L3 MIAGE – 2011/2012

Bases de Données

14 décembre 2011

Examen

Durée 3h. Documents et dictionnaire électronique autorisés, appareils mobiles de communication interdits.

Exercice 1 : L'association sportive de Lille 1 organise régulièrement des tournois de tennis. Les étudiants et personnels de Lille 1 peuvent s'y inscrire, en fournissant comme identifiant leur adresse mail à l'université. Afin de se faciliter la tâche, l'association sportive fait appel aux étudiants de MIAGE pour définir une application de gestion des tournois. Elle désire créer une base de données contenant un certain nombre d'informations sur les tournois, les joueurs et les matches.

- un tournoi possède un identifiant (entier positif), une date limite d'inscription, un nombre maximum de joueurs possibles, un nombre réel de joueurs inscrits (inférieur ou égal au nombre maximum possible), et un nombre de tours qui dépend du nombre de joueurs inscrits.
- un joueur possède un nom, un prénom, une adresse email à Lille 1 qui servira d'identifiant. Il peut s'inscrire à n'importe quel tournoi organisé par l'association sportive.
- un match dans un tournoi se joue entre 2 joueurs, et l'un des deux sera le vainqueur. Pour simplifier, on ne mémorisera pas les scores des matches. Les matches d'un tournoi peuvent être représentés par un arbre binaire : aux feuilles se trouvent les matches du premier tour, puis les vainqueurs du premier tour se rencontrent dans les matches du second tour ... ainsi de suite jusqu'à la finale racine de l'arbre[1]. Pour un tournoi donné, comme l'indique la figure 2, on attribue un niveau et un numéro à chaque match, en fonction de son niveau et de sa position de gauche à droite dans l'arbre des tournois.

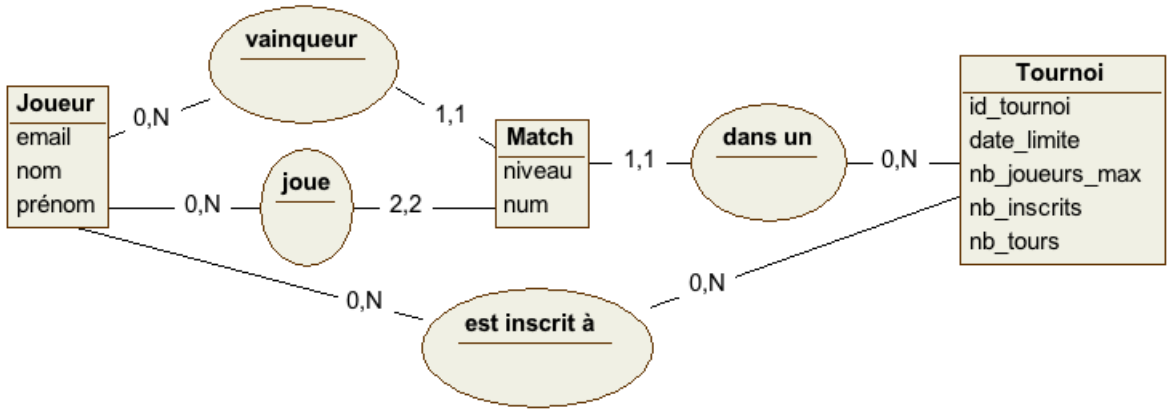

Figure 1: Modèle Conceptuel de Données

[1]Pour simplifier, on supposera dans la question 1.6 que le nombre de joueurs inscrits est une puissance de 2

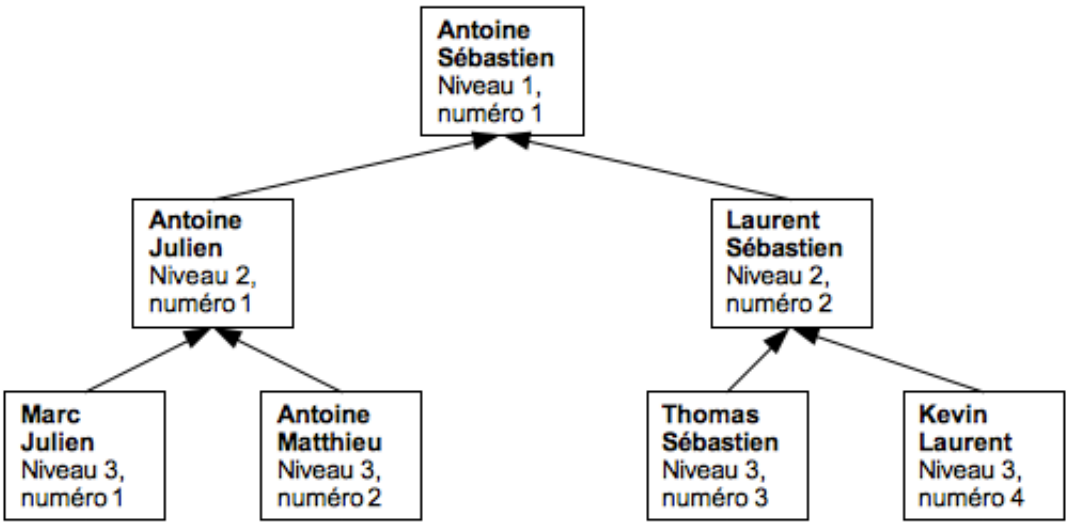

Figure 2: Arbre de tournoi avec 8 joueurs

Voici les tables de la base de données, traduction du MCD de la figure 1 :

```
JOUEUR(email, nom, prenom)
TOURNOI(id_tournoi, date_limite, nb_joueurs_max, nb_inscrits, nb_tours)
INSCRIPTION(email, id_tournoi)
MATCH(id_tournoi, niveau, numero, joueur1, joueur2, vainqueur)
```

Question 1.1 : Donner les instructions SQL de création des tables.

Question 1.2 : Donner les requêtes SQL qui permettent d'obtenir les informations suivantes :

1. Les tournois dont le nombre de joueurs inscrits est strictement inférieur au nombre maximum de joueurs.
   SCHEMA : `(id_tournoi, nb_inscrits, nb_joueurs_max)`
2. Les joueurs qui ne se sont inscrits à aucun tournoi.
   SCHEMA : `(email, nom, prenom)`
3. Le détail des participants aux matches du tournoi d'identifiant 15.
   SCHEMA : `(niveau, numero, nom_j1, prenom_j1, nom_j2, prenom_j2)`
4. Les joueurs qui ont gagné au moins 2 matches pour le tournoi 15.
   SCHEMA : `(email, nom, prenom)`
5. Les joueurs avec le nombre de matches qu'ils ont joués (pas forcément gagnés) sur l'ensemble des tournois.
   SCHEMA : `(email, nom, prenom, nb_matches)`
6. Les tournois avec leur vainqueur (on ne s'intéresse pas aux tournois pas terminés pour lesquels il n'y a pas encore de vainqueur)
   SCHEMA : `(id_tournoi, email, nom, prenom)`

7. Les joueurs qui ont gagné le plus de matches sur l'ensemble des tournois.
   SCHEMA : `(email, nom, prenom)`

Question 1.3 : La valeur de la colonne `nb_inscrits` de la table `TOURNOI` peut se déduire de la table `INSCRIPTION`. Définir un trigger qui calcule automatiquement la colonne `nb_inscrits` en fonction des insertions, modifications et suppressions dans la table `INSCRIPTION`.

Question 1.4 : Le nombre de tours d'un tournoi se déduit du nombre d'inscrits. Définir un trigger qui calcule automatiquement la colonne `nb_tours` en fonction des modifications de la colonne `nb_inscrits` sur la table `INSCRIPTION`. Pour cette question, pour un nombre de joueurs $2^{n-1} < x \leq 2^n$, il y a n tours dans le tournoi. Vous aurez donc besoin de `log(2,n)` qui calcule $log(n)$ et de `ceil(n)` qui calcule l'entier k tel que $k - 1 < n \leq k$. Par exemple, `ceil(2.3)` vaut 3, `ceil(4)` vaut 4.

Par la suite, on définit le paquetage suivant :

```
PACKAGE TOURNOI_PAQ AS
  -- inscrit un joueur à un tournoi
  procedure inscription(le_joueur JOUEUR.EMAIL%type,
                        le_tournoi TOURNOI.ID_TOURNOI%type) ;

  -- crée les matches du premier tour en fonction des inscriptions
  procedure creer_premier_tour(le_tournoi TOURNOI.ID_TOURNOI%type) ;

  -- indique le vainqueur d'un match
  procedure fixe_resultat_match(le_tournoi MATCH.ID_TOURNOI%type,
                                le_niveau MATCH.NIVEAU%type,
                                le_numero MATCH.NUMERO%type,
                                le_vainqueur MATCH.VAINQUEUR%type) ;

  -- renvoie l'identifiant du vainqueur d'un tournoi
  function vainqueur(le_tournoi TOURNOI.ID_TOURNOI%type)
  return MATCH.VAINQUEUR%type ;

  DATE_LIMITE_DEPASSEE Exception ;
  pragma exception_init(DATE_LIMITE_DEPASSEE, -20200);

  PLUS_DE_PLACE Exception ;
  pragma exception_init(PLUS_DE_PLACE, -20201);
END;
```

Question 1.5 : Ecrire le corps de la procédure `inscription` qui permet d'inscrire un joueur, et déclenche l'exception `DATE_LIMITE_DEPASSEE` si la date courante est postérieure à la date limite d'inscription du tournoi, et l'exception `PLUS_DE_PLACE` lorsque le nombre de joueurs inscrits a déjà atteint le nombre maximum de joueurs.

Question 1.6 : Ecrire le corps de la procédure `fixe_resultat_match` qui permet de préciser le vainqueur d'un match. Lorsque l'on connaît les résultats de deux matches "frères" dans

l'arbre des matches, on peut générer le match du tour suivant, opposant les vainqueurs de ces deux matches. Sur la figure 2, lorsque les matches 1 et 2 du premier tour (niveau = 3) sont joués, on peut insérer dans `MATCH` le match 1 du second tour (niveau=2). Pour rappel `mod(n,2)` calcule le reste de la division entière de n par 2. Pour simplifier, on suppose dans cette procédure que le nombre de joueurs (donc de matches à chaque tour) est une puissance de 2.

Exercice 2 : Nous allons nous intéresser dans cet exercice à un programme java qui permet d'utiliser les tables et le package de l'exercice précédent. Soit la classe java `Tennis` qui contient une variable d'instance `connect` de type `java.sql.Connection` initialisée dans le constructeur de la classe par une connexion à la base de l'exercice 1.
Pour les deux questions qui suivent, vous écrirez en dehors des méthodes les déclarations et initialisations des `Statement` utilisés.

Question 2.1 : Ecrire une méthode `String vainqueur(int idTournoi)` qui appelle la fonction `vainqueur` du paquetage `TOURNOI_PAQ` et renvoie une chaîne de caractères avec l'email, le nom et le prénom du vainqueur du tournoi dont l'identifiant est passé en paramètre.

Question 2.2 : Ecrire une méthode `void lesInscrits(int idTournoi)` qui affiche les noms et prénoms des joueurs inscrits au tournoi dont l'identifiant est donné en paramètre.

Question 2.3 : Si on écrit une méthode `inscription` qui appelle la procédure correspondante dans le paquetage `TOURNOI_PAQ`, comment savoir si une `SQLException` déclenchée lors de l'appel de la procédure PL/SQL correspond à l'exception `DATE_LIMITE_DEPASSEE` ou `PLUS_DE_PLACE` ?

Exercice 3 : Soit le schéma relationnel $R(A, B, C, D)$ qui vérifie les dépendances fonctionnelles

$$S = \{A \to C, AB \to D, D \to B, AC \to B\}$$

.

Question 3.1 : Quelles sont les clefs de la relation R ?

Question 3.2 : En utilisant les axiomes d'Armstrong, montrer que S est équivalent à

$$S' = \{A \to C, A \to D, D \to B, A \to B\}$$

.

Question 3.3 : Appliquer l'algorithme de normalisation 3NF pour calculer une décomposition en sous-schémas 3NF de R, décomposition sans perte d'information et avec préservation des dépendances fonctionnelles .

Question 3.4 : Est-ce que la décomposition trouvée à la question précédente est BCNF ? Justifier.