L3 MIAGE – 2018/2019

Bases de Données

janvier 2019

Création de schéma et requêtes SQL

Exercice 1 :
La bibliothèque universitaire dispose d'une base de données permettant de conserver les informations liées aux livres. Un livre possède une référence, un titre, un éditeur et est écrit par un ou plusieurs auteurs. La bibliothèque peut acheter un livre parce qu'il est recommandé par un enseignant, pour une ou plusieurs matières. Le gestionnaire de la bibliothèque veut savoir pour quelles matières on a acheté chaque livre, et quels sont les enseignants qui ont recommandé l'achat d'un livre. Dans cette base, on mémorise donc des livres et leurs auteurs, des matières et des enseignants. Une matière est liée à un diplôme ; on utilise les codes diplômes connus dans toute l'université, un code diplôme étant un entier sur 6 chiffres. On donne un numéro (entier strictement positif) à chaque matière d'un diplôme. Ainsi, une matière est identifiée de manière unique par son code diplôme et son numéro. Pour un livre et une matière, on veut connaître le nom et prénom de l'enseignant qui a demandé d'acheter ce livre, et son adresse email ou son téléphone. Bien sûr, il n'est pas impossible que plusieurs enseignants conseillent le même livre, mais pour un livre et une matière donnés, on ne mémorisera qu'un seul enseignant. On veut absolument pouvoir contacter cette personne : on imposera qu'au moins l'une des deux informations `email` ou `telephone` soit connue.

Question 1.1 : Représentez le schéma de la base de données en utilisant un MCD ou un diagramme de classe UML.

Question 1.2 : Ecrivez (et testez en TP!) les ordres SQL permettant de créer les tables. N'oubliez pas de déclarer le maximum de contraintes, en particulier les clefs primaires et clefs étrangères.

Question 1.3 : En TP : insérez quelques données dans la base. Voici pour vous aider une liste de livres :

1. *SQL pour Oracle* de Christian Soutou et Olivier Teste. Editions Eyrolles
2. *Design Patterns, Tête la Première* de Eric et Elisabeth Freeman. Edition O'Reilly
3. *Les réseaux* de Guy Pujolle. Editions Eyrolles
4. *Réseaux et Télécoms* de Claude Servin. Editions Dunod
5. *Les Bases de données pour les nuls* de Fabien Fabre. Editions Générales First

Le premier et le dernier livre sont conseillés par Anne-Cécile Caron, pour "Bases de Données" en licence 3 MIAGE (code diplôme 123456), le dernier est conseillé par Bruno Bogaert pour "technologie du web" en Licence 2 informatique (code diplôme 123232). Le second livre est conseillé par Yves Roos pour "Conception Orientée Objet" en licence 3 MIAGE. Le troisième et le quatrième livre sont conseillés par Laurent Noé pour "Réseaux" en licence 3 MIAGE et licence 3 informatique (code diplôme 123789).

Question 1.4 : Donnez les requêtes SQL permettant d'obtenir les informations suivantes :

1. Les livres avec leurs éditeurs.
   SCHÉMA : `(id_livre, titre, nom_editeur)`

2. Toutes les adresses email non `null` des enseignants, sans doublon.
   SCHÉMA : `(email)`

3. Les livres écrits par Guy Pujolle.
   SCHÉMA : `(titre)`

4. Les titres des livres dont le nom d'un auteur commence par 'F'.
   SCHÉMA : `(titre)`
   Si un livre a deux auteurs dont le nom commence par F, il n'apparaîtra qu'une fois.

5. Les titres des livres avec leur nombre d'auteurs.
   SCHÉMA : `(titre, nombre_auteurs)`

6. Le nombre moyen d'auteurs par livre
   SCHÉMA : `(nombre_moyen_auteurs)`

7. Les livres et le nombre de fois où ils sont conseillés (0 si le livre n'est pas conseillé).
   SCHÉMA : `(id_livre, titre, nb_conseils)`

8. Les livres conseillés pour au moins trois matières.
   SCHÉMA : `(id_livre, titre)`

9. Les livres qui ne sont conseillés par aucun enseignant.
   SCHÉMA : `(id_livre, titre)`

10. Le nom (ou les noms) de l'éditeur qui a le plus de livres dans la base.
    SCHÉMA : `(nom)`

11. Le nombre moyen de livres conseillés par les enseignants (on ne s'intéresse qu'aux enseignants qui ont effectivement conseillé des livres)
    SCHÉMA :`(nb_moyen_conseils)`

Exercice 2 :
On demande à quelques personnes d'aller à des séances de cinéma et de donner leur avis. Celui-ci peut varier entre `'tres bon','pas mal','moyen','mauvais','sans avis'`. Leurs réponses sont stockées dans une base de données dont voici le schéma :

```
TD2_CINEMA(ci_ref,ci_nom, ci_ville)
TD2_FILM(fi_ref,fi_titre,fi_annee)
TD2_SEANCE(se_ref,ci_ref, fi_ref,se_horaire)
TD2_SPECTATEUR(sp_ref,sp_nom,sp_prenom,sp_mail)
TD2_ASSISTE(sp_ref,se_ref,se_avis)
```

Question 2.1 : Représentez le modèle de données sous la forme d'un MCD ou d'un diagramme de classes. Quelle alternative peut-on donner à la relation entre *SEANCE* et *SPECTATEUR* ?

Question 2.2 : Donnez les instructions SQL qui permettent de créer le schéma de cette base de données.

Question 2.3 : Ecrivez en SQL les requêtes suivantes :

1. Quels sont les films de 2014 ?
   SCHÉMA : `(fi_ref, fi_titre, fi_annee)`

2. Où a-t-on pu voir le film 'gravity' ?
   SCHÉMA : `(ci_nom, se_horaire)`

3. Combien de spectateurs interrogés sont allés à la séance de référence `se10` ?
   SCHÉMA : `(nb_spectateurs_se10)`

4. Quel est la moyenne du nombre de spectateurs interrogés par séance ?
   SCHÉMA : `(nb_moyen)`

5. Donnez la liste des spectateurs avec le nombre de séances auxquelles ils sont allés.
   SCHÉMA : `(sp_ref, sp_nom, sp_prenom, nb_seances)`

6. Quels sont les noms des spectateurs qui ont trouvé `mauvais` tous les films qu'ils ont vus ?
   SCHÉMA : `(sp_nom, sp_prenom)`

7. Quel est le titre du film (ou des films) qui a eu le plus d'avis `très bon`, toutes séances confondues ?
   SCHÉMA : `(fi_titre)`

8. Donnez pour chaque cinéma, le (ou les) film qui a eu le plus d'avis `très bon`, toutes séances confondues dans ce cinéma.
   SCHÉMA : `(ci_nom, ci_ville, fi_titre, nb_tres_bon)`

9. Quels sont les couples (spectateur, film) pour lesquels le spectateur a vu plusieurs fois le film mais n'a pas toujours donné le même avis.
   SCHÉMA : `(sp_ref, sp_nom, sp_prenom, fi_titre)`

10. Donnez pour chaque cinéma, son id, son nom, le nombre de séances programmées le week-end , le nombre de séances programmées la semaine. Vous pouvez utiliser la fonction `to_char (date, 'D')` qui renvoie le numéro du jour (1 pour lundi, 7 pour dimanche).
    SCHÉMA : `(ci_nom, ci_ville, nb_we, nb_hors_we)`

11. Donnez, pour chaque film, son titre, le nombre d'avis `mauvais`, le nombre d'avis `moyen`,le nombre d'avis `pas mal`, le nombre d'avis `tres bon`.
    SCHÉMA : `(fi_titre, nb_mauvais, nb_moyen, nb_pas_mal, nb_tres_bon)`

12. Quel est le jour de la semaine où les spectateurs sont les plus bienveillants, tous films confondus, i.e. le jour où le nombre d'avis 'très bon' est proportionnellement le plus important.
    SCHÉMA : `(jour)`

13. Dans la requête précédente, on ne distingue pas les cinémas. Donnez pour chaque cinéma le jour de la semaine où les spectateurs sont les plus bienveillants, tous films confondus.
    SCHÉMA : `(ci_nom, ci_ville, jour)`

14. On considère qu'un film est un succès si au moins 80% des avis sont `tres bon` ou bien `pas mal`. Quels sont les titres des films à succès ? SCHÉMA : `(fi_titre)`

15. On voudrait comparer les goûts des spectateurs à Lille et à Paris et, pour cela, avoir une liste : titre de film, (nb d'avis `très bon` à Lille / nb d'avis à Lille), (nb d'avis `très bon` à Paris / nb d'avis à Paris). Proposez une solution. `(fi_titre, ratio_lille, ratio_paris)`

Pour plusieurs requêtes, la fonction `decode` d'Oracle peut vous être utile. Cette fonction permet de définir une valeur sous condition. La syntaxe est

```
decode({valeur de test}, {valeur de comparaison}, {valeur retournée en cas de correspondance},
       [{valeur de comparaison}, {valeur retournée en cas de correspondance}, [ ... ]],
       {valeur retournée si aucune correspondance n'est trouvée}
       )
```

Question 2.4 : En utilisant les instructions SQL qui permettent de modifier la base,

1. Ajoutez-vous dans la table des spectateurs.
2. Comment supprimer les séances antérieures à juillet 2013 (on suppose que l'horaire comporte l'heure et la date), et les avis relatifs à ces séances ?
3. Modifiez la base pour que le spectateur nommé `Breton` prenne la nouvelle identité `Normand`.
4. Le spectateur nommé `Roux` décide de changer d'avis pour la séance `se4`, et de donner un avis `pas mal`. Modifiez la base en conséquence.

On ajoute une colonne `derniereProjection` à la table `TD2_FILM`, qui contiendra l'horaire de la dernière séance à laquelle ce film a été projeté.

Question 2.5 : Quel est l'ordre SQL qui permet d'ajouter cette colonne ?

Cet attribut est dit *redondant* car l'information est déjà présente dans la base de donnée (dans d'autres tables que `TD2_FILM`).

Question 2.6 : Sans utiliser cette nouvelle colonne, donner la requête SQL qui permet d'obtenir l'horaire de la dernière séance du film de référence `fi1`. Cette requête prouve que l'attribut `derniereProjection` est redondant.

Question 2.7 : Donner l'instruction `update` qui permet de mettre à jour la table `TD2_FILM` en calculant la valeur de `dernieProjection` pour tous les films.