--Creation de séquences nécessaires pour l'implémentation des clés primaires de chaque table

```
CREATE SEQUENCE seqConnection
   INCREMENT BY 1 START WITH 1 NOMAXVALUE NOMINVALUE;

CREATE SEQUENCE seqUtilisateur
   INCREMENT BY 1 START WITH 1 NOMAXVALUE NOMINVALUE;

CREATE SEQUENCE seqReservation
   INCREMENT BY 1 START WITH 1 NOMAXVALUE NOMINVALUE;

CREATE SEQUENCE seqPlanning
   INCREMENT BY 1 START WITH 1 NOMAXVALUE NOMINVALUE;

CREATE SEQUENCE seqBien
   INCREMENT BY 1 START WITH 1 NOMAXVALUE NOMINVALUE;

CREATE SEQUENCE seqLocalisation
   INCREMENT BY 1 START WITH 1 NOMAXVALUE NOMINVALUE;

CREATE SEQUENCE seqService
   INCREMENT BY 1 START WITH 1 NOMAXVALUE NOMINVALUE;

CREATE SEQUENCE seqCategorie
   INCREMENT BY 1 START WITH 1 NOMAXVALUE NOMINVALUE;

CREATE SEQUENCE seqSousCategorie
   INCREMENT BY 1 START WITH 1 NOMAXVALUE NOMINVALUE;

CREATE SEQUENCE seqDemande
   INCREMENT BY 1 START WITH 1 NOMAXVALUE NOMINVALUE;
```

```
CREATE SEQUENCE seqAvis
   INCREMENT BY 1 START WITH 1 NOMAXVALUE NOMINVALUE;

CREATE SEQUENCE seqPublicite
   INCREMENT BY 1 START WITH 1 NOMAXVALUE NOMINVALUE;

CREATE SEQUENCE seqFavoris
   INCREMENT BY 1 START WITH 1 NOMAXVALUE NOMINVALUE;



-- On commence la création des tables
-- conection servira lors de la connection d'une personne, une fonction sera appelée et se chargera
-- d'assurer la création/modification/connection/suppression du compte

CREATE TABLE connection
(
   idConnection INT NOT NULL PRIMARY KEY --idConnection -> est un nombre non null lors de l'insertion des données et également une clé primaire de connection.
);



-- La table utilisateur sert à garder les informations relatives a tous les utilisateurs
CREATE TABLE utilisateur
(
   idUtilisateur     INT       DEFAULT seqUtilisateur.NEXTVAL PRIMARY KEY,
--On utilise une séquence pour incrémenter la clé primaire idUtilisateur
   adresse          VARCHAR2(100)                                          NOT NULL,
   codePostal       VARCHAR2(5)
      CONSTRAINT ck_utilisateur_codePosal CHECK (codePostal = TO_NUMBER(codePostal) AND LENGTH(codePostal) = 5) NOT NULL,                    --Grâce a TO_NUMBER on oblige le fait que codePostal ne peut contenir que des chiffres
```

```
    --Avec lenght on lui dit que le nombre de chiffres a inscrire est de 5.

    --NOT NULL implique qu'il faut impérativement saisir un code postal lors de l'insertion de données.

    mobile          VARCHAR2(10)

        CONSTRAINT ck_utilisateur_mobile CHECK (mobile = TO_NUMBER(mobile) AND LENGTH(mobile)
= 10)           NOT NULL,                       -- La meme chose que avec codePostal

    adresseMail     VARCHAR2(50)                                    NOT NULL CHECK
(adresseMail LIKE '_%@_%._%'), --LIKE oblige a inscrire l'attribut d'une certaine manière lors de
l'insertion.

    --Ici on dit que adresseMail doit s'écrire avec 1 caractère ou plus suivi d'un symbole '@' qui est lui
suivi d'un caractère ou plus qui sont suivis d'un point '.' etc.

    motDePasse      VARCHAR2(100)                                   NOT NULL,

    infoPartenaire  NUMBER(1) DEFAULT 0

        CONSTRAINT ck_utilisateur_infoPartenaire CHECK (infoPartenaire = 1 OR infoPartenaire = 0),
--CHECK pose une contrainte sur la variable, dans ce cas on lui dis que l'attribut infoPartenaire prend
comme valeur 1 ou 0 et rien d'autre.

    -- le fait de choisir un bool est plus simple d'utilisation qu'un varchar2

    idConnection    INT                                             NOT NULL,
-- idConnection permet la relation entre la connection d'un utilisateur et ses attributs

    biographie      VARCHAR2(200),

    solde           FLOAT   DEFAULT 0.0,
--DEFAULT est la valeur inscrite d'office quand on insere l'attribut avec null

    totalVente      INT     DEFAULT 0,

    totalAchat      INT     DEFAULT 0,

    nbrDemandes     INT     DEFAULT 0,

    delayMoyenReponse INTEGER,

    tauxReponse     FLOAT   DEFAULT 0.0,

    nbrAvisTotal    INT     DEFAULT 0,

    moyenneNotes    FLOAT   DEFAULT 0,
-- Ici la moyenne de toutes les notes recu par l'utilisateur

    abonnement      NUMBER(1) DEFAULT 0

        CONSTRAINT ck_utilisateur_abonnement CHECK (abonnement = 1 OR abonnement = 0)

);


-- La table avis servira a recueillir tous les avis recu pour un utilisteur
```

```
CREATE TABLE avis
(
   idAvis         INT   DEFAULT seqAvis.NEXTVAL PRIMARY KEY,
   avisUtilisateur VARCHAR2(200),                                  -- Ici l'avis de l'utilisateur
   noteUtilisateur FLOAT DEFAULT 0 CONSTRAINT ck_avis_noteUtilisateur CHECK (noteUtilisateur BETWEEN 0 AND 5), -- On ajoute une contrainte qui dit que noteUtilisateur est un nombre decimal compris entre 0 et 5
   idUtilisateur   INT NOT NULL                                    -- Ici idUtilisateur sera le lien entre l'utilisateur et les avis qu'il a recu
);

-- ici plusieurs tables vont être créées une pour chaque spécificité

-- la table particulier, sert a rentrer les informations d'un particulier
CREATE TABLE utilisateurParticulier
(
   idUtilisateurParticulier INT PRIMARY KEY,
   nom                  VARCHAR2(50) NOT NULL,
   prenom                 VARCHAR2(50) NOT NULL
);

-- la table Entreprise, sert a rentrer les informations d'une entreprise
CREATE TABLE utilisateurEntreprise
(
   idUtilisateurEntreprise INT PRIMARY KEY,
   nomCommercial          VARCHAR2(50) NOT NULL,
   metier               VARCHAR2(50) NOT NULL
);

-- la table Entrepreneur, sert a rentrer les informations d'un Entrepreneur
CREATE TABLE utilisateurEntrepreneur
(
```

```
   idUtilisateurEntrepreneur INT PRIMARY KEY,
   nom                  VARCHAR2(50) NOT NULL,
   prenom               VARCHAR2(50) NOT NULL,
   nomCommercial        VARCHAR2(50) NOT NULL,
   metier               VARCHAR2(50) NOT NULL
);

-- la table association, sert a rentrer les informations d'une association
CREATE TABLE utilisateurAssociation
(
   idUtilisateurAssociation INT PRIMARY KEY,
   nom                  VARCHAR2(50) NOT NULL,
   prenom               VARCHAR2(50) NOT NULL,
   nomCommercial        VARCHAR2(50) NOT NULL
);

-- La table administrateur sert a connaitre quel utilisateur est administrateur
CREATE TABLE administrateur
(
   idAdministrateur INT PRIMARY KEY -- idAdministrateur est a la fois clef primaire et étrangère, elle sera la même que idUtilisateur
);

-- La table Vendeur sert a connaitre quel utilisateur est Vendeur
CREATE TABLE vendeur
(
   idVendeur INT PRIMARY KEY -- idVendeur est a la fois clef primaire et étrangère, elle sera la même que idUtilisateur
);

-- La table clients sert a connaitre quel utilisateur est client
CREATE TABLE clients
```

```
(
   idClient INT PRIMARY KEY -- idclient est a la fois clef primaire et étrangère, elle sera la même que idUtilisateur
);

-- la table planning sert a savoir a qu'elle date et heure un service est proposé par un vendeur
CREATE TABLE planning
(
   idPlanning INT DEFAULT seqPlanning.NEXTVAL PRIMARY KEY,
   date_heure DATE NOT NULL, -- Contient une date et une heure
   idVendeur  INT  NOT NULL  -- permet le lient entre plannin et vendeur permet de savoir quel vendeur propose ce servire
);

-- la table reservation permet a un client de prendre une reservation sur un service proposé
CREATE TABLE reservation
(
   idReservation INT      DEFAULT seqReservation.NEXTVAL PRIMARY KEY,
   libre        NUMBER(1) DEFAULT 0 CONSTRAINT ck_reservation_libre CHECK (libre = 1 OR libre = 0), -- Bool permettehnt de savoir si le créneau est libre (1 libre, 0 pas libre)
   idPlanning    INT NOT NULL,                    -- la relation entre le planning et la reservation
   idClient      INT NOT NULL                     -- la relation entre la reservation et le vlient pour savoir quel client réserve cette plage horaire
);

-- la table catégorie permet de connaire la catégorie du service proposé
CREATE TABLE categorie
(
   idCategorie INT DEFAULT seqCategorie.NEXTVAL PRIMARY KEY,
   nom        VARCHAR2(50) NOT NULL -- nom du service
);
```

```
-- la localisation permet de savoir dans qu'elle localisation est proposé le service

CREATE TABLE localisation

(

   idLocalisation INT DEFAULT seqLocalisation.NEXTVAL PRIMARY KEY,

   adresse       VARCHAR2(100) NOT NULL,                                    -- adresse du service proposé

   codePostal    VARCHAR2(5) CONSTRAINT ck_localisation_codePostal CHECK (codePostal = TO_NUMBER(codePostal) AND LENGTH(codePostal) = 5), -- code postal du service deamndé

   ville         VARCHAR2(30)  NOT NULL                                     -- ville du service proposé

);


-- la table bien permet de connaitres toutes les infos du bien proposé

CREATE TABLE bien

(

   idBien        INT DEFAULT seqBien.NEXTVAL PRIMARY KEY,

   locationEmprunt VARCHAR2(20) CONSTRAINT ck_bien_locationEmprunt CHECK (locationEmprunt LIKE 'LOCATION' OR locationEmprunt like 'EMPRUNT'), -- Contraine -> locationEmprunt peut etre renseigné comme "LOCATION" ou "EMPRUNT" lors de l'insertion rien d'autre.

   nom           VARCHAR2(50) NOT NULL,                                     -- nom du bien

   prix          FLOAT       NOT NULL,                                      -- prix du bien

   description   VARCHAR2(200),                                             -- petite description du bien proposé

   idLocalisation  INT        NOT NULL,                                     -- relation entre la localisation et le bien pour permettre d'y affecter un lieu ou il sera vendu

   idPlanning    INT         NOT NULL,                                      -- relation entre le planning et le bien pour permettre d'y affecter une date

   idVendeur     INT         NOT NULL,                                      -- relation entre le vendeur et le bien pour permettre de savoir qui vend ce bien

   idCategorie   INT         NOT NULL                                       -- relation entre la catégorie et le bien pour permettre de savoir dans qu'elle catégorie classer ce bien

);


-- la table bien permet de connaitres toutes les infos du service proposé

CREATE TABLE services
```

```
(
   idService      INT DEFAULT seqService.NEXTVAL PRIMARY KEY,
   nom            VARCHAR2(50) NOT NULL, -- nom du service
   prix           FLOAT        NOT NULL, -- prix du service
   description    VARCHAR2(200),        -- petite description du service proposé
   idLocalisation INT          NOT NULL, -- relation entre la localisation et le service pour permettre d'y affecter un lieu ou il sera proposé
   idPlanning     INT          NOT NULL, -- relation entre le planning et le service pour permettre d'y affecter une date
   idVendeur      INT          NOT NULL, -- relation entre le vendeur et le service pour permettre de savoir qui propose ce service
   idCategorie    INT          NOT NULL  -- relation entre la catégorie et le service pour permettre de savoir dans qu'elle catégorie classer ce service
);



-- la table demande permet d'enregistrer les demandes que peut faire un utilisateur
CREATE TABLE demande
(
   idDemande     INT DEFAULT seqDemande.NEXTVAL PRIMARY KEY,
   description   VARCHAR2(200),  -- description de la demande
   prixMax       FLOAT NOT NULL, -- prix maximum que l'utilisateur est pret a mettre
   idUtilisateur INT   NOT NULL  -- relation entre le l'utilisateur et la demande pour permettre de savoir qui fait cette demande
);


-- la table publicite permet d'enregistrer les publicite que peut faire un administrateur du site
CREATE TABLE publicite
(
   idPublicite      INT DEFAULT seqPublicite.NEXTVAL PRIMARY KEY,
   description      VARCHAR2(200), -- description de la pub
   idAdministrateur INT NOT NULL   -- relation entre le l'utilisateur et la publicité pour permettre de savoir qui fait cette publicité
```

```
);

-- la table favoris permet d'enregistrer les favoris que peut avoir un utilisateur
CREATE TABLE favoris
(
   idFavoris INT DEFAULT seqFavoris.NEXTVAL PRIMARY KEY,
   idClient  INT NOT NULL, -- relation qui permet de savoir qui a ce favoris
   idVendeur INT NOT NULL  -- relation qui permet de savoir qui a été enregistré en tant que favoris
);

ALTER TABLE utilisateur
   ADD CONSTRAINT fk_utilisateur_idConnection FOREIGN KEY (idConnection)
      REFERENCES connection (idConnection) ON DELETE CASCADE; -- ON DELETE CASCADE supprime dans la table utilisateur un idConnection automatiquement lorsque celui ci est supprimé

ALTER TABLE avis
   ADD CONSTRAINT fk_avis_idUtilisateur FOREIGN KEY (idUtilisateur)
      REFERENCES utilisateur (idUtilisateur) ON DELETE CASCADE;

ALTER TABLE utilisateurParticulier
   ADD CONSTRAINT fk_utilisateurParticulier_idUtilisateurParticulier FOREIGN KEY (idUtilisateurParticulier)
      REFERENCES utilisateur (idUtilisateur) ON DELETE CASCADE;

ALTER TABLE utilisateurEntreprise
   ADD CONSTRAINT fk_utilisateurEntreprise_idUtilisateurEntreprise FOREIGN KEY (idUtilisateurEntreprise)
      REFERENCES utilisateur (idUtilisateur) ON DELETE CASCADE;

ALTER TABLE utilisateurEntrepreneur
   ADD CONSTRAINT fk_utilisateurEntrepreneur_idUtilisateurEntrepreneur FOREIGN KEY (idUtilisateurEntrepreneur)
```

```
    REFERENCES utilisateur (idUtilisateur) ON DELETE CASCADE;

ALTER TABLE utilisateurAssociation
  ADD CONSTRAINT fk_utilisateurAssociation_idUtilisateurAssociation FOREIGN KEY
(idUtilisateurAssociation)
    REFERENCES utilisateur (idUtilisateur) ON DELETE CASCADE;

ALTER TABLE administrateur
  ADD CONSTRAINT fk_administrateur_idAdministrateur FOREIGN KEY (idAdministrateur)
    REFERENCES utilisateur (idUtilisateur) ON DELETE CASCADE;

ALTER TABLE vendeur
  ADD CONSTRAINT fk_vendeur_idVendeur FOREIGN KEY (idVendeur)
    REFERENCES utilisateur (idUtilisateur) ON DELETE CASCADE;

ALTER TABLE clients
  ADD CONSTRAINT fk_clients_idClient FOREIGN KEY (idClient)
    REFERENCES utilisateur (idUtilisateur) ON DELETE CASCADE;

ALTER TABLE planning
  ADD CONSTRAINT fk_planning_idVendeur FOREIGN KEY (idVendeur)
    REFERENCES VENDEUR (idVendeur) ON DELETE CASCADE;

ALTER TABLE reservation
  ADD CONSTRAINT fk_reservation_idPlanning FOREIGN KEY (idPlanning)
    REFERENCES planning (idPlanning) ON DELETE CASCADE;

ALTER TABLE reservation
  ADD CONSTRAINT fk_reservation_idClient FOREIGN KEY (idClient)
    REFERENCES clients (idClient) ON DELETE CASCADE;
```

```
ALTER TABLE bien
   ADD CONSTRAINT fk_bien_idLocalisation FOREIGN KEY (idLocalisation)
      REFERENCES localisation (idLocalisation) ON DELETE CASCADE;

ALTER TABLE bien
   ADD CONSTRAINT fk_bien_idVendeur FOREIGN KEY (idVendeur)
      REFERENCES vendeur (idVendeur) ON DELETE CASCADE;

ALTER TABLE bien
   ADD CONSTRAINT fk_bien_idCategorie FOREIGN KEY (idCategorie)
      REFERENCES categorie (idCategorie) ON DELETE CASCADE;

ALTER TABLE services
   ADD CONSTRAINT fk_services_idLocalisation FOREIGN KEY (idLocalisation)
      REFERENCES localisation (idLocalisation) ON DELETE CASCADE;

ALTER TABLE services
   ADD CONSTRAINT fk_services_idPlanning FOREIGN KEY (idPlanning)
      REFERENCES planning (idPlanning) ON DELETE CASCADE;

ALTER TABLE services
   ADD CONSTRAINT fk_services_idVendeur FOREIGN KEY (idVendeur)
      REFERENCES vendeur (idVendeur) ON DELETE CASCADE;

ALTER TABLE services
   ADD CONSTRAINT fk_services_idCategorie FOREIGN KEY (idCategorie)
      REFERENCES categorie (idCategorie) ON DELETE CASCADE;

ALTER TABLE demande
   ADD CONSTRAINT fk_demande_idUtilisateur FOREIGN KEY (idUtilisateur)
      REFERENCES utilisateur (idUtilisateur) ON DELETE CASCADE;
```

```
ALTER TABLE publicite
   ADD CONSTRAINT fk_publicite_idAdministrateur FOREIGN KEY (idAdministrateur)
      REFERENCES ADMINISTRATEUR (idAdministrateur) ON DELETE CASCADE;

ALTER TABLE favoris
   ADD CONSTRAINT fk_favoris_idCLient FOREIGN KEY (idCLient)
      REFERENCES Clients (idCLient) ON DELETE CASCADE;

ALTER TABLE favoris
   ADD CONSTRAINT fk_favoris_idVendeur FOREIGN KEY (idVendeur)
      REFERENCES Vendeur (idVendeur) ON DELETE CASCADE;
```