

TP SQL

1 Environnement

1.1 Se connecter à la base

Connectez-vous à la base postgresql comme indiqué dans le document annexe (Connexion postgresQL).

Le département dispose déjà d'un serveur postgresQL sur lequel un compte a été attribué à chaque étudiant.

Plus exactement, au lieu de créer une base par étudiant, au département on a créé une seule base de donnée dans laquelle vous avez tous des comptes. Vous courrez donc le risque que la première table créée apparaisse chez tout le monde (ce qui empêchera les autres de créer une table similaire puisque le système leur dira qu'il existe déjà une telle table). C'est pourquoi nous allons dans la prochaine étape créer des schémas qui se comportent un peu comme un "répertoire de fichiers" en permettant d'isoler une partie de la base de donnée. Nous allons donc créer un schéma par étudiant et s'assurer que les tables que vous créez ou manipuler sont bien dans ce schéma.

Comme indiqué dans le document annexe, lorsque vous êtes physiquement chez vous et non à l'université, vous pouvez tout de même vous connecter à la base, en vous connectant par `ssh` sur une des machines du département. Il est aussi très facile d'installer postgresQL (soit directement, soit avec docker) mais il est préférable que vous preniez l'habitude de travailler dans l'environnement proposé par le département puisque c'est celui sur lequel vous devrez travailler (sur des comptes en mode examen créés spécialement pour l'occasion) lors du TP noté.

1.2 Création d'un "schéma postgresql"

Créer un "schéma postgresql", que vous nommerez `tp1_[jdoe]` (remplacer `jdoe` par votre nom, ex: `tp1_appert`), et adaptez le chemin de recherche¹.

N'oubliez pas que chaque fois que vous relancerez le client (par exemple lors de la prochaine séance de TP) le chemin de recherche est réinitialisé à sa valeur par défaut ("public"), et donc il vous faudra préciser sa valeur. Si vous avez respecté la convention de nommage ci-dessus cela ne devrait pas poser de problème, sinon, l'instruction `\dn` pourra vous servir à lister les schémas sous `psql`.

1.3 Création du schéma de la base et chargement de l'instance

1.3.1 Création d'une table, contraintes d'intégrité, mises à jours

Nous allons commencer par quelques manipulations sur une table toute simple. Vous trouverez ci-dessous les instructions dont vous avez besoin, mais vous pouvez jeter un bref coup d'oeil à la documentation de postgresql: <https://www.postgresql.org/docs/current/sql-altertable.html>

```
-- Instructions SQL pour les créer une table
CREATE TABLE nom_table(
attribut_1 type_1,
...
attribut_n type_n
);

/* et pour ajouter une contrainte d'intégrité nommée
(vous pouvez nommer la contrainte comme vous voulez mais la convention ci-dessous est pratique): */
ALTER TABLE nom_table ADD CONSTRAINT nomtable_pkey PRIMARY KEY (attribut);
ALTER TABLE nom_table ADD CONSTRAINT nomtable_attr_fkey FOREIGN KEY (attribut) REFERENCES table2(attr2);
```

- Créer une table `t` contenant deux attributs: `nom` de type text, et `age` de type int.
- Ajouter une contrainte à votre table, qui définit `nom` comme clé primaire de la table.
- Insérer dans la table les lignes:

```
Marie, 10
Henri, 5
```

¹cf transparents de cours, ou: <https://docs.postgresql.fr/current/ddl-schemas.html>

- Observer et comprendre ce qui se passe lorsque l'on insère successivement les lignes

```
Philippe, 5
Alix, 5
Marie, 4
Foy, 14
```




- Augmenter l'âge de Henri de 2, puis encore de 3 (donc de 5 au total). Avoir deux personnes ayant 10 ans pose-t-il problème ?
- Effacer de la table toutes les personnes ayant entre 6 ans (inclus) et 11ans (exclus).
- Créer une table **Filleul** comportant deux attributs: **filleul** et **parrain**. Ajouter une contrainte pour garantir que le filleul et le parrain sont bien des noms apparaissant dans la table *t*.
- Insérer dans cet ordre les lignes suivantes

```
Philippe, Foy
Alix, Foy
```

- Que se passe-t-il si l'on remplace (met à jour) dans la table **Filleul** le parrain d'Alix par "Alma"?

1.3.2 Chargement de la base de donnée du TP

Chargez la base de donnée : exécuter le script **base-northwind-postgresql.sql** ou bien adaptez le script **base-northwind-postgresql-from-csv.sql**. Puis découvrez depuis la base de donnée (i.e., pas depuis le script) le schéma de la base en utilisant les instructions idoines du client **psql**.

Il s'agit d'une base de donnée d'une entreprise fictive appelée *Northwind Traders*, où on retrouve les tables typiques de ce genre de bases de données: fournisseurs (suppliers), clients (customers), commandes (orders)... Cette base a été utilisée par Microsoft pour présenter les fonctionnalités de son SGBD **Access**. Le premier script est une suite d'instructions SQL de type **INSERT**, alors que le second charge plus efficacement les données depuis des fichiers .csv

2 Requêtes SQL

Exprimez en SQL les requêtes suivantes (on vous a fourni en général une illustration du résultat attendu):

1. Liste des employés travaillant au Royaume-Uni (UK) avec leur nom, prénom, et ville.

```
firstname | lastname | city
-----+-----+-----
Steven    | Buchanan | London
...
(4 rows)
```

2. Nombre d'employés

```
nb_employes
-----
          9
(1 row)
```

3. Liste des villes dans lesquelles travaille au moins un employé.

```
city
-----
London
Tacoma
Seattle
Redmond
Kirkland
(5 rows)
```

4. Cardinalité de la requête précédente et nombre de pays dans lesquels on trouve des employés.

```
nb_villes | nb_pays
-----+-----
          5 |          2
(1 row)
```

5. Nom du (ou des) fournisseur produisant de la Chartreuse verte.

```
company_name | product_name
-----+-----
Aux joyeux ecclésiastiques | Chartreuse verte
(1 row)
```

6. Liste des compagnies avec les catégories de produits qu'elles fournissent (on n'affichera pas les compagnies ne produisant aucun produit). Les paires seront triées à l'intérieur d'une même catégorie par ordre alphabétique de compagnie, et triées plus généralement par ordre alphabétique inverse de catégorie.

```
company_name | category_name
-----+-----
Escargots Nouveaux | Seafood
...
Aux joyeux ecclésiastiques | Beverages
...
Refrescos Americanas LTDA | Beverages
(49 rows)
```

7. Prix du produit le plus cher

```
prix
-----
263.50
(1 row)
```

8. Ajouter sur une première colonne le nom du produit le plus cher, et sur une seconde colonne le prix correspondant (vous avez le droit d'utiliser une sous requête). En cas d'ex-aequo votre requête affichera tous les produits les plus chers.

Ensuite, proposer une seconde façon de le calculer dans le cas où on sait que le produit le plus cher est unique (ou bien s'il n'est pas unique alors on acceptera n'importe quel choix parmi les réponses possibles), en utilisant la fonction **LIMIT** qui limite le résultat d'une requête à ses premiers résultats: **LIMIT 5** n'affichera par exemple que les 5 premières lignes du résultat.²


```
product_name | prix
-----+-----
...a vous de le découvrir
(1 row)
```

9. Liste des compagnies fournissant des fruits de mer ou des boissons, par ordre alphabétique.

```
company_name
-----
Aux joyeux ecclésiastiques
...
(15 rows)
```

10. Liste des compagnies fournissant des fruits de mer mais pas de boissons, par ordre alphabétique.

```
company_name
-----
Escargots Nouveaux
...
(7 rows)
```

11. Liste des compagnies fournissant des fruits de mer ( Seafood) et des boissons ( Beverage), par ordre alphabétique. Vous fournirez deux requêtes différentes.

```
company_name
-----
...
(1 row)
```

²En fait vous il y a beaucoup de façon de faire même sans utiliser **LIMIT**: vous pourriez utiliser différents types de sous-requêtes, voire même une différence ensembliste...

12. Liste des compagnies fournissant des fruits de mer et des boissons, par ordre alphabétique. Vous afficherez aussi le prix auquel on peut obtenir cette combinaison de produits (calculé à partir du prix unitaire de chaque).

```
company_name
-----
... le prix est 77.50
(1 row)
```

13. Montant total de l'ensemble des commandes (*Indice: il faut prendre en compte 3 attributs: le prix unitaire hors réduction, le pourcentage de réduction et la quantité commandée*)

```
montant
-----
1265793.0395
(1 row)
```

14. Nombre de commandes et leur montant total par pays de client, de fournisseur, et par catégorie.

```
customer_country | supplier_country | category | nb_commandes | montant
-----+-----+-----+-----+-----
Argentina      | Australia        | Confections | 1 | 104.7000
...
```

15. Nombre de clients ayant un fax.

```
nb
---
69
(1 row)
```

16. Nombre de clients ayant un fax, et nombre de clients n'ayant pas de fax, en indiquant "fax" et "pas de fax" selon le cas. Proposer 2 solutions: la première utilisant deux **SELECT**, la seconde n'en utilisant qu'un. Pour la seconde, vous aurez sans doute à utiliser un **CASE**.

```
fax | count
-----+-----
no fax | 22
fax | 69
(2 rows)
```

17. Liste des clients situés en Loire atlantique (ce département français a pour numéro 44) et ayant un fax.

```
contact_name
-----
Janine Labrune
Carine Schmitt
(2 rows)
```

18. Liste des employés ayant traité au moins 50 commandes. Attention, 2 employés peuvent avoir le même nom de famille, auquel cas ils doivent être tous deux représentés. Enfin, on affichera le résultat trié par ordre alphabétique sur les employés, mais lorsque 2 employés ont le même nom on les départagera en affichant d'abord celui qui a le plus de commandes.

```
lastname | nb
-----+-----
Callahan | 104
...
Suyama | 67
(7 rows)
```

19. Créer une table **orders2** contenant toutes les commandes (lignes de la table **orders**) pour lesquelles **order_id** est (strictement) inférieur à 10264.
20. Pour chaque employé, afficher son nom de famille suivi du nombre de commandes de la table **orders2** qu'il a traité. Attention, on souhaite afficher les employés n'ayant aucune commande (leur nombre de commande

sera donc 0). En outre il faut considérer que 2 employés peuvent avoir le même nom de famille, auquel cas ils doivent être tous deux représentés. Et le résultat devra être trié en affichant en premier les employés ayant traité le plus de commandes. Bien entendu vous ne devez pas utiliser la table **orders** mais seulement les tables **orders2** et **employees**. *Indication: a priori la seule solution que je considère élégante n'utilise qu'un **SELECT**. Je considérerais donc sans doute toute autre solution comme partiellement incorrecte.*

```
lastname | nb
-----+-----
Peacock  | 6
Leverling | 3
..;
Fuller   | 0
(9 rows)
```