluation_seq_1 OWNED BY
Evaluation.id_evaluation;

CREATE SEQUENCE enseignant_id_enseignant_seq_1;

CREATE TABLE Enseignant (
    id_enseignant INTEGER NOT NULL DEFAULT
nextval('enseignant_id_enseignant_seq_1'),
    CONSTRAINT id_enseignant PRIMARY KEY
(id_enseignant)
);

ALTER SEQUENCE enseignant_id_enseignant_seq_1 OWNED BY
Enseignant.id_enseignant;

CREATE SEQUENCE etudiant_id_etudiant_seq_1;

CREATE TABLE Etudiant (
```

```
        id_etudiant INTEGER NOT NULL DEFAULT
nextval('etudiant_id_etudiant_seq_1'),
        CONSTRAINT id_etudiant PRIMARY KEY (id_etudiant)
);

ALTER SEQUENCE etudiant_id_etudiant_seq_1 OWNED BY
Etudiant.id_etudiant;

CREATE SEQUENCE personne_id_personne_seq;

CREATE TABLE Personne (
        id_personne INTEGER NOT NULL DEFAULT
nextval('personne_id_personne_seq'),
        id_evaluation INTEGER NOT NULL,
        personne_prenom VARCHAR(30) NOT NULL,
        personne_nom VARCHAR(30) NOT NULL,
        id_enseignant INTEGER NOT NULL,
        id_etudiant INTEGER NOT NULL,
        CONSTRAINT id_personne PRIMARY KEY (id_personne)
);

ALTER SEQUENCE personne_id_personne_seq OWNED BY
Personne.id_personne;

CREATE SEQUENCE module_id_module_seq;

CREATE TABLE Module (
        id_module INTEGER NOT NULL DEFAULT
nextval('module_id_module_seq'),
        module_code VARCHAR(30) NOT NULL,
        module_UE VARCHAR(30) NOT NULL,
        module_intitule VARCHAR(30) NOT NULL,
        id_personne INTEGER NOT NULL,
        CONSTRAINT id_module PRIMARY KEY (id_module)
);

ALTER SEQUENCE module_id_module_seq OWNED BY Module.id_module;

ALTER TABLE Evaluation ADD CONSTRAINT note_evaluation_fk
FOREIGN KEY (id_note)
```

```
REFERENCES Note (id_note)
ON DELETE NO ACTION
ON UPDATE NO ACTION
NOT DEFERRABLE;

ALTER TABLE Personne ADD CONSTRAINT evaluation_personne_fk
FOREIGN KEY (id_evaluation)
REFERENCES Evaluation (id_evaluation)
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 Personne ADD CONSTRAINT etudiant_personne_fk
FOREIGN KEY (id_etudiant)
REFERENCES Etudiant (id_etudiant)
ON DELETE NO ACTION
ON UPDATE NO ACTION
NOT DEFERRABLE;

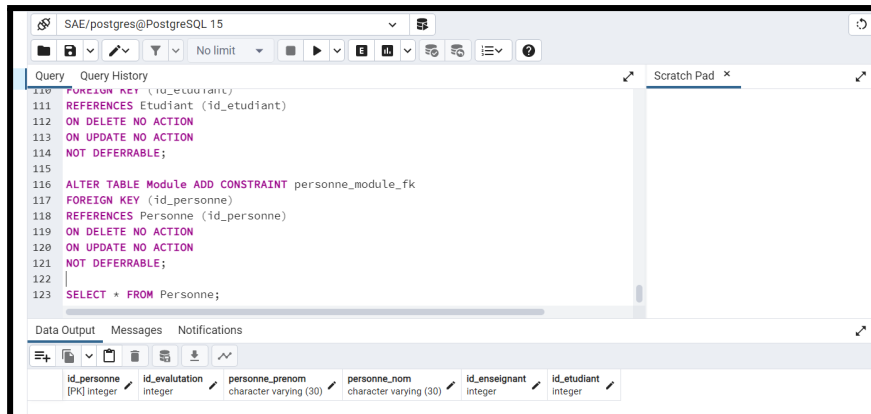
ALTER TABLE Module ADD CONSTRAINT personne_module_fk
FOREIGN KEY (id_personne)
REFERENCES Personne (id_personne)
ON DELETE NO ACTION
ON UPDATE NO ACTION
NOT DEFERRABLE;
```

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

Pour commencer, il y a une grande différence qui saute aux yeux, la taille du script. Le logiciel fait des commandes que nous n'avons pas encore vu en cours, cela reste compréhensible avec des tutoriels, mais nous voyons une grande différence entre un humain et une machine, en termes de temps de réalisation.

### 3. Peuplement des tables

## i. Description commentée des différentes étapes de votre script de peuplement



```
110 FOREIGN KEY (id_etudiant)
111 REFERENCES Etudiant (id_etudiant)
112 ON DELETE NO ACTION
113 ON UPDATE NO ACTION
114 NOT DEFERRABLE;
115
116 ALTER TABLE Module ADD CONSTRAINT personne_module_fk
117 FOREIGN KEY (id_personne)
118 REFERENCES Personne (id_personne)
119 ON DELETE NO ACTION
120 ON UPDATE NO ACTION
121 NOT DEFERRABLE;
122
123 SELECT * FROM Personne;
```

The screenshot shows a PostgreSQL 15 query editor interface. The main window displays a SQL script with line numbers 110 through 123. The script defines a foreign key constraint for the 'Module' table, referencing the 'Personne' table. The constraint is named 'personne\_module\_fk' and is not deferrable. Below the query editor, there is a 'Data Output' section showing the results of the 'SELECT \* FROM Personne;' query. The output table has five columns: 'id\_personne' (integer, primary key), 'id\_evaluation' (integer), 'personne\_prenom' (character varying (30)), 'personne\_nom' (character varying (30)), and 'id\_enseignant' (integer). The 'id\_etudiant' column is also listed in the output table.

Nous pouvons constater que le type entité **Personne** a été créé, Il ne reste qu' à faire des teste par rapport entre les notes des étudiants en vérifiant que les notes et les dates sont bien enregistrés et les modules avec ses responsables..