TP1 – Base de données – SQL

Rendre le TP avant minuit en le chargeant sur *Tomuss*. Le rapport doit contenir vos requêtes SQL et/ou vos réponses aux questions, avec vos commentaires/explications et captures d'écran.

Introduction

Le point d'entrée aux serveurs est `bd-pedago.univ-lyon1.fr`

- il redirige ses ports vers les différents serveurs (postgres, mongo, etc.)
- accessible uniquement depuis le campus :
  - utiliser le réseau EDUROAM en wifi
  - utiliser un tunnel SSH ou le VPN depuis l'exterieur, cf. `https ://documentation.univ-lyon1.fr/`

Dans ce TP, nous travaillons sur une instance de la base de données PostgreSQL. Le port TCP sur lequel écoute l'instance est celui par défaut 5432. Pour se connecter, avec **LoginPostgres** l'identifiant de la forme pgNUMERO_ETUDIANT et le mot de passe **MotDePasse** tous les deux donnés dans `https ://tomuss.univ-lyon1.fr/`. la base a le même nom que le login (ici **LoginPostgres**).

Pour se connecter au serveur Postgres, on peut utiliser la ligne de commande interactive `psql`, ou les clients DBeaver ou pgAdmin. Dans le cadre de ce TP, nous vous recommandons d'utiliser DBeaver.

Mise en Place

1. Lancez le logiciel *DBeaver* dans la session **Windows**
2. Cliquez sur le bouton "New Database Connection"
3. Sélectionnez la base de données *PostgreSQL* se trouvant dans l'onglet *SQL*
4. Cliquez sur *Next*
5. Remplissez les informations suivantes :
   - *Host* : bd-pedago.univ-lyon1.fr
   - *Database* : voir **LoginPostres** dans `https ://tomuss.univ-lyon1.fr/`
   - *User* : voir **LoginPostres** dans `https ://tomuss.univ-lyon1.fr/`
   - *Password* : voir **MotDePasse** dans `https ://tomuss.univ-lyon1.fr/`

6. Testez la connection en cliquant sur "Test Connection ...". Un message vous indique qu'il faut télécharger et installer des libriaries manquantes peut être affiché. Autorisez-le de les télécharger.

7. Cliquez sur *Finish*

Une fois la connection est établie, vous pouvez commencez le TP. Pour exécuter des commandes SQL, cliquez sur le menu *SQL Editor* > *SQL Editor*

1 Type de données

Définissez le type de donnée le mieux approprié pour spécifier :

1. un nom de jour de la semaine
2. un nom de mois de l'année
3. un numéro de semaine
4. un code postal

cf. https ://www.postgresql.org/docs/9.2/datatype-character.html

2 Base de données « Université »

Nous considérons la base de données d'une université, composée essentiellement des tables suivantes :

```
Etudiant (numetu, nom, prenom, datenaiss, rue, cp, ville)
Matiere (codemat, libelle, coef)
Epreuve (numepreuve, lieu, codemat)
Notation (numetu, numepreuve, note)
```

2.1 Définition des données

L'objectif de cet exercice est de se familiariser avec les commandes : `CREATE`, `DROP`, `ALTER`, en créant certaines tables présentées dans le modèle relationnel ci-dessus.

1. Créer la table **`Etudiant`** représentant les étudiants de l'université. Cette table contient une clé primaire **`numetu`** constituée de 3 caractères, un nom de l'étudiant `nom` (20 caractères), un prénom `prenom` (20 caractères), une date de naissance `datenaiss` (date) et une adresse composée d'une `rue` (50 caractères), d'un code postal `cp` (5 caractères) et d'un nom de `ville` (15 caractères). Vérifier si la création s'est bien passée !

2. Créer la table **`Matiere`** représentant les matières enseignées à l'université. Cette table contient une clé primaire **`codemat`** constituée de 10 caractères max, un libellé `libelle` (50 caractères), et un coefficient numérique `coef` (deux chiffres et un décimal) pour stocker le coefficient de la matière. Vérifier si la création s'est bien passée !

3. Créer la table **Epreuve** représentant les épreuves effectuées à l'université. Chaque épreuve est identifiée par la clé primaire **numepreuve** de type entier. De plus l'attribut `lieu` permet de spécifier le lieu de l'épreuve. Cette table contient également une clé étrangère `codemat` permettant d'identifier la matière concernée par l'épreuve. Vérifier si la création s'est bien passée.

4. Créer la table **Notation** permettant d'enregistrer les notes des étudiants. Cette table contient deux clés étrangères `numetu` et `numepreuve` qui sont forcément non nulles. Le couple **numetu** et **numepreuve** forme la clé primaire de cette table. De plus, on stocke dans cette table l'attribut `note` de type numérique (trois chiffres et deux décimales). Vérifier si la création s'est bien passée.

5. Après coup, vous vous rendez compte qu'il est nécessaire de stocker une information supplémentaire sur la date de l'épreuve effectuée à l'université. Ajouter dans la table **Epreuve** un attribut `datepreuve` constitué de 8 caractères (pour stocker par exemple 05122013).

6. Finalement le choix d'attribuer le type char à l'attribut `datepreuve` n'était pas judicieux. Modifier l'attribut `datepreuve` de la table **Epreuve** pour lui associer le type date :

```
ALTER TABLE Epreuve
ALTER COLUMN datepreuve TYPE DATE
USING datepreuve::date;
```

7. Créer la table **Enseignant** représentant les enseignants de l'université. Cette table contient une clé primaire **numens** constituée de 3 caractères, un `nom` (20 caractères), un `prenom` (20 caractères) une `ville` représentant son adresse (20 caractères) et une date d'embauche `datemb` (date). Vérifiez que la création s'est bien passée !

8. Insérer l'enseignant François Pignon dans la table **Enseignant**. François est embauché le 01/09/2009 et habite à Villeurbanne et identifié par le numéro 130.

9. Modifier l'adresse de l'enseignant Pignon en déclarant sa nouvelle adresse à Écully. Vérifiez que le n-uplet est mis à jour !

10. Finalement, la table **Enseignant** ne vous intéresse pas. Supprimez-la.

11. Afin d'améliorer les performances d'accès aux données de notre base, nous proposons de définir des index sur les clés primaires de chacune des tables.

    (a) Créer l'index `XET` sur la clé primaire de la table **Etudiant**.
    (b) Créer l'index `XM` sur la clé primaire de la table **Matiere**.
    (c) Créer l'index `XEP` sur la clé primaire de la table **Epreuve**.

12. Finalement les index sur les tables ne présentent pas d'intérêt. Supprimez-les.

13. Comment vous y prendriez-vous pour supprimer toutes les tables ? Y a-t-il un ordre de suppression à respecter ?

2.2 Insertion des données

L'objectif de cet exercice est de voir comment insérer des tuples dans les différentes tables de notre base (Utilisation de la commande `INSERT INTO`).

1. Insérer dans la table **`Etudiant`** l'étudiant Albert DUPONT. Albert est né le premier juin 1980 et habite dans la rue de la République à Lyon. Il est identifié par la valeur 110. Vérifiez que le n-uplet est bien inséré !
2. Insérer dans la table **`Etudiant`** l'étudiante Marie MARTIN se trouvant à l'adresse « Rue des Acacias » à Écully. Essayez de lui associer l'identifiant 110. Que se passe-t-il ? Pourquoi ?
3. Pour remplir les tuples dans les tables, il vous suffit de récupérer le script `tp1.sql`. Ce script vous permet d'insérer les tuples dans les tables déjà créées.

   Copiez le script dans votre terminal et exécutez ! Maintenant passez à la suite…

2.3 Requêtes SQL

1. Donner la liste de tous les étudiants, classée par ordre alphabétique décroissant.
2. Donner le libellé et le coefficient (exprimé en pourcentage) de chaque matière.
3. Donner le nom et prénom des étudiants domiciliés à Lyon.
4. Donner la liste des notes supérieures ou égales à 10.
5. Donner la liste des épreuves dont la date se situe entre le 1er janvier et le 30 juin 2004.
6. Donner le nombre total d'épreuves.
7. Donner le nombre de notes indéterminées (`NULL`).
8. Donner la liste des notes en précisant pour chacune le nom et le prénom de l'étudiant qui l'a obtenu.
9. Donner les moyennes de notes de chaque étudiant (indiquer le nom et le prénom), classées de la meilleure à la moins bonne.
10. Donner les moyennes des notes pour les matières (indiquer le libellé) comportant plus d'une épreuve.

3 Base de données « Représentant »

Exemple choisi : La société X utilise le logiciel de gestion de base de données *Access* pour gérer ses clients et ses représentants.

Voici la liste des tables crées dans *Access* :

La table **Representant**

num_rep	nom_rep	ad_rep	cp_rep	vil_rep	age_rep
1	Delmotte	18 rue Aristide Briand	750	Paris	26
2	Hinaud	25 rue Martel	941	Fontenay-sous-Bois	31
3	Lapierre	89 rue Gaston Daguenet	951	Argenteuil	52
4	Latour	7 rue du Four	917	Fleury-Mérogis	44
5	Lemoine	5 rue Auboïs	917	Fleury-Mérogis	28
6	Lemoine	12 route des Gardes	931	Bondy	34

La table **Couvrir**

num_rep	cod_dep
1	75
1	94
2	93
2	94
3	91
3	75
4	95
5	93
5	91
6	92
6	95

La table **Departement**

cod_dep	nom_dep	chef_secteur
75	Paris	Pons
91	Essonne	Bertrand
92	Hauts-de-Seine	Fischer
93	Seine-Saint-Denis	Fischer
94	Val de Marne	Bertrand
95	Val d'Oise	Bertrand

La table **Client**

code_clt	nom_clt	**num_rep**	**num_cat**
1	Boccard	1	1
2	Raldi	2	1
3	Pierrol	2	3
4	Engeli	2	3
5	Atr	4	2
6	Partoli	4	3
…			

La table **Categorie_tarifaire**

num_cat	nom_cat	remise
1	Entreprises	10 %
2	Collectivités	50 %
3	Particuliers	0 %

Le modèle relationnel correspondant :

```
Representant (num_rep, nom_rep, ad_rep, cp_rep, vil_rep, age_rep)
Couvrir (#num_rep, #code_dep)
Departement (code_dep, nom_dep, chef_secteur)
Client (code_clt, nom_clt, #num_rep, #num_cat)
Categorie_tarifaire (num_cat, nom_cat, remise)
```

Écrire les requêtes suivantes :

1. Afficher la liste des clients appartenant à la catégorie tarifaire n°1, classée par ordre alphabétique.
2. Afficher la liste des clients (code, nom de client) rattachés au représentant Hinaud.
3. Afficher la liste des clients bénéficiant d'une remise de 10 %.
4. Afficher la liste des représentants (Numéro et nom) dépendant du chef de secteur Pons.
5. Afficher la liste des départements (code, nom, chef de secteur).
6. Afficher la liste des chefs de secteur.