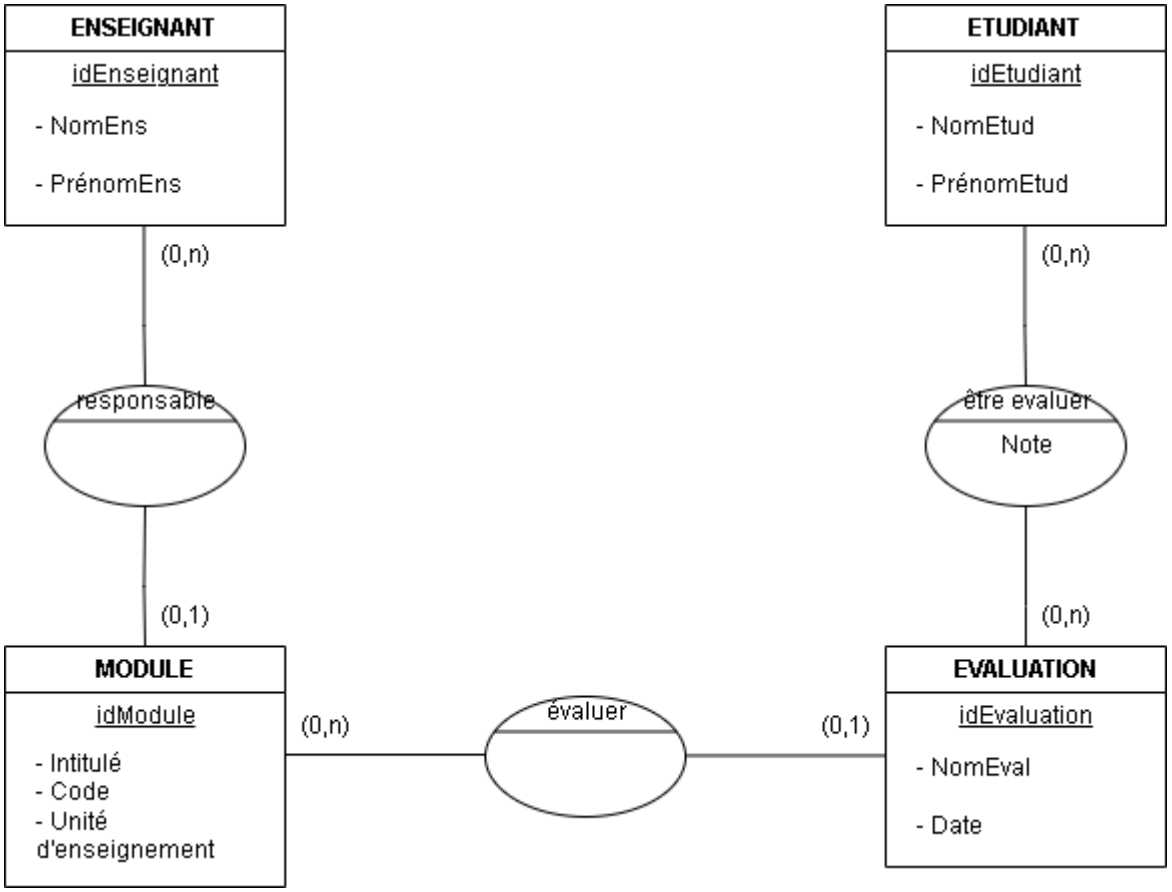
DONG Sophie BUT INFO 1 - Tlaloc

SAE Création d'une base de données : Notations

Sommaire :

1.1 - Modélisation et script de création "sans AGL"

1 - Modèle entités-associations respectant la syntaxe du cours



2 - Schéma relationnel

ENSEIGNANT(idEnseignant, nomEns, prénomEns)
ETUDIANT(idEtudiant, nomEtu, prénomEtu)
MODULE(idModule,intitulé,code,unité d'enseignement, idEnseignant*)
EVALUATION(idEvaluation, nomEval, date, idModule*)
ÊTRE EVALUER(idEtudiant*, idEvaluation*, Note)

Légende : * → représente les clés étrangères

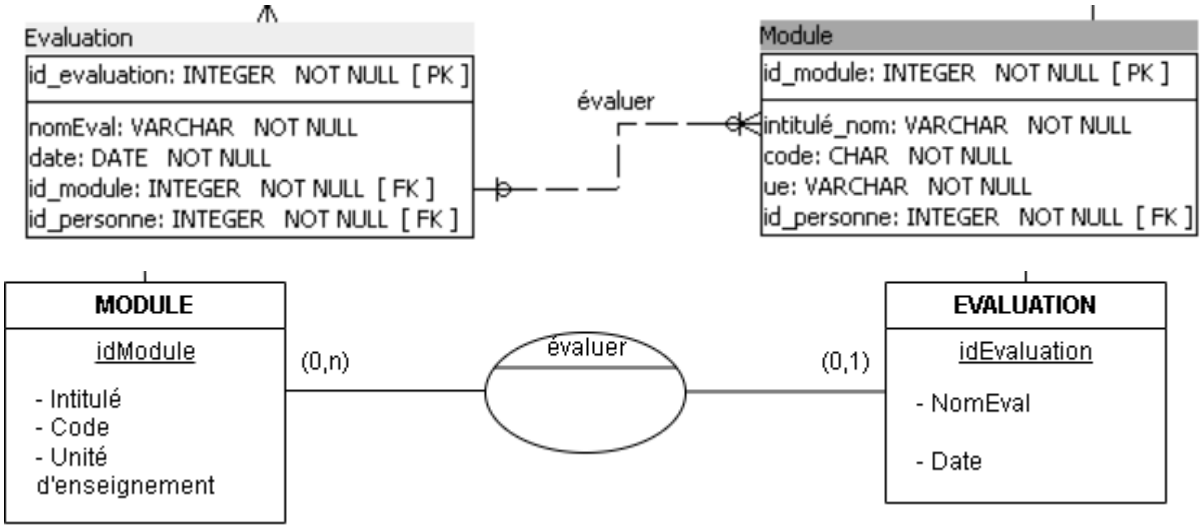
3 - Script SQL de création des tables

```
CREATE TABLE personne (id_personne INTEGER PRIMARY KEY, nom
VARCHAR NOT NULL, prenom VARCHAR NOT NULL);
CREATE TABLE module (id_module INTEGER PRIMARY KEY, intitulé
VARCHAR NOT NULL, code CHAR(4) UNIQUE, unite_enseignement CHAR(4)
UNIQUE, id_enseignant INTEGER REFERENCES personne (id_personne) ON
DELETE CASCADE);
```

```
CREATE TABLE evaluation (id_evaluation INTEGER PRIMARY KEY,
nomEval VARCHAR NOT NULL, date DATE, id_module INTEGER REFERENCES
module(id_module));
CREATE TABLE etre_evaluer (id_etudiant INTEGER REFERENCES personne
(id_personne) ON DELETE CASCADE, id_evaluation INTEGER REFERENCES
evaluation ON DELETE CASCADE, note FLOAT, PRIMARY KEY
(id_etudiant, id_evaluation));
```

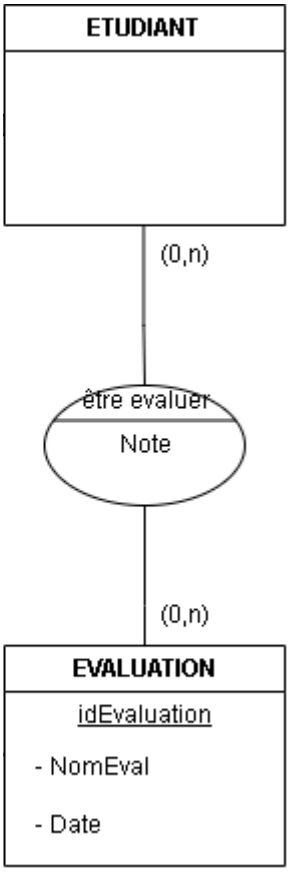
1.2 - Modélisation et script de création "avec AGL"

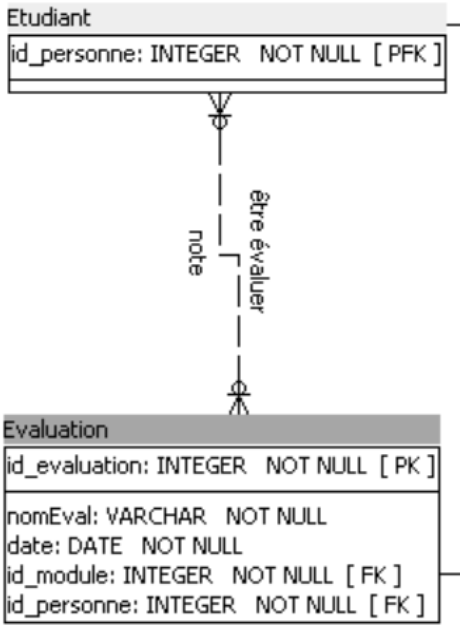
1 - Illustrations comparatives cours/AGL commentée d'une association fonctionnelle



Comparaison avec AGL	**Comparaison sans AGL**
Les cardinalités sont représentées sous forme de symbole logique	Les cardinalités sont représentées entre parenthèse
Le type des attributs est indiqué	Le type des attributs n'est pas indiqué
Les clés étrangères sont indiquées avec l'acronyme 'FK'	Les clés étrangères n'apparaissent pas
Les clés primaires sont indiquées avec l'acronyme 'PK'	Les clés primaires sont soulignées

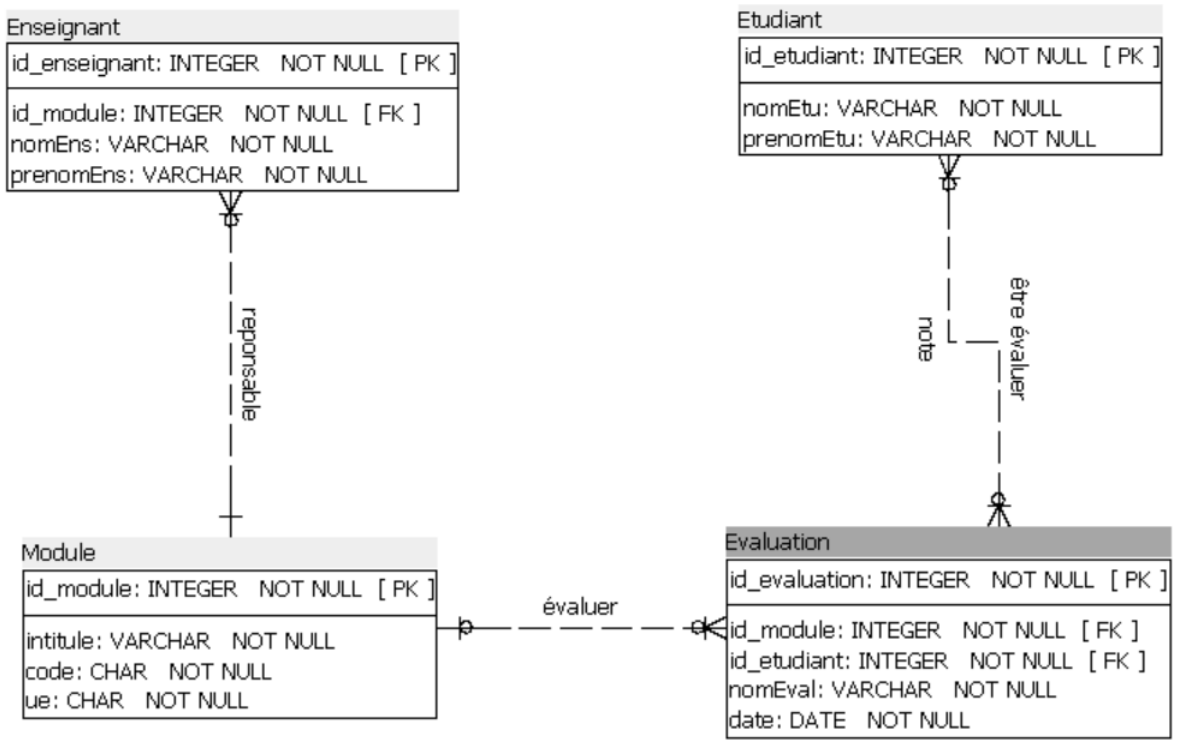
2 - Illustration comparatives cours/AGL commentée d'une association maillée





Comparaison avec AGL	**Comparaison sans AGL**
Les cardinalités sont représentées sous forme de symbole logique	Les cardinalités sont représentées entre parenthèse
Les clés étrangères qui sont clé primaire sont indiquées par 'PFK'	Les clés primaires sont soulignées mais n'est pas montrer explicitement qu'il s'agit d'une clé étrangère
Le type des attributs est indiqué	Le type des attributs n'est pas indiqué
Les clés étrangères sont indiquées avec l'acronyme 'FK'	Les clés étrangères n'apparaissent pas
Les clés primaires sont indiquées avec l'acronyme 'PK'	Les clés primaires sont soulignées

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



4 - Script SQL de création des tables généré automatiquement par l'AGL

```
CREATE TABLE Module (
                id_module INTEGER NOT NULL,
                intitule VARCHAR NOT NULL,
                code CHAR NOT NULL,
                ue CHAR NOT NULL,
                CONSTRAINT id_module PRIMARY KEY (id_module)
);


CREATE TABLE Enseignant (
                id_enseignant INTEGER NOT NULL,
                id_module INTEGER NOT NULL,
                nomEns VARCHAR NOT NULL,
                prenomEns VARCHAR NOT NULL,
                            CONSTRAINT  id_enseignant  PRIMARY  KEY
(id_enseignant)
);


CREATE TABLE Etudiant (
                id_etudiant INTEGER NOT NULL,
                nomEtu VARCHAR NOT NULL,
                prenomEtu VARCHAR NOT NULL,
```

```
                CONSTRAINT id_etudiant PRIMARY KEY (id_etudiant)
);


CREATE TABLE Evaluation (
                id_evaluation INTEGER NOT NULL,
                id_module INTEGER NOT NULL,
                id_etudiant INTEGER NOT NULL,
                nomEval VARCHAR NOT NULL,
                date DATE NOT NULL,
                            CONSTRAINT  id_evaluation  PRIMARY  KEY
(id_evaluation)
);

ALTER TABLE Enseignant ADD CONSTRAINT Module_Enseignant_fk
FOREIGN KEY (id_module)
REFERENCES Module (id_module)
ON DELETE CASCADE
ON UPDATE CASCADE
NOT DEFERRABLE;

ALTER TABLE Evaluation ADD CONSTRAINT Module_Evaluation_fk
FOREIGN KEY (id_module)
REFERENCES Module (id_module)
ON DELETE CASCADE
ON UPDATE CASCADE
NOT DEFERRABLE;

ALTER TABLE Evaluation ADD CONSTRAINT Etudiant_Evaluation_fk
FOREIGN KEY (id_etudiant)
REFERENCES Etudiant (id_etudiant)
ON DELETE CASCADE
ON UPDATE CASCADE
NOT DEFERRABLE;
```

5 - Discussion sur les différences entre les scripts produits manuellement et automatiquement

Les scripts produits manuellement sont plus courts que ceux produits automatiquement. Nous avons écrit que le strict nécessaire pour la création des tables avec les contraintes qu'on demande exactement pour les types d'attributs contrairement aux scripts automatiques qui ne prennent pas en compte les contraintes mais seulement le type avec pour contrainte par défaut 'NOT NULL'.

1.3 - Peuplement des tables et requêtes

1 - Description commentée des différentes étapes de votre script de peuplement

Premièrement → créer une table temporaire contenant toutes les clés et attributs du fichier csv.

```
CREATE TABLE sae (id_enseignant INTEGER,nom_enseignant VARCHAR,prenom_enseignant VARCHAR,id_module INTEGER,code VARCHAR,ue VARCHAR,intitule_module VARCHAR, nom_evaluation VARCHAR,date_evaluation DATE,note FLOAT,id_etudiant INTEGER,nom_etudiant VARCHAR,prenom_etudiant VARCHAR);
```

Ensuite copier dans la table temporaire le contenu du fichier csv

```
COPY sae
FROM 'C:\Users\Public\data.csv'
DELIMITER ';'
CSV Header;
```

Puis créer les différentes tables :

```
CREATE TABLE enseignant (id_enseignant INTEGER PRIMARY KEY, nom_enseignant VARCHAR NOT NULL, prenom_enseignant VARCHAR NOT NULL);
CREATE TABLE etudiant (id_etudiant INTEGER PRIMARY KEY, nom_etudiant VARCHAR NOT NULL, prenom_etudiant VARCHAR NOT NULL);
CREATE TABLE module (id_module INTEGER PRIMARY KEY, code VARCHAR NOT NULL, ue VARCHAR NOT NULL, intitule_module VARCHAR NOT NULL, id_enseignant INTEGER REFERENCES enseignant(id_enseignant) ON DELETE CASCADE);
CREATE TABLE evaluation (id_evaluation INTEGER PRIMARY KEY, nom_evaluation VARCHAR NOT NULL, date_evaluation VARCHAR NOT NULL, id_module INTEGER REFERENCES module (id_module) ON DELETE CASCADE);
CREATE TABLE etre_evaluer (id_etudiant INTEGER REFERENCES etudiant (id_etudiant) ON DELETE CASCADE,id_evaluation INTEGER REFERENCES evaluation (id_evaluation) ON DELETE CASCADE, note FLOAT, PRIMARY KEY (id_etudiant,id_evaluation));
```

Et pour finir insérer le contenu de la table temporaire qu'il faut dans les tables correspondantes :

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

```
INSERT INTO etudiant (SELECT DISTINCT id_etudiant, nom_etudiant, prenom_etudiant FROM sae);
INSERT INTO module (SELECT DISTINCT id_module, code, ue, intitule_module FROM sae);
INSERT INTO evaluation (nom_evaluation, date_evaluation) SELECT nom_evaluation, date_evaluation FROM sae;
INSERT INTO etre_evaluer (SELECT id_etudiant, id_evaluation, note FROM sae);
```

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

Requête 1 :

```
SELECT nom_etudiant, prenom_etudiant FROM etudiant JOIN etre_evaluer USING (id_etudiant) WHERE etre_evaluer.nom_evalutation='Examen final' AND etre_evaluer.note>=10;
```

Cette requête peut être intéressante et très utile pour pouvoir avoir les noms et prénoms des étudiants ayant eu une note égale ou supérieure à 10 à l'examen final qui détermine s'ils ont la possibilité de passer à l'année supérieure de leurs études.

Requête 2 :

```
SELECT COUNT(DISTINCT id_etudiant) AS nb_etudiant FROM etudiant;
```

Cette requête est intéressante car elle nous permet d'avoir le nombre exact d'étudiants dans la promotion sans éventuel doublon.