Fédération sportive de Hand-ball

Rapport de projet SGBD

Projet de gestion de bases de données - S7

– I2 - ENSEIRB-MATMECA –

Encadrants : M. Sylvain Lombardy - M. Mohamed Mosbah

Par Ismail Elomari alaoui - Hamza Benmendil - Mohamed Fayçal Boullit - Mouhcine EL Hammadi

8 décembre 2020

Table des matières

Introduction

Contexte

L'objectif du projet est de mettre en œuvre, sur un cas pratique, les notions et les méthodes vues dans le module *SGBD*. 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 (consultations, mises à jour, etc.). Pour produire une base de données propre et robuste, des hypothèses et des contraintes seront fixés, au cas de manque d'information ou d'ambiguïté. Ces hypothèses seront bien évidement expliqués dans ce rapport.

Description du projet

Ce projet consiste à réaliser une base de données gérant les rencontres de $hand-ball$ entre les clubs d'une fédération. Plusieurs informations sur les contraintes entre différentes entités (clubs, joueurs, équipes, ...) ont été données. Nous devrons ensuite produire plusieurs bouts de codes et mécanismes qui assurent les contraintes d'intégrité et toute autre contrainte ou hypothèse qui nous semble logique. L'idée sublime est de pouvoir stocker toutes ces informations dans la base, et pouvoir disposer plusieurs requêtes de services (consultation, statistiques et mise à jour).

1 Modélisation des données

1.1 Description du contexte de l'application (entités, associations, règles de gestion)

On était amené à introduire tout un nombre d'entités et associations afin de répondre aux exigences de l'énoncé et concevoir un modèle conceptuel permettant de modéliser la base de données à faire.
Les entités qu'on a trouvé nécessaires à fournir sont les suivantes :

- Personnes *et ses entités dérivantes :*
  - Joueurs
  - Entraineurs
- Clubs
- Catégories
- Equipes
- Rencontres
- Saisons

Un nombre d'associations est donc ajouté afin de relier ces entités, et d'exprimer les différentes relations qu'elles pourraient avoir entre elle et finalement produire un modèle entité-association cohérent avec ce qui est demandé.

1.2 Modèle entité-association

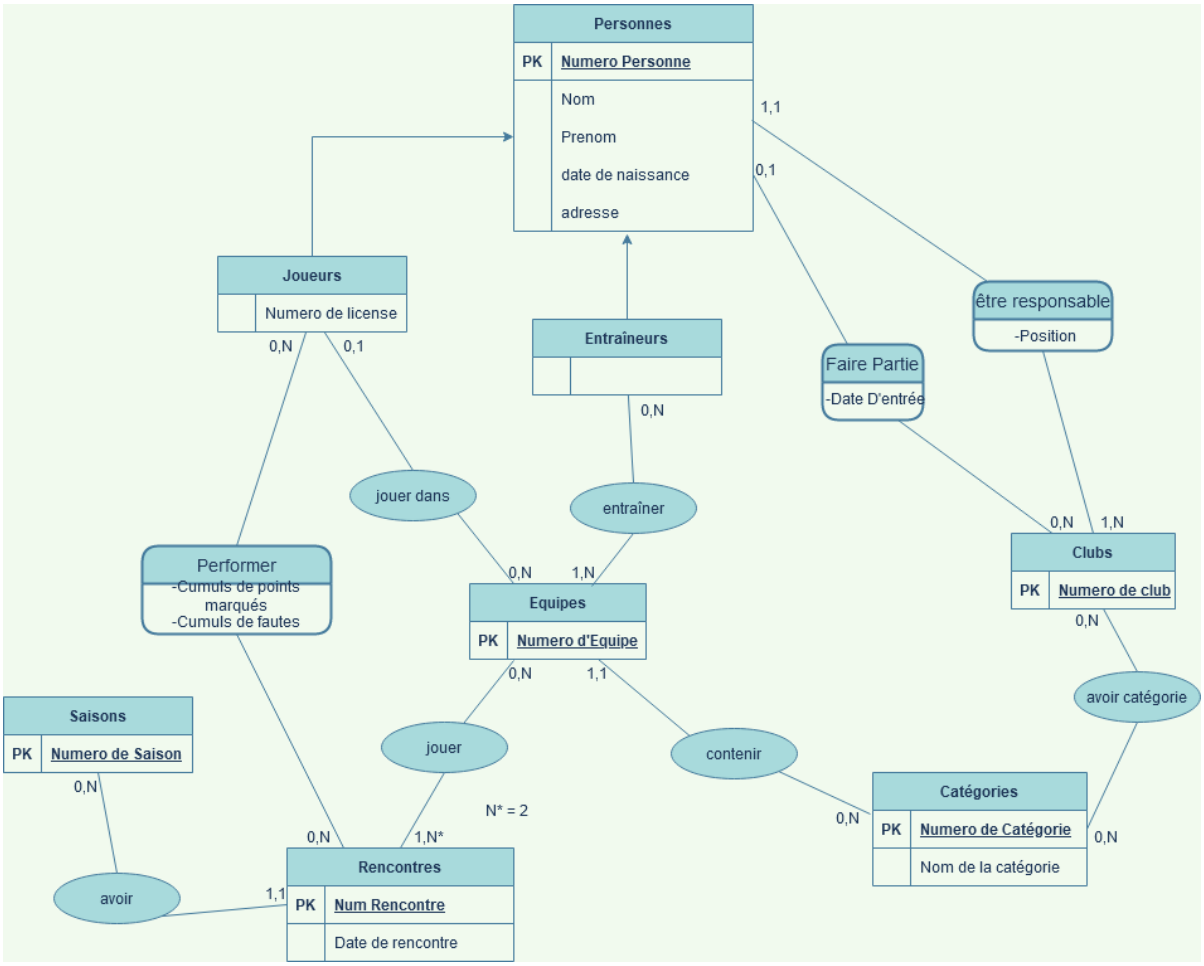

FIGURE 1 – Version prototype du schéma conceptuel

Un nombre de changements a été fait sur la version prototype avant d'arriver à la version finale et ceci pour des raisons qui s'imposaient :

En fait, pour sauvegarder l'historique des performances des joueurs à chaque rencontre, un problème flagrant se présentait devant nous lors de la première implémentation du modèle conceptuel : l'historique du joueur reste intact malgré le changement des équipes, or l'historique des rencontres se modifie et affiche des résultats incohérents puisqu'il prend en considération le changement fait ce qui n'est pas réaliste.

On a fait appel à une association ternaire Performer qui offre la solution souhaitée, or, les attributs de Performer ne dépendent pas de la clé principale de l'entité Equipes ce qui exige une liaison de Performer à une CIF. Ceci produit un deuxième chemin de l'entité Equipes à Rencontres, ce qui nous force à supprimer l'association jouer qui les liaient afin de normaliser le modèle.

Le premier modèle souffrait d'une autre anomalie qui s'affichait lors des résultats. Les équipes de même catégorie étaient tous partie de tous les clubs en même temps, ce qui n'est pas réaliste, c'est la raison pour laquelle on a dû changer les associations qui liaient les trois entités Equipes, Catégories et Clubs pour qu'il y a un lien direct entre Equipes et chacune des autres entités.

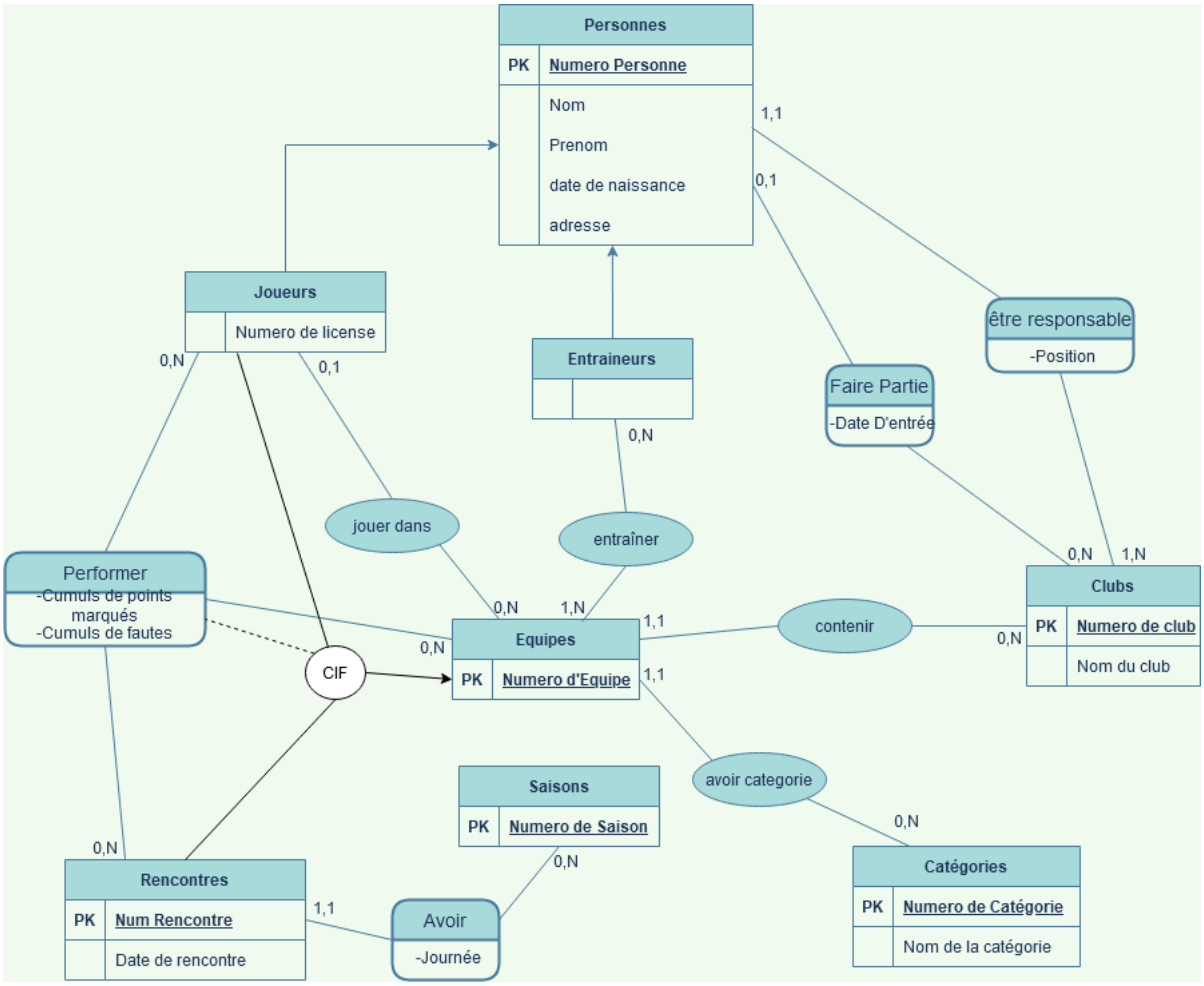

FIGURE 2 – Version finale du schéma conceptuel

1.3 Liste des opérations prévues sur la base (consultation, mise à jour, statistiques)

Après avoir conçu le modèle conceptuel de la base et implémenter son schéma relationnel, on cherche à pouvoir effectuer les opérations suivantes à travers des requêtes :

- **Consultation :** Cette opération nous permet de pouvoir accéder aux diverses informations sur les entités implémentées que ce soit les joueurs, les clubs, les équipes, etc...
- **Statistiques :** Une opération qui s'explique, elle va nous permettre de calculer les moyennes des buts lors de différentes situations, aussi bien que le classement des joueurs, équipes et clubs.
- **Mise à jour :** est une opération qui nous permettra de mettre à jour la base de données par le biais d'ajout, suppression et modification des différentes entités.

2 Schéma relationnel

2.1 Passage au relationnel

Il fallait traduire alors le modèle conceptuel qu'on a sous les mains en un schéma relationnel, ceci est fait comme suit :

D'abord, on traduit les entités en des tables : PERSONNE(**NUMERO PERSONNE**, NOM PERSONNE, ...), CLUB(), CATEGORIE(), SAISON(), RENCONTRE()...

En particulier, comme l'entité JOUEUR hérite de PERSONNE elle hérite aussi sa clé primaire et devient une clé primaire et étrangère : JOUEUR(#**NUMERO PERSONNE**, NUMERO LICENCE,...) .

De plus, on s'abstient de créer une table pour l'entité Entraîneurs puisqu'elle ne nous sert pas à grande chose pendant l'implémentation des opérations qu'on a prévu a faire et à appliquer à la base. Cependant, on a créé les tables ENTRAINER(#**NUMERO PERSONNE**, #**NUMERO EQUIPE**) et PERFORMER() des associations équivalentes dans le modèle conceptuel.

En effet, comme l'association Performer est une association ternaire liée à une CIF, on la traduit par une table qui s'écrit de la façon suivante : PERFORMER(#**NUMERO RENCONTRE**, #**NUMERO PERSONNE**, #**NUMERO EQUIPE**,...), alors que l'association Entraîner est de cardinalité n, m et est donc representée par une table ici.

2.2 Contraintes d'intégrité, dépendances fonctionnelles

Afin que les données saisies dans la base soient conformes aux données attendues par la base, plusieurs contraintes ont été assurées dès la création des tables.

En fait, on utilise le mot clé **not null** pour assurer que les clés primaires et étrangères ne pouvant avoir une valeur nulle, ainsi que les champs d'attributs nécessaires à la définition d'une entité.

Le mot clé **constraint** nous a servi essentiellement pour les contraintes liées aux clés primaires et clés étrangères ainsi au niveau de détection des erreurs en renommant les contraintes, ci-dessous un exemple d'une contrainte sur la table PERFORMER ayant une clé étrangère `NUMERO_RENCONTRE` de la table RENCONTRE.

```
1 alter table PERFORMER
2     add constraint fk1_performer foreign key (NUMERO_RENCONTRE)
3         references RENCONTRE (NUMERO_RENCONTRE);
```

Afin d'assurer un modèle relationnel en 3e forme normale, nous devons s'assurer que les attributs de chaque entité dépendent fonctionnellement de la ou les clés primaires de cette entité. De plus, il faut s'assurer qu'il n'existe aucune dépendance fonctionnelle entre deux colonnes en dehors de ces fameuses clés primaires. Toutes ces points ont été pris en compte pendant la conception de notre modèle relationnel. Pour vous convaincre, on prendra la table la plus "compliquée" de notre modèle PERFORMER : Les champs de cette table de base (modèle conceptuel) sont ceux des cumul de points et de fautes. Ces champs dépendaient fonctionnellment de l'identifiant des tables RENCONTRE et JOUEUR, mais pas de celui de EQUIPE. C'est pourquoi la clé primaire de cette dernière table

n'est qu'une clé étrangère dans la table PERFORMER. En effet, cela est mis en évidence par l'utilisation de CIF dans le modèle conceptuel (figure 2), comme ceci est déjà expliqué dans la partie correspondante.

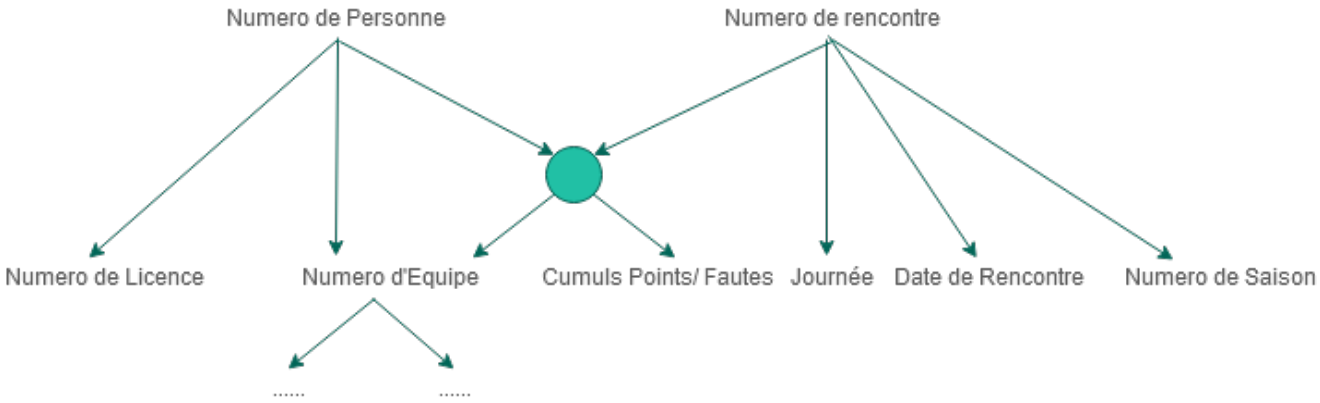

FIGURE 3 – Graphe de couverture des dépendances fonctionnelles de l'association Performer

2.3 Schéma relationnel en 3^e *forme normale*

+Personnes(**Num_Personne**, Nom, Prenom, Date_de_naissance, Adresse, #Num_Club *(non vide)* , Position, Date Entrée)
+Clubs(**Num_Club,** Nom de Club**)**
+Equipes(**Num_Equipe** , #Num_Catégorie *(non vide)*, #Num_Club*(non vide)*)
+Catégories(**Num_Catégorie**, Nom de la catégorie)
+Rencontres(**Num_Rencontre**, Date de rencontre, #Num_saison *(non vide)*, Journee)
+Saisons(**Num_Saison**)
+Joueurs(**#Num_Personne,** Numero de license, #Num_Equipe)
+Performer(**#Num_Rencontre, #Num_Personne**, Cumuls de points marqués, Cumuls de fautes, #Num_Equipe)
+Entraîner(**#Num_Personne, #Num_Equipe)**

FIGURE 4 – Version finale du modèle relationnel

3 Implantation (base *SQL* : *Oracle*)

3.1 Création de la base de données, en prenant en compte les contraintes d'intégrité (scripts de création, suppression, insertion)

On fait appel à des scripts `.sql` pour créer la base de données, y insérer des données et les supprimer quand nécessaire.

Le script `base.sql` est responsable de la création de la base, mais aussi de la suppression de celle qui la précède en faisant appel à l'instruction **drop** au tout début du fichier sur toutes les tables qu'elle contenait, pour pouvoir

ainsi créer une nouvelle.

Le script `données.sql`, de sa part, est capable d'insérer de nouvelles données à la table. En particulier, ceci est fait par l'instruction **insert**, mais ce qui est plus pertinent et intéressant est le fait que les joueurs, les entraîneurs et les responsables héritent tous des personnes et ainsi ont sa clé primaire comme clé primaire et étrangère chacun d'eux, ce qui exige à l'insertion d'une donnée de chacune de ces 3 tables soit de lui faire attribuer une personne(et donc sa clé primaire) si possible, soit de créer une personne et incrémenter la sienne. Ce processus est fait de façon automatique grâce à l'utilisation d'une séquence appelée **AUTO_INCREMENTE**, définie dans le script `base.sql` et utilisée dans `données.sql` lors de l'insertion.

3.2 Implémentation des commandes SQL réalisant les opérations retenues

L'implémentation des opérations de consultation et de statistiques qu'on entend appliquer à la base a été faite grâce aux requêtes SQL qu'on peut trouver dans le fichier `requetes.sql`. Ces requêtes utilisent dans leur grande majorité des vues **views** crées dans le script `base.sql`, qui nous simplifient la tâche et qui font la base de notre travail. En effet, l'opération consultation nous offre un nombre d'options qu'on se permet de citer :

1. Liste des clubs
2. Liste des équipes
3. Liste des joueurs
4. Liste des joueurs à une date donnée
5. Résultat des rencontres à une date
6. Feuille d'un match
7. Feuille résumé d'un match
8. Performance d'une équipe
9. Performance des clubs

Comme précédemment dit, l'utilisation des vues était primordiale à la création des requêtes des opérations qu'on applique sur la base. Ceci est au mieux vu dans l'exemple de la requête suivante qui fournit le résultat des rencontres à une date.

```
SELECT NUMERO_RENCONTRE, NUMERO_EQUIPE1, SCORE_EQUIPE1, SCORE_EQUIPE2, NUMERO_EQUIPE2
FROM SCORE_MATCH
WHERE DATE_RENCONTRE = '11-SEP-20';
```

```
-- Score de l'équipe 1 dans un match à tous les dates
create or replace view SCORE_EQUIPE1 as
SELECT R.NUMERO_RENCONTRE as NUMERO_RENCONTRE, R.DATE_RENCONTRE as DATE_RENCONTRE, R.
    NUMERO_EQUIPE1, sum(P.CUMUL_POINT) as SCORE, sum(P.CUMUL_FAUTE) as FAUTE, max(PERS.DATE_ENTREE
    ) as DATE_ENTREE_MAX
FROM RENCONTRE R, PERFORMER P, JOUEUR J, PERSONNE PERS
```

```
WHERE P.NUMERO_RENCONTRE = R.NUMERO_RENCONTRE AND P.NUMERO_PERSONNE = J.NUMERO_PERSONNE AND R.
    NUMERO_EQUIPE1 = J.NUMERO_EQUIPE AND PERS.NUMERO_PERSONNE = J.NUMERO_PERSONNE
GROUP BY R.NUMERO_RENCONTRE, R.NUMERO_EQUIPE1;


-- Score de l'équipe 2 dans un match à tous les dates
create or replace view SCORE_EQUIPE2 as
SELECT R.NUMERO_RENCONTRE as NUMERO_RENCONTRE, R.DATE_RENCONTRE as DATE_RENCONTRE, R.
    NUMERO_EQUIPE2, sum(CUMUL_POINT) as SCORE, sum(CUMUL_FAUTE) as FAUTE, max(PERS.DATE_ENTREE) as
     DATE_ENTREE_MAX
FROM RENCONTRE R, PERFORMER P, JOUEUR J, PERSONNE PERS
WHERE P.NUMERO_RENCONTRE = R.NUMERO_RENCONTRE AND P.NUMERO_PERSONNE = J.NUMERO_PERSONNE AND J.
    NUMERO_EQUIPE = R.NUMERO_EQUIPE2 AND PERS.NUMERO_PERSONNE = J.NUMERO_PERSONNE
GROUP BY R.NUMERO_RENCONTRE, R.NUMERO_EQUIPE2;


-- Score d'un mactch à tous les dates
create or replace view SCORE_MATCH as
SELECT R.NUMERO_RENCONTRE, R.DATE_RENCONTRE as DATE_RENCONTRE, R.NUMERO_EQUIPE1, SE1.SCORE as
    SCORE_EQUIPE1, SE2.SCORE as SCORE_EQUIPE2, R.NUMERO_EQUIPE2
FROM RENCONTRE R, SCORE_EQUIPE1 SE1, SCORE_EQUIPE2 SE2
WHERE R.NUMERO_RENCONTRE = SE1.NUMERO_RENCONTRE AND R.NUMERO_RENCONTRE = SE2.NUMERO_RENCONTRE;
```

L'opération statistique propose de sa part une multitude d'options :

1. Moyenne des points marqués par rencontre à une date donnée
2. Moyenne des points marqués depuis le début de la saison
3. Classement des meilleurs joueurs d'une journée pour une catégorie
4. Classement des équipes

Nommément, l'exemple suivant est celui de la requête la plus compliqué dans toutes les opérations, celle du classement des meilleurs joueurs d'une journée pour une catégorie. L'utilisation de la vue **FEUILLE_MATCH** était fondamentale à son exécution.

```
SELECT FM.NUMERO_PERSONNE, FM.CUMUL_POINT, FM.CUMUL_FAUTE
FROM RENCONTRE R JOIN FEUILLE_MATCH FM ON R.NUMERO_RENCONTRE = FM.NUMERO_RENCONTRE
    JOIN JOUEUR J ON FM.NUMERO_PERSONNE = J.NUMERO_PERSONNE
    JOIN EQUIPE E ON E.NUMERO_EQUIPE = J.NUMERO_EQUIPE
WHERE R.DATE_RENCONTRE = '18-SEP-20' AND E.NUMERO_CATEGORIE = 1
ORDER BY FM.CUMUL_POINT DESC, FM.CUMUL_FAUTE ASC;
```

```
create or replace view FEUILLE_MATCH as
SELECT R.NUMERO_RENCONTRE, R.DATE_RENCONTRE as DATE_RENCONTRE, Pe.NUMERO_PERSONNE as
    NUMERO_PERSONNE, Pe.NOM_PERSONNE, Pe.PRENOM_PERSONNE, J.NUMERO_LICENCE, J.NUMERO_EQUIPE, P.
    CUMUL_POINT, P.CUMUL_FAUTE
```

```
3 FROM PERSONNE Pe, JOUEUR J, RENCONTRE R, PERFORMER P
4 WHERE Pe.NUMERO_PERSONNE = J.NUMERO_PERSONNE AND P.NUMERO_PERSONNE = Pe.NUMERO_PERSONNE AND R.
      NUMERO_RENCONTRE = P.NUMERO_RENCONTRE;
```

4 Utilisation

4.1 Description de l'environnement d'exécution

Afin de pouvoir utiliser le programme, il est nécessaire de le compiler et l'exécuter dans la machine `oracle` de l'ENSEIRB-MATMECA pour avoir accès à notre base déjà créée sur le serveur. Sinon, il faut changer le `nom d'utilisateur`, le `mot de passe` et l'`URL` sur le fichier `Main.java` pour se connecter sur votre serveur local. Dans ce cas, il faut créer d'abord les tables et les données en exécutant les scripts `base.sql` et `donnees.sql` manuellement avant d'exécuter le projet.

4.2 Notice d'utilisation

Après le réglage de l'environnement de l'exécution, il suffit de taper `make` pour compiler le programme, puis `make run` pour l'exécuter. Une dernière règle, `make clean`, est utilisée pour supprimer es fichiers binaires et `html` précédemment crées.

4.3 Description des interfaces éventuelles (JDBC, HTML)

Dans notre projet, nous avons choisi d'utiliser le langage `java` avec la librairie `JDBC`. Le programme établit une connexion avec une base oracle puis affiche un menu sous forme d'un texte contenant 3 opérations ; consultation, statistiques et mise à jour. Selon le choix, un nouveau menu apparaît avec les opérations à effectuer.

— Dans le cas des opérations de la consultation ou des statistiques, un fichier `html` est généré dans un répertoire nommé `html` dans la racine du projet qui contient le résultat de l'opération sous forme d'un tableau.

— La dernière opération donne la possibilité de mettre à jour la base soit en ajoutant, supprimant ou modifiant une entité. Cette mise à jour prend en considération l'ajout et la suppression d'une entité qui dépend de l'existence d'une autre. Par exemple, la suppression d'une personne implique une suppression du joueur, si cette personne est bien évidemment un joueur, portant le même `NUMERO_PERSONNE`. De plus, un changement de club par un joueur implique aussi la modification de la date d'entrée au club, qui est un attribut de la table PERSONNE.

NUMERO_PERSONNE	NOM_PERSONNE	PRENOM_PERSONNE	DATE_DE_NAISSANCE	ADRESSE	NUMERO_CLUB	POSITION	DATE_ENTREE
1	SAUTET	CLAUDE	1994-01-01	null	1	null	2018-01-01
2	PINOTEAU	CLAUDE	1989-02-21	null	1	null	2019-01-03
3	DAVOY	ERIC	1993-10-31	null	1	null	2017-01-08
4	ZIDI	CLAUDE	1996-03-21	null	1	null	2019-01-09
5	AUTAN-LARA	CLAUDE	2000-01-11	null	1	null	2019-01-10
6	ROHMER	ERIC	1998-01-21	null	1	null	2019-01-12
7	MALLE	LOUIS	1997-06-22	null	1	null	2018-01-02
8	BESSON	LUC	1993-11-26	null	1	null	2017-01-02
9	PREMINGER	OTTO	1997-12-03	null	1	null	2019-01-06
10	BEINEIX	JEAN-JACQUES	1999-09-02	null	1	null	2020-01-08
11	GERONIMI	C.	1993-01-01	null	2	null	2018-01-19
12	LYNE	ADRIAN	1996-02-02	null	2	null	2017-01-22
13	TRUFFAUT	FRANCOIS	2000-12-06	null	2	null	2015-01-21
14	COCTEAU	JEAN	2001-01-02	null	2	null	2020-01-30
15	BOURVIL	BOURVIL	2001-01-01	null	2	null	1995-01-30
16	MALAVOY	CHRISTOPHE	2001-01-01	null	2	null	2020-01-30
17	ROBERT	YVES	2001-01-01	null	2	null	2020-01-30
18	MANESSE	GASPARD	2001-01-01	null	2	null	2020-01-30
19	BELLI	AGOSTINA	2001-01-01	null	2	null	2020-01-30
20	BRASSEUR	CLAUDE	2001-01-01	null	3	null	2020-01-30
21	MARLAUD	PHILIPPE	2001-01-01	null	3	null	2020-01-30
22	BELMONDO	JEAN-PAUL	2001-01-01	null	3	null	2020-01-30
23	ROURKE	MICKEY	2001-01-01	null	3	null	2020-01-30

FIGURE 5 – Exemple du tableau généré par la consultation de tous les personne dans la base de données

Conclusion

Retour sur expérience

En conclusion, ce premier projet nous a permis d'approfondir nos connaissances, que ce soit notre compréhension des systèmes de gestion de bases de données et en programmation en SQL, mais aussi en création des interfaces graphiques pouvant simplifier l'utilisation de notre base par les clients. En effet, c'est sur ce coté que nous avions eu le plus de problèmes, puisqu'aucun membre du groupe n'avait de connaissances extérieures des langages `php, html, ...` C'est pourquoi d'ailleurs après de longues réflexions, nous avons créé une interface en mode texte en JAVA, générant des fichiers `html`. Ce choix nous a donné la flexibilité de bien mettre en évidence nos compétences en ce qui est plus intéressant : la création d'une base de données robuste, propre, sans redondance et sans perte ni d'information ni de mémoire.

Nous avons également développé notre esprit d'équipe et de collaboration, ce qui est pour nous une préparation essentielle à ce qui nous attend dans les années à venir.

Remerciements

Nous remercions M. Sylvain LOMBARDY et M. Mohamed MOSBAH, qui nous ont accompagnés tout au long du projet en nous encadrant et en répondant à toutes nos questions. Leurs réflexions sur l'importance de la rigueur dans la création d'une base de données nous ont été utile pour ce projet et nous seront probablement utiles pour le reste de notre vie de programmeur.