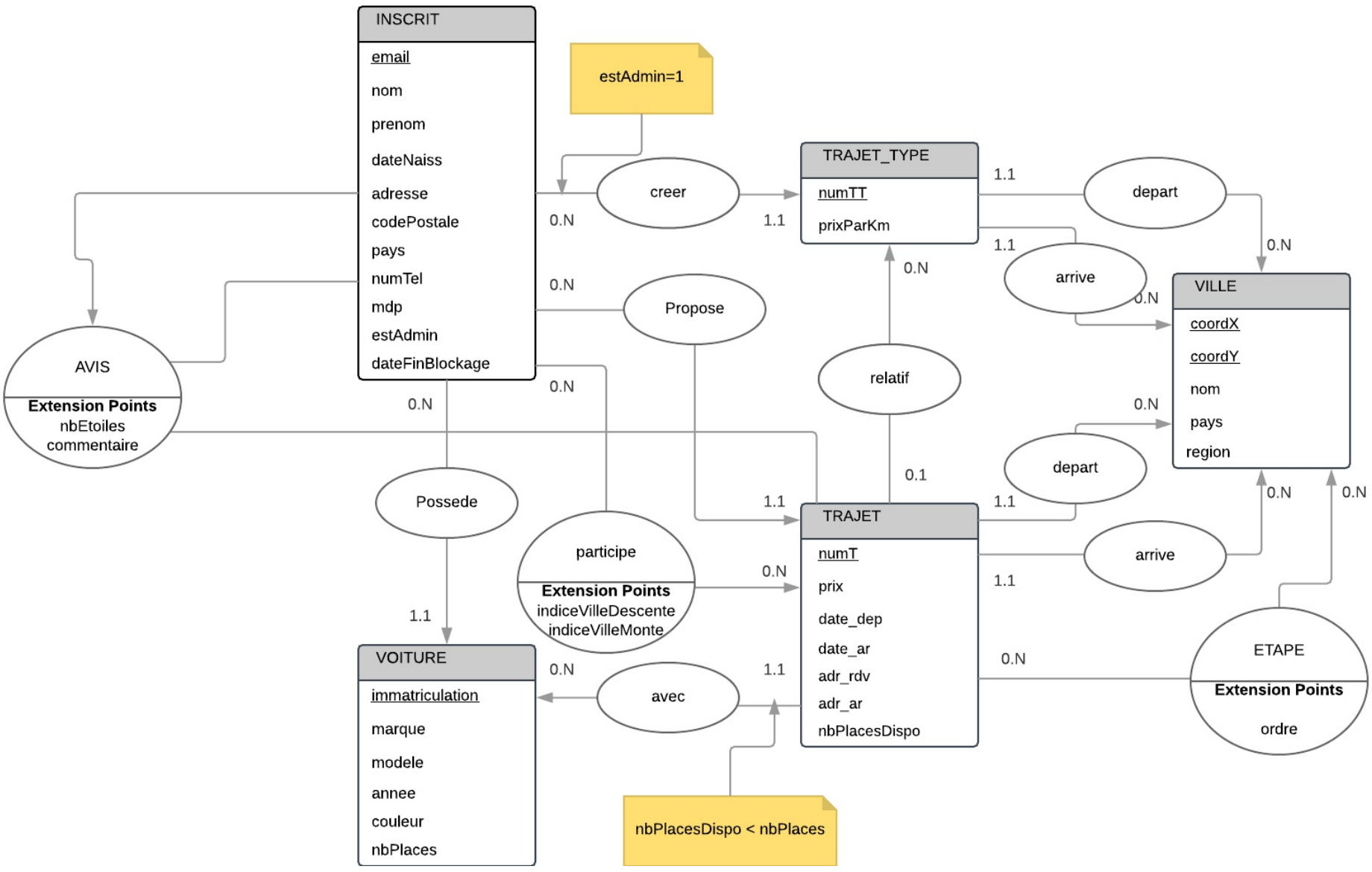
RAPPORT DU PROJET BDD L3 2017

Wissem Soussi et Jérémie Dautheribes

Décembre 2017

1) MODÈLE ENTITÉ ASSOCIATION



2) MODÈLE RELATIONNEL

INSCRIT (email , nom , prenom , dateNaiss , adresse , codePostale , pays , numTel , mdp , estAdmin , dateFinBlockage)

VOITURE (immatriculation , marque , modele , annee , couleur , nbPlaces , *emailProprietaire*)

VILLE (coordX , coordY , nomV , pays , region)

TRAJET_TYPE (numTT , prixParKm , *villeDepX* , *villeDepY* , *villeArrX* , *villeArrY* , *emailAdmin*)

TRAJET (numT , prix , date_dep , date_ar , adr_rdv , adr_ar , nbPlacesDispo , *conducteur* , *vehiculeImm* , *villeDepX* , *villeDepY* , *villeArrX* , *villeArrY* , *numTrajetType*)

ETAPES (*numT* , *coordX* , *coordY* , ordre)

PARTECIPER (*numT* , *emailCovoitureur* , iVM , iVD)

AVIS (*numT* , *numDonneur* , *numRéceveur*, nbEtoile, commentaire)

3) ÉCLAIREMENT SUR LA CONCEPTION

Puisque nous n'avons pas réalisé la partie web du projet, nous avons décidé de rajouter des triggers et procédure pour imiter une utilisation telle qu'elle aurait du être dans la partie web. Nous allons maintenant expliqué un peu le rôle des tables, triggers et procedures que nous avons codés.

- Fonctionnalités liées à un internaute non membre (simple visiteur) :

Dans le fichier “invite.sql”, nous avons décrit ce qu'un internaute qui rentre dans le site sans s'identifier peut faire : il peut rechercher des trajets en insérant la ville de départ, la ville d'arrivée, une date de trajet et une “tolérance” de jours : si par exemple la date est le 10/01/2018 et la tolérance est de 5 jours, les trajets affichés seront ceux du 05/01/2018 jusqu’à 15/01/2018. Si la borne inférieure de cet intervalle est inférieure à la date courante alors la borne inférieure va être changée par la date courante. (j'ai pas compris ce que tu veux dire) Les trajets vont être affichés par ordre croissant des dates de départ et ceux qui n’ont plus de places disponibles ne seront pas pris en compte. Pour chaque trajet on affichera d’une façon supplémentaire les dates de départ et d'arrivée, la distance en kilomètres, le prix et le nom du conducteur (sans autres informations sur ce dernier puisque l'internaute est juste un visiteur non identifié). Pour avoir plus d’informations sur le conducteur d’un trajet et pour participer à ce dernier il faudra s’inscrire (fichier “invite.sql”) et devenir membre du site.

- Fonctionnalités liées aux membres et admin

Le membre sera identifié par un email, et il devra renseigner obligatoirement son nom, prénom, sa date de naissance, son numéro de téléphone et un mot de passe (fichier “createTablesMYSQL.sql” -> `TABLE inscrit`). La notation des membres est assurée par un système d'étoile (de 0 à 5, comme usuellement). Il n’y a pas de limite d’âge, par contre la date de naissance ne doit pas bien-sûr être supérieure à la date courante et nous avons choisi de fixe l'âge maximal à 120 ans (fichier “createTablesMYSQL.sql” -> `TRIGGER VERIF_INSCRIT`).

Note: chaque trigger dans la base de données est dupliqué en trigger `before insert` et trigger `before update` puisque MYSQL ne permet pas de faire `insert OR update` . Pour l'utilisation des `CHECK`, nous les avons laissés dans le code même s'ils ne sont pas gérés par MySQL (ils sont quand même parsés). Ils ont néanmoins bien été implémentés en utilisant des triggers.

Un membre peut être obtenir le statut d'admin uniquement grâce à un autre admin, nous considérons donc qu’à la création de la base, il y aura un membre défini comme admin par l’administrateur (fichier “admin.sql” -> `PROCEDURE Promotion_Membre`).

Un admin peut aussi bannir un membre en définissant une date où le bannissement prendra fin. De la même façon, il pourra débloquer un membre en définissant une date de fin blocage antécédent à la date courante (fichier “admin.sql” -> `PROCEDURE Blockage_Membre`).

Un admin peut aussi enregistrer dans la base de données des villes (fichier “admin.sql”) en mettant leur nom, pays et région relative et identifiées par les coordonnées GPS (longitude et latitude) (fichier “createTablesMYSQL.sql” -> `TABLE ville`).

Enfin l’admin peut créer des trajets types identifiés par une ville de départ, une ville de retour, et avec un prix par kilomètre maximal, c'est à dire que les membres ne pourront pas proposer ce trajet avec un prix par kilomètre supérieur à celui renseigné dans la table (fichier “createTablesMYSQL.sql” -> `TABLE trajet_type`). Le trajet type est referencé par l’admin qui l’a créé (fichier “triggersMysql.sql” -> `TRIGGER VERIF_TRAJET_TYPE_INSERT`).

Un membre peut enregistrer dans la base de données une voiture qu’il utilise pour le covoiturage; la voiture sera identifiée par l’immatriculation, et devra obligatoirement avoir une marque, un modèle, un nombre de places et l’email du propriétaire qui doit être un membre du site (fichier “createTablesMYSQL.sql” -> `TABLE voiture`). La date d’immatriculation, si spécifiée, doit être inférieure à la date d’aujourd’hui et supérieure au 1883 (premières voitures commercialisées) et le nombre de places se situe entre 0 et 9 (si supérieur à 9 on parle de mini-bus et non plus de voiture) (fichier “createTablesMYSQL.sql” -> `TRIGGER VERIF_VOITURE`).

Un membre propose un trajet ....(table trajet)

conditions trajets....(trigger trajet et procedures(fonctions) liees)

Un membre participe à un trajet ....(table participer)

conditions participer....(trigger participer et procedures(fonctions) liees)

Un membre peut donner un avis ....(table avis)

conditions avis....(trigger avis et procedures(fonctions) liees)

Un membre peut faire la recherche comme l'invite' mais il voit tous les infos sur le conducteur

EXEMPLE DE CODE SQL DANS MARKDOWN ;)

```
insert into voiture (immatriculation, marque, modele, annee, couleur, nbPlaces, emailProprietaire)
values ("value", "value", "value", "value", "value", "value", "value");
```