ANANDASANGAR
Agash
BUT1
Tlaloc

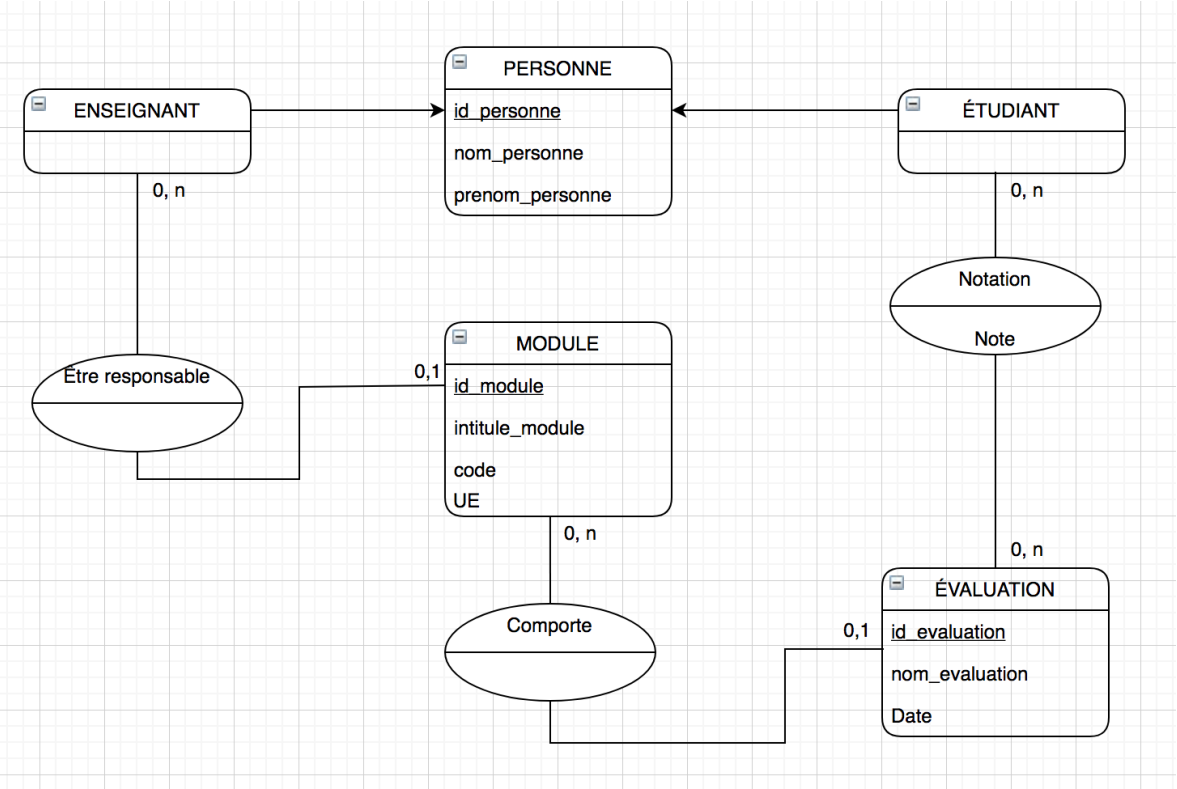
RAPPORT SAE 1.04 BASE DE DONNÉES

SOMMAIRE :

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_enseigant 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

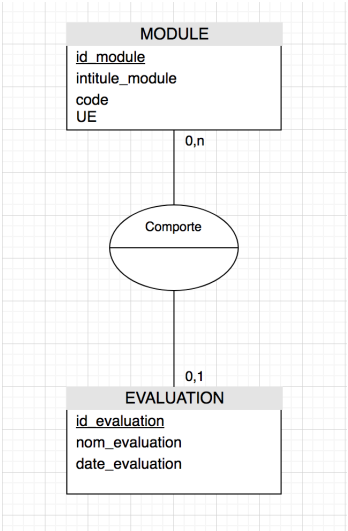
3. Script SQL de création des tables

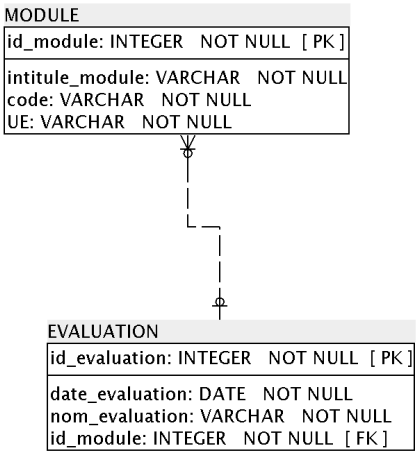
```sql
DROP TABLE notation;
DROP TABLE etudiant;
DROP TABLE evaluation;
DROP TABLE module;
DROP TABLE enseignant;
DROP TABLE personne;

CREATE TABLE personne (
    id_personne INT PRIMARY KEY,
    nom_personne VARCHAR NOT NULL,
    prenom_personne VARCHAR NOT NULL
  );


CREATE TABLE etudiant (
    id_etudiant INT REFERENCES personne (id_personne)ON DELETE CASCADE,
    PRIMARY KEY (id_etudiant)
  );

CREATE TABLE enseignant (
    id_enseignant INT REFERENCES personne (id_personne)ON DELETE CASCADE,
    PRIMARY KEY (id_enseignant)
  );

CREATE TABLE module (
    id_module INT PRIMARY KEY,
    code VARCHAR NOT NULL,
    ue VARCHAR NOT NULL,
    intitule_module VARCHAR NOT NULL,
    id_enseignant INT REFERENCES personne (id_personne)ON DELETE CASCADE
  );

CREATE TABLE evaluation (
    id_evaluation INT PRIMARY KEY,
    nom_evaluation VARCHAR NOT NULL,
    date_evaluation VARCHAR NOT NULL,
    id_module INT REFERENCES module (id_module) ON DELETE CASCADE
  );

CREATE TABLE notation (
    id_etudiant INT REFERENCES personne (id_personne)ON DELETE CASCADE,
    id_evaluation INT REFERENCES evaluation (id_evaluation)ON DELETE CASCADE,
    PRIMARY KEY (id_etudiant, id_evaluation),
    note REAL CHECK (note > 0 OR note IS NULL)
  );
```

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

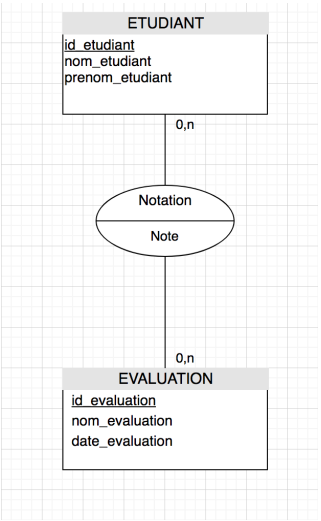


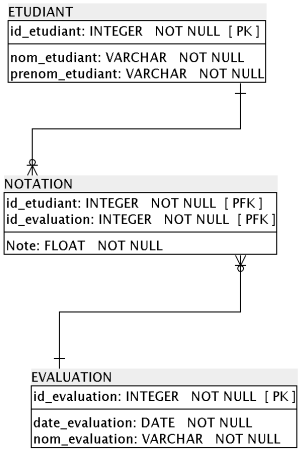


cours AGL

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

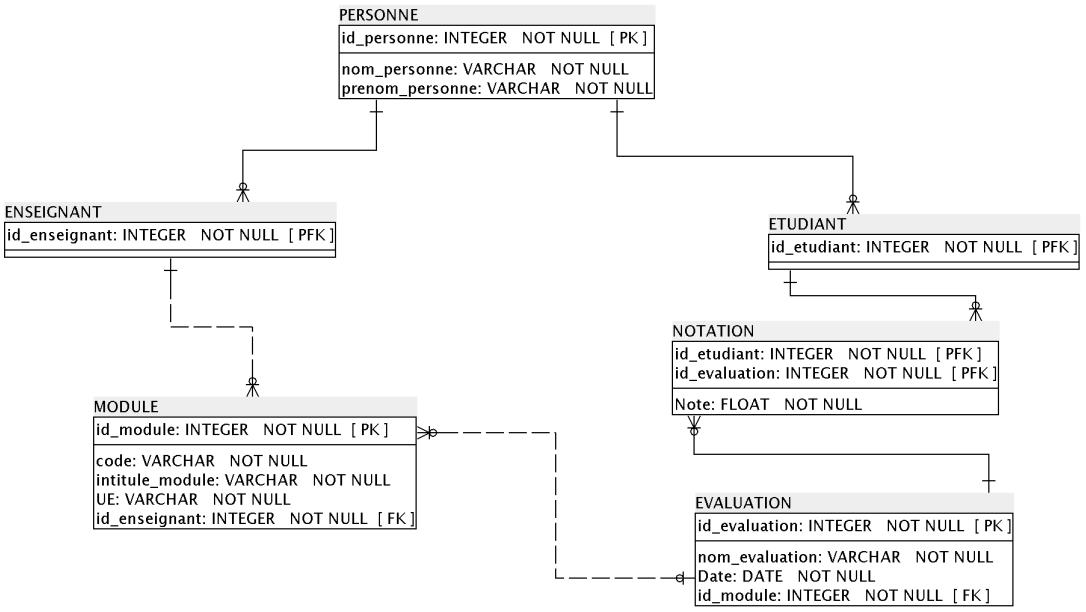




cours AGL

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

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


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


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


CREATE TABLE public.MODULE (
                id_module INTEGER NOT NULL,
                code VARCHAR NOT NULL,
                intitule_module VARCHAR NOT NULL,
                UE VARCHAR NOT NULL,
                id_enseignant INTEGER NOT NULL,
                CONSTRAINT id_module PRIMARY KEY (id_module)
);


CREATE TABLE public.EVALUATION (
                id_evaluation INTEGER NOT NULL,
                nom_evaluation VARCHAR NOT NULL,
                Date DATE NOT NULL,
                id_module INTEGER NOT NULL,
                CONSTRAINT id_evaluation PRIMARY KEY (id_evaluation)
);


CREATE TABLE public.NOTATION (
                id_etudiant INTEGER NOT NULL,
                id_evaluation INTEGER NOT NULL,
                Note REAL NOT NULL,
                CONSTRAINT id_notation PRIMARY KEY (id_etudiant, id_evaluation)
);

```

```sql

ALTER TABLE public.ENSEIGNANT ADD CONSTRAINT personne_enseignant_fk
FOREIGN KEY (id_enseignant)
REFERENCES public.PERSONNE (id_personne)
ON DELETE CASCADE
ON UPDATE CASCADE
NOT DEFERRABLE;

ALTER TABLE public.ETUDIANT ADD CONSTRAINT personne_etudiant_fk
FOREIGN KEY (id_etudiant)
REFERENCES public.PERSONNE (id_personne)
ON DELETE CASCADE
ON UPDATE CASCADE
NOT DEFERRABLE;

ALTER TABLE public.NOTATION ADD CONSTRAINT etudiant_notation_fk
FOREIGN KEY (id_etudiant)
REFERENCES public.ETUDIANT (id_etudiant)
ON DELETE NO ACTION
ON UPDATE NO ACTION
NOT DEFERRABLE;

ALTER TABLE public.MODULE ADD CONSTRAINT enseignant_module_fk
FOREIGN KEY (id_enseignant)
REFERENCES public.ENSEIGNANT (id_enseignant)
ON DELETE CASCADE
ON UPDATE CASCADE
NOT DEFERRABLE;

ALTER TABLE public.EVALUATION ADD CONSTRAINT module_evaluation_fk
FOREIGN KEY (id_module)
REFERENCES public.MODULE (id_module)
ON DELETE CASCADE
ON UPDATE CASCADE
NOT DEFERRABLE;

ALTER TABLE public.NOTATION ADD CONSTRAINT evaluation_notation_fk
FOREIGN KEY (id_evaluation)
REFERENCES public.EVALUATION (id_evaluation)
ON DELETE NO ACTION
ON UPDATE NO ACTION
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 :

```
DROP TABLE temp;

CREATE TABLE temp (
id_enseignant INT,
nom_enseignant VARCHAR NOT NULL,
prenom_enseignant VARCHAR NOT NULL,
id_module INT,
code VARCHAR NOT NULL,
UE VARCHAR NOT NULL,
intitule_module VARCHAR NOT NULL,
nom_evaluation VARCHAR NOT NULL,
date_evaluation VARCHAR,
note REAL CHECK (note > 0 OR note IS NULL),
id_etudiant INT,
nom_etudiant VARCHAR NOT NULL,
prenom_etudiant VARCHAR NOT NULL
);

COPY temp FROM '/Users/agash/Desktop/data.csv' DELIMITER ';' CSV Header;
```

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”.

```sql
DROP TABLE etudiant;
DROP TABLE evaluation;
DROP TABLE module;
DROP TABLE enseignant;


CREATE TABLE etudiant (
    id_etudiant INT,
    nom_etudiant VARCHAR NOT NULL,
    prenom_etudiant VARCHAR NOT NULL
  );

CREATE TABLE enseignant (
    id_enseignant INT,
    nom_enseignant VARCHAR NOT NULL,
    prenom_enseignant VARCHAR NOT NULL
  );

CREATE TABLE module (
    id_module INT,
    code VARCHAR NOT NULL,
    UE VARCHAR NOT NULL,
    intitule_module VARCHAR NOT NULL
  );

CREATE TABLE evaluation (
    nom_evaluation VARCHAR NOT NULL,
    date_evaluation VARCHAR NOT NULL,
    note REAL CHECK (note > 0 OR note IS NULL)
  );
```

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

```sql
CREATE TABLE etudiant (
    id_etudiant INT,
    nom_etudiant VARCHAR NOT NULL,
    prenom_etudiant VARCHAR NOT NULL
  );

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

CREATE TABLE enseignant (
    id_enseignant INT,
    nom_enseignant VARCHAR NOT NULL,
    prenom_enseignant VARCHAR NOT NULL
  );

INSERT INTO enseignant (SELECT id_enseignant, nom_enseignant, prenom_enseignant FROM temp);

CREATE TABLE module (
    id_module INT,
    code VARCHAR NOT NULL,
    UE VARCHAR NOT NULL,
    intitule_module VARCHAR NOT NULL
  );

INSERT INTO module (SELECT id_module, code, UE, intitule_module FROM temp);

CREATE TABLE evaluation (
    nom_evaluation VARCHAR NOT NULL,
    date_evaluation VARCHAR NOT NULL,
    note REAL CHECK (note > 0 OR note IS NULL)
  );

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.