IT203
ENSEIRB-MATMECA
Informatique Deuxième Année

Fédération sportive de HandBall - SGBD

Élèves :
Nathan Decou
Sébastien Delpeuch
Antoine Pringalle
Killian Trolès

Enseignant :
Sylvain Lombardy

08 / 12 / 2020

Introduction

Dans le cadre du cours de Système de Gestion de bases de données, il nous a été demandé d'implémenter et de réfléchir à la mise en place d'une base de données et d'un outil qui nous permettra d'interagir avec cette dernière.

Deux sujets nous sont proposés. Un premier sujet consiste à réaliser une base de données pour gérer des trajets entre étudiants qui désirent rentrer en groupe de soirée. Le second consiste à réaliser une base de données gérant les rencontres de handball entre les clubs d'une fédération. Nous avons choisi de traiter le second sujet.

La structure du rapport suit le cheminement de réflexion pour la construction d'une SGBD. Tout d'abord nous présenterons la modélisation des données et présenterons le contexte du sujet, ensuite nous transformerons le modèle associatif en un modèle relationnel en s'attardant sur la normalisation des modèles. Une fois les étapes théoriques terminées nous présenterons nos choix d'implémentation de la base de données et finalement nous détaillerons l'utilisation de cette dernière.

1 Modélisation des données

Dans la logique de l'élaboration d'un système de gestion de base de données, nous devons dans un premier temps appréhender le besoin du client puis modéliser la base que nous voulons créer via un modèle entité-association.

1.1 Description du contexte

Les clubs sportifs existent dans toutes les métropoles, villes et villages français. La pratique du sport en loisir passe, quelque soit l'âge, par un club sportif. Cette multitude de club sportif invite donc à la création de fédérations sportives et donc au déroulement de match entre les différents clubs.

L'objectif ici est de créer une base de données permettant de gérer les rencontres entre les différents clubs, plus particulièrement dans le handball. Nous devons être capables d'y enregistrer plusieurs clubs, ces derniers possédant des équipes composées de joueurs qui appartiennent à des catégories.

De plus nous devons être capables de modéliser une rencontre et de répertorier l'historique des rencontres. Chaque match oppose deux équipes d'au moins 7 joueurs et d'au maximum 15 joueurs, à une date avec des scores. En somme nous voulons pouvoir déterminer à une date quelle équipe est en tête du classement, quel joueur a le meilleur score etc.

Finalement, dans chaque partie qu'une équipe dispute il y a des joueurs. Les joueurs sont des membres des clubs et cherchent à remporter les matchs avec leurs équipes. De plus nous enregistrons, pour chaque match, le nombre de fautes et le score qu'a effectué le joueur. D'autres informations seront aussi prises en compte comme la direction du club mais nous reviendrons en détail sur les notions supplémentaires lors de l'établissement du modèle entité-association.

1.2 Opérations prévues

Une fois que nous avons précisé le contexte du sujet nous allons étudier les différentes opérations que nous allons avoir besoin de réaliser. Cette étude est importante puisqu'elle va nous

orienter dans nos choix de modélisation. Les requêtes vont se répartir en deux types, les requêtes de consultations et les requêtes de mise à jour. L'objectif ici n'est pas de faire un catalogue précis de toutes les requêtes que nous avons implémentées mais de préciser les types de requêtes que nous avons implémentées. Le meilleur moyen pour obtenir ce catalogue précis est de se rendre en **??** et de suivre les explications pour lancer le site web.

Concernant les requêtes de consultation nous devons tout d'abord assurer des requêtes basiques. C'est à dire pouvoir fournir des informations sur les clubs, les équipes, les joueurs. De plus nous devons pouvoir sortir le classement des joueurs pour une catégorie pour une saison, le classement des équipes, la moyenne des scores sur une saison, la moyenne des scores sur une date, le classement des joueurs sur une saison, le classement des joueurs sur une date, le score des matchs à une date. En plus de celà nous rajoutons la possiblité d'effectuer des recherches dans la base par mot clé.

Attardons nous sur les requêtes de consultation, hormis l'obtention d'informations générales nous pouvons voir que la majeure partie des requêtes est concentrée sur les résultats des matchs et sur les résultats des équipes. Nous pouvons d'ores et déjà prévoir dans notre modélisation un moyen simple de relier une équipe et les matchs qu'elle a effectués pour limiter la complexité temporelle. En revanche nous pouvons aussi remarquer que le lien entre un joueur et un club n'est pas exploité pour les requêtes, nous pourrons en tenir compte dans le modèle pour limiter la complexité spatiale.

Si nous regardons les requêtes de mise à jour, en plus des requêtes classiques d'ajout, suppression, modification d'un joueur, club, match nous détaillerons ultérieurement des procédures permettant d'appeler plusieurs requêtes de mises à jour pour, par exemple, pour remplir une feuille de match.

Suite à la précision des différentes requêtes que nous prévoyons nous pouvons mettre en forme notre modèle entité-association.

1.3 Modèle entité-association

Avant de commencer l'implémentation de la base de données, il est important de dresser un modèle entité-association. Pour ce faire nous devons dans un premier temps lister les différentes entités en étudiant le sujet fourni. Chaque entité sera agrémentée d'attributs permettant de stocker les informations nécessaires à la base de données. De plus nous allons détailler les différentes associations entre nos entités.

L'une des entités les plus importantes est l'entité Personne qui modélise les humains participants aux matchs de handball, chaque personne a un nom, prénom, une date de naissance, une adresse, un numéro de téléphone et une date d'adhésion. La question d'ajouter la date d'adhésion comme attribut d'une personne est intéressante, nous verrons en construisant la totalité du modèle association que nos choix conduisent à ce que toutes les personnes concernées par les matchs soient des membres du club. La date d'adhésion a donc toute sa place en tant qu'attribut de Personne. Dans une optique d'extension de la base de donnée, cet attribut pourrait être mis à nul pour signifier que la personne n'est pas membre du club.

L'entité personne donne naissance à une autre entité, l'entité Joueur. Cette dernière aura, par héritage, les attributs de Personne mais aussi des attributs supplémentaires comme son numéro de licence.

Comme nous l'avons dit plus tôt, en France, les regroupements sportifs sont organisés autour de clubs. Nous avons donc créé une entité Club. Cette entité est reliée à l'entité Personne via l'association Administrer qui permet de traduire qu'une personne possède une fonction (Président, Trésorier ...) dans un club.

Étant donné que nous organisons des matchs entre différentes équipes nous avons réalisé une entité Équipe, cette entité possède des associations avec plusieurs entités

- L'association Être dans avec l'entité Club qui permet de traduire qu'une équipe est rattachée à un club
- L'association Entraîner avec l'entité Personne qui permet d'affecter une personne en tant qu'entraîneur d'une équipe. L'entraîneur peut entraîner plusieurs équipes et toutes les équipes ont exactement un entraîneur.
- L'association Être membre avec l'entité Joueur qui signifie simplement qu'un joueur est membre d'une équipe. Un joueur est forcément membre d'une équipe et une équipe possède entre 7 et 15 joueurs. C'est via cette association que nous établissons un lien entre Joueur et Club. Cette manière de procéder est discutable. En effet elle nous oblige pour savoir si un Joueur fait partie d'un club à passer par l'entité Équipe pour le décider. L'autre solution aurait été d'ajouter une association entre Joueur et Club pour signifier le lien entre les deux. Cette solution aurait diminué la complexité en temps pour savoir si un Joueur appartient à un Club mais aurait augmenté la complexité spatiale. Nous avons vu en **??** que nous prévoyons peu d'opérations nécessitant de savoir si un Joueur appartient à un club. Il nous parait donc plus logique de privilégier cette solution.

Comme précisé dans le sujet chaque équipe est placée dans une catégorie, nous avons donc une entité Catégorie qui possède comme attribut le Nom de la catégorie (cadet, junior, senior ...). Cette entité possède une association Appartenir avec l'entité Équipe, l'association a pour attribut "Niveau" qui correspond au numéro de la catégorie. Nous pouvons donc grâce à ce système attribuer à une équipe l'étiquette "Cadet I" par exemple.

Pour terminer, nous introduisons une entité centrale de notre base de donnée puisqu'elle nous permettra de stocker l'historique des matchs. Cette entité est l'entité Rencontre qui possède l'attribut Date. La relation avec les équipes est modélisée via l'association Participer qui relie rencontre et équipe. Une équipe peut participer à autant de rencontres qu'elle veut mais une rencontre concerne exactement deux équipes.

De plus nous avons ajouté une association Action reliant un joueur et une rencontre. Cette association comporte 3 attributs : score qui correspond à la somme des buts qu'un joueur à marqué pendant une rencontre. De manière analogue nous avons un attribut fautes représentant le nombre total de fautes d'un joueur pendant une rencontre. Finalement nous avons un booléen titulaire permettant de savoir si le joueur était titulaire ou remplaçant lors du match. L'association action nous permet donc d'obtenir des informations plus précises sur le déroulé du match, nous pouvons savoir pour n'importe quelle rencontre combien de points a marqué tel joueur etc. Cette association nous permettra aussi de calculer les scores finaux des équipes lors des rencontres.

Finalement la **??** résume le modèle entité-association du problème et précise les cardinalités des différentes associations.

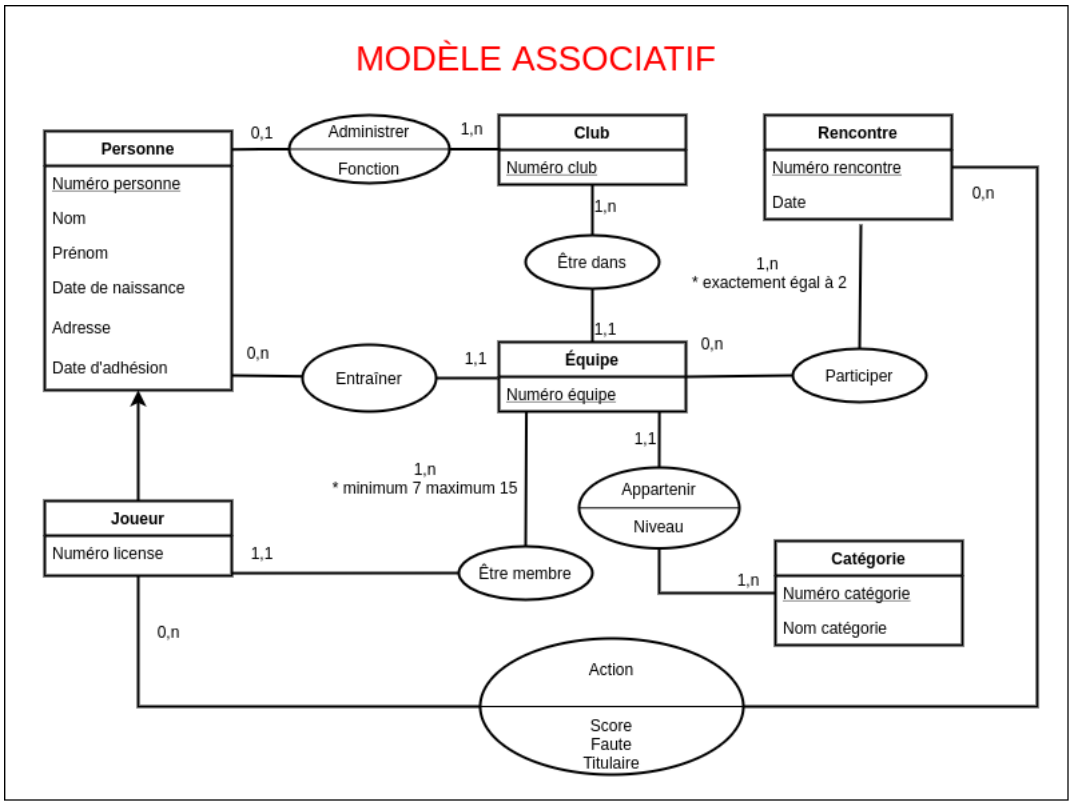


FIGURE 1 – Modèle entité-association de la base de donnée

2 Schéma relationnel

Une fois le modèle conceptuel effectué avec les différentes entités, leurs attributs associés et les cardinalités des associations, nous pouvons dresser le schéma relationnel associé.

Pendant la construction de ce modèle relationnel, nous reviendrons sur certains choix effectués lors de l'établissement du modèle entité-association notamment sur la relation entre équipe et rencontre. Nous étudierons la normalisation de notre modèle.

2.1 Transformation du modèle associatif en modèle relationnel

Pour passer du modèle conceptuel au modèle relationnel, nous avons appliqué les règles vues en cours. Toutes les entités ont été traduites en relations ce qui donne dans un premier temps le modèle relationnel suivant

- Personne(Numéro Personne, Nom, Prénom, Date de naissance, Adresse)
- Club(Numéro Club)
- Catégories(Numéro Catégorie, Nom Catégorie)
- Rencontre(Numéro Rencontre, Date)
- Équipe(Numéro Équipe)
- Joueur(#Numéro Personne, Numéro license)

En observant les cardinalités des associations, nous pouvons nous rendre compte que certaines vont devenir des relations et d'autres vont ajouter des clés étrangères dans les relations. Suite à la traduction des associations nous obtenons le modèle relationnel suivant

- Personne(Numéro Personne, Nom, Prénom, Date de naissance, Adresse, #Numéro Club, Fonction)

— Club(Numéro Club)
— Catégories(Numéro Catégorie, Nom Catégorie)
— Rencontre(Numéro Rencontre, Date, #Numéro Équipe 1, #Numéro Équipe 2)
— Équipe(Numéro équipe,#Numéro Personne, #Numéro Catégorie, Niveau, #Numéro Club)
— Joueur(#Numéro Personne, Numéro license, #Numéro équipe)
— Action(#Numéro Personne, #Numéro Rencontre, score, faute, titulaire)

Nous pouvons justifier quelques choix de transformation sur ce modèle relationnel. Tout d'abord attardons nous sur la traduction de l'entité Rencontre, une traduction pure de l'association Participer nous aurait conduit à la création d'une nouvelle table Participer. Cependant nous pouvons remarquer que l'association possède une cardinalité avec Rencontre exactement égale à 2. Cela nous permet de considérer l'association Participer comme 2 associations $1 : n$, cette considération induit la disparition de la table Participer et l'apparition dans Rencontre de deux clés étrangères correspondant aux numéros des équipes participant à la rencontre. Cette transformation nous permet d'avoir une simplification des requêtes lors de l'implémentation.

2.2 Contraintes d'intégrités

Notre projet possède certaines contraintes d'intégrité qu'il est nécessaire de vérifier pour maintenir le fonctionnement de la base de données, nous allons dans cette partie succintement décrire les différentes contraintes d'intégrités à vérifier.

Dans un premier temps, il est nécessaire de vérifier toutes les contraintes d'intégrités créées par les clés étrangères. Cette étape est relativement simple. Suite à cela nous pouvons remarquer des contraintes d'intégrité moins évidentes.

— Vérifier que le joueur qui réalise une action dans une rencontre appartient bien à l'une des deux équipes de la rencontre
— Vérifier qu'une rencontre est effectuée uniquement contre des équipes de même catégorie
— Vérifier que chaque équipe à entre 7 et 15 joueurs
— Vérifier que la date d'une rencontre n'est pas dans le futur
— Vérifier qu'une rencontre a exactement 14 (7 par équipes) actions comportant le champ « titulaire »à 1 après la date de la rencontre
— Vérifier que lors d'une rencontre, une équipe ne joue pas contre elle-même

La vérification des contraintes d'intégrité sera explicitée en **??**.

2.3 Normalisation

Une fois le schéma relationnel établi et les cardinalités correctement définies nous pouvons nous attarder sur la normalisation de notre base de données. Il est aisé de voir que notre modèle respecte la première forme normale car tous les attributs sont élémentaires, c'est à dire non décomposables. Le modèle respecte aussi la deuxième forme normale, en effet toutes les clé primaires ne sont formées que d'un attribut, sauf celle de l'entité Action, cependant les deux clés sont des clés étrangères issues de la traduction elle respecte donc forcément la deuxième forme normale. In fine nous pouvons voir que le modèle respecte aussi la troisième forme normale puisque dans toutes les relations, tous les attributs dépendent directement de sa clé et non d'autres attributs. La troisième forme normale est appuyée par le graphe de dépendance fonctionnelles disponible en **??**.

Pour déterminer si notre modèle respecte la forme normale de Boyce-Codd il faut l'étudier plus en détail. Tout d'abord rappelons la troisième forme normale de Boyce-Codd.

Un type entité ou un type association est en forme normale de Boyce-Codd si, et seulement si, il est en troisième forme normale et si aucun attribut faisant partie de la clé dépend d'un attribut ne faisant pas partie de la clé. Nous pouvons confirmer cela grâce au graphe de dépendance fonctionnelles disponible en **??**. Cela nous permet donc de confirmer que notre modèle est en forme normale de Boyce-Codd.

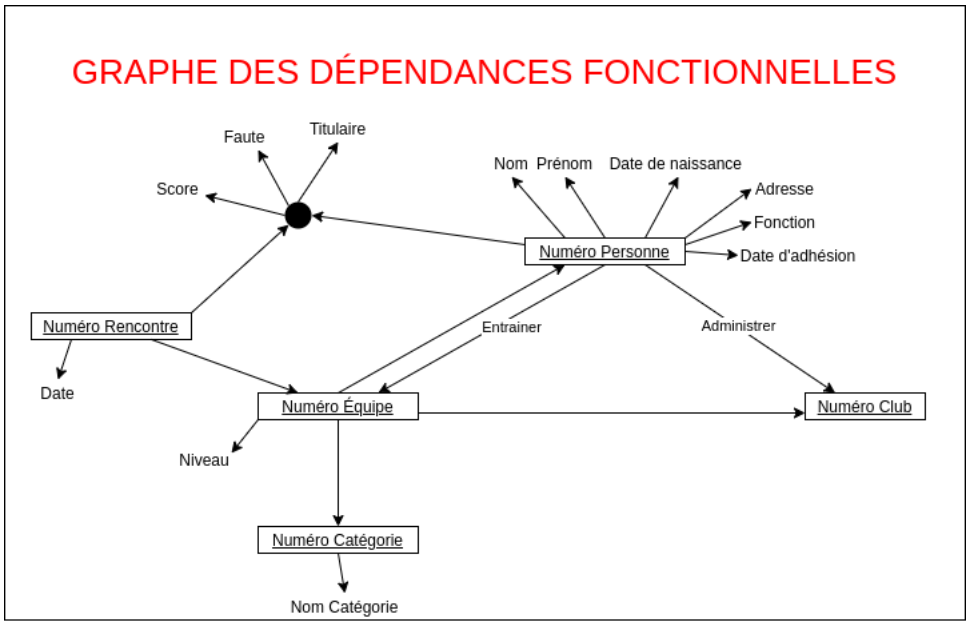


FIGURE 2 – Graphe de dépendance fonctionnelles

Cela nous permet d'avancer que notre modèle possède une certaine qualité. En effet les formes normales sont différents stades de qualité qui permettent d'éviter la redondance, source d'anomalies.

À ce stade, nous avons établi le modèle entité-association après avoir contextualisé le projet. Suite à cela nous avons transformé le modèle entités-associations en modèle relationnel tout en justifiant le choix que nous avons effectué pour les scores dans les rencontres. Finalement nous avons étudié la normalisation de notre modèle, nous pouvons donc sereinement implémenter la base de données.

3 Implémentation de la base de données

Nous pouvons maintenant passer à l'implémentation. Nous avons donc dans un premier temps écrit les fonctions permettant de créer la base de données et d'ajouter ou supprimer des éléments dans les tables. Nous avons ensuite écrit les requêtes de consultation et de statistiques, puis nous avons mis en place un générateur de données, pour finir par implémenter une interface php pour manipuler nos bases.

La création de la base de donnée est une étape relativement basique, nous avons écrit différentes requêtes SQL pour créer chacune des tables tout en respectant le modèle relationnel détaillé en **??**. Les différentes clés primaires sont sous la forme d'un entier et marquées de AUTO_INCREMENT pour simplier l'ajout d'un élément dans la base de données.

L'objectif de cette section n'est pas de détailler précisément l'implémentation de toutes les requêtes, fonctions et procédures que nous avons réalisées mais d'analyser les implémentations les plus intéressantes. Nous allons tout d'abord expliciter les différentes commandes SQL ainsi que le générateur de données que nous avons fait en python enfin nous reviendrons rapidement sur l'implémentation php.

3.1 Implémentation des commandes SQL

Dans cette partie nous allons revenir sur les différentes procédures sql permettant de manipuler la base. L'objectif ici n'est pas de faire la liste exhaustive des procédures mais de nous attarder sur les plus importantes. Nous allons donc détailler la requête de statistique permettant de réaliser le classement des équipes et celle permettant d'obtenir la moyenne des points marqués depuis le début de la saison. Consernant le SGBD nous avons fait le choix d'utiliser mysql pour sa rapidité, sa simplicité et la facilité d'insertion d'API diverses (notamment pour l'interface php). Mysql a aussi l'avantage d'être portable et donc de permettre à notre projet de fonctionner sur presque n'importe quelle distribution. Finalement mysql est open-source ce qui tient à coeur aux membres de l'équipe.

Dans un premier temps nous allons étudier une requête relativement simple, celle consistant à calculer la moyenne des points marqués lors de la saison. Cette requête se décompose en 3 phases, tout d'abord nous prenons la table comportant toutes les rencontres et nous réalisons successivement deux jointures internes avec la table des actions puis avec la table des joueurs. Cela nous permet donc d'avoir une table avec les numéros de rencontre, les dates des rencontre, les numéros des équipes et les scores des équipes. Nous réalisons ensuite une selection sur les dates pour ne sélectionner que les matchs qui sont dans la saison demandée. Pour finir nous utilisons la fonction de moyenne sur les scores et nous renvoyons son résultat. La complexité de cette requête prend sa source dans la première étape lors des jointures internes, nous pouvons la caractériser par $\mathcal{O}(\max(nbr_{\text{rencontres}} \times nbr_{\text{actions}}, nbr_{\text{rencontres}} \times nbr_{\text{personnes}}))$.

Nous allons maintenant étudier une requête plus complexe, celle du classement des équipes. Cette requête se décompose en 4 phases.

Dans un premier temps nous créons une vue qui répertorie pour toutes les rencontres le score de chaque équipe. Cette table est obtenue en parcourant les différentes actions des différentes rencontres et en sommant les bons scores. Nous appellons la table associée à cette vue V_1.

Une fois que nous avons cette table nous allons réaliser une auto jointure sur cette table.

Ainsi pour chaque rencontre nous allons obtenir deux lignes, une ligne qui contient le numéro de la recontre, le nom de l'équipe A, le score de l'équipe A et le score de l'équipe B et une autre ligne qui contient le numéro de la rencontre, le nom de l'équipe B, le score de l'équipe B et le score de l'équipe A respectivement dans ces ordres la table associé à cette vue est appellée V_2.

À ce stade, nous cherchons à savoir, pour chaque équipe, les rencontres qu'elle a gagnées, perdues ou lorsqu'il y a eu un match nul. Pour ce faire nous utilisons des sélections et des sommes, *in fine* nous obtenons trois tables, une avec les numéros des équipes et leur nombre de matchs gagnés, une avec les numéros des équipes et leur nombre de matchs perdus et une avec les numéros des équipes et leur nombre de matchs nuls. Nous notons les trois tables G, P et N.

Il ne reste plus qu'à ordonner le classement, nous récupérons la table brute contenant toutes les équipes et nous réalisons des jointures gauches avec les trois tables trouvées précédemment, nous ordonnons la table finalement trouvé par le nombre de victoire de manière décroissante (puis le nombre de matchs nuls de manière décroissante puis le nom de défaite de manière croissante).

Nous avons donc grâce à cette méthode le classement de toutes les équipes. Nous pouvons étudier la complexité de cette requête, dans un premier temps il faut connaître tous les scores des équipes, pour ce faire nous sommes obligés de parcourir toutes les rencontres, pour chaque rencontre nous devons parcourir la totalité des actions. Lorsque la clé étrangère #Num_Rencontre de l'action correspond au numéro de la rencontre que nous visitons, nous devons vérifier à quel joueur appartient l'équipe et ajouter le score de l'action au score total de l'équipe. Nous pouvons souligner que vérifier si un joueur appartient à l'équipe 1 ou l'équipe 2 est relativement aisé puisqu'un joueur possède une clé étrangère #Num_Équipe et qu'une rencontre possède deux clés étrangères correspondant aux deux équipes qui s'affrontent. Cela se fait en temps constant, cette opération est possible grâce à la modification du modèle relationnel explicité en **??**. Si nous avions fait la traduction fidèle nous aurions dû parcourir la table participer pour retrouver quelles étaient les équipes en compétition et l'opération aurait été linéaire en fonction du nombre de participation. En somme obtenir V_1 se fait en temps quadratique en fonction du nombre d'action et du nombre de rencontres.

Une fois que nous avons la table V_1 nous effectuons une auto jointure et nous admettons qu'une autojointure sur un identifiant est linéaire, la création de V_2 se fait donc en temps linéraire en fonction de la taille de la table c'est à dire en fonction du nombre d'équipe.

Nous voulons maintenant obtenir les tables G, P et N, l'opération est relativement simple il suffit de parcourir la table pour la sélection et de la parcourir pour la projection. Nous avons donc pour chaque table une complexité temporelle quadratique en fonction de la taille de la table V_2.

Il ne reste plus qu'à réaliser les jointures gauches, c'est dernières sont faites les unes à la suite des autres et non de manière imbriquées comme le laisse penser l'algèbre relationnelle. Ainsi chaque jointure est effectué en temps quadratique en fonction du nombre d'équipe. Finalement nous admettons que le tri implémenté par ORDER_BY possède une complexité au pire quadratique en fonction du nombre d'équipe.

En somme la complexité de cette requête trouve sa source à deux endroits, soit à la création de V_1 soit lors de l'étape finale. La complexité du classement est donc

$$\mathcal{O}(\max(nbr_{\text{rencontres}} \times nbr_{\text{actions}}, nbr_{\text{joueurs}} \times nbr_{\text{joueurs}}))$$

Ces implémentations nous permettent de manipuler la base pour avoir les fonctionnalités majeures de l'application. Bien entendu les requêtes basiques (créations de clubs, de joueur, mise à jour d'un club ...) sont aussi implémentées pour permettre une manipulation correcte de la base de données. Pour réaliser les requêtes nous avons du utiliser des procédures et des vues disponibles dans mysql, nous verrons dans la **??** que nous avons aussi utilisé les triggers pour réaliser un contrôle sur la base de données.

Nous avons de plus rajouté une fonctionnalité permettant d'effectuer des recherches dans la base de données, en sql cela se traduit par des requêtes de recherche pour chaque table. Les requêtes sont relativement simples elles prennent en paramètre les informations que l'on connait sur la table (par exemple pour la recherche de joueur on peut donner son nom) puis via une successions de sélections la requête crée la table contenant tous les éléments qui contiennent les paramètres donnés en entrée (par exemple pour la recherche d'un joueur avec son nom, la requête reverra tous les joueurs possédant ce nom). Les requêtes de recherche possèdent une complexité temporelle au pire quadratique en fonction de la taille de la table.

À ce stade tous les outils nécessaires à la base de données sont implémentés, nous allons maintenant brievement étudier le générateur de données ainsi que l'interface que nous avons mis en place pour permettre à n'importe qui d'utiliser la base de données et du générateur de données utilisé.

3.2 Création d'un générateur de données

Étant donné que nous ne disposons pas de données réelles issues de la fédération française de handball nous avons fait le choix de créer un générateur de données aléatoires.

Ce choix est motivé par le fait que notre client peut être amené à utiliser notre livrable sur de nombreuses données et non sur un échantillon que nous aurions pu créer à la main. Le générateur offre la possibilité de créer rapidement plusieurs milliers de données, de tester nos différentes requêtes dessus et donc de mettre en évidence les potentiels ralentissements.

La générateur prend deux paramètres, le nombre de clubs m et le nombre de match par catégorie n. Il crée ensuite entre 1 et 5 équipes par club et les répartit dans des catégories et des niveaux. Une fois qu'il a le nombre d'équipes il peut déduire le nombre de personnes nécessaires pour chaque club, in fine chaque équipe possèdera entre 7 et 15 joueurs et un coach (qui dans la plupart des cas est un membre de l'équipe) et chaque club possèdera 3 personnes ayant des fonctions dans le club. Les personnes sont des combinaisons aléatoires de noms et prénoms prédéfinis, d'adresse aléatoires, de dates de naissance aléatoires (entre 1980 et 2010) et de dates d'adhésion aléatoires.

Pour réaliser les rencontres, le générateur va étudier la liste des équipes et la liste des catéroies, il va générer alors n matchs entre les équipes d'une même catégorie aléatoirement, la date de la rencontre est aussi aléatoire entre 2017 et 2020.

Comme nous l'avons décrit en **??** chaque joueur réalise une action pendant un match, cette action comporte les informations sur son score, son nombre de fautes et s'il était titulaire ou pas. Pour remplir cette table le générateur va, pour chaque équipe de chaque rencontre, choisir aléatoirement 7 joueurs qui seront titulaires qui auront un score entre 0 et 5 et un nombre de fautes entre 0 et 3. Si un joueur n'est pas titulaire tous les champs sont à 0.

Ce générateur en trois temps nous permet d'obtenir un grand nombre de données permet-

tant de tester notre base sur des modèles plus proches de la réalité. Il est à noter que nous sommes conscients que le générateur peut encore être amélioré pour ajouter de la cohérence et se rapprocher de la réalité mais ce n'est pas l'objectif recherché, pour l'instant le générateur nous permet d'obtenir des données respectant les contraintes que le modèle impose. Les tests de ces contraintes ainsi que les améliorations de ces contraintes seront détaillées en **??**.

En conclusion nous avons étudié la complexité de notre générateur, cette dernière est indicative puisque le générateur peut ne pas être utilisé du tout ou être utilisé une fois à la création, cela permet de donner un ordre d'idée du temps que le générateur va mettre à créer un très large nombre de données. Cette complexité est de

$$\begin{aligned}\mathcal{O}\max((&nbr_{\text{club}} \times nbr_{\text{équipe}} \times nbr_{\text{joueur par équipe}}),\\ &(nbr_{\text{rencontre}} \times nbr_{\text{joueur total}} \times nbr_{\text{joueur par équipe}}),\\ &(nbr_{\text{catégories}} \times nbr_{\text{rencontre par catégories}} \times nbr_{\text{club}}))\end{aligned}$$

3.3 Implémentation de l'interface php

Nous allons maintenant décrire l'interface permettant d'intérargir avec la base de données que nous avons implémentée.

Nous avons choisi d'utiliser une interface php pour permettre à n'importe qui d'intérargir avec notre base. L'interface php communique avec la base mysql via le paquet phpmysql-d. Notre interface possède deux modes de fonctionnement, soit un mode de consultation, soit un mode d'insertion.

L'idée de la première partie est de pouvoir rapidement voir les différentes tables (joueurs, rencontres etc) mais aussi le résultat des requêtes de statistiques (classement des équipes, moyennes des points etc). L'implémentation php appelle donc la requête définit en mysql et stocke le résultat. Ce résultat est ensuite parcouru et affiché à l'écran. Nous pouvons remarquer que cela introduit une complexité supplémentaire puisque il faut rajouter celle de parcours de la table résultante de la requête SQL à celle de la requête en elle même.

Table brute de Personne

numero personne	nom	prenom	date de naissance	adresse	date adhesion	fonction	numero club
1	maël	candice	1980-02-27	146 rue maël	1985-02-27		1
2	hugo	ethan	1983-11-13	147 rue hugo	1988-11-13		1
3	maël	emy	1986-05-18	64 rue maël	1991-05-18		1
4	arthur	raphaël	2005-03-15	23 rue arthur	2010-03-15		1
5	paul	hugo	2005-01-23	85 rue paul	2010-01-23		1
6	timéo	arthur	1986-07-09	46 rue timéo	1991-07-09		1
7	eden	mathéo	2004-10-23	118 rue eden	2009-10-23		1
8	zoé	baptiste	2007-02-27	67 rue zoé	2012-02-27		1
9	léa	raphaël	1998-07-25	41 rue léa	2003-07-25		1
10	léonie	mia	1987-02-14	11 rue léonie	1992-02-14		1

FIGURE 3 – Exemple d'une consultation des bases

Le deuxième mode permet d'insérer et de supprimer des éléments dans la base de données. Ce système repose sur le principe de formulaire, lorsque le client souhaite ajouter des données il peut juste remplir le formulaire correspondant. L'interface php appellera alors une requete SQL avec les données issues du formulaire pour ajouter ou supprimer une donnée, si la requête échoue

le client en sera informé et l'erreur provenant de mysql sera affichée.

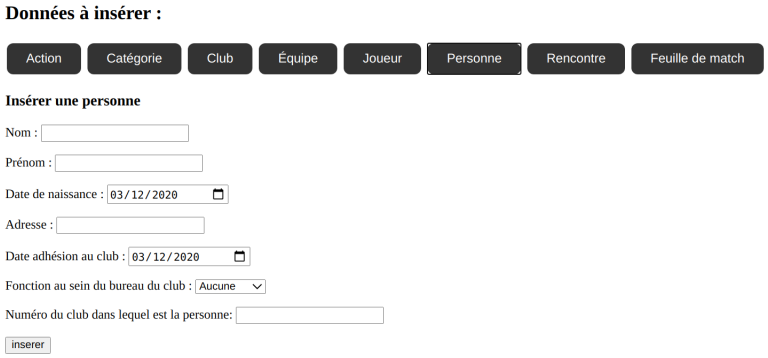


FIGURE 4 – Exemple de l'ajout d'un personne

Dans ce deuxième mode nous avons ajouté un formulaire offrant la possibilité à l'utilisateur de remplir directement une feuille de match durant une rencontre, cette dernière contient toutes les informations d'une rencontre avec les noms des équipes, la date, les différentes actions etc. Une fois remplie nous utilisons le langage php pour insérer correctement les différentes informations dans les différentes bases.

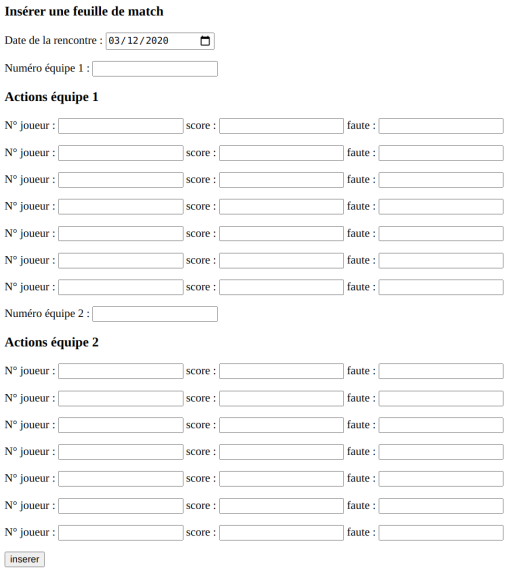


FIGURE 5 – Exemple de l'ajout d'une feuille de match

Nous pouvons aussi noter l'ajout du moteur de recherche via des formulaires, cela permet de rechercher simplement et efficacement n'importe quelle donnée, chaque recherche appelle une requête sql que nous avons présentée plus tôt.

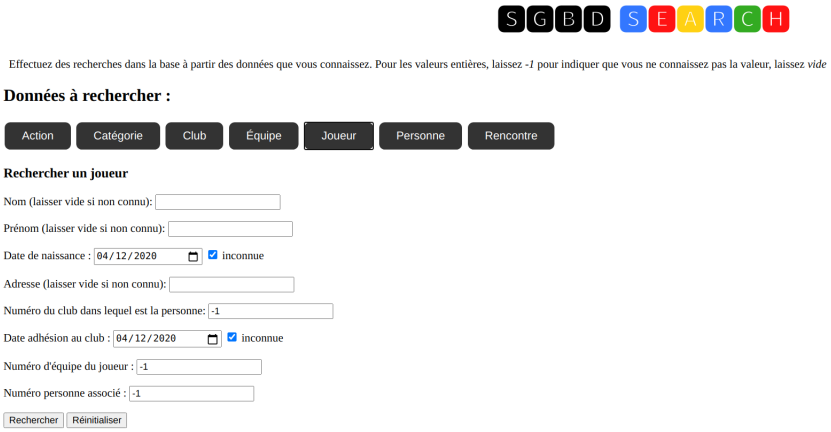


FIGURE 6 – Exemple de la recherche d'un joueur

Les deux modes d'utilisation de l'interface php permettent d'obtenir et de manipuler toutes les informations nécessaires pour le client, des perspectives d'améliorations seront abordées en **??**.

En conclusion nous avons présenté les différentes implémentations que nous avons utilisés, elles sont découpées en trois domaines, l'implémentation SQL permettant de manipuler les données, l'implémentation python permet quant à elle de créer des données pour la base et finalement l'ajout du php nous permet d'obtenir une interface pour que l'utilisateur puisse consulter et modifier simplement la base. À ce stade nos différentes implémentations sont fonctionnelles et la base est prête à être utilisée, nous allons détailler dans la prochaine partie comment l'utiliser et les perspectives d'améliorations.

4 Utilisation de la base de donnée

Maintenant que nous avons décrit la modélisation et l'implémentation de la base de données nous allons procéder à la description de son utilisation. Pour ce faire nous allons successivement présenter l'environnement nécessaire à l'exécution du projet, puis nous décrirons succintement comment utiliser notre livrable. Enfin il sera présenté les tests qui garentissent le bon fonctionnement de la base de données ainsi que les options d'évolutions de cette dernière.

4.1 Installer et exécuter le projet

Dans un premier temps il faut posséder les paquets suivants mysql, php, python3, php-mysql, apache2 et mysql-server. Sur Ubuntu les paquets sont installables grâce à la commande sudo apt install <nom>.

Une fois les différents paquets installés vous devez récupérer notre livrable. L'archive comprenant le livrable est disponible ici [1].

Si vous n'avez pas de jeu de données il vous faut créer ce dernier. Pour cela déplacez vous à la racine du projet et exécutez la commande python3 generator/main.py. Cela va générer des fichiers .csv dans le dossier generated_csv_file/. Il faut ensuite placer ces fichiers dans le dossier sécurisé de mysql (ou définir le dossier précédent comme sécurisé pour mysql). Sur ubuntu ce dossier se trouve dans /var/lib/mysql-file/. Si vous avez déjà un jeu de données, il suffit de les placer dans le dossier sécurisé.

Il faut ensuite démarrer et se connecter à mysql en lançant simplement la commande mysql. Une fois connecté vous pouvez réaliser la commande source all.sql pour initialiser la base. Pour vérifier que l'opération est un succès vous pouvez faire la commade desc Personne vous obtiendrez normalement la figure suivante

```
mysql> desc Personne;
+----------------+-------------------+------+-----+---------+----------------+
| Field          | Type              | Null | Key | Default | Extra          |
+----------------+-------------------+------+-----+---------+----------------+
| num_personne   | smallint unsigned | NO   | PRI | NULL    | auto_increment |
| nom            | varchar(40)       | NO   |     | NULL    |                |
| prenom         | varchar(40)       | NO   |     | NULL    |                |
| date_naissance | date              | NO   |     | NULL    |                |
| adresse        | varchar(100)      | YES  |     | NULL    |                |
| date_adhesion  | date              | NO   |     | NULL    |                |
| fonction       | varchar(30)       | YES  |     | NULL    |                |
| num_club       | smallint unsigned | NO   | MUL | NULL    |                |
+----------------+-------------------+------+-----+---------+----------------+
8 rows in set (0.00 sec)
```

Figure 7 – Résulat de la commande desc Personne

À ce stade vous pouvez, si vous le souhaitez quitter mysql avec exit. Pour lancer l'interface vous devez vous déplacer dans le dossier site/, dans ce dossier vous devez modifier le fichier connectdb.php, plus particulièrement les champs $username et $password (ce sont les mêmes identifiant que pour se connecter à mysql) [2]. Une fois le fichier enregistré lancer le script site.sh puis allez dans votre navigateur à l'adresse localhost :8000.

Vous pouvez maintenant naviguer dans le site et réaliser les différentes opérations souhaitées, la description de l'utilisation du site n'est pas réalisée ici puisqu'elle est relativement intuitive. Une précision est tout de même apportée, lorsque vous faites des insertions ou des suppressions avec les formulaires aucune vérification n'est réalisée directement sur les formulaires. Si votre saisie est incorecte vous obtiendrez une erreur provenant de mysql.

1. https://sdelpeuch.github.io/projetsgbd.zip
2. Si vous ne possédez pas d'identifiant particulier pour mysql et que vous lancez ce dernier avec sudo mysql il faudra alors mettre le $username à root et le $password vide. Il faudra ensuite lancer la commande sudo php -S localhost :8000

4.2 Tests et Options d'évolutions

Nous allons pour conclure décrire les différents tests et les options d'évolutions du livrable.

Dans un premier temps nous allons expliciter les tests qui s'assurent de l'intégrité de la base, l'intégrité de la base est vérifiée par différents moyens.

Le moyen le plus simple est l'ajout de contrainte sur les clés étrangères qui permet de nous assurer des opérations éléments, par exemple qu'un joueur est toujours rataché à une personne. Ensuite nous avons l'ajout des caractéristiques NOT NULL et UNIQUE sur les attributs des différentes tables cela nous permet par exemple de rendre chaque joueur unique en mettant sa clé UNIQUE dans le cas contraire une personne aurait pu devenir 2 joueurs. L'ajout de NOT NULL permet d'assurer facilement la présence de certains éléments essentiels dans les tables par exemple la clé étrangère #num_personne dans la table équipe (qui représente l'entraineur) est non nulle, forçant chaque équipe à avoir un entraineur. Nous pouvons aussi retrouver cette fonctionnalité pour forcer chaque joueur à apartenir à une équipe.

En plus de ces vérifications basiques, nous avons rajouté des vérifications plus poussées, ces vérifications sont les traductions des contraintes d'intégrités explicitées en **??** ainsi chaque contrainte d'intégrité est traduite en une procédure SQL ces procédures sont appelées via des trigger lorsque des modifications / insertions / suppressions sont effectuées sur les différentes bases. Par exemple si vous voulez insérer un joueur via le formulaire, le trigger va se déclencher et appeler la requête qui vérifie s'il n'y a pas déjà 15 joueurs dans cette équipe. Si non, l'action va continuer en appelant les bonnes requetes d'insertion, si oui l'utilisateur verra une erreur. Dans le pire des cas les triggers peuvent entrainer un ajout d'une complexité temporelle quadratique en fonction de la taille de la table à tester. Ces vérifications nous permettent d'effectuer des tests poussés sur la base à chaque action de l'utilisateur l'empêchant totalement de créer des situations incohérentes.

Nous allons maintenant décrire les possibilités d'amélioration de notre projet. La plus importante que nous n'avons pas réalisée par manque de temps est la vérification des données rentrées par le client côté php et non côté sql, cela permettrait de refuser immédiatement une demande d'insertion / suppresion si le formulaire rempli par le client est erroné et de ne pas appeler les requêtes mysql pour rien. Concernant le modèle, une piste d'amélioration pourrait être d'augmenter la granularité temporelle. En effet pour l'instant nous pouvons être précis jusqu'au nombre de points et au nombre de faute qu'a fait un joueur dans un match. Nous pourrions augmenter cela en changeant l'association « action » pour enregistrer la minute à laquelle le joueur à marqué un but ou fait une faute. Bien que cette amélioration soit relativement simple à effectuer cela entrainerait irrémédiablement un allourdissement du temps de traitement des différentes requêtes. Cette amélioration n'a donc pas été implémentée et dans un cadre réel nous aurions attendu une demande spécifique du client sur ce point pour rajouter cela.

Concernant l'interface php, elle pourrait être améliorée en mettant en place un système d'utilisateur et des droits en fonction des utilisateurs, un simple joueur pourrait voir uniquement les informations le concernant, le président d'un club pourrait voir toutes les informations d'un club et rentrer des feuilles de match et le président de la fédération aurait accès à toutes les informations. Nous avons décidé de ne pas développer cette amélioration car elle ne rentre pas dans le cadre de ce projet.

Finalement concernant le générateur, il est évident qu'il reste beaucoup de travail dessus pour

avoir quelque chose de réaliste et supprimer les aberrations (rien n'empêche quelqu'un né en 1980 d'être dans la catégorie poussin ou qu'une équipe soit entraînée par quelqu'un né en 2010). Ces aberrations ne sont pas gênantes pour l'utilisation de la bdd puisque nous supposons, dans une utilisation réelle, que la concordance des âges etc est à la charge du client.

Conclusion

En somme, nous avons implémenté une interface permettant de gérer une base de données permettant de répertorier des rencontres de handball. Nous avons tout d'abord modélisé les attentes de notre client via un modèle entité-association. Suite à la validation de ce modèle, puis la transformation en modèle relationnel, l'implémentation de notre base via Mysql a été effectuée. Suite à cela nous avons détaillé l'implémentation des requêtes les plus utilisées et complexes, nous avons aussi abordé les fonctions plus complexes permettant de réaliser des séries de requêtes comme la fonction permettant de remplir une feuille de match. Nous expliquons ensuite rapidement la mise en place de l'interface php, permettant à notre client de ne pas avoir besoin de connaissances en informatique pour manipuler la base. Finalement une notice d'utilisation est détaillée dans la dernière partie, cette partie comprend aussi les possibilités d'amélioration.

Concernant le résultat final, toutes les implémentations prévues sont fonctionnelles, bien que nous ayons plusieurs pistes d'amélioration pour améliorer le projet nous pouvons avancer que nous avons répondu aux attentes du client en fournissant une base de données permettant de gérer une fédération de handball ainsi qu'une interface permettant de manipuler cette base de données. Les principales forces de cette base de données étant sa rapidité (complexité quadratique en fonction du nombre de joueur pour la requête la plus imposante), sa portabilité obtenue grâce à l'utilisation de mysql et sa facilité d'utilisation puisque une fois lancée nous pouvons tout gérer depuis une page web relativement intuitive.

Pour conclure, l'équipe a beaucoup appris lors de ce projet. La répartition du travail a été réalisée selon les affinités de chacun avec les différents domaines mais nous avons veillé à ce que chacun puisse connaître la base dans tous les domaines. En somme ce projet à déjà donné des idées à certains membres de l'équipe pour réaliser un SGBD pour des projets personnels.