TD Informatique TSI2

DATABASE : SQL (Structured Query Language)

PARTIE II : ALL PARAGRAPHES language SQL

SELECT FROM, WHERE, AND OR, AS, MAX, COUNT, ORDER BY, JOIN, ON, GROUP BY, HAVING, DISTINCT

SOLUTION

EXERCICE 01

On considère le schéma relationnel suivant.

Relation 1 : **Frais (id, nom, prix, #id_marque)** avec la clé primaire **id**. Les champs **id** et **id_marque** sont de type entier, le champ nom est de type chaîne de caractères, le champ **prix** est de type flottant. Le prix est en euro.

Relation 2 : **Marque(id, nom)** avec la clé primaire **id**. Le champ **id** est de type entier, le champ **nom** est de type **chaîne de caractères**. Le champ **id_marque** de la table **Frais** est une clé étrangère en relation avec la clé primaire **id** de la relation **Marque**.

Écrivons les requêtes suivantes en SQL :

1. le nom de tous les produits frais qui ont un prix strictement inférieur à 10 euros ?

SELECT nom FROM Frais WHERE prix < 10 ;

2. le prix moyen du lait en format 1 litre. On suppose que dans la table **Frais**, on trouve ce produit avec le nom **lait1L** ?

SELECT AVG(prix) FROM Frais WHERE nom ="lait1L" ;

3. les noms de toutes les marques proposant du lait au format 1 litre et le prix du produit pour chaque marque ?

SELECT Marque.nom FROM Frais JOIN Marque ON id_marque = Marque.id
WHERE Frais.nom ="lait1L" ;

4. les noms des marques qui proposent du lait en format 1 litre au prix le plus bas. Il peut y avoir plusieurs marques dans le résultat ?

On reprend la requête précédente dans laquelle on rajoute dans **WHERE** à la condition **Frais = "lait1L"** celle que **prix** est le prix minimum pour la lait de un litre.
Cette requête supplémentaire est **SELECT MIN(prix) MIN FROM Frais WHERE nom ="lait1L" ;**

On obtient :
SELECT Marque.nom FROM Frais JOIN Marque ON id_marque = Marque.id
WHERE Frais.nom ="lait1L"
AND prix = (SELECT MIN(prix) MIN FROM Frais WHERE nom ="lait1L") ;

EXERCICE 02

Gestion d'une librairie. Part II

On reprend l'énoncé de l'exercice 04 de la partie I. Rappelons la situation.
Une librairie est gérée à l'aide d'une base de données. Le modèle relationnel contient les cinq relations décrites ci-dessous avec leur schéma :

Livre (Id, Titre, PrixHT, Année, #Id_genre, #Id_editeur

Auteur (Id, Nom, Prénom) Écrit (#Id_auteur, #Id_titre)

Genre (Id, Nom) Éditeur (Id, Nom)

Les champs **Id**, **Id_auteur**, **Id_titre** sont des clés primaires et tous les champs du genre **Id_NomTable** sont des clés étrangères. Plus précisément :
le champ **Id_editeur** est une clé étrangère en référence à la clé primaire **Id** de la table **Éditeur**
Le champ **Id_genre** est une clé étrangère en référence à la clé primaire **Id** de la table **Genre**
Le champ **Id_titre** est une clé étrangère en référence à la clé primaire **Id** de la table **Livres**
Le champ **Id_auteur** est une clé étrangère en référence à la clé primaire **Id** de la table **Auteur**
Tous les champs **Id** ou commençant par **Id** et le champ **Année** sont du type entier. Le champ **PrixHT** est de type flottant et les autres champs sont de type chaîne de caractères.

Écrire les requêtes suivantes en SQL :

1. Les titres des livres avec le nom de leur éditeur.

SELECT Titre, Nom FROM Livre JOIN Éditeur ON Id_éditeur = Éditeur.Id ;

2. Les titres des livres du genre **"science"**

On renommera la table **Genre** en table **g** pour pouvoir abréger **Genre.Id** en **g.Id** et on gardera cet alias pour la suite.

SELECT Titre FROM Livre JOIN Genre AS g ON Id_genre = g.Id
WHERE Nom = "science" ;

3. Les titres et les prix des livres du genre **"policier"** coûtant moins de 20 euros HT.

SELECT Titre, PrixHT FROM Livre JOIN Genre AS g ON Id_genre = g.Id
WHERE Nom = "policier" AND PrixHT < 20 ;

4. Les années de parution de livre dans le genre **"science"** sans doublons

SELECT DISTINCT Année FROM Livre JOIN Genre AS g ON Id_genre = g.Id
WHERE Nom ="science" ;

5. Le prix total de tous les livres parus en 2021 dans le genre **"science"**

SELECT SUM(PrixHT) FROM Livre JOIN Genre AS g ON Id_genre = g.Id
WHERE Nom ="science" AND Année = 2021 ;

6. Les identifiants des livres dont l'auteur a pour prénom Walter

SELECT id_titre FROM Écrit JOIN Auteur ON Id_auteur = Auteur.Id
WHERE prénom ="Walter" ;

7. La liste des livres portant le même titre.

SELECT Titre FROM Livre GROUP BY Titre HAVING COUNT(*) > 1 ;

EXERCICE 03

On dispose d'une base de données **world** constituée de trois tables.

Le schéma relationnel est le suivant :

city (Id, Name, #CountryCode, Population)

country (Code, Name, Continent, SurfaceArea, IndepYear, Population, LifeExpancety, GNP, #Capital)

countrylanguage(Id, #CountryCode, Language, IsOfficial, Percentage)

Les clés primaires sont les champs nommés **Id** pour les tables **city** et **countrylanguage** et **Code** pour la table **country**.
Les champs nommés **CountryCode** sont des clés étrangères en références à la clé primaire **Id** de la table **city**

Une ligne de la table **city** est par exemple :
(1, Kabul, AFG, 1 780 000) avec les types (entier, chaîne, chaîne, entier)

Une ligne de la table **country** est par exemple :
(AFG, Afghanistan, Asia, 652090.00, 1919, 22 720 000, 45.9, 5976.00,1) avec les types (chaîne, chaîne, chaîne, flottant, entier, entier, flottant, flottant, entier).

Une ligne de la table **countrylanguage** est par exemple :
(117, AFG, Pashto, T, 52.4) avec les types (entier, chaîne, chaîne, chaîne, flottant), la lettre **T** signifie **True** et la lettre **F** signifie **False**

Enfin **LifeExpectancy** signifie espérance de vie. Quant à **GNP** cela signifie le produit intérieur brut.

Écrivons les requêtes suivantes en SQL.

1. La superficie et la population de la France

SELECT SurfaceArea, Population FROM country WHERE Name = "France" ;

2. La liste des continents

SELECT DISTINCT continent FROM country ;

On met **DISTINCT** pour ne pas repeter les continents plein de fois.

3. Les villes dont la population est supérieure à six millions d'habitants, rangée de la plus peuplée à la moins peuplée

SELECT Name FROM City WHERE Population > 6 000 000
ORDER BY Population DESC ;

4. Les pays avec le continent et le GNP (PNB) où l'espérance de vie est supérieure ou égal à 80 ans

SELECT Name, Continent, GNP FROM country WHERE LifeExpectancy >= 80 ;

5. Le nombre de pays dans le continent Asie

SELECT COUNT(*) AS "Nombre_pays_Asie FROM country
WHERE Continent = "Asia" ;

6. Le nombre de pays européens dont la population est supérieure à 30 millions d'habitants

SELECT COUNT(*) FROM country
WHERE Continent = "Europe" AND Population > 30 000 000 ;

7. La capitale du Portugal

SELECT city.Name FROM city JOIN country ON Id.city = Capital
WHERE country.Name ="Portugal" ;

8. Les langues parlées au Vietnam

SELECT Language FROM countrylanguage JOIN country ON CountryCode = Code.country
WHERE Name = "Vietnam" ;

9. La langue officielle parlée au Vietnam

SELECT Language FROM countrylanguange JOIN country ON CountryCode = Code.country
WHERE Name = "Vietnam" AND IsOfficial = "T" ;

10. Les noms des villes d'Océanie avec le pays correspondant

SELECT city.Name, country.Name FROM country JOIN city
ON Code.country = CountryCode WHERE Continent = "Asia" ;

11. Les noms des villes qui sont aussi des noms de pays

SELECT Name.city FROM city INTERSECT SELECT Name.country FROM country ;

12. Le nombre de pays dont la langue officielle est le francais

SELECT COUNT(*) FROM countrylanguage
WHERE Language = "French" AND IsOfficial ="T" ;

13. Les pays avec l'espérance de vie et le GNP par habitant, triés suivant l'espérance de vie par ordre décroissant. On pourra renommer le champ **GNP/Population** par **"PNB par habitant"**

SELECT Name, LifeExpectancy, GNP / Population AS "PNB par habitant"
FROM country ORDER BY LifeExpectancy DESC ;

14. Les pays indépendants depuis 1970 triés suivant l'année d'indépendance par ordre décroissant

SELECT Name FROM country WHERE IndepYear > 1970 ORDER BY IndepYear DESC ;

15. Les pays européens où une partie de la population parle anglais avec le pourcentage de la population parlant anglais

SELECT Name, Percentage FROM country JOIN countrylanguage
ON Code.country = CountryCode WHERE Continent = "Europe"
AND Language = "Englisch" ;

16. La liste des pays européens avec leur densité de population rangés dans l'ordre décroissant des densités de population ; La densité de population **Population / SurfaceArea** pourra être renommé **"Densité"**

SELECT Name, Population / SurfaceArea AS "Densité" FROM country
WHERE Continent = "Europe" ORDER BY "Densité" DESC ;

17. Les villes d'Afrique qui ne sont pas des capitales avec le pays

SELECT city.Name, country.Name FROM country JOIN city
ON city.Code = CountryCode WHERE Continent ='Africa' AND city.Id != Capital ;

Remarque :

Gross national product (GNP) is an estimate of the total value of all the final products and services turned out in a given period by the means of production owned by a country's residents. GNP is commonly calculated by taking the sum of personal consumption expenditures, private domestic investment, government expenditure, net exports, and any income earned by residents from overseas investments, then subtracting income earned by foreign residents. Net exports represent the difference between what a country exports minus any imports of goods and services.
Ce qui donne en francais :
Le produit national brut (PNB) est une estimation de la valeur totale de tous les produits et services finaux produits au cours dâune période donnée par les moyens de production détenus par les résidents dâun pays. Le PNB est généralement calculé en prenant la somme des dépenses de consommation personnelle, de lâinvestissement intérieur privé, des dépenses publiques, des exportations nettes et de tout revenu tiré par les résidents dâinvestissements à lâétranger, puis en soustrayant le revenu gagné par les résidents étrangers. Les exportations nettes représentent la différence entre ce quâun pays exporte moins les importations de biens et de services.

EXERCICE 04

Extrait d'un sujet de concours d'informatique tronc commun

1. L'attribut **nom** ne peut être clé primaire de la table **fonction** car les 3 premiers enregistrements de l'exemple de table **fonction** possède le même attribut **nom** (ici **rectangles)** alors que les enregistrements sont distincts.

Q2-a Il suffit d'utiliser les fonctions d'agrégation **COUNT** et **AVG** sur les attributs souhaités.

SELECT COUNT(nom), AVG(ram) FROM ordinateurs ;

2-b On commence par récupérer les noms des ordinateurs ayant utilisé l'algorithme **rectangle** et la fonction *li*. Pour cela, on croise les tables **ordinateur** et **fonction** selon la condition d'égalité des attributs **nom** et **teste_sur** et on projette selon l'attribut **nom** (dans la table **ordinateurs**) sous condition que l'algorithme **rectangle** et la fonction *li* aient été choisis.

SELECT ordinateurs.nom FROM ordinateurs JOIN fonctions
ON ordinateurs.nom = fonctions.teste_sur
WHERE fonctions.algorithme ="rectangles" AND fonctions.nom = "li" ;

On peut faire une variante en renommant les tables.

SELECT o.nom FROM ordinateurs AS o JOIN fonctions AS f
ON o.nom = f.teste_sur WHERE f.algorithme ="rectangles" AND f.nom = "li" ;

Il suffit alors de créer la table des noms extrait de la table **ordinateurs** et de lui enlever tous les enregistrements présents dans la table créé ci-dessus grâce à la commande **EXCEPT**

SELECT o.nom FROM ordinateurs EXCEPT
SELECT o.nom FROM ordinateurs AS o JOIN fonctions AS f
ON o.nom = f.teste_sur WHERE f.algorithme ="rectangles" and f.nom = "li" ;

2-c On croise les tables **ordinateurs** et **fonctions** selon la condition d'égalité des attributs **nom** et **teste_sur**, on ne conserve que les enregistrements associé àÂ la fonction *Ei*, on projette selon les attributs **algorithme**, nom du pc et **gflops** puis on trie le tout (grâce àÂ la commande **ORDER BY**) par ordre décroissant du temps d'exécution (grâce à la commande **DESC**).

SELECT f.algorithme, o.nom, o.gflops FROM ordinateurs AS o JOIN fonctions AS f
ON o.nom = f.teste_sur WHERE f.nom = "Ei" ORDER BY f.temps_exec DESC ;

Remarque 1 : La première requête est simple et bien réussie par la plupart des candidats.
La deuxième requête est sélective car elle est bien plus complexe et requiert soit la connaissance de la commande **EXCEPT**, soit une bonne maitrise des jointures.
La troisième requête est discriminante mais elle est classique et vous l'avez probablement rencontré à plusieurs reprises durant ce TD notamment. (À connaître donc).

Remarque 2 : Quel est l'utilité de la fonction li introduite plus haut ?
La question de la répartition des nombres premiers a été étudiée notamment par Euclide, Riemann, Gauβ et Legendre. On étudie dans cette partie les propriétés de la fonction $\pi(n)$ qui renvoie le nombre de nombres premiers de $[\![1, n]\!]$.

Il a été prouvé que $\dfrac{n}{\ln(n)-1} < \pi(n)$ pour tout $n \geqslant 5393$. Le calcul de $\pi(n)$ dépend de la capacité à calculer de manière exhaustive tous les nombres premiers de $[\![1, N]\!]$. Or le temps nécessaire à ce calcul devient rapidement très grand lorsque N augmente. Il existe en revanche diverses méthodes pour calculer une valeur approchée de $\pi(n)$. Une méthode utilise la fonction intégral li :

$$li : \mathbb{R}_+ \setminus \{1\} \to \mathbb{R},\ x \mapsto \int_0^x \frac{dt}{\ln(t)}.$$

Attention, si x franchit 1, donc si $x > 1$, l'intégrale généralisée $li(x)$ doit être interprétée comme sa valeur principale de Cauchy qui est définie comme :

$$li(x) = \lim_{\epsilon \to 0^+} \left(\int_0^{1-\epsilon} \frac{dt}{\ln(t)} + \int_{1+\epsilon}^{x} \frac{dt}{\ln(t)} \right) \quad (3)$$

L'intérêt de li pour compter les nombres premiers vient de la formule suivante :

$$\lim_{x \to +\infty} \frac{\pi(\lfloor x \rfloor)}{li(x)} = 1.$$

On souhaite développer un programme permettant de calculer une valeur approchée de li. On compare ensuite les résultats obtenus à une implémentation de référence qui est nommée ref_li, réputée très précise. Et on fait une estimation de li par quadrature numérique d'où plus haut.