TP noté: entrepôts de données

M1 ISD 2024-2025
Internet et toute forme de communication interdits (sauf cours).
Barême purement indicatif et susceptible de changer.
La note attribuée sera MIN(note obtenue, 20).

Instructions

Écrire vos réponses dans le fichier `m1isd-oct24-nom-prenom-reponses.sql`. Renommer le fichier en remplaçant "nom" et "prénom" par votre nom et votre prénom : si vous vous appelez Paul Michu, le fichier devra donc être nommé: `m1isd-oct24-michu-paul-reponses.sql`. À la fin du TP, déposez le fichier sur `https://link.infini.fr/dw1_isd`.

1 Création de la base de données

Pour créer et charger la base de donnée de ce TP, le script dump-zoo.sql vous est donné.
Exécuter les deux commandes suivantes ou passer par pgAdmin ou DBeaver comme vous préférez :

```
createdb -h localhost -U postgres zoo
psql -h localhost -U postgres -d zoo -f dump-zoo.sql
```

Ne modifiez pas les commentaires dans le script à rendre.

```
Vous devriez avoir chargé (INUTILE de tout vérifier):
100040 lignes dans person
10020 dans employee
...
```

2 Requêtes

- Écrire en SQL les requêtes ci-dessous **en respectant les noms de colonne** des illustrations.
- N'utilisez qu'un `SELECT` par requête: ceci vous interdit en particulier d'utiliser des sous-requêtes.

1. Lister les personnes, avec la ville et état où ils habitent, et le nombre de personnes habitant dans la même ville, et le nombre de personne habitant dans le même état. On affichera le résultat trié par état. On départagera les employés d'un même état par la ville, puis par leur nom. **(5pt)**

```
     fullname     |       city       |    us_state     | meme_ville | meme_etat
------------------+------------------+-----------------+------------+-----------
Karen Reed        | Birmingham       | Alabama         |          1 |         1
Lisa Carroll      | Mesa             | Arizona         |          1 |         2
Denise Simmons    | Phoenix          | Arizona         |          1 |         2
Wendell Gray      | Fresno           | California      |          1 |         2
Diana Robinson    | Sacramento       | California      |          1 |         2
Phyllis Watkins   | Colorado Springs | Colorado        |          1 |         1
...
(40 rows)
```

2. Pour chaque employé, afficher son nom (`employeefullname`), son type (`employeetype`) et sa date d'embauche (`hiredate`), ainsi qu'un numero indiquant son ancienneté comme suit: 1 si c'est le premier employé recruté, 2 si c'est le second, 3 pour le 3e, etc (si les 2 premiers employés sont recrutés en même temps, on affichera 1 pour eux, puis 2 pour les employés recrutés à la date suivante, 3 pour les suivants, etc). On affichera le résultat trié par employé. **(5pt)**

```
employeefullname | employeetype |  hiredate  | numero
-----------------+--------------+------------+--------
Brandon Perez    | Office       | 2013-05-10 |     18
Chris Turner     | Office       | 2000-05-24 |      9
Cornelius Adisa  | Zookeeper    | 2012-01-08 |     17
Denise Simmons   | Office       | 2016-04-12 |     19
...
(20 rows)
```

3. Comme précédent, mais ajouter, pour chaque employé:

   - une autre colonne c1 représentant le nombre d'employés du même type recrutés jusqu'à la date de recrutement de l'employé (si 4 employés de type "gardien de zoo" sont recrutés le premier jour, on affichera 4 pour ces employés).
   - (bonus 1pt:) et une dernière colonne indiquant le précédent employé recruté (en supposant qu'il n'y a pas d'ex-aequo).

   On affichera le résultat par ordre chronologique inverse d'embauche: les employés les plus récents en premier. **(5pt)**

```
 employeefullname | employeetype |  hiredate  | numero | c1 |       c2
------------------+--------------+------------+--------+----+-----------------
Lara Gall         | Office       | 2017-01-17 |     20 | 12 | Denise Simmons
Denise Simmons    | Office       | 2016-04-12 |     19 | 11 | Brandon Perez
Brandon Perez     | Office       | 2013-05-10 |     18 | 10 | Cornelius Adisa
Cornelius Adisa   | Zookeeper    | 2012-01-08 |     17 |  8 | Joseph Samadi
...
```

3 Optimisation d'une requête complexe avec des index (5pt)

Dans cet exercice, vous devez optimiser une requête SQL complexe en ajoutant un ou plusieurs index à une base de données PostgreSQL. Vous avez le choix des index à créer, et vous serez évalué sur l'amélioration des performances de la requête.

Instructions

1. Exécutez une requête SQL complexe fournie.
2. Identifiez et créez les index nécessaires pour optimiser cette requête.
3. Mesurez la performance avant et après la création des index et soumettez vos résultats.

Étapes

1. **Requête SQL complexe :**

   Exécutez la requête suivante et analysez son plan d'exécution avant d'ajouter des index :

   ```
   EXPLAIN ANALYZE
   SELECT p.fullname, e.employeefullname
   FROM person p
   JOIN employee e ON p.fullname = e.employeefullname
   WHERE p.city = 'New York' AND e.hiredate >= '2020-01-01';
   ```

2. **Création du ou des index :**

   Choisissez et créez un ou plusieurs index qui, selon vous, amélioreront la performance de la requête ci-dessus. Utilisez un B-tree et choisissez les colonnes à indexer.

3. **Mise à jour des statistiques :** Vous pouvez mettre à jour les statistiques de la base de données pour que le plan d'exécution soit recalculé :

   ```
   ANALYZE table_name;
   ```

4. **Mesure de la performance après optimisation :**

   Exécutez la requête à nouveau après avoir créé les index et comparez les temps d'exécution. Soumettez les plans d'exécution avant et après la création des index.

5. **Suppression des index :**

   Une fois l'optimisation terminée, vous pouvez supprimer le ou less index créés :

```
DROP INDEX idx_name;
```