TP1 : Construction d'une base de données sous Oracle

L'objectif de ce TP est de vous familiariser avec l'environnement de développement proposé par Oracle pour implanter un modèle relationnel, gérer les droits des utilisateurs, créer et interroger des données.

A - Création des tables de la base de données

Cahier des charges

Nous souhaitons concevoir une base de données pour stocker des informations sur plusieurs championnats de football européens. Chaque équipe a un nom unique (par exemple, `Paris Saint-Germain`) et un nom court, comportant 3 lettres, utilisé notamment pour afficher le score lors d'une rencontre télévisée (par exemple, `PSG`). Une équipe appartient à une ligue d'un pays donné (par exemple, la `Ligue 1` en `France`). Dans cette base de données, nous ne gérerons que la ligue la plus importante d'un pays donné (la ligue 1 pour la France). Chaque joueur a un nom, prénom, une date de naissance, une taille (en mètre) et un poids (en kg). Seul le nom sera obligatoirement renseigné dans la base de données. Pour la taille et le poids, nous conserverons deux décimales (par exemple, `1.75` pour la taille et `70.25` pour le poids). Les joueurs s'engagent avec une équipe pour une saison donnée (par exemple, la saison `2017/2018` qui a débuté le `01/08/2017` et se termine le `30/06/2018`). Au cours de la saison, ils peuvent changer d'équipe (un joueur peut donc avoir été engagé avec plusieurs équipes pendant une saison donnée). Un match implique une équipe locale et une équipe visiteur. Il a lieu à une certaine date et aboutit à un score final entre les deux équipes. Pour chaque match joué, on souhaite connaître les buteurs et la minute pendant laquelle ils ont marqué (on supposera, même si c'est déjà arrivé, qu'un joueur ne peut pas marquer deux buts pendant la même minute).

Un schéma entité-association possible, correspondant à ce cahier des charges, est proposé sur la figure 1.

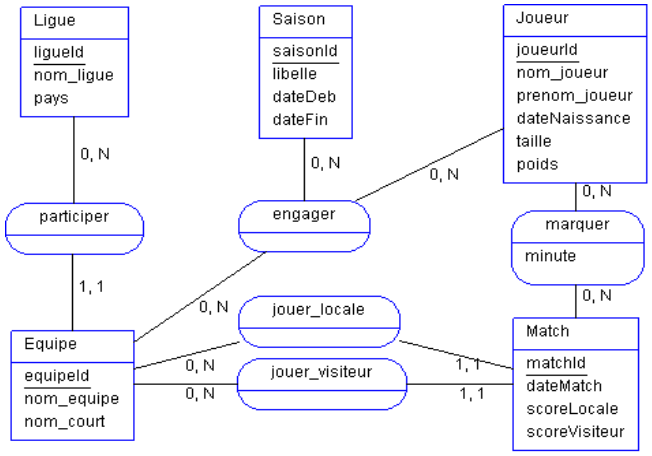

FIGURE 1 – Schéma entité-association

Questions

A-1. Faites un script SQL permettant de créer les tables de la base de données. Pour chaque colonne, vous essaierez de mettre un type adapté (qui évite de réserver de la place inutilement). Vous considérerez, pour cela, que les chaînes de caractères sont limitées à 100 caractères. La syntaxe de création des tables est donnée dans l'aide ci-dessous.

A-2. Modifiez votre script pour intégrer les contraintes de clés primaires et étrangères. Vous nommerez explicitement chacune de ces contraintes en suivant les conventions suivantes : la clé primaire d'une table `T` est nommée `PK_T`. Une clé étrangère d'une table T_1 qui référence une table T_2 est nommée `FK_`T_1`_`T_2. S'il y a plusieurs clés étrangères de T_1 vers T_2, on ajoutera un numéro aux noms de ces contraintes pour les différencier.

A-3. Avec des commandes SQL, ajoutez la colonne `seconde` à la table `Marquer` et modifiez la colonne `nom_ligue` de la table `Ligue` pour que ce soit une chaîne de caractères d'au plus 150 caractères.

Aide

Syntaxe de création d'une table :

```
CREATE TABLE nom_table (
  nom_colonne type_colonne,
  nom_colonne type_colonne,
  ...,
  CONSTRAINT nom_contrainte PRIMARY KEY (nom_colonne),
  CONSTRAINT nom_contrainte
    FOREIGN KEY (nom_colonne) REFERENCES nom_table (nom_colonne)
);
```

Les différents types de données disponibles sous Oracle sont décrits à l'adresse `http://www.ss64.com/orasyntax/datatypes.html`. Les types habituellement utilisés sont : `VARCHAR2` (chaîne de caractères), `NUMBER` (nombre entier ou réel avec précision indiquée), `INT` (un `NUMBER(38)`), `DATE`.

Modification d'une table (ajout/modification/suppression de colonnes et contraintes) ;

```
ALTER TABLE nom_table ADD nom_colonne type_colonne;
ALTER TABLE nom_table DROP COLUMN nom_colonne;
ALTER TABLE nom_table MODIFY nom_colonne type_colonne;
ALTER TABLE nom_table ADD CONSTRAINT nom_contrainte ....;
ALTER TABLE nom_table DROP CONSTRAINT nom_contrainte;
```

Suppression d'une table :

```
DROP TABLE nom_table CASCADE CONSTRAINTS;
```

L'option `CASCADE CONSTRAINTS` est optionnelle ; elle permet d'enlever les références sur la table avant de la supprimer.

B - Rétro-conception d'une base de données

Nous souhaitons remplir la base de données avec des données de précédentes saisons. Pour cela, nous allons utiliser une base de données réelle, disponible sur Internet, nommée `European Soccer Database` (`https://www.kaggle.com/hugomathien/soccer`). Une copie de cette base de données est disponible dans le schéma de l'utilisateur `dataL3`. Pour voir ces tables sous SQLDeveloper, allez dans

`Autres utilisateurs` → `dataL3` (en bas du panel de gauche ; sous Tables, Vues, etc.). Pour faire des requêtes sur ces tables, il suffit de préfixer leurs noms par `dataL3` (par exemple, `SELECT * from dataL3.Team`).

Les tables ont été copiées telles qu'on les trouve dans l'`European Soccer Database`. Le but est de vous apprendre à comprendre et utiliser une base de données qui n'est pas forcément bien conçue (ce que l'on trouve malheureusement souvent dans l'industrie).

Questions

B-1. Reconstituez grossièrement le schéma entité-association (MCD) de cette base de données en utilisant le logiciel `AnalyseSI`. Pour lancer ce logiciel, récupérez et exécutez le jar qui se trouve sur `https://launchpad.net/analysesi`. Comme certaines tables comportent de nombreuses colonnes, vous ne définirez que quelques colonnes significatives pour chaque entité. De même, il y a de nombreuses associations entre les matchs et les joueurs. Vous n'en définirez qu'une ou deux, les autres étant similaires. Vous avez une aide ci-dessous qui décrit quelques colonnes de certaines tables.

B-2. Listez les problèmes de conception et d'implémentation de cette base de données.

Aide

Quelques infos sur certaines tables.

- `Match` : les colonnes `home_player_n` (où `n` est compris entre 1 et 11) correspondent aux 11 joueurs de l'équipe locale qui ont débuté le match. Les colonnes `home_player_xn` et `home_player_yn` correspondent à la position du joueur sur le terrain. Les colonnes similaires, préfixées par `away`, correspondent à l'équipe visiteur. Les colonnes suivantes (`b365h`, `b365d`, etc.) sont des statistiques sur le match.
- `Player` : plusieurs identifiants sont définis : `id` est un identifiant interne à la base de données, `player_api_id` et `player_fifa_api_id` sont probablement des identifiants externes (qui proviennent d'autres bases de données). C'est également le cas d'autres tables comme, par exemple, la table `Match`.
- `Player_Attributes` et `Team_Attributes` : contiennent des caractéristiques sur les joueurs et équipes à une date donnée (ces statistiques évoluent avec le temps comme, par exemple, la vitesse, l'accelération, etc.).

C - Chargement de la base de données

Maintenant que vous devriez mieux comprendre l'`European Soccer Database`, nous allons l'utiliser pour charger progressivement des données dans notre base de données. Pour cela, nous allons utiliser des instructions `INSERT` qui ont la forme suivante :

```
INSERT INTO nom_table (nom_colonne1, nom_colonne2, ...) SELECT ...
```

Cette instruction permet d'insérer le résultat d'une requête SQL dans la table `nom_table`.

C-1. Chargez la table `Ligue` à partir des tables `League` et `Country`. Pour remplir la clé primaire de la table `Ligue`, vous utiliserez la colonne `id` de `League`. Résultats attendus : 11 lignes chargées sans valeur null.

C-2. Chargez la table `Equipe` à partir des tables `Team` et `Match`. La table `Match` vous permettra de récupérer la ligue correspondant à l'équipe insérée. Tous les matchs joués par une équipe (en tant qu'équipe locale ou visiteur) sont associés à la même ligue qui correspond à celle de l'équipe. Pour remplir la clé primaire de la table `Equipe`, vous utiliserez la colonne

`team_api_id` de `Team`. Résultats attendus : 299 lignes insérées. Vous pourrez vérifier que le `PSG` est bien associé à la ligue `France Ligue 1`.

C-3. Créez les saisons à partir de la colonne `season` de la table `Match`. Vous remplirez la clé primaire de la table `Saison` automatiquement grâce à une *séquence* (voir aide ci-dessous). Pour la date de début et date de fin de la saison, nous considérons que toutes les saisons commencent le 1er août et terminent le 30 juin de l'année suivante (par exemple, la saison `2017/2018` débute le `01/08/2017` et se termine le `30/06/2018`). Pour cette question, vous aurez besoin d'utiliser les fonctions `to_date` et `substr` dont vous trouverez facilement la documentation sur Internet. Vous aurez aussi besoin de l'opérateur de concaténation `||`. Résultats attendus : 8 saisons allant de 2008 à 2016.

Aide

Pour donner une valeur automatique à une clé primaire, Oracle ne propose pas de type entier qui s'incrémente automatiquement (`AUTOINCREMENT`). A la place, il propose d'utiliser des *séquences*. Une séquence est un compteur créé par la syntaxe suivante :

```
CREATE SEQUENCE sequence_name;
```

De nombreuses options permettent de définir le comportement du compteur (valeur de départ, incrémentation, etc.). Pour obtenir la valeur courante de la séquence, on utilise la syntaxe `sequence_name.CURRVAL` ; `sequence_name.NEXTVAL` permet d'obtenir la valeur suivante.

Une séquence n'est pas directement rattachée à une colonne d'une table. Pour le faire, il est nécessaire de définir un déclencheur (*trigger*) de la forme suivante :

```
CREATE OR REPLACE TRIGGER nom_trigger
BEFORE INSERT ON nom_table FOR EACH ROW
BEGIN
  SELECT nom_sequence.NEXTVAL INTO :new.cle_primaire FROM DUAL;
END;
/
```

Ce déclencheur permet de donner à la colonne `cle_primaire` la prochaine valeur de la séquence `nom_sequence` avant l'insertion dans la table `nom_table`. `DUAL` est une table vide qui existe par défaut sous Oracle. Le slash (`/`) est nécessaire pour que le déclencheur soit compilé.

C-4. Chargez la table `Joueur` à partir de la table `Player`. Pour remplir la clé primaire de la table `Joueur`, vous utiliserez la colonne `player_api_id` de `Player`. Pour décomposer les valeurs de `player_name` en nom et prénom, nous considérons (même si ce n'est pas toujours vrai) que le prénom est le premier mot de la chaîne de caractères et que le reste constitue le nom de famille ; par exemple, que le prénom de `Angel Di Maria` est `Angel` et que le nom de famille est `Di Maria`. En plus de la fonction `substr`, vous aurez besoin de la fonction `instr` pour chercher un caractère dans une chaîne. Enfin, pensez à convertir les autres champs : `birthday` est de type chaîne de caractères, `height` est définie en `cm` et `weight` en `livres`. Résultats attendus : 11060 lignes insérées. Vous pourrez vérifier que `Angel Di Maria`, né le `14/02/1988` de taille `1.8m` et de poids `74.84kg` est bien présent.

C-5. Chargez la table `Match` à partir de celle de la base `European Soccer Database`. Vous chargerez la clé primaire de la table avec la colonne `id`. Résultats attendus : 25979 matchs insérés dont le match `Olympique de Marseille-Paris Saint-Germain` du `07/02/2016` dont le score était `1-2`.

C-6. Plus difficile. Chargez la table `Engager` à partir de la table `Match`. Pour faire ce chargement, nous partons du fait suivant : si un joueur joue un match de la saison `s1` en tant que `home_player_1`, alors il était engagé avec l'équipe `home_team_api_id` lors de la saison

s1. Le même principe s'applique pour les 11 positions (home_player_1, home_player_2, etc.). On peut aussi suivre la même logique pour les 11 joueurs de l'équipe visiteur (ils étaient engagés avec cette équipe lors de la saison correspondant au match). Commencez par écrire la requête permettant de trouver des lignes de la table Engager grâce à la colonne home_player_1 (c'est-à-dire les équipes avec lesquelles ont été engagés les joueurs qui ont joué à domicile en tant que gardien). Insérez cette requête dans le script genere_requete.sql disponible sur https://www.lias-lab.fr/ftppublic/enseignement/Oracle/L3/ (dans la variable query en veillant à ce que cette requête soit écrite en minuscule). Exécutez ce script, il devrait vous générer la requête à exécuter pour charger la table Engager (attention, ce scrit. Résultats attendus : 35 002 lignes insérées dont le fait que Lavyin Kurzawa a joué de la saison 2010/2011 à la saison 2015/2016 à l'AS Monaco, puis a signé pendant la saison 2015/2016 au PSG.

Nous allons maintenant légèrement compléter et modifier les données avec les commandes suivantes.

```
INSERT INTO nom_table (nom_colonne1, nom_colonne2, ...);
               VALUES (valeur1, valeur2, ....)
UPDATE nom_table SET nom_colonne1 = valeur1,
                     nom_colonne2 = valeur2, ... WHERE condition;
DELETE FROM nom_table WHERE condition;
```

C-7. Insérez le fait que lors du match Olympique de Marseille-Paris Saint-Germain du 07/02/2016, le but de Marseille a été marqué à la 25ème minute par Rémy Cabella et les deux buts de Paris à la 2ème et 71ème minute par, respectivement, Zlatan Ibrahimovic et Angel Di Maria.

C-8. Modifiez les noms des pays dans la table Ligue pour qu'ils soient tous en majuscule (fonction upper).

C-9. Faites une fleur à l'ES Troyes AC, supprimez le match qui a la plus grande différence de but entre l'équipe visiteur et l'équipe locale (bref, le match où l'équipe à domicile a pris la plus grosse raclée).

D - Gestion des utilisateurs

Trois types d'utilisateurs sont amenés à utiliser la base de données :

- des administrateurs qui saisissent en début de saison les données initiales (nouvelles équipes, nouveaux joueurs, infos sur la saison, les contrats des joueurs et les matchs prévus) ;
- des commentateurs qui rentrent les données sur un match (score et éventuels buteurs) à la fin d'une rencontre ;
- des internautes qui ont le droit de consulter les données sur l'ensemble des tables mais pas de les modifier.

Questions

D-1. Créez un rôle pour chacun des types d'utilisateurs (voir aide ci-dessous). Pour éviter les conflits avec les autres utilisateurs, vous préfixerez le nom des rôles par votre nom d'utilisateur ;

D-2. Attribuez vos rôles aux utilisateurs admin, internaute et commentateur (leur mot de passe est tp) ;

D-3. Faites un ensemble de tests permettant de vérifier que chaque utilisateur n'a que les droits nécessaires.

Aide

Quelques instructions pour gérer les droits des utilisateurs.

```
CREATE ROLE rôle; -- création d'un rôle
GRANT rôle TO utilisateur; -- affectation d'un rôle à un utilisateur
GRANT droit1, droit2, ... ON nom_table TO utilisateur_ou_rôle
REVOKE droit1, droit2, ... ON nom_table FROM utilisateur_ou_rôle
```

E - Interrogation de la base de données

Pour chacune des requêtes suivantes, écrivez la requête SQL correspondante.

E-1. Quel est le joueur le plus jeune contenu dans la base de données ? Vous afficherez son nom et son âge. La fonction `sysdate` permet d'avoir la date du jour. La fonction `trunc` peut également vous être utile. Résultat attendu : `Leko Jonathan` (son âge dépend de la date d'exécution de la requête).

E-2. Compléter la requête précédente pour afficher le ou les équipes dans lesquelles il a joué et le ou les ligues associées. Résultat attendu : `Leko Jonathan, West Bromwich Albion, England Premier League`.

E-3. Quel était le joueur le plus jeune de la saison 2008/2009 ? Vous ferez attention à ne considérer que les joueurs qui jouaient en 2008/2009 (c'est-à-dire, qui étaient engagés avec une équipe lors de cette saison). Vous afficherez son nom et l'âge qu'il avait au début de la saison 2008/2009. Résultat attendu : `Kaminski Thomas, 15 ans`.

E-4. Affichez le joueur le plus jeune de chaque saison contenue dans la base de données. Pour chacun, vous afficherez son nom et l'âge qu'il avait au début de la saison pendant laquelle il était le plus jeune. Résultats attendus : 8 lignes dont `Kaminski Thomas, 15 ans en 2008/2009`, `Robertson Clark, 15 ans en 2009/2010`, ..., `Leko Jonathan, 16 ans en 2015/2016`.

E-5. Affichez le nombre total de buts pour chaque ligue lors de la saison 2015/2016 ? Le résultat devra être trié par nombre total de buts décroissant. Résultats attendus : 11 résultats dont `Spain LIGA BBVA, 1043 buts`, ..., `Switzerland Super League, 531 buts`.

E-6. Affichez les équipes du championnat de `France Ligue 1` qui avaient gagné plus de 10 matchs à domicile lors de la saison 2015-2016. Le résultat devra être trié par nombre de victoires décroissant. Résultat attendu : 4 lignes dont `Paris Saint-Germain, 15 victoires`, ..., `SC Bastia, 11 victoires`.

E-7. Dans les données du championnat 2015/2016, il y a un match retour qui n'a pas été joué (le match aller a été joué mais pas le match retour). Quelles sont les deux équipes concernées ? Pour faire cette requête, vous utiliserez l'opérateur `MINUS` d'Oracle (différence de deux requêtes) et ferez l'hypothèse que les matchs allers sont avant le 31/12/2015 et les matchs retours après. Résultat attendu : `Paris Saint-Germain, ES Troyes AC`.

E-8. Quelles sont les équipes qui, lors de la saison 2015/2016 de `France Ligue 1`, ont fait le même score aux matchs aller et retour (par exemple, à l'aller `OM-PSG 1-0` et au retour `PSG-OM 1-0`). Vous afficherez aussi ce score. Résultat attendu : 10 lignes dont `GFC Ajaccio-Olympique de Marseille 1-1`, ..., `Angers SCO-SC Bastia 1-0`.

E-9. Affichez pour chaque joueur qui a été engagé avec le `PSG` en 2015/2016 son nombre total de buts marqués contre l'`Olympique de Marseille` pendant cette saison. S'il n'a pas marqué de but, on souhaite qu'il soit affiché avec le nombre `0`. On triera le résultat par nombre de buts marqué décroissant et, en cas d'égalité, selon l'ordre alphabétique des noms. Indice : pour faire cette requête, vous aurez besoin d'une jointure externe. Autre indice : la fonction `count` appliquée à une colonne retourne le nombre de valeurs non null de cette colonne. Résultat attendu : 25 résultats dont `Di Maria, 1`, ..., `Verratti, 0`.