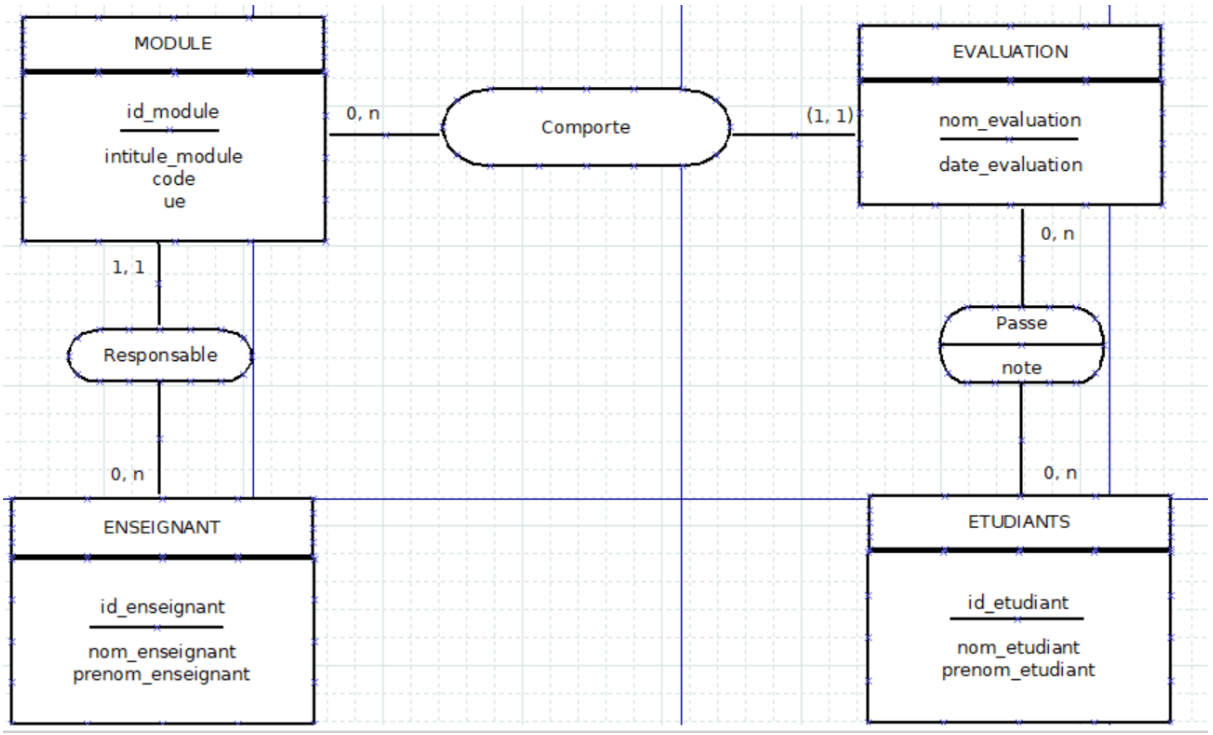
NAIM Charlotte BUT Informatique, groupe ZEUS

Base de donnée et langage SQL

Sommaire

1. Modélisation et script de création “sans AGL”

1) Modèle entités-associations



2) Schéma relationnel

ENSEIGNANT (id_enseignant, nom_enseignant, prenom_enseignant)

ETUDIANT (id_etudiant, nom_etudiant, prenom_etudiant)

MODULE (id_module, id_enseignant, intitule_module, code, ue) id_enseignant qui fait référence à ENSEIGNANT

EVALUATION (nom_evaluation, id_module, date_evaluation) id_module qui fait référence à MODULE

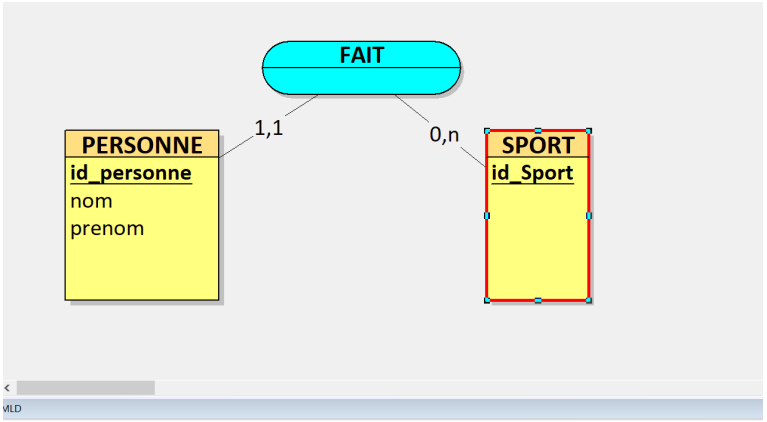
PASSE (id_etudiant, note)

3) Script SQL de création des tables

```
CREATE TABLE ENSEIGNANT(
id_enseignant INTEGER PRIMARY KEY,
nom_enseignant VARCHAR(50) NOT NULL,
prenom_enseignant VARCHAR(50) NOT NULL
);

CREATE TABLE ETUDIANT(
id_etudiant INTEGER PRIMARY KEY,
nom_etudiant VARCHAR(50) NOT NULL,
prenom_etudiant VARCHAR(50) NOT NULL
);

CREATE TABLE MODULE(
id_module INTEGER PRIMARY KEY,
id_enseignant INTEGER SECOND KEY,
intitule_module VARCHAR(200) NOT NULL,
code VARCHAR(10) NOT NULL,
ue VARCHAR(10) NOT NULL
);

CREATE TABLE EVALUATION(
nom_evaluation VARCHAR(100) PRIMARY KEY,
id_module INTEGER SECOND KEY,
date_evaluation DATE,
);

CREATE TABLE PASSE(
id_etudiant INTEGER PRIMARY KEY,
note DOUBLE PRECISION
);
```

2. Modélisation et script de création avec AGL

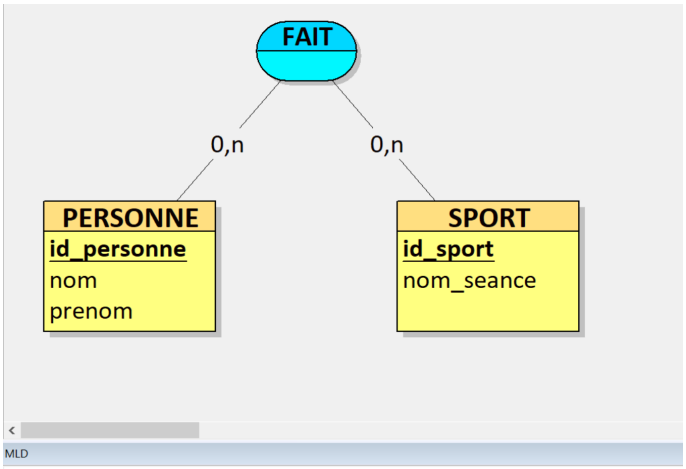
1) Illustration comparatives cours/AGL commentée d'une association fonctionnelle



SPORT = (id_Sport VARCHAR(50));
PERSONNE = (id_personne VARCHAR(50), nom VARCHAR(50), prenom VARCHAR(50), #id_Sport);

La différence se trouve dans le schéma relationnel, dans le cours pour personne on aurait: PERSONNE = (id_personne, id_Sport, nom, prénom)
alors qu'avec AGL on a id_Sport a la fin avec un # et à chaque attribut on a le type.

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



PERSONNE = (id_personne INT, nom VARCHAR(50), prenom VARCHAR(50));
SPORT = (id_sport VARCHAR(50), nom_seance VARCHAR(50));
FAIT = (#id_personne, #id_sport);

La différence se trouve dans le schéma relationnel. C'est toujours le # et les types sur chaque attributs qui changent, de ce qu'on a vu en cours.

3) Modèle entités-associations réalisé avec l'AGL

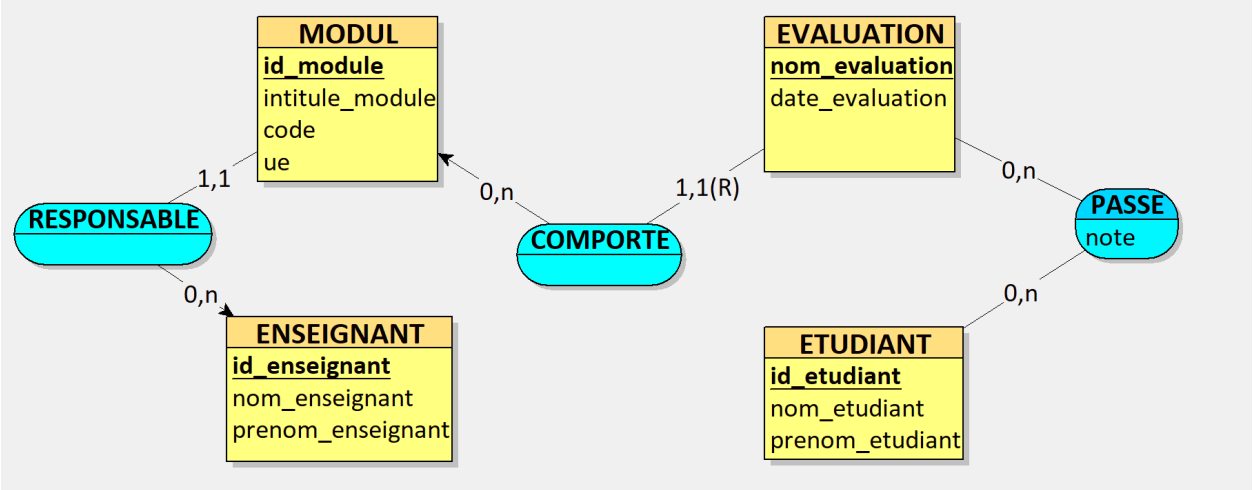


Schéma relationnel généré avec AGL

MLD

ENSEIGNANT = (**id_enseignant** *INT*, nom_enseignant *VARCHAR(50)*, prenom_enseignant *VARCHAR(50)*);
ETUDIANT = (**id_etudiant** *INT*, nom_etudiant *VARCHAR(50)*, prenom_etudiant *VARCHAR(50)*);
MODUL = (**id_module** *INT*, intitule_module *VARCHAR(200)*, code *VARCHAR(10)*, ue *VARCHAR(10)*, *#id_enseignant*);
EVALUATION = (***#id_module***, **nom_evaluation** *VARCHAR(100)*, date_evaluation *DATE*);
PASSE = (***#(#id_module, nom_evaluation), #id_etudiant***, note *INT*);

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

```
CREATE TABLE ENSEIGNANT(
   id_enseignant INT,
   nom_enseignant VARCHAR(50) NOT NULL,
   prenom_enseignant VARCHAR(50) NOT NULL,
   PRIMARY KEY(id_enseignant)
);

CREATE TABLE ETUDIANT(
   id_etudiant INT,
   nom_etudiant VARCHAR(50) NOT NULL,
   prenom_etudiant VARCHAR(50) NOT NULL,
   PRIMARY KEY(id_etudiant)
);

CREATE TABLE MODUL(
   id_module INT,
   intitule_module VARCHAR(200) NOT NULL,
   code VARCHAR(10) NOT NULL,
   ue VARCHAR(10) NOT NULL,
   id_enseignant INT NOT NULL,
   PRIMARY KEY(id_module),
   FOREIGN KEY(id_enseignant) REFERENCES ENSEIGNANT(id_enseignant)
);

CREATE TABLE EVALUATION(
   id_module INT,
   nom_evaluation VARCHAR(100),
   date_evaluation DATE,
   PRIMARY KEY(id_module, nom_evaluation),
   FOREIGN KEY(id_module) REFERENCES MODUL(id_module)
);

CREATE TABLE PASSE(
   id_module INT,
   nom_evaluation VARCHAR(100),
   id_etudiant INT,
   note INT,
   PRIMARY KEY(id_module, nom_evaluation, id_etudiant),
   FOREIGN KEY(id_module, nom_evaluation) REFERENCES EVALUATION(id_module,
nom_evaluation),
   FOREIGN KEY(id_etudiant) REFERENCES ETUDIANT(id_etudiant)
);
```

5) Discussion sur les différences entre les scripts produit manuellement et automatiquement

Pour les schémas entités-associations, sur AGL on a des couleurs et des flèches.
Le (R) pour les cardinalité, signifie que 1,1 est entre parenthèses.
= Plus esthétique avec AGL

Pour les scripts, il y a quelques différences entre le script sans AGL et celui généré par AGL.
Tout d'abord, la table MODUL est sans E dans looping car MODULE est un mot réservé en SQL.
Concernant les clés primaires, dans mon script on a par exemple pour la table ENSEIGNANT (comme vue en cours):

id_enseignant INTEGER PRIMARY KEY

alors que sur AGL, on a:

id_enseignant INT,
PRIMARY KEY(id_enseignant)

Autre différence, il y a plusieurs tables ou il y a des clés étrangères. Par exemple, sur la table EVALUATION (la différence est la ligne avec un point rouge).
Sans AGL:

```
CREATE TABLE EVALUATION(
nom_evaluation VARCHAR(100) PRIMARY KEY,
id_module INTEGER SECOND KEY,
date_evaluation DATE,
);
```

Avec AGL:

```
CREATE TABLE EVALUATION(
id_module INT,
nom_evaluation VARCHAR(100),
date_evaluation DATE,
PRIMARY KEY(id_module, nom_evaluation),
FOREIGN KEY(id_module) REFERENCES MODUL(id_module)
);
```

Dans la table PASSE, ce ne sont pas les mêmes, avec AGL on a id_module et nom_evaluation en plus qui sont des clés secondaires.

3. Peuplement des tables et requêtes

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

Création des différentes tables

```
CREATE TABLE ENSEIGNANT(
   id_enseignant INT,
   nom_enseignant VARCHAR(50) NOT NULL,
   prenom_enseignant VARCHAR(50) NOT NULL,
   PRIMARY KEY(id_enseignant)
);

CREATE TABLE ETUDIANT(
   id_etudiant INT,
   nom_etudiant VARCHAR(50) NOT NULL,
   prenom_etudiant VARCHAR(50) NOT NULL,
   PRIMARY KEY(id_etudiant)
);

CREATE TABLE MODUL(
   id_module INT,
   intitule_module VARCHAR(200) NOT NULL,
   code VARCHAR(10) NOT NULL,
   ue VARCHAR(10) NOT NULL,
   id_enseignant INT NOT NULL,
   PRIMARY KEY(id_module),
   FOREIGN KEY(id_enseignant) REFERENCES ENSEIGNANT(id_enseignant)
);

CREATE TABLE EVALUATION(
   id_module INT,
   nom_evaluation VARCHAR(100),
   date_evaluation DATE,
   PRIMARY KEY(id_module, nom_evaluation),
   FOREIGN KEY(id_module) REFERENCES MODUL(id_module)
);

CREATE TABLE PASSE(
   id_module INT,
   nom_evaluation VARCHAR(100),
   id_etudiant INT,
   note INT,
   PRIMARY KEY(id_module, nom_evaluation, id_etudiant),
```

```
  FOREIGN KEY(id_module, nom_evaluation) REFERENCES EVALUATION(id_module,
nom_evaluation),
  FOREIGN KEY(id_etudiant) REFERENCES ETUDIANT(id_etudiant)
);
```

Commande pour voir si les tables ont bien été créé

```
SELECT * FROM ENSEIGNANT;
SELECT * FROM ETUDIANT;
SELECT * FROM MODUL;
SELECT * FROM EVALUATION;
SELECT * FROM PASSE;
```

Création table INTERMEDIAIRE + ajout des données du fichiers .csv dans la table INTERMEDIAIRE + suppression de la table INTERMEDIAIRE

```
CREATE TABLE INTERMEDIAIRE(
        id_enseignant INT,
        nom_enseignant VARCHAR(50) NOT NULL,
        prenom_enseignant VARCHAR(50) NOT NULL,
        id_module INT,
        code VARCHAR(10) NOT NULL,
        ue VARCHAR(10) NOT NULL,
        intitule_module VARCHAR(200) NOT NULL,
        nom_evaluation VARCHAR(100),
        date_evaluation DATE,
        note REAL,
        id_etudiant INT,
        nom_etudiant VARCHAR(50) NOT NULL,
        prenom_etudiant VARCHAR(50) NOT NULL
);

COPY INTERMEDIAIRE FROM '.\data.csv' DELIMITER ';' CSV HEADER;

INSERT INTO ENSEIGNANT SELECT DISTINCT id_enseignant, nom_enseignant,
prenom_enseignant FROM INTERMEDIAIRE;
INSERT INTO ETUDIANT SELECT DISTINCT id_etudiant, nom_etudiant, prenom_etudiant
FROM INTERMEDIAIRE;
INSERT INTO MODUL SELECT DISTINCT id_module, code, ue,intitule_module,
id_enseignant FROM INTERMEDIAIRE;
INSERT INTO EVALUATION SELECT DISTINCT nom_evaluation, date_evaluation, note
FROM INTERMEDIAIRE;
```

2) Présentation commentée de deux requêtes intéressantes sur la base de donnée

Première requête
Retourne le nom et prénom des personnes qui sont en dessous de la moyenne, on constate qu'au moins chaque étudiant à une fois une note inférieur à 10.

```
SELECT DISTINCT nom_etudiant, prenom_etudiant FROM ETUDIANT
JOIN EVALUATION ON nom_etudiant = id_etudiant
WHERE note < 10;
```

Deuxième requête
Retourne le nom et prénom de tous les enseignants mais qu'une seule fois, il n'y a pas de doublon.

```
SELECT DISTINCT nom_enseignant, prenom_enseignant FROM ENSEIGNANT
```