Modèle relationnel

Exo 1 : Un loueur de vélo souhaite utiliser une SGBD pour gérer les locations. Un vélo est défini par un numéro de location (unique), une taille (*e.g.* 24 pouces) et une couleur. Un client possède un prénom et un numéro de téléphone (unique). Lorsqu'un vélo est loué à un client, on sauvegarde la durée de la location.

Après avoir schématisé les associations de ce problème, proposer le modèle relationnel correspondant.

Exo 2 : Une bibliothèque souhaite mettre en place une SGBD. Les livres sont définies un titre, un auteur, un numéro unique d'ISBN. Les personnes sont définies par un nom, un prénom, une adresse et un courriel unique. Une personne peut emprunter durant 1 mois un maximum de cinq livres.

Après avoir schématisé les associations de ce problème, proposer le modèle relationnel correspondant.

Exo 3 : Un garagiste souhaite mettre en place unie SGBD pour lister toutes les voitures (déjà immatriculées) qu'il a à vendre. Une voiture possède une immatriculation unique, une marque et un modèle (*e.g.* "123ABC91", "Renault", "twingo"). Les voitures possèdent également une ou plusieurs couleurs. Pour le garagiste, un client est caractérisé par son nom, son compte bancaire (unique) et son numéro de téléphone. Lorsqu'un client achète une voiture, le prix et la date de l'achat sont sauvegardés.

Après avoir schématisé les associations de ce problème, proposer le modèle relationnel correspondant.

Exo 4 : Un salle de spectacle possède un nom (unique), est située dans une ville et propose des spectacles. Un spectacle possède un titre, un producteur et une année de création (titre et producteur permet d'identifié le spectacle). Un spectateur possède un numéro d'adhérent et un nom. Lorsqu'un spectateur choisit un spectacle dans une salle, on sauvegarde une date et un prix.

Après avoir schématisé les associations de ce problème, proposer le modèle relationnel correspondant.

Implémentation sous mySQL

Exo 5 : On considère la bdd 'tournoi' définie par les schémas suivants :

- ◇ **joueur**(idjoueur : `int`, nom : `str`, age : `int`, ceinture : `str`)
- ◇ **match**(idjoueur1 : `int`, idjoueur2 : `int`, heure : `str`, diffscore : `int`)

L'âge d'un joueur ne peut être que positif. Les ceintures possibles sont 'blanc', 'jaune', 'orange', 'vert', 'bleu', 'marron', 'noir' et 'rouge'. L'heure sera au format heure de mySQL et *idjoueurX* sera une clef étrangère.

Proposer un code mySQL qui implémente ces tables ainsi que les contraintes de cette bdd.

Exo 6 : Considérant la bdd de l'exo précédent, proposer un code mySQL qui

a) ajoute les joueurs (1, Teddy Riner, 31, noir), (2, Ryoko Tani, 45, noir), (3, Yasuhiro Yamashita, 63, noir), (4, Agathe, 12, orange), (5, Bobby, 14, orange).

b) met à jour la ceinture de Yasuhiro en rouge et la ceinture d'Agathe en vert.

c) entre le résultat du match Agathe-Bobby gagné par Agathe à 15h30 sur un score de waza-ari (7) contre un koka (3) ; entre le résultat du match Riner-Agathe gagné par Riner à 17h00 sur un score de yuko (5) contre un koka (3).

Exo 7 : On considère la bdd 'cinema' définie par les schémas suivants :

- ◇ **cinema**(idcinema : `int`, nom : `str`, ville : `str`)
- ◇ **film**(titre : `str`, sortie : `int`, duree : `int`)
- ◇ **seance**(idcinema : `int`, titre : `str`, sortie : `int`, jour : `date`)

Le nom d'une ville ainsi que le titre d'un film comportera moins de 25 caractères. La date de sortie d'un film sera une année après 1 970 tandis que sa durée sera exprimée en minutes. Pour une séance, le jour de la séance sera du type date et les autres attributs sont des clefs étrangères.

Proposer un code mySQL qui implémente ces tables ainsi que les contraintes de cette bdd.

Exo 8 : Considérant la bdd de l'exo précédent, proposer un code mySQL qui

a) ajoute les cinemas (1, Melies, Evry), (2, Truffaut, Bondoufle), (3, Tarkovski, Corbeil), (4, Hitchcock, Etiolles), (5, Kubrick, Courcouronnes).

b) ajoute les films (Voyage dans la lune, 1902, 16), (les 400 coups, 2959, 9), (Solaris, 1972, 144), (Psycho, 1960, 109), (Full Metal Jacket, 2987, 116).

c) mets à jour les films (les 400 coups, 1959, 99) et (Full Metal Jacket, 1987, 116).

d) entre une séance à Boudoufle de Solaris le 15 décembre 2020 ; entre une séance à Courcouronnes de Psycho le 12 novembre 2020.

Exo 9 : en considérant la bdd 'gamers' du cours, proposer un code mySQL qui

a) ajoute une table **materiel**(systeme : `str`, idjeu : `int`).
Le système aura une valeur possible parmi 'PCwindows', 'PClinux', 'console' ou 'gameboy' ; enfin *idjeu* sera un clef étrangère.

b) ajoute un jeu (1, Starcraft, 1998) et les matériels (PCwindows, 1) et (console, 1).

c) efface la table **materiel**.

Opérations simples

Sauf mention contraire, on considère la base de données 'gamers' du cours.

Exo 10 : On considère la bdd suivante qui contient les hauteurs de neige dans une station.

- **lieu**(idlieu : `int`, nom : `str`, altitude : `int` , gps : `str`)
- **mesure**(idlieu `int`, date : `str`, hauteur : `int`)

La date sera au format `DATE` de mySQL, à savoir '`AAAA-MM-JJ`'.

idlieu	nom	altitude	gps
1	"Rocher Goulet"	1833	"Lat45.08 :Lon6.09"
2	"Grande Sure"	2114	"Lat45.10 :Lon6.06"
3	"Alpette"	2041	"Lat45.14 :Lon6.09"
4	"Lac Blanc"	2528	"Lat45.12 :Lon6.11"
5	"Glacier Sarenne"	2930	"Lat45.12 :Lon6.13"

idlieu	date	hauteur
1	"2020-10-12"	2
2	"2020-10-12"	12
4	"2020-10-12"	34
5	"2020-10-12"	89
1	"2020-10-13"	0
2	"2020-10-13"	11
3	"2020-10-13"	25
4	"2020-10-13"	33
1	"2020-10-14"	10
2	"2020-10-14"	27

Corriger, si besoin, les requêtes suivantes et donner leurs résultats.

a)
```
SELECT nom, MAX(altitude) FROM lieu ;
```

b)
```
SELECT AVERAGE(hauteur) WHERE idlieu = 4 ;
```

c)
```
SELECT l.nom, m.date, m.hauteur FROM mesure AS m
  INNER JOIN lieu AS l
  WHERE m.date >= "2020-10-13" AND m.hauteur > 20 ;
```

Exo 11 : On considère la bdd de l'exercice 5. Proposer une requête qui permette d'obtenir ce qui est demandé :

a) le nom et l'âge de toutes les ceintures noires.

b) la moyenne d'âge de toutes les ceintures noires.

c) l'heure de tous les combats où le combattant1 a gagné.

d) le nom des combattants1 ayant gagné leur combat

Exo 12 : Avec la bdd du cours, proposer une requête qui permette d'obtenir ce qui est demandé :

a) la liste des prénoms féminins.

b) la liste des prénoms féminins ayant plus de 15 ans.

c) la liste des prénoms masculins étant nés après 2000 trié du plus jeune au plus âgé.

Exo 13 : Proposer une requête qui permette d'obtenir ce qui est demandé :

a) la liste des titres des jeux vidéos.

b) la liste des jeux vidéos sortis en 2005.

c) la liste des jeux vidéos sortis en 2005 et en 2010.

d) la liste des jeux vidéos sortis entre 2005 et 2015 (on pourra utiliser la clause `BETWEEN x AND y`).

Exo 14 : Proposer une requête qui permette d'obtenir ce qui est demandé :

a) les titres et les années de sorties des jeux dont le titre commence par la lettre "f"

b) idem des jeux dont le titre contient le mot "war".

c) le nombre total de jeux dans la bdd

d) la liste des 10 jeux vidéos les plus récents (on pourra utiliser la clause `LIMIT x, y` pour obtenir toutes les réponses entre la x^è et la y^è)

Exo 15 : Proposer une requête qui permette d'obtenir ce qui est demandé :

a) le nombre d'avis négatifs ('bof' ou 'nul') dans la bdd

b) le nombre d'avis positifs ('top' ou 'bien') concernant le jeu Tetris

c) le nombre d'avis positifs récents (moins de 5 ans) concernant le jeu Tetris

Exo 16 : Proposer une requête qui permette d'obtenir ce qui est demandé :

a) la liste des personnes de plus de 15 ans.

b) la moyenne des années de sorties des jeux vidéos

c) la moyenne des années de naissance des personnes

d) la moyenne des âges des personnes

Opérations plus complexes

Sauf mention contraire, on considère la base de données 'gamers' du cours.

Exo 17 : Que permet d'obtenir la requête suivante ?

```
SELECT p.prenom, jeu.titre FROM personne AS p
   INNER JOIN joueur AS j ON j.idpersonne = p.idpersonne
   INNER JOIN jeu ON jeu.idjeu = j.idjeu
   WHERE j.avis = 'top' AND p.prenom >= "L%"
   ORDER BY p.prenom
   LIMIT 10 ;
```

Exo 18 : Que permet d'obtenir la requête suivante ?

```
SELECT p.prenom, jeu.titre, j.annee FROM personne AS p
   INNER JOIN joueur AS j ON j.idpersonne = p.idpersonne
   INNER JOIN jeu ON jeu.idjeu = j.idjeu
   WHERE j.avis IN ('top', 'bien') AND jeu.titre LIKE '%super%'
   ORDER BY j.annee DESC, p.prenom
   LIMIT 10 ;
```

Exo 19 : À l'aide des requêtes des deux exercices précédents, proposer une requête qui renvoie le prénom associé au titre du jeux pour lesquels il a donné un avis 'top'. Les réponses seront limitées à douze et seront triées du jeu le plus récent au plus ancien.

Exo 20 : Que permet d'obtenir la requête suivante ?

```
SELECT COUNT(*) / ssreq.tt * 100 FROM joueur AS j
   INNER JOIN ( SELECT COUNT(*) AS tt FROM joueur ) AS ssreq
   WHERE j.avis = 'bof' ;
```

Exo 21 : Proposer une requête qui permette d'obtenir ce qui est demandé :

a) les prénoms des personnes ayant joué à Tetris

b) les prénoms des personnes ayant joué à Tétris, classés par ordre d'année de sortie du jeu

c) la moyenne des avis positifs ('top' et 'bien') sur le jeu Tétris

Exo 22 : Proposer une requête qui permette de connaître le nombre moyen de jeux auxquels les personnes ont joué.

Exo 23 : Que permet d'obtenir la requête suivante ?

```
SELECT p.prenom FROM personne AS p
   WHERE p.idpersonne NOT IN ( SELECT idpersonne FROM joueur ) ;
```

Exo 24 : Proposer une requête qui permette de lister tous les prénoms des personnes qui n'ont pas joué aux jeux de types 'Action' ou 'Dota-like'.