



# ÉCOLE CENTRALE LYON

## MOD 3.2 - SYSTÈMES DE GESTION DE BASES DE DONNÉES RAPPORT

---

### BE : Basketball

---

#### *Élèves :*

Nathan CYWIE  
Fransisco FRIAS

#### *Enseignant :*

Mohsen ARDABILIAN

8 novembre 2024

## Table des matières

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Conception du modèle de données</b>	<b>2</b>
<b>3</b>	<b>Implantation et alimentation d'une base de données</b>	<b>13</b>
<b>4</b>	<b>Réalisation d'interfaces</b>	<b>19</b>
4.1	Installation de l'application et création de la base de données . . . . .	19
4.2	Utilisation de la base de données . . . . .	22
4.3	Présentation du développement de l'application . . . . .	23
4.3.1	Ajout de données . . . . .	24
4.3.2	Affichage de données . . . . .	24
4.3.3	Suppression de données . . . . .	26
4.3.4	Affichage des données d'un joueur . . . . .	27
<b>5</b>	<b>Queries</b>	<b>27</b>
5.1	Requête 1 . . . . .	27
5.2	Requête 2 . . . . .	28
5.3	Requête 3 . . . . .	29
5.4	Requête 4 . . . . .	29
5.5	Requête 5 . . . . .	31
5.6	Requête 6 . . . . .	32
5.7	Requête 7 . . . . .	32
<b>6</b>	<b>Bibliographie</b>	<b>33</b>

## 1 Introduction

Ce projet a été réalisé dans le cadre du MOD 3.2 - Systèmes de bases de données. Il a pour objectif de nous familiariser avec l'implantation pratique et la gestion de tels systèmes. Parmi la liste de cinq sujets proposés, nous avons retenu le troisième projet. Le but de ce projet est de créer une base de données pour gérer les informations sur les joueurs de basket-ball qui jouent dans les clubs ou les équipes nationales de différents pays.

Ce rapport contient tout d'abord une conception du modèle de données avec la modélisation Entité-Association. Nous y explicitons ensuite l'implantation et alimentation d'une base de la données avec le schéma relationnel et les requêtes de création des entités de la base. Enfin, nous revenons sur la réalisation d'interfaces applicatives permettant d'interagir avec la base de données.

Le code source commenté est disponible dans le dossier "Projet\_Basketball" fourni avec ce rapport.

## 2 Conception du modèle de données

Notre domaine d'application, informations sur les joueurs de basket-ball, présente une certaine complexité. Il devient complexe de raisonner à son sujet en termes de tables et de colonnes. Il n'en reste pas moins qu'un niveau de raisonnement indépendant des technologies de gestion de données s'avère rapidement indispensable. C'est pour cette raison qu'on fera appel à un autre mode de description du domaine d'application : le modèle Entité-association. Ce modèle permet de décrire plus naturellement les concepts du domaine et leurs informations, sans s'attacher à la manière dont ces informations seront représentées en tables et colonnes. La description du domaine s'exprimera sous la forme d'un schéma conceptuel. Le schéma conceptuel est un modèle mental, abstrait, des concepts du domaine d'application et de ses informations et n'est donc pas directement implantable dans un ordinateur. Le schéma dans cette section a été réalisé à l'aide de l'AGL DB-MAIN, spécialisé dans les activités d'ingénierie des applications de bases de données.

Le domaine d'application est perçu comme étant constitué d'entités concrètes ou abstraites. Ainsi, dans le contexte des joueurs de basket-ball, on peut cerner un domaine d'application dans lequel on repère des équipes, des sponsors, des liges, des matchs, des joueurs et des points. On considère que chacun d'eux est une entité du domaine et que chaque entité appartient à une classe ou type d'entités. On définit naturellement six types d'entités qu'on nommera TEAMS, SPONSOR, LEAGUES, GAMES, PLAYERS, et EVENTS et qu'on représentera d'une manière graphique comme indiqué dans la Figure 1.



FIGURE 1 – Les six types d'entités du domaine d'application

Chaque joueur est caractérisé par un Nom, date de naissance, pays d'origine et taille.

On modélisera ces faits en dotant le type d'entités PLAYERS des attributs NAME\_PLAYER, BIRTH\_DATE, COUNTRY\_NAME and HEIGHT. Selon le même raisonnement, on assignera à TEAMS les attributs NAME\_TEAM, CITY et COUNTRY\_NAME. Il convient de noter que ce dernier prend en compte à la fois les clubs et les équipes nationales. De cette façon, nous allons désigner les attributs pour le reste des types d'entités. Les EVENTS sont un type d'entité utilisé pour représenter les points et les statistiques qui sont comptabilisés au cours d'un match.

On spécifiera le type de chaque attribut : numérique, caractère, date, etc. ainsi que sa longueur.

<b>TEAMS</b> NAME_TEAM varchar2 (100 byte) CITY varchar2 (50 byte) COUNTRY varchar2 (60 Byte)	<b>SPONSOR</b> NAME_SPONSOR varchar2 (100 byte) CITY varchar2 (100 byte) AMOUNT number (12,2) COUNTRY varchar2 (60 Byte)	<b>LEAGUES</b> NAME_LEAGUE varchar2 (100 byte) COUNTRY varchar2 (60 Byte)	<b>GAMES</b> TEAM_1 varchar2 (100 byte) TEAM_2 varchar2 (100 byte) GAME_DATE date SEASON number STAGE varchar2 (100 byte)	<b>PLAYERS</b> LASTNAME_PLAYER varchar2 (100 byte) NAME_PLAYER varchar2 (100 byte) DATE OF BIRTH date HEIGHT number (3,2) CITIZENSHIP varchar2 (60 Byte)	<b>EVENTS</b> NAME_EVENT varchar2 (50 byte) POINTS number
--	--	---	--	---	---

FIGURE 2 – Les six types d'entités et leurs attributs

Chacun de ces types d'entités est associé aux autres d'une manière ou d'une autre. Ce type de lien est appelé type d'associations et il crée dans chaque type d'entité un rôle particulier. En combinant les symboles des classes fonctionnelles et du caractère obligatoire des rôles, on obtient la cardinalité associée aux rôles.

Notre schéma doit être normalisée. Il doit encore satisfaire des critères de forme qui lui conféreront des qualités essentielles telles que la facilité d'utilisation des (futurs) données, la lisibilité, l'expressivité, la concision, la clarté ou l'évolutivité. On doit assurer que notre schéma respecte ces critères par le biais d'un processus qui consiste à transformer une relation de manière que chaque fait y soit représenté une et une seule fois.

Notre diagramme Entité-Association réalisé avec DBMAIN est le suivant :

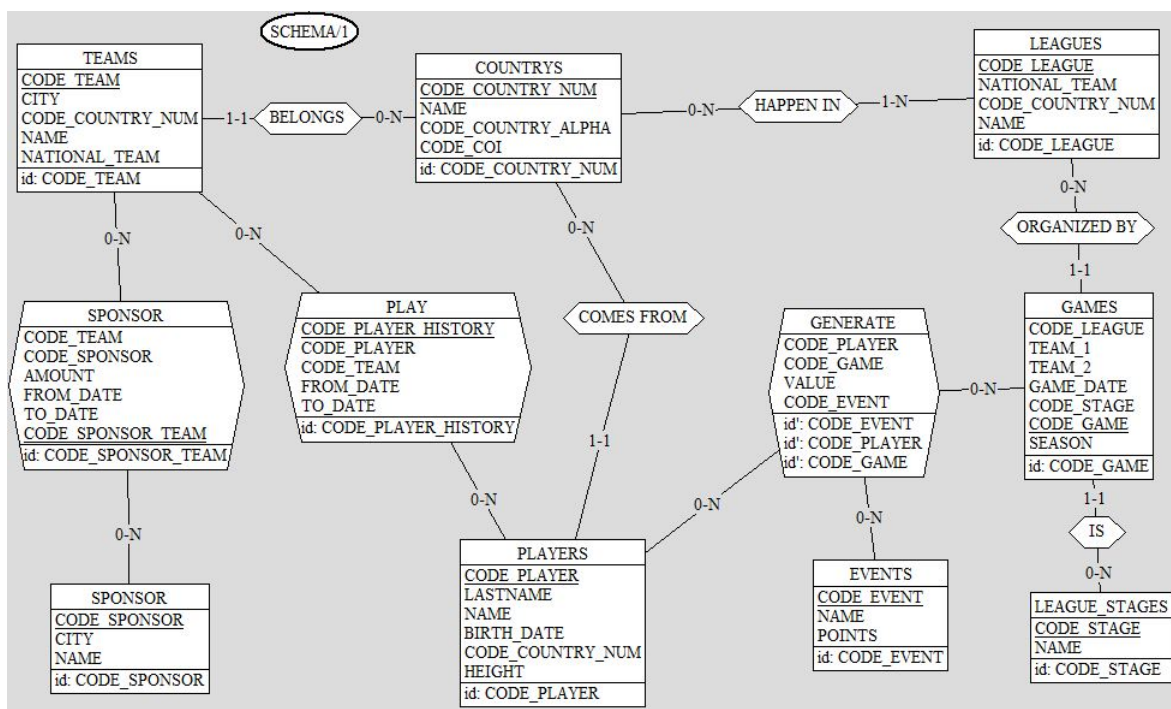


FIGURE 3 – Diagramme Entité-Association réalisé avec DBMAIN

Dans ce dernier schéma, nous pouvons visualiser un large éventail de types d'entités et de types d'associations, que nous allons maintenant clarifier.

Comme nous l'avons déjà vu, nous avons en principe six entités bien définies.

**TEAMS** : Représente les clubs et les équipes nationales. Ce type d'entité possède des attributs qui nous permettent de les différencier :

#### *CODE\_TEAM*

Cet attribut est automatiquement généré par le système lorsque l'utilisateur saisit les données de l'équipe.

Contrainte d'unicité : Elle fonctionne comme Clé Primaire et doit être unique dans la table.

Contrainte d'entité : Cet attribut doit être non Null.

Contrainte domaine : doit être un NUMBER.

#### *NAME*

Représente le nom des équipes.

Contrainte d'entité : Cet attribut doit être non Null.

Contrainte domaine : doit être un VARCHAR2 (100 Byte)

#### *CITY*

Représente le nom de la ville où se trouve le club ou la capitale du pays que l'équipe nationale représente.

Contrainte d'entité : Cet attribut doit être non Null.

Contrainte domaine : doit être un VARCHAR2 (50 Byte)

### *CODE\_COUNTRY\_NUM*

Représente le pays du club ou de l'équipe en question.

Contrainte d'entité : Cet attribut doit être non Null.

Contrainte domaine : doit être un NUMBER.

Contrainte référentielle : a un rôle de référence. Pour jouer correctement ce rôle de référence, il est nécessaire que l'ensemble des valeurs de CODE\_COUNTRY\_NUM doivent être un sous-ensemble des valeurs de l'identifiant cible, CODE\_COUNTRY\_NUM dans la table COUNTRY, et ceci a tout instant.

### *NATIONAL\_TEAM*

Cette valeur détermine s'il s'agit d'un club ou d'une équipe nationale.

Contrainte domaine : doit être un NUMBER. Ce numéro doit être 0 ou 1. Si 0, cette équipe représente un club. Au cas où il s'agit de 1, cette équipe représente une équipe nationale.

**SPONSOR** : Représente les sponsors liés au monde du sport, notamment le basket-ball. Ce type d'entité possède des attributs qui nous permettent de les différencier :

### *CODE\_SPONSOR*

Cet attribut est automatiquement généré par le système lorsque l'utilisateur saisit les données du sponsor.

Contrainte d'unicité : Elle fonctionne comme Clé Primaire et doit être unique dans la table.

Contrainte d'entité : Cet attribut doit être non Null.

Contrainte domaine : doit être un NUMBER.

### *NAME*

Représente le nom du sponsor.

Contrainte d'entité : Cet attribut doit être non Null.

Contrainte domaine : doit être un VARCHAR2 (100 octets).

### *CITY*

Représente le nom de la ville où se trouve le sponsor.

Contrainte d'entité : Cet attribut doit être non Null.

Contrainte domaine : doit être un VARCHAR2 (100 octets).

**PLAYERS** : représente les joueurs qui ont joué dans un club ou une équipe nationale. Ce type d'entité possède des attributs qui nous permettent de les différencier :

### *CODE\_PLAYER*

Cet attribut est automatiquement généré par le système lorsque l'utilisateur saisit les données du joueur.

Contrainte d'unicité : Elle fonctionne comme Clé Primaire et doit être unique dans la table.

Contrainte d'entité : Cet attribut doit être non Null.

Contrainte domaine : doit être un NUMBER.

#### *LASTNAME*

Représente le nom de famille du joueur.

Contrainte d'entité : Cet attribut doit être non Null.

Contrainte domaine : doit être un VARCHAR2 (50 octets).

#### *NAME*

Représente le nom du joueur.

Contrainte domaine : doit être un VARCHAR2 (50 octets).

#### *BIRTH\_DATE*

Représente la date de naissance du joueur.

Contrainte d'entité : Cet attribut doit être non Null.

Contrainte domaine : doit être une DATE.

#### *HEIGHT*

Représente la taille du joueur.

Contrainte d'entité : Cet attribut doit être non Null.

Contrainte domaine : doit être un NUMBER (3,2).

#### *CODE\_COUNTRY\_NUM*

Représente le pays du joueur en question.

Contrainte d'entité : Cet attribut doit être non Null.

Contrainte domaine : doit être un NUMBER.

Contrainte référentielle : a un rôle de référence. Pour jouer correctement ce rôle de référence, il est nécessaire que l'ensemble des valeurs de CODE\_COUNTRY\_NUM doivent être un sous-ensemble des valeurs de l'identifiant cible, CODE\_COUNTRY\_NUM dans la table COUNTRY, et ceci a tout instant.

**LEAGUES** : représente les ligues et les championnats de basket-ball. Ce type d'entité possède des attributs qui nous permettent de les différencier :

#### *CODE\_LEAGUE*

Cet attribut est automatiquement généré par le système lorsque l'utilisateur saisit les données relatives à la ligue ou au championnat.

Contrainte d'unicité : Elle fonctionne comme Clé Primaire et doit être unique dans la table.

Contrainte d'entité : Cet attribut doit être non Null.

Contrainte domaine : doit être un NUMBER.

#### *NAME*

Représente le nom officiel de la ligue ou du championnat.

Contrainte d'entité : Cet attribut doit être non Null.

Contrainte domaine : doit être un VARCHAR2 (100 octets).

#### *NATIONAL\_TEAM*

Cette valeur détermine s'il s'agit d'un club ou d'une équipe nationale.

Contrainte domaine : doit être un NUMBER. Ce numéro doit être 0 ou 1. S'il est 0, l'équipe représente un club. Au cas où il s'agit de 1, cette équipe représente une équipe nationale.

**GAMES** : Représente les matchs de basket-ball qui ont été joués. Ce type d'entité possède des attributs qui nous permettent de les différencier :

#### *CODE\_GAME*

Cet attribut est automatiquement généré par le système lorsque l'utilisateur saisit les données du jeu.

Contrainte d'unicité : Elle fonctionne comme Clé Primaire et doit être unique dans la table.

Contrainte d'entité : Cet attribut doit être non Null.

Contrainte domaine : doit être un NUMBER.

#### *TEAM\_1* et *TEAM\_2*

Représentez les deux équipes qui ont disputé le match.

Contrainte d'entité : Cet attribut doit être non Null.

Contrainte domaine : doit être un NUMBER.

Contrainte référentielle : a un rôle de référence. Pour jouer correctement ce rôle de référence, il est nécessaire que l'ensemble des valeurs de *TEAM\_1* et *TEAM\_2* doivent être un sous-ensemble des valeurs de l'identifiant cible, *NAME* dans la table *TEAMS*, et ceci a tout moment.

#### *GAME\_DATE*

Représente la date à laquelle le jeu a été joué.

Contrainte d'entité : Cet attribut doit être non Null.

Contrainte domaine : doit être une DATE.

#### *CODE\_LEAGUE*

Représente la ligue ou le championnat qui organise le match.

Contrainte d'entité : Cet attribut doit être non Null.

Contrainte domaine : doit être un NUMBER.

Contrainte référentielle : a un rôle de référence. Pour jouer correctement ce rôle de référence, il est nécessaire que l'ensemble des valeurs de *CODE\_LEAGUE* doivent être un sous-ensemble des valeurs de l'identifiant cible, *CODE\_LEAGUE* dans la table *LEAGUES*, et ceci a tout moment.

#### *CODE\_STAGE*

Représente le stage de classification du tournoi auquel appartient le match.

Contrainte domaine : doit être un NUMBER.

Contrainte référentielle : a un rôle de référence. Pour jouer correctement ce rôle de référence, il est nécessaire que l'ensemble des valeurs de *CODE\_STAGE* doivent être un sous-ensemble des valeurs de l'identifiant cible, *CODE\_STAGE* dans la table *LEAGUES\_STAGE*, et ceci a tout moment.

#### *SEASON*



Représente la saison dans laquelle se déroule la ligue ou le championnat qui organise le match.

Contrainte d'entité : Cet attribut doit être non Null.

Contrainte domaine : doit être un NUMBER.

**EVENTS** : représente les points et les statistiques collectés pendant un match. Ce type d'entité possède des attributs qui nous permettent de les différencier :

#### *CODE\_EVENT*

Cet attribut est automatiquement généré par le système lorsque l'utilisateur saisit les données des types de scores et de statistiques.

Contrainte d'unicité : Elle fonctionne comme Clé Primaire et doit être unique dans la table.

Contrainte d'entité : Cet attribut doit être non Null.

Contrainte domaine : doit être un NUMBER.

#### *NAME*

Représente le nom officiel du score ou de la statistique.

Contrainte d'entité : Cet attribut doit être non Null.

Contrainte domaine : doit être un VARCHAR2 (50 octets).

#### *POINTS*

Représente la valeur absolue fournie par chaque score ou statistique.

Contrainte domaine : doit être un NUMBER.

Le processus de normalisation nous a permis de différencier deux autres types d'entités :

**COUNTRYS** : Représente les pays. Ce type d'entité possède des attributs qui nous permettent de les différencier :

#### *CODE\_COUNTRY\_NUM*

Cet attribut est automatiquement généré par le système lorsque l'utilisateur saisit les données relatives au pays.

Contrainte d'unicité : Elle fonctionne comme Clé Primaire et doit être unique dans la table.

Contrainte d'entité : Cet attribut doit être non Null.

Contrainte domaine : doit être un NUMBER.

#### *CODE\_COUNTRY\_ALPHA*

Il s'agit d'un code à 3 lettres représentant le pays de manière unique au niveau international. Contrainte domaine : doit être un VARCHAR2 (3 octets).

#### *NAME*

Représente le nom officiel du pays.

Contrainte d'entité : Cet attribut doit être non Null.

Contrainte domaine : doit être un VARCHAR2 (60 octets).

### *CODE\_COI*

Il s'agit d'un code représentant le comité olympique du pays uniquement au niveau international.

Contrainte domaine : doit être un VARCHAR2 (3 octets).

**LEAGUE\_STAGES** : Représente les différents stades de qualification qui peuvent exister dans un tournoi. Ce type d'entité possède des attributs qui nous permettent de les différencier :

### *CODE\_STAGE*

Cet attribut est automatiquement généré par le système lorsque l'utilisateur saisit les données de l'étape de classification.

Contrainte d'unicité : Elle fonctionne comme Clé Primaire et doit être unique dans la table.

Contrainte d'entité : Cet attribut doit être non Null.

Contrainte domaine : doit être un NUMBER.

### *CODE\_LEAGUE*

Représente la ligue ou le championnat qui organise le match.

Contrainte d'entité : Cet attribut doit être non Null.

Contrainte domaine : doit être un NUMBER.

Contrainte référentielle : a un rôle de référence. Pour jouer correctement ce rôle de référence, il est nécessaire que l'ensemble des valeurs de CODE\_LEAGUE doivent être un sous-ensemble des valeurs de l'identifiant cible, CODE\_LEAGUE dans la table LEAGUES, et ceci a tout moment.

### *NAME*

Représente le nom officiel des différentes phases de qualification qui peuvent exister dans un tournoi.

Contrainte d'entité : Cet attribut doit être non Null.

Contrainte domaine : doit être un VARCHAR2 (100 octets).

Tous ces types d'entités sont liés les uns aux autres par les types d'associations. Nous allons examiner chacune d'entre elles en détail. Nous commencerons par les trois types d'associations les plus importants, qui possèdent leurs propres attributs et identifiants, et nous poursuivrons avec les autres, dont la structure est plus simple. Pour ce faire, nous utiliserons la notion de proposition élémentaire. Les propositions de ce type, qu'on qualifiera de binaires, puisqu'elles comportent deux termes, affirment l'existence de deux concepts, représentés respectivement par le sujet et l'objet, et d'un lien, représenté par le verbe, entre ces deux concepts.

Aussi, nous allons chercher à préciser les propriétés d'un type d'associations. On mesurera cette propriété en indiquant, pour chaque type d'entités, les nombres maximum et minimum d'associations auxquelles une entité participe, nombres que nous représenterons d'abord par la classe fonctionnelle et le caractère obligatoire/facultatif des types d'associations, propriétés qui seront ensuite synthétisées sous la forme d'une propriété

de cardinalité. On définit ainsi trois classes fonctionnelles de types d'associations : un-à-plusieurs, un-à-un et plusieurs-à-plusieurs.

**SPONSOR** : cette association relie les entités TEAMS et SPONSOR.

Propositions :

Les clubs et les équipes peuvent être parrainés par plusieurs sponsors.

Les sponsors peuvent parrainer plusieurs clubs et équipes.

Toute entité SPONSOR peut être associée à une entité TEAMS et vice versa.

La cardinalité est donnée par 0 :N pour les deux.

Ses attributs sont :

#### *CODE\_SPONSOR\_TEAM*

Cet attribut est automatiquement généré par le système lorsque l'utilisateur saisit les données relatives à l'historique du sponsor pour chaque équipe.

Contrainte d'unicité : Fonctionne comme Clé Primaire et doit être unique dans la table.

Contrainte d'entité : Cet attribut doit être non Null.

Contrainte domaine : doit être un NUMBER.

#### *CODE\_TEAM*

Représente l'équipe qui a ou avait un lien avec le sponsor.

Contrainte domaine : doit être un NUMBER.

Contrainte référentielle : a un rôle de référence. Pour jouer correctement ce rôle de référence, il est nécessaire que l'ensemble des valeurs de CODE\_TEAM doivent être un sous-ensemble des valeurs de l'identifiant cible, CODE\_TEAM dans la table TEAMS, et ceci à tout instant.

#### *CODE\_SPONSOR*

Représente le sponsor qui a ou avait un lien avec l'équipe.

Contrainte domaine : doit être un NUMBER.

Contrainte référentielle : a un rôle de référence. Afin de jouer correctement ce rôle de référence, il est nécessaire que tous les valeurs de CODE\_SPONSOR doivent être un sous-ensemble des valeurs de l'identifiant cible, CODE\_SPONSOR dans la table SPONSOR, et ceci à tout instant.

#### *AMOUNT*

Représente le montant convenu dans le lien établi entre l'équipe et le sponsor.

Contrainte d'entité : Cet attribut doit être non Null.

Contrainte domaine : doit être un NUMBER (12,2).

#### *FROM\_DATE*

Représente la date à laquelle le lien commence.

Contrainte domaine : doit être un DATE.

#### *TO\_DATE*

Représente la date à laquelle le lien se termine.

Contrainte domaine : doit être un DATE.

**PLAY** : cette association relie les entités TEAMS et PLAYERS.

Propositions :

Les joueurs peuvent jouer pour plusieurs clubs au cours de leur carrière.

Les clubs peuvent jouer avec plusieurs joueurs.

Toute entité PLAYERS peut être associée à une entité TEAMS et vice versa.

La cardinalité est donnée par 0 :N pour les deux.

Ses attributs sont :

#### *CODE\_PLAYER\_HISTORY*

Cet attribut est automatiquement généré par le système lorsque l'utilisateur saisit les données de l'historique des joueurs de chaque équipe.

Contrainte d'unicité : Fonctionne comme Clé Primaire et doit être unique dans la table.

Contrainte d'entité : Cet attribut doit être non Null.

Contrainte domaine : doit être un NUMBER.

#### *CODE\_TEAM*

Représente l'équipe qui a ou avait un lien avec le joueur.

Contrainte domaine : doit être un NUMBER.

Contrainte référentielle : a un rôle de référence. Pour jouer correctement ce rôle de référence, il est nécessaire que l'ensemble des valeurs de CODE\_TEAM doivent être un sous-ensemble des valeurs de l'identifiant cible, CODE\_TEAM dans la table TEAMS, et ceci à tout instant.

#### *CODE\_PLAYER*

Représente le joueur qui a ou avait un lien avec l'équipe.

Contrainte domaine : doit être un NUMBER.

Contrainte référentielle : a un rôle de référence. Pour jouer correctement ce rôle de référence, il est nécessaire que l'ensemble des valeurs de CODE\_PLAYER doivent être un sous-ensemble des valeurs de l'identifiant cible, CODE\_PLAYER dans la table PLAYERS, et ceci à tout instant.

#### *FROM\_DATE*

Représente la date à laquelle le lien est initié.

Contrainte domaine : doit être un DATE.

#### *TO\_DATE*

Représente la date à laquelle le lien se termine.

Contrainte domaine : doit être un DATE.

**GENERATE** : cette association relie les entités GAMES, EVENTS et PLAYERS.

Dans ce cas, nous utiliserons des propositions complexes, car plus de deux termes seront présents.

Propositions :

Les joueurs génèrent des points lors des matchs.

Les points peuvent être générés par les joueurs lors du match.

Lors des matchs, les points sont générés par les joueurs.

Toute entité PLAYERS peut être associée à une entité EVENTS ou GAMES et vice versa.

La cardinalité est donnée par 0 :N pour les trois.

Ses attributs sont :

#### *CODE\_EVENT*

Représente les types d'annotations qui peuvent être générés.

Contrainte d'unicité : Fonctionne comme Clé Primaire et doit être unique dans la table.

Contrainte domaine : doit être un NUMBER.

Contrainte référentielle : a un rôle de référence. Pour jouer correctement ce rôle de référence, il est nécessaire que l'ensemble des valeurs de CODE\_EVENT doivent être un sous-ensemble des valeurs de l'identifiant cible, CODE\_EVENT dans la table EVENTS, et ceci à tout instant.

#### *CODE\_PLAYER*

Représente le joueur qui a généré des points.

Contrainte d'unicité : Fonctionne comme Clé Primaire et doit être unique dans la table.

Contrainte d'entité : Cet attribut doit être non Null.

Contrainte domaine : Doit être un NUMBER.

Contrainte référentielle : a un rôle de référence. Pour jouer correctement ce rôle de référence, il est nécessaire que l'ensemble des valeurs de CODE\_PLAYER doivent être un sous-ensemble des valeurs de l'identifiant cible, CODE\_PLAYER dans la table PLAYERS, et ceci à tout instant.

#### *CODE\_GAME*

Représente le jeu dans lequel les points ont été générés.

Contrainte d'unicité : Fonctionne comme Clé Primaire et doit être unique dans la table.

Contrainte d'entité : Cet attribut doit être non Null.

Contrainte domaine : Doit être un NUMBER.

Contrainte référentielle : a un rôle de référence. Pour jouer correctement ce rôle de référence, il est nécessaire que l'ensemble des valeurs de CODE\_GAME doivent être un sous-ensemble des valeurs de l'identifiant cible, CODE\_GAME dans la table GAMES, et ceci à tout instant.

#### *VALUE*

Représente le nombre de fois où un joueur a fait une contribution pour un score particulier.

Contrainte domaine : doit être un NUMBER.

**BELONGS** : cette association relie les entités TEAMS et COUNTRYS.

Propositions :

Les clubs appartiennent à un pays.

Les pays peuvent avoir plusieurs clubs.

Chaque entité COUNTRY peut être associée à une entité TEAMS.

Chaque entité TEAMS est associée à une entité COUNTRYS.

La cardinalité est donnée par 0 :N pour les PAYS et 1 :1 pour les TEAMS.

**HAPPEN IN** : cette association relie les entités LEAGUES et COUNTRYS.

Propositions :

Les ligues peuvent se dérouler dans plusieurs pays.

Les pays peuvent se produire dans plusieurs ligues.

Chaque entité COUNTRY peut être associée à une entité LEAGUES.

Chaque entité LEAGUES est associée à une entité COUNTRYS.

La cardinalité est donnée par 0 :N pour les PAYS et 1 :N pour les LEAGUES.

**ORGANIZED BY** : cette association lie les entités LEAGUES et GAMES.

Propositions :

Les ligues organisent plusieurs matchs.

Les matchs sont organisés par une ligue.

Chaque entité LEAGUES peut être associée à une entité GAMES.

Chaque entité GAMES est associée à une entité LEAGUES.

La cardinalité est donnée par 0 :N pour les LEAGUES et 1 :1 pour les GAMES.

**IS** : cette association relie les entités LEAGUE\_STAGES et GAMES.

Propositions :

Les matchs sont une étape de classification.

Les phases de qualification peuvent comporter plusieurs matchs.

Chaque entité LEAGUE\_STAGES peut être associée à une entité GAMES.

Chaque entité GAMES est associée à une entité LEAGUE\_STAGE.

La cardinalité est donnée par 0 :N pour LEAGUE\_STAGES et 1 :1 pour GAMES.

**COMES FROM** : cette association relie les entités COUNTRYS et PLAYERS.

Propositions :

Les joueurs viennent d'un seul pays.

De chaque pays peuvent provenir plusieurs joueurs.

Chaque entité COUNTRYS peut être associée à une entité PLAYERS.

Chaque entité PLAYERS est associée à une entité COUNTRYS.

La cardinalité est donnée par 0 :N pour les COUNTRYS et 1 :1 pour les PLAYERS.

### 3 Implantation et alimentation d'une base de données

L'étape suivante consistera à traduire le schéma conceptuel en un schéma relationnel et créer les requêtes optimisées en SQL permettant de créer les tables. Le langage SQL

offre des commandes de définition et de modification des structures. Ces commandes permettent de définir (créer), de supprimer et de modifier une table, un domaine, une colonne ou une contrainte. L'ensemble de ces commandes forme un sous-langage de SQL dénommé SQL-DDL (Data Definition Language).

Notre base de données est définie par le schéma suivant :

```
01 | CREATE DATABASE basketball;
02 | USE basketball;
03 |
04 |
05 | --Tables definition
06 |
07 | CREATE TABLE COUNTRYS(
08 |     NAME VARCHAR(60) NOT NULL,
09 |     CODE_COUNTRY_ALPHA VARCHAR(3),
10 |     CODE_COUNTRY_NUM INT,
11 |     CODE_COI VARCHAR(3)
12 | );
13 | CREATE TABLE EVENTS (
14 |     CODE_EVENT INT,
15 |     NAME VARCHAR(50) NOT NULL,
16 |     POINTS INT
17 | );
18 | CREATE TABLE LEAGUES (
19 |     CODE_LEAGUE INT,
20 |     NAME VARCHAR(100) NOT NULL,
21 |     NATIONAL_TEAM INT NOT NULL,
22 |     CODE_COUNTRY_NUM INT
23 | );
24 | CREATE TABLE PLAYERS (
25 |     LASTNAME VARCHAR(50) NOT NULL,
26 |     NAME VARCHAR(50),
27 |     BIRTH_DATE DATE NOT NULL,
28 |     CODE_COUNTRY_NUM INT NOT NULL,
29 |     CODE_PLAYER INT,
30 |     HEIGHT DECIMAL(3, 2) NOT NULL
31 | );
32 | CREATE TABLE SPONSORS (
33 |     CODE_SPONSOR INT,
34 |     NAME VARCHAR(100) NOT NULL,
35 |     CITY VARCHAR(100) NOT NULL
36 | );
37 | CREATE TABLE STAGES (
38 |     CODE_STAGE INT NOT NULL,
39 |     NAME VARCHAR(100) NOT NULL
40 | );
41 | CREATE TABLE TEAMS (
42 |     CODE_TEAM INT,
43 |     NAME VARCHAR(100) NOT NULL,
44 |     CITY VARCHAR(50) NOT NULL,
45 |     CODE_COUNTRY_NUM INT NOT NULL,
46 |     NATIONAL_TEAM INT
47 | );
48 | CREATE UNIQUE INDEX COUNTRY_PK ON COUNTRYS (CODE_COUNTRY_NUM);
```

```
49 | CREATE UNIQUE INDEX EVENT_PK ON EVENTS (CODE_EVENT);
50 | CREATE UNIQUE INDEX LEAGUES_PK ON LEAGUES (CODE_LEAGUE);
51 | CREATE UNIQUE INDEX LEAGUE_STAGE_PK ON STAGES (CODE_STAGE);
52 | CREATE UNIQUE INDEX PLAYER_PK ON PLAYERS (CODE_PLAYER);
53 | CREATE UNIQUE INDEX SPONSOR_PK ON SPONSORS (CODE_SPONSOR);
54 | CREATE UNIQUE INDEX TEAM_PK ON TEAMS (CODE_TEAM);
55 | CREATE TABLE GAMES (
56 |     CODE_LEAGUE INT NOT NULL, TEAM_1 INT NOT NULL,
57 |     TEAM_2 INT NOT NULL, GAME_DATE DATE NOT NULL,
58 |     CODE_GAME INT, CODE_STAGE INT, SEASON INT NOT NULL
59 | );
60 | CREATE TABLE GAMES_EVENTS (
61 |     CODE_GAME INT, CODE_EVENT INT, CODE_PLAYER INT,
62 |     VALUE_ INT
63 | );
64 | CREATE TABLE PLAYERS_HISTORY (
65 |     CODE_PLAYER INT, CODE_TEAM INT, FROM_DATE DATE NOT NULL,
66 |     TO_DATE DATE, CODE_PLAYER_HISTORY INT
67 | );
68 | CREATE TABLE SPONSOR_TEAM_HISTORY (
69 |     CODE_TEAM INT,
70 |     CODE_SPONSOR INT,
71 |     AMOUNT DECIMAL(12, 2) NOT NULL,
72 |     FROM_DATE DATE,
73 |     TO_DATE DATE,
74 |     CODE_SPONSOR_TEAM INT
75 | );
76 | CREATE UNIQUE INDEX GAMES_EVENT_PK ON GAMES_EVENTS (CODE_GAME,
77 |     CODE_PLAYER);
77 | CREATE UNIQUE INDEX GAMES_PK ON GAMES (CODE_GAME);
78 | CREATE UNIQUE INDEX PK_CODE_PLAYER_HISTORY ON PLAYERS_HISTORY (
79 |     CODE_PLAYER_HISTORY);
79 |
80 |
81 | --Constraints definition
82 |
83 | ALTER TABLE
84 |     COUNTRYS
85 | ADD
86 |     CONSTRAINT COUNTRY_PK PRIMARY KEY (CODE_COUNTRY_NUM);
87 | ALTER TABLE
88 |     EVENTS
89 | ADD
90 |     CONSTRAINT EVENT_PK PRIMARY KEY (CODE_EVENT);
91 | ALTER TABLE
92 |     LEAGUES
93 | ADD
94 |     CONSTRAINT LEAGUES_PK PRIMARY KEY (CODE_LEAGUE);
95 | ALTER TABLE
96 |     PLAYERS
97 | ADD
98 |     CONSTRAINT PLAYER_PK PRIMARY KEY (CODE_PLAYER);
99 | ALTER TABLE
100 |     SPONSORS
101 | ADD
102 |     CONSTRAINT SPONSOR_PK PRIMARY KEY (CODE_SPONSOR);
103 | ALTER TABLE
```



```

104 | STAGES
105 | ADD
106 | CONSTRAINT LEAGUE_STAGE_PK PRIMARY KEY (CODE_STAGE);
107 | ALTER TABLE
108 | TEAMS
109 | ADD
110 | CONSTRAINT TEAM_PK PRIMARY KEY (CODE_TEAM);
111 | ALTER TABLE
112 | GAMES
113 | ADD
114 | CONSTRAINT GAMES_PK PRIMARY KEY (CODE_GAME);
115 | ALTER TABLE
116 | GAMES_EVENTS
117 | ADD
118 | CONSTRAINT GAMES_EVENT_PK PRIMARY KEY (CODE_GAME, CODE_PLAYER);
119 | ALTER TABLE
120 | PLAYERS_HISTORY
121 | ADD
122 | CONSTRAINT PK_CODE_PLAYER_HISTORY PRIMARY KEY (
123 | CODE_PLAYER_HISTORY);
124 | ALTER TABLE
125 | SPONSOR_TEAM_HISTORY
126 | ADD
127 | PRIMARY KEY (CODE_SPONSOR_TEAM);
128 | ALTER TABLE
129 | LEAGUES
130 | ADD
131 | (
132 | CONSTRAINT FK_CODE_NUM_TO_COUNT FOREIGN KEY (CODE_COUNTRY_NUM
133 | ) REFERENCES COUNTRYS (CODE_COUNTRY_NUM)
134 | );
135 | ALTER TABLE
136 | PLAYERS
137 | ADD
138 | (
139 | CONSTRAINT FK_CODE_NUM_TO_COUNTRYS FOREIGN KEY (
140 | CODE_COUNTRY_NUM) REFERENCES COUNTRYS (CODE_COUNTRY_NUM)
141 | );
142 | ALTER TABLE
143 | TEAMS
144 | ADD
145 | (
146 | CONSTRAINT FK_CODE_NUM_T_TO_COUNTS FOREIGN KEY (
147 | CODE_COUNTRY_NUM) REFERENCES COUNTRYS (CODE_COUNTRY_NUM)
148 | );
149 | ALTER TABLE
150 | GAMES
151 | ADD
152 | (
153 | CONSTRAINT FK_CODE_STAGE_TO_STAGE FOREIGN KEY (CODE_STAGE)
154 | REFERENCES STAGES (CODE_STAGE)
155 | );
156 | ALTER TABLE
157 | GAMES_EVENTS
158 | ADD
159 | (
160 | CONSTRAINT FK_CODE_EVENT_TO_EVENTS FOREIGN KEY (CODE_EVENT)

```

```
REFERENCES EVENTS (CODE_EVENT),
156 |     CONSTRAINT FK_CODE_GAME_TO_GAMES FOREIGN KEY (CODE_GAME)
REFERENCES GAMES (CODE_GAME),
157 |     CONSTRAINT FK_CODE_PLAYES_TO_PLAYER FOREIGN KEY (CODE_PLAYER)
REFERENCES PLAYERS (CODE_PLAYER)
158 | );
159 | ALTER TABLE
160 |     PLAYERS_HISTORY
161 | ADD
162 | (
163 |     CONSTRAINT FK_CODE_PLAYER_H_TO_PLAYERS FOREIGN KEY (
CODE_PLAYER) REFERENCES PLAYERS (CODE_PLAYER),
164 |     CONSTRAINT FK_CODE_TEAM_H_TO_TEAMS FOREIGN KEY (CODE_TEAM)
REFERENCES TEAMS (CODE_TEAM)
165 | );
166 | ALTER TABLE
167 |     SPONSOR_TEAM_HISTORY
168 | ADD
169 | (
170 |     CONSTRAINT FK_CODE_SPONSOR_TO_SPONSORS FOREIGN KEY (
CODE_SPONSOR) REFERENCES SPONSORS (CODE_SPONSOR),
171 |     CONSTRAINT FK_CODE_TEAM_TO_TEAMS FOREIGN KEY (CODE_TEAM)
REFERENCES TEAMS (CODE_TEAM)
172 | );
173 |
174 |
175 | --Data loading
176 |
177 | LOAD DATA LOCAL INFILE 'basketball/Data/COUNTRYS.csv' INTO TABLE
COUNTRYS FIELDS TERMINATED BY ';' LINES TERMINATED BY 'n'
IGNORE 1 ROWS;
178 | LOAD DATA LOCAL INFILE 'basketball/Data/TEAMS.csv' INTO TABLE
teams FIELDS TERMINATED BY ';' LINES TERMINATED BY 'n' IGNORE
1 ROWS;
179 | LOAD DATA LOCAL INFILE 'basketball/Data/EVENT.csv' INTO TABLE
events FIELDS TERMINATED BY ';' LINES TERMINATED BY 'n' IGNORE
1 ROWS;
180 | LOAD DATA LOCAL INFILE 'basketball/Data/STAGES.csv' INTO TABLE
stages FIELDS TERMINATED BY ';' LINES TERMINATED BY 'n' IGNORE
1 ROWS;
181 | LOAD DATA LOCAL INFILE 'basketball/Data/GAMES.csv' INTO TABLE
games FIELDS TERMINATED BY ';' LINES TERMINATED BY 'n' IGNORE
1 ROWS;
182 | LOAD DATA LOCAL INFILE 'basketball/Data/LEAGUES.csv' INTO TABLE
leagues FIELDS TERMINATED BY ';' LINES TERMINATED BY 'n'
IGNORE 1 ROWS;
183 | LOAD DATA LOCAL INFILE 'basketball/Data/PLAYERS.csv' INTO TABLE
players FIELDS TERMINATED BY ';' LINES TERMINATED BY 'n'
IGNORE 1 ROWS;
184 | LOAD DATA LOCAL INFILE 'basketball/Data/PLAYERS_HISTORY.csv' INTO
TABLE players_history FIELDS TERMINATED BY ';' LINES
TERMINATED BY 'n' IGNORE 1 ROWS;
185 | LOAD DATA LOCAL INFILE 'basketball/Data/SPONSORS.csv' INTO TABLE
sponsors FIELDS TERMINATED BY ';' LINES TERMINATED BY 'n'
IGNORE 1 ROWS;
186 | LOAD DATA LOCAL INFILE 'basketball/Data/SPONSOR_TEAM_HISTORY.csv'
INTO TABLE sponsor_team_history FIELDS TERMINATED BY ';'
```

```

        LINES TERMINATED BY 'n' IGNORE 1 ROWS;
187 | LOAD DATA LOCAL INFILE 'basketball/Data/GAMES_EVENT.csv' INTO
        TABLE games_events FIELDS TERMINATED BY ';' LINES TERMINATED
        BY 'n' IGNORE 1 ROWS;
    
```

Pour notre application, l'importation de la base de données se fera avec un dump SQL mais nous avons quand même inclus les données sous forme de fichier csv dans le fichier Data.

Une fois la table créée, il est possible de visualiser l'architecture avec l'application MySQL Workbench.

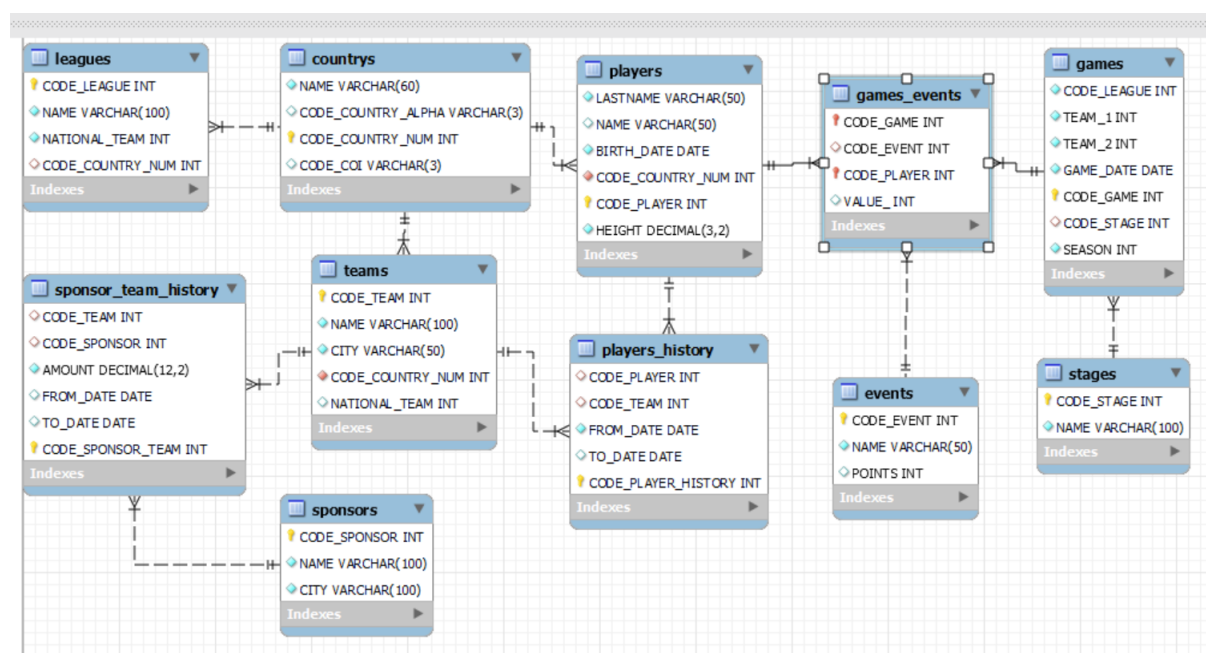


FIGURE 4 – Diagramme généré par MySQL Workbench

La base de données actuelle a été alimentée par des données réelles et fictives. Ces données fictives, bien que non réelles, ont été définies pour s'aligner sur la réalité, maintenir la cohérence de l'ensemble de données et être utiles à des fins pratiques pour ce travail.

Parmi les données réelles, nous pouvons trouver les équipes. Ici, une liste de 6 équipes nationales est affichée : France, Espagne, Allemagne, Brésil, Serbie et Grèce. En outre, nous trouvons 153 équipes appartenant à chacun de ces pays, avec leurs villes d'origine correspondantes. Parmi eux, nous pouvons citer le Real Madrid et le Barcelone dans les villes de Madrid et de Barcelone respectivement, l'ASVEL de la ville de Villeurbanne, le Bayern Munich de la ville de Munich, Flamengo de la ville de Rio de Janeiro, Mega Bemax de la ville de Belgrade et l'AEK de la ville d'Athènes.

Les ligues et les championnats sont également réels. Dans la base de données, nous pouvons trouver la ligue principale de chacun des 6 pays mentionnés ci-dessus et également le championnat mondial organisé par la FIBA.

Quant aux sponsors, aux joueurs, aux matchs et à leurs historiques respectifs, ces données ont été créées artificiellement grâce à une fonctionnalité de l'outil de travail qui génère aléatoirement des noms dans chacun des champs. Pour les matchs, des simulations ont été effectuées à la main entre les équipes de chacune des phases de qualification. Ce processus a été réalisé avec soin pour respecter la cohérence des données, en détaillant les dates et les saisons de chaque match. Les dates contractuelles entre les sponsors et les équipes ont également été soigneusement sélectionnées pour être cohérentes avec le reste des données.

## 4 Réalisation d'interfaces

### 4.1 Installation de l'application et création de la base de données

Pour ce projet, nous avons choisi de réaliser une solution sur le réseau local sous un OS Windows 10. L'installation de ce réseau local (localhost) s'effectuera avec XAMPP qui permet l'utilisation de phpMyAdmin pour administrer les données.

XAMPP est facilement téléchargeable ici : <https://www.apachefriends.org/fr/download.html>.

Une fois le logiciel téléchargé, il est nécessaire d'ouvrir XAMPP et de lancer MySQL et Apache en appuyant sur Start. A ce stade, on doit retrouver l'écran suivant, et on appuie sur le bouton Admin sur la ligne MySQL.

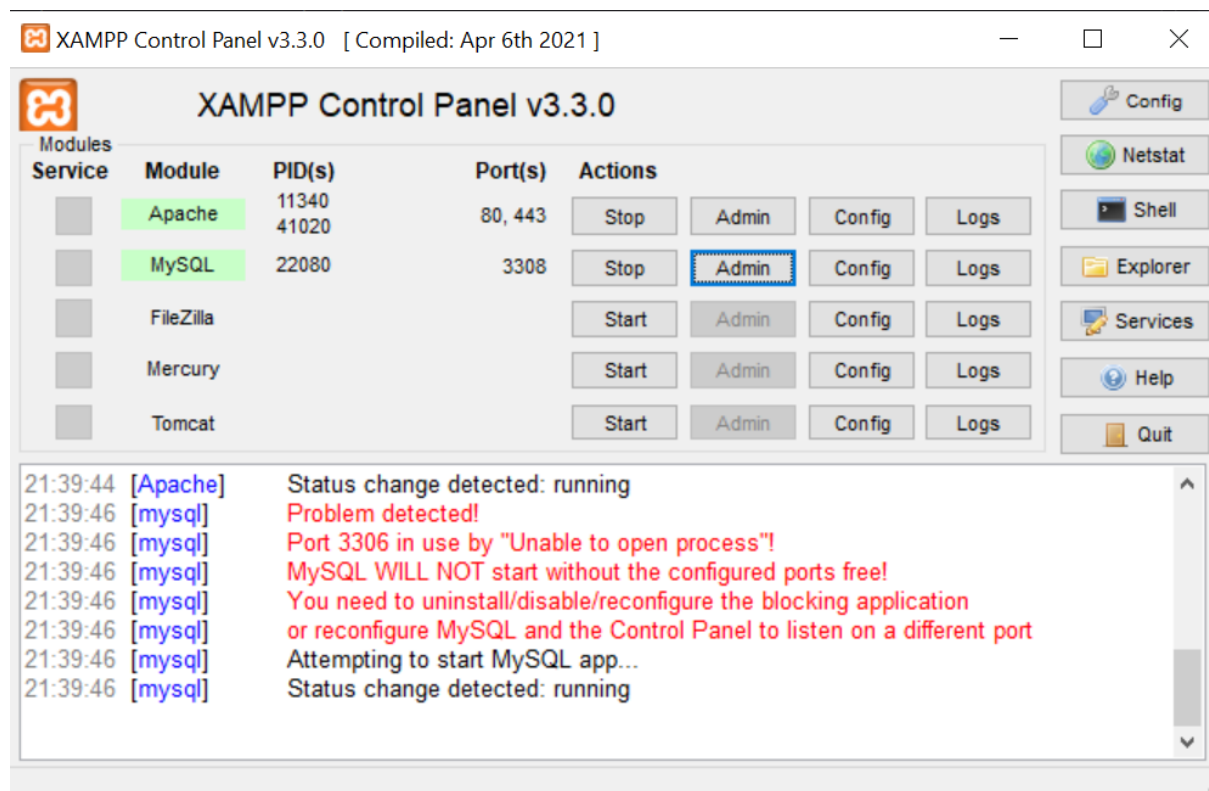


FIGURE 5 – Fenêtre XAMPP

Il faut ensuite exécuter les requêtes suivantes dans la Console de requêtes SQL en bas de la fenêtre.

```
01 | CREATE DATABASE basketball;  
02 | USE basketball;
```

Autrement dit, il faut copier ce texte dans la console et exécuter en faisant CTRL+Entrée, comme indiqué dans la figure suivante.

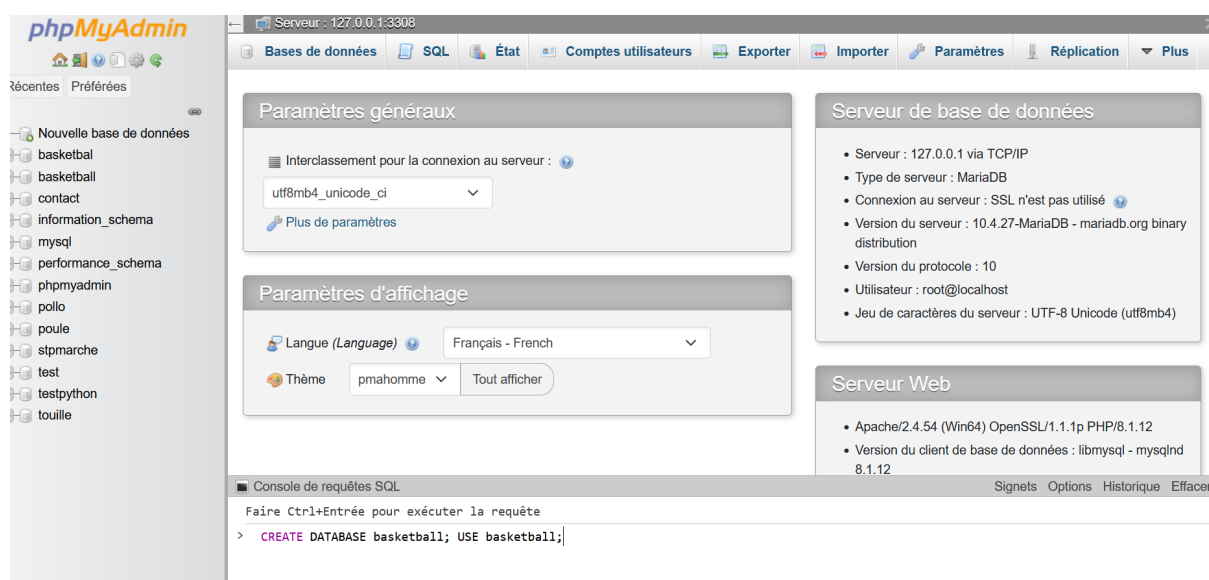


FIGURE 6 – Fenêtre XAMPP

On va ensuite finir de générer la base de données en exécutant le dump SQL fourni dans le dossier. Il s'agit du fichier dump.sql. Pour ce faire, on va dans l'onglet Importer et on clique sur "Parcourir" pour aller sélectionner le fichier SQL. Une fois le fichier sélectionné, il suffit de cliquer sur importer en bas de la page. La base de données est alors remplie.

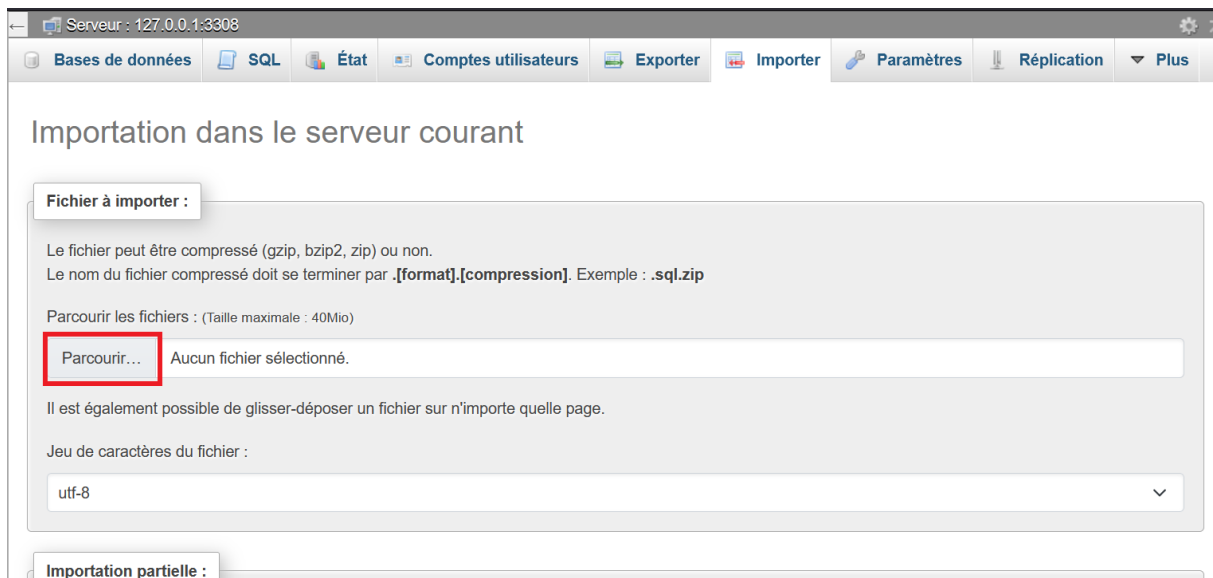


FIGURE 7 – Fenêtre XAMPP

Lors de l'installation de XAMPP, un dossier nommé XAMPP doit avoir été créé dans le disque C :. Pour finir la mise en place de l'application, il faut ouvrir ce dossier, ouvrir le dossier htdocs contenu dans XAMPP, et y copier le dossier basketball. Il sera ensuite possible d'ouvrir l'accueil de l'application dans un navigateur web en copiant le lien <http://localhost/basketball/dataForm.php> dans la barre de recherche. On doit ensuite voir la page suivante qui est l'accueil de notre application.

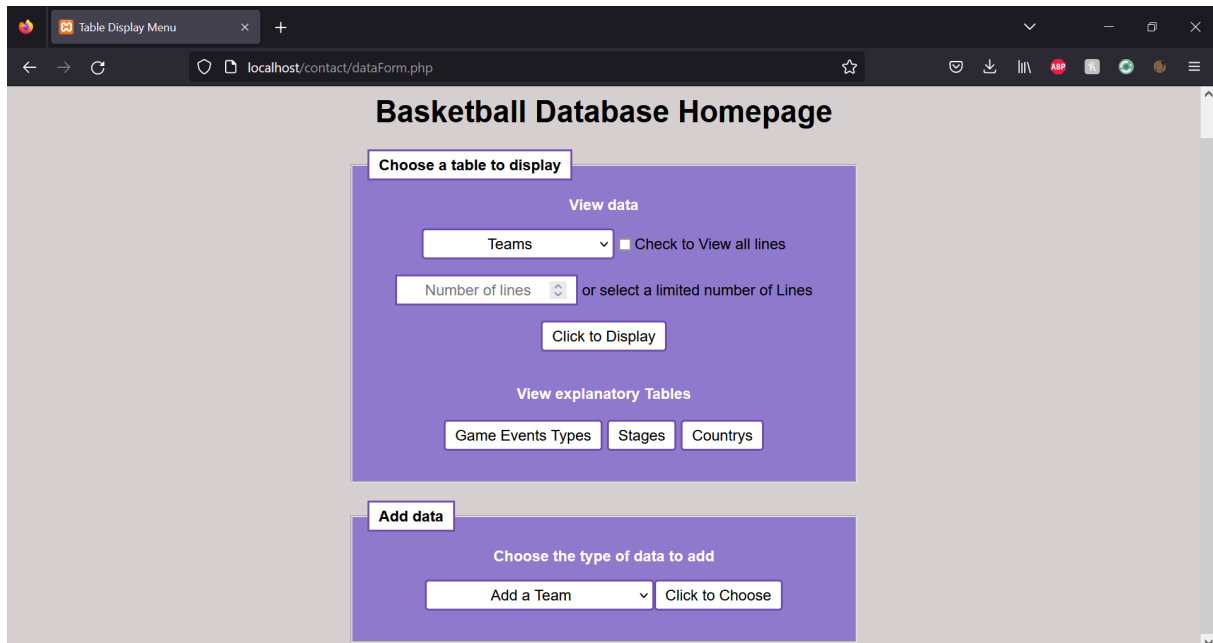


FIGURE 8 – Fenêtre XAMPP

## 4.2 Utilisation de la base de données

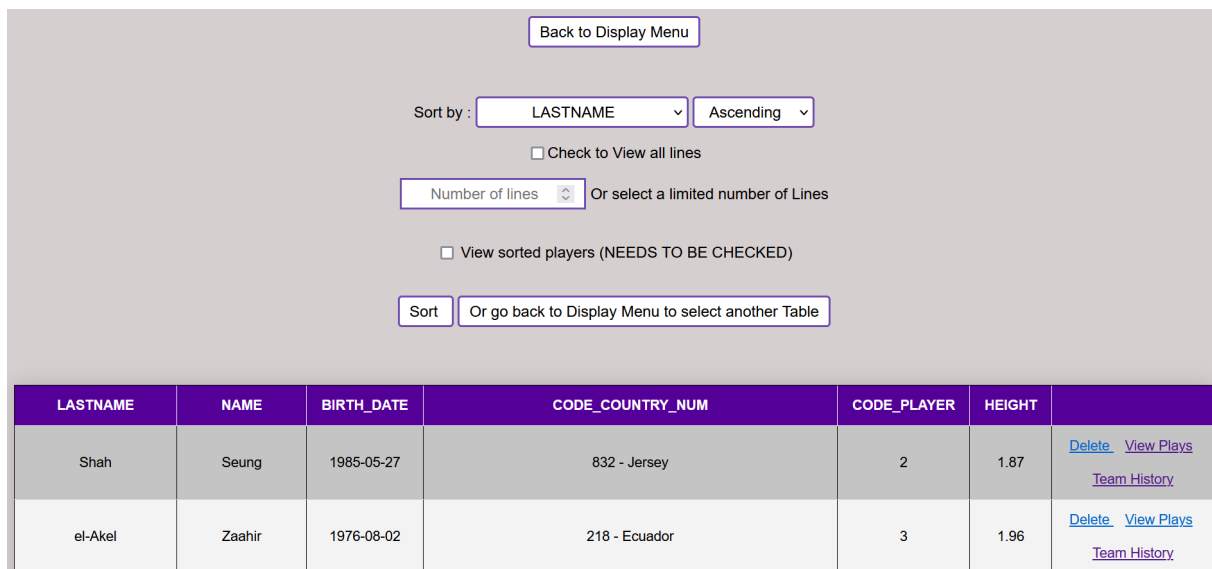
Ayant des connaissances trop limitées en développement d'interface web, nous avons développé une application assez basique et sommes partis du principe que l'utilisateur peut modifier la base de données à sa guise. Nous n'avons donc pas différencié les types d'utilisateurs.

La page d'accueil offre donc plusieurs choix à l'utilisateur :

- il peut choisir de visualiser des données en sélectionnant la table à visualiser et en choisissant ou non de voir l'intégralité de la table ou un nombre limité de lignes,
- il peut choisir de visualiser des tables plus descriptives et n'ayant pas vocation à être modifiées (table des codes de pays, des codes d'actions d'un joueur, et des codes de stades dans une compétition),
- il peut choisir d'ajouter des données en sélectionnant une table.

S'il choisit de visualiser une table, en prenant l'exemple des joueurs, la page suivante donne l'aperçu demandé de la table, et donne l'option de demander un nouvel extrait de la base de données en classant les données suivant un critère. Dans ce cas, il est également possible de demander pour chaque joueur :

- la suppression du joueur
- la visualisation de l'historique des équipes dans lesquelles le joueur a joué
- la visualisation de l'historique des actions d'un joueur dans ses matchs joués




The screenshot shows a web application interface for displaying a table of players. At the top, there is a button labeled "Back to Display Menu". Below it, there are sorting options: "Sort by:" followed by a dropdown menu set to "LASTNAME" and another dropdown set to "Ascending". There is also a checkbox labeled "Check to View all lines" which is currently unchecked. Below this, there is a text input field labeled "Number of lines" with a value of "10" and a button labeled "Or select a limited number of Lines". Further down, there is a checkbox labeled "View sorted players (NEEDS TO BE CHECKED)" which is also unchecked. At the bottom of the form area, there is a "Sort" button and a button labeled "Or go back to Display Menu to select another Table". Below the form area is a table with the following data:

LASTNAME	NAME	BIRTH_DATE	CODE_COUNTRY_NUM	CODE_PLAYER	HEIGHT	
Shah	Seung	1985-05-27	832 - Jersey	2	1.87	<a href="#">Delete</a> <a href="#">View Plays</a> <a href="#">Team History</a>
el-Akel	Zaahir	1976-08-02	218 - Ecuador	3	1.96	<a href="#">Delete</a> <a href="#">View Plays</a> <a href="#">Team History</a>

FIGURE 9 – Fenêtre XAMPP

Pour ce qui est de l'ajout de données, la page suivante demande donc les détails de la ligne et la génère. La figure suivante donne l'exemple de l'ajout d'une équipe.



The screenshot shows a web application interface. At the top, there is a button labeled "Back to Display Menu". Below it, a purple rectangular box contains the text "Please fill in the required attributes:". Inside this box, there are four input fields: a text box labeled "Name", a text box labeled "City", a dropdown menu currently showing "Afghanistan" with a downward arrow, and another dropdown menu labeled "Club" with a downward arrow. At the bottom of the purple box is a button labeled "Add Team".

FIGURE 10 – Fenêtre XAMPP

### 4.3 Présentation du développement de l'application

Le développement de l'application s'est fait en php et en html, avec le code php permettant d'interagir avec la base de données et de générer le code HTML adapté aux requêtes de l'utilisateur.

Pour ordonner les fichiers php, le dossier de l'application est scindé en plusieurs sous dossiers :

- un dossier add contenant les pages permettant d'ajouter des lignes aux tables (on a un fichier par table mais la structure des fichiers reste la même)
- un dossier scripts contenant des scripts php courts auxquels les autres pages font appel : connexion établissant la connexion avec la base de données, delete permettant la suppression de données, getTeams permettant de visualiser plusieurs noms d'équipes et showTables permettant d'afficher des tableaux de données
- un dossier simpleTables permettant d'afficher les trois tables descriptives précédemment évoquées.



#### 4.3.1 Ajout de données

Pour l'ajout des données, la sélection de la table se fait sur la page d'accueil au sein d'un élément HTML `<form>` qui renvoie vers le fichier `add.php` en donnant l'information de la table à afficher.

Le fichier `add.php` est une succession de cas représentant les différents types d'éléments ajoutables. Si l'on regarde le premier cas où l'on ajoute une équipe, on affiche encore une fois un élément HTML `<form>` qui demande les informations à l'utilisateur : le nom de l'équipe, la ville, le pays, et s'il s'agit d'une équipe nationale ou d'un club.

```
echo '<form name="frmContact" class="needs-validation " method="post" action="add/addTeam.php">
<p>
<input type="text" class="styled-text" name="txtName" id="txtName" placeholder="Name" value="" required>
</p>
<p>
<input type="text" class="styled-text" name="txtCity" id="txtCity" placeholder="City" value="" required>
</p>
<p>
<select class="styled-button" name="txtCountry" id="txtCountry" value="" required >;
$resultBig = $con->query('SELECT * FROM COUNTRYS ORDER BY NAME');

while($row = mysqli_fetch_array($resultBig))
{
echo "<option value=" . $row['CODE_COUNTRY_NUM'] . ">" . $row['NAME'] . "</option>";
}
echo '</select>
</p>
<p>
<select class="styled-button" name="txtNat" id="txtNat" value="" required >
<option value="0">Club</option>
<option value="1">National Team</option>
</select>
</p>
<p>&nbsp;</p>
<p>
<input type="submit" name="Submit" id="Submit" value="Add Team" class="styled-button">
</p>
</form>';
```

FIGURE 11 – Ajout d'une équipe

On note que pour le pays, une requête SQL est mise en place pour afficher une liste déroulante avec les éléments de la table `countrys`.

Une fois le formulaire rempli, on fait appel au fichier `addTeam.php` qui exécute la requête SQL.

#### 4.3.2 Affichage de données

Pour l'affichage des données, la sélection se fait encore via un formulaire HTML qui fait ensuite appel au fichier `display.php`. ce fichier remplit deux rôles : il affiche la table demandée par l'utilisateur et il propose à l'utilisateur de classer les données suivant l'une des colonnes du tableau.

Pour ce qui est de l'affichage des données, `display.php` fonctionne en 3 temps :

1. envoi d'une requête SQL pour avoir le nom des en-têtes
2. affichage des en-têtes du tableau
3. envoi d'une seconde requête pour obtenir les données à afficher
4. appel de la fonction showTable codée dans le fichier showTables avec comme arguments le nom de la table et le résultat de la requête SQL

```
44 <?php
45 $sql = "SHOW COLUMNS FROM " . $table;
46 $result = mysqli_query($con, $sql);
47 echo "<table class='styled-table' border='1'>
48 <thead><tr>";
49 while ($row = mysqli_fetch_array($result)) {
50     echo "<th>";
51     echo $row["Field"] . "</th>";
52 }
53
54 echo "<th></th></tr></thead>";
55
56 if(!empty($infinite)){
57     $resultBig = $con->query("SELECT * FROM " . $table);
58     else
59     {$resultBig = $con->query("SELECT * FROM " . $table . " LIMIT " . $lines);
60     }
61
62
63 require "scripts/showTables.php";
64
65 showTable($table, $resultBig);
66 ?>
67
```

FIGURE 12 – Affichage de données

La fonction showTable distingue les différents cas de table et génère un affichage plus lisible que les tables SQL de base. En effet, si la table SQL affiche souvent des informations via des codes, on a préféré compléter cet affichage avec du texte plus évocateur pour l'utilisateur. Ainsi, si l'on prend l'exemple de l'affichage des équipes, on a complété le code du pays de l'équipe par le nom du pays, ce qui passe par une requête HTML. Un autre exemple peut être l'affichage du nom et du prénom d'un joueur plutôt que seulement son code joueur qui est peu évocateur pour un utilisateur humain.

```
7      if ($table == "teams") {
8
9          while ($row = mysqli_fetch_array($resultBig)) {
10             echo "<tr>";
11             echo "<td>" . $row["CODE_TEAM"] . "</td>";
12             echo "<td>" . $row["NAME"] . "</td>";
13             echo "<td>" . $row["CITY"] . "</td>";
14
15             $sql2 =
16                 "SELECT NAME FROM COUNTRYS WHERE CODE_COUNTRY_NUM=" .
17                 $row["CODE_COUNTRY_NUM"];
18             $result2 = $con->query($sql2);
19             $row2 = mysqli_fetch_array($result2);
20             echo "<td>" .
21                 $row["CODE_COUNTRY_NUM"] .
22                 " - " .
23                 $row2["NAME"] .
24                 "</td>";
25             echo "<td>" . $row["NATIONAL_TEAM"] . "</td>";
26             echo "<td>";
27             echo "<a href='scripts/delete.php?choice=search&tab=" .
28                 urlencode($table) .
29                 "&val=" .
30                 urlencode($row["CODE_TEAM"]) .
31                 "&type=" .
32                 urlencode("CODE_TEAM") .
33                 "'> Delete </a>";
34         }
35         echo "</table>";
```

FIGURE 13 – Extrait de la fonction showTable

Une fois l’affichage effectué via display.php l’utilisateur peut donc choisir de visualiser la table classée par rapport à l’une des colonnes. Ce choix se fait encore via un formulaire HTML et le processus d’affichage est réalisée vi la page displayBis.php qui est quasiment une copie de display.php et dont on ne détaillera pas le fonctionnement une seconde fois.

#### 4.3.3 Supression de données

Notons également que c’est ici que l’utilisateur a la possibilité de supprimer des lignes en cliquant sur le bouton Delete sur la colonne de droite. Le script delete.php représenté ci-après exécute ensuite la requête de suppression.

```
10 $value = $_GET["val"];
11 $column = $_GET["type"];
12 $table = $_GET["tab"];
13
14 $sql = "DELETE FROM " . $table . " WHERE " . $column . " = " . $value;
15 $result = mysqli_query($con, $sql);
16
17 try {
18     $result = mysqli_query($con, $sql);
19     echo "Deletion completed";
20 } catch (Throwable $e) {
21     echo "Sorry but this element may not be deleted.";
22 }
23
```

FIGURE 14 – Extrait du fichier delete.php

#### 4.3.4 Affichage des données d'un joueur

Dans le cas de la table player, on peut également cliquer sur les boutons View Plays et Team History affichant respectivement les actions d'un joueur et les équipes pour lesquelles il a joué.

On peut également cliquer sur le bouton View Plays si l'on visualise la table Games si l'on veut afficher les actions d'une rencontre.

Ces boutons font appel aux fichiers viewPlays.php et viewPlayerHistory.php dont le contenu est assez proche des autres manières de représenter les tableaux que nous avons déjà présentés.

## 5 Queries

Cette partie donne les requêtes SQL permettant de répondre aux questions posées dans l'énoncé.

### 5.1 Requête 1

Qui sont les 10 premiers joueurs qui ont totalisé le plus grand nombre de points pour leur équipe nationale ?

```
01 | SELECT
02 | *
03 | FROM
04 | (
05 |     SELECT
06 |         players.lastname,
07 |         countrys.name,
08 |         SUM(
09 |             events.points * games_events.value_
10 |         ) points
```

```
11 | FROM
12 |     games_events ,
13 |     games ,
14 |     events ,
15 |     leagues ,
16 |     players ,
17 |     countrys
18 | WHERE
19 |     games_events.code_player = players.code_player
20 |     and games_events.code_event = events.code_event
21 |     and games_events.code_game = games.code_game
22 |     and games.code_league = leagues.code_league
23 |     and national_team = 1
24 | GROUP BY
25 |     players.lastname ,
26 |     countrys.name
27 | ORDER BY
28 |     SUM(events.points) desc
29 | ) sub
30 | LIMIT
31 |     10;
```

## 5.2 Requête 2

Qui sont les 3 premiers joueurs qui ont le plus grand pourcentage de lancers francs dans une finale du Championnat d'Europe 2002 ?

```
01 | SELECT
02 |     *
03 | FROM
04 |     (
05 |         SELECT
06 |             lastname ,
07 |             game_date ,
08 |             (hits * 100)/ trie porcentaje
09 |         FROM
10 |             (
11 |                 SELECT
12 |                     players.lastname ,
13 |                     games.game_date ,
14 |                     SUM(
15 |                         IF(
16 |                             games_events.code_event, 11, games_events.value_
17 |                         )
18 |                     ) trie ,
19 |                     SUM(
20 |                         IF(
21 |                             games_events.code_event, 7, games_events.value_
22 |                         )
23 |                     ) hits
24 |                 FROM
25 |                     basketball.games_events ,
26 |                     basketball.games ,
27 |                     basketball.players ,
28 |                     leagues
```

```
29 |         WHERE
30 |             games_events.code_event in (7, 11)
31 |             and games_events.code_player = players.code_player
32 |             and games_events.code_game = games.code_game
33 |             and games.code_league = 1
34 |             and games.code_stage = 1
35 |             and games.season = 2016
36 |         GROUP BY
37 |             players.lastname,
38 |             games.game_date
39 |     ) sub
40 |     ORDER BY
41 |         porcentaje desc
42 | ) sub2
43 | LIMIT
44 |     3;
```

### 5.3 Requête 3

Quel club a la taille moyenne la plus élevée ?

```
01 | SELECT
02 |     *
03 | FROM
04 |     (
05 |         SELECT
06 |             teams.code_team,
07 |             teams.name,
08 |             AVG (players.height) OVER (PARTITION BY teams.code_team)
09 |             porentaje
10 |         FROM
11 |             teams,
12 |             basketball.players,
13 |             basketball.players_history
14 |         WHERE
15 |             players.code_player = players_history.code_player
16 |             AND players_history.TO_DATE IS NULL
17 |             AND players_history.code_team = teams.code_team
18 |         ORDER BY
19 |             porentaje DESC
20 |     ) sub
21 | LIMIT
22 |     1;
```

### 5.4 Requête 4

Quel sponsor a sponsorisé le plus grand nombre d'équipes nationales ayant remporté le championnat du monde ?

```
01 | SELECT
02 |     sponsors.name,
03 |     teams.code_team,
04 |     count(*) sponsored
05 | FROM
```

```
06 | (
07 |     SELECT
08 |         code_league ,
09 |         code_game ,
10 |         code_team
11 |     FROM
12 |     (
13 |         SELECT
14 |             games.code_game ,
15 |             players_history.code_team ,
16 |             games.code_league ,
17 |             SUM(
18 |                 games_events.value_ * events.points
19 |             ) goles ,
20 |             RANK () OVER (
21 |                 PARTITION BY games.code_game
22 |                 ORDER BY
23 |                     SUM(
24 |                         games_events.value_ * events.points
25 |                     ) DESC
26 |             ) orden
27 |         FROM
28 |             basketball.games ,
29 |             basketball.games_events ,
30 |             basketball.players_history ,
31 |             basketball.events
32 |         WHERE
33 |             games.code_league = 7
34 |             AND games.code_stage = 1
35 |             AND games.season >= 2011
36 |             AND games_events.code_game = games.code_game
37 |             AND games_events.code_player = players_history.
code_player
38 |             AND players_history.code_team in (games.team_1, games.
team_2)
39 |             AND games.game_date BETWEEN players_history.from_date
40 |             AND coalesce(
41 |                 players_history.TO_DATE ,
42 |                 SYSDATE()
43 |             )
44 |             AND games_events.code_event = events.code_event
45 |             AND events.points > 0
46 |         GROUP BY
47 |             games.code_game ,
48 |             players_history.code_team ,
49 |             games.code_league
50 |         ORDER BY
51 |             goles DESC
52 |     ) sub
53 |     WHERE
54 |         orden = 1
55 | ) winners ,
56 | teams ,
57 | sponsors ,
58 | sponsor_team_history
59 | WHERE
60 |     teams.code_team = winners.code_team
```

```
61 |     and sponsor_team_history.code_team = teams.code_team
62 |     and sponsors.code_sponsor = sponsor_team_history.code_sponsor
63 | GROUP BY
64 |     sponsors.name,
65 |     teams.code_team
66 | ORDER BY
67 |     sponsored DESC;
```

## 5.5 Requête 5

Pour chaque club, quels sont les joueurs qui ont le plus grand pourcentage de 3 points dans la saison en cours ?

```
01 | SELECT
02 |     players,
03 |     team,
04 |     (hits * 100) / trie pourcentage
05 | FROM
06 |     (
07 |         SELECT
08 |             teams.name team,
09 |             players.name players,
10 |             SUM(
11 |                 if (
12 |                     games_events.code_event, 5, games_events.value_
13 |                 )
14 |             ) trie,
15 |             SUM(
16 |                 if (
17 |                     games_events.code_event, 6, games_events.value_
18 |                 )
19 |             ) hits
20 |         FROM
21 |             teams,
22 |             basketball.players_history,
23 |             basketball.players,
24 |             basketball.games_events,
25 |             basketball.games
26 |         WHERE
27 |             teams.code_team = players_history.code_team
28 |             AND players_history.code_player = games_events.code_player
29 |             AND players_history.code_player = players.code_player
30 |             AND players_history.code_team in (games.team_1, games.
team_2)
31 |             AND games.game_date BETWEEN players_history.from_date
32 |             AND coalesce(
33 |                 players_history.TO_DATE,
34 |                 SYSDATE()
35 |             )
36 |             AND games_events.code_event IN (5, 6)
37 |             AND games_events.code_game = games.code_game
38 |             AND games.code_league = 1
39 |             AND games.season = 2011
40 |         GROUP BY
41 |             teams.name,
```



```
42 |         players.name
43 |     ) sub
44 | ORDER BY
45 |     porcentaje DESC;
```

## 5.6 Requête 6

Pour un club donné, quel joueur a le plus grand nombre de passes décisives par match ?

```
01 | SELECT
02 |     team,
03 |     code_game,
04 |     players,
05 |     (assists * 100) / sum(assists) over (partition by code_game)
      porcentaje
06 | FROM
07 |     (
08 |         SELECT
09 |             teams.name team,
10 |             games.code_game,
11 |             players.name players,
12 |             sum(games_events.value_) assists
13 |         FROM
14 |             teams,
15 |             basketball.players_history,
16 |             basketball.players,
17 |             basketball.games_events,
18 |             basketball.games
19 |         WHERE
20 |             teams.code_team = players_history.code_team
21 |             AND players_history.code_player = games_events.code_player
22 |             AND players_history.code_player = players.code_player
23 |             AND players_history.code_team in (games.team_1, games.
team_2)
24 |             AND games.game_date BETWEEN players_history.from_date
25 |             AND coalesce(
26 |                 players_history.TO_DATE, SYSDATE()
27 |             )
28 |             AND games_events.code_game = games.code_game
29 |             AND games_events.code_event = 1
30 |             AND games.code_league = 1
31 |             AND games.season = 2011
32 |         GROUP BY
33 |             teams.name,
34 |             games.code_game,
35 |             players.name
36 |     ) sub
37 | ORDER BY
38 |     porcentaje DESC;
```

## 5.7 Requête 7

Quels sont les clubs qui ont remporté l'Euroleague plus de 3 fois ?

```
01 | SELECT
02 |     code_team,
03 |     COUNT(*) win
04 | FROM
05 |     (
06 |         SELECT
07 |             code_team
08 |         FROM
09 |             (
10 |                 SELECT
11 |                     players_history.code_team,
12 |                     SUM(
13 |                         games_events.value_ * events.points
14 |                     ) goles
15 |                 FROM
16 |                     games,
17 |                     games_events,
18 |                     players_history,
19 |                     events
20 |                 WHERE
21 |                     games.code_league = 1
22 |                     AND games.code_stage = 1
23 |                     AND games_events.code_game = games.code_game
24 |                     AND games_events.code_player = players_history.
code_player
25 |                     AND games.game_date BETWEEN players_history.from_date
26 |                     AND coalesce(
27 |                         players_history.TO_DATE,
28 |                         SYSDATE()
29 |                     )
30 |                     AND games_events.code_event = events.code_event
31 |                     AND events.points > 0
32 |                 GROUP BY
33 |                     players_history.code_team
34 |                 ORDER BY
35 |                     goles DESC
36 |             ) sub2
37 |         LIMIT
38 |             1
39 |     ) sub
40 | GROUP BY
41 |     code_team
42 | HAVING
43 |     COUNT(*) > 3
44 | ORDER BY
45 |     win DESC;
46 | );
```

## 6 Bibliographie

- G. Gardarin. BASES DE DONNÉES, Eyrolles, 2003.
- J.-L. Hainaut. Bases de données : Concepts, utilisation et développement. Dunod,

2022.

- Support de cours MOD 3.2 - Systèmes de bases de données, Centrale Lyon.