PSI
2017-2018

Exercices : Bases de données - Exploitation

Exercice 1 On a une base de données `base_triangles.db`, consitituée d'une seule table, dont le modèle logique est :
triangle(idt :integer, ab :integer, ac :integer, bc :integer)
Chaque élément de la table représente les longueurs d'un triangle ABC, ainsi qu'un identificateur unique.

1. Indiquer ce que fait les requêtes SQL suivantes :
   `SELECT COUNT(*) FROM triangles`
   `SELECT * FROM triangles WHERE ab+ac+bc=100`
   `SELECT COUNT(*) FROM triangles WHERE ab=ac`
2. Déterminer, à l'aide de requêtes SQL :
   (a) la plus petite valeur des produits $AB \times AC \times BC$, pour les triangles (ABC) de périmètre supérieur ou égal à 100 ;
   (b) les longueurs correspondants au(x) triangle(s) pour le(s)quel(s) le minimum précédent est atteint ;
   (c) tous les triangles rectangles en A ;
   (d) le nombre de tels triangles ;
   (e) le maximum des périmètres des triangles rectangles en A ;
   (f) tous les triangles équilatéraux.

Correction :

1. `SELECT COUNT(*) FROM triangles` : Nombre de triangles dans la base de données
   `SELECT * FROM triangles WHERE ab+ac+bc=100` : Les triangles de périmètre 100
   `SELECT COUNT(*) FROM triangles WHERE ab=ac` : le nombre de triangles isocèle en A (équilatérale compris)
2. Déterminer, à l'aide de requêtes SQL :
   (a) `SELECT MIN(ab*ac*bc) FROM triangles WHERE ab+ac+bc>=100`
   (b) `SELECT ab,ac,bc FROM triangles WHERE ab*ac*bc=(SELECT MIN(ab*ac*bc) FROM triangles WHERE ab+ac+bc>=100) AND ab+ac+bc>=100`
   (c) `SELECT * FROM triangles WHERE ab*ab+ac*ac-bc*bc=0`
   (d) `SELECT COUNT(*) FROM triangles WHERE ab*ab+ac*ac-bc*bc=0`
   (e) `SELECT MAX(ab+ac+bc) FROM triangles WHERE ab*ab+ac*ac-bc*bc=0`
   (f) `SELECT * FROM triangles WHERE ab=ac AND ab=bc`

Exercice 2 Soit la BDD définie par le modèle logique suivant :
- ACTOR (id, name)
- MOVIE (id, title, year, score, votes, director)
  où director réfère à ACTOR(id)
- CASTING (movieid, actorid, ord)
  où movieid réfère à MOVIE(id) et actorid réfère à ACTOR(id)

1. Trouver les films dont le score est le plus élevé.
2. Sortir la liste des films ayant comptabilisé le moins de votes mais dont le score est supérieur à 8.
3. Trouver les réalisateurs qui comptabilisent les meilleurs scores moyens pour leurs films.
4. Pareil pour les acteurs.
5. Sortir le casting du film "Star Wars"
6. Sortir la liste des films dont le réalisateur est aussi acteur.

Correction :

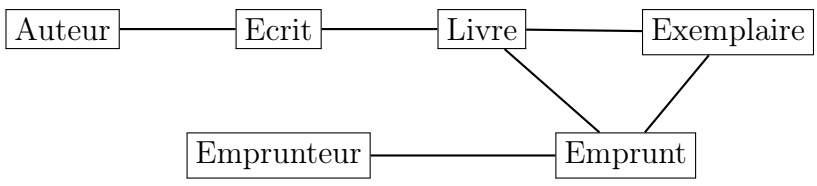
1. `SELECT * FROM MOVIE WHERE score=(SELECT MAX(score) FROM MOVIE)`
2. `SELECT * FROM MOVIE WHERE votes=(SELECT min(votes) FROM MOVIE WHERE score>8) AND score>8`
3. `SELECT ACTOR.name,AVG(MOVIE.score) AS scoreavg FROM ACTOR JOIN MOVIE ON MOVIE.director=ACTOR.id GROUP BY MOVIE.director WHERE scoreavg=(SELECT max(scoreavg) FROM (SELECT AVG(score) AS scoreavg FROM MOVIE GROUP BY director))`
4. `SELECT ACTOR.name,AVG(MOVIE.score) AS scoreavg FROM (CASTING JOIN MOVIE ON MOVIE.id=CASTING.movieid JOIN ACTOR ON CASTING.actorid=ACTOR.id) GROUP BY ACTOR.id WHERE scoreavg=(SELECT max(scoreavg) FROM (SELECT AVG(score) AS scoreavg FROM MOVIE JOIN CASTING ON MOVIE.id=CASTING.movieid GROUP BY CASTING.actorid))`
5. `SELECT ACTOR.name FROM ACTOR JOIN CASTING ON ACTOR.id=CASTING.actorid WHERE CASTING.movieid=(SELECT id FROM MOVIE WHERE title="Star Wars")`
6. `SELECT MOVIE.title FROM MOVIE JOIN CASTING ON MOVIE.id=CASTING.movieid WHERE CASTING.actorid=MOVIE.director`

Exercice 3 La base de données d'une bibliothèque est constituée du modèle logique suivante :
- Auteur(`CODE_AUTEUR, NOM_AUTEUR, PRENOM_AUTEUR`)
  contenant la liste des auteurs.
- Livre(`NO_LIVRE, ISBN, TITRE, EDITEUR, DATE_EDITION`)
  contenant la liste des livres.
- Ecrit(`CODE_AUTEUR, NO_LIVRE`)
  associant à chaque auteur les livres qu'il a écrits.
- Exemplaire(`NO_LIVRE, NO_EXEMPL, ETAT, DATE_ACQ`)
  contenant la liste des exemplaires des livres de la bibliothèque.
- Emprunteur(`NO_INSCR, NOM_EMPR, PRENOM_EMPR, ADR_EMPR`)
  contenant la liste des emprunteurs.
- Emprunt(`NO_LIVRE, NO_EXEMPL, NO_INSCR, DATE_EMPR`)
  contenant la liste des emprunts en cours.

1. En regardant ce modèle, donner l'ensemble des clefs primaires et des clefs étrangères avec leur référence.
2. Donner son modèle conceptuel.
3. Exprimer les requêtes suivantes en SQL.
   (a) Rechercher le titre des livres écrits par Yves Fonctionrecurs.
   (b) Rechercher les numéro, nom et prénom des emprunteurs qui ont en prêt un exemplaire du livre numéro 10.
   (c) Rechercher le nombre d'emprunts en cours.
   (d) Pour chaque livre, rechercher son numéro, son titre et le nombre de ses exemplaires en cours de prêt.
   (e) Rechercher le numéro et le titre des livres de la bibliothèque édités depuis le 1 septembre 2014.
   (f) Rechercher le nom, le prénom et le code des auteurs ayant écrit le plus de livres de la bibliothèque.
   (g) Rechercher le numéro et le titre des livres disponibles (c-à-d dont au moins un exemplaire n'est pas emprunté).
   (h) Rechercher le numéro et le titre des livres dont tous les exemplaires sont empruntés.
   (i) Rechercher le numéro et le titre des livres disponibles (c-à-d dont au moins un exemplaire n'est pas emprunté) édités après le 1 septembre 2010 dont le titre contient le mot "motivation".

Correction :

1. Les clefs primaires :
   - Auteur : `CODE_AUTEUR`
   - Livre : `NO_LIVRE`
   - Ecrit : `CODE_AUTEUR,NO_LIVRE`
   - Exemplaire : `NO_EXEMPL`
   - Emprunteur : `NO_INSCR`
   - Emprunt : `NO_LIVRE,NO_EXEMPL,NO_INSCR`

   Les clefs étrangères :
   - Auteur : $\emptyset$
   - Livre : $\emptyset$
   - Ecrit : `CODE_AUTEUR` réfère à `Auteur.CODE_AUTEUR`
     `NO_LIVRE` réfère à `Exemplaire.NO_LIVRE`
   - Exemplaire : `NO_LIVRE` réfère à `Livre.NO_LIVRE`
   - Emprunteur : $\emptyset$
   - Emprunt : `NO_LIVRE` réfère à `Livre.NO_LIVRE`
     `NO_EXEMPL` réfère à `Exemplaire.NO_EXEMPL`
     `NO_INSCR` réfère à `Emprunteur.NO_INSCR`
2. 

3. (a)
```
SELECT Livre.TITRE FROM Livre JOIN Ecrit ON Livre.NO_LIVRE=Ecrit.NO_LIVRE
WHERE Ecrit.CODE_AUTEUR=(SELECT CODE_AUTEUR FROM Auteur WHERE
NOM_AUTEUR="Yves Fonctionrecurs")
```

   (b)
```
SELECT Emprunteur.NO_INSCR, Emprunteur.NOM_EMPR, Emprunteur.PRENOM_EMPR
FROM Emprunteur JOIN Emprunt ON Emprunteur.NO_INSCR=Emprunt.NO_INSCR
WHERE NO_LIVRE=10
```

   (c)
```
SELECT COUNT(*) FROM EMPRUNT
```

   (d)
```
SELECT Livre.NO_LIVRE, Livre.TITRE, COUNT(Exemplaire.NO_EXEMPL)
FROM Livre JOIN Exemplaire ON Livre.NO_LIVRE=Exemplaire.NO_LIVRE
GROUP BY Livre.NO_LIVRE
```

   (e)
```
SELECT NO_LIVRE TITRE FROM Livre WHERE DATE_EDITION>2014-09-01
```

   (f)
```
SELECT Auteur.*,COUNT(Ecrit.NO_LIVRE) AS NbLivre FROM Auteur JOIN Ecrit
ON Auteur.CODE_AUTEUR=Ecrit.CODE_AUTEUR GROUP BY Auteur.CODE_AUTEUR
WHERE NbLivre=(SELECT MAX(Nblivre) FROM (SELECT COUNT(NO_LIVRE)
AS Nblivre FROM Ecrit GROUP BY CODE_AUTEUR))
```

   (g)
```
SELECT DISTINCT Livre.NO_LIVRE,Livre.TITRE FROM Livre JOIN Exemplaire
ON Livre.NO_LIVRE=Exemplaire.NO_LIVRE WHERE Exemplaire.NO_EXEMPL NOT IN
(SELECT NO_EXEMPL FROM Emprunt)
```

   (h)
```
SELECT NO_LIVRE,TITRE FROM Livre WHERE NO_LIVRE NOT IN (SELECT NO_LIVRE
FROM Exemplaire WHERE NO_EXEMPL NOT IN (SELECT NO_EXEMPL FROM Emprunt))
```

   (i)
```
SELECT DISTINCT Livre.NO_LIVRE,Livre.TITRE FROM Livre JOIN Exemplaire
ON Livre.NO_LIVRE=Exemplaire.NO_LIVRE WHERE (Exemplaire.NO_EXEMPL NOT IN
(SELECT NO_EXEMPL FROM Emprunt))) AND (Livre.DATE_EDITION>2010-09-01)
AND (Livre.TITRE LIKE '%motivation%')
```