TP 5 — Agrégation et groupements de données

Au TP 3, nous avons vu comment utiliser l'instruction `SELECT` en combinaison avec la clause `WHERE` pour filtrer selon certaines conditions. Dans ce TP, nous allons voir qu'il est possible d'aller beaucoup plus loin et d'écrire des requêtes permettant de trier les données selon divers critères appliqués aux attributs (par ordre chronologique, par ordre alphabétique inversé, par ordre croissant, etc.). Lorsque ces opérations ont un sens, il est également possible de calculer des moyennes, des maxima, ou de regrouper ensemble des données qui vérifient une condition commune.

Nous allons de nouveau travailler avec la base de données `Northwind`. La structure des requêtes sera toujours la même :

```SQL
SELECT [ALL | DISTINCT] liste_de_selection
FROM liste_de_tables
[WHERE conditions] -- Condition sur le WHERE
[GROUP BY attribut -- Pour grouper les attributs
[HAVING condition]] -- Condition portant sur l'attribut du GROUP BY
[ORDER BY liste_criteres_de_tri [ASC | DESC]];
```

Notations utilisées : grammaire de SQL

Le code présenté ci-desus utilise les symbole `[]` et `|`. Ce ne sont pas des caractères de SQL, mais c'est la notation standard pour décrire la *grammaire* d'un langage de programmation, c'est-à-dire ses *règles d'écriture* :

- `[a]` signifie que `a` est une option, non obligatoire ;
- `a | b` signifie qu'il est possible d'utiliser soit `a`, soit `b` (mais pas les deux en même temps).

La grammaire permet de décrire de façon abstraite, à la fois synthétique et précise, les règles d'écriture d'une requête. On en déduit que :

- la directive `SELECT` est obligatoire et suivie d'une liste d'attributs. Elle peut (facultativement) être précisée avec un des mots-clés parmi `ALL` ou `DISTINCT` ;
- la clause `WHERE` est facultative, mais si elle est présente elle est obligatoirement suivie d'une expression conditionnelle ;
- la directive `GROUP BY` est facultative, et peut (mais ce n'est pas obligatoire) être précisée avec une clause `HAVING` ;
- la directive `ORDER BY` est facultative, s'accompagne d'une liste d'attributs précisant l'ordre du tri, et éventuellement du sens (croissant ou décroissant).

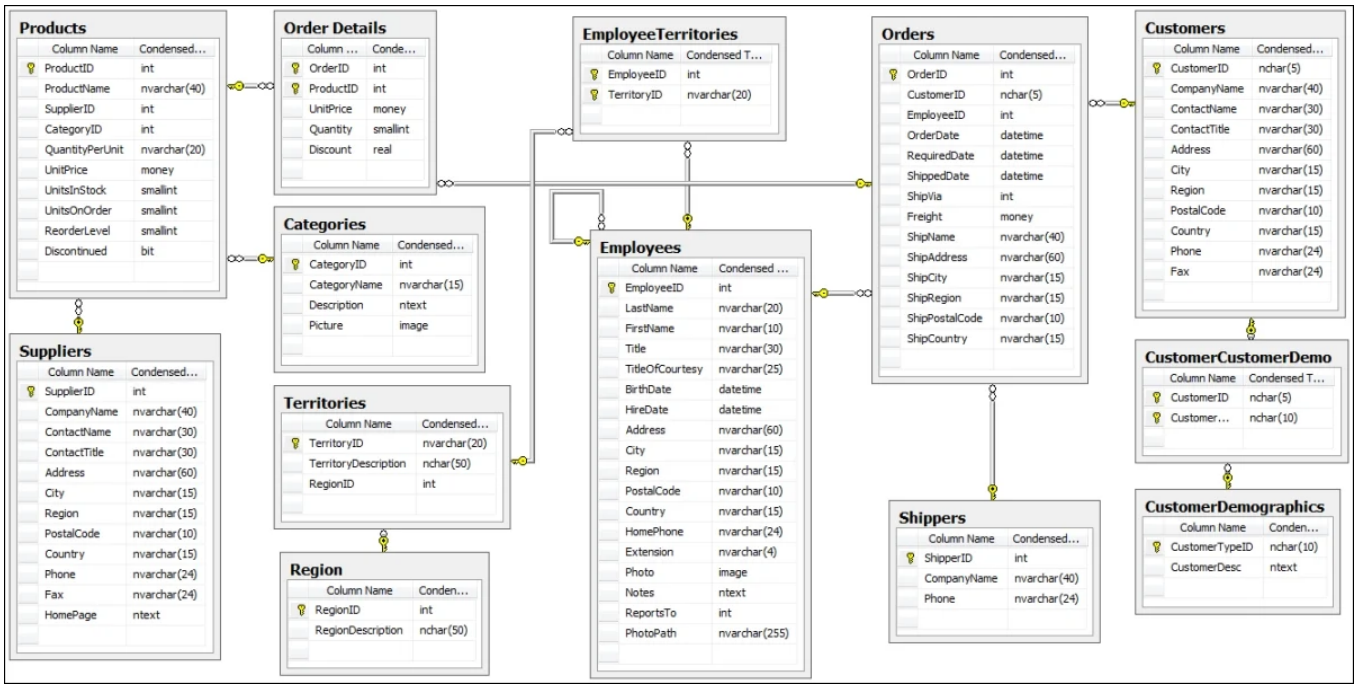
1 Mise en place

Comem d'habitude, créez un dossier `R105_BDD▸TP4`. Copiez-y la base de données `northwind.sqlite3` utilisée lors du TP précédent, et créez le fichier `r105_tp4_<nomDeFamille>.sql` depuis DBeaver.

À rendre

Vous devrez déposer sur EPREL votre fichier `r105_tp4_<nomDeFamille>.sql`, contenant l'ensemble des requêtes écrites pendant ce TP.

On vous redonne ici le diagramme entité-relations de cette base :



2 Exercices

1) Écrire une requête qui renvoie la liste des commandes et le *nombre* total d'articles par commande (et pas la *liste* des articles dans les commandes). N'oubliez pas de prendre en compte le nombre d'exemplaires commandés par article. On rappelle que la liste des produits est contenue dans la table `Products`, les commandes sont dans la table `Orders`. Et la table `OrderDetails` permet d'obtenir la liaison entre les deux (cf. TP précédent pour plus de détails).

2) Écrire une requête qui donne le nombre de client·es pour chaque titre de contact (attribut `ContactTitle` dans la table `Customers`.

3) Trouver le nombre total de client·es par pays et par ville.

4) Afficher toutes les commandes d'une valeur supérieure ou égale à 7 000 $.

5) Trouvez les cinq premiers pays dont les frais de transport moyens sont les plus élevés au cours des 24 derniers mois.