# Projet Base de Données

Equipe 8

10 avril 2017

# 1 Conception

## 1.1 Analyse du problème

### 1.1.1 Identification des propriétés

On identifie les données, propriétés élémentaires à partir de la description de l'application.

{NomSport, TarifBase, TypeTerrain, IdStage, HeureDébut, HeureFin, Jour, NomTerrain, IdCommune, HeureOuverture, HeureFermeture, Capacité, TypeTerrain, IdPersonne, Nom, Prenom, Email, Telephone, Num, Rue, IdMembre, DateInscription, Prix, DateNaissance, IdMoniteur}

## 1.2 Identification des contraintes

Table 1 – Dépendances fonctionnelles

| **Dépendances fonctionnelles** |
|---|
| NomSport → TarifBase, TypeTerrain |
| IdStage → HeureDébut, HeureFin, Jour, NomTerrain, NomSport |
| NomTerrain, IdCommune → HeureOuverture, HeureFermeture, Capacité, TypeTerrain |
| IdPersonne → Nom, Prenom, Email, Telephone, Num, Rue, IdCommune |
| IdStage, IdMembre → DateInscription, Prix |
| IdMembre → DateNaissance, IdPersonne |
| IdMoniteur → IdPersonne |

Table 2 – Contraintes de valeur

| **Contraintes de valeur** |
|---|
| Nom, Prénom, Rue uniquement composé de lettres |
| Email est composé d'un '@' et le nom de domaine de premier niveau est composé de 2 lettres (ex : '.fr') |
| TarifBase $> 0$ |
| HeureDebut $<$ HeureFin |
| Capacite $> 0$ |
| HeureOuverture $<$ HeureFermeture |
| Prix $> 0$ |
| Ext(IdMembre) $\subseteq$ Ext(IdPersonne) |
| Ext(IdMoniteur) $\subseteq$ Ext(IdPersonne) |
| Ext(IdMoniteur) $\bigcup$ Ext(IdMembre) = Ext(IdPersonne) |

TABLE 3 – Contraintes de Multiplicité

| **Constraintes de Multiplicité** |
|---|
| Un stage ne se déroule que sur un terrain |
| Un stage est encadré par un ou plusieurs moniteurs |
| Un moniteur est habilité à encadrer un ou plusieurs sports |
| Un moniteur ne peut être expert que dans un seul sport |
| Une personne ne peut avoir qu'une adresse |
| Un stage n'a qu'un seul superviseur (au moins un) |
| Un sport ne peut que se pratiquer sur certains types de terrains |
| Un stage ne se déroule que sur un seul terrain |
| Un terrain ne peut être que d'un type |
| Un terrain ne peut se trouver que dans une commune |

TABLE 4 – Autres Contraintes

| **Autres Contraintes** |
|---|
| Il faut un moniteur pour dix stagiaires |
| Pour un stage donné, un moniteur ne peut pas être stagiaire |
| Si possible, le superviseur du stage est un expert dans le sport concerné |
| Les personnes habitant la commune où se déroulera le stage bénéficient d'une réduction de 10% |
| Pour un stage donné, nombre d'inscrits <capacité du terrain où se déroule le stage |
| Pour un stage donné, HeureOuverture du terrain <HeureDébut du stage et HeureFermeture >HeureFin |

## 1.3 La conception Entités/Associations

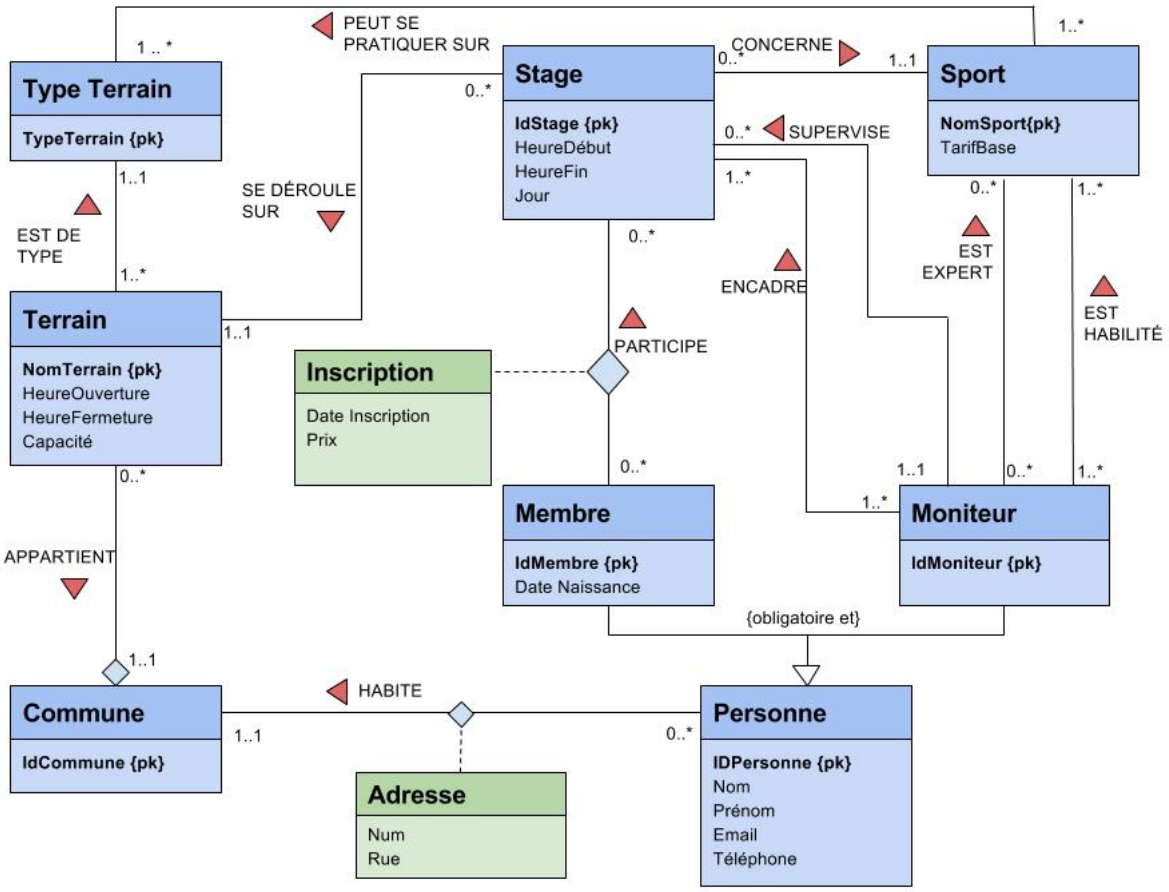


FIGURE 1 – Diagramme UML

## 1.4 Traduction en relationnel

**Légende** : Les attributs soulignés sont les clées.

## 1.5 Entités simples

**Sport(NomSport,TarifBase)**
**TypeTerrain(TypeTerrain)**
**Commune(IdCommune)**
**Terrain(NomTerrain,HeureOuverture,HeureFermeture,Capacité)**
**Stage(IdStage,HeureDébut,HeureFin,Jour)**
**Personne(IdPersonne,Nom,Prénom,Email,Téléphone)**

**Remarque :** nous aurions pu définir une *Commune* sous forme de champs à l'intérieur de l'entité *Terrain* et relier *Commune* à *Terrain* mais nous avons opté pour créer une entité *Commune* car cela permettait de clarifier les dépendances.

## 1.6 Sous-types d'entités

Afin de différencier un 'simple' membre de l'association qui peut participer à des stages d'un moniteur qui en plus encadre des stages nous avons créer deux sous-entités Membre et

Moniteur qui héritent des champs de l'entité Personne.
Les champs *IdMembre* et *IdMoniteur* font ainsi directement référence aux champs *IdPersonne*.

**Membre(IdMembre,DateNaissance)**
IdMembre référence IdPersonne de Personne

**Moniteur(IdMoniteur)**
IdMoniteur référence IdPersonne de Personne

## 1.7 Entité faible

L'entité *Terrain* est faible par rapport à l'entité *Commune* car un terrain à besoin d'une commune en plus de son nom pour pouvoir s'identifier (des terrains de communes différentes pouvant avoir le même nom).
**Terrain(IdCommune,NomTerrain,HeureOuverture,HeureFermeture,Capacité)**

## 1.8 Associations binaires de cardinalité 1..1

**Stage(IdStage,HeureDébut,HeureFin,Jour,NomSport,NomTerrain,IdMoniteur)**
NomSport référence NomSport de Sport
(NomTerrain,IdCommune) référencent (NomTerrain,IdCommune) de Terrain
IdMoniteur référence IdMoniteur de Moniteur

**Terrain(NomTerrain,IdCommune,HeureOuverture,HeureFermeture, Capacité,TypeTerrain)**
TypeTerrain référence TypeTerrain de Type Terrain

**Personne(IdPersonne,IdCommune,Num,Rue,Nom,Prénom,Email,Téléphone)**
IdCommune référence IdMoniteur de Moniteur

## 1.9 Associations binaires de cardinalité x..*

**Participe(IdStage, IdMembre, Prix, DateInscription)**
IdStage référence IdStage de Stage
IdMembre référence IdMembre de Membre

**Encadre(IdMoniteur,IdStage)**
IdMoniteur référence IdMoniteur de Moniteur
IdStage référence IdStage de Stage

**Habilité(IdMoniteur,NomSport)**
IdMoniteur réference IdMoniteur de Moniteur
NomSport référence NomSport de Sport

**Expert(IdMoniteur, NomSport)**
IdMoniteur réference IdMoniteur de Moniteur
NomSport référence NomSport de Sport

**Possibilite_Pratiquer(NomSport, TypeTerrain)**
NomSport référence NomSport de Sport
TypeTerrain référence TypeTerrain de Type Terrain

## 1.10 Formes Normales

Le schéma relationnel est en troisième forme normale BCK. Tous les attributs non clés ne dépendent pas d'un ou plusieurs attributs ne participant pas à la clé.

# 2 Analyse des fonctionnalités et implantation en SQL2

Toutes les requêtes qui vont suivre sont exécutées via une transaction. De plus, les champs saisis par l'utilisateur dans un formulaire sont vérifiés en Java avant d'être convertis en requête SQL. On s'assure ainsi que chaque saisie est valide lors de la validation du formulaire. En faisant cela, on évite d'avoir une base de données incohérente lorsque les contraintes sont difficiles à exprimer en SQL. En outre, cela permet d'avoir un message personnalisé pour chaque type d'erreur.

Notre application permet d'ajouter dans la base de données :

— Un membre
— Un moniteur
— Un stage

Elle permet aussi l'affichage de caractéristiques intéressantes pour l'association.

## 2.1 Création d'une personne dans la base de données

Lorsque l'utilisateur veut ajouter un membre ou un moniteur, il doit remplir un formulaire. Le formulaire est différent selon la nature de l'ajout. Dans tous les cas, les caractéristiques communes (nom, prénom...) sont enregistrés dans la table `Personne` grâce à l'instruction suivante :

```
1 INSERT INTO Personne(IDPERSONNE, NOM, PRENOM, EMAIL, TELEPHONE, NUM, RUE,
      IDCOMMUNE)
2 VALUES(IDPersonne.nextval, 'Nom', 'Prenom', 'adresse@email.com', 'telephone',
      num, 'rue', codePostal);
```

Si la commune n'est pas présente dans la base, on la rajoute.

```
1 SELECT IDCOMMUNE FROM COMMUNE WHERE IDCOMMUNE = commune; -- test de la presence
2 INSERT INTO COMMUNE(IDCOMMUNE) VALUES(commune); -- ajout de la commune
```

### 2.1.1 Ajout d'un membre

Une fois que la personne a été enregistrée, il faut la rajouter dans la table `Membre`.

```
1 INSERT INTO MEMBRE(IDMEMBRE, DATENAISSANCE) VALUES(idMembre, TO_DATE('
      dateNaissance','dd/mm/yyyy'));
```

### 2.1.2 Ajout d'un moniteur

De la même manière, le moniteur est enregistré dans la table `Moniteur`.

```
1 INSERT INTO MONITEUR(IDMONITEUR) VALUES(idMoniteur);
```

Chaque habilitation est enregistré dans la table `Habilite`.

```
INSERT INTO HABILITE VALUES('nomSport', idMoniteur);
```

L'expertise d'un moniteur est enregistré dans la table `Expert`

```
INSERT INTO EXPERT VALUES('nomSport', idMoniteur);
```

## 2.2 Création d'un stage

Un stage est aussi créé par un formulaire.

```
INSERT INTO STAGE VALUES (IDSTAGE.nextval, TO_DATE('jour:heureDebut', 'dd/mm/yyyy:hh24:mi'), TO_DATE('jour:heureFin', 'dd/mm/yyyy:hh24:mi'), 'nomSport', 'nomTerrain', idCommune, idMoniteur); -- idMoniteur de celui qui supervise le stage
```

Le terrain sélectionné doit être compatible avec le créneau du stage.

Ensuite, il faut trouver les stagiaires et les moniteurs disponibles.

Par exemple, tous les stagiaires disponibles entre les dates `debut` et `fin` sont obtenus via la requête :

```
(SELECT DISTINCT p.NOM, p.PRENOM, IDPERSONNE FROM MEMBRE m, PERSONNE p WHERE p.IDPERSONNE = m.IDMEMBRE) -- tous les stagiaires
MINUS -- sauf
(SELECT DISTINCT pers.NOM, pers.PRENOM, IDPERSONNE FROM MEMBRE m, STAGE s, PARTICIPE p, PERSONNE pers
WHERE m.IDMEMBRE = p.IDMEMBRE AND p.IDSTAGE = s.IDSTAGE AND pers.IDPERSONNE = m.IDMEMBRE -- les stagiaires
AND ( (debut < s.HEUREDEBUT AND s.HEUREDEBUT < fin) -- qui ont un stage debutant entre debut et fin
      OR
        (debut < s.HEUREFIN AND s.HEUREFIN < fin ) -- ou qui ont un stage finissant entre debut et fin
      OR
         (s.HEUREDEBUT <= debut AND fin <= s.HEUREFIN) -- ou qui sont totalement occupes entre debut et fin
     )
);
```

Nous avons écrit quelque chose de similaire pour les moniteurs.

Chaque moniteur encadrant la séance est ajouté dans la table `Encadre`

```
INSERT INTO ENCADRE VALUES(idMoniteur, idStage);
```

Pour l'ajout d'un stagiaire, il faut d'abord déterminer s'il habite dans la même commune que celle du lieu du stage, afin d'appliquer la réduction le cas échéant.

```
SELECT IDCOMMUNE FROM PERSONNE, MEMBRE WHERE IDMEMBRE = IDPERSONNE AND IDMEMBRE = idMembre; -- On regarde dans quelle commune le stagiaire habite
INSERT INTO PARTICIPE VALUES(SYSDATE, prix, idMembre, idStage); -- Ajout du stagiaire, SYSDATE correspond a la date d'inscription (par defaut, celle du systeme)
```

## 2.3 Affichage des statistiques

### 2.3.1 Nombre total de stagiaires

Pour trouver le nombre total de stagiaires, on sélectionne tous les id dans la table recensant tous les participants.

```
SELECT COUNT(DISTINCT IDMEMBRE) FROM PARTICIPE;
```

### 2.3.2 Nombre moyen d'inscrits par stage

On énumère tous les stages de la base de donnée :

```
SELECT COUNT(*) FROM STAGE;
```

Puis on énumère toutes les inscriptions (une personne peut être inscrite à plusieurs stages).

```
SELECT COUNT(*) FROM PARTICIPE;
```

Il n'y a plus qu'à calculer le rapport du nombre d'inscriptions sur le nombre de stages.

### 2.3.3 Total des recettes de l'association pour l'année en cours

Pour connaître les recettes de l'association de l'année en cours, on additionne tous les prix payés par les participants. Cette addition ne s'applique que pour les participants s'étant inscrits entre le 1er janvier 2017 et le 31 décembre 2017.

```
SELECT SUM(PRIX) FROM PARTICIPE WHERE DATEINSCRIPTION <= TO_DATE('2017/12/31','
    yyyy/mm/dd') AND DATEINSCRIPTION >= TO_DATE('2017/01/01','yyyy/mm/dd');
```

### 2.3.4 Terrains les plus utilisés

Pour énumérer les terrains les plus utilisés on récupère les couples (terrain, utilisation) depuis la table recensant les stages. On trie ces utilisations dans l'ordre décroissant. Ici, nous avons choisi de ne pas mettre de limite dans la liste, mais nous pourrions choisir d'afficher seulement les 5 terrains les plus utilisés par exemple.

```
SELECT NOMTERRAIN, COUNT(*) AS THECOUNT FROM STAGE GROUP BY NOMTERRAIN ORDER BY
    THECOUNT DESC;
```

### 2.3.5 Ratio supervision/encadrement pour les moniteurs

Pour calculer le ratio du nombre de supervisions par le nombre d'encadrements pour chaque moniteur, nous avons choisi de traiter cela en deux temps. On récupère d'abord la liste des moniteurs qui supervisent au moins un stage, avec leur id et le nombre de stages que chacun supervise. On fait de même pour les moniteurs qui encadrent au moins un stage. On crée un dictionnaire correspondant à chaque requête (associant un id avec un nombre de supervisions/encadrements), puis on parcourt les id d'un de ces dictionnaires en vérifiant si chaque

id est également présent dans l'autre dictionnaire. Si un id est dans les deux dictionnaires, on calcule le ratio en récupérant le nombre de stages supervisés dans le premier dictionnaire, et le nombre de stages encadrés dans le deuxième dictionnaire. Pour l'affichage, on récupère les noms et prénoms associés à l'id du membre sélectionné.

On a accès aux superviseurs dans la table `Stage` :

```
SELECT IDMONITEUR, COUNT(*) FROM STAGE GROUP BY IDMONITEUR;
```

La table `Encadre` nous permet de récupérer les moniteurs qui encadrent au moins un stage :

```
SELECT IDMONITEUR, COUNT(*) FROM ENCADRE GROUP BY IDMONITEUR;
```

Pour récupérer l'identité d'un moniteur à partir d'un id :

```
SELECT NOM, PRENOM from PERSONNE,MONITEUR where IDPERSONNE = IDMONITEUR AND
    idPERSONNE = " + id;
```

# 3 Bilan du projet

## 3.1 Organisation

Les premières séances, nous avons travaillé ensemble sur la conception. Nous avons suivi la démarche proposée en TD. Une fois l'analyse faite, nous avons divisé les tâches en formant 3 groupes de 2 :

- Un premier groupe s'est chargé de construire les tables en SQL
- Le deuxième groupe s'est chargé de comprendre le fonctionnement de JDBC et d'implémenter les premières requêtes avec Java avec les tables existantes
- Le troisième groupe s'est chargé de l'interface graphique

## 3.2 Cohérence de notre démarche

Notre organisation nous a permis d'avancer efficacement de façon parallèlle.
Chaque groupe n'avait quasiment pas besoin du travail des autres.
Nous nous sommes dès le départ mis d'accord sur les noms des tables et des attributs.
Ainsi, le groupe chargé des requètes JDBC pouvait écrire les requètes avant même que les tables soient créées et complétées.
Le groupe chargé de la mise en place de l'interface java pouvait préparer la fenêtre et même prévoir les requètes.
Par exemple, pour l'ajout d'un stage par un utilisateur, le groupe chargé des requètes JDBC donnait simplement le nom de la méthode et les attributs nécessaires pour exécuter la commande.

Nous avons réussi à nous tenir à cette organisation. Les différentes tâches étant assez liées, certains membres se sont penchés sur plusieurs sujets. Nous avons essayé de tous réaliser des tests pour vérifier le bon fonctionnement de notre table et de notre programme.

## 3.3 Justification de nos choix de conception

Un cahier des charges aussi ouvert impliquait beaucoup de libertés quant aux choix de conception.
Nous avons choisi de séparer les Membres et les Moniteurs qui sont tous des Personnes.
L'avantage réside dans les associations d'encadrement, d'expertise, d'habilitation, de supervisement et de participation. Avec une seule table personne, le schéma aurait été plus confus. La date de naissance propre à un membre fut également une raison.

Nous avons également choisi de créer une table Commune. Une personne y habite et un Terrain s'y situe. On évite la renondance et on peut appliquer la réduction au stagiaire plus aisément.

## 3.4 Difficultés rencontrées

— Lors de la conception, nous avons eu du mal à comprendre le cahier des charges qui était ambigu.
— Il n'était pas toujours pratique d'accéder à la base de données depuis nos machines personnelles pour la consulter et la modifier.

Globalement, nos problèmes se sont très rapidement résolus grâce à une communication efficace.

# 4 Mode d'emploi de l'application

L'application est dispose de plusieurs onglets. L'onglet `Ajouter` permet d'ajouter, selon le souhait de l'utilisateur, un stage, un membre ou un moniteur dans la base de données. L'onglet `Statistiques` permet de regarder les statistiques de cette base de données.

## 4.1 Ajout d'informations

Cette section se réfère à l'onglet `Ajouter`.

### 4.1.1 Ajout d'un stage

Pour ajouter un stage, l'utilisateur sélectionne la catégorie `Stage`. Un formulaire apparaît, dans lequel il doit indiquer l'heure de début et de fin du stage, le jour de l'année concerné par ce stage, le sport et le terrain. Le format de saisie doit être respecté. Si l'utilisateur ne le respecte pas, une boîte de dialogue apparaît et l'en avertit. Il doit ensuite choisir les stagiaires, les moniteurs et un superviseur.

### 4.1.2 Ajout d'un membre

Pour ajouter un membre dans la base, l'utilisateur sélectionne la catégorie `Personne`, puis `Membre`. Un formulaire apparaît dans lequel l'utilisateur doit indiquer l'identité complète du membre, en faisant attention au format.

### 4.1.3 Ajout d'un moniteur

De façon similaire, l'utilisateur remplit le formulaire concernant l'identité du moniteur à ajouter, cette fois-ci en sélectionnant la catégorie `Moniteur`. L'utilisateur sélectionne également les sports que le moniteur peut encadrer, ainsi que son domaine d'expertise.

## 4.2 Affichage des statistiques

Pour afficher les statistiques, l'utilisateur sélectionne l'onglet `Statistiques`, puis peut cliquer sur différents boutons selon ce qui l'intéresse. Le bouton `Actualiser` permet à l'utilisateur d'avoir accès aux statistiques de la base de données après un ajout d'informations récent.