Mini projet

Introduction aux bases de données

Base de données d'une salle de gym

Julien CHAN PENG

Adrien ASSOUAD

L2 - C

« Se dépasser, se surpasser », tel est le slogan de la salle de sport Malloc Your Muscle. Cette dernière souhaitait créer sa base de données afin de gérer plus facilement leurs adhérents et leurs formules d'abonnements ainsi que les sports qu'elle propose. Afin de réaliser leur souhait, nous avons divisé l'avancement de ce projet de création d'une base de données en deux parties. Durant la première séance du projet, nous avons créé la base de données avec toutes ses tables ainsi que ses contraintes. Puis, nous avons finalisé le projet en insérant des données dans la base et en rédigeant le rapport.

Comprendre la structure et le contenu de la base de données est primordiale afin d'établir le dictionnaire de données ainsi que le modèle E/A qui en découle, en justifiant nos choix d'attributs, de cardinalités et d'associations. Cela nous a permis d'avoir une image claire des données et des relations entre les entités.

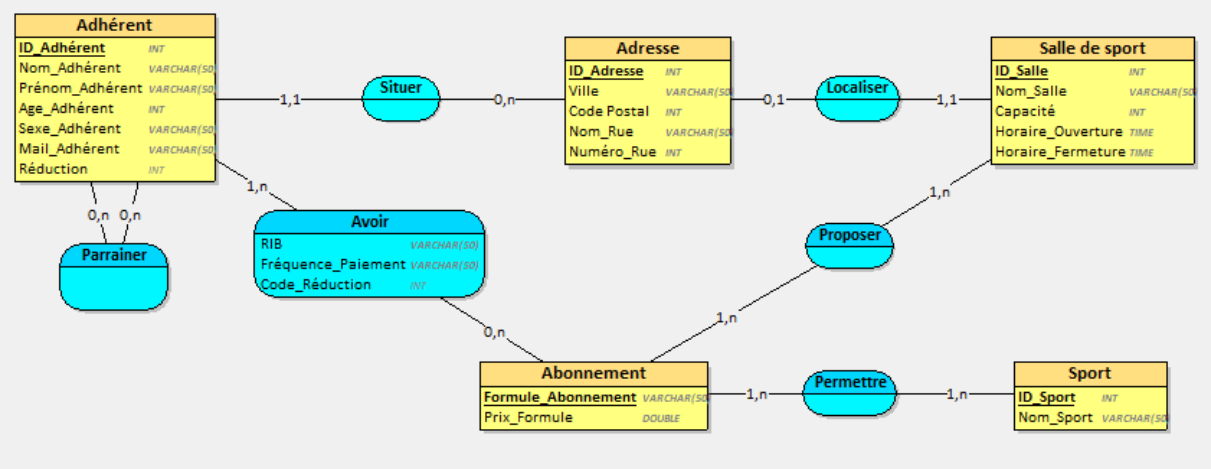
Une fois que le modèle E/A a été finalisé, nous avons donné le modèle relationnel correspondant et appliqué les règles de normalisation étudiées pour garantir que la base de données était sous une forme normalisée et qu'il n'existait aucune redondance.

Après la normalisation, nous avons créé la base de données en ajoutant les contraintes étudiées, à la fois les contraintes d'intégrité et de validation, pour garantir la cohérence et la validité des données. Ensuite, nous avons peuplé la base de données avec les contraintes ajoutées, en veillant à ce que les données respectent les exigences et les contraintes spécifiées.

Enfin, nous avons interrogé la base de données en utilisant les opérateurs étudiés, en nous basant sur des scénarios que nous avions imaginés, pour tester et valider la fonctionnalité de la base de données. Cela nous a permis d'identifier d'éventuelles incohérences, erreurs ou exigences manquantes.

Dans le cas de la base de données de Malloc Your Muscle, nous avons choisi de nous inspirer des offres des grandes chaînes de salles de sport telles que Fitness Park et Basic Fit afin d'en déduire les différentes entités de la base : Adhérent, Adresse, Salle, Abonnement et Sport.

Voici le modèle E/A créé sur Looping (modèle conceptuel de données) :



Choix des cardinalités :

- Un adhérent habite à une et une seule adresse mais une adresse peut avoir 0 ou plusieurs personnes.
- La salle de sport est localisée à une et une seule adresse mais une adresse ne peut avoir 0 ou 1 salle de sport.
- Un adhérent possède un ou plusieurs mais un abonnement peut avoir 0 ou plusieurs adhérents.
- La salle de sport propose plusieurs formules d'abonnement et un abonnement minimum est proposé par la salle de sport.
- Un abonnement peut permettre un ou plusieurs sports et un sport est minimum proposé par un abonnement.

En MLD (modèle logique de données) :

Sport = (ID_Sport *INT*, Nom_Sport *VARCHAR(50)*);
Abonnement = (Formule_Abonnement *VARCHAR(50)*, Prix_Formule *DOUBLE*);
Adresse = (ID_Adresse *INT*, Ville *VARCHAR(50)*, Code_Postal *INT*, Nom_Rue *VARCHAR(50)*, Numéro_Rue *INT*);
Salle_de_sport = (ID_Salle *INT*, Nom_Salle *VARCHAR(50)*, Capacité *INT*, Horaire_Ouverture *TIME*, Horaire_Fermeture *TIME, #ID_Adresse*);
Adhérent = (ID_Adhérent *INT*, Nom_Adhérent *VARCHAR(50)*, Prénom_Adhérent *VARCHAR(50)*, Age_Adhérent *INT*, Sexe_Adhérent *VARCHAR(50)*, Mail_Adhérent *VARCHAR(50)*, Réduction *INT, #ID_Adresse*);
Avoir = (*#ID_Adhérent, #Formule_Abonnement*, RIB *VARCHAR(50)*, Fréquence_Paiement *VARCHAR(50)*, Code_Réduction *INT*);
Permettre = (*#ID_Sport, #Formule_Abonnement*);
Proposer = (*#ID_Salle, #Formule_Abonnement*);
Parrainer = (*#ID_Adhérent, #ID_Adhérent_1*);

Grâce aux deux modèles de données (conceptuel et logique), nous avons pu réaliser plus facilement notre base de données sur MySQL avec la création des tables (entités seulement) :

```sql
-- Création de la table Sport
CREATE TABLE Sport(
    ID_Sport INT,
    Nom_Sport VARCHAR(50) NOT NULL,
    PRIMARY KEY(ID_Sport)
);

-- Création de la table Abonnement
CREATE TABLE Abonnement(
    Formule_Abonnement VARCHAR(50),
    Prix_Formule DOUBLE NOT NULL,
    PRIMARY KEY(Formule_Abonnement)
);
```

```sql
-- Création de la table Salle_de_sport
CREATE TABLE Salle_de_sport(
    ID_Salle INT,
    Nom_Salle VARCHAR(50) NOT NULL,
    Capacité INT NOT NULL,
    Horaire_Ouverture TIME NOT NULL,
    Horaire_Fermeture TIME NOT NULL,
    ID_Adresse INT NOT NULL,
    PRIMARY KEY(ID_Salle),
    UNIQUE(ID_Adresse),
    FOREIGN KEY(ID_Adresse) REFERENCES Adresse(ID_Adresse)
);
```

```sql
-- Création de la table Adhérent
CREATE TABLE Adhérent(
   ID_Adhérent INT,
   Nom_Adhérent VARCHAR(50) NOT NULL,
   Prénom_Adhérent VARCHAR(50) NOT NULL,
   Age_Adhérent INT NOT NULL,
   Sexe_Adhérent VARCHAR(50) NOT NULL,
   Mail_Adhérent VARCHAR(50) NOT NULL,
   Réduction INT,
   ID_Adresse INT NOT NULL,
   PRIMARY KEY(ID_Adhérent),
   FOREIGN KEY(ID_Adresse) REFERENCES Adresse(ID_Adresse)
);
```

```sql
-- Création de la table Adresse
CREATE TABLE Adresse(
   ID_Adresse INT,
   Ville VARCHAR(50) NOT NULL,
   Code_Postal INT NOT NULL,
   Nom_Rue VARCHAR(50) NOT NULL,
   Numéro_Rue INT NOT NULL,
   PRIMARY KEY(ID_Adresse)
);
```

En ce qui concerne les contraintes, nous avons réalisé plusieurs vérifications :

- un adhérent doit avoir minimum 16 ans pour s'inscrire en salle de sport ;

```sql
-- Age minimum requis
Alter table Adhérent
Add constraint CHK_AGE check (Age_Adhérent >= 16);
```

- le mail d'un adhérent doit respecter la syntaxe suivante ;

```sql
-- Syntaxe du mail
Alter table Adhérent
ADD CONSTRAINT CHK_MAIL CHECK (
    Mail_Adhérent LIKE '%@icloud.com' OR
    Mail_Adhérent LIKE '%@yahoo.fr' OR
    Mail_Adhérent LIKE '%@outlook.fr' OR
    Mail_Adhérent LIKE '%@gmail.com' OR
    Mail_Adhérent LIKE '%@efrei.net'
);
```

- le RIB (relevé d'identité bancaire) doit avoir 27 caractères et doit commencer par 'FR' ;

```sql
-- Syntaxe du RIB (27 caractères)
Alter table Avoir
Add constraint CHK_RIB check(LEFT(RIB, 2) = "FR" AND length(RIB) = 27);
```

- le choix du paiement doit être mensuel ou annuel ;

```sql
-- Vérification du paiement de l'abonnement
Alter table Avoir
Add constraint CHK_FREQUENTE check(Fréquence_Paiement in ("Mensuel","Annuel"));
```

- le choix de l'abonnement doit être soit Basic, soit Premium, soit Ultimate ;

```sql
-- Vérification du choix de l'abonnement
Alter table Abonnement
Add constraint CHEK_ABONNEMENT check (Formule_Abonnement in ("Basic","Premium","Ultimate"));
```

Une fois toutes les contraintes ajoutées, nous avons pu ensuite peupler notre base de données en renseignant les données liées à la salle de sport et celles des adhérents dans le fichier *Malloc_your_muscle_JeuxDonnées.sql* .

```
-- Association des données de la salle de sport
insert into Adresse value (1,"Paris",75000,"Rue Basic Frite",1);
insert into Salle_de_Sport value (1,"Malloc Your Muscle Paris",200,"06:00:00","23:00:00",1);
insert into Sport value (1,"Musculation"),(2,"Cardio"),(3,"Combat"),(4,"Escalade");
insert into Abonnement value ("Basic",19.99),("Premium",29.99),("Ultimate",39.99);
insert into Permettre value
(1,"Basic"),(2,"Basic"),
(1,"Premium"),(2,"Premium"),(3,"Premium"),
(1,"Ultimate"),(2,"Ultimate"),(3,"Ultimate"),(4,"Ultimate");
insert into Proposer value
(1,"Basic"),(1,"Premium"),(1,"Ultimate");
```

Nous avons décidé de créer vint-cinq adhérents mais il est simple d'ajouter de nouveaux adhérents en reprenant cette syntaxe de code.

```
-- Création du premier adhérent
insert into Adresse value (2,"Vitry",94400,"Rue de la Fraternité",31);
insert into Adhérent value (1,"Chan Peng","Julien",20,"Homme","julien.chanpeng@gmail.com",null,2);
insert into Avoir value (1,"Ultimate","FR00000000000000000000000001","Mensuel",NULL);

-- Création du second adhérent
insert into Adresse value (3,"Paris",75013,"Rue Damesme",53);
insert into Adhérent value (2,"Assouad","Adrien",20,"Homme","adrien.assouad@efrei.net",null,3);
insert into Avoir value (2,"Ultimate","FR00000000000000000000000002","Annuel",NULL);

-- Création du troisième adhérent
insert into Adresse value (4,"Vitry",9440,"Voie Schumann",35);
insert into Adhérent value (3,"Bial","Thibault",20,"Homme","thibault.bial@icloud.com",null,4);
insert into Avoir value (3,"Basic","FR00000000000000000000000003","Annuel",NULL);
```

Pour le quatrième adhérent, nous avons choisi de l'affecter à la même adresse que le premier adhérent d'où la ligne manquante sur l'adresse.

```
-- Création du quatrième adhérent
insert into Adhérent value (4,"Chan Peng","Arnaud",22,"Homme","arnaud.chanpeng@outlook.fr",null,2);
insert into Avoir value (4,"Premium","FR00000000000000000000000004","Mensuel",NULL);

-- Création du cinquième adhérent
insert into Adresse value (5,"Villejuif",94800,"Avenue de la République",32);
insert into Adhérent value (5,"Chabchoub","Kamel",40,"Homme","kamel.chabchoub@efrei.net",null,5);
insert into Avoir value (5,"Premium","FR00000000000000000000000005","Mensuel",NULL);
```

La totalité des attributs de chaque entité est en « not null » sauf pour l'attribut « Réduction » puisqu'il faut avoir un adhérent parrain et un adhérent parrainé.

Voici la création du système de parrainage :

```
-- Création du système de parrainage avec le 1er et le 2e adhérent
insert into Parrainer value (1,2);
update Adhérent set Réduction = 10 where ID_Adhérent = 1;
update Adhérent set Réduction = 5 where ID_Adhérent = 2;
```

```sql
-- Création du système de parrainage avec le 2e et le 3e adhérent
insert into Parrainer value (2,3);
update Adhérent set Réduction = 10 where ID_Adhérent = 2;
update Adhérent set Réduction = 5 where ID_Adhérent = 3;
```

Afin de nous assurer que l'insertion de données soit conforme, nous avons réalisé des contraintes différentes de celles du fichier *Malloc_your_muscle_ContraintesValidation.sql* .

```sql
-- Vérification des contraintes lors du peuplement
update Adhérent set Age_Adhérent =  12 where ID_Adhérent = 1;

update Adhérent set Mail_Adhérent = "ju.peng@wanadoo.de" where ID_Adhérent = 1;

update Avoir set RIB = "DE00000000000000000000000005" where ID_Adhérent = 1;

update Avoir set RIB = "FR000000000000000005" where ID_Adhérent = 1;

update Avoir set Fréquence_Paiement = "Hebdomadaire" where ID_Adhérent = 1;

update Adhérent set Formule_Abonnement = "Supreme" where ID_Adhérent = 1;
```

Ainsi, une erreur est affichée par le compilateur lors de l'exécution du programme.

⊗	55	13:12:31	update Adhérent set Age_Adhérent = 12 where ID_Adhérent = 1	Error Code: 3819. Check constraint 'CHK_AGE' is violated.	0.000 sec
⊗	56	13:12:35	update Adhérent set Mail_Adhérent = "ju.peng@wanadoo.de" ...	Error Code: 3819. Check constraint 'CHK_MAIL' is violated.	0.000 sec
⊗	57	13:12:37	update Avoir set RIB = "DE00000000000000000000000005" w...	Error Code: 3819. Check constraint 'CHK_RIB' is violated.	0.000 sec
⊗	58	13:12:49	update Avoir set RIB = "FR000000000000000005" where ID_...	Error Code: 3819. Check constraint 'CHK_RIB' is violated.	0.000 sec
⊗	59	13:12:53	update Avoir set Fréquence_Paiement = "Hebdomadaire" wher...	Error Code: 3819. Check constraint 'CHK_FREQUENTE' is viol...	0.015 sec
⊗	60	13:12:54	update Adhérent set Formule_Abonnement = "Supreme" where...	Error Code: 1054. Unknown column 'Formule_Abonnement' in 'f...	0.000 sec

Une fois notre de base de données peuplée, nous pouvons effectuer différentes requêtes de commandes pour chercher des informations ou vérifier la fonctionnalité de la base.

```
-- Recherche des noms et prenoms des adhérents qui ont un un abonnement Ultimate
select Nom_Adhérent, Prénom_Adhérent, Formule_Abonnement from Adhérent, Avoir where
Avoir.ID_Adhérent = Adhérent.ID_Adhérent and Avoir.Formule_Abonnement = "Ultimate";
```

Nom_Adhérent	Prénom_Adhérent
Chan Peng	Julien
Assouad	Adrien
Dupont	Cécile
Garcia	Pierre
Martin	Claire
Lecomte	David
Girard	Sylvie
Garcia	Manon
Moreau	Lucie
Dupuis	Emma

```
-- Recherche du RIB des adhérents
select Nom_Adhérent, Prénom_Adhérent, RIB
from Adhérent, Avoir where Avoir.ID_Adhérent = Adhérent.ID_Adhérent;
```

Nom_Adhérent	Prénom_Adhérent	RIB
Chan Peng	Julien	FR00000000000000000000000001
Assouad	Adrien	FR00000000000000000000000002
Bial	Thibault	FR00000000000000000000000003
Chan Peng	Arnaud	FR00000000000000000000000004
Chabchoub	Kamel	FR00000000000000000000000005
Dupont	Cécile	FR00000000000000000000000006
Leroy	Marc	FR00000000000000000000000007
Durand	Sophie	FR00000000000000000000000008
Garcia	Pierre	FR00000000000000000000000009
Lefevre	Laura	FR00000000000000000000000010
Dupont	Nicolas	FR00000000000000000000000011
Martin	Claire	FR00000000000000000000000012

Cette requête affiche donc le RIB des 25 adhérents mais par soucis de taille, l'image fait apparaître les 12 premiers adhérents.

```
-- Recherche des noms et prenoms des adhérents qui ont étaient parrainés et/ou étaient parrains
select GROUP_CONCAT(distinct Parrain.Nom_Adhérent, ' ', Parrain.Prénom_Adhérent
order by Parrain.Nom_Adhérent, Parrain.Prénom_Adhérent separator ', ') as Adhérents_Parrains,
GROUP_CONCAT(distinct Parrainé.Nom_Adhérent, ' ', Parrainé.Prénom_Adhérent
order by Parrainé.Nom_Adhérent, Parrainé.Prénom_Adhérent separator ', ') as Adhérents_Parrainés
from Adhérent as Parrain
join Parrainer on Parrain.ID_Adhérent = Parrainer.ID_Adhérent
join Adhérent as Parrainé on Parrainer.ID_Adhérent_1 = Parrainé.ID_Adhérent
group by Parrain.Nom_Adhérent, Parrain.Prénom_Adhérent;
```

Adhérents_Parrains	Adhérents_Parrainés
Assouad Adrien	Bial Thibault
Chan Peng Julien	Assouad Adrien
Garcia Manon	Garcia Thomas
Moreau Lucie	Moreau Mathieu

Cette requête a été l'une des plus difficile à mettre en place au niveau de la logique

```sql
-- Recherche d'une personne non adhérente
select * from Adhérent where Prénom_Adhérent = "Bob" and Nom_Adhérent = "Razozcki";
```

ID_Adhérent	Nom_Adhérent	Prénom_Adhérent	Age_Adhérent	Sexe_Adhérent	Mail_Adhérent	Réduction	ID_Adresse
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Il s'agit ici de vérifier la fonctionnalité de la base en cas d'une personne non référencée dans les données.

```sql
-- Affichage des abonnements avec les sports inclus et les prix
select Abonnement.Formule_Abonnement, Prix_Formule, GROUP_CONCAT(Nom_Sport order by Nom_Sport separator ', ')
as Sports_Disponibles from Abonnement, Permettre, Sport where
Abonnement.Formule_Abonnement = Permettre.Formule_Abonnement and Permettre.ID_Sport = Sport.ID_Sport
group by Abonnement.Formule_Abonnement, Prix_Formule;
```

Formule_Abonnement	Prix_Formule	Sports_Disponibles
Basic	19.99	Cardio, Musculation
Premium	29.99	Cardio, Combat, Musculation
Ultimate	39.99	Cardio, Combat, Escalade, Musculation

```sql
-- Informations de la salle
select Nom_Salle, Capacité, Horaire_Ouverture, Horaire_Fermeture, Numéro_Rue, Nom_Rue, Ville, Code_Postal
from Salle_de_sport, Adresse where Salle_de_sport.ID_Adresse = Adresse.ID_Adresse;
```

Nom_Salle	Capacité	Horaire_Ouverture	Horaire_Fermeture	Numéro_Rue	Nom_Rue	Ville	Code_Postal
Malloc Your Muscle Paris	200	06:00:00	23:00:00	1	Rue Basic Frite	Paris	75000

```sql
-- Selection des adhérents par odre décroissant en fonction de l'âge
select Nom_Adhérent, Prénom_Adhérent, Age_Adhérent
from Adhérent
order by Age_Adhérent desc;
```

Nom_Adhérent	Prénom_Adhérent	Age_Adhérent
Chabchoub	Kamel	40
Dupont	Cécile	40
Garcia	Pierre	35
Girard	Sylvie	34
Lecomte	David	33
Dupont	Nicolas	32
Roux	Marie	31
Moreau	Mathieu	31
Durand	Sophie	30
Moreau	Thomas	30
Petit	Philippe	29
Fournier	Isabelle	29
Moreau	Lucie	29
Lefevre	Laura	28
Leroux	Caroline	28
Martin	Claire	27
Dupuis	Emma	27
Lemoine	Alexandre	26
Garcia	Manon	26

```sql
-- Affiche les adhérents dont l'âge est supérieur à la moyenne d'âge
SELECT Nom_Adhérent, Prénom_Adhérent, Age_Adhérent, (SELECT AVG(Age_Adhérent)
    FROM Adhérent) as Moyenne_Age
FROM Adhérent
WHERE Age_Adhérent > (
    SELECT AVG(Age_Adhérent)
    FROM Adhérent);
```

Nom_Adhérent	Prénom_Adhérent	Age_Adhérent	Moyenne_Age
Chabchoub	Kamel	40	28.6400
Dupont	Cécile	40	28.6400
Durand	Sophie	30	28.6400
Garcia	Pierre	35	28.6400
Dupont	Nicolas	32	28.6400
Petit	Philippe	29	28.6400
Roux	Marie	31	28.6400
Lecomte	David	33	28.6400
Fournier	Isabelle	29	28.6400
Girard	Sylvie	34	28.6400
Moreau	Thomas	30	28.6400
Moreau	Lucie	29	28.6400
Moreau	Mathieu	31	28.6400

Cette requête est intéressante puisqu'elle fait intervenir les sous-requêtes ainsi que la fonction Average (AVG) qui permet d'obtenir la moyenne.

```sql
-- Nombre total d'adhérents dans chaque ville
SELECT Adresse.Ville, COUNT(Adhérent.ID_Adhérent) AS Nombre_Adhérents
FROM Adhérent
JOIN Adresse ON Adhérent.ID_Adresse = Adresse.ID_Adresse
GROUP BY Adresse.Ville;
```

Ville	Nombre_Adhérents
Vitry	5
Paris	2
Villejuif	1
Fontenay-sous-Bois	1
Vincennes	1
Ivry-sur-Seine	1
Alfortville	1
Charenton-le-Pont	2
Le Kremlin-Bicêtre	3
Gentilly	1
Maisons-Alfort	1
Saint-Mandé	1
Nogent-sur-Marne	1
Joinville-le-Pont	1
Arcueil	1
Le Perreux-sur-Ma...	1
Chennevières-sur-...	1

Ces différentes requêtes peuvent être souvent demandées par la salle de sport Malloc Your Muscle d'où la création de ces commandes. D'autres requêtes sont disponibles dans le fichier *Malloc_your_muscle_exploration.sql* .

Pour conclure, nous avons apporté une solution à la demande de la salle de sport Malloc Your Muscle en concevant une base de données opérationnelle en fonction de leurs besoins. L'un des points forts de la création de cette base de données est la simplicité et l'efficacité de la mettre en œuvre.

Cependant, nous avons noté quelques améliorations possibles pour cette base de données qui peuvent être l'ajout de données supplémentaires telles que plusieurs salles de sport dans différentes villes, des options d'abonnement en plus des sports accessibles ou encore la possibilité d'affecter des coachs à certains adhérents.

Ce type de projet est très formateur et intéressant à mettre en œuvre afin de développer ses connaissances en base de données, renforcer ses capacités d'analyse et ses compétences de réflexion. La prise en main de MySQL Workbench est bénéfique car c'est une application et un langage très populaire dans le monde professionnel, ce qui nous permet de nous préparer au mieux dans notre insertion professionnelle.