**BLOC2 BDD.2 : SQL avancé**

|  |
| --- |
| Objets d’un traitement informatique  |

# La gestion des jointures

Leclerc souhaite enregistrer les catégories de client. Une catégorie est identifié par un num et a un nom (ouvrier/cadre….)

**Question 1** : Faire le MCD de ce modèle.

Voici un extrait des tables :

Client



Catégorie



**question 2.** Afficher la liste des clients ainsi que leurs catégories (déduire le tableau de résultat avant de faire la requête)

**question 3.** Colorier la partie représentant la requête de la question 2

catégorie

Client

##  Jointure interne (INNER JOIN)

Une jointure interne permet de sélectionner les enregistrements ayant des correspondances entre deux fichiers joints.

*Par exemple, pour lister les clients ayant déjà passé des commandes, il est nécessaire de lier les fichiers "Client" et "Commande" grâce à une jointure interne. La requête sélectionnera uniquement les clients liés à au moins un numéro de commande. Les clients n'ayant jamais passé de commande ne seront pas listés.*

**question 4 :** colorier la partie qui représente un "INNER JOIN"

Table1

Table2

SELECT \*

FROM table1 **INNER JOIN** table2

**ON** (table1.champPK = table2.champFK)

Cela équivaut à notre jointure habituelle :

SELECT \*

FROM table1,table2,

WHERE table1.champPK = table2.champFK

**question 5** : reprendre la requête à la Q2 en faisant une jointure "INNER JOIN".

## Jointure externe (OUTER JOIN)

Une jointure externe permet de sélectionner à la fois :

* les enregistrements ayant des correspondances entre les deux fichiers joints.
* les enregistrements n'ayant pas de correspondance dans le premier, le second ou dans tous les fichiers joints.

Par exemple, pour connaître le montant total dépensé par chaque client :

* En utilisant une jointure interne, seuls les clients ayant déjà passé une commande (dont le montant dépensé sera différent de 0) seront sélectionnés.
* En utilisant une jointure externe, tous les clients seront sélectionnés, même ceux n'ayant jamais passé de commande.

**question 6** : Trouver les 3 types de jointure externe en coloriant les différentes zones

Jointure externe 1 :

Table1

Table2

Table1

Table2

Jointure externe 2 :

Table1

Table2

Jointure externe 3 :

**Les différents types de jointures externes**

|  |  |
| --- | --- |
|  |  |
| FULL OUTER JOIN | Permet de sélectionner :* les enregistrements satisfaisant à la jointure.
* les enregistrements du premier fichier cité qui ne satisfont pas la jointure. Le nom du premier fichier cité est placé à gauche de la jointure OUTER JOIN.
* les enregistrements du second fichier cité qui ne satisfont pas la jointure. Le nom du second fichier cité est placé à droite de la jointure OUTER JOIN.
 |
| LEFT OUTER JOIN | **Jointure la plus courante**. Permet de sélectionner :* les enregistrements satisfaisant à la jointure.
* les enregistrements du premier fichier cité qui ne satisfont pas la jointure. Le nom du premier fichier cité est placé à gauche de la jointure OUTER JOIN.
 |
| RIGHT OUTER JOIN | Permet de sélectionner :* les enregistrements satisfaisant à la jointure.
* les enregistrements du second fichier cité qui ne satisfont pas la jointure. Le nom du second fichier cité est placé à droite de la jointure OUTER JOIN.
 |

**question 7** : Afficher la liste des clients ainsi que leurs catégories (déduire le tableau de résultat avant de faire la requête)

**En faisant attention cette fois à sélectionner les clients qui n'ont pas de catégorie**

Client

catégorie

**question 8** : Que se passe t-il si vous inversez client et catégorie dans le FROM ?

Client

catégorie

**question 9** : Afficher la liste des clients ainsi que leurs catégories - en affichant les clients sans catégorie et les catégories sans client (déduire le tableau de résultat avant de faire la requête)

Client

catégorie

## Pour aller encore plus loin avec les jointures





**TD7**

# Les vues

Une vue est une table virtuelle, c'est-à-dire dont les données ne sont pas stockées dans une table de la base de données, et dans laquelle il est possible de rassembler des informations provenant de plusieurs tables. On parle de "vue" car il s'agit simplement d'une représentation des données dans le but d'une exploitation visuelle.

**CREATE VIEW** nomdelavue

**AS**

SELECT…..

Il est parfois plus simple (même nécessaire) de passer par une vue pour effectuer une requête SQL.

ref : SLAM1\_TD1 (**étape 15** : Faire la requête qui permet de retourner le jury le plus sévère)

& PPE (stage)

# Les "multi" requêtes (UNION, MINUS)

## UNION

La commande UNION de SQL permet de mettre bout-à-bout les résultats de plusieurs requêtes utilisant elles-mêmes la commande SELECT. C’est donc une commande qui permet de concaténer les résultats de 2 requêtes ou plus. Pour l’utiliser il est nécessaire que chacune des requêtes à concaténer retournes le même nombre de colonnes, avec les mêmes types de données et dans le même ordre.

A savoir : par défaut, les enregistrements exactement identiques ne seront pas répétés dans les résultats. Pour effectuer une union dans laquelle même les lignes dupliquées sont affichées il faut plutôt utiliser la commande UNION ALL.

SELECT \* FROM table1

**UNION**

SELECT \* FROM table2

Soit une partie du MCD du ppe stage suivant :



MPD correspondant :



**Question 10 :** Afficher la globalité des informations d'un élève

**Question 11 :** Afficher la globalité des informations d'un Prof

**Question 12** : Afficher la globalité des informations des élèves et des profs

**Question 13** : Créer la vue "user" de la question 12 en rajouter une colonne type (afficher 1 quand l'utilisateur est un élève, 2 quand il est un prof)



**TD8**

# Les sous requêtes

Dans le langage SQL une sous-requête (aussi appelé « requête imbriquée » ou « requête en cascade») consiste à exécuter une requête à l’intérieur d’une autre requête. Une requête imbriquée est souvent utilisée au sein d’une clause WHERE ou de HAVING pou remplacer une ou plusieurs constante.

## Syntaxe

Il y a plusieurs façons d’utiliser les sous-requêtes. De cette façon il y a plusieurs syntaxes envisageables pour utiliser des requêtes dans des requêtes.

### Requête imbriquée qui retourne un seul résultat

L’exemple ci-dessous est une exemple typique d’une sous-requête qui retourne un seul résultat à la requête principale.

SELECT \*

FROM `table`

WHERE `nom\_colonne` = (

 SELECT `valeur`

 FROM `table2`

 LIMIT 1

 )

Cet exemple montre une requête interne (celle sur « table2 ») qui renvoi une seule valeur. La requête externe quant à elle, va chercher les résultat de « table » et filtre les résultats à partir de la valeur retournée par la requête interne.

**A noter :** il est possible d’utiliser n’importe quel opérateur d’égalité tel que =, >, <, >=, <= ou <>.

### Requête imbriquée qui retourne une colonne (plusieurs résultats)

Une requête imbriquée peut également retournée une colonne entière. Dès lors, la requête externe peut utiliser la commande IN (ou NOT IN) pour filtrer les lignes qui possèdent une des valeurs retournées par la requête interne. L’exemple ci-dessous met en évidence un tel cas de figure:

SELECT \*

FROM `table`

WHERE `nom\_colonne` IN (

 SELECT `colonne`

 FROM `table2`

 WHERE `cle\_etrangere` = 36

 )

## Exemples

Exemple 1 : Imaginons un site web qui permet de poser des questions et d’y répondre. Un tel site possède une base de données avec une table pour les questions et une autre pour les réponses.



**Question 14 :** Trouver la requête permettant de connaître la question liée à la dernière réponse ajoutée sur l’application.

**Question 15** : Trouver la requête permettant d'obtenir les questions liées à toutes les réponses comprises entre 2 dates.

**Question 16** : Trouver la requête permettant d'obtenir les questions sans réponse

****

**TD9**