ENSEIRB-MATMECA
Filière Informatique

— **Projet de SGBD** —

Gestion d'une flotte de vélos électriques

Hugo Langlais Antoine Gaudy Guillaume Fornes

Encadré par S. Lombardy et S. Mosbah

Semestre 7 - 2021

Table des matières

1 Introduction

Ce rapport présente notre travail et avancement au cours du projet de SGBD proposé dans le cadre du cours d'IT204. Il s'agit de mettre en œuvre les notions et méthodes étudiées dans ce module, par la mise en place en partant de zéro d'une base de donnée cohérente(consultations, mises à jour, etc...) répondant à un problème, en passant par la modélisation des données, la gestion des dépendances... Ce projet a été réalisé en trinôme, par Hugo Langlais, Antoine Gaudy et Guillaume Fornes.

2 Modélisation des données

Afin d'élaborer un système de gestion base de données permettant de répondre au problème, nous devons dans un premier temps comprendre les besoins du client afin de modéliser par la suite un modèle entité association de la base que nous souhaitons créer.

2.1 Contexte de la base de donnée

Les systèmes de vélos empruntables momentanément pour un trajet ou plus se multiplient. Ces systèmes sont appréciés en ville, car ils permettent au client d'éviter d'avoir à acheter un vélo et résolvent le problème de la nécessitée d'un emplacement de stockage personnel de ce dernier.
Notre client désire ici développer son propre système de location de vélos électriques et souhaite donc mettre en place une base de donnée afin de gérer ce dernier. L'objectif est donc de pouvoir gérer une quantité de vélo, d'adhérents ainsi que les différentes stations des différentes communes concernées par le système. De plus il faudra pouvoir modéliser et répertorier les différents emprunts, ceci concernant donc un vélo, son utilisateur adhérent au système, le temps de trajet et le kilométrage effectué.

2.2 Modèle entité-association

Une fois le contexte étudier, il nous faut analyser les différentes opérations que désire pouvoir effectuer notre client. Elles aideront à définir notre schéma d'entité association, car le système devra s'assurer de la possibilité de leur réalisation. Celles-ci seront définies comme des requêtes dans notre base, séparées en deux types.
D'une part, nous aurons des requêtes de consultation, permettant d'obtenir des informations sur les vélos, stations, adhérents... De plus le client désire pouvoir lister les vélos par station, ceux en cours d'utilisation, les stations dans une commune, ou encore la liste des adhérents ayant emprunté au moins deux vélos différents dans une journée. Elles nous permettent d'établir les relations nécessaires entre les différentes tables, ainsi que des informations sur les attributs de ces dernières
D'autre part, il devra y avoir de requêtes de statistiques, notamment sur la moyenne du nombre d'usagers par vélo par jours, la moyenne des distances parcourues par les vélos par semaine, le classement des stations par nombre de places disponibles pour une commune, et enfin le classement des vélos par charges de batterie pour une station donnée.

Afin de modéliser les différents objets de notre système, nous allons créer les entités `VELO`, `STATION` et `ADHERENT` contenant les attributs nécessaires recherchés par les requêtes de consultation. Hormis cette consultation d'informations générales sur certaines entités, La majorité de ces requêtes concerne les emprunts effectués. Ainsi nous allons considérer l'entité `EMPRUNT` dont la position se dessine comme centrale dans la base de donnée. De plus pour éviter une redondance, une entité `COMMUNE` devra être

créer pour être liée avec les stations et les adhérents. Enfin nous avons choisi de créer une table `ETAT` associée à `VELO` afin de définir une liste possible d'états pour éviter encore une fois une répétition.

Concentrons-nous maintenant sur les associations de notre modèle. L'entité `emprunt`, centrée dans notre base, a donc nécessairement des associations de cardinalité 1, n avec `VELO`, `ADHERENT` et `STATION` (de départ), qui sont les entités concernant et définissant un emprunt. On retrouvera aussi une association 1, n entre `VELO` et `STATION`, pour exprimer le stockage de ce premier. Enfin une association 1, 1 *distance* relie l'entité `STATION` à elle-même pour définir la distance séparant 2 station.

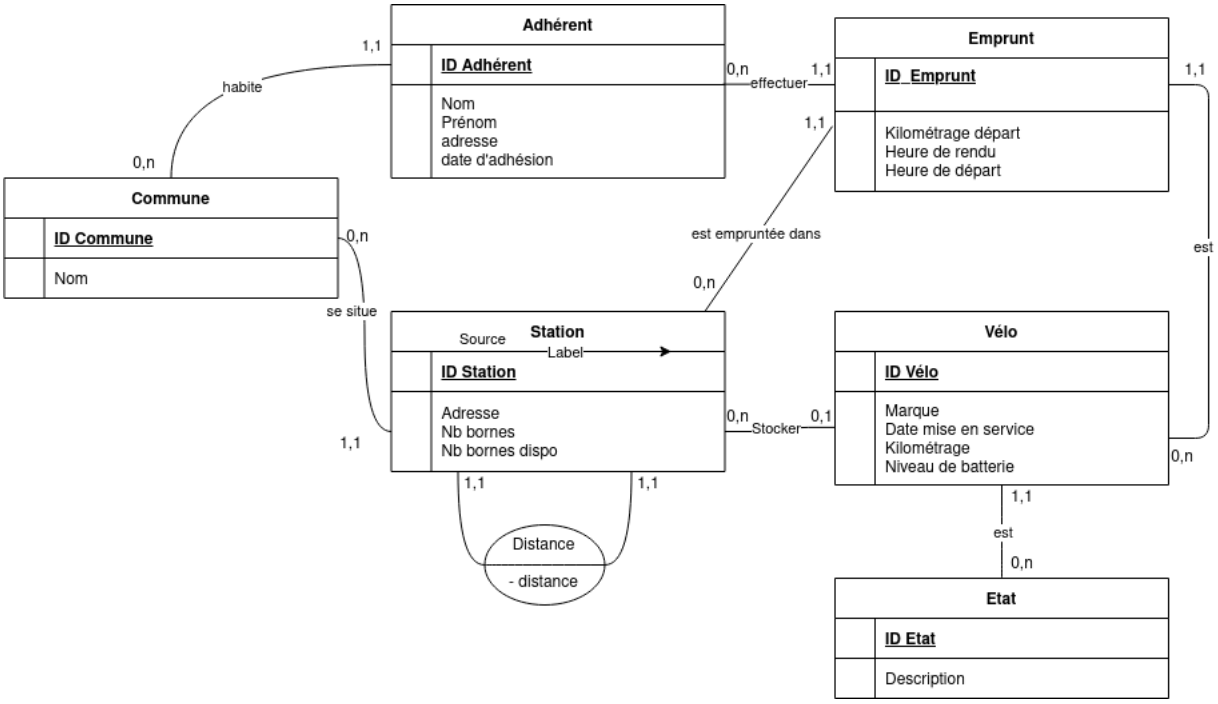

FIGURE 1 – Schéma entité-association

2.3 Schéma relationnel

Passons maintenant à la traduction de ce modèle associatif en un modèle relationnel.
Dans notre modèle entité association, on retrouve premièrement des associations 1, n. Dans ce cas la, la traduction au modèle relationnel se fait en définissant comme clé étrangère, pour l'entité de côté 1, la clé primaire de l'entité de côté n.

Le deuxième cas que l'on retrouve est celui de la relation de cardinalité 1, 1 qu'est *Distance* entre deux entité `STATION`. Celle-ci se retrouvera donc comme une table dans le modèle relationnel avec comme clé primaire les 2 clés étrangères des deux stations. Nous avons donc finalement le modèle relationnel suivant :

— ADHERENT (ID adherent, nom, prénom, adresse, date d'adhésion, #ID commune (non nul)) ;

— COMMUNE (ID commune, nom) ;
— STATION (ID station, adresse, nb bornes, nb bornes dispo, #ID commune (non nul)) ;
— DISTANCE (#ID station1 (non nul), #ID station2 (non nul), distance) ;
— VELO (ID vélo, marque, date mise en service, kilométrage, niveau de batterie, #ID station, #ID état (non nul)) ;
— ETAT (ID état, description) ;
— EMPRUNT (ID emprunt, km départ, heure) ;

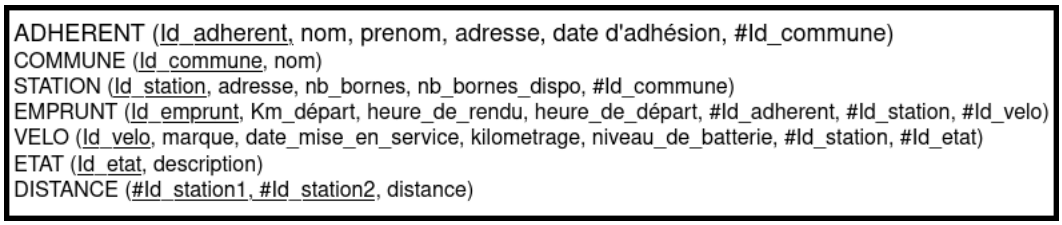

FIGURE 2 – Schéma relationnel de la base

3 Implémentation

Maintenant que nous disposons de notre schéma relationnel en 3e forme, passons à l'étape de la création de la base de donnée en sql.
Pour ce faire nous avons choisis d'utiliser MariaDB pour sa simplicité, sa rapidité, et le fait qu'il soit open-source, ce qui est important pour nous trois.

3.1 Création et remplissage de la base de donnée

3.1.1 Les tables

Nous avons donc commencé par implémenter les différentes tables brutes correspondantes à notre modèle relationnel, puis essayer de les remplir à l'aide de requêtes simples. De cette manière nous avons pu commencer à tester des premières requêtes basiques pour observer des début de résultats, avant de commencer à complexifié et compléter la base pour la rendre plus cohérente.

3.1.2 Les Triggers

Les différents triggers de notre implémentation sont faits en sorte que les données de la base restent cohérentes au fil des modifications apportées. La plupart des triggers servent majoritairement à modifier les champs des différentes tables lorsqu'une modification ou une insertion est apportée dans une table.

Lorsqu'on ajoute une station le nombre de bornes disponibles est mis automatiquement au maximum. Quand on veut ajouter un vélo dans une station le nombre de bornes disponibles est mis a jour si il y en a assez de libre.

3.1.3 Les procédures

Nous avons décidé de faire des procédures pour faciliter l'ajout d'emprunts et le rendu de ces derniers. Il suffit d'appeler la procédure avec l'id du vélo que l'on veut emprunter et l'id de la personne qui emprunte et la procédure gère si l'adhérent a déjà un emprunt en cours ou non, si le vélo est déjà emprunté, la mise a jour du nombre de bornes disponibles dans la station, la mise a jour des informations du vélo et enfin l'insertion de l'emprunt dans la base. Il existe une version de la procédure qui prends également une date en plus et crée un emprunt à cette date.

Une autre procédure gère le rendu des emprunts, il faut lui donner un id de station dans laquelle on veut rendre un vélo et l'id du vélo en question. Cette procédure vérifie si la station en question a assez de bornes disponibles, si le vélo est bel et bien en train d'être emprunté. La procédure met également a jour le kilométrage du vélo, le nombre de bornes disponibles dans la station et met l'heure de rendu à l'heure a laquelle la procédure a été appelée. Une autre procédure similaire peut prendre une heure en paramètre afin de rendre l'emprunt a une heure spécifiée. Nous avons également une procédure qui créée un nombre d'emprunts aléatoires sur la période des 7 derniers jours et les rend dans l'heure qui suit leur emprunt. Cette procédure nous sert à faire nos statistiques sur une plage de données assez significative.

3.2 Implémentation des requêtes

3.2.1 Les consultations

Les différentes requêtes de consultation de la base ont été assez simples à faire car c'était des requêtes basiques de sélection avec condition ou non.

3.2.2 Les statistiques

Les différentes statistiques ont été implémentées de deux différentes manières, Les requêtes simples ont été réalisées directement en SQL basique tandis que les statistiques plus complexes ont été faites via des fonctions.

4 Utilisation

Une fois nos bases de données implémentées, nous pouvions les manipuler depuis le terminal. Celle-ci étant peu ergonomique, nous avons décidé de développer une interface web.

4.1 Description de l'interface utilisateur

Le projet est basé sur 2 serveurs. Le premier est un serveur MariaDB, il s'occupe de gérer la base. Le deuxième est un serveur NodeJS qui fournit l'interface graphique du projet.

Nous avons construit le serveur Node afin qu'il mettre à disposition des API pour la page principale. Pour cela, nous avons utilisé le module express. Ce dernier permet de gérer très facilement le rootage, en utilisant des expressions régulières notamment, et de répondre différemment suivant la méthode de requête utilisée. C'est le serveur NodeJS qui s'occupe d'envoyer les requêtes au serveur MariaDB. L'utilisation d'API permet de rentre l'interface très incrémentale, ajouter une fonctionnalité revient à ajouter une route. Pour le rendu, nous avons utilisé Jquery et Bootstrap, afin de manipuler plus efficacement le DOM et de désigner rapidement l'interface. Pour l'affichage des tables, le plug-in Datatables s'est montré particulièrement efficace. La recherche sur les champs et la pagination sont

très utiles pour avoir une information rapidement.

La procédure d'installation ainsi que les dépendances du projet sont détaillées dans le fichier */front/README.md*.

4.2 Possibilité et utilisations

La page principale est découpée en 4 sections :
— Une section d'affichage du contenu d'une table, avec des boutons d'édition et de suppression
— Une fenêtre d'édition qui souvre après sélection d'une ligne
— Une section d'affichage des statistiques du projet
— Un formulaire pour insérer des éléments dans la table sélectionnée

L'ensemble des tables peut être affiché et modifié (sauf la table *distance*), ainsi que leurs champs, excepté leurs clés primaires. Nous avons décidé de ne pas pouvoir modifier la table distance afin de garder la cohérence de celle-ci. Pour l'affichage des statistiques, nous avons implémenté les indicateurs précisés dans le sujet.

Pour l'insertion d'éléments dans la base, nous avons ajouté au formulaire uniquement les champs strictement nécessaires. Le calcul des clés primaires, la création et le rendu d'un emprunt se font via des procédures. Nous avons par exemple uniquement besoin de l'id de l'adhérent et du vélo utilisé pour créer un emprunt, le reste des champs est calculé automatiquement. Notons qu'il n'y a aucune vérification du les informations saisies, le projet est donc vulnérable aux injections SQL.

5 Conclusion

Ce projet a été une excellente occasion de mettre en œuvre et d'approfondir nos connaissances en bases de données et en développement d'interface utilisateur. Un des éléments intéressants de ce projet est son incrémentabilité, ce qui nous a permit de continuer à l'améliorer sur toute sa durée avec les différentes fonctionnalités que nous avons apprises. Plusieurs amélioration étaient encore envisageables, comme la création d'une table `MARQUE` qui, à l'instar de `ETAT`, aurait permit d'éviter une répétition.