Requêtes SQL

ITC PC

M. Charles

Parenthèse Python

Exemple

Quel affichage est produit par les codes suivants ?

```
L = [3, 42, -1, 483]
L2 = L
print(L)
print(L2)
```

Exemple

Quel affichage est produit par les codes suivants ?

```
L = [3, 42, -1, 483]
L2 = L
print(L)
print(L2)
```

```
L = [3, 42, -1, 483]
L[1] = 43
L2 = L
print(L)
print(L2)
```

Exemple

Quel affichage est produit par les codes suivants ?

```
L = [3, 42, -1, 483]
L2 = L
print(L)
print(L2)
```

```
L = [3, 42, -1, 483]
L[1] = 43
L2 = L
print(L)
print(L2)
```

```
L = [3, 42, -1, 483]
L2 = L
L[1] = 43
print(L)
print(L2)
```

Opération `copy()`

À votre avis, que produit le code suivant ?

```
L = [3, 42, -1, 483]
L2 = L.copy()
L[1] = 43
print(L)
print(L2)
```

Opération `copy()`

À votre avis, que produit le code suivant ?

```
L = [3, 42, -1, 483]
L2 = L.copy()
L[1] = 43
print(L)
print(L2)
```

Que se passe-t-il ?

Opération copy()

À votre avis, que produit le code suivant ?

```
L = [3, 42, -1, 483]
L2 = L.copy()
L[1] = 43
print(L)
print(L2)
```

Que se passe-t-il ?

Que fait le code suivant ?

```
L = [[3, 42], [-1, 483]]
L2 = L.copy()
L[0][1] = 43
print(L)
print(L2)
```

Opération copy()

À votre avis, que produit le code suivant ?

```
L = [3, 42, -1, 483]
L2 = L.copy()
L[1] = 43
print(L)
print(L2)
```

Que se passe-t-il ?

Que fait le code suivant ?

```
L = [[3, 42], [-1, 483]]
L2 = L.copy()
L[0][1] = 43
print(L)
print(L2)
```

Pourquoi ?

Exercice (CCINP 2024 modifié)

Écrire une fonction `copie(L:list) -> list` qui renvoie une copie profonde de la structure de liste de listes. Il est interdit d'utiliser la fonction `deepcopy` du module `copy`. L'opérateur `copy` des listes est autorisé.

Exercice (CCINP 2024 modifié)

Écrire une fonction `copie(L:list) -> list` qui renvoie une copie profonde de la structure de liste de listes. Il est interdit d'utiliser la fonction `deepcopy` du module `copy`. L'opérateur `copy` des listes est autorisé.

```
def copie(L: list) -> list:
    L2 = []
    for i in range(len(L)):
        L2.append(L[i].copy())
    return L2
```

Formatage des résultats

DISTINCT

Les requêtes SQL peuvent produire des tables contenant plusieurs fois la même ligne. Lorsqu'on souhaite éliminer ces doublons, on ajoute en général le mot-clé `DISTINCT` juste après le `SELECT`. Exemple :

```
SELECT DISTINCT idLivreur
FROM Commande
WHERE Jour=04-14 ;
```

LIMIT

Il est possible de ne conserver que les n premières lignes d'une table avec le mot-clé `LIMIT`. Exemple :

```
SELECT idClient
FROM   Commande
WHERE  Jour=04-14
LIMIT  3 ;
```

On peut aussi *jeter* les k premières lignes avec le mot-clé `OFFSET`. Exemple :

```
SELECT idClient
FROM   Commande
WHERE  Jour=04-14
LIMIT  1
OFFSET 2 ;
```

Il est également possible de trier les lignes d'une table selon leurs valeurs dans certaines colonnes. Pour cela on ajoute en fin de requête SQL la clause `ORDER BY col ASC` pour trier dans l'ordre croissant selon la colonne `col`, ou `ORDER BY col DESC` pour trier dans l'ordre décroissant. Exemples :

```
SELECT   Ref AS Pizza, Prix
FROM     Produit
LIMIT    5
ORDER BY Prix ASC ;
```

```
SELECT   idClient
FROM     Commande
LIMIT    3
ORDER BY Jour DESC ;
```

Opérations sur les lignes et les colonnes

Le langage SQL permet également de réaliser des **calculs sur les colonnes** ou des **opérations statistiques**. Considérons par exemple la table `Économie` suivante.

`Pays`	`Continent`	`Habitants`	`PIB`
France	Europe	67	2775
Royaume-Uni	Europe	66	2828
Allemagne	Europe	83	4000
Espagne	Europe	47	1425
États-Unis	Amérique	333	20494
Canada	Amérique	38	1711
Mexique	Amérique	128	1223

Pays	Continent	Habitants	PIB
France	Europe	67	2775
Royaume-Uni	Europe	66	2828
Allemagne	Europe	83	4000
Espagne	Europe	47	1425
États-Unis	Amérique	333	20494
Canada	Amérique	38	1711
Mexique	Amérique	128	1223

Si l'on souhaite calculer le PIB par habitant, il suffit de créer une colonne `PIB / Habitants`. On procède ainsi :

```
SELECT Pays, PIB / Habitants FROM Économie ;
```

```sql
SELECT Pays, PIB / Habitants FROM Économie ;
```

Résultat :

Pays	PIB / Habitants
France	41.42
Royaume-Uni	42.84
Allemagne	48.19
Espagne	30.31
États-Unis	61.54
Canada	45.03
Mexique	9.55

NB : on aurait aussi pu renommer la seconde colonne, trier et ne garder que le top 3, par exemple en écrivant

```sql
SELECT Pays, PIB / Habitants as Niveau
FROM   Économie
ORDER BY Niveau DESC
LIMIT  3 ;
```

Fonctions d'agrégation

Les **fonctions d'agrégation** travaillent en agrégeant, c'est-à-dire en regroupant, toutes les lignes d'une table en une seule.

Celles à connaitre sont

- `SUM`,
- `AVG`,
- `COUNT`,
- `MIN`,
- `MAX`.

Par exemple, si on souhaite obtenir le nombre total d'habitants dans la table `Économie`, on pourra écrire

```
SELECT SUM(Habitants) FROM Économie ;
```

On obtient alors une table n'ayant qu'une seule ligne, où la colonne `SUM(Habitants)` contient la somme de toutes les valeurs qui étaient dans la colonne `Habitants`.

SUM(Habitants)
762

Lorsqu'on utilise les fonctions d'agrégation `MIN` ou `MAX`, il est possible d'ajouter d'autres colonnes pour déterminer sur quelle ligne le minimum ou le maximum est atteint :

```
SELECT Pays, MIN(PIB) FROM Économie ;
```

aura pour résultat

Pays	MIN(PIB)
Mexique	1223

Groupage

Parfois, on souhaite réaliser une opération statistique non pas sur *l'ensemble* de la table, mais sur des **groupes de lignes**.

Par exemple, on peut vouloir obtenir le nombre total d'habitants par continent, et la fonction `SUM` ne devrait alors s'appliquer qu'aux groupes de lignes correspondant à un même continent.

On utilise pour cela le mot-clé `GROUP BY`.

```sql
SELECT   Continent, SUM(Habitants)
FROM     Économie
GROUP BY Continent ;
```

Résultat :

Continent	SUM(Habitants)
Europe	263
Amérique	499

```sql
SELECT   Continent, SUM(Habitants)
FROM     Économie
GROUP BY Continent ;
```

Résultat :

Continent	SUM(Habitants)
Europe	263
Amérique	499

Remarque : si on oublie d'afficher la colonne selon laquelle on effectue le groupage, on risque d'avoir du mal à comprendre à quoi correspond la table obtenue.

Spécificités de `COUNT`

La fonction d'agrégation `COUNT` sert à compter le nombre de lignes d'une table, ou le nombre de lignes de chaque groupe si le mot-clé `GROUP BY` a été utilisé.

- `COUNT(*)` permet de compter le nombre de lignes de la table.
- `COUNT(col)` permet de compter le nombre de lignes pour lesquelles la valeur de l'attribut `col` n'est pas `NULL` (*un peu HP*).
- `COUNT(DISTINCT col)`permet de compter le nombre de valeurs distinctes dans la colonne `col` (*et différente de `NULL`*).

Filtrage des agrégats : `HAVING`

Reprenons la table `Économie` suivante.

Pays	Continent	Habitants	PIB
France	Europe	67	2775
Royaume-Uni	Europe	66	2828
Allemagne	Europe	83	4000
Espagne	Europe	47	1425
États-Unis	Amérique	333	20494
Canada	Amérique	38	1711
Mexique	Amérique	128	1223

```sql
SELECT SUM(Habitants) FROM Économie ;
```

SUM(Habitants)
762

```sql
SELECT Pays, MIN(PIB) FROM Économie ;
```

Pays	MIN(PIB)
Mexique	1223

```
SELECT   Continent, SUM(Habitants)
FROM     Économie
GROUP BY Continent ;
```

Résultat :

Continent	SUM(Habitants)
Europe	263
Amérique	499

Les opérations de groupage s'effectuent **en amont** des calculs.

1. On groupe, puis
2. on effectue les calculs groupe par groupe.

Filtrage avec `HAVING`

On peut filtrer les résultats **en aval** avec le mot-clé `HAVING` :

```
SELECT Continent, SUM(Habitants)
FROM   Économie GROUP BY Continent
HAVING SUM(Habitants) > 300 ;
```

Résultat :

Continent	SUM(Habitants)
Amérique	499

WHERE versus HAVING

Le `WHERE` est évalué en premier, le `HAVING` en dernier.

1. On sélectionne des lignes avec `WHERE`,
2. on groupe les résultats avec `GROUP BY`,
3. on effectue les calculs (`SUM`, `MIN`…) groupe par groupe,
4. on filtre les résultats avec `HAVING`.

Exemple

```
SELECT   Continent, SUM(Habitants)
FROM     Économie
WHERE    PIB > 2000
GROUP BY Continent
HAVING   SUM(Habitants) > 300 ;
```

Opérations binaires

Principe

Cette fois-ci, on cherche à **combiner deux tables** pour en déduire une **troisième** table.

Opérations ensemblistes simples

Assez rares en pratique. S'appliquent à deux tables ayant même schéma, par exemple résultant de deux requêtes différentes sur une même table.

On considère chaque table comme l'ensemble de ses lignes et on applique les opérations considérées sur les deux ensembles obtenus.

- Réunion : `T1 UNION T2`
- Intersection : `T1 INTERSECT T2`
- Différence : `T1 EXCEPT T2`

Produit cartésien

(Théorique mais prépare l'introduction de la jointure)

Si T_1 et T_2 sont deux tables de schéma respectif S_1 et S_2, alors $T_1 \times T_2$ est une table de schéma $S_1 \cup S_2$, dont les lignes sont toutes les combinaisons possibles d'une ligne de T_1 et d'une ligne de T_2.

Produit cartésien

(Théorique mais prépare l'introduction de la jointure)

Si T_1 et T_2 sont deux tables de schéma respectif S_1 et S_2, alors $T_1 \times T_2$ est une table de schéma $S_1 \cup S_2$, dont les lignes sont toutes les combinaisons possibles d'une ligne de T_1 et d'une ligne de T_2.

En SQL : `T1, T2`

Exemple : `SELECT Adresse, Modèle FROM Client, Véhicule ;`

Jointures

Permettent de **croiser** les données de deux tables différentes, en complétant les données de l'une avec les données de l'autre.

Si T_1 et T_2 sont deux tables, leur **jointure** suivant une condition donnée est ce que l'on obtient en

1. formant le produit cartésien $T_1 \times T_2$ puis
2. en sélectionnant les lignes vérifiant la condition.

En SQL : `T1 JOIN T2 ON condition`.

Remarque

La requête

```
SELECT * FROM T1 JOIN T2 ON condition ;
```

équivaut donc à

```
SELECT * FROM T1, T2 WHERE condition ;
```

mais elle est implémentée avec des algorithmes efficaces (HP).

Aux concours, il y aura presque sûrement zéro produit cartésien, et presque sûrement au moins une jointure.

Exemple

Que réalise la requête suivante ?

```
SELECT DISTINCT Prénom FROM
    (Commande JOIN Livreur
     ON Commande.idLivreur = Livreur.idLivreur)
WHERE Jour = 04-14 ;
```

Exemple 2

Que réalise la requête suivante ?

```
SELECT Adresse, P
FROM (
    (SELECT idClient AS I, Prénom as P
        FROM Commande JOIN Livreur
        ON Commande.idLivreur = Livreur.idLivreur
        WHERE Jour = 04.14)
    JOIN Client
    ON idClient = I) ;
```

Remarques : jointures multiples, autojointures

Dans la série « c'est rare mais ça existe » :

- on peut effectuer une jointure sur un nombre quelconque de tables :

```
SELECT * FROM t1 JOIN t2 JOIN t3 ...
ON condition ;
```

- on peut effectuer une jointure d'une table avec elle-même (autojointure), ce qui peut par exemple être utile en cas de liens hiérarchiques entre les livreurs de l'entreprise.

Un exercice : Mines 2022

Contexte

Recherche dans une BDD de matériaux magnétiques. Quatre questions de difficulté croissante.

Structure de la BDD

La BDD considérée contient trois tables ayant les schémas suivants :

- `matériaux(id_matériau int, nom text, t_curie int)`
- `fournisseurs(id_fournisseur int, nom_fournisseur text)`
- `prix(id_prix int, id_mat int, id_four int, prix_kg money)`

Structure de la BDD

La BDD considérée contient trois tables ayant les schémas suivants :

- `matériaux(id_matériau int, nom text, t_curie int)`
- `fournisseurs(id_fournisseur int, nom_fournisseur text)`
- `prix(id_prix int, id_mat int, id_four int, prix_kg money)`

Les deux premières tables contiennent des **détails** (sur les matériaux et les fournisseurs), la troisième **relie** les informations des deux premières (qui propose quoi à quel prix).

Exemple donné par le sujet

id_matériau	nom	t_curie
4534	cobalt	1 388
1254	dioxyde de chrome	386
8713	nickel	627
8284	YIG	560
…	…	…

id_fournisseur	nom_fournisseur
145	Worldwide Materials
13	Materials Company
…	…

Exemple donné par le sujet

id_prix	id_mat	id_four	prix_kg
1	4567	145	50,40
2	8671	13	1357,30
3	1763	145	52,75
...	...	...	...

Analyse de la structure de la table

- Clés primaires ?
- Clés étrangères ?
- Nature des enregistrements ?

Question 6

Écrire une requête permettant d'obtenir le nom de tous les matériaux qui ont une température de Curie strictement inférieure à 500 kelvins.

Question 6

Écrire une requête permettant d'obtenir le nom de tous les matériaux qui ont une température de Curie strictement inférieure à 500 kelvins.

1. Quels sont les mots importants dans cette phrase ?
2. Comment se traduisent-ils en termes de BDD ?
3. Comment traduire cela en SQL ?

Interlude

Un client potentiel souhaite acheter 4,5 kilogrammes de nickel (d'identifiant 8713, que l'on pourra utiliser directement dans les requêtes) et sélectionner le fournisseur le moins cher.

Question 7

Écrire une requête permettant d'obtenir les noms de tous les fournisseurs proposant du nickel et le prix proposé par chacun pour 4,5 kilogrammes de nickel.

Question 7

Écrire une requête permettant d'obtenir les noms de tous les fournisseurs proposant du nickel et le prix proposé par chacun pour 4,5 kilogrammes de nickel.

1. Que souhaite-t-on obtenir ?
2. Quelles opérations effectuer et dans quel ordre ?
3. Comment traduire cela en SQL ?

Question 8

Modifier ou compléter la requête précédente afin d'obtenir le nom du fournisseur de nickel le moins cher ainsi que le prix à payer chez ce fournisseur pour ces 4,5 kilogrammes de nickel.

En cas d'égalité du prix optimal entre plusieurs fournisseurs, on obtiendra les noms de tous les fournisseurs possibles.

Question 9

Écrire une requête permettant d'obtenir le nom de tous les matériaux et le prix moyen pour un kilogramme de chacun de ces matériaux (la moyenne étant calculée pour tous les fournisseurs proposant ce matériau), en se limitant aux prix moyens strictement inférieurs à 50 euros par kilogramme.