BENALI Insaf (20221680)
GUINDO Aissata (20225308)

Rapport SQL

Liste des entités (leurs attributs et clés primaires) (cf : MCD)

PK = primary key

Étudiant

num_etudiant INTEGER: [PK] ,
nom_etud VARCHAR(50) ,
prenom_etud VARCHAR(50),
date_naiss_etud DATE,
age NUMERIC (5,2),
nationalité VARCHAR(20) ,
mail_etud VARCHAR(50) ,
telephone_etud INTEGER ,

Enseignant

Id_ens VARCHAR (50) : [PK],
nom_ens VARCHAR(50) ,
prenom_ens VARCHAR(50),
date_naiss_ens DATE ,
age NUMERIC (5,2),
mail_ens VARCHAR (50),
telephone_ens INTEGER ,

Cours

code_cours VARCHAR(50) : [PK],
nom_cours VARCHAR (50),
Semestre SMALLINT,

departement VARCHAR(50) ,

Examen

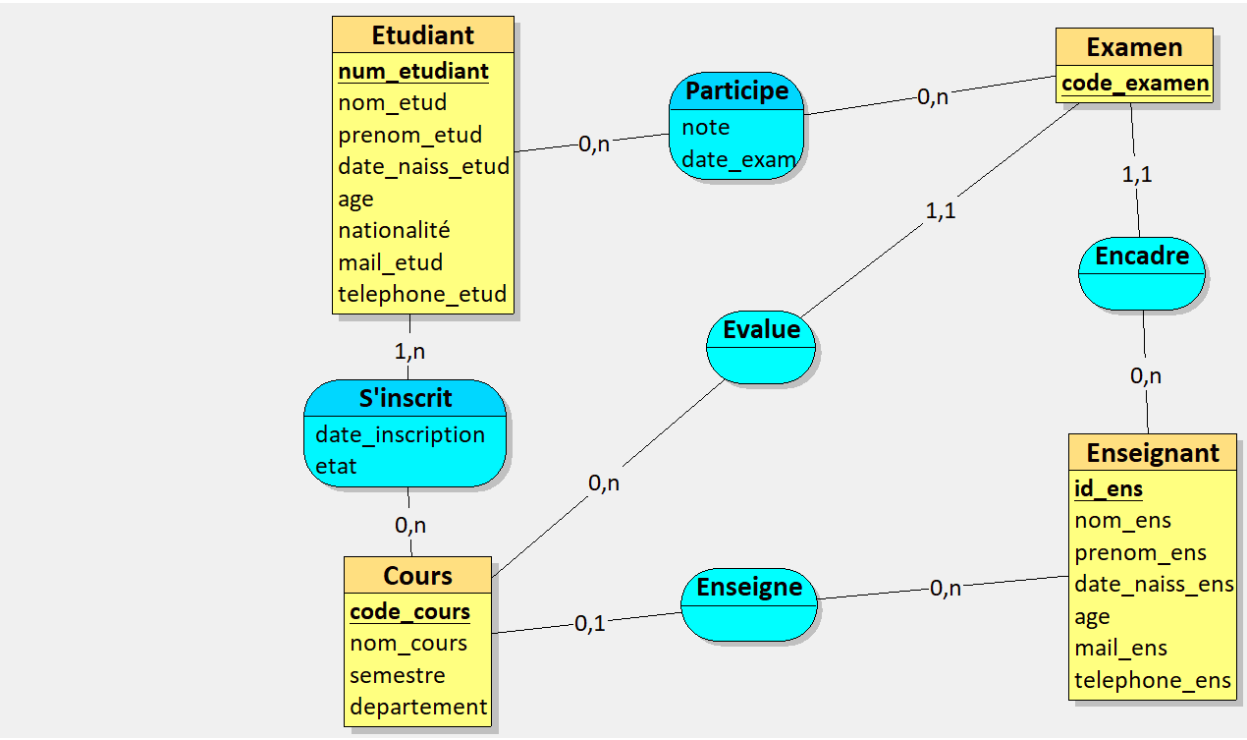
code_examen VARCHAR(50) :[PK] ,

S’inscrit (Table d'association entre Étudiant et Cours)

date_inscription DATE ,
etat BOOLEAN ,

Participe (Table d'association entre Étudiant et Examen)

note NUMERIC (5,2),
date_exam TIMESTAMP,



Liste des relations avec cardinalités et clés étrangères

1. Étudiant - Cours (Table **d'association s'inscrit**) :

   - Cardinalité : 1.n pour Étudiant

   - Cardinalité : 0.n pour Cours

   Motivation :

   - Un étudiant doit forcément avoir une inscription, mais peut en avoir plusieurs.

   - Un cours peut être inscrit par aucun étudiant ou plusieurs étudiants.

   ❖ **Clés étrangères dans la table s'inscris** :

   - num_etudiant : Etudiant.num_etudiant

   - code_cours : Cours.code_cours

2. Étudiant - Examen (Table **d'association Participation**)

   - Cardinalité : 0.n pour Étudiant

   - Cardinalité : 0.n pour Examen

   Motivation :

   - On a la possibilité qu'un étudiant ne fasse aucun examen, mais il peut également faire plusieurs examens.

   - Un examen peut être fait par aucun ou plusieurs étudiants.

❖ **Clés étrangères dans la table Participe :**

- num_etudiant : Etudiant.num_etudiant
- code_examen : Examen.code_examen

3. Enseignant - Examen (association **'Encadre'**)

- Cardinalité : 1.1 pour Examen
- Cardinalité : 0.n pour Enseignant

Motivation :

- Un examen doit être encadré par au moins un enseignant et un seul (l'enseignant qui enseigne le cours)
- Un enseignant peut encadrer aucun examen (cas où l'enseignant n'enseigne aucun cours) mais il peut aussi enseigner plusieurs examens (cas où l'enseignant enseigne plusieurs cours).

❖ **Clés étrangères dans la table Examen :**

- Id_ens : Enseignant.id_ens

4. Cours- Examen (association '**Evalue**')

- Cardinalité : 1.1 pour Examen
- Cardinalité : 0, n pour Cours.

Motivation :

- Un examen avec un id donné évalue au moins 1 cours et un seul.

- Parcours peut être évalué par aucun examen (cas où seule la présence constitue une note finale ou cas où le cours est évalué par un projet à rendre) mais aussi par plusieurs examens (cas de contrôle continue par exemple ou cas d'examen de 1ere session et 2eme session).

❖ **Clés étrangères dans la table Examen :**

- Code_cours : Cours.code_cours

5. Cours- Enseignant (association '**Enseigne**')

- Cardinalité : 0.1 pour Cours

- Cardinalité : 0.n Enseignant

Motivation :

- Un cours peut ne pas être enseigné par un enseignant (cas où on n'a pas encore trouvé de chargé de cours) et un cours peut être enseigné par au plus un enseignant.

- Un enseignant peut enseigner aucun cours comme il peut enseigner plusieurs cours.

❖ **Clés étrangères dans la table Cours :**

- Identifiant : Enseignant.identifiant.

Liste des entités avec leurs attributs ,clés primaires et clés étrangères (cf : MLD)

PK: primary key
FK: foreign key

Étudiant

num_etudiant INTEGER : [PK] ,
nom_etud VARCHAR(50) ,
prenom_etud VARCHAR(50),
date_naiss_etud DATE,
age NUMERIC (5,2) ,
nationalité VARCHAR(20) ,
mail_etud VARCHAR(50) ,
telephone_etud INTEGER ,

Enseignant

Id_ens VARCHAR (50) : [PK],
nom_ens VARCHAR(50) ,
prenom_ens VARCHAR(50),
date_naiss_ens DATE ,
age NUMERIC (5,2),
mail_ens VARCHAR (50),
telephone_ens INTEGER ,

Cours

code_cours VARCHAR(50) : [PK] ,
nom_cours VARCHAR (50),
Semestre SMALLINT ,
departement VARCHAR(50) ,
id_ens VARCHAR(50) : [FK],

Examen

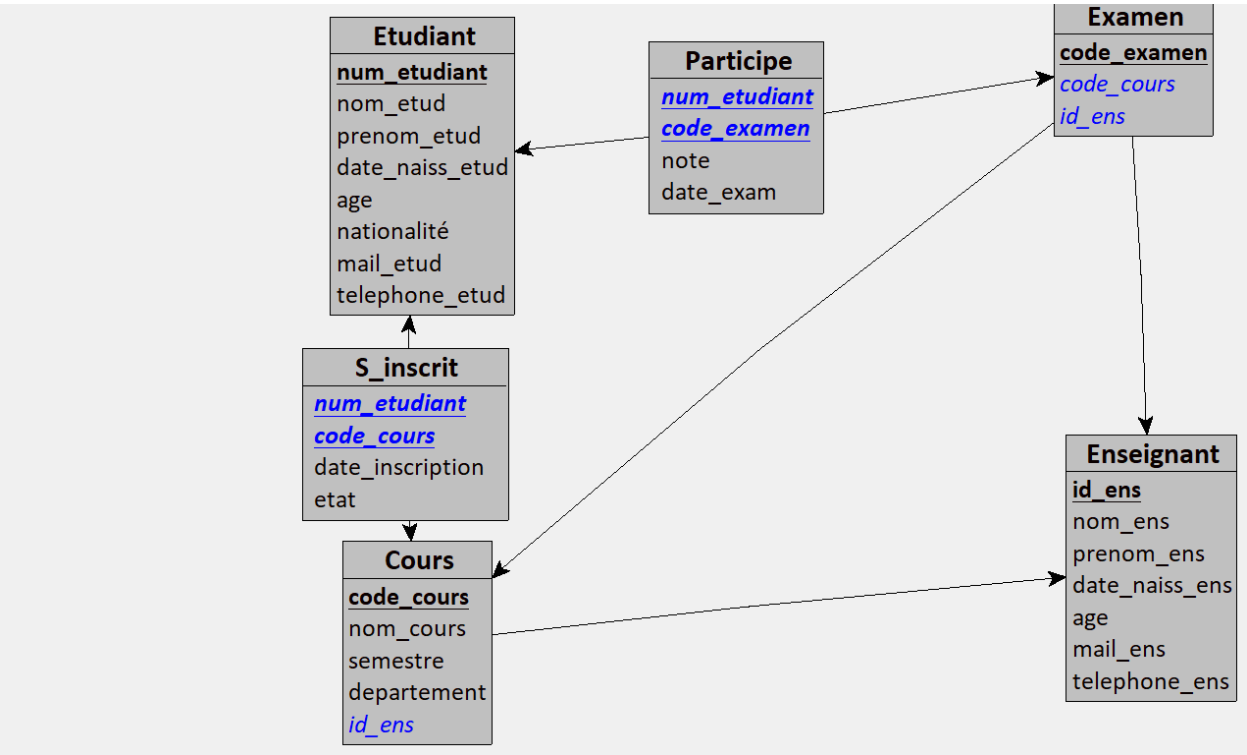
code_examen VARCHAR(50) : [PK] ,
code_cours VARCHAR(50) :[FK] ,
id_ens VARCHAR(50) : [FK] ,

S’inscrit (Table d'association entre Étudiant et Cours)

date_inscription DATE ,
etat BOOLEAN ,
num_etudiant INTEGER :[FK],
code_cours VARCHAR :[FK] ,

Participe (Table d'association entre Étudiant et Examen)

note NUMERIC (5,2),
date_exam TIMESTAMP,
num_etudiant INTEGER : [FK],
code_examen VARCHAR : [FK],



Décisions de modélisation

1. Les relations many-to-many exemple:

   Étudiant ↔ Cours
   Étudiant ↔ Examen
   Nécessitent des tables d'association pour inclure des attributs supplémentaires (date_inscription, état, note, date d'examen).
   Ces tables d'associations contiennent les clés primaires des deux tables qu'elles relient et ces clés deviennent des clés étrangères dans la table d'association.

2. On a mis « note » dans la classe d'association **'Participe'** entre Etudiant et Examen car à chaque fois qu'un étudiant participe à un examen il obtient une note à cet examen donc la liaison entre Etudiant et Examen génère forcément une note et cela se passe à une date donnée (date_exam).

3. Bien que l'inscription soit un sujet clé de ce projet on n'a pas de table inscription, on a plutôt mis une classe d'association 's'inscris' entre Etudiant et Cours (dans laquelle on a spécifié la date de l'inscription et l'état de l'inscription). L'étudiant est lié à un Cours par une inscription qui se passe à une date donnée. On a mis l'attribut **'état '** (BOOLEAN) pour pouvoir différencier les étudiant dont l'inscription est toujours en cours (état = true) des étudiants dont l'inscription a expiré (état = false) (ex : était inscris en 2021 mais n'est plus inscris).

4. On a mis une relation entre 'Cours' et 'Examen' « **Evalue** » car si un examen existe alors il évalue un cours et cela nous permet par exemple d'avoir le nombre total d'examen pour chaque cours.

5. Il y a une relation entre ‘Cours’ et ‘Enseignant’ « **Enseigne** » car un enseignant enseigne un cours et un cours est enseigné par un enseignant.
   Les deux sont donc forcément liés et cela nous permet de savoir quel enseignant enseigne un cours et également savoir un cours est enseigné par quel enseignant.

6. Il y a une association entre **‘Examen’** et **‘Enseignant** ‘ « **Encadre** » car un examen a toujours un encadrant et cette association nous permet de trouver l’enseignant en charge d’un examen donné.

Création de la base (Requête SQL):

```
CREATE TABLE Etudiant (
   num_etudiant INTEGER,
   nom_etud VARCHAR(50)  NOT NULL,
   prenom_etud VARCHAR(50) ,
   date_naiss_etud DATE NOT NULL,
   age NUMERIC (5,2)   NOT NULL,
   nationalité VARCHAR (20)  NOT NULL,
   mail_etud VARCHAR (50) ,
   telephone_etud INTEGER,
   PRIMARY KEY(num_etudiant)
);

CREATE TABLE Enseignant(
   id_ens VARCHAR(50) ,
   nom_ens VARCHAR(50)  NOT NULL,
   prenom_ens VARCHAR(50)  NOT NULL,
   date_naiss_ens DATE NOT NULL,
   age NUMERIC(5,2)  ,
   mail_ens VARCHAR(50) ,
```

```
  telephone_ens INTEGER,
  PRIMARY KEY(id_ens)
);

CREATE TABLE Cours(
  code_cours VARCHAR(50) ,
  nom_cours VARCHAR(50)  NOT NULL,
  semestre SMALLINT NOT NULL,
  departement VARCHAR(50) ,
  id_ens VARCHAR(50) ,
  PRIMARY KEY(code_cours),
  FOREIGN KEY(id_ens) REFERENCES Enseignant(id_ens)
);

CREATE TABLE Examen(
  code_examen VARCHAR(50) ,
  code_cours VARCHAR(50)  NOT NULL,
  id_ens VARCHAR(50)  NOT NULL,
  PRIMARY KEY(code_examen),
  FOREIGN KEY(code_cours) REFERENCES Cours(code_cours),
  FOREIGN KEY(id_ens) REFERENCES Enseignant(id_ens)
);

CREATE TABLE Participe(
  num_etudiant INTEGER,
  code_examen VARCHAR(50) ,
  note NUMERIC(5,2)  ,
  date_exam TIMESTAMP NOT NULL,
  PRIMARY KEY(num_etudiant, code_examen),
  FOREIGN KEY(num_etudiant) REFERENCES Etudiant(num_etudiant),
  FOREIGN KEY(code_examen) REFERENCES Examen(code_examen)
);

CREATE TABLE S_inscrit(
  num_etudiant INTEGER,
```

```
    code_cours VARCHAR(50) ,
    date_inscription DATE NOT NULL,
    etat BOOLEAN NOT NULL,
    PRIMARY KEY(num_etudiant, code_cours),
    FOREIGN KEY(num_etudiant) REFERENCES Etudiant(num_etudiant),
    FOREIGN KEY(code_cours) REFERENCES Cours(code_cours)
);
```