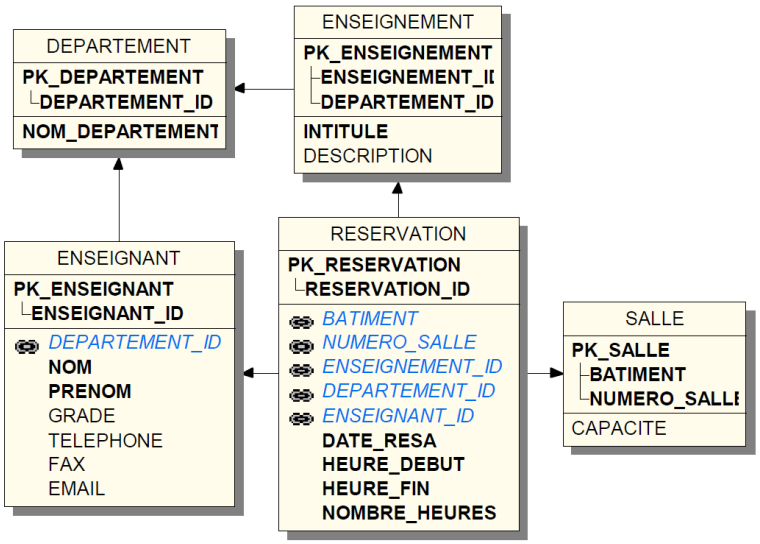
TRAVAUX PRATIQUES 3

Langage SQL

BASE DE DONNÉES DE GESTION DE LA SCOLARITÉ

Soit le schéma relationnel suivant de la base de données d'un système de gestion de la scolarité d'une faculté :



1. Créer le schéma de données de la base en exécutant dans l'interpréteur de requêtes SQL le script `creationBDscolarite.sql` (à télécharger à partir de ce lien https://gist.github.com/elyeslamine/5fb369462451ed915abd)

- *Une version papier du script de création de la base exemple est donnée dans la partie Annexe*

Soit l'extension suivante de la table Département

– Table Département

```
departement_id | nom_departement
---------------+----------------
             1 | ISIS
             2 | INFO
             3 | SHS
```

2. Remplir la table Département en utilisant une séquence pour générer automatiquement le numéro de département.
3. Exécuter le script BDScolariteInsertion.sql pour peupler les autres tables (à télécharger à partir de ce lien https://gist.github.com/elyeslamine/e9573c1a720393570935)
4. Créer une vue permettant de visualiser le nombre de réservation par enseignant.
5. Quels sont les enseignants ayant le plus de réservations
6. Quels sont les noms et les prénoms des enseignants n'ayant aucune réservation ?

7. Créer la vue Info_Enseignant (Matricule, Nom, Prenom, email) à partir de la table Enseignant. Vérifier son contenu.
8. À travers la vue Info_Enseignant, modifier l'adresse mel de l'enseignant Lamine du « elyes.lamine@univ-jfc.fr » en «elyes.lamine@gmail.com». Consulter le contenu de la vue Info_Enseignant et de la table Enseignant.
9. Insérez à partir de la vue Info_Enseignant un nouvel enseignant.
10. Créer une fonction Sql GetSalleCapaciteSuperieurA permettant de récupérer la liste des salles ayant une capacité supérieure à une certaine valeur donnée en paramètre.
11. Créer une fonction Sql GetDepartement_ID permettant de récupérer l'identificateur du département à partir de son nom donné en paramètre. Tester cette fonction pour afficher les nom et prénom des enseignants du département « ISIS »
12. Écrire une fonction pgsql SonDepartement qui admette un numéro d'enseignant en paramètre et qui renvoie comme résultat le nom du département de l'enseignant. Tester la fonction.
13. Écrire en pl/pgsql une fonction MoyCapacite sans paramètre qui renvoie la capacité moyenne des salles.
14. Utiliser la fonction pgsql MoyCapacite pour afficher les salles (batiment, numero et capacité) qui ont une capacité superieur à la moyenne,
15. Utiliser la fonction pgsql MoyCapacite pour afficher les salles (batiment, numero et capacité) qui ont une capacité est égal à la capacité moyenne à 15% près (c'est-à-dire ceux dont la capacité est comprise entre 85% et 115% de la capacité moyenne).
16. Écrire une fonction pgsql Collegues qui admet un numéro d'enseignant en paramètre et qui renvoie comme résultat le nom et le prénom de ses collègues du même département, l'enseignant lui-même ne devant pas faire partie de la liste des collègues.
17. Écrire une fonction pgsql numlignes qui renvoie le nombre de lignes de la table dont le nom est passé en paramètre.
18. Ecrire le trigger postgresql qui convertit le nom du nouveau Enseignant inséré en majuscule.
19. Créer un déclencheur attaché à l'événement « insertion d'une nouvelle réservation ». Ce trigger va générer une valeur pour le numéro de la nouvelle réservation.
20. Créer un déclencheur nommé trace ayant pour but de créer un fichier journal.
21. Créer une vue permettant de visualiser le nombre de réservation par enseignant.
22. Quels sont les noms et les prénoms des enseignants pour lesquels il existe au moins deux réservations ? (utiliser EXISTS puis une autre solution en utilisant la vue créée précédemment).
23. Quels sont les enseignants ayant le plus de réservations (Utiliser la Vue définie à la question 18 et le mot-clé ALL) ?
24. Quels sont les noms et les prénoms des enseignants n'ayant aucune réservation ?
25. Proposer un déclencheur permettant de vérifier que, lors de l'ajout ou de la modification d'une réservation, que cette réservation est possible et si ce n'est pas le cas, va afficher un message d'erreur. Par exemple, le déclencheur va vérifier que le créneau horaire choisi pour une réservation n'est pas contenu dans le(s) créneau(x) horaire(s) de réservations existantes ou ne chevauche pas le(s) créneau(x) horaire(s) de réservations existantes.

```
DROP TABLE IF EXISTS Reservation;
DROP TABLE IF EXISTS Salle;
DROP TABLE IF EXISTS Enseignement;
DROP TABLE IF EXISTS Enseignant;
DROP TABLE IF EXISTS Departement;
-- Script de création des relations

CREATE TABLE Departement
(
 Departement_id     SERIAL,
 Nom_Departement   varchar(25) NOT NULL,
 CONSTRAINT UN_Nom_Departement UNIQUE (nom_departement),
 CONSTRAINT PK_Departement PRIMARY KEY(Departement_ID)
);

CREATE TABLE Enseignement
(
  Enseignement_ID  int4 NOT NULL,
  Departement_ID   int4 NOT NULL,
  Intitule         varchar(60) NOT NULL,
  Description      varchar(1000),
  CONSTRAINT PK_Enseignement PRIMARY KEY (Enseignement_ID, Departement_ID),
  CONSTRAINT  "PK_Enseignement_Departement"  FOREIGN  KEY  (Departement_ID)
REFERENCES  Departement  (Departement_ID)  ON  UPDATE  RESTRICT  ON  DELETE
RESTRICT
) ;
CREATE TABLE Enseignant
(
 Enseignant_ID     integer,
 Departement_ID    integer NOT NULL,
 Nom                 varchar(25) NOT NULL,
 Prenom            varchar(25) NOT NULL,
 Grade             varchar(25)
 CONSTRAINT CK_Enseignant_Grade
 CHECK (Grade IN ('Vacataire', 'Moniteur','ATER', 'MCF', 'PROF')),
 Telephone         varchar(10) DEFAULT NULL,
 Fax               varchar(10) DEFAULT NULL,
 Email             varchar(100) DEFAULT NULL,
 CONSTRAINT PK_Enseignant PRIMARY KEY (Enseignant_ID),
 CONSTRAINT  "FK_Enseignant_Departement_ID"  FOREIGN  KEY  (Departement_ID)
REFERENCES  Departement  (Departement_ID)  ON  UPDATE  RESTRICT  ON  DELETE
RESTRICT
);
CREATE TABLE Salle
(
 Batiment         varchar(1),
 Numero_Salle   varchar(10),
 Capacite   integer CHECK (Capacite >1),
 CONSTRAINT PK_Salle PRIMARY KEY (Batiment, Numero_Salle)
);
```

```
CREATE TABLE Reservation
(
 Reservation_ID     integer,
 Batiment           varchar(1) NOT NULL,
 Numero_Salle       varchar(10) NOT NULL,
 Enseignement_ID    integer NOT NULL,
 Departement_ID     integer NOT NULL,
 Enseignant_ID      integer NOT NULL,
 Date_Resa          date NOT NULL DEFAULT CURRENT_DATE,
 Heure_Debut          time NOT NULL DEFAULT CURRENT_TIME,
 Heure_Fin      time NOT NULL DEFAULT '23:00:00',
 Nombre_Heures        integer NOT NULL,
 CONSTRAINT PK_Reservation PRIMARY KEY (Reservation_ID),
 CONSTRAINT  "FK_Reservation_Salle"  FOREIGN  KEY  (Batiment,Numero_Salle)
REFERENCES  Salle  (Batiment,Numero_Salle)  ON  UPDATE  RESTRICT  ON  DELETE
RESTRICT,
 CONSTRAINT        "FK_Reservation_Enseignement"        FOREIGN        KEY
(Enseignement_ID,Departement_ID)          REFERENCES          Enseignement
(Enseignement_ID,Departement_ID) ON UPDATE RESTRICT ON DELETE RESTRICT,
 CONSTRAINT   "FK_Reservation_Enseignant"   FOREIGN   KEY   (Enseignant_ID)
REFERENCES  Enseignant  (Enseignant_ID)  ON  UPDATE  RESTRICT  ON  DELETE
RESTRICT,
 CONSTRAINT CK_Reservation_Nombre_Heures CHECK (Nombre_Heures >=1),
 CONSTRAINT CK_Reservation_HeureDebFin CHECK (Heure_Debut < Heure_Fin)
);
```