université PARIS DIDEROT PARIS 7

BD3

Projet de construction de Base de Données

Novembre – Décembre 2014

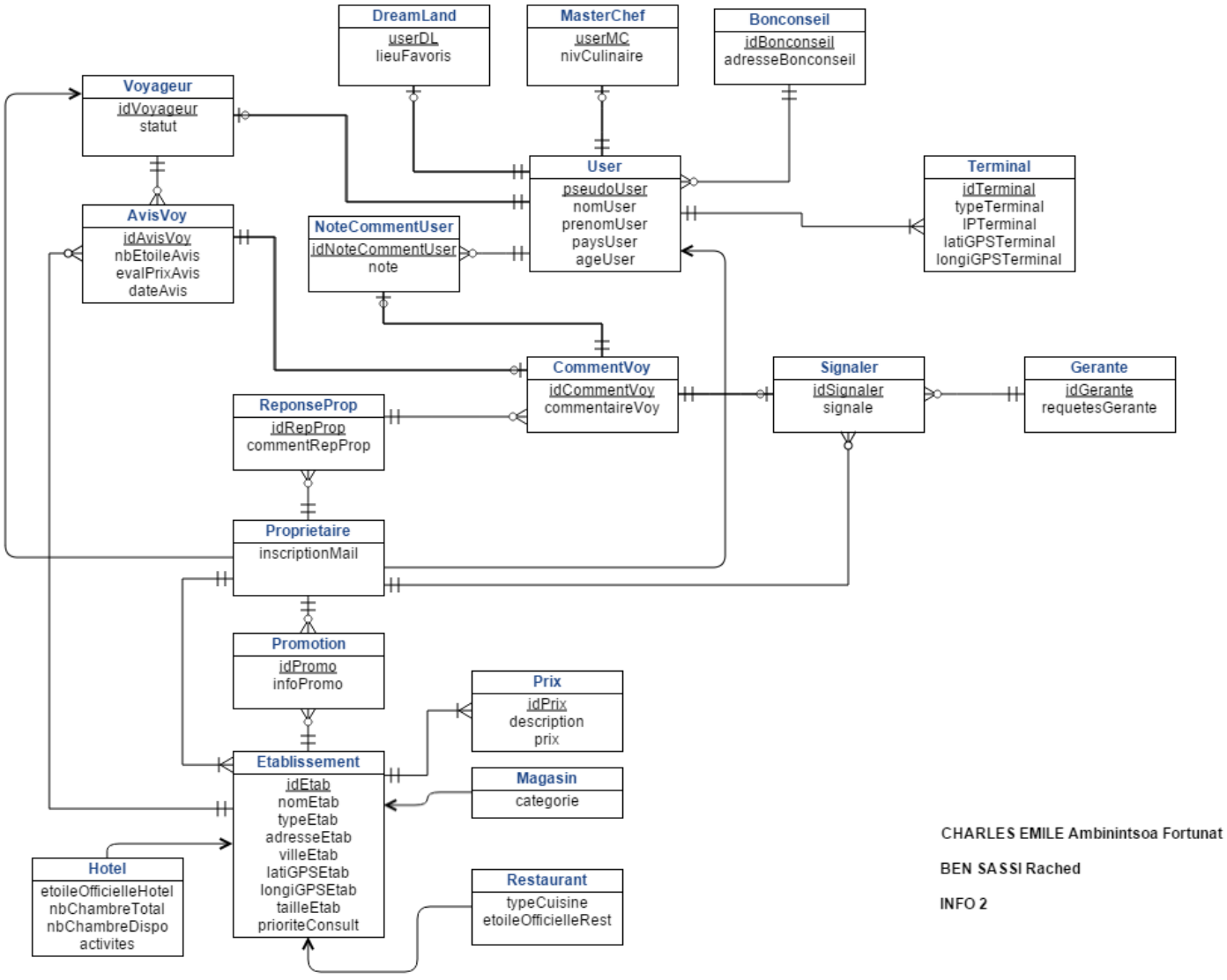
CHARLES EMILE Ambinintsoa Fortunat
BEN SASSI Rached

Table des matières

Introduction

Ce projet a pour but de lancer un marché en pleine croissance avec la startup Bonconseil mettant en œuvre un site se basant sur les recommandations des voyageurs. Nous devions nous occuper de la partie modélisation en respectant le cahier des charges fourni par Bonconseil et en mettant en relation plusieurs acteurs tels que la gérante, l'utilisateur, le voyageur, le propriétaire d'établissement, les hôtels, les restaurants ainsi que les magasins et les partenaires de Bonconseil. De plus, nous avions eu pour mission de créer la base de données correspondant à la modélisation. D'autre part, nous étions également chargés de développer les commandes SQL des fonctionnalités demandées par le cahier des charges.

I/ Modélisation



CHARLES EMILE Ambinintsoa Fortunat

BEN SASSI Rached

INFO 2

II/ Tables

1/ Liste des tables

User (**pseudoUser**, nomUser, prenomUser, paysUser, ageUser, **idBonconseil#**)
NoteCommentUser (**idNoteCommentUser**, note, **pseudoUser#**, **idCommentVoy#**)
Terminal (**idTerminal**, typeTerminal, IPTerminal, coordGPSTerminal, **pseudoUser#**)
Bonconseil (**idBonconseil**, adresseBonconseil)
MasterChef (**userMC**, nivCulinaire, **pseudoUser#**)
DreamLand (**userDL**, lieuFavoris, **pseudoUser#**)

Voyageur (**idVoyageur**, statut, **pseudoUser#**)
AvisVoy (**idAvisVoy**, nbEtoileAvis, evalPrixAvis, dateAvis, **idVoyageur#**, **idEtab#**)
CommentVoy (**idCommentVoy**, commentaireVoy, **idAvisVoy#**, **idVoyageur#**, **pseudoUser#**)

Proprietaire (**idVoyageur#**, **pseudoUser#**, inscriptionMail)
ReponseProp (**idRepProp**, commentRepProp, **idCommentVoy#**, **idVoyageur#**, **pseudoUser#**)
Signaler (**idSignaler**, signal, **idCommentVoy#**, **idVoyageur#**, **pseudoUser#**, **idGerante#**)
Promotion (**idPromo**, infoPromo, **idVoyageur#**, **pseudoUser#**, **idEtab#**)

Etablissement (**idEtab**, nomEtab, adresseEtab, villeEtab, typeEtab, coordGPSEtab, tailleEtab, prioriteConsult, **idVoyageur#**, **pseudoUser#**)
Hotel (**idEtab#**, etoileofficielleleHotel, nbChambreTotal, nbChambreDispo, activites)
Magasin (**idEtab#**, categorie)
Restaurant (**idEtab#**, typeCuisine, etoileofficielleleRest)
Prix(**idPrix**, description, prix, **idEtab#**)

Gerante(**idGerante**, requetesGerante)

Notation :

- Le rouge correspond aux clés primaires
- Le vert correspond aux clés étrangères
- Le orange correspond aux clés primaires et étrangères

2/ Commandes SQL

```
CREATE TABLE User (
        pseudoUser varchar(20), nomUser varchar(30), prenomUser varchar(30), paysUser varChar(30),
        ageUser integer check (ageUser > 0),
        PRIMARY KEY (pseudoUser),
        INDEX (pseudoUser)
) ENGINE=INNODB ;

CREATE TABLE MasterChef (
        userMC INT NOT NULL AUTO_INCREMENT, pseudoUser varchar(20), nivCulinaire varchar(10),
        PRIMARY KEY (userMC),
        INDEX (userMC),
        FOREIGN KEY (pseudoUser) REFERENCES User(pseudoUser) ON UPDATE CASCADE ON
        DELETE CASCADE
) ENGINE=INNODB ;
```

```
CREATE TABLE DreamLand (
        userDL INT NOT NULL AUTO_INCREMENT,pseudoUser varchar(20), lieuFavoris varchar(50),
        PRIMARY KEY (userDL),
        INDEX (userDL),
        FOREIGN KEY (pseudoUser) REFERENCES User(pseudoUser) ON UPDATE CASCADE ON
        DELETE CASCADE
) ENGINE=INNODB ;

CREATE TABLE Bonconseil (
        idBonconseil varchar(6),pseudoUser varchar(20), adresseBonconseil varchar(100),
        PRIMARY KEY (idBonconseil),
        INDEX (idBonconseil),
        FOREIGN KEY (pseudoUser) REFERENCES User(pseudoUser) ON UPDATE CASCADE ON
        DELETE CASCADE
) ENGINE=INNODB ;

CREATE TABLE Voyageur (
        idVoyageur INT NOT NULL AUTO_INCREMENT, pseudoUser varchar(10), statut varchar(10),
        PRIMARY KEY (idVoyageur),
        INDEX(idVoyageur),
        FOREIGN KEY (pseudoUser) REFERENCES User(pseudoUser) ON UPDATE CASCADE ON
        DELETE CASCADE
) ENGINE=INNODB ;

CREATE TABLE Etablissement (
        idEtab INT NOT NULL AUTO_INCREMENT, pseudoUser varChar(10), idVoyageur int not null,
        nomEtab varchar(30), adresseEtab varchar(50), villeEtab varchar(50), typeEtab varchar(20),
        latiGPSEtab float(50), longiGPSEtab float(50), tailleEtab integer, prioriteConsult integer check
        (prioriteConsult>=0 && prioriteConsult<=1),
        PRIMARY KEY (idEtab),
        INDEX (idEtab),
        FOREIGN KEY (idVoyageur) REFERENCES Voyageur(idVoyageur) ON UPDATE CASCADE ON
        DELETE CASCADE,
        FOREIGN KEY (pseudoUser) REFERENCES User(pseudoUser) ON UPDATE CASCADE ON
        DELETE CASCADE
) ENGINE=INNODB ;

CREATE TABLE Prix (
        idPrix INT NOT NULL AUTO_INCREMENT, idEtab int not null, description varchar(100), prix
        integer check (prix >= 0),
        PRIMARY KEY (idPrix),
        FOREIGN KEY (idEtab) REFERENCES Etablissement (idEtab) ON UPDATE CASCADE ON
        DELETE CASCADE
) ENGINE=INNODB ;

CREATE TABLE AvisVoy (
        idAvisVoy INT NOT NULL AUTO_INCREMENT, idVoyageur int not null, idEtab int not null,
        nbEtoileAvis integer check (nbEtoileAvis >= 0 && nbEtoileAvis <= 5), evalPrixAvis integer check
        (evalPrixAvis >= 0), dateAvis date,
        PRIMARY KEY (idAvisVoy),
        INDEX (idAvisVoy),
        FOREIGN KEY (idVoyageur) REFERENCES Voyageur(idVoyageur) ON UPDATE CASCADE ON
        DELETE CASCADE,
        FOREIGN KEY (idEtab) REFERENCES Etablissement(idEtab) ON UPDATE CASCADE ON
        DELETE CASCADE
```

```
) ENGINE=INNODB ;

CREATE TABLE CommentVoy (
        idCommentVoy INT NOT NULL AUTO_INCREMENT, pseudoUser varchar(10), idVoyageur
        int not null, idAvisVoy int not null, commentaireVoy varchar(300),
        PRIMARY KEY (idCommentVoy),
        INDEX (idCommentVoy),
        FOREIGN KEY (idAvisVoy) REFERENCES AvisVoy(idAvisVoy) ON UPDATE CASCADE ON
        DELETE CASCADE,
        FOREIGN KEY (idVoyageur) REFERENCES Voyageur(idVoyageur) ON UPDATE CASCADE ON
        DELETE CASCADE,
        FOREIGN KEY (pseudoUser) REFERENCES User(pseudoUser) ON UPDATE CASCADE ON
        DELETE CASCADE
) ENGINE=INNODB ;

CREATE TABLE NoteCommentUser (
        idNoteCommentUser INT NOT NULL AUTO_INCREMENT, pseudoUser varchar(10),
        idCommentVoy int not null, note integer    check (note >= 0 && note <= 5),
        PRIMARY KEY (idNoteCommentUser),
        INDEX (idNoteCommentUser),
        FOREIGN KEY (pseudoUser) REFERENCES User(pseudoUser) ON UPDATE CASCADE ON
        DELETE CASCADE,
        FOREIGN KEY (idCommentVoy) REFERENCES CommentVoy(idCommentVoy) ON UPDATE
        CASCADE ON DELETE CASCADE
) ENGINE=INNODB ;

CREATE TABLE Terminal (
        idTerminal INT NOT NULL AUTO_INCREMENT, pseudoUser varchar(10), typeTerminal text not
        null, IPTerminal varchar(15),
        latiGPSTerminal float(50), longiGPSTerminal float(50),
        PRIMARY KEY (idTerminal),
        INDEX (idTerminal),
        FOREIGN KEY (pseudoUser) REFERENCES User(pseudoUser) ON UPDATE CASCADE ON
DELETE CASCADE
) ENGINE=INNODB ;

CREATE TABLE Proprietaire (
        pseudoUser varchar(10), idVoyageur int not null, inscriptionMail enum('FALSE','TRUE'),
        PRIMARY KEY (pseudoUser, idVoyageur),
        INDEX (pseudoUser),
        INDEX (idVoyageur),
        FOREIGN KEY (idVoyageur) REFERENCES Voyageur(idVoyageur) ON UPDATE CASCADE ON
        DELETE CASCADE,
        FOREIGN KEY (pseudoUser) REFERENCES User(pseudoUser) ON UPDATE CASCADE ON
        DELETE CASCADE
) ENGINE=INNODB ;

CREATE TABLE ReponseProp (
        idRepProp INT NOT NULL AUTO_INCREMENT, pseudoUser varchar(10), idVoyageur int not
        null,    idCommentVoy int not null, commentRepProp text not null,
        PRIMARY KEY (idRepProp),
        INDEX (idRepProp),
        FOREIGN KEY (idCommentVoy) REFERENCES CommentVoy(idCommentVoy) ON UPDATE
        CASCADE ON DELETE CASCADE,
        FOREIGN KEY (idVoyageur) REFERENCES Voyageur(idVoyageur) ON UPDATE CASCADE ON
```

```
        DELETE CASCADE,
        FOREIGN KEY (pseudoUser) REFERENCES User(pseudoUser) ON UPDATE CASCADE ON
        DELETE CASCADE
) ENGINE=INNODB ;

CREATE TABLE Gerante (
        idGerante varchar(6), requetesGerante text not null,
        PRIMARY KEY (idGerante),
        INDEX (idGerante)
) ENGINE=INNODB ;

CREATE TABLE Signaler (
        idSignaler INT NOT NULL AUTO_INCREMENT, pseudoUser varchar(10), idVoyageur int not null,
        idGerante varchar(6), idCommentVoy int not null, signale varchar(500),
        PRIMARY KEY (idSignaler),
        INDEX (idSignaler),
        FOREIGN KEY (idCommentVoy) REFERENCES CommentVoy(idCommentVoy) ON UPDATE
        CASCADE ON DELETE CASCADE,
        FOREIGN KEY (idVoyageur) REFERENCES Voyageur(idVoyageur) ON UPDATE CASCADE ON
        DELETE CASCADE,
        FOREIGN KEY (pseudoUser) REFERENCES User(pseudoUser) ON UPDATE CASCADE ON
        DELETE CASCADE,
        FOREIGN KEY (idGerante) REFERENCES Gerante(idGerante) ON UPDATE CASCADE ON
        DELETE CASCADE
) ENGINE=INNODB ;

CREATE TABLE Promotion (
        idPromo INT NOT NULL AUTO_INCREMENT, pseudoUser varChar(10), idVoyageur int not null,
        idEtab int not null, infoPromo text not null,
        PRIMARY KEY (idPromo),
        INDEX (idPromo),
        FOREIGN KEY (idVoyageur) REFERENCES Voyageur(idVoyageur) ON UPDATE CASCADE ON
        DELETE CASCADE,
        FOREIGN KEY (pseudoUser) REFERENCES User(pseudoUser) ON UPDATE CASCADE ON
        DELETE CASCADE,
        FOREIGN KEY (idEtab) REFERENCES Etablissement(idEtab) ON UPDATE CASCADE ON
        DELETE CASCADE
) ENGINE=INNODB ;

CREATE TABLE Hotel (
        idEtab int not null, etoileofficielleleHotel integer check (etoileofficielleleHotel >= 0 &&
        etoileofficielleleHotel <= 5), nbChambreTotal integer check (nbChambreTotal > 0),
        nbChambreDispo integer check (nbChambreDispo <= nbChambreTotal),
        PRIMARY KEY (idEtab),
        INDEX (idEtab),
        FOREIGN KEY (idEtab) REFERENCES Etablissement(idEtab) ON UPDATE CASCADE ON
        DELETE CASCADE
) ENGINE=INNODB ;

CREATE TABLE Magasin (
        idEtab int not null, categorie text not null,
        PRIMARY KEY (idEtab),
        INDEX (idEtab),
        FOREIGN KEY (idEtab) REFERENCES Etablissement(idEtab) ON UPDATE CASCADE ON
        DELETE CASCADE
```

```
) ENGINE=INNODB ;

CREATE TABLE Restaurant (
        idEtab int not null, typeCuisine text not null,
        etoileofficielleleRest integer check (etoileofficielleleRest >= 0 && etoileofficielleleRest <= 5),
        PRIMARY KEY (idEtab),
        INDEX (idEtab),
        FOREIGN KEY (idEtab) REFERENCES Etablissement(idEtab) ON UPDATE CASCADE ON
        DELETE CASCADE
) ENGINE=INNODB ;
```

III/ Description des choix effectués

1/ Tables

Bonconseil :
La table *Bonconseil* est la table de la startup contenant son ID au niveau national et son adresse.

MasterChef :
La table *MasterChef* est un partenaire de Bonconseil. Elle permet d'indiquer le niveau culinaire des utilisateurs de Bonconseil s'ils sont membres de *MasterChef*. Un utilisateur ayant un bon niveau culinaire sera plus crédible lors des avis postés sur un restaurant si c'est un Voyageur de Bonconseil et il sera plus crédible lors d'une notation d'un commentaire d'un restaurant si c'est un simple utilisateur de Bonconseil. Elle contient l'ID MasterChef, le niveau culinaire de l'utilisateur ainsi que le pseudo de l'utilisateur permettant d'avoir un lien avec la startup Bonconseil.

DreamLand :
La table *DreamLand* est un partenaire de Bonconseil, elle permet d'indiquer le lieu favori des utilisateurs de Bonconseil s'ils sont membres de *DreamLand*. Un utilisateur étant un Voyageur de Bonconseil et postant un avis sur un établissement dans la ville où son lieu favori se situe, aura plus de chance de connaître les bons plans de la ville. De même pour la notation d'un commentaire portant sur des établissements se situant dans la ville de son lieu favori. Elle contient l'ID DreamLand, le lieu favori de l'utilisateur et le pseudo de l'utilisateur permettant d'avoir un lien avec la startup Bonconseil.

User :
La table *User* est une table regroupant tous les utilisateurs de Bonconseil. Elle contient le pseudo, le nom, le prénom, le pays, l'age de l'utilisateur et l'ID de Bonconseil permettant de savoir que l'utilisateur est bien inscrit dans la startup Bonconseil.

Terminal :
La table *Terminal* est une table regroupant tous les terminaux utilisés par les utilisateurs. Elle comprend notamment l'ID du terminal, le type du terminal (mobile, tablette, fixe), l'adresse IP et les coordonnées GPS du terminal (latitude, longitude) permettant la localisation de l'utilisateur et le pseudo de l'utilisateur permettant de savoir à quel utilisateur appartient le terminal.

Voyageur :
La table *Voyageur* est une table regroupant tous les voyageurs de Bonconseil. Elle contient l'ID du voyageur et l'attribut « statut » qui informera les utilisateurs du niveau que le voyageur a atteint et le pseudo de l'utilisateur qui permet ainsi de mettre en relation le voyageur et l'utilisateur. Plus un voyageur donne d'avis, plus son statut évolue. Un voyageur ayant un bon statut aura donc une meilleure crédibilité lors du dépôt d'un avis sur un établissement.

AvisVoy :
La table *AvisVoy* est une table regroupant tous les avis des Voyageurs. Elle est composée de l'ID de l'avis, du nombre d'étoiles de l'avis, de l'évaluation des prix estimés par le voyageur sur un établissement, la date de l'Avis, l'ID du voyageur postant l'avis ainsi que l'ID de l'établissement sur lequel le voyageur poste l'avis.

CommentVoy :
La table *CommentVoy* est une table regroupant les commentaires associés aux avis des voyageurs. Les voyageurs ont le choix d'en mettre un ou non. Elle est composée de l'ID du commentaire, du commentaire en lui même, l'ID de l'avis auquel il correspond, le pseudo et l'ID du voyageur qui a posté l'avis. Elle permet de décrire explicitement l'avis en lui même.

NoteCommentUser :
La table *NoteCommentUser* est une table regroupant les notes des utilisateurs de Bonconseil des commentaires des voyageurs de leurs avis. La note est comprise entre 0 et 5. Elle permet de notifier les commentaires utiles ou non. Elle contient l'ID de la note, la note du commentaire, l'ID du commentaire noté et le pseudo de l'utilisateur qui note le commentaire.

Proprietaire :
La table *Proprietaire* est une table regroupant tous les propriétaires d'établissement de Bonconseil. Elle comprend l'ID du voyageur mais aussi l'ID de l'utilisateur car un propriétaire est un voyageur et un utilisateur. Elle contient aussi un attribut « inscriptionMail » lui permettant de préciser s'il veut recevoir ou non les commentaires sur ses établissements.

ReponseProp :
La table *ReponseProp* est une table regroupant les réponses des propriétaires aux avis que les voyageurs leur ont adressés sur leurs établissements. Elle contient l'ID de la réponse ainsi que la réponse, l'ID du commentaire de l'avis sur lequel le propriétaire répond et comme le propriétaire est un voyageur et un utilisateur alors il y a aussi le pseudo et l'ID du voyageur (ici le propriétaire qui répond à l'avis). Elle permet au propriétaire de répondre aux avis leur étant adressé.

Signaler :
La table *Signaler* est une table regroupant les signalements des propriétaires sur des commentaires jugés insultant portant sur leurs établissements. Elle est composée de l'ID du signalement ainsi que le signalement, l'ID du commentaire à signaler, l'ID de la gérante ainsi que le pseudo et l'ID du voyageur (ici le propriétaire qui signale le commentaire insultant). Elle permet au propriétaire de le signaler directement à la gérante.

Gerante :
La table *Gerante* est une table regroupant les requêtes que la gérante peut utiliser sur son logiciel. Elle est composée de l'ID de la gérante et des requêtes qu'elle peut effectuer. La gérante peut changer le statut d'un voyageur, connaître les voyageurs avec les commentaires jugés les plus utiles par les utilisateurs, radier un voyageur ayant plus de 30% de commentaires notés moins bien que 2/5 par les utilisateurs, connaître les voyageurs ayant posté des avis en utilisant plusieurs comptes utilisateurs avec le même IP permettant ainsi de limiter les fraudes, décerner des prix aux meilleurs établissements dans chaque ville, trouver les utilisateurs n'ayant donné des avis que sur des hôtels et restaurants 5 étoiles, connaître la plus grande différence entre le nombre d'étoiles officielles et le nombre d'étoiles moyen décerné par les utilisateurs.

Promotion :
La table *Promotion* est une table regroupant les promos en cours sur les établissements. Elle est composée de l'ID de la promotion ainsi que la promotion, l'ID de l'établissement ayant la promotion, le pseudo et l'ID du voyageur (ici le propriétaire) qui met la promotion. Elle permet au propriétaire d'informer les utilisateurs des promotions en cours sur leurs établissements.

Etablissement :
La table *Etablissement* est une table regroupant tous les établissements des propriétaires de Bonconseil. Elle contient notamment l'ID de l'établissement, le nom, le type, l'adresse, la ville, les coordonnées GPS (latitude, longitude), la taille et l'attribut « prioriteConsult » qui est une option payante qui permet au propriétaire de faire apparaître ses établissements en tête de liste dans les consultations des utilisateurs, le pseudo et l'ID du voyageur (ici le propriétaire).

Prix :
La table *Prix* est une table regroupant les prix de chaque prestation ou article des établissements. Elle contient l'ID du prix, la description de la prestation ou de l'article ainsi que le prix et l'ID de l'établissement dans lequel s'appliquent les prix. Elle permet de gérer tous les prix de chaque établissement.

Hotel :
La table *Hotel* est une table regroupant les hôtels. Elle contient l'ID de l'établissement, le nombre d'étoiles officielles, le nombre de chambres total, le nombre de chambres disponibles et l'activité.

Magasin :
La table *Magasin* est une table regroupant les magasins. Elle contient l'ID de l'établissement, la catégorie du magasin (bricolage, prêt-à-porter …).

Restaurant :
La table *Restaurant* est une table regroupant les restaurants. Elle contient l'ID de l'établissement, le type de cuisine (indienne, italienne, française …) et le nombre d'étoiles officielles.

2/ Les cardinalités

User – Bonconseil :
A zéro ou plusieurs *utilisateur* correspond une et une seule startup *Bonconseil*.
Il peut y avoir zéro ou plusieurs *utilisateur* dans la startup *Bonconseil*.

User – MasterChef :
A un et un seul *utilisateur* correspond zéro ou un partenariat avec *MasterChef*.
Un *utilisateur* de Bonconseil peut aussi être ou non *utilisateur* de *MasterChef*.

User – DreamLand :
A un et un seul *utilisateur* correspond zéro ou un partenariat avec *DreamLand*.
Un *utilisateur* de Bonconseil peut aussi être ou non *utilisateur* de *DreamLand*.

User – Terminal :
A un et un seul *utilisateur* correspond un ou plusieurs *Terminal(s)*.
Un *utilisateur* de Bonconseil peut avoir un ou plusieurs *Terminal*.

User – Voyageur :
A un et un seul *utilisateur* correspond zéro ou plusieurs *voyageur(s)*.
Un *utilisateur* de Bonconseil peut aussi être ou non un *voyageur* de Bonconseil.

User – NoteCommentUser :
A un et un *utilisateur* correspond zéro ou plusieurs *note(s) de commentaire*.
Un *utilisateur* peut noter ou non plusieurs commentaires.

Voyageur – AvisVoy :
A un et un *voyageur* correspond zéro ou plusieurs *avis voyageur.*
Un *voyageur* peut donner ou non plusieurs *avis voyageur.*

AvisVoy – Etablissement :
A un et un *établissement* correspond zéro ou plusieurs *avis voyageur.*
Un *établissement* peut avoir ou non plusieurs *avis voyageur.*

AvisVoy – CommentVoy :
A un et un *avis voyageur* correspond zéro ou un *commentaire du voyageur.*
Un *avis voyageur* peut être accompagné ou non un *commentaire de ce voyageur.*

CommentVoy – NoteCommentUser :
A un et un *commentaire de voyageur* correspond zéro ou une *note d'un utilisateur.*
Un *commentaire de voyageur* peut être *noté* ou non par un *utilisateur.*

CommentVoy – Signaler :
A un et un *commentaire de voyageur* correspond zéro ou un *signalement d'un propriétaire*.
Un *commentaire de voyageur* peut avoir zéro ou un *signalement d'un propriétaire*.

CommentVoy – ReponseProp :
A un et un *commentaire de voyageur* correspond zéro ou plusieurs *réponse(s) du propriétaire*.
Un *propriétaire* peut répondre ou non à un *commentaire d'un voyageur.*

ReponseProp – Propriétaire :
A un et un *propriétaire* correspond zéro ou plusieurs *réponse(s) du propriétaire* aux commentaires des voyageurs.
Un *propriétaire* peut avoir ou non plusieurs *réponses* aux commentaires des voyageurs.

Propriétaire – Signaler :
A un et un *propriétaire* correspond zéro ou plusieurs *signalement(s)*.
Un *propriétaire* peut signaler ou plusieurs fois ou non.

Propriétaire – Etablissement :
A un et un *propriétaire* correspond un ou plusieurs *établissements*.
Un *propriétaire* peut avoir un ou plusieurs *établissements*.

Propriétaire – Promotions :
A un et un *propriétaire* correspond zéro ou plusieurs *promotions*.
Un *propriétaire* peut donner plusieurs promotions ou non.

Propriétaire hérite de Voyageur et de User :
Le *propriétaire* hérite des tables Voyageur et User car il est à la fois *voyageur* et *utilisateur.*

Signaler – Gerante :
A une et une *gérante* correspond zéro ou plusieurs *signalement(s)*.
Une *gérante* peut avoir plusieurs *signalements* ou non.

Promotions – Etablissement :
A un et un *établissement* correspond zéro ou plusieurs *promotions*.
Un *établissement* peut avoir plusieurs *promotions* ou non.

Etablissement – Prix :
A un et un *établissement* correspond un ou plusieurs *prix*.
Un *établissement* peut avoir un ou plusieurs *prestations* ou *articles* ayant un ou plusieurs *prix*.

Hotel hérite d'Etablissement :
L'*hôtel* hérite d'*établissement* car il a toutes les propriétés d'établissement en plus des propriétés de l'*hôtel*.

Magasin hérite d'Etablissement :
Le *magasin* hérite d'*établissement* car il a toutes les propriétés d'établissement en plus des propriétés du *magasin*.

Restaurant hérite d'Etablissement :
Le *restaurant* hérite d'*établissement* car il a toutes les propriétés d'établissement en plus des propriétés du restaurant.

3/ Changements effectués sur la modélisation

a/ Modification des variables et des tables

Terminal :
Dans la table *Terminal*, nous avons remplacé *coordGPSTerminal* par *latiGPSTerminal* et *longiGPSTerminal* car ces attributs sont beaucoup plus faciles à manipuler par la suite.
Par exemple pour avoir la distance entre deux coordonnées GPS, il est plus aisé d'avoir des coordonnées distinctes : sqrt((latitude2 – latitude1)² + (longitude2 - longitude1)²).

Etablissement :
Dans la table *Etablissement*, nous avons remplacé *coodGPSEtab* par *latiGPSEtab* et *longiGPSEtab* pour les mêmes raisons que pour les changements de la table *Terminal*.
Nous avons aussi ajouté deux attributs : *villeEtab* et *typeEtab*.
L'attribut *villeEtab* nous permet d'avoir directement la ville de l'établissement. C'est beaucoup plus pratique pour les requêtes ayant pour critère la recherche par *ville*.
L'attribut *typeEtab* nous permet de préciser le type de l'*établissement* (*hôtel, magasin ou restaurant*).

Prix :
Nous avons ajouté cette table pour que les *établissements* puissent afficher le *prix* des *articles* ou des *prestations* qu'ils vendent. On ne pouvait pas le mettre en tant qu'attribut car il n'y a pas qu'un seul *prix* par *établissement* mais plusieurs *prix* pour plusieurs *articles* ou *prestations* différentes.

Signaler :
Nous avons remplacé *signal* par *signale* car *signal* est une commande de SQL, ce qui nous a provoqué des erreurs de syntaxe.

b/ Modification des cardinalités

User – MasterChef :
Nous avons remplacé la cardinalité *User* « 0-n » - « 0-1 » *MasterChef* par *User* « 1-1 » - « 0-1 » *MasterChef* car à un et un *utilisateur* de *Bonconseil* correspond ou non à un *utilisateur* de *MasterChef*.

User – DreamLand :
Nous avons remplacé la cardinalité *User* « 0-n » - « 0-1 » *DreamLand* par *User* « 1-1 » - « 0-1 » *DreamLand* car à un et un *utilisateur* de *Bonconseil* correspond ou non à un *utilisateur* de *DreamLand*.

AvisVoy – ReponseProp :
Nous avons supprimé la cardinalité *AvisVoy* « 1-1 » - « 0-n » *ReponseProp* car un *propriétaire* ne peut répondre qu'à un *commentaire d'un avis de voyageur* et non directement à l'*avis d'un voyageur.*

Propriétaire - CommentVoy :
Nous avons supprimé la cardinalité *Propriétaire* « 1-1 » - « 0-n » *CommentVoy* car un *propriétaire* n'a pas de lien direct avec le *commentaire* d'un *voyageur.*

Etablissement – Prix :
Nous avons ajouté la cardinalité *Etablissement* « 1-1 » - « 1-n » *Prix* car à un et un *établissement* correspond un ou plusieurs *prix*.

IV/ Requêtes

1/ Question : Un utilisateur peut se désinscrire.

```
DELETE from User where pseudoUser="Ailwenn";
```

2/ Question : Un utilisateur peut chercher des établissements en fonction de plusieurs critères (type, ville, prix, etc.). L'affichage de ces établissements peut être par exemple en fonction du prix (ordre décroissant) ou des avis des voyageurs.

```
SELECT DISTINCT nomEtab,typeEtab,prix FROM Etablissement NATURAL JOIN Prix NATURAL JOIN
Hotel WHERE typeEtab="Hotel" && villeEtab="Paris" && prix >=10 && prix <= 800 &&
etoileofficielleleHotel >= 2 ORDER BY Prix DESC;
```

Affiche:

```
+---------------+----------+------+
| nomEtab       | typeEtab | prix |
+---------------+----------+------+
| Hotel Carlton | Hotel    |  725 |
| Hotel Carlton | Hotel    |  569 |
| MangerDormir  | Hotel    |  160 |
| MangerDormir  | Hotel    |  130 |
| MangerDormir  | Hotel    |  120 |
| MangerDormir  | Hotel    |   80 |
+---------------+----------+------+
```

3/ Question : Un utilisateur peut se faire afficher tous les restaurants d'une ville trie par nombre moyenne d'étoiles. Les restaurants qui ont payé apparaissent d'abord.

Cette requête nous donne les Restaurants notés ordonnés par leur moyenne d'étoiles, mais les restaurants qui ont payé, apparaissent en priorité en haut du classement.

```
SELECT a.idEtab, nomEtab, AVG(nbEtoileAvis) AS Moyenne,prioriteConsult FROM AvisVoy a,
Etablissement e WHERE a.idEtab=e.idEtab && typeEtab="Restaurant" GROUP BY a.idEtab,
nomEtab ORDER BY prioriteConsult DESC, Moyenne;
```

Affiche:

```
+---------+-------------------------+------------+--------------------+
| idEtab | nomEtab                  | Moyenne | prioriteConsult |
+---------+-------------------------+------------+--------------------+
|     3   | Lafourchette            |  3.6667  |           1        |
|    13   | Secret Square           |  5.0000  |           1        |
|    12   | La Petite Bretagne      |  3.0000  |           0        |
|     2   | Chez Raymond            |  3.5000  |           0        |
|    14   | Le Rosmarino            |  4.0000  |           0        |
|     1   | L'orange Bleue          |  4.2500  |           0        |
+---------+-------------------------+------------+--------------------+
```

4/ Question : Un utilisateur peut chercher tous les magasins de bricolage avec un nombre d'étoiles supérieur à 3 en moyenne et qui se trouvent à moins de 5km de sa position.

La première requête permet d'avoir tous les établissements qui ont au moins un Avis et d'avoir la moyenne de leur Avis.

```
Create view Avis as select Etablissement.idEtab,nomEtab,avg(nbEtoileAvis) as moyenne,typeEtab from AvisVoy,Etablissement where AvisVoy.idEtab=Etablissement.idEtab group by nomEtab;
```

La deuxième requête permet d'obtenir tous les Magasins qui ont pour catégorie le bricolage et ayant une moyenne supérieure à 3.

```
Create view brico3 as select * from Magasin natural join Avis where categorie="bricolage" && moyenne>3;
```

La Troisième requête sélectionne un utilisateur.

```
CREATE VIEW TermUser AS SELECT * FROM Terminal WHERE pseudoUser="Ailwenn" && typeTerminal="mobile";
```

La Quatrième requête sélectionne la (les) position(s) de(s) Magasin(s) ayant une moyenne supérieure à 3.

```
Create view GPSEtab as select brico3.idEtab,brico3.nomEtab,longiGPSEtab,latiGPSEtab from brico3 natural join Etablissement where Etablissement.idEtab=brico3.idEtab;
```

Cette Requête permet de regrouper toutes les informations obtenues.

```
CREATE VIEW GPS AS SELECT * FROM TermUser NATURAL JOIN GPSEtab ;
```

Cette requête nous donne le nom des (ou du) Magasin dans un rayon de 5 km ainsi que la distance pour y parvenir en Km.

```
SELECT nomEtab,((sqrt((latiGPSEtab-latiGPSTerminal)*(latiGPSEtab- latiGPSTerminal)+(longiGPSEtab-longiGPSTerminal)*(longiGPSEtab-longiGPSTerminal)))*100) as "Distance en Km" FROM GPS WHERE ((sqrt((latiGPSEtab-latiGPSTerminal)*(latiGPSEtab-latiGPSTerminal) +(longiGPSEtab-longiGPSTerminal)*(longiGPSEtab- longiGPSTerminal)))*100)<=5;
```

Affiche :

```
+------------+----------------------+
| nomEtab    | Distance en Km       |
+------------+----------------------+
| BricoBazar | 0.041136594412894543 |
+------------+----------------------+
```

5/ Question : Un utilisateur peut chercher tous les restaurants indiens à Paris dont tous les avis des voyageurs (qui ne sont jamais notés moins bien que 3) sont supérieurs à 4.

Réunion des utilisateurs ayant des notes supérieures à 3.

create view MinNote as Select pseudoUser,note,idCommentVoy from NoteCommentUser where note >= 3;

Sélection des établissements

select distinct nomEtab from MinNote,Restaurant ,CommentVoy ,AvisVoy , Etablissement where typeCuisine="Indien" && MinNote.idCommentVoy=CommentVoy.idCommentVoy && CommentVoy.idAvisVoy=AvisVoy.idAvisVoy && AvisVoy.idEtab=Etablissement.idEtab && nbEtoileAvis>4;

Affiche :

```
+---------------------+
| nomEtab             |
+---------------------+
| Residence Calaluna  |
| Secret Square       |
+---------------------+
```

6/ Question : Un utilisateur peut chercher l'ensemble des hôtels pour lesquels tous les avis sont supérieurs à 3 et dont les étoiles officielles sont supérieures à 2.

Réunion des Hôtels répondant aux critères.

CREATE VIEW QualiteHotel AS SELECT idEtab FROM Hotel WHERE idEtab NOT IN(SELECT a.idEtab FROM AvisVoy a NATURAL JOIN Hotel WHERE nbEtoileAvis<3|| etoileofficielleleHotel <2);

SELECT idEtab,nomEtab FROM QualiteHotel NATURAL JOIN Etablissement;

Affiche:

```
+--------+--------------------+
| idEtab | nomEtab            |
+--------+--------------------+
| 6    | Residence Calaluna |
| 15    | Hotel Carlton      |
| 17    | Le Girona          |
+--------+--------------------+
```

7/ Question : Un voyageur peut donner un avis sur un établissement.

insert into AvisVoy(idVoyageur,idEtab,nbEtoileAvis,evalPrixAvis,dateAvis) values (5,2,4,5,"2014-12-24");

8/ Question : Un voyageur peut chercher tous les établissements pour lesquels il a donné un avis en 2012.

SELECT a.idEtab, nomEtab FROM Etablissement e, AvisVoy a, Voyageur v WHERE a.idEtab=e.idEtab && v.pseudoUser="Fowlloch" && dateAvis LIKE "2012%";

Affiche :

```
+--------+--------------+
| idEtab | nomEtab      |
+--------+--------------+
| 004    | MangerDormir |
+--------+--------------+
```

9/ Question : Un voyageur peut donner un avis sur n'importe quel établissement mais une fois au maximum par an et par établissement. Donnez une requête qui donne tous les voyageurs qui ne respectent pas cette règle.

Donne une table AnneeAvis en remplaçant les dates par des années.

CREATE VIEW AnneeAvis AS SELECT idAvisVoy,idVoyageur,idEtab,YEAR(dateAvis) AS Annee FROM AvisVoy;

Donne les utilisateurs ayant 2 tuples ou plus dans la table AnneeAnnee, ce qui correspond au non respect de la règle.

SELECT AnneeAvis.idVoyageur,pseudoUser FROM AnneeAvis,Voyageur where AnneeAvis.idVoyageur=Voyageur.idVoyageur GROUP BY idVoyageur,Annee,idEtab HAVING COUNT(*)>=2;

Affiche :

```
+------------+------------+
| idVoyageur | pseudoUser |
+------------+------------+
|          3 | Flydower   |
+------------+------------+
```

10/ Question : Un voyageur obtient des bons de réduction, s'il donne plus de 10 avis par mois pendant un an. Donnez une requête qui donne ceux qui ont le droit aux bons.

Avis total et nombre d'avis par voyageur par mois par rapport à l'année.

CREATE VIEW MoisAnneeAvis AS SELECT idAvisVoy, idVoyageur, idEtab, MONTH(dateAvis) AS Mois, YEAR(dateAvis) AS Annee FROM AvisVoy;

CREATE VIEW AvisTotalParMois AS SELECT idVoyageur,Mois,Annee,COUNT(Mois) AS cpt FROM MoisAnneeAvis GROUP BY Mois,idVoyageur,Annee;

Ceux qui ont un nombre d'avis supérieur ou égal à 10.

CREATE VIEW 10AvisTotalParMois AS SELECT idVoyageur AS VoyPot,Mois AS MoisPot,Annee AS AnneePot ,cpt AS cptPot FROM AvisTotalParMois WHERE cpt>=10;

On groupe par utilisateur et par année.

CREATE VIEW DroitPotentiel2 AS SELECT VoyPot,COUNT(MoisPot) AS nbrMois FROM 10AvisTotalParMois GROUP BY VoyPot,AnneePot;

Ceux qui ont le droit aux bons.

SELECT VoyPot FROM DroitPotentiel2 WHERE nbrMois=12;

11/ Question : On peut connaitre la note moyenne ainsi que le prix moyen (tel qu'estimé par les voyageurs) d'un établissement.

Select nomEtab,AVG(nbEtoileAvis) as "Note Moyenne",AVG(evalPrixAvis) as "Moyenne des Prix" FROM AvisVoy a, Etablissement e WHERE a.idEtab=e.idEtab && nomEtab="BricoBazar";

Affiche :

```
+------------+--------------+------------------+
| nomEtab    | Note Moyenne | Moyenne des Prix |
+------------+--------------+------------------+
| BricoBazar |       4.0000 |           3.6667 |
+------------+--------------+------------------+
```

12/ Question : Un propriétaire peut consulter tous les avis sur tous ses établissements.

SELECT nomEtab,idAvisVoy FROM AvisVoy a, Etablissement e, Proprietaire p WHERE a.idEtab=e.idEtab && p.pseudoUser=e.pseudoUser && p.pseudoUser="Runvald";

Affiche :

```
+---------------+-----------+
| nomEtab       | idAvisVoy |
+---------------+-----------+
| Lafourchette  |         9 |
| Lafourchette  |        14 |
| Lafourchette  |        25 |
| HotelIbis     |         1 |
| HotelIbis     |         4 |
| HotelIbis     |        16 |
| HotelIbis     |        27 |
| LVMH          |        19 |
| LVMH          |        30 |
| Secret Square |         3 |
| Secret Square |        34 |
| Secret Square |        45 |
| Secret Square |        46 |
| HotelF1       |        37 |
| GHM           |        40 |
| Sephora       |        41 |
+---------------+-----------+
```

13/ Question : Un propriétaire peut se faire afficher tous les établissements concurrents (proches, dans la même ville, etc.). Par exemple, il peut se faire afficher tous les restaurants à Paris.

SELECT e.idEtab,e.nomEtab FROM Etablissement e,Proprietaire p WHERE e.idVoyageur=p.idVoyageur && p.idVoyageur!=0004 && typeEtab="Restaurant" && villeEtab="Paris" GROUP BY idEtab;

Affiche :

```
+--------+---------------+
| idEtab | nomEtab       |
+--------+---------------+
|      3 | Lafourchette  |
|     13 | Secret Square |
|     14 | Le Rosmarino  |
+--------+---------------+
```

14/ Question : Un propriétaire peut se faire afficher tous les meilleurs avis pour chacun de ses établissements.

On fait la jointure de la table AvisVoy avec Etablissement sur les identifiants des Etablissements pour obtenir une table regroupant les avis et les noms d'établissement.

CREATE VIEW EtabProp AS SELECT idAvisVoy,nomEtab,a.idEtab,e.idVoyageur,a.nbEtoileAvis FROM AvisVoy a, Etablissement e WHERE a.idEtab=e.idEtab;

On cherche les avis des établissements du propriétaire ayant pour identifiant 4.

SELECT nomEtab,idAvisVoy,MAX(nbEtoileAvis) FROM EtabProp WHERE idVoyageur=4 GROUP BY nomEtab;

Affiche :

```
+--------------------+-----------+--------------------+
| nomEtab            | idAvisVoy | MAX(nbEtoileAvis)  |
+--------------------+-----------+--------------------+
| Chez Raymond       | 13        |                  4 |
| L'orange Bleue     | 8         |                  5 |
| La Petite Bretagne | 33        |                  3 |
+--------------------+-----------+--------------------+
```

15/ Question : Un propriétaire peut se faire afficher ses établissements qui ont eu au moins deux fois 5 ou 4 étoiles par un utilisateur.

Regroupement des établissements ayant des notes supérieures ou égales à 4.

create view NoteSup4 as select idVoyageur,idEtab as id,nbEtoileAvis from AvisVoy where nbEtoileAvis>=4;

Pour avoir les noms des établissements on fait la jointure avec la table Etablissement, de plus on compte le nombre de notes supérieures ou égales à 4 pour chaque établissement.

CREATE VIEW VUE AS SELECT COUNT(id) AS nbAVisSup4,id,nomEtab FROM NoteSup4,Etablissement WHERE NoteSup4.id=Etablissement.idEtab GROUP BY id,nomEtab;

SELECT pseudoUser,VUE.nomEtab FROM VUE,Etablissement WHERE id=idEtab && nbAvisSup4>1 && pseudoUser="Stavefrid";

Affiche :

```
+------------+--------------------+
| pseudoUser | nomEtab            |
+------------+--------------------+
| Stavefrid  | Residence Calaluna |
| Stavefrid  | BricoBazar         |
+------------+--------------------+
```

16/ Question : Un propriétaire peut se faire afficher tous ses établissements dont l'avis donné par le même utilisateur a baissé d'au moins 2 étoiles en au maximum 2 ans

create view ToutAvis as select idVoyageur,nbEtoileAvis as note,idEtab,dateAvis from AvisVoy;

La requête suivante donne pour chaque voyageur la note maximale de ses avis,

Create view MaxAvis as select idVoyageur,max(nbEtoileAvis) as note,idEtab,dateAvis from AvisVoy group by idEtab;

select * from ToutAvis ta,MaxAvis ma where ta.idVoyageur=ma.idVoyageur && ta.note+2<=ma.note && (year(ta.dateAvis)-year(ma.dateAvis)<2 || year(ma.dateAvis)-year(ta.dateAvis)<2) group by ta.idVoyageur;

Affiche :

```
+--------------+-------+---------+-------------+--------------+------+---------+----------------+
| idVoyageur | note | idEtab | dateAvis   | idVoyageur | note | idEtab | dateAvis     |
+--------------+-------+---------+--------------+--------------+-------+---------+----------------+
|      1     |    2 |   10  | 2008-12-23 |        1 |     5 |      5 | 2003-12-23 |
|      3     |    2 |    5   | 2008-10-06 |        3 |     4 |      2 | 2008-10-29 |
|      7     |    2 |    4   | 2013-12-23 |        7 |     5 |      3 | 2008-12-23 |
+--------------+-------+---------+--------------+--------------+-------+---------+----------------+
```

17/ Question : La gérante peut changer le statut d'un voyageur.

UPDATE Voyageur SET statut="bronze" WHERE statut="standard" && pseudoUser="Runvald";

Avant :

```
+------------+------------+----------+
| idVoyageur | pseudoUser | statut   |
+------------+------------+----------+
| 0002       | Runvald    | standard |
+------------+------------+----------+
```

Après :

```
+------------+------------+----------+
| idVoyageur | pseudoUser | statut   |
+------------+------------+----------+
| 0002       | Runvald    | bronze   |
+------------+------------+----------+
```

18/ Question : La gérante peut effacer les avis non-conformes (c'est-à-dire tous les avis en trop d'un voyageur sur le même établissement la même année).

CREATE VIEW AvisAnnee AS SELECT idVoyageur,idEtab,YEAR(dateAvis) AS Annee FROM AvisVoy;

CREATE VIEW NbrAvis AS SELECT idVoyageur,idEtab,Annee,COUNT(Annee) AS cptAvis FROM AvisAnnee GROUP BY idVoyageur,idEtab,Annee;

CREATE VIEW AvisNonConforme AS SELECT * FROM NbrAvis WHERE cptAvis >= 2;

CREATE VIEW AvisSup AS SELECT * FROM AvisVoy NATURAL JOIN AvisNonConforme ORDER BY(dateAvis);

SELECT * FROM AvisSup;

Affiche les avis non-conformes :

idVoyageur	idEtab	idAvisVoy	nbEtoileAvis	evalPrixAvis	dateAvis	Annee	cptAvis
3	1	12	5	4	2008-10-23	2008	2
3	1	42	4	4	2008-11-23	2008	2

SELECT * FROM AvisSup WHERE dateAvis NOT IN(SELECT MIN(dateAvis) FROM AvisSup);

Affiche les avis en trop, les avis à supprimer :

idVoyageur	idEtab	idAvisVoy	nbEtoileAvis	evalPrixAvis	dateAvis	Annee	cptAvis
3	1	42	4	4	2008-11-23	2008	2

A partir de cette liste, la gérante peut alors supprimer les avis en trop des voyageurs :

DELETE FROM AvisVoy WHERE idAvisVoy = 42;

19/ Question : La gérante peut effacer des commentaires jugés insultants.

On regarde les signalements.

select * from Signaler

On supprime les Commentaires signalés.

delete from CommentVoy where idCommentVoy IN(SELECT idCommentVoy from Signaler);

20/ Question : La gérante peut connaître les voyageurs avec les commentaires jugés les plus utiles.

On calcule la moyenne des commentaires de chaque utilisateur.

CREATE VIEW ComUtile AS SELECT idVoyageur,AVG(note) AS Utile FROM NoteCommentUser n, CommentVoy c WHERE n.idCommentVoy=c.idCommentVoy GROUP BY idVoyageur;

On prend les utilisateurs ayant au moins une moyenne pour ses (son) commentaires, supérieure ou égale à 4.

SELECT idVoyageur FROM ComUtile WHERE Utile >= 4;

Affiche :

```
+-------------+
| idVoyageur |
+-------------+
| 3          |
+-------------+
```

21/ Question : La gérante peut radier un voyageur qui a trop de commentaires jugés négatifs (plus de 30% des commentaires notés moins bien que 2).

Création de la table des utilisateurs ayant au moins une note en dessous de 2.

create view noteNul as select pseudoUser, note from NoteCommentUser where note<2;

Création de la table des utilisateurs ayant au moins une note au dessus de 2.

create view noteBien as select pseudoUser, note from NoteCommentUser where note>=2;

Compte le nombre de notes en dessous de 2 pour chaque utilisateur.

create view nbNoteNul as select pseudoUser,count(*) as nbNoteNul
from noteNul group by pseudoUser;

Compte le nombre de notes au dessus de 2 pour chaque utilisateur.

create view nbNoteBien as select pseudoUser,count(*) as nbNoteBien
from noteBien group by pseudoUser;

Création de la table des utilisateurs avec le nombre de notes en dessous de 2 ainsi que le nombre de notes au dessus de 2.

Create view nbNul select * from nbNoteNul natural left join nbNoteBien;

create view radier as select pseudoUser,nbNoteNul,nbNoteBien from nbNull
having(nbNoteNul)>=((nbNoteBien+nbNoteNul)*0.30) || nbNoteBien is NULL;

delete from Voyageur where pseudoUser in(select pseudoUSer from radier);

22/ Question : La gérante peut connaître les voyageurs qui ont envoyé des avis en utilisant deux comptes d'utilisateur avec le même numéro IP.

Tout ceux qui sont voyageurs.

```
CREATE VIEW VerifIP AS SELECT v.pseudoUser,IPTerminal FROM Terminal t,Voyageur v WHERE
t.pseudoUser=v.pseudoUser;
```

Jointure de la table Voyageur avec AvisVoy.

```
CREATE VIEW AvisTotal AS SELECT idAvisVoy,pseudoUser FROM AvisVoy NATURAL JOIN Voyageur;
```

Tous les utilisateurs qui ont donné un avis.

```
CREATE VIEW InfoIPTotal AS SELECT * FROM VerifIP NATURAL JOIN AvisTotal;
```

Regroupe tous les utilisateurs apparaissant sous 2 (ou plus) IP différentes.

```
CREATE VIEW FraudeUtilisateur AS SELECT IPTerminal FROM VerifIP GROUP BY IPTerminal HAVING
COUNT(pseudoUser)>=2;

SELECT pseudoUser FROM VerifIP WHERE IPTerminal IN(SELECT IPTerminal FROM
FraudeUtilisateur);
```

Affiche :

```
+--------------+
| pseudoUser |
+--------------+
| Ailwenn      |
| Flydower     |
+--------------+
```

23/ Question : La gérante peut décerner des prix aux meilleurs établissements de chaque type (hôtel, etc.) dans chaque ville. Donnez une requête pour l'aider dans cette tâche.

Tous les restaurants.

```
CREATE VIEW RestTotal AS SELECT idEtab,nomEtab,villeEtab FROM Etablissement NATURAL JOIN
Restaurant;
```

Moyenne des notes des restaurants.

```
CREATE VIEW MoyNoteRest AS SELECT a.idEtab, AVG(nbEtoileAvis) AS Moyenne,villeEtab FROM
AvisVoy a,RestTotal r WHERE a.idEtab=r.idEtab GROUP BY idEtab;
```

Meilleure Moyenne par Ville.

```
CREATE VIEW MaxEtoileVilleRest AS SELECT villeEtab,MAX(Moyenne) AS  MaxMoyenne FROM
MoyNoteRest GROUP BY villeEtab;

SELECT idEtab,villeEtab FROM MoyNoteRest WHERE Moyenne IN(SELECT          MaxMoyenne FROM
MaxEtoileVilleRest) && villeEtab IN(SELECT villeEtab FROM          MaxEtoileVilleRest) ;
```

Affiche :

```
+--------+-------------+
| idEtab | villeEtab   |
+--------+-------------+
| 001    | Paris       |
| 002    | Montpellier |
+--------+-------------+
```

Même chose pour les Magasins.

```
CREATE VIEW MagTotal AS SELECT idEtab,nomEtab,villeEtab FROM Etablissement          NATURAL JOIN
Magasin;

CREATE VIEW MoyNoteMag AS SELECT a.idEtab, AVG(nbEtoileAvis) AS      Moyenne,villeEtab FROM
AvisVoy a,MagTotal m WHERE a.idEtab=m.idEtab GROUP   BY idEtab;

CREATE VIEW MaxEtoileVilleMag AS SELECT villeEtab,MAX(Moyenne) AS MaxMoyenne FROM
MoyNoteMag GROUP BY villeEtab;

SELECT idEtab,villeEtab FROM MoyNoteMag WHERE Moyenne IN(SELECT          MaxMoyenne FROM
MaxEtoileVilleMag) && villeEtab IN(SELECT villeEtab FROM          MaxEtoileVilleMag);
```

Affiche :

```
+--------+--------------------+
| idEtab | villeEtab          |
+--------+--------------------+
| 007    | Paris              |
| 009    | Paris              |
| 011    | Nogent-sur-Marne   |
| 019    | Paris              |
| 020    | Paris              |
+--------+--------------------+
```

Même chose pour les hôtels.

CREATE VIEW HotelTotal AS SELECT idEtab,nomEtab,villeEtab FROM Etablissement NATURAL JOIN Hotel;
->Tous les hotels

CREATE VIEW MoyNoteHotel AS SELECT a.idEtab,AVG(nbEtoileAvis) AS Moyenne,villeEtab FROM AvisVoy a,HotelTotal h WHERE a.idEtab=h.idEtab GROUP BY nomEtab;
->Moyenne note hotel total (hotel noté)

CREATE VIEW MaxEtoileVille AS SELECT villeEtab,MAX(Moyenne) AS MaxMoyenne FROM MoyNoteHotel GROUP BY villeEtab;

SELECT idEtab,villeEtab FROM MoyNoteHotel WHERE Moyenne IN(SELECT MaxMoyenne FROM MaxEtoileVille) && villeEtab IN(SELECT villeEtab FROM MaxEtoileVille) GROUP BY villeEtab;
->Pour les HOTELS

Affiche :

```
+--------+-----------+
| idEtab | villeEtab |
+--------+-----------+
| 017    | Marseille |
| 005    | Metz      |
| 015    | Paris     |
| 006    | Sartene   |
+--------+-----------+
```

24/ Question : La gérante peut trouver tous les utilisateurs n'ayant donné des avis que sur des hôtels et restaurants haut de gamme (nombre d'étoiles officielles ¿=5).

La liste des avis pour les hôtels.

CREATE VIEW AvisHotel AS SELECT idAvisVoy,idVoyageur,etoileofficielleleHotel FROM AvisVoy a,Hotel h WHERE a.idEtab=h.idEtab;

La liste des avis pour les restaurants.

CREATE VIEW AvisRest AS SELECT idAvisVoy,idVoyageur,etoileofficielleleRest FROM AvisVoy a,Restaurant r WHERE a.idEtab=r.idEtab;

Union de la liste des avis pour les hôtels et la liste des avis pour les restaurants.

CREATE VIEW AvisHotelRest AS SELECT * FROM AvisHotel UNION SELECT * FROM AvisRest;

SELECT idVoyageur FROM AvisHotelRest WHERE idVoyageur NOT IN(SELECT idVoyageur FROM AvisHotelRest WHERE etoileofficielleleHotel<=4) GROUP BY idVoyageur;

Affiche :

```
+------------+
| idVoyageur |
+------------+
| 5          |
| 9          |
+------------+
```

25/ Question : La gérante peut connaître l'établissement avec la plus grande différence entre les étoiles officielles et la moyenne des étoiles donnée par les voyageurs.

Donne le nom, le nombre d'étoiles officielles et l'identifiant des hôtels.

```
create view offh as select e.idEtab,nomEtab,etoileofficielleleHotel as etoile from Etablissement e,Hotel h where h.idEtab=e.idEtab;
```

Donne le nom, le nombre d'étoiles officielles et l'identifiant des Restaurants.

```
create view offr as select e.idEtab,nomEtab,etoileofficielleleRest as etoile from Etablissement e,Restaurant r where r.idEtab=e.idEtab;
```

Donner la moyenne des avis de chaque établissement.

```
create view noff as select idEtab,avg(nbEtoileAvis) as avis from AvisVoy group by idEtab;
```

Donne la table étoile officielle et avis pour les hôtels.

```
create view h as select * from offh natural left join noff;
```

Donne la table étoile officielle et avis pour les hôtels.

```
create view r as select * from offr natural left join noff;
```

Calcul des différences pour les restaurants.

```
create view a as select idEtab,nomEtab,max(abs(etoile-avis)) as difference rom r;
```

Calcul des différences pour les Hôtels.

```
create view b as select idEtab,nomEtab,max(abs(etoile-avis)) as difference from h;
```

Donne le nom de l'établissement ayant la plus grande différence entre le nombre d'étoiles officielles et du site.

```
select (select nomEtab from a where difference>(select difference from b)) as nomEtabRestaurant, (select nomEtab from b where difference> (select difference from a)) as nomEtabHotel from a,b;
```

Affiche :

```
+-------------------+--------------+
| nomEtabRestaurant | nomEtabHotel |
+-------------------+--------------+
| L'orange Bleue    | NULL         |
+-------------------+--------------+
```

Conclusion

Nous avons modélisé le projet de la startup Bonconseil en nous basant sur le cahier des charges fourni par Bonconseil puis nous avons créé les tables correspondantes à cette modélisation. Les commandes SQL pour les fonctions de la base de donnée sont fonctionnelles et ont été testées en créant une simulation entre les différentes entités. La base de donnée est donc fonctionnelle et prête à être utilisée par la startup Bonconseil.