TP SQL avancé

Pour chaque question, un aperçu du résultat désiré est représenté à la suite de la requête.

Quelques fonctions SQL pour Postgres:

Environnement

Les requêtes nécessitant des extensions du group by et la clause OVER() ne peuvent être formulées sur les versions trop ancienne de Postgres (ex: la version installée par défaut au département). Une solution est d'installer la dernière version de postgres, en utilisant Docker par exemple, comme indiqué dans les docs.

1 Premiers pas

Lab. Ex 2.1

Chargez les données en exécutant le script créant la base dans psql. Un schéma de la Base de donnée Northwind est fourni en Figure 1 ci-dessous. Vérifier depuis le client postgresql (en affichant la liste des tables et le schéma de quelques tables) que le schéma est bien correct. *Ce fichier est en fait un portage sous Oracle de la base "Northwind traders' conçue par Microsoft pour illustrer les possibilités d'Access.*

Lab. Ex 2.2

Quelle est la table de Faits? Y-a-t-il une dimension récursive (c'est-à-dire, qui se fait référence à elle-même)?

2 Extensions OLAP de la clause GROUP BY

Remarque: sauf mention contraire, "pays"... signifie "shipping country" (qui se trouve être identique à "customer country").

Lab. Ex 2.3

Rédiger des requêtes SQL pour calculer les expressions suivantes. Vous n'êtes pas autorisés à utiliser l'UNION de requêtes.

1. nombre de clients (customers) par pays.

```
   country    | nb
--------------+----
 Argentina    |  3
 Spain        |  5
 Switzerland  |  2
(21 rows)
```

2. nombres de commandes (orders) par pays, (pays et ville), et au total. Trier le résultat par ordre alphabétique sur pays puis ville.

```
SHIP_COUNTRY    SHIP_CITY          NBORDERS
--------------- --------------- ----------
Argentina       Buenos Aires            16
Argentina                               16
Austria         Graz                    30
Austria         Salzburg                10
Austria                                 40
Belgium         Bruxelles                7
...
(92 rows)
```

Ex1.3.2 (gauche)

```
C_COUNTRY       S_COUNTRY          QUANTITY    NBORDER
--------------- --------------- ---------- ----------
Argentina       Australia               13          2
Argentina       Canada                  10          1
Argentina       Denmark                  3          1
...
(308 rows)
```

Ex1.3.3 (dessus)

3. nombre de commandes et quantité d'éléments envoyés (selon la table `OrderDetails`) pour chaque paire (pays Client, pays Fournisseur (supplier)). Trier le résultat par pays du client d'abord, du fournisseur ensuite.

4. nombre de commandes à tous les niveaux de détail quand on s'intéresse seulement à l'origine géographique des clients et fournisseurs, au niveau des pays et du total seulement. (i.e., pareil que précédent mais avec en plus les totaux au niveau de chaque type de pays, et grand total).

```
C_COUNTRY       S_COUNTRY         QUANTITY    NBORDER
--------------- --------------- ---------- ----------
Argentina       Australia               13          2
...
Argentina                              339         16
...
                USA                   6828        244
                                     51317        830
(347 rows)
```

5. prix total (Quantity* UnitPrice * (1-Discount)) des commandes avec un fournisseur francais, pour chaque pays, region, et ville. Le pays doit être affiché chaque fois que la région l'est. On ne veut pas afficher le total. Proposer 2 solutions; s'appuyant sur une fonction différente pour étendre le GROUP BY.

```
SHIP_COUNTRY    SHIP_REGION     SHIP_CITY            PRICE
--------------- --------------- --------------- ----------
...
UK                              London                4452
UK                                                    4452
UK              Essex           Colchester             510
...
Switzerland                                          3458.4
...
(109 rows)
```

```
S_COUNTRY  S_CITY            NBORDERS
---------- --------------- ----------
Argentina  Buenos Aires            16
Argentina  whole country           16
Austria    Graz                    30
...
Venezuela  whole country           46
                                  830
(92 rows)
```

Ex1.3.5 (dessus) Ex1.3.6 (right)

6. modifier votre requête de question 2 afin d'afficher la chaine 'whole country' au lieu de NULL pour chaque ligne additionnant les nombres d'ordres de toutes les villes d'*un* pays.

3 Fenêtres de partitionnement

Lab. Ex 2.4

Ecrire des requêtes SQL calculant les expressions suivantes.

1. pour chaque commande, afficher son numéro (`order_id`), sa date (`order_date`), la ville d'expédition, le pays d'expédition, le poids de la commande (`freight`) le nombre de commandes expédiées dans cette ville, le nombre de commandes expédiées dans le pays, le poids total des commandes expédiées à ce pays, et le poids total des commandes effectuées ce jour là dans ce pays.

```
 order_id | order_date | ship_city | ship_country | freight | n_v | n_p | poids_p  | poids_p_j
----------+------------+-----------+--------------+---------+-----+-----+----------+-----------
    10825 | 1998-01-09 | Aachen    | Germany      |   79.25 |   6 | 122 | 11283.28 |     79.25
    10363 | 1996-11-26 | Aachen    | Germany      |   30.54 |   6 | 122 | 11283.28 |     30.54
...
(830 rows)
```

2. pour chaque commande, afficher sa date (`order_date`), la ville d'expédition, le pays d'expédition, le nombre de commandes expédiées dans cette ville jusqu'à cette date (incluse), le nombre de commandes expédiées dans ce pays jusqu'à cette date incluse, le nombre de commandes expédiées (n'importe où) jusqu'à ce jour (inclus) et le nombre de commandes expédiées dans ce pays le même jour.

```
 order_id | order_date |    ship_city    | ship_country | nc_v | nc_p | nc  | n_pj
----------+------------+-----------------+--------------+------+------+-----+------
    10248 | 1996-07-04 | Reims           | France       |    1 |    1 |   1 |    1
    10249 | 1996-07-05 | Munster         | Germany      |    1 |    1 |   2 |    1
    10250 | 1996-07-08 | Rio de Janeiro  | Brazil       |    1 |    1 |   4 |    1
...
(830 rows)
```

3. ajouter à la requête précédente le nombre de commandes expédiées dans le pays de la commande jusqu'à cette date exclue.

```
 order_id | order_date |    ship_city    | ship_country | nc_v | nc_p | nc  | n_pj | nc_p2
----------+------------+-----------------+--------------+------+------+-----+------+-------
    10248 | 1996-07-04 | Reims           | France       |    1 |    1 |   1 |    1 |     0
    10249 | 1996-07-05 | Munster         | Germany      |    1 |    1 |   2 |    1 |     0
...
(830 rows)
```

4. pour chaque commande, afficher le numéro de la commande précédente (chronologiquement), et le numéro de la commande précédente expédiée vers la même ville.

```
 order_id | num_p | num_p_v
----------+-------+---------
    10248 |       |
    10249 | 10248 |
    10250 | 10249 |
    10251 | 10250 |
    10252 | 10251 |
    10253 | 10252 |   10250
...
(830 rows)
```

5. nombres de commandes par pays et ville sur une colonne, ainsi que sur d'autres colonnes nombre total de commandes sur le pays et nombre maximal de commandes réalisées sur une ville de ce pays.

```
SHIP_COUNTRY    SHIP_CITY         NBORDERS   NBORDCTY NBORMAXCTY
--------------- --------------- ---------- ---------- ----------
Argentina       Buenos Aires            16         16         16
Austria         Graz                    30         40         30
Austria         Salzburg                10         40         30
...
(70 rows)
```

6. villes triées à l'intérieur d'un pays par ordre de commande, en affichant ce nombre de commandes et le rang. On ne veut pas sauter de valeurs pour le rang en cas d'ex-aequo.

```
SHIP_COUNTRY    SHIP_CITY         NBORDERS       RANK
--------------- --------------- ---------- ----------
Argentina       Buenos Aires            16          1
Austria         Salzburg                10          1
Austria         Graz                    30          2
Belgium         Bruxelles                7          1
Belgium         Charleroi               12          2
...
(70 rows)
```

7. ajouter à la requête précédente le pourcentage du nombre d'ordre réalisé par la ville à l'intérieur du pays.

```
SHIP_COUNTRY    SHIP_CITY         NBORDERS       RANK PERCENTG
--------------- --------------- ---------- ---------- --------
Argentina       Buenos Aires            16          1     1.00
Austria         Salzburg                10          1      .25
Austria         Graz                    30          2      .75
Belgium         Bruxelles                7          1      .37
...
(70 rows)
```

Ex. 2.4.7 (gauche)

```
  ORDER_ID      PRICE
---------- ----------
...
     11071      484.5
     11073        300
     11074    232.085
(427 rows)
```

Ex. 2.4.8 (droite)

8. prix total de chaque commande, en éliminant toutes les commandes dont le prix dépasse 110% du prix la précédente (OrderID) (vous avez le droit d'imbriquer des requêtes). En d'autres termes, on trie par OrderID croissant les commandes. Puis, si la commande d'OrderID 103 et de prix 24 est suivie de celle d'OrderID 205, on élimine du résultat la commande 205 si son prix est 26.2 ou plus.

9. produits les plus vendus par année, avec la quantité. Proposer une réponse utilisant une clause de fenêtrage (partition by) et une autre sans (vous pouvez imbriquer des requêtes). *Indication: syntaxe pour extraire l'année à partir d'une date dans postgresql :* `EXTRACT(YEAR FROM une_date)`

```
      YEAR PRODUCT_NAME                                QTITY
---------- ---------------------------------------- ----------
      1996 Gorgonzola Telino                               444
      1997 Gnocchi di nonna Alice                          971
      1998 Konbu                                           659
```

4 Fonctions de classement

Commencez par mettre à jour la base de données :

```
UPDATE orders
SET freight = CASE
   WHEN order_id = 10248 THEN 10000
   WHEN order_id = 10249 THEN 10000
   WHEN order_id = 10250 THEN 8000
   ELSE freight
END
WHERE order_id IN (10248, 10249, 10250);
```

Une fois cet exercice **terminé**, remettez les valeurs à leurs valeurs initiales :

```
UPDATE orders
SET freight = CASE
   WHEN order_id = 10248 THEN 32.38
   WHEN order_id = 10249 THEN 11.61
   WHEN order_id = 10250 THEN 65.83
   ELSE freight
END
WHERE order_id IN (10248, 10249, 10250);
```

Rédiger des requêtes SQL pour calculer les expressions suivantes.

1. classez les commandes par montant de frais de transport avec des ex-aequo.

```
order_id   order_date  freight  rank
-------- ------------ -------- -----
10249      1996-07-05    10000     1
10248      1996-07-04    10000     1
10250      1996-07-08     8000     3
...
(830 rows)
```

Ex. 4.1 (gauche)

```
order_id   order_date  freight  rank
-------- ------------ -------- -----
10249      1996-07-05    10000     1
10248      1996-07-04    10000     1
10250      1996-07-08     8000     2
...
(830 rows)
```

Ex. 4.2 (droite)

2. classez les commandes par montant de frais de transport sans sauter de position en cas d'ex-aequo.
3. numérotez chaque commande en fonction du montant de frais de transport sans tenir compte des ex-aequo.

```
order_id   order_date  freight  row_number
-------- ------------ -------- ----------
10249      1996-07-05    10000          1
10248      1996-07-04    10000          2
10250      1996-07-08     8000          3
...
(830 rows)
```

Ex. 4.3 (gauche)

```
order_id   order_date  freight  tile
-------- ------------ -------- -----
10248      1996-07-04    10000     1
10249      1996-07-05    10000     1
10250      1996-07-08    41.34     1
...
10514     1997-04-22  789.95      2
...
10972     1998-03-24    0.02    120
(830 rows)
```

Ex. 4.4 (droite)

4. divisez les commandes en 120 segments égaux, classés par montant de frais de transport.
5. affichez les frais de transport de la deuxième commande suivante.

```
order_id   order_date  freight  next_freight
-------- ------------ -------- -------------
10248      1996-07-04    10000         41.34
10249      1996-07-05    10000          8000
10250      1996-07-08     8000         51.30
...
(830 rows)
```

Ex. 4.5 (gauche)

```
order_id   order_date  freight  previous_freight
-------- ------------ -------- -----------------
10248      1996-07-04    10000                 0
10249      1996-07-05    10000             10000
10250      1996-07-08    41.34             10000
...
(830 rows)
```

Ex. 4.6 (droite)

6. comparez les frais de transport de chaque commande avec ceux de la commande précédente.
7. retourne les frais de transport de la première commande dans la liste triée par date.

```
order_id   order_date  freight  first_freight
-------- ------------ -------- --------------
10248      1996-07-04    10000          10000
10249      1996-07-05    10000          10000
10250      1996-07-08    41.34          10000
...
(830 rows)
```

Ex. 4.7 (gauche)

```
order_id   order_date  freight  last_freight
-------- ------------ -------- -------------
10248      1996-07-04    10000          6.19
10249      1996-07-05    10000          6.19
10250      1996-07-08    41.34          6.19
...
(830 rows)
```

Ex. 4.8 (droite)

8. affichez les frais de transport de la dernière commande dans la fenêtre

5 Requêtes récursives

Lab. Ex 2.5
Utiliser une requête récursive pour créer une table listant les entiers de 1 à 60.

Lab. Ex 2.6 (Requêtes récursives: hiérarchies)

Ecrire une requête récursive, basée sur un CTE (Common Table Expression), qui affiche pour chaque employé:

- pour chaque employé, indenter l'employé en fonction de son niveau hiérarchique (ex: le grand patron n'a aucune indentation, un employé à distance 2 du grand patron aura une indentation de 4 espaces, etc.)
- pour chaque employé, un attribut de type textuel affichera son chemin jusqu'au grand patron
- les employés sont affichés selon l'ordre préfixe (ordre d'apparition dans un parcours en profondeur).

6 Création d'un schéma ROLAP

Vous faites partie du département BI de PSeezer, une entreprise de streaming musical. Votre objectif est de modéliser un schéma ROLAP en flocon de neige à partir des données opérationnelles fournies. Ce schéma devra répondre aux besoins d'analyse de l'entreprise pour comprendre les tendances d'écoute des utilisateurs et les performances des artistes.

Sources de données :

1. **Données des écoutes** : Historique des morceaux joués par les utilisateurs (qui écoutent quoi, quand, et comment).
2. **Données des albums et artistes** : Informations sur les albums, les morceaux, les artistes, et les contrats avec les labels.

Objectifs :

Votre mission est de créer un schéma ROLAP permettant de répondre aux questions suivantes :

1. Quels sont les artistes les plus écoutés par genre dans chaque pays ?
2. Quelles sont les périodes de l'année où les utilisateurs premium écoutent le plus de musique ?
3. Quels albums sont les plus populaires dans différentes régions géographiques ?
4. Quels dispositifs (mobile, ordinateur, etc.) sont les plus utilisés pour écouter de la musique par tranche d'âge ?

Étapes :

1. **Identifiez les faits et mesures :** Déterminez les mesures principales à analyser. Par exemple, le nombre de streams (écoutes) pourrait être une mesure centrale.
2. **Créez les dimensions nécessaires :** Pensez aux axes d'analyse que l'entreprise pourrait utiliser, comme :
   - Le **temps** (année, trimestre, mois, jour),

- La **localisation** (continent, pays, ville),
- Les **utilisateurs** (âge, genre, type d'abonnement),
- Les **artistes** (genre musical, label),
- Les **albums et morceaux** (durée, popularité),
- Le **dispositif** utilisé pour écouter la musique (mobile, ordinateur).

3. **Organisez vos tables de dimension en flocon de neige :** Certaines dimensions peuvent être organisées de façon hiérarchique (par exemple, Ville → Pays → Continent pour la localisation, ou Artiste → Genre → Label).

4. **Reliez les dimensions à une table de faits :** Une table de faits centralise les mesures (ex : `streams_count`) et relie les différentes dimensions via des clés étrangères.

Tips :

- **Pensez à la hiérarchie** : L'une des dimensions doit avoir une hiérarchie en flocon de neige, comme la localisation (continent → pays → ville). Identifiez les autres dimensions qui pourraient aussi bénéficier d'une structure hiérarchique.

- **Choisissez bien vos mesures :** Quels sont les indicateurs de performance les plus importants pour l'entreprise ? (ex : nombre d'écoutes, popularité des morceaux).

- **Regardez les types de données disponibles :** Pensez aux caractéristiques des utilisateurs, des morceaux, et des artistes. Vous devrez peut-être ajouter des tables pour gérer ces informations.

- **Gardez la simplicité :** N'essayez pas de faire trop complexe, mais assurez-vous que votre schéma permet de répondre aux questions analytiques posées.

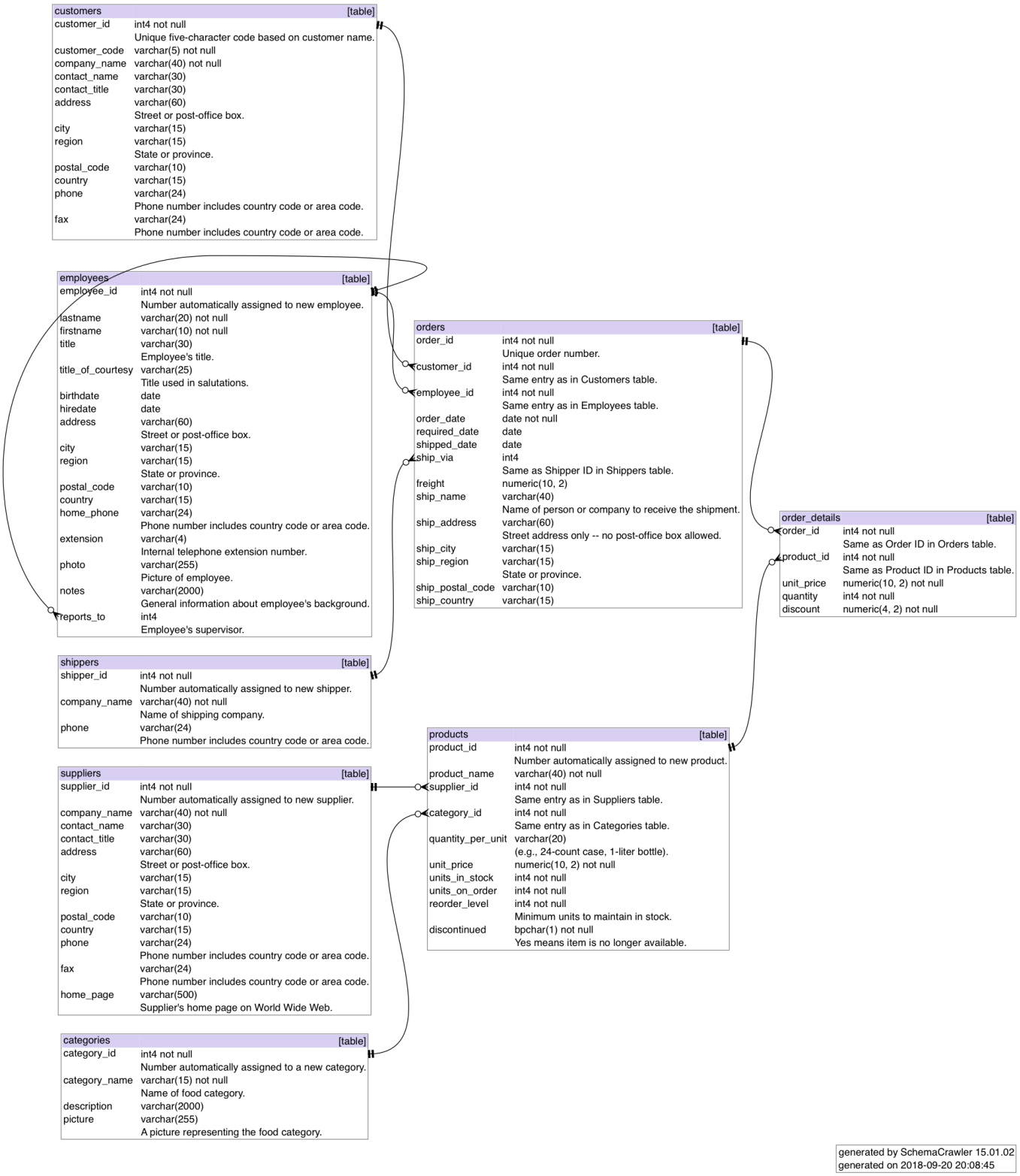

Figure 1: Schéma (approximatif) des relations de la base de donnée Northwind