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, INTERSECT, UNION, EXCEPT

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**.

Écrire les requêtes suivantes en SQL :

1. le nom de tous les produits frais qui ont un prix strictement inférieur à 10 euros ;
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** ;
3. les noms de toutes les marques proposant du lait au format 1 litre et le prix du produit pour chaque marque ;
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.

Indications *: donnons les instructions et les clauses qu'on utilise à chaque question dans l'ordre d'écriture*
Pour **1***, on utilise* **SELECT, FROM, WHERE**
Pour **2***, on utilise* **SELECT, AVG, FROM, WHERE**
Pour **3***, on utilise* **SELECT, FROM, JOIN, ON, WHERE**
Pour **4***, on utilise* **SELECT, FROM, JOIN, ON, WHERE, AND** *et dans* **AND** *on a une requête SQL imbriquée qui vaut* **prix** *dans laquelle, on a* **SELECT, MIN, FROM, WHERE**.

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.

T.S.V.P →

Écrire les requêtes suivantes en SQL :

1. les titres des livres avec le nom de leur éditeur ;
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 ;
3. les titres et les prix des livres du genre **"policier"** coûtant moins de 20 euros HT ;
4. les années de parution de livre dans le genre **"science"** sans doublons ;
5. le prix total de tous les livres parus en 2021 dans le genre **"science"** ;
6. les identifiants des livres dont l'auteur a pour prénom Walter ;
7. la listes des livres portant le même titre.

Indications *: donnons les instructions et les clauses qu'on utilise à chaque question dans l'ordre d'écriture*
Pour **1**, *on utilise* **SELECT, FROM, JOIN, ON**
Pour **2**, *on utilise* **SELECT, FROM, AS, JOIN, ON, WHERE**
Pour **3**, *on utilise* **SELECT, FROM, JOIN, ON, WHERE, AND**
Pour **4**, *on utilise* **SELECT, DISTINCT, FROM, JOIN, ON, WHERE**
Pour **5**, *on utilise* **SELECT, SUM , FROM, JOIN, ON, WHERE, AND**
Pour **6**, *on utilise* **SELECT, FROM, JOIN, ON, WHERE**
Pour **7**, *on utilise* **SELECT, FROM, GROUP, BY, HAVING COUNT(*)**

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.

Écrire les requêtes suivantes en SQL :

1. la superficie et la population de la France ;
2. la liste des continents ;
3. les villes dont la population est supérieure à six millions d'habitants, rangée de la plus peuplée à la moins peuplée ;
4. les pays avec le continent et le GNP (PNB) où l'espérance de vie est supérieure ou égal à 80 ans ;
5. le nombre de pays dans le continent Asie. On appelle **"Nb_Pays_Asie"** le résultat ;
6. le nombre de pays européens dont la population est supérieure à 30 millions d'habitants ;
7. la capitale du Portugal ;
8. les langues parlées au Vietnam ;
9. la langue officielle parlée au Vietnam ;

10. les noms des villes d'Océanie avec le pays correspondant ;
11. les noms des villes qui sont aussi des noms de pays ;
12. le nombre de pays dont la langue officielle est le francais ;
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"** ;
14. les pays indépendants depuis 1970 triés suivant l'année d'indépendance par ordre décroissant ;
15. les pays européens où une partie de la population parle anglais avec le pourcentage de la population parlant anglais ;
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"** ;
17. les villes d'Afrique avec leurs pays respectifs qui ne sont pas des capitales avec le pays.

Indications *: donnons les instructions et les clauses qu'on utilise à chaque question dans l'ordre d'écriture*
Pour **1**, **4**, *on utilise* **SELECT, FROM, WHERE**
Pour **2**, *on utilise* **SELECT, DISTINCT, FROM**
Pour **3**, *on utilise* **SELECT, FROM, WHERE, ORDER BY, DESC**
Pour **5**, *on utilise* **SELECT, COUNT, AS, FROM, WHERE**
Pour **6**, *on utilise* **SELECT, COUNT, FROM, WHERE, AND**
Pour **7**, *pour* **8**, *pour* **10**, *on utilise* **SELECT, FROM, JOIN, ON, WHERE**
Pour **9**, *pour* **17**, *on utilise* **SELECT, FROM, JOIN, ON, WHERE, AND**
Pour **11**, *on utilise* **SELECT, FROM, INTERSECT, SELECT, FROM**
Pour **12**, *on utilise* **SELECT, COUNT, FROM, WHERE, AND**
Pour **13**, *on utilise* **SELECT, /, AS, FROM, ORDER BY, DESC**
Pour **14**, *on utilise* **SELECT, FROM, WHERE, ORDER BY, DESC**
Pour **15**, *on utilise* **SELECT, FROM, JOIN, ON, WHERE, AND**
Pour **16**, *on utilise* **SELECT, /, AS, FROM, WHERE, ORDER BY, DESC**

EXERCICE 04

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

On souhaite mener des tests à grande échelle pour évaluer les performances réelles de codes qu'un programmateur humain peut développer.
Donnons un exemple. Soit la fonction intégral li :

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

On construit des algorithmes de calcul d'intégration numérique avec des codes Python pour approcher les valeurs de li pour certains x.
Puis on effectue un grand nombre de tests sur une multitude d'ordinateurs. Les données sont ensuite centralisées dans une base de données composée de deux tables.

La première table est **ordinateurs** et permet de stocker des informations utilisés pour les tests.
Ses attributs sont :
• **nom**, clé primaire, le nom de l'ordinateur, de type entier.
• **gflops** la puissance de l'ordinateur en milliards d'opérations flottantes par seconde, de type entier
• **ram** la quantité de mémoire vive de l'ordinateur en Go, de type entier.

Exemple du contenu de cette table :

nom	gflops	ram
nyarlathotep114	69	32
nyarlathotep119	137	32
...		
ahubniggurath42	133	16
azathoth137	85	8

T.S.V.P →

La seconde table est **fonctions** et stocke de l'information sur les tests effectués pour différentes fonctions en cours de développement. Ses attributs sont :

- **id** l'identifiant du test effectué (de type entier)
- **nom** le nom de la fonction testée (par exemple une procédure Python), de type chaîne.
- **algorithme** le nom de l'algorithme qui permet le calcul de la fonction testée (par exemple BBS si l'on teste une fonction de géneration de nombres aléatoires), de type chaîne.
- **teste_sur** le nom du PC sur lequel le test a été effectué, de type chaîne.
- **temps_exec** le temps d'exécution du test en millisecondes, de type entier.

Exemple du contenu de cette table :

id	nom	algorithme	teste_sur	temps_exec
1	*li*	rectangles	nyarlathotep165	2638
2	*li*	rectangles	shubniggurath28	736
3	*li*	trapezes	nyarlathotep165	4842
...				
2154	*Ei*	puiseux	nyarlathotep145	2766
2155	*aleatoire*	BBS	azathoth145	524

1. Expliquer pourquoi il n'est pas possible d'utiliser l'attribut **nom** comme clé primaire de la table **fonctions**.
2. Écrire des requêtes SQL permettant de :
   (a) Connaître le nombre d'ordinateurs disponibles et leur quantité moyenne de mémoire vive.

   Indications *: penser à* **COUNT** *et à* **AVG**

   (b) Écrire une première requête où on récupére la liste des ordinateurs ayant utilisé l'algorithme **"rectangles"** pour **"li"**.

   Écrire une seconde requête où on extrait les noms des PC sur lesquels l'algorithme "rectangles" n'a pas été testé pour la fonction nommée **"li"**

   Indications *: Pour la première requête, on utilise dans l'ordre d'écriture les fonctions ou clauses* **SELECT FROM JOIN ON WHERE AND**

   Pour la seconde requête, on utilise **SELECT FROM** *puis on utilisera* **EXCEPT** *et on reécrit la première requête.*

   (c) Pour la fonction nommée Ei, trier les résultats des tests du plus lent au plus rapide. Pour chaque test, retenir le nom de l'algorithme utilisé, le nom du PC sur lequel il a été effectué et la puissance du PC.

   Indications *: On utilisera dans l'ordre* **SELECT, FROM, JOIN, ON, WHERE, ORDER BY, DESC**