CM 1 - Définition, contrôle et manipulation

# 1 - Administration PostgreSQL

## 1.1 - Création, modification et suppression

Les commandes suivantes sont valables pour les bases, les schémas, les tables et les rôles.

| Création | CREATE TABLE my\_table(); |
| --- | --- |
| Renommage | ALTER TABLE my\_table\_1 RENAME TO my\_table\_2; |
| Changement de propriétaire | ALTER TABLE my\_table OWNER TO my\_role; |
| Attribution de droits | GRANT ALL ON TABLE my\_table TO my\_role; |
| Révocation de droits | REVOKE ALL ON TABLE my\_table FROM my\_role; |
| Suppression | DROP TABLE my\_table; |

## 1.2 - Sélection d’un schéma

| Voir les schémas actifs | SHOW search\_path; |
| --- | --- |
| Se placer dans le schéma my\_schema | SET search\_path TO my\_schema; |

## 1.3 - Remaniement d’une table

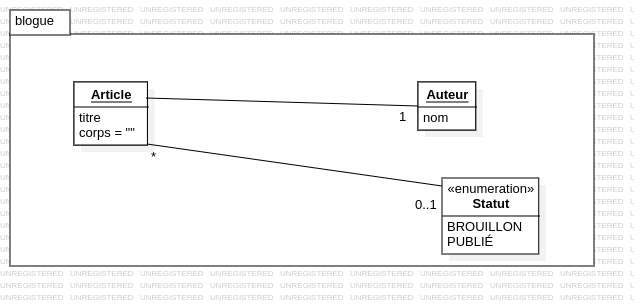
| Ajouter un champ my\_field de type TEXT à my\_table | ALTER TABLE my\_table ADD COLUMN my\_field TEXT; |
| --- | --- |
| Supprimer le champ my\_field de my\_table | ALTER TABLE my\_table DROP COLUMN my\_field; |
| Renommer le champ my\_field en my\_attribute | ALTER TABLE my\_table RENAME my\_field TO my\_attribute |
| Ajouter une contrainte de non nullité | ALTER TABLE my\_table ALTER COLUMN my\_column SET NOT NULL; |
| Ajouter une valeur par défaut | ALTER TABLE my\_table ALTER COLUMN my\_column SET DEFAULT ''; |
| Supprimer la contrainte my\_constraint de my\_table | ALTER TABLE my\_table  DROP CONSTRAINT my\_constraint |
| Ajouter une contrainte de clé étrangère table\_2(k) sur table\_1 | ALTER TABLE table\_1 ADD CONSTRAINT table\_1\_fkey FOREIGN KEY (k) REFERENCES table\_2(k) ON UPDATE CASCADE; |

# 2 - Types de base de PostgreSQL

| INTEGER (INT) | Nombres entiers |
| --- | --- |
| DECIMAL (DEC) | Nombres décimaux |

| TEXT | Chaîne de caractère |
| --- | --- |
| DATE | Dates |

# 3 - Définitions des tables



| CREATE SCHEMA blog;  CREATE TYPE blog.STATUS AS ENUM('brouillon', 'publié');  CREATE TABLE blog.authors(  author\_id SERIAL PRIMARY KEY,  name TEXT NOT NULL );  CREATE TABLE blog.articles (  article\_id SERIAL PRIMARY KEY,  title TEXT NOT NULL,  body TEXT DEFAULT '',  author INT NOT NULL REFERENCES blog.authors(author\_id)  ON UPDATE CASCADE  ON DELETE SET NULL,  status blog.STATUS NOT NULL ); |
| --- |

# 4 - Manipulation des données

| Insérer deux brouillons vides dans la table blog.articles. | INSERT INTO blog.articles (title, author, status) VALUES ('Tux porté disparu', 1, 'brouillon'),  ('Tux peut-être mort', 1, 'brouillon'); |
| --- | --- |
| Supprimer l’article n°5 de la table blog.articles. | DELETE FROM blog.articles WHERE article\_id = 5; |
| Renommer l’article n°5 en « Tux retrouvé ». | UPDATE blog.articles SET title = 'Tux retrouvé' WHERE article\_id = 5; |

Interrogation des données

# 1 - Requêtes simples

* Sélectionner les champs my\_field\_1 et my\_field\_2 de la table my\_table:

| SELECT mt.my\_field\_1, mt.my\_field\_2  FROM my\_table AS mt |
| --- |

* Sélectionner le champ my\_field de la table my\_table et supprimer les doublons:

| SELECT DISTINCT mt.my\_field  FROM my\_table AS mt; |
| --- |

* Sélectionner le champs my\_field de la table my\_table et trier l’affichage par ordre décroissant:

| SELECT mt.my\_field AS mf  FROM my\_table AS mt ORDER BY mt DESC; |
| --- |

* Sélectionner le champs my\_field de la table my\_table pour des valeurs comprises entre my\_min et my\_max:

| SELECT mt.my\_field FROM my\_table AS mt WHERE mt.my\_field BETWEEN my\_min AND my\_max; |
| --- |

* Arrondir my\_result à 0.01:

| SELECT ROUND(mu\_result, 2); |
| --- |

* Faire précéder my\_result de « Résultat: »:

| SELECT FORMAT('Résultat: %s', my\_result); |
| --- |

# 2 - Agrégation

* Calculer la somme des valeurs du champ my\_field de la table my\_table :

| SELECT SUM(mt.my\_field)  FROM my\_table AS mt; |
| --- |

* Compter les occurrences des différentes valeurs du champ my\_field dans la table my\_table de clé primaire my\_id:

| SELECT mt.my\_field AS mf, COUNT(mt.my\_id)  FROM my\_table AS mt GROUP BY mf; |
| --- |

* Calculer la moyenne du champ my\_value pour chaque valeur du champ my\_field dans la table my\_table lorsque my\_field n’est pas nul :

| SELECT mt.my\_field AS mf, AVG(my\_value)  FROM my\_table AS mt GROUP BY mf  HAVING mf IS NOT NULL; |
| --- |

# 3 - Expressions de tables communes

**CTE**

Common Table Expression

* Stocker dans une table un résultat auxiliaire provisoire:

| WITH my\_aux\_table(my\_id, my\_value)  AS  (SELECT ...)  SELECT ... |
| --- |

Consignes pour les travaux pratiques

# Début de TP

## Clonage du dépôt Git

| git clone https://git.unilim.fr/votre-dépôt-r2.06.git cd votre-dépôt-r2.06 |
| --- |

## Connexion au serveur PostgreSQL

1. Connectez-vous au serveur Postgres ainsi :

psql

1. Tapez la commande \? pour voir les commandes disponibles

La touche Espace permet de changer de page, et la touche B de revenir en arrière.

1. Tapez la commande \h pour obtenir un aide-mémoire SQL.
2. Puis tapez par exemple \h CREATE DATABASE pour voir la fiche de documentation de l’instruction CREATE

DATABASE.

| \c my\_base my\_role | se connecter à my\_base en tant que my\_role |
| --- | --- |
| \l+ | lister les bases de données |
| \du+ | lister les rôles |
| \dn+ | lister les schémas |
| \dt+ | lister les tables |
| \dp+ | lister les permissions |
| \d+ my\_table | examiner la structure de my\_table |

# Pendant le TP

• Mettez bien le point-virgule à la fin de chaque instruction SQL.

• Utilisez la complétion automatique du terminal (touche TAB).

• Rappelez les commandes précédentes au moyen de la touche FLÈCHE HAUT;

• Tapez Ctrl + A pour aller au début de la ligne de commande.

• Tapez Ctrl + E pour aller à la fin de la ligne de commande.

## Exécution de commandes à distance

### Directement

| psql -c 'DROP TABLE my\_table;' |
| --- |

### Depuis un fichier SQL

| psql < my\_file.sql |
| --- |

# Fin de TP

## Sauvegardes diverses

* Sauvegarde de la base my\_database en entier :

| mkdir tp1 pg\_dump my\_database > tp1/my\_database.sql |
| --- |

* Sauvegarde de la structure de la base my\_database (sans le contenu) :

| pg\_dump my\_database --schema-only > tp1/my\_database\_schema.sql |
| --- |

* Sauvegarde du contenu de la base my\_database (sans la structure) :

| pg\_dump my\_database --data-only > tp1/my\_database\_data.sql |
| --- |

* Sauvegarde des rôles :

| pg\_dumpall --roles-only > tp1/my\_roles.sql |
| --- |

## Synchronisation Git

1. Ajoutez les sauvegardes à l’index de Git :

| git add \* |
| --- |

1. Vérifiez l’état de l’index de Git :

| git status |
| --- |

1. Créez une validation :

| git commit -m "message explicatif" |
| --- |

1. Envoyez vos validations sur la forge :

| git push |
| --- |

1. Vérifiez que vos fichiers et vos validations apparaissent bien sur https://git.unilim.fr/votre-dépôt-r2.06.git.