Projet BD - Rapport

Ludovic Carré, Alexandre Mouchet, Maxime Deloche, Vincent Lefoulon

I. Dépendances fonctionnelles et contraintes

DF	Cont. Valeur	Cont. Multiplicité
NomCategorie → DureeMax NomCategorie → PrixHoraire NomCategorie → MontantCaution	Categorie ⊂ {VoitureElectrique, Velo, VeloElectrique, VeloRemorque, Utilitaire} MontantCaution >= 0 DureeMax > 0 PrixHoraire > 0	NomCategorie --> NumVehicule (0..*)
NumVehicule → NomCategorie NumVehicule → NbSieges	NbSieges > 0	NumVehicule --> NomStation (0..1) *Un véhicule peut être garé ou non.* NumVehicule --> NumLocation (0..*) *Un véhicule peut être loué ou non.*
NomStation → AdresseStation NomStation, NomCategorie → NbPlacesDispo	NbPlacesDispo >= 0	NomStation --> NbPlacesDispo, NomCategorie (1..*) *Une station a au moins des places pour une catégorie.* NomStation --> NumLocation (0..*) *D'une station peuvent partir zéro ou plusieurs locations.* NomStation --> NumVehicule (0..*) *Un ou plusieurs véhicules sont garés à une station.*
NumCBAbonne → NomAbonne NumCBAbonne → PrenomAbonne NumCBAbonne → DateNaissanceAbonne NumCBAbonne → AdresseAbonne	DateNaissanceAbonne < TODAY	NumCBAbonne, NomCategorie ->> NumForfait (0..1) *Un abonné a au plus un forfait par catégorie* NumCBAbonne ->> NumLocation (0..*)
NumForfait →NomCategorie NumForfait →NumCBAbonne NumForfait →Prix	Prix > 0	
NumForfaitDureeLimite →Duree NumForfaitDureeLimite →DateDebut	Ext(NumForfaitDureeLimite) ⊂ Ext(NumForfait)	

NumForfaitDureeLimite →Reduction		
NumForfaitNombreLimite →NbLocations	Ext(NumForfaitNombreLimite) ⊂ Ext(NumForfait)	
NumLocation → NumCBAbonne NumLocation → NumVehicule NumLocation → DateDebut NumLocation → NomStation		NumLocation, NomStation --> DateFin (0..1) *Une location, si elle est terminée, a une station d'arrivée et une date de fin.*

Pour chaque catégorie de véhicule, on précise la durée maximale d'utilisation et le prix horaire de location, ainsi que le montant de la caution qui sera prélevée si la durée maximale est dépassée.

Chaque véhicule a un identifiant unique, appartient à une catégorie, et offre un certain nombre de places.

Chaque station, localisée par son adresse, offre un certain nombre de places pour les catégories de véhicules qu'elle peut accueillir.
Dans différents endroits de la ville se trouvent des stations (identifiées par leur nom) où les abonnés peuvent louer un véhicule d'une certaine catégorie.

Les abonnés, identifiés par leur numéro de carte bancaire et décrits par leur nom, prénom, date de naissance et adresse postale, ont une réduction de 25% sur le tarif plein pour chaque forfait et de chaque location s'ils ont moins de 25 ans ou plus de 65 ans. Chaque abonné peut avoir un ou plusieurs forfaits (au plus un par catégorie de véhicule).

Un forfait, identifié de façon unique, concerne une seule catégorie de véhicule et un seul abonné. Il permet soit un nombre illimité de locations pendant une durée limitée (jour, mois ou année), soit un nombre de locations limité pour une période illimitée.
Les forfaits à durée limitée sont décrits par leur durée, la date de début de validité, un prix et une remise fonction de la durée, appliquée aux locations.
Les forfaits à locations limitées sont décrits par leur prix et le nombre maximum de locations (gratuites).

On souhaite garder une trace de tous les locations des véhicules: abonnés, dates, heures et stations de départ et d'arrivée.
On souhaite également connaître à tout moment la station où est garé n'importe quel véhicule qui ne soit pas en location.

Les véhicules à gérer sont de diverses catégories, ils peuvent être des voitures électriques, des vélos, des vélos électriques, des vélos avec remorque, ou des petits utilitaires.

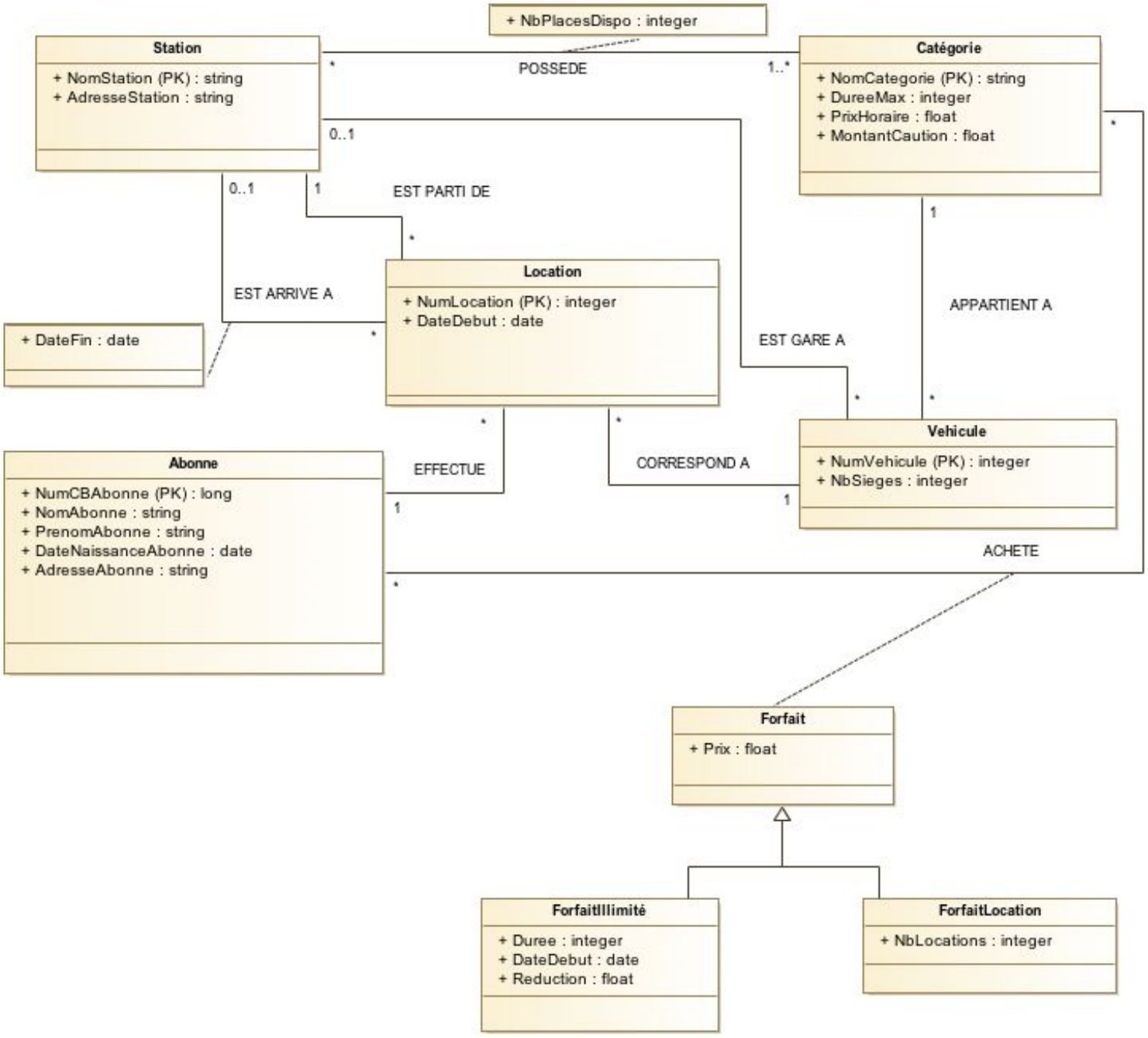
Contraintes prises en charge par la base de données

- La vérification de l'intégrité des valeurs pour les attributs:
    - MontantCaution >= 0
    - DureeMax > 0
    - PrixHoraire > 0
    - NbSieges > 0
    - NbPlacesDispo >= 0
    - DateNaissanceAbonne < TODAY
    - Prix > 0
    - 0 <= Reduction < 1
- Un abonné ne pourra prendre qu'un forfait à durée limitée par catégorie.
- Un abonné ne pourra prendre qu'un forfait à nombre de locations limité par catégorie.

Contraintes non prises en charge par la base de données

- Une réduction de 25% sur le tarif plein pour chaque forfait et de chaque location s'ils ont moins de 25 ans ou plus de 65 ans.
- Une réduction peut être appliquée à une location si l'abonné possède un forfait à durée limitée.
- Chaque location est gratuite la première heure.
- Un abonné ne peut pas prendre un forfait à durée limitée et un forfait à nombre de locations limité pour une même catégorie.
- Lorsqu'un forfait est terminé, le tuple est supprimé.
- Lorsqu'un véhicule est loué, le tuple correspondant au nom de la station où il était garé doit être supprimé.
- Lorsqu'un véhicule est garé, la station, où il est garé, doit correspondre avec la dernière station où il a été rendu. (Si celui-ci a déjà été loué au moins une fois)
- Lorsqu'un véhicule est garé à une station à la fin d'une location, la station doit posséder une place disponible pour ce type de véhicule.
- Un véhicule ne peut pas être loué deux fois sur la même période.
- Un véhicule ne peut pas être loué si celui-ci est loué par un autre abonné.

II. Schéma entité - association



III. Traduction en relationnel

1. Schéma

- **Station**(NomS, AdresseStation)
- **Categorie**(NomCategorie, DureeMax, PrixHoraire, MontantCaution)
- **StationCategorie**(NomStation(Station.NomStation), NomCategorie(Categorie.NomCategorie), NbPlacesDispo)
- **Abonne**(NumCBAbonne, NomAbonne, PrenomAbonne, DateNaissanceAbonne, AdresseAbonne)
- **Vehicule**(NumVehicule, NbSieges, NomCategorie(Categorie.NomCategorie))
- **Location**(NumLocation, DateDebut, StationDebut(Station.NomStation), NumCBAbonne(Abonne.NumCBAbonne), NumVehicule(Vehicule.NumVehicule))
- **StationLocation**(NumLocation(Location.NumLocation), StationFin(Station.NomStation), DateFin)
- **StationVehicule**(NumVehicule(Vehicule.NumVehicule), NomStation(Station.NomStation))
- **Forfait**(Abonne(Abonne.NumCBAbonne), Categorie(Categorie.NomCategorie), Prix)
- **ForfaitDureeLimitee**(Abonne(Abonne.NumCBAbonne), Categorie(Categorie.NomCategorie), Duree, DateDebut, Reduction)
- **ForfaitNbLimite**(Abonne(Abonne.NumCBAbonne), Categorie(Categorie.NomCategorie), NbLocationsRestantes)

2. Forme normale

- **1FN :** OK => tout attribut a une valeur atomique
- **2FN :** OK => tout attribut non clé est pleinement dépendant des clés
- **3FN :** OK => tout attribut non clé ne dépend pas d'autres attributs non-clés
- **3FNBCK :** OK => tous les ensembles de dépendances fonctionnelles ne contiennent pas de redondance, ce qui implique qu'ils sont des couvertures minimales de leurs fermetures et puisque ces ensembles sont 3FN ils sont 3FN-BCK, pas de redondances dans les dépendances.

IV. Requêtes SQL

1. Création de tables

- script @create_database.sql

2. Insertions dans les tables

- script @inserts_database.sql

3. Suppression des tables

- script @drop_database.sql

En pratique, le script @create_all.sql appelle ces 3 scripts : il supprime les tables si elles existent, les recrée et les remplit avec les valeurs initiales.

4. Test du fonctionnement des requêtes

- script @tests_requests.sql

5. Requêtes SQL

a. Requêtes de fonctionnalité

Les requêtes SQL des fonctionnalités se trouvent dans le fichier SQL_request.md.

b. Requêtes de vérification

Nous avons ajouté une 6e entrée à l'IHM permettant de vérifier l'intégralité de la table. Nous vérifions :

- qu'il y a suffisamment de places pour tous les véhicules garés
- que les stations où les véhicules sont garés sont cohérentes avec les locations terminées
- qu'il y a au plus un forfait par catégorie et par abonné
- qu'il n'y a pas de forfait terminé
- qu'il n'y a pas plusieurs locations pour le même véhicule en même temps

Les requêtes sont présentes dans le fichier SQL_check_requests.md.

V. Mode d'emploi de l'application

1. Lancement

- *make vehlib* : compile l'application
- *make run* : lance l'application

2. Fonctionnement

Notre application est basée sur une architecture MVC :

- Model : traitement des données et interaction avec la base de données
- Controller : gestion des requêtes de l'utilisateur
- View : interface utilisateur

VI. Bilan

Le travail s'est effectué en suivant les étapes une par une, dans l'ordre. Pour les trois premières, l'analyse, la conception entités/associations et la construction du schéma relationnel, nous avons fait 2 équipes de 2, chacune étudiant le sujet de son côté. Nous avons ensuite comparé les résultats et avons échangé sur les points où nos opinions divergeaient, en interrogeant le professeur.

Nous avons par exemple eu du mal à déterminer quelles classes seraient des associations, particulièrement pour la classe Forfait. Il s'est agit de déterminer si la contrainte d'unicité entre un forfait, un utilisateur et un type de véhicule se devait d'être représentée par une classe association, ce qui implique l'unicité par construction, ou par une classe classique. Dans le premier cas, il faut supprimer les forfaits à la fin de leur validité et dans le second, l'ensemble des forfaits est conservé dans la table et la contrainte est prise en compte au niveau de l'application.

Ces étapes nous ont pris un temps non négligeable, et nous avons ensuite opté pour une découpe du travail restant afin de ne pas prendre trop de retard :

- Un étudiant a créé les tables et les a remplies
- Un étudiant a traduit les fonctionnalités demandées en requêtes SQL
- Un étudiant s'est occupé de l'IHM en Java
- Un dernier étudiant s'est chargé de la connexion à la base de données en Java

Nous avons changé les requêtes plusieurs fois lorsque des précisions sur les fonctionnalités ont été ajoutées sur le forum.

L'implémentation en Java n'a pas posé de problème particulier. Pour la création et le remplissage des

tables, il aurait été plus pratique de pouvoir bénéficier des fonctionnalités de SQL 3. Nous nous sommes régulièrement interrogés sur la manière de répartir les calculs : faut-il en effectuer un maximum en SQL, ou peut-on effectuer quelques traitements en Java lorsque c'est beaucoup plus simple ?

Nous n'avons besoin d'utiliser une transaction que dans le cas de la facturation d'une location : en effet, c'est la seule fonctionnalité où il faut modifier des tables. Pour le niveau d'isolation des transactions, nous aurions choisi le niveau 'Repeatable Read' : les autres transactions ne peuvent pas modifier les données lues par la transaction et la transaction ne peut pas lire les données non validées par les autres transactions. Mais le serveur nous retourne une erreur indiquant que les seuls niveaux valides sont 'Read Committed' et 'Serializable' : nous avons supposé que ceci venait de la configuration du serveur, et avons donc choisi le niveau 'Serializable', même s'il ajoute des sécurités non nécessaires.