Deuxième TP de bases de données
ECG

Julien Reichert

2023/2024

2

Introduction

Dans ce TP, nous allons changer de base de données pour découvrir des requêtes plus avancées mentionnées comme non exigibles mais apparaissant au programme. Nous commencerons le travail sur la nouvelle base de données par une révision de la syntaxe de base.

Pour commencer, l'exercice de bases de données du sujet zéro paru en décembre 2022 permettra de se faire une idée de ce que les futurs sujets pourront aborder (sachant que, pour la session 2023, aucun exercice sur les bases de données ne semble avoir été recensé...).

Partie I. Étude d'une base de données.

On dispose d'une base de données comportant deux tables `vehicule` et `annonce` décrites ci-dessous.

- La table `vehicule` recense des informations sur les modèles de véhicules en vente sur le marché. Elle est composée des attributs suivants.
  - `id_vehicule` (de type `INTEGER`) : un code permettant d'identifier de façon unique chaque référence de véhicule (marque et modèle).
  - `marque` (de type `TEXT`) : le nom du constructeur du véhicule.
  - `modele` (de type `TEXT`) : le modèle du véhicule, un constructeur proposant en général plusieurs modèles de véhicules à la vente.
  - `prix_neuf` (de type `INTEGER`) : prix de vente du véhicule neuf.
- La table `annonce` regroupe des informations sur un grand nombre d'annonces de véhicules d'occasion. Chaque enregistrement correspond à une annonce et possède les attributs suivants.
  - `id_annonce` (de type `INTEGER`) : un code permettant d'identifier chaque annonce de façon unique.
  - `id_vehicule` (de type `INTEGER`) : l'identifiant du modèle de véhicule vendu, qui correspond à l'identifiant utilisé dans la table `vehicules`.
  - `annee` (de type `INTEGER`) : année de première mise en circulation du véhicule.
  - `km` (de type `INTEGER`) : nombre de kilomètres parcourus par le véhicule au moment de la revente.
  - `prix_occasion` (de type `INTEGER`) : prix de vente du véhicule d'occasion.

1. En justifiant brièvement, identifier une clef primaire dans chacune des tables `vehicule` et `annonce`, ainsi qu'une clef étrangère dans la table `annonce`.

2. Écrire une requête SQL permettant d'extraire les noms de tous les modèles de véhicules mis en vente par le constructeur `Dubreuil Motors`.

3. Expliquer le fonctionnement de la requête SQL suivante et préciser l'effet éventuel de cette requête sur chacune des tables `vehicule` et `annonce`.

```sql
UPDATE annonce
SET prix_occasion = prix_neuf
FROM vehicule
WHERE vehicule.id_vehicule = annonce.id_vehicule
    AND vehicule.prix_neuf < annonce.prix_occasion
```

4. À l'aide d'une jointure, écrire une requête SQL permettant d'obtenir, sur une même table, la liste de toutes les annonces de la table `annonce` avec les attributs suivants :

   - l'identifiant de l'annonce `id_annonce`.
   - le kilométrage `km`,
   - le prix de vente du véhicule neuf `prix_neuf`,
   - le prix de l'annonce d'occasion `prix_occasion`.

Le TP

Les identifiants de connexion restent valides. De nouvelles tables indépendantes de celles utilisées pour le commerce en ligne ont été ajoutées entre temps.

Il s'agit de gérer des paris (sans enjeu !) sur les matchs de championnats sportifs comme la coupe du monde 2022. Les nouvelles tables sont les suivantes :

- `Equipes`, dont les attributs sont `Id_equipe` (entier) et `Pays` (chaîne) ;
- `Matchs`, dont les attributs sont `Id_match` (entier), `Date` (chaîne `"MM/JJ"`), `Phase` (chaîne de caractères pouvant valoir `"Groupe A"`, `"Quarts de finale"`, ...), `Equipe1` (entier) et `Equipe2` (entier) ;
- `Parieurs`, dont les attributs sont `Id_parieur` (entier) et `Nom_parieur` (chaîne) ;
- `Paris`, dont les attributs sont `Parieur` (entier), `Id_match` (entier), `Score1` (entier) et `Score2` (entier).
- `Resultats`, dont les attributs sont `Id_match` (entier), `Buts1` (entier) et `Buts2` (entier).

À partir de cette structure, on peut imaginer différentes manières de marquer des points sur les paris : en pronostiquant le bon score ou en se contentant d'annoncer le bon vainqueur (ou de deviner correctement qu'il y a un match nul).

Exercice 1 (révision) : Afficher la liste des parieurs.

Exercice 2 (révision) : Afficher les matchs où aucun but n'a été marqué.

Exercice 3 : Afficher les matchs où une équipe a marqué au moins trois buts.

Une possibilité est d'utiliser des opérateurs booléens dans la condition, mais en pratique pour des requêtes plus compliquées il existe aussi un opérateur pour la réunion (ainsi que pour l'intersection et pour la différence) sur des résultats de requêtes.

```
SELECT ...
UNION
SELECT ...
```

(Pour les autres opérations : `INTERSECT` et `EXCEPT`.)

Exercice 4 : Afficher les identifiants des équipes qui sont « receveuses » (`Equipe1`) pour les quarts de finale.

Exercice 5 : Même exercice mais en faisant apparaître le nom du pays à la place de l'identifiant et en renommant cet attribut `Pays` en `Receveur`.

Un attribut peut être renommé en utilisant le mot-clé `AS` suivi du nouveau nom. En pratique le mot-clé `AS` peut être omis mais c'est moins lisible.

```
SELECT `attribut` AS `alias` FROM ...
```

Par ailleurs, le renommage peut aussi concerner des tables. Ce sera nécessaire si une jointure fait intervenir deux fois la même table, causant une ambigüité qu'on ne peut pas lever autrement.

```
... `table` JOIN `meme_table` AS `autre_nom` ...
```

Exercice 6 : Afficher les quarts de finale avec les deux équipes, en renommant les attributs `Receveur` et `Visiteur`.

Exercice 7 : Afficher le nombre de paris effectués.

La fonction `COUNT` permet de récupérer le nombre de valeurs (sans éliminer les doublons) d'un attribut **en tant qu'attribut fictif, donc qui peut être mis après un** `SELECT`. Comme il y a pour chaque attribut une valeur par enregistrement (on exclut les valeurs `NULL`, hors-programme), n'importe quel attribut peut être compté, et en pratique on utilisera `COUNT(*)`. Attention, le moteur utilisé ici signale une erreur de syntaxe si la parenthèse est précédée d'une espace.

Un cas particulier : on écrit `COUNT(DISTINCT `attribut`)` si on veut éliminer les doublons de l'attribut en question lors d'un comptage.

Exercice 8 : Afficher le nombre de personnes ayant fait au moins un pari.

Exercice 9 : Afficher le plus grand nombre de buts marqués par une équipe qui était `Equipe1`.

Cette fois-ci, la fonction à utiliser est `MAX`, sur un attribut là encore. D'autres fonctions

sont au programme : `MIN`, `SUM` (somme) et `AVG` (moyenne).

Exercice 10 : Compter le nombre de points du parieur numéro 1 en comptant un point pour chaque pari où le score exact a été trouvé.

Exercice 11 : Même question mais en comptant un point pour chaque pari où le résultat (gagné / nul / perdu) est correct.

Pour faire le classement des parieurs, un regroupement est nécessaire. Ceci est hors-programme mais pourra faire l'objet d'une démonstration en fin de séance. On en profitera pour illustrer `ORDER BY` pour trier les résultats affichés, et l'ensemble du programme sera traité.