Elise METAYER
Fanny SHEHABI

ESIR1 - INFO

ECOLE SUPERIEURE D'INGENIEURS DE RENNES

TP5 - BASES DE DONNEES

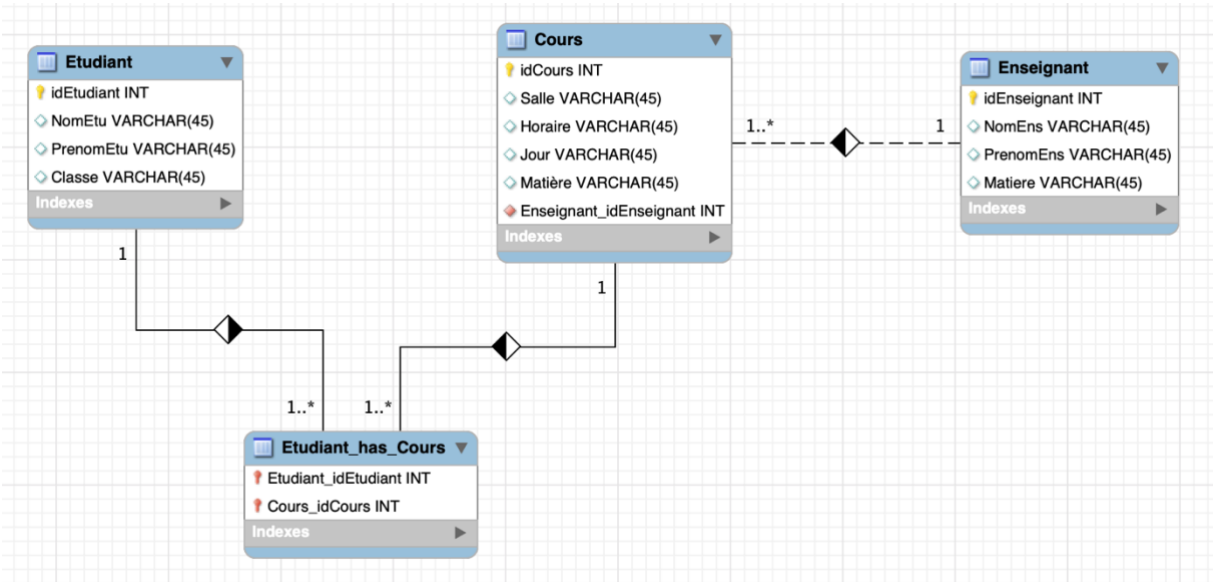
Dans ce TP, nous allons réaliser des projets de conception grâce à l'outil MySQLWorkbench.

I. Exercice 1 : Gestion d'emploi du temps

Analyse des besoins : nous avons besoin de faire une base de données pour pouvoir gérer les emplois du temps des salles, des élèves et des professeurs. Il faut pourvoir accéder à ces emplois du temps pour une heure donnée et éviter les doublons. En effet un professeur ne peut pas donner deux cours différents à une même heure (par exemple).

Nous allons modéliser les problèmes suivants : il est possible que plusieurs groupes d'une seule ou de plusieurs filières assiste à un même cours. Il est possible de choisir la salle en fonction du nombre d'élève et du type de cours (TD, TP, CM).

Nous avons réalisé différentes versions de notre schéma. La première version est extrêmement simpliste étant donné qu'elle ne contient que les éléments les plus importants.

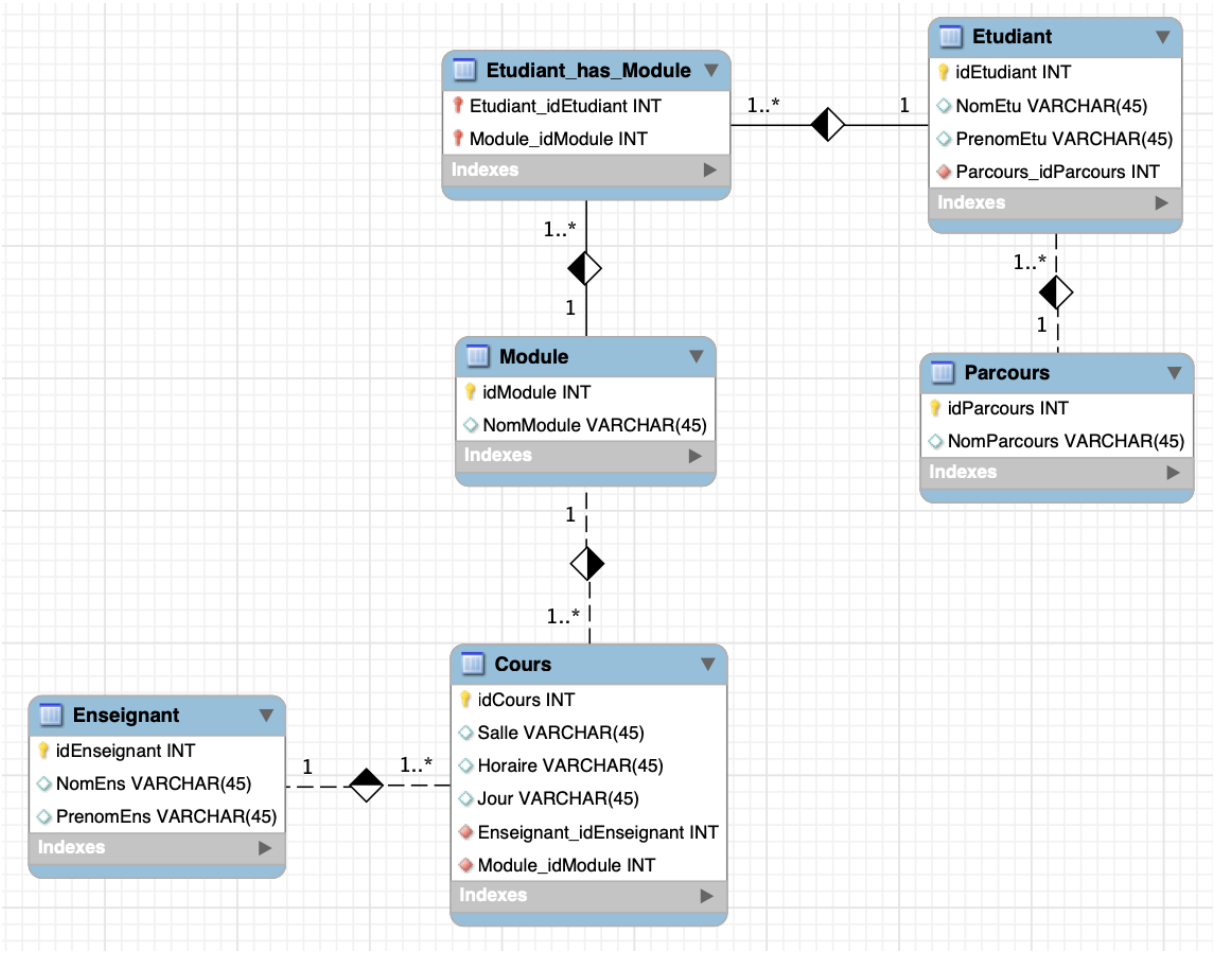


Version 1 : Schéma avec les éléments les plus importants

La table Cours est un élément central de notre schéma. On le voit dès la première version. C'est cette table qui va contenir, la salle, l'horaire, le jour, la matière ainsi que les identifiants de certaines autres tables.

Cette première version contient également les tables Etudiant et Enseignant qui sont des éléments essentiels de notre base de données.

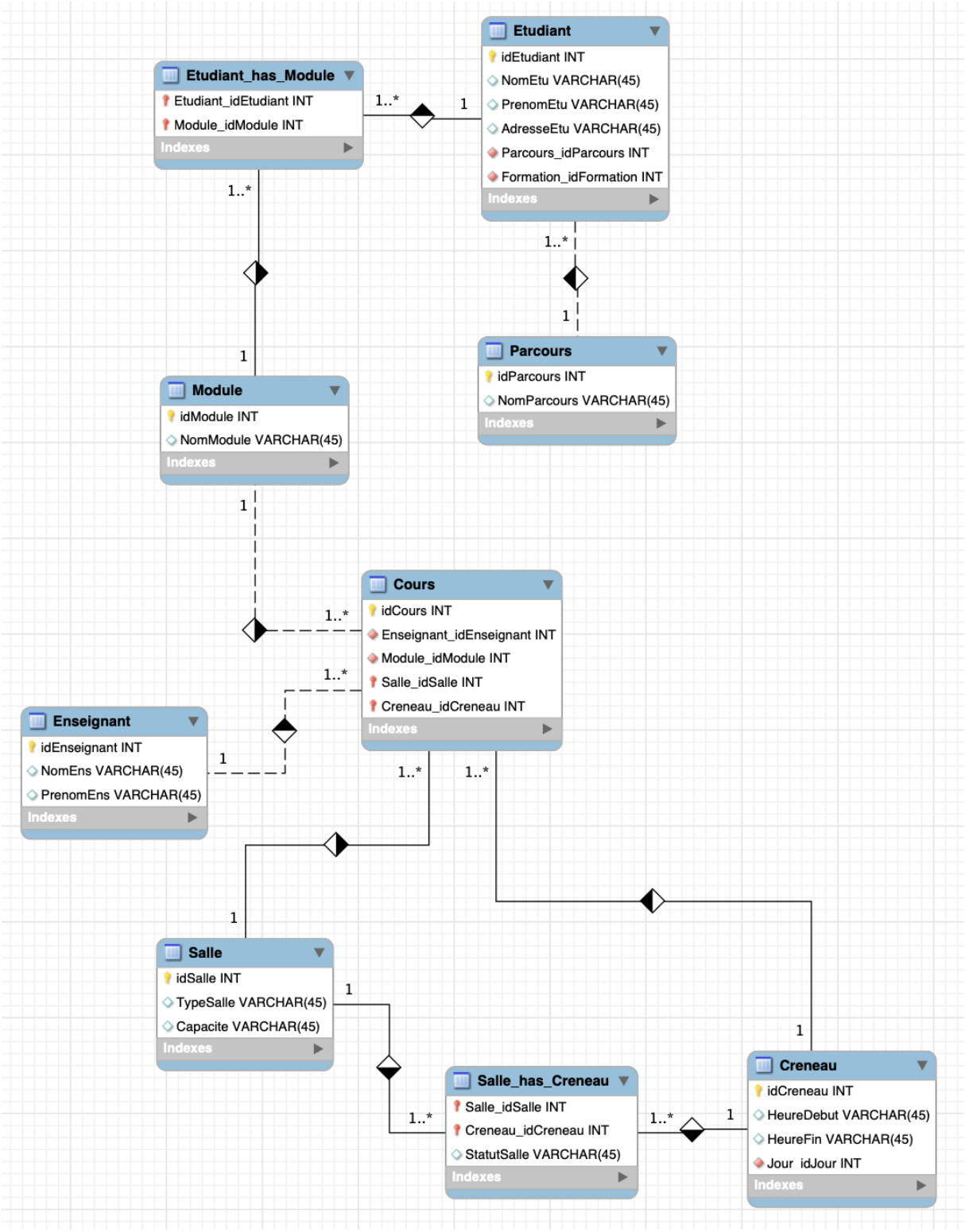
Nous avons ensuite détaillé notre schéma afin de pouvoir réaliser plus de choses :



Version 2 : Plus détaillée

Cette version permet, entre autres choses, de lier un étudiant à un parcours spécifique mais aussi de lier les étudiants aux modules via la table « Etudiant_has_Module ». On peut alors obtenir la liste des étudiants d'un parcours qui suivent un module ou un cours spécifique. Par ailleurs, cette version permet également que des étudiants de différents parcours suivent un même module.

Un étudiant suit effectivement un parcours et un parcours peut être suivi par plusieurs étudiants. Un module est suivi par plusieurs étudiants (de différents parcours) et un étudiant peut suivre plusieurs modules.



Version 3 : Version Finale

On remarque que la table Cours est toujours centrale dans le schéma. On a ajouté les tables Salle, Créneau, ainsi que Salle_has_Creneau au schéma de la version 2.

La table Salle possède alors l'attribut TypeSalle qui nous sera, par exemple, utile pour lister les salles de TP. La table Salle_has_Creneau possède l'attribut StatutSalle qui nous permettra, entre autres choses, de lister les salles libres.

Un cours a lieu dans une seule salle et pendant un seul créneau mais une salle peut être utilisée dans plusieurs cours (à des créneaux différents). De même, plusieurs cours peuvent avoir lieu sur un même créneau dans des salles différentes.

Cette version détaille davantage l'aspect temporel grâce à l'ajout de la table Créneau.

Cette version finale n'est cependant pas parfaite, en effet nous avons décidé d'ignorer les problèmes qui seraient liés au volume horaire (nous n'avons pas de durée dans nos classes).

Résumons la version finale :

- Enseignant(idEnseignant, NomEns, PrenomEns)
- Module(idModule, NomModule)
- Salle(idSalle, TypeSalle, Statut, Capacité)
- Jour(idJour)
- Creneau(idCreneau, HeureDebut, HeureFin, Jour)
- Module(idModule, NomModule)
- Parcours(idParcours, NomParcours)
- Etudiant(idEtudiant, NomEtu, PrenomEtu, Parcours_idParcours)
- Etudiant_has_Module(Etudiant_idEtudiant, Module_idModule)
- Cours(idCours, Salle_idSalle, Enseignant_idEnseignant, Creneau_idCreneau, Module_idModule)
- Salle_has_Creneau(Salle_idSalle, Creneau_idCreneau, StatutSalle)

II. EXERCICE 2 : Base de données de références bibliographiques

Analyse des besoins : nous avons besoin de faire une base de données pour pouvoir gérer les références bibliographiques des livres données par les enseignants au début des cours. Il faut pourvoir accéder aux collections par année d'étude ou par parcours et pouvoir donner un degré d'importance à chaque livre en fonction du nombre de fois où il est proposé, en effet plusieurs professeurs peuvent proposer le même livre.

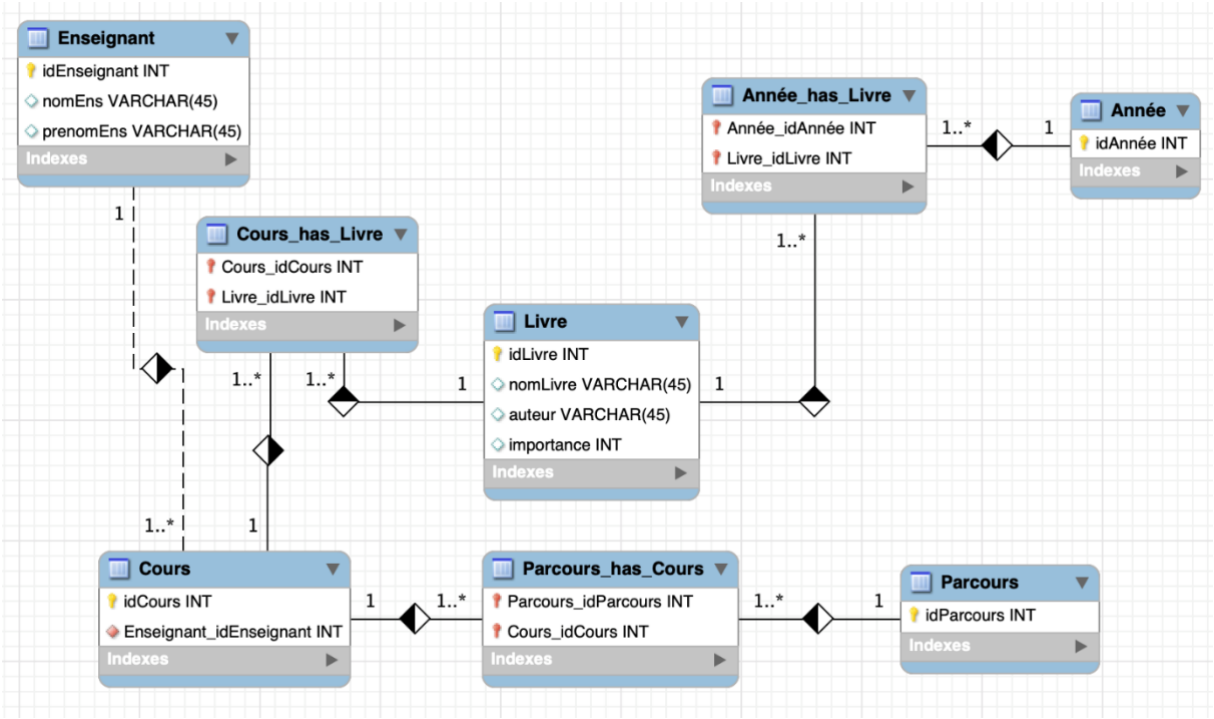


Schéma base de données 'Références bibliographiques'

Résumons le schéma :

- Enseignant (idEns, nomEns, prenomEns)
- Cours (idCours, #Enseignant_idEnseignant)
- Livre (idLivre, nomLivre, auteur)
- Parcours (idParcours)
- Annee (idAnnee)
- Cours_has_Livre (#Cours_idCours, #Livre_idLivre)
- Parcours_has_Cours (#Parcours_idParcours, #Cours_idCours)
- Annee_has_Livre (#Annee_idAnnee, #Livre_idLivre)

«

Afin de gérer l'importance d'un livre qui serait cité dans plusieurs cours, nous avons ajouté un attribut 'importance' à la table Livre. Avant l'insertion des données dans les tables, nous avons créé un TRIGGER qui met à jour cet attribut après insertion dans la table Cours_has_Livre.

Grâce aux tables Cours_has_Livre et Annee_has_Livre, il est alors possible de connaître la collection de livre proposée par année d'étude ou par parcours (cf ficher `requetes.sql`).

III. EXERCICE 3 : Pour aller plus loin

Tout d'abord, on veut créer un déclencheur qui ajoute la date de chaque modification dans la table Livre. Pour cela, nous avons créé une table Modification dans laquelle nous insérons la date de chaque modification. (voir fichier `bdd_ex2.sql`)

Enfin, on souhaite créer une vue qui contient tous les livres proposés. Afin que celle-ci soit actualisée au fur et à mesure des insertions dans la table Livre, nous avons également créer un déclencheur. (voir fichier `bdd_ex2.sql`)