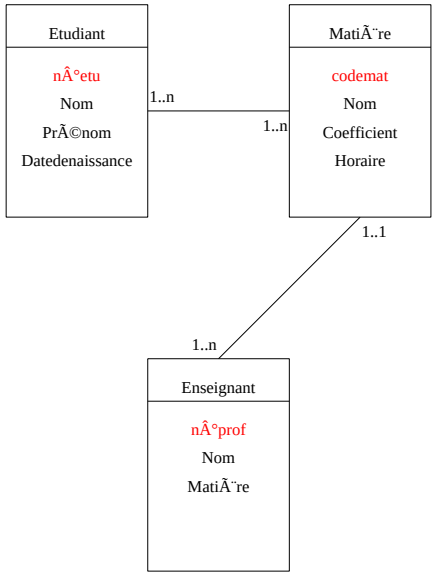
DS d'informatique n°2 : corrigé

PTSI Lycée Eiffel

27 mars 2015

Exercice 1

1. Non, un professeur ne peut enseigner qu'une seule matière, puisque l'attribut matière dans la table Enseignant ne peut prendre qu'une seule valeur. Oui, un étudiant peut avoir des notes dans plusieurs matières, la table Notes admet pour clé primaire le couple (n°etu, codemat), et un même étudiant peut donc apparaitre associé à plusieurs matières dans cette table. Par contre, il ne peut avoir qu'une seule note dans chaque matière.
2. Aucune raison de créer quatre entités, trois suffisent puisque la relation Notes provient d'une association entre les entités Etudiant et Matières.



3. CREATE TABLE GroupeTD (n°groupe : integer, matiere : varchar, horaire : varchar, salle : integer)
4. En supposant qu'un étudiant appartient à un seul et unique groupe de TD, il suffit d'ajouter un attribut dans la relation Etudiant contenant le groupe de TD de l'étudiant (ce sera alors une clé secondaire de cette table).
5. (a) SELECT Nom, Prénom, Datedenaissance FROM Etudiant ORDER BY Nom
   (b) SELECT Nom FROM Enseignant WHERE Matière='Mathémétiques'
   (c) SELECT Nom FROM Matières WHERE Coefficient >=3 AND Horaire >=5
   (d) SELECT MAX(note) FROM Notes,Matière WHERE Nom='Informatique' AND Notes.codemat=Matières.codemat

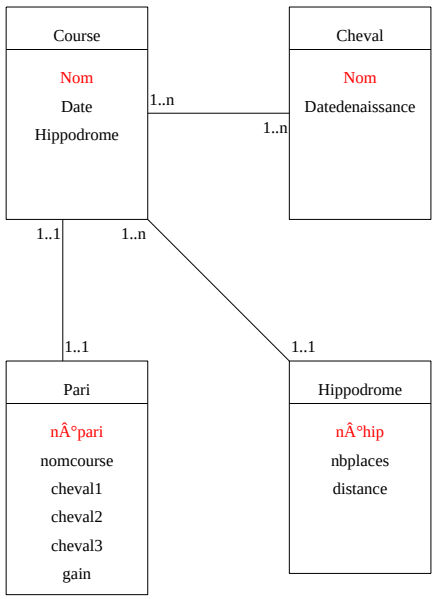
(e) SELECT SUM(Coefficient*note)/COUNT(note) FROM Etudiant, Matière, Note WHERE Etudiant.n°etu=Notes.n°etu AND Matière.codemat=Notes.codemat AND Etudiant.Nom='Durand' AND Etudiant.Prénom='François'

(f) SELECT Nom FROM Matière,Notes WHERE Matière.codemat=Notes.codemat AND (SELECT COUNT(note) FROM Matières,Notes WHERE Matières.codemat=Notes.codemat AND ? ? ?) >=50

En fait, je dois admettre que trouver une condition qui ne garde que les matières apparaissant 50 fois dans la table est hors de portée avec ce qu'on a vu en SQL.

Exercice 2

1.



Ce schéma ne contient pas toutes les données concernant chaque course (liste des chevaux participants etc), c'est volontaire ! En effet, ces données sont cachées dans l'association entre l'entité Course et l'entité Cheval, qui donnera lieu à une cinquième relation dans le modèle relationnel, qui contiendra les données correspondantes.

2. On va donc créer cinq relations. On considère que l'attribut Hippodrome de la relation Course correspond au n°hip de la relation Hippodrome, et que les attributs cheval1, cheval2 et cheval3 de la relation Pari correspondent aux numéros des chevaux correspondants dans la course concernée par le pari :
   - Course (Nom : varchar, Date : date, Hippodrome : integer)
   - Cheval (Nom : varchar, Datedenaissance : date)
   - Hippodrome (n°hip : integer, nbplaces : integer, distance : integer)
   - Pari (n°pari : integer, nomcourse : varchar, cheval1 : integer, cheval2 : integer, cheval3 : integer, gain : integer)
   - Participants (nomcourse : varchar, nomcheval : varchar, numéro : integer, cote : varchar, classement : integer)

   Il est sous-entendu que la clé primaire de la relation Participants est constituée du couple (nomcourse, nomcheval).

3. (a) $\pi_{n^\circ hip,distance}(Hippodrome)$
   (b) $\pi_{Nom}(\sigma_{Datedenaissance>=2005}(Cheval))$.
   (c) $\pi_{nomcheval,classement}(\sigma_{nomcourse='Prixd'Amrique}(Participants))$
   (d) $\pi_{nomcourse}(\sigma_{gain>=50}(Pari))$
   (e) $\pi_{Nom,Date,Hippodrome}((Pari \bowtie_{Pari.nomcourse=Partipants.nomcourse,*} Participants) \bowtie_{nomcourse=Nom} Course)$, la condition supplémentaire * étant horrible à écrire dans l'algèbre relationnelle puisqu'on doit avoir $cheval1 = \pi_{nomcheval}(\sigma_{classement=1}(Participants))$, puis de même pour les deux autres chevaux. Notons que ce serait à peine plus court d'écrire une requête SQL effectuant la même chose...