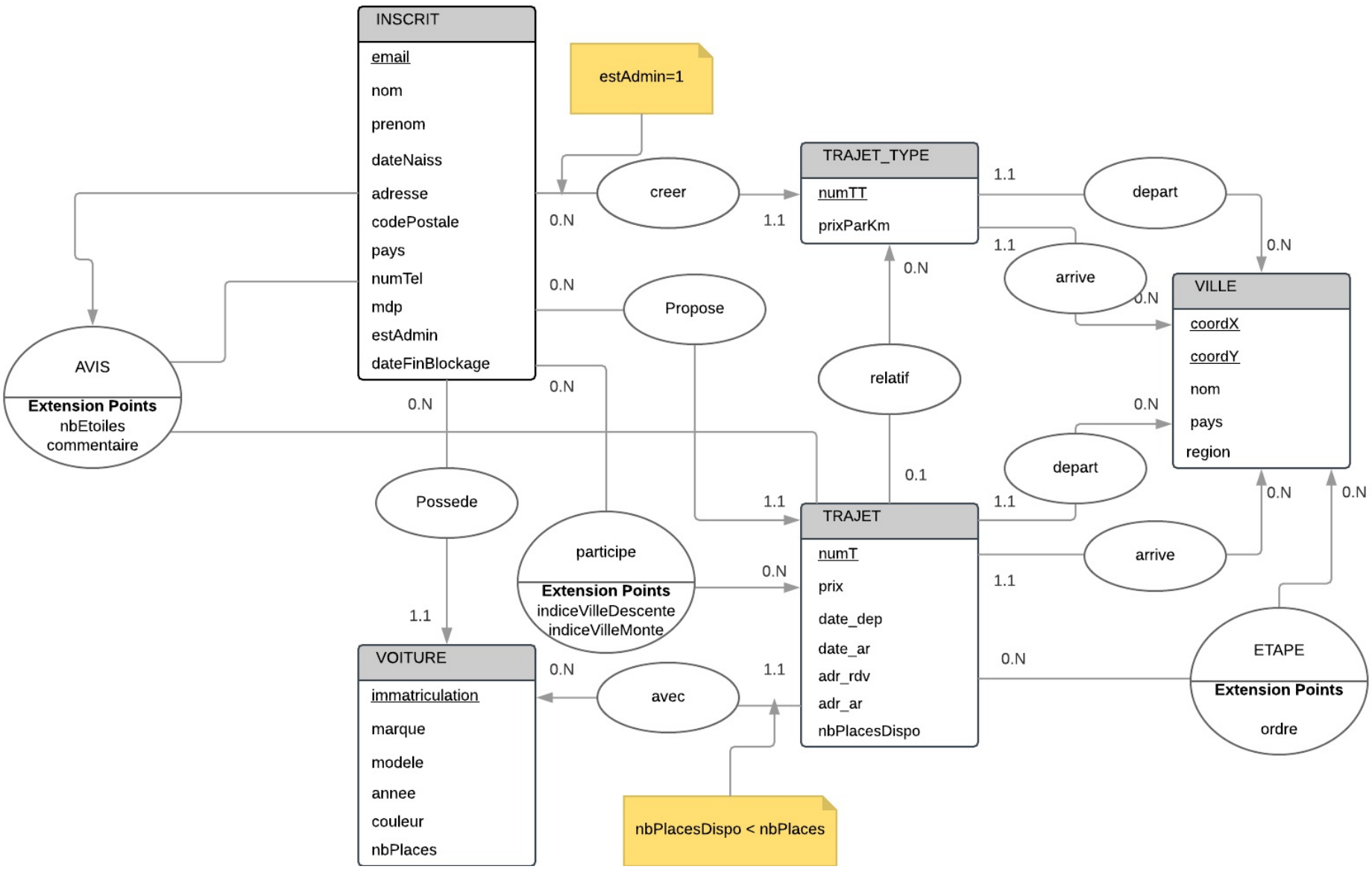
RAPPORT DU PROJET BDD L3 2017

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 , dateFinBlocage)

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)

PARTICIPER (*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 antérieure à la date courante, alors la borne inférieure sera la date courante. 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 kilomètre, 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`). Un membre peut faire une recherche de trajet comme l'invité mais il verra toutes les informations propres au conducteur

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 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 antérieure 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 référencé 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 place 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`).

À FAIRE : RAJOUTER L'INSERTION AUTOMATIQUE DE L'HEURE D'ARRIVÉE AUTOMATIQUEMENT (mettre à jour en conséquences)

Un membre peut proposer un trajet. Pour cela il doit insérer obligatoirement son identifiant (email), sa voiture utilisée, le nombre de place qu'il souhaite autoriser, les heures de départ et d'arrivée, les villes de départ et d'arrivée ainsi que leur adresse. Avant l'insertion du trajet dans la base, de nombreuses conditions seront vérifiées (fichier "triggersMysql.sql" -> `TRIGGER VERIF_TRAJET`).
Tout d'abord, si le membre est bloqué, il ne pourra pas créer de trajet. Si le trajet proposé correspond à un trajet type, alors le trajet sera référencé par ce trajet type (et le prix sera vérifié pour qu'il soit inférieur ou égal à celui du trajet type). Sinon nous vérifions, grâce à une procédure, que le prix au km est inférieur ou égal à 0,10€/km (limite arbitraire qui nous parait assez réaliste). La distance du trajet est calculée automatiquement par une procédure à partir de la longitude et la latitude des villes d'arrivée et de départ, en prenant un trajet en ligne droite et en prenant en compte la rondité de la terre. Nous vérifions également que le membre possède une voiture, si non, il devra en ajouter une. La voiture sélectionnée doit être celle du conducteur ; cela n'aurait pas trop d'intérêt si nous avions l'interface web, mais dans le contexte de la BDD nous avons préféré le rajouter. Le nombre de place proposé doit être supérieur à 0 et inférieur aux capacités de la voiture (le conducteur prenant forcément une place). Nous vérifions aussi la date départ, qu'elle soit bien supérieure à la date courante, et qu'elle se situe dans, au maximum, les six prochains mois à venir. Enfin, pour la date d'arrivée, nous vérifions qu'elle se situe bien après la date de départ.

Un trajet peut comporter plusieurs villes étapes entre la ville de départ et la ville d'arrivée (fichier “createTablesMYSQL.sql” -> `TABLE etapes`). Ces

villes étapes sont identifiées par leur ordre de parcours dans le trajet.

Un membre peut participer à un trajet, en insérant les villes où il va monter et descendre entre les villes disponibles dans le trajet: ville de départ, villes étapes, ville d'arrivée (fichier “createTablesMYSQL.sql” -> `TABLE participer`). Si la ville “d’embarcation” (iVM) est la ville de départ alors sa valeur est NULL, vice-versa iVD est NULL si la ville de descente est la ville d'arrivée : personne ne peut donc descendre à la ville de départ et personne ne va monter dans la ville d'arrivée (logique et convenable). Autrement, iVM et iVD vont contenir l’ordre respectivement de la ville de d'embarcation et de la ville de dépôt dans ce (sous-)trajet (l’attribut ordre dans la table `etapes`).

La participation est ajoutée seulement s’il y a de la place disponible pour ce sous-trajet. C’est-à-dire que si pour ce bout de trajet (de iVM à IVD) il y a une étape où il n'y a pas de place disponible, la participation ne peut pas être prise en compte. Aussi, le membre et le conducteur du trajet ne doivent pas être bloqués au moment de la souscription du trajet. Enfin l’ordre de la ville de monté/d'embarcation doit être bien sûr inférieur à l’ordre de la ville de descente (on ne peut pas descendre dans une ville qui a déjà été parcourue) (fichier “triggersMysql.sql” -> `TRIGGER VERIF_PARTICIPER_NODRIVER`).

Un membre peut donner un avis avec un système d’étoile (comme décrit plus haut). Le nombre d’étoile est obligatoire, mais l’utilisateur peut rajouter s’il le souhaite un commentaire écrit. Un avis est associé au minimum à un numéro de trajet, à l’identifiant du donneur de l’avis, à l’identifiant du receveur de l’avis et donc au nombre d’étoile. Avant l’insertion d’un avis, plusieurs conditions doivent être respectées. En premier lieu, l’avis ne peut pas être posté avant la fin du trajet. Le donneur de l’avis ne peut pas se donner un avis à lui-même. Si le donneur de l’avis est un covoitureur, alors il ne peut donner son avis qu’au conducteur. Enfin, s’il est conducteur, il peut donner un avis à chaque autre covoitureur présent lors du trajet.