Enseirb-matmeca 2020/2021
Département Informatique / 2A

Fédération sportive de Hand-ball

Projet SGBD

Réalisé par

- Mohammed Boudali
- Mohamed Amghar
- Jalal Izekki
- Saad Margoum

Encadré par

M. Sylvain Lombardy

Table des matières

1 Introduction

Ce projet intitulé "Fédération sportive de Hand-ball" a pour objectif de mettre en oeuvre, sur un cas pratique, les notions et les méthodes de construction d'une base de données vues dans le cours.
Le projet démarre par une modélisation des données, et aboutit à la création d'une base de données relationnelle et à l'implémentation d'un certain nombre d'opérations (consultation, mises à jour, etc.) ainsi que d'une interface graphique qui simplifie l'accès à la base de données. Ce rapport à pour but de décrire les modalités de réalisation de ces éléments.

2 Modélisation des données

2.1 Sujet du projet

Dans ce projet, nous avons crée une base de données pour la gestion des rencontres de hand-ball entre des clubs d'une fédération. Un club de hand-ball doit posséder un bureau qui sera dirigé par un ensemble de **résponsables** occupant chacun un poste précis. Chaque club contient plusieurs équipes qui, elles, sont composées de plusieurs joueurs et entraîneurs. Les équipes sont classés en plusieurs catégories et peuvent participer à des rencontres pendant un saison donné. Une rencontre aboutit à un score final et un nombre de fautes commises par chaque joueurs. ces éléments doivent être enregistrés dans la base de données.

2.2 Modèle conceptuel et contraintes fonctionnelles

2.2.1 Description du modèle conceptuel

La réalisation du modèle conceptuel de données est la première étape dans la réalisation du projet ; c'est l'étape de la traduction du sujet en entités et associations.

Notre schéma entités/associations comporte les entités suivantes :

- **Clubs** : un club est défini par un identifiant unique et un autre attribut qui stocke son nom.
- **Responsables** ; un responsable est défini par un identifiant unique, son nom, son prénom, son poste, son email et son numéro de téléphone.
- **Équipes** : Une équipe est définie par son numéro d'équipe, son nom, sa catégorie et son ordre dans cette catégorie.
- **Entraîneurs** : Un entraîneur est identifié par son numéro d'identification, son nom, son prénom, son email et son numéro de téléphone.
- **Joueurs** ; un joueur est identifié par son numéro de licence (attribut unique pour chaque joueur), son nom, son prénom, sa date de naissance, son email et son numéro de téléphone.
- **Date d'entrée** : Cette entité a été introduite pour garder l'historique de changement d'équipe par les joueurs et les entraîneurs. Cette entité contient un seul attribut qui est la date d'entrée.
- **Saisons** : Une saison est définie par un numéro de saison (clé primaire) et sa date de début.
- **Rencontres** : Une rencontre comporte comme attribut son numéro (identifiant unique), sa date de rencontre, le score de l'équipe accueillante et celui de l'équipe reçue. Notre première intuition était de se passer des deux derniers attribut car ils peuvent être calculés à partir de la somme des buts marqués par les joueurs de chaque équipe. Cependant, cette approche n' pas été adoptée pour ne pas complexifier les calculs et les associations.

2.2.2 contraintes fonctionnelles

- Les cardinalités :
  1. Un club contient **1 ou N** responsable(s), un responsable ne peut appartenir qu'à un seul club.
  2. Un club contient **1 ou N** équipe(s), une équipe ne peut appartenir qu'à un seul club.

3. Une équipe peut avoir plusieurs joueurs et entraîneurs, qui ne doivent pas forcément appartenir à l'équipe en même temps (possibilité de changer l'équipe). Une équipe contient **1 ou N** joueur(s) et **0 ou N** entraîneur(s) qui ont rejoint l'équipe en **0 ou N** date(s).
4. Une équipe peut accueillir ou recevoir **1 ou N** rencontre(s).
5. Une rencontre est organisée entre **deux** équipes, une accueillante et une reçue. dans une rencontre,**1 ou N** joueurs participent.
6. **1 ou N** joueur(s) peuvent jouer pendant une saison et pendant chaque rencontre.

— Parmi les dépendances fonctionnelles existante dans ce modèle :
- DF (numéro de licence, numéro de rencontre)$\longrightarrow$nombre de points marqués
- DF (numéro de licence, numéro de rencontre)$\longrightarrow$nombre de fautes commises

2.2.3 Modèle conceptuel

Le schéma de la figure 2 de l'annexe représente le modèle conceptuel obtenu.

2.3 évolution du modèle conceptuel

Au cours de la réalisation du projet, nous avons modifié certains attributs soit pour mieux simplifier la structuration de la base des données et améliorer son fonctionnement. Voici les modifications que nous avons introduit et leurs raisons :

— Dans notre première version, l'entité `date d'entrée` ne faisait pas partie de notre modèle. En effet, il s'avère que la date d'entrée qui était sous forme d'un attribut dans notre premier modèle n'était pas suffisante pour tracer les déplacement d'un joueur d'une équipe à l'autre.

— Dans la première version de notre modèle, l'entité `Saison` contenait un attribut pour stocker la date de fin de la saison. Cependant, la date de fin d'une saison représente la date de début de la saison suivante. Il était donc nécessaire de le supprimer pour ne pas dupliquer les données.

— Dans la première version, l'entité `Rencontre` contenait un attribut : nombre saison. En considérant que chaque saison a une longueur fixe (un an par exemple) et les saisons se suivent, on s'est contenté de chercher la saison dont $date\_de\_rencontre - date\_debut\_saison \leq longueur\_saison$, on a donc pu supprimer l'attribut numéro de saison de l'entité pour éviter la redondance.

3 Schéma relationnel

3.1 Modèle relationnel

Le schéma de la figure 3 de l'annexe représente le modèle relationnel que nous avons réalisé à partir du modèle conceptuel en assurant le respect des règles de cohérence définies sur les données. Il permet de représenter, d'une manière plus concrète, les relations entre les différentes entités de la base de données.

3.2 Contraintes d'intégrité

Afin d'assurer la cohérence relationnelle de la base de données après chaque modification, des contraintes d'intégrité ont été implémentées. Elles concernent notamment les clés primaires et étrangères de chaque entité ainsi que l'action à effectuer après la suppression d'un élément de la base de données :

— Chaque entité du modèle relationnel est traduite en une table qui dispose d'une clé primaire. Cette clé primaire est unique pour chaque instance de l'objet. Pour les joueurs, nous avons choisi de considérer leur numéro de licence comme clé primaire.

— Des contraintes référentielles ont été ajoutée pour indiquer que les valeurs introduites dans une colonne figurent dans une autre colonne comme étant des clés primaires. Ce traitement concerne

notamment les entités RESPONSABLES, EQUIPES, RENCONTRES, SAISONS_JOUEES, COMMENCER_A_JOUER, COMMENCER_A_ENTRAINER et PARTICIPER.

— L'option on_delete avec le comportement cascade a été ajoutée à toutes les entités ayant des références sur d'autres entités pour être supprimées en cascade. Cela veut dire qu'un objet doit être supprimé s'il dispose d'une clé étrangère qui référence un autre objet lorsque celui-ci est supprimé.

3.3 Vérifications des formes normales

Pendant la réalisation du modèle conceptuel, nous avons pris en compte la nécessité de respecter les formes normales :

— Première forme normale : Tout attribut d'un objet ne peut prendre qu'une seule valeur à un instant donné.

— Deuxième forme normale : Toutes les clés primaires sont composée d'un seul champ sauf pour les cas suivants :

  — Dans les entités COMMENCER_A_JOUER, COMMENCER_A_ENTRAINER, SAISONS_JOUEES, la clé primaire est composée de la liste de tous les attributs.

  — Dans l'entité PARTICIPER, les attributs NOMBRE_POINTS_MARQUES et NOMBRE_FAUTES_COMMISES dépendent du joueur aussi bien que de la rencontre, et donc de la clé primaire (NUMERO_DE_RENCONTRE, NUMERO_DE_LICENCE) en entier.

  La deuxième forme normale est donc respectée par toutes les entités.

— Troisième forme normale : les attributs de toutes les entités ne dépendent que de la clé primaire.

4 Implémentation

4.1 Les outils de développement utilisés

Le système de gestion de bases de données que nous avons choisi pour implémenter la base de données est **MySQL**.

4.2 Requêtes

Dans ce projet nous avons implémenté quatre types de requêtes :

1. **Requêtes structurelles** :
   Ces requêtes permettent la création et la suppression des tables ainsi que l'ajout des contraintes qui assurent la cohérence des données et le bon fonctionnement des applications qui utilisent la base de données .
2. **Requêtes de consultation** :
   Elles sont utilisées pour consulter des données qui existent dans la base de données et créer des tables qui contiennent plus d'informations par utilisation des formules de l'algèbre relationnelle comme les jointures, les projections, le produit cartésien etc. . .
3. **Requêtes de statistiques** :
   Ces requêtes permettent de créer des tables qui fournissent des statistiques comme les classements des joueurs, des équipes etc. . ..
   Un exemple de ces requêtes est la requête du classement des équipes dans une saison :

```
create or replace view TAB (SAISON,NUM_EQUIPE,RENCONTRE,DATE,W,L,N)
    as select S.NUMERO_SAISON SAISON,
    R.NUMERO_EQUIPE_ACCUEILLANTE as NUM_EQUIPE,
    R.NUMERO_DE_RENCONTRE as RENCONTRE,
    R.DATE_DE_RENCONTRE as DATE,
    count(case when R.SCORE_EQUIPE_ACCUEILLANTE>R.SCORE_EQUIPE_RECUE then 1
```

```
    else null end) as W,
    count(case when R.SCORE_EQUIPE_ACCUEILLANTE<R.SCORE_EQUIPE_RECUE then 1
    else null end) as L,
    count(case when R.SCORE_EQUIPE_ACCUEILLANTE=R.SCORE_EQUIPE_RECUE then 1
    else null end) as N
    from RENCONTRES R,SAISONS S
    where DATEDIFF(R.DATE_DE_RENCONTRE,S.DATE_DE_DEBUT) < 365
    and DATEDIFF(R.DATE_DE_RENCONTRE,S.DATE_DE_DEBUT)>0
    group by 1,2,3,4

    union

    select S.NUMERO_SAISON SAISON,
    R.NUMERO_EQUIPE_RECUE as NUM_EQUIPE,
    R.NUMERO_DE_RENCONTRE as RENCONTRE,
    R.DATE_DE_RENCONTRE as DATE,
    count(case when R.SCORE_EQUIPE_ACCUEILLANTE<R.SCORE_EQUIPE_RECUE then 1
    else null end) as W,
    count(case when R.SCORE_EQUIPE_ACCUEILLANTE>R.SCORE_EQUIPE_RECUE then 1
    else null end) as L,
    count(case when R.SCORE_EQUIPE_ACCUEILLANTE=R.SCORE_EQUIPE_RECUE then 1
    else null end) as N
    from RENCONTRES R, SAISONS S
    where DATEDIFF(R.DATE_DE_RENCONTRE,S.DATE_DE_DEBUT) < 365
    and DATEDIFF(R.DATE_DE_RENCONTRE,S.DATE_DE_DEBUT)>0
    group by 1,2,3,4
;

% pour un classement dans une saison donnée

select TAB.NUM_EQUIPE,E.NOM_EQUIPE,SUM(TAB.W) as W,SUM(TAB.L) as L,SUM(TAB.N) as N
    from TAB , EQUIPES E where E.NUMERO_EQUIPE = TAB.NUM_EQUIPE
    and TAB.SAISON = 2
    group by 1,2
    order by 3 DESC, 4 ASC ,5 DESC;

```

nous avons d'abord crée une vue `TAB` qui contient pour chaque équipe tous les matchs joués dans une saison avec les résultats de la rencontre dans 3 colonnes **W,L,N** qui correspondent respectivement à un match gagné, un match perdu et un match nul, la valeur d'un champ de ces colonnes est soit 0 ou 1 selon qu'il s'agisse du résultat du match ou non. Par exemple si une équipe a gagné un match $W = 1$, $L = 0$ et $N = 0$. Ensuite, nous avons effectué une somme sur les colonnes de chaque équipe afin de calculer le nombre des matchs gagnés, perdus et nuls dans une saison donnée, et finalement nous avons classé les équipes par ordre de W descendant, L ascendant, N descendant.

4. **Requêtes de mise à jour** : ces requêtes permettent l'ajout, la suppression et la mise à jour des lignes d'une table. Un exemple d'une requête permettant de supprimer un joueur est le suivant :

```
        delete from JOUEURS where NOM_JOUEUR = 'JEAN';
        commit;

```

5 Interface

Une interface graphique a été implémentée pour simplifier la consultation et la modification de la base de données ainsi que la visualisation de certaines statistiques. Cette interface a été réalisée avec l'utilisation

du framework **Django** ; il s'agit d'un outil haut-niveau qui ne manipule pas directement les requêtes SQL. Cependant, nous avons fait le choix d'utiliser le module `connections` qui permet d'exécuter des requêtes SQL directement dans les vues et récupérer leur résultats dans un tuple. Toutes les requêtes utilisées dans l'interface peuvent donc être trouvées dans le fichier `views.py` dans le répertoire `mainApp`.

5.1 Étapes de la réalisation

Dans un permier temps, il a été important de savoir comment lier l'application `Django` avec la base de données SQL. Pour ce faire, nous avons modifier la variable `DATABASES` du fichier `settings.py` de la manière suivante :

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'fshb',
        'USER': 'root',
        'PASSWORD': '',
    }
}
```

Cette modification permet de remplacer la base de données par défaut de l'application **Django** par la nouvelle base appelée `fshb` que nous avons créée à l'utilisation de MySQL. Les attributs `USER` et `PASSWORD` doivent contenir respectivement le nom de l'utilisateur et le mot de passe du serveur MySQL. Pour pouvoir utiliser les entités définies dans la bases de données comme étant des objets, nous avons utilisé la commande suivante :

```
$  python3 manage.py inspectdb > mainApp/models.py
```

Cette commande permet de créer des modèles en introspectant la base de donnés `fshb` et les copier dans le fichier `mainApp/models.py`.
Ensuite, nous avons lancé la commande `migrate` pour installer les enregistrements de base de données supplémentaires, comme les permissions d'administration et les types de contenus :

```
$  python3 manage.py migrate
```

La base de données est donc désormais prête à être utilisée par `Django`.

5.2 fonctionnalités de l'interface

L'interface permet de visualiser les données de la base d'une manière simplifiée. Plusieurs vues ont été implémentées pour afficher la liste des clubs, des rencontres, des équipes de chaque club, des joueurs de chaque équipes ainsi que les données de chacun de ces éléments. D'autres vues permettent d'ajouter de nouvelles données à la base ou d'en supprimer ; il est possibles de créer ou supprimer de nouveaux joueurs, clubs, équipes, responsables, entraîneurs, rencontres depuis l'interface (Cf. la figure 1). Enfin, nous avons ajouter des vues pour permettre de visualiser quelques statistiques à une date ou une saison donnée.

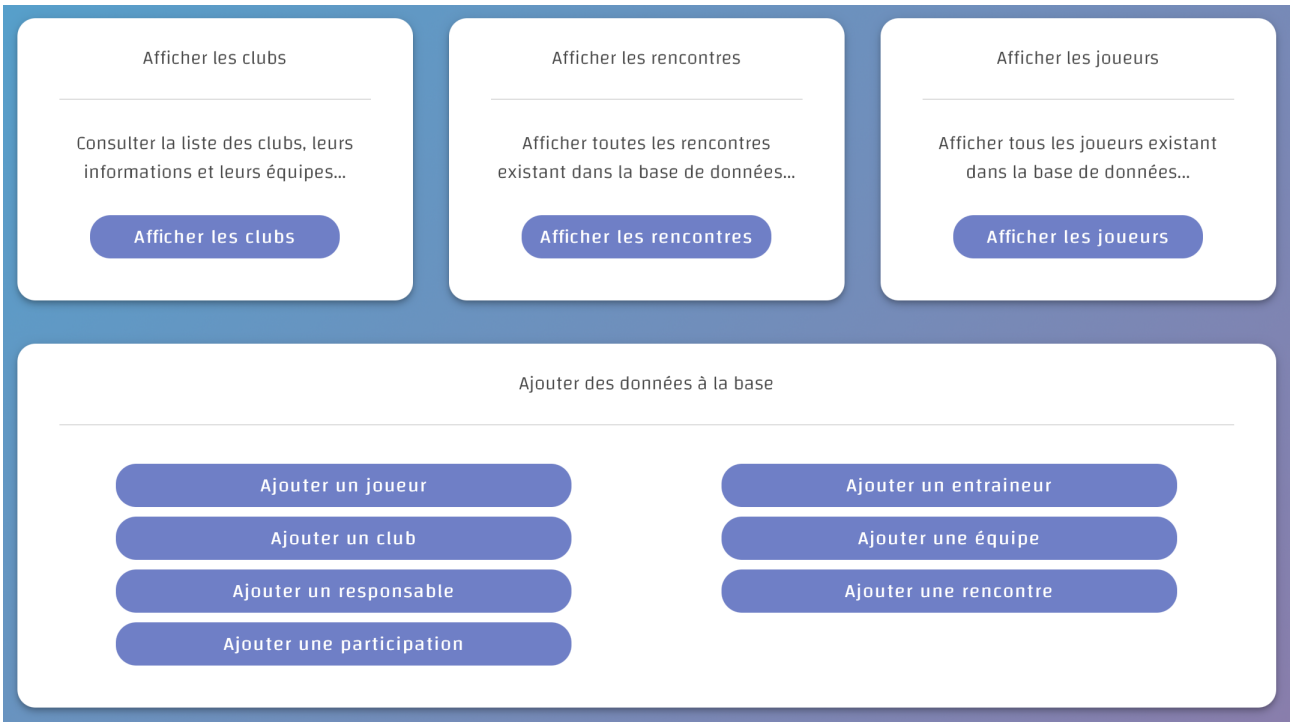

FIGURE 1 – Certaines fonctionnalités de l'interface graphique

6 Conclusion

Pour conclure, le projet a été très utile pour nous. En effet, il a servi comme introduction aux systèmes de gestion de bases de données, et leurs diverses applications dans la vie réelle. Il a été simple et plutôt intuitive, mais en même temps il nous a permis de bien apprendre à manipuler les bases de données. Il nous a aussi permis de découvrir de nouveaux outils comme `Django` pour présenter les données d'une manière simplifiée.

La majeure partie du temps que nous avons consacré au projet a été dans le but d'établir la modélisation des données et d'éviter les redondances tout en respectant les règles de normalisation.

Parmi les améliorations du projet qui peuvent être envisagées, on cite les suivantes :

— Ajouter des "triggers" qui permettent de déclencher des erreurs en cas d'incohérence dans les données (exemple : si le numéro de l'équipe accueillante est égal à celui de l'équipe reçue ..).
— L'enrichissement du modèle de la base de données notamment par l'ajout de quelques détails (comme tickets de matches, leurs prix, le nombre de spectateurs, le nom de l'arbitre ..) qui pourraient être utiles dans la vie réelle.
— L'ajout de certaines autres requêtes. Par exemple : le nombre des joueurs d'une équipe ou d'un club, les transferts des joueurs entre équipes etc...
— la dénormalisation de la base données peut optimiser les performances (éviter de faire plusieurs jointures ,etc..), mais elle présente des risques d'incohérence.

Annexes

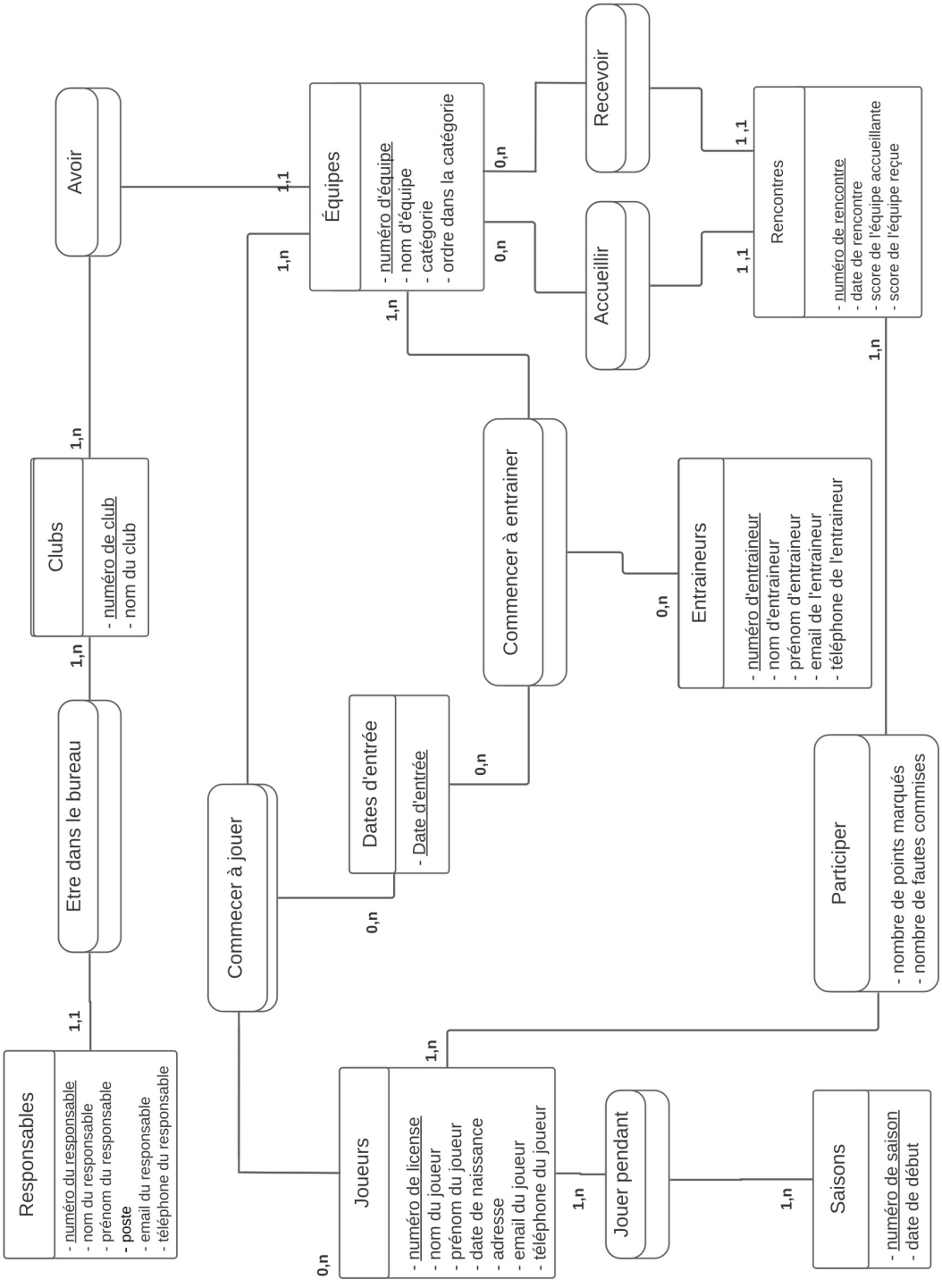

FIGURE 2 – Modèle conceptuel de la base de données

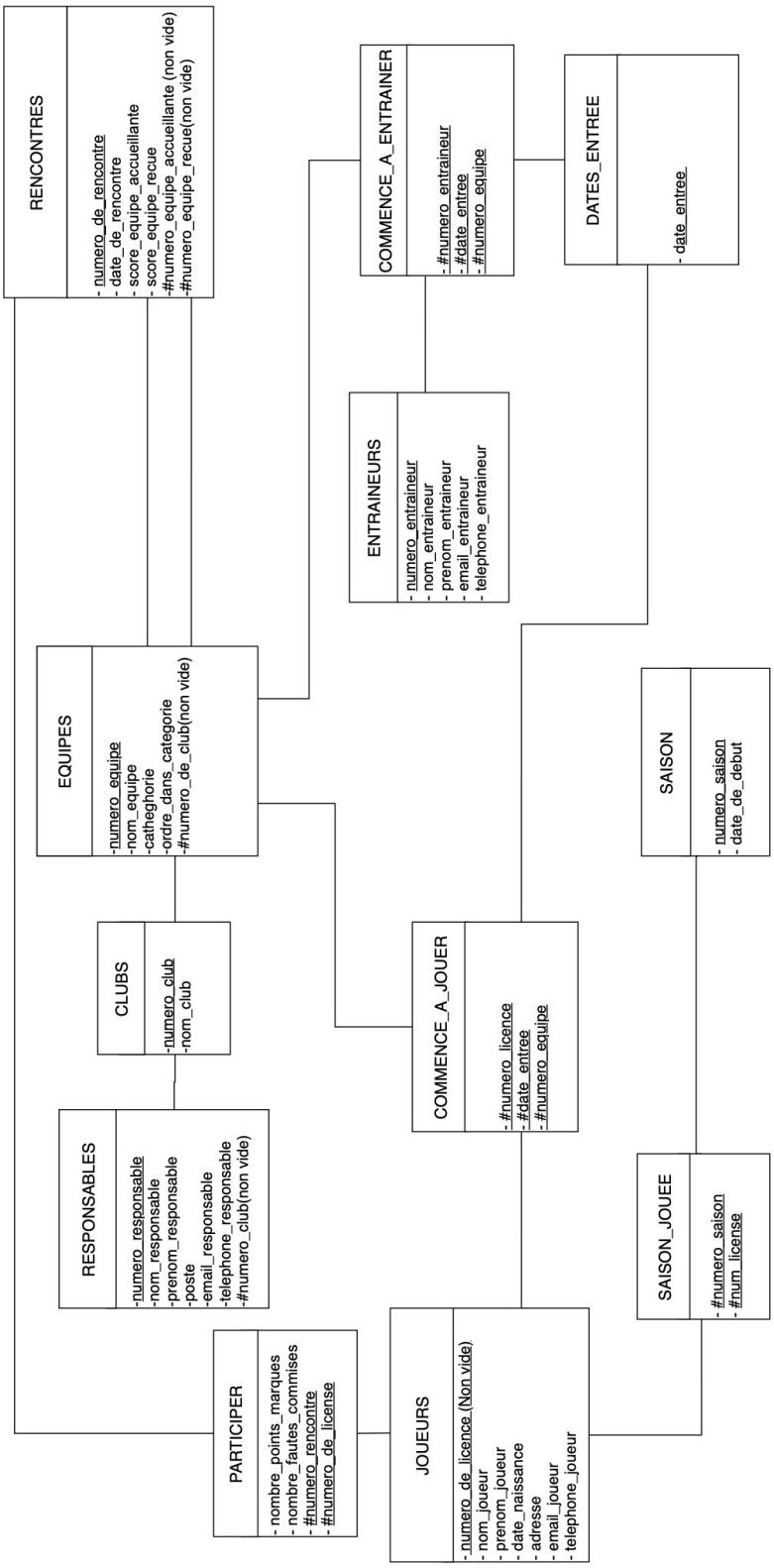

FIGURE 3 – Modèle relationnel de la base de données