TP BDA (Séance n° 1)
Vues et Privilèges d'accès avec Rappels de SQL et PLSQL

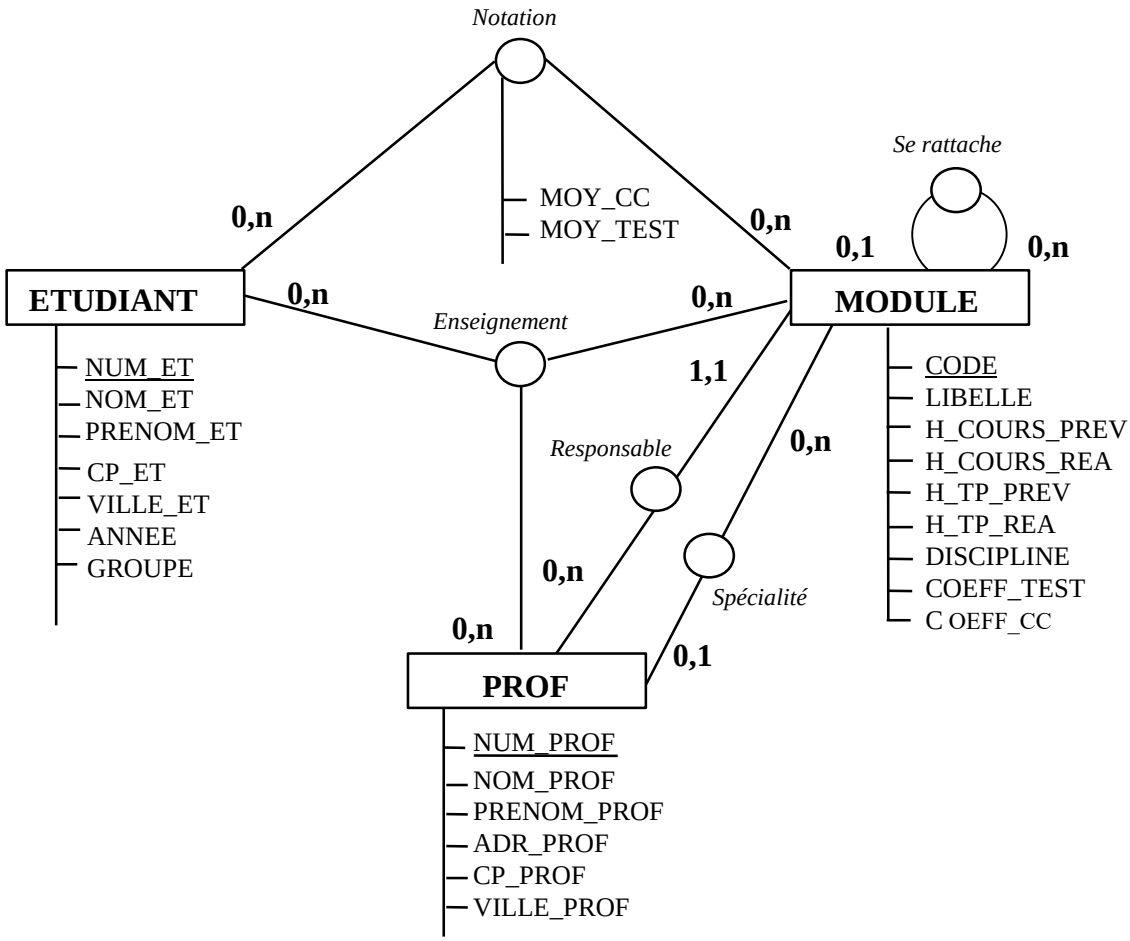
Préambule : *Rappel de la Base de données exemple*

La BD exemple, sur laquelle vous allez travailler, a déjà été créée. Elle permet d'effectuer la gestion pédagogique (simplifiée) du département Informatique de l'IUT. Elle a été élaborée à partir du dictionnaire des données suivant :

NOM	LIBELLE	DOMAINE[1]	CI
NUM_ET	Numéro d'un étudiant	D_NUM_ET : Number(6,0)	valeurs uniques
NOM_ET	Nom d'un étudiant	D_NOM_ET : Varchar2(20)	
PRENOM_ET	Prénom d'un étudiant	D_PRENOM_ET : Varchar2(15)	
CP_ET	Code postal d'un étudiant	D_CP : Varchar2(5)	
VILLE_ET	Ville d'un étudiant	D_VILLE : Varchar2(20)	
ANNEE	Année d'étude d'un étudiant	D_ANNEE : Number(2,0)	valeurs dans [1...2]
GROUPE	Numéro de groupe d'un étudiant	D_GROUPE : Number(1,0)	valeurs dans [1...6]
NUM_PROF	Numéro d'un professeur	D_NUM_PROF : Number(3,0)	valeurs uniques
NOM_PROF	Nom d'un professeur	D_NOM_PROF : Varchar2(30)	
PRENOM_PROF	Prénom d'un professeur	D_PRENOM_PROF : Varchar2(20)	
ADR_PROF	Adresse d'un professeur	D_ADR	
CP_PROF	Code postal d'un professeur	D_CP	
VILLE_PROF	Ville d'un professeur	D_VILLE	
CODE	Code d'un module	D_CODE : Varchar2(6)	valeurs uniques
LIBELLE	Libellé d'un module	D_LIBELLE : Varchar2(30)	
H_COURS_PREV	Nombre d'heures de cours Prévues pour un module	D_NB_H : Number(3,0)	
H_COURS_REA	Nombre d'heures de cours Réalisées pour un module	D_NB_H	
H_TP_PREV	Nombre d'heures de TP Prévues pour un module	D_NB_H	
H_TP_REA	Nombre d'heures de TP Réalisées pour un module	D_NB_H	
DISCIPLINE	Discipline enseignée	D_DISCIPLINE : Varchar2(15)	valeurs parmi : {Langues, Math, Informatique…}
COEFF_CC	Coefficient du contrôle continu Pour un module (en %)	D_COEFF : Number(3,0)	valeurs dans [0..100]
COEFF_TEST	Coefficient du test pour Un module (en %)	D_COEFF	valeurs dans [0...100]
CODEPERE	Code du module père d'un module	D_CODE : Varchar2(6)	
MOY_CC	Moyenne de contrôle continu pour un étudiant dans un module	D_NOTE : Number(2,0)	valeurs dans [0...20]
MOY_TEST	Moyenne de test pour un étudiant dans un module	D_NOTE	valeurs dans [0...20]

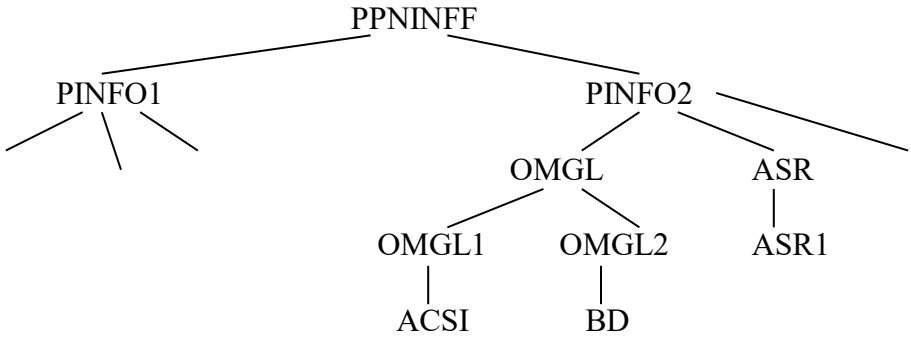
[1] Les types syntaxiques, utilisés pour la description des domaines, sont disponibles dans Oracle.

A partir du dictionnaire des données, le MCD, établi selon le modèle Entité/Association, est le suivant :



On suppose que pour chaque module, il existe un et un seul professeur responsable de la coordination des enseignements dans la module (TA Responsable). De même, on suppose que chaque professeur peut avoir une spécialité (correspondant à une matière enseignée). S'il en a plusieurs, on se contente de stocker sa spécialité principale (TA Spécialité).

Le découpage pédagogique du programme du DUT est représenté via le TA unaire Se rattache. Plus précisément, le programme est vu comme une hiérarchie : le PPN est la racine de cet arbre. Ce PPN se décompose en modules, eux mêmes se décomposant en sous-modules et ainsi de suite. Les feuilles de cet arbre correspondent aux matières enseignées. Par exemple, le programme pédagogique du DUT pourrait être (partiellement) représenté par l'arborescence suivante :



Le programme pédagogique (PPNINF) se décompose en deux, le programme de première année

(PINFO1) et celui de deuxième année (PINFO2). Ce dernier se découpe en différents modules (OMGL, ASR…). Le module OMGL se décompose à nouveau en plusieurs modules (OMGL1, OMGL2…). Dans un souci de simplification, on considère que les modules, quel que soit leur niveau dans la hiérarchie, sont décrits exactement par les mêmes attributs (et sont donc **stockés dans le même TE** MODULE). ***On appelle matière des modules qui sont feuilles dans la hiérarchie. Donc seules les entités du TE MODULE correspondant à des matières participent aux TA Enseignt et Notation.***

Le schéma relationnel normalisé de la base exemple, dérivé du MCD précédent, est donné ci-après. Par convention les clefs primaires sont soulignées et les clefs étrangères sont indiquées en gras et en italique.

ETUDIANT (NUM_ET, NOM_ET, PRENOM_ET, CP_ET, VILLE_ET, ANNEE, GROUPE)

PROF (NUM_PROF, NOM_PROF, PRENOM_PROF, ADR_PROF, CP_PROF, VILLE_PROF, ***MAT_SPEC***)

MODULE (CODE, LIBELLE, H_COURS_PREV, H_COURS_REA, H_TP_PREV, H_TP_REA, DISCIPLINE, COEFF_TEST, COEFF_CC, ***RESP***, ***CODEPERE***)

ENSEIGNT (***CODE, NUM_PROF, NUM_ET***)

NOTATION (***NUM_ET, CODE***, MOY_CC, MOY_TEST)

Remarque :
Les attributs RESP dans MODULE et MAT_SPEC dans PROF représentent respectivement le numéro d'un professeur responsable d'une matière et le code d'une matière dont un professeur est spécialiste. Ils sont tous deux clefs étrangères et sont associés aux clefs primaires NUM_PROF et CODE. La clef étrangère CODEPERE dans la relation MODULE permet de traduire le TA Se rattache et fait référence à la clef primaire CODE. Les autres clefs étrangères portent le même nom que les clefs primaires associées.

Au niveau des valeurs saisies dans la BD, Oracle fait une distinction entre minuscules et majuscules. Par contre les mots clefs SQL, les noms d'attributs ou de relations peuvent être indifféremment tapés en minuscules ou majuscules.

Par convention, toutes les DONNEES déjà saisies dans la BD IUT l'ont été en majuscules !!

De plus considérons les tables systèmes suivant du dictionnaire de données ORACLE :

USER_TABLES (TABLE_NAME, TABLESPACE_NAME, ...)
USER_TAB_COLUMNS (TABLE_NAME, COLUMN_NAME, DATA_TYPE, DATA_LENGTH)
USER_CONS_COLUMNS (TABLE_NAME, COLUMN_NAME, CONSTRAINT_NAME,...)
USER_CONSTRAINTS (TABLE_NAME, CONSTRAINT_NAME, CONSTRAINT_TYPE, SEARCH_CONDITION, …)
USER_OBJECTS (OBJECT_NAME, OBJECT_TYPE…)

La valeur de l'attribut CONSTRAINT_TYPE peut être 'P' (Primary), 'R' (References) ou 'C'(Check)
La valeur de l'attribut OBJECT_TYPE peut être TABLE, VIEW, FUNCTION, PROCEDURE, TRIGGER, …

Première étape : Consultation des tables Systèmes

Q1 Quel est le nom de tous les attributs de la relation PROF ?

Q2 Donnez la liste des tables, fonctions et des procédures stockées de la base IUT ?

Q3 Quelles sont toutes les contraintes d'intégrité définies sur les relations de données ?

Q4 Retrouvez le nom des contraintes définies sur toutes les relations de la base IUT ayant un attribut de type NUMBER.

Q5 Retrouvez le nom des contraintes et le nom de l'attribut de type NUMBER sur lequel elles portent pour toutes les relations de la base IUT.

Q6 Retrouvez le nom des contraintes, le nom de l'attribut sur lequel elles portent et le type de cet attribut, pour toutes les relations de la base IUT ayant un attribut de type NUMBER.

Q7 Trouvez le nom et la spécification des contraintes de domaine dans la base IUT.

Q8 Quels sont les attributs portant le même nom dans des relations différentes de la base ?
Affichez nom attribut 1, nom relation 1, nom attribut 2, nom relation 2.

Q9 Comment formuler la requête précédente en évitant que les couples soient affichés deux fois ((A1, A2) et (A2, A1)) ?

Deuxième étape : Création et manipulation de vues

Les vues à créer ont un double objectif : assurer la prise en compte des contraintes d'intégrité de la base et également limiter la consultation des données pour certains utilisateurs.

Q10 Créez une vue appelée PROF_INFO2 donnant le numéro, le nom et le prénom des professeurs enseignant en deuxième année.
Puis consultez la vue par une requête d'interrogation.

Q11 Il s'agit de prendre en compte l'intégrité de domaine de l'attribut DISCIPLINE dans la relation MODULE. En effet, ces valeurs admissibles ne peuvent appartenir qu'à l'ensemble suivant : {Informatique, Gestion, Maths}. Créez une vue DIS, permettant d'assurer que toute insertion à travers cette vue vérifie la contrainte de domaine énoncée. Vérifiez l'opération réalisée en tentant une insertion invalide.

Q12 Dans la base actuellement définie, la contrainte suivante n'est pas vérifiée : "Le responsable d'une matière doit forcément enseigner cette matière". Comment prendre en compte cette contrainte d'intégrité dynamique par une vue ?

Q13 Définissez la vue nécessaire pour prendre en compte toutes les contraintes d'intégrité de référence mises en jeu lors d'une insertion dans la relation ENSEIGNT ?

Q14 Définissez la vue nécessaire pour prendre en compte toutes les contraintes d'intégrité de référence mises en jeu lors d'une suppression de matière dans la relation MODULE ?

Troisième étape : Gestion des privilèges d'accès aux données

Q15 Donnez, à tous, l'autorisation de consulter la relation ETUDIANT (sans aucun autre droit).

Q16 Effectuez la consultation de tous les étudiants résidant à Marseille, dans l'espace de travail d'un autre utilisateur. Vérifiez que vous n'avez que le droit de consultation, en tentant une opération de mise à jour sur cette relation.

Q17 Accordez le droit de consultation et de mise à jour des données de la relation PROF à un autre utilisateur, qui vous accordera les mêmes privilèges.
Consultez puis effectuez une modification sur la relation PROF de l'autre utilisateur. Ce dernier doit ensuite consulter sa propre relation PROF. Que constatez-vous ?

Quatrième étape : Développement en PL/SQL

Q18 Ecrire un bloc anonyme permettant d'afficher les codes et libellé des matières en gérant l'exception. Ecrire une version avec un curseur déclaré et une version avec un curseur non déclaré

Q19 Il s'agit de définir un programme PL/SQL permettant l'insertion automatique de tuples dans une relation GROUPE. On désire, en effet, pouvoir consulter à tout instant les effectifs des différents groupes de deuxième année. Or cette information est implicitement contenue dans la relation ETUDIANT. Un programme, baptisé 'ins_groupe', doit permettre de l'extraire et de réaliser les insertions correspondantes dans la relation GROUPE, préalablement créée.

Pour définir ce programme, ***il vous est demandé de ne pas utiliser de curseur mais de réaliser les étapes suivantes*** :

- Commencez par créer, en sqlplus, la structure de la relation GROUPE qui doit comporter seulement deux attributs : NUMERO, défini comme un entier sur une position et EFFECTIF dont le type syntaxique est NUMBER (3, 0).

- Définissez un bloc PL/SQL intégrant les requêtes SQL nécessaires et permettant :
  - De stocker dans une variable NB_GROUPE le nombre de groupes existant dans la relation ETUDIANT ;
  - Pour chaque valeur de groupe, de calculer l'effectif ;
  - D'insérer, dans la relation GROUPE, le tuple constitué du numéro du groupe et de son effectif calculé.

- Après l'exécution, contrôlez les résultats obtenus en consultant l'extension de la relation concernée. En phase de test, effectuez des **`ROLLBACK`**, pour défaire les opérations d'insertion réalisées.

- Terminez le programme 'ins_groupe', en en faisant une transaction, i.e. en insérant comme première et dernière instruction du bloc, l'ordre **`COMMIT`**. Testez l'exécution.

- Enfin, tenez compte dans 'ins_groupe' de la possibilité de n'avoir aucun tuple dans ETUDIANT.