Exercice 1 : SQL

CCINP type A, sujet 0

On considère le schéma de base de données suivant, qui décrit un ensemble de fabricants de matériel informatique, les matériels qu'ils vendent, leurs clients et ce qu'achètent leurs clients. Les attributs des clés primaires des six premières relations sont soulignés.

```
Production(NomFabricant, Modele)
Ordinateur(Modele, Frequence, Ram, Dd, Prix)
Portable(Modele, Frequence, Ram, Dd, Ecran, Prix)
Imprimante(Modele, Couleur, Type, Prix)
Fabricant(Nom, Adresse, NomPatron)
Client(Num, Nom, Prenom)
Achat(NumClient, NomFabricant, Modele, Quantite)
```

Chaque client possède un numéro unique connu de tous les fabricants. La relation `Production` donne pour chaque fabricant l'ensemble des modèles fabriqués par ce fabricant. Deux fabricants différents peuvent proposer le même matériel. La relation `Ordinateur` donne pour chaque modèle d'ordinateur la vitesse du processeur (en Hz), les tailles de la Ram et du disque dur (en Go) et le prix de l'ordinateur (en €). La relation `Portable`, en plus des attributs précendents, comporte la taille de l'écran (en pouces). La relation `Imprimante` indique pour chaque modèle d'imprimante si elle imprime en couleur (vrai/faux), le tye d'impression (laser ou jet d'encre) et le prix (en €). La relation `Fabricant` stocke les nom et adresse de chaque fabricant, ainsi que le nom de son patron. La relation `Client` stocke les noms et prénoms de tous les clients de tous les fabricants. Enfin, la relation `Achat` regroupe les quadruplets (client c, fabriquant f, modèle m, quantité q) tels que le client de numéro c a acheté q fois le modèle m au fabricant f. On suppose que l'attribut `Quantite` est toujours strictement positif.

Q. 1 Proposer une clé primaire pour la relation `Achat` et indiquer ses conséquences en terme de modélisation.

Q. 2 Identifier l'ensemble des clés étrangères éventuelles de chaque table.

Q. 3 Donner en SQL des requêtes répondant aux question suivantes :

$a)$ Quels sont les numéros des modèles des matériels (ordinateur, portable ou imprimante) fabriqués par l'entreprise du nom de Durand ?

$b)$ Quels sont les noms et adresses des fabricants produisant des portables dont le disque dur a une capacité d'au moins 500 Go ?

$c)$ Quels sont les noms des fabricants qui produisent au moins 10 modèles différents d'imprimantes ?

$d)$ Quels sont les numéros des clients n'ayant acheté aucune imprimante ?

Exercice 2 : Langage de Dyck

CCINP type B, 2024

Cet énoncé est accompagné d'un ou code compagnon `ccinp_2024_dyck.c` fournissant certaines des fonctions mentionnées dans l'énoncé : il est à compléter en y implémentant les fonctions demandées.
La ligne de compilation `gcc -o ccinp_2024_dyck.exe -Wall *.c -lm` vous permet de créer un exécutable `ccinp_2024_dyck.exe` à partir du ou des fichiers C fournis. Vous pouvez également utiliser l'utilitaire `make`. En ligne de commande, il suffit d'écrire `make`. Dans les deux cas, si la compilation réussit, le programme peut être exécuté avec la commande `./ccinp\_2024\_dyck.exe`.
Il est possible d'activer davantage d'avertissements et un outil d'analyse de la gestion de la mémoire avec la ligne de compilation `gcc -o main.exe -g -Wall -Wextra -fsanitize=address *.c -lm` ou en écrivant `make safe`. L'examinateur pourra vous demander de compiler avec ces options.

Si vous désirez forcer la compilation de tous les fichiers, vous pouvez au préalable nettoyer le répertoire en faisant `make clean` *et relancer une compilation.*
La compilation du code compagnon initial avec `make safe` *provoque des warnings attendus qui seront résolus lors de l'implémentation des fonctions demandées par le sujet.*

On s'intéresse dans cet exercice aux mots de Dyck, c'est-à-dire aux mots bien parenthésés. Dans ce type de mots, toute parenthèse ouverte "(" est fermée ")" et une parenthèse ne peut être fermée si elle ne correspond pas à une parenthèse préalablement ouverte.
Par exemple pour deux couples de parenthèses, "(())" et "()()" sont des chaînes de parenthèses bien formées. "())(" et ")()(" ne le sont pas.
On admet que le nombre de mots bien parenthésés à n couples de parenthèses est donné par le nombre de Catalan C_n. Ceux-ci sont définis par la formule suivante.

$$C_n = \frac{(2n)!}{(n+1)!n!} \text{ pour } n \geqslant 0$$

On rappelle que le type `uint64_t` est un type entier non signé codé sur 64 bits.

Q. 1 Complétez dans le code compagnon la fonction dont le prototype est `uint64_t catalan(int n)`. Vous pouvez utiliser une fonction auxiliaire si cela vous semble pertinent.

Q. 2 Que va-t-il se passer si on tente d'afficher `catalan(n)` pour `n` un peu grand ? Le constatez-vous ici ?

On cherche maintenant à afficher le nombre de mots (chaînes) bien parenthésés avec n fixé couples de parenthèses, ainsi que les mots eux-mêmes.
Un algorithme de force brute pour déterminer toutes les chaînes à n couples de parenthèses bien formées consiste à générer toutes les possibilités puis à ne garder que les chaînes bien formées.

Q. 3 Complétez dans le code compagnon la fonction dont le prototype est `bool verification(char * mot)`. Cette fonction renvoie `true` si le mot fourni en paramètre `mot` est bien parenthésé, `false` sinon.

Q. 4 Quelle est la complexité de cette vérification ?

Q. 5 Quelle est la complexité finale de l'algorithme de force brute ?

On appelle `n` le nombre de couples de parenthèses voulu. Dans le fichier compagnon fourni, le nombre de couples a été limité à 18.
On vous propose de coder l'énumération des chaînes de parenthèses bien formées en appliquant l'algorithme de backtracking suivant, dont on admet qu'il est correct : on compte le nombre de parenthèses ouvertes `o` et le nombre de parenthèses fermées `f` dans une chaîne de caractères courante (vide au départ).
- Si `o = f = n`, on a trouvé une chaîne bien formée.
- Si `o < n`, on ajoute une parenthèse ouvrante et on relance.
- Si `f < o`, on ajoute une parenthèse fermante et on relance.

Cet algorithme est à implémenter dans la fonction dont le prototype est `void dyck(char s[N], int o, int f, int n)` qui affiche sur la sortie standard les chaînes de parenthèses bien formées avec `n` couples de parenthèses lorsque `s` est la chaîne de caractère courante, `o` est son nombre de parenthèses ouvrantes et `f` est son nombre de parenthèses fermantes.

Q. 6 Compléter la fonction `dyck` pour afficher les chaînes bien parenthésées avec 5 couples de parenthèses.

Q. 7 Adapter la fonction `dyck` pour calculer le nombre de mots obtenus. Combien de mots trouvez-vous pour 16 couples de parenthèses ?

Q. 8 Adapter la fonction `dyck` pour stocker les mots bien parenthésés dans une liste chaînée et les afficher après l'appel à la fonction. Vous trouverez dans le code compagnon une structure qui peut vous aider.