Introduction aux bases de données et au langage SQL

Résumé

Ce TP a pour but la découverte d'un environnement de gestion de bases de données et un premier contact avec la notion de requête. On suppose pour ce TP qu'une version de phpMyAdmin est accessible. Connectez vous à votre espace et accédez à la base de données dans laquelle vous souhaitez travailler, par exemple une base nommée TPBanque. On suppose qu'elle est actuellement vide.

1 Connexion et mise en place

Vous entrez dans l'interface de gestion de votre base de données. Pour le moment, elle est vide, ce qui est indiqué par le `(0)` à côté du nom de la base dans la marge de gauche, ainsi que le message *« Aucune table n'a été trouvée dans cette base. »* dans la zone principale. Vous pouvez remarquer en particulier que les onglets `Rechercher`, `Requête` et `Exporter` sont désactivés. Nous allons remplir cette base de données avec des fichiers générés au préalable.

- Cliquer sur l'onglet `Importer`
- Cliquer sur le bouton `Choisir un fichier` – ou équivalent
- Sélectionner le fichier `bd-banque-simple.sql` puis cliquer sur `Exécuter`
- Réitérer avec dans l'ordre avec les fichiers
  - `bd-banque-simple-random-data-client.sql`
  - `bd-banque-simple-random-data-compte.sql` et
  - `bd-banque-simple-random-data-operation.sql`.

Rendez vous dans l'onglet `Structure`. Vous pouvez observer que 3 *Tables* ont été créées, dont les noms sont `client`, `compte` et `operation`. En cliquant sur l'icône à droite du nom d'une table, vous pouvez afficher son contenu. Lorsque vous visualisez une table, vous pouvez cliquer sur l'onglet `Structure` et voir les propriétés de cette table.

■ Faire le bilan des attributs de chacune des tables, c'est-à- dire pour chaque table, donner le nom des colonnes ainsi que leur type.

Allez dans l'onglet `Structure` de la base `compte`. Au milieu de la page se trouve un lien *gestion des relations* ou *vue relationnelle* sur certaines versions : cliquez dessus. On voit dans la page qui s'affiche que l'attribut `idproprietaire` est relié à l'attribut `idclient` de la table `client` par une *clef étrangère* : cela signifie que l'attribut `idproprietaire` de la table `compte` ne peut prendre QUE des valeurs existantes de l'attribut `idclient` de la table `client`.

■ Quelles relations de clef étrangère relève-t-on dans la table `operation` ?

L'attribut `idcompte` *est relié à l'attribut* `idcompte` *de la table* `compte`*. L'*`idcompte` *de la table* `compte` *est une clef primaire : il permet d'identifier de façon unique les comptes.*

On a donc une vision assez nette des différentes tables présentes dans la BDD, ainsi que des interactions qui existent entre les tables. Nous allons ici nous intéresser à la recherche d'infomations dans les tables.

2 Recherche simple

Aller dans la base `client` via le menu de gauche et ouvrez l'onglet `Rechercher`.

■ À l'aide de la recherche, afficher le client dont l'identificateur est le 42.

Lorsque l'on effectue une recherche, l'écran affiche deux informations importantes. Dans le cadre supérieur s'affiche la *requête* correspondant à la recherche. La syntaxe d'une requête est bien particulière. On doit normalement avoir

```
SELECT *
FROM  `client`
WHERE  `idclient`=42
```

▷ le symbole ` est un guillemet à l'envers (touche 7). Il n'est obligatoire que lorsque le nom d'attribut contient un espace.

On traduit cette requête en français par : « sélectionner tous les attributs des lignes de la table ‘client‘ dont l’idclient est égal à 42 ».

On peut aussi ne sélectionner que certains attributs. Retournez dans l’onglet Rechercher et cliquez sur *Options* en bas de la page.

■ Grâce à la fonction de Recherche, affichez les nom et prenom de tous les clients dont l’identificateur est inférieur strictement à 15.

On doit obtenir une requête de la forme :

```
SELECT  nom ,  prenom
FROM  client
WHERE  idclient <15
```

que l’on peut traduire en français par « sélectionner les attributs nom et prenom des lignes de la table client dont l’idclient est strictement inférieur à 15 ».

■ Sous la requête affichée se trouve un lien *modifier*. Cliquer dessus et modifier à la main la requête afin d’afficher : « les nom, prenom et ville des clients dont le prénom est "Jordan" ».

Il suffit de remplacer la requête par la requête suivante :

```
SELECT  nom ,  prenom, ville
FROM  client
WHERE  prenom="Jordan"
```

▷ Le symbole % placé dans une chaîne est un passe-partout qui signifie *n’importe quelle chaîne*

On remarquera l’utilisation de guillemets doubles pour délimiter la chaîne de caractères.

■ En retournant dans l’onglet Rechercher, trouver comment afficher « les nom et prenom de tous les clients dont le prénom commence par la lettre J ».

On met dans le champs de recherche la chaîne de caractère J% *qui permet de rechercher n’importe quelle chaîne en mettant une contrainte sur la première lettre. Si l’on écrit* J%n *(question suivante), on aura les prénoms commençant par* J *et finissant par* n*. Si on écrit par exemple* % t% e%*, on aura les prénoms contenant un* t *et un* e*, le* t *étant placé avant le* e*.*

Le langage SQL reconnaît les connecteurs logiques, comme par exemple *OU* et *ET*. On peut notamment les utiliser dans la clause WHERE.

■ Modifier la requête afin d’afficher « la ville des clients dont le prénom commence par la lettre J et finit par la lettre n ET l’identificateur est inférieur à 25 ».

On remplace la requête par la requête suivante :

```
SELECT  ville
FROM  client
WHERE  prenom="J\%n"
  AND  idclient<=25
```

On remarque l’utilisation du AND *qui permet de réaliser une intersection.*

■ Écrire une requête permettant d’afficher « les montant et informations des opérations réalisées sur les comptes d’identificateur 1 et 13 ». Vous devriez obtenir les résultats suivants :

montant	informations
20.17	Virement
26.19	Guichet
307.93	Cheque
-360.29	Guichet
526.86	Virement
7.95	Cheque
131.47	Cheque

La requête obtenue devrait être de la forme suivante :

```
SELECT  montant ,  informations
FROM  operations
WHERE  idcompte = 1 OR idcompte=13
```

On remarque l’utilisation du OR *qui permet de réaliser une union.*

Ainsi donc, pour effectuer une recherche simple dans une base de données, on écrit la requête de la façon suivante :

```
SELECT les attributs à afficher
FROM la table dans laquelle on cherche
WHERE les critères de recherche, séparés par OR ou AND par exemple
```

■ Écrire et tester les requêtes fournissant les informations suivantes :
- les identifiants de tous les comptes de type `Livret A` (72 enregistrements) ;
- les identifiants d'opération et montant de toutes les opérations effectuées au guichet pour un montant compris entre 0 et 100 (57 enregistrements)

3 Croiser les tables

Nous avons pour le moment effectué des recherches sur une table unique. On se rend cependant assez vite compte que cet aspect est limité. Ne pourrait-on pas par exemple afficher les nom et prénom des propriétaires de comptes possédant un Livret A ? Ou bien visualiser pour chaque compte toutes les opérations qui ont été effectuées ?

Prenons l'exemple de l'affichage des nom et prénom des propriétaires de chaque Livret A. La requête doit renvoyer par exemple le nom, le prenom et l'identificateur du compte. Chaque ligne du résultat sera donc un triplet. Contrairement aux requêtes précédentes, les attributs ne viennent pas d'une seule table. On pourrait décrire les triplets comme étant les éléments de l'ensemble suivant :

$$\left\{ (nom,prenom,idcompte)\ tq \quad \begin{array}{l} (idclient,nom,prenom,ville) \in \texttt{client}\ et \\ (idcompte,idproprietaire,type) \in \texttt{compte}\ et \\ idclient = idproprietaire \end{array} \right\}$$

que l'on traduit de la façon suivante : Afficher tous les triplets (*nom*,*prenom*,*idcompte*) tels que :
- il existe un `idclient` et une `ville` tels que (`idclient`,`nom`,`prenom`,`ville`) est une ligne de la table `client` ;
- il existe un `idproprietaire` et un `type` tels que (`idcompte`,`idproprietaire`,`type`) est une ligne de la table `compte` ;
- les attributs `idproprietaire` et `idclient` ont la même valeur.

L'égalité entre les attributs s'appelle une *jointure*. On construit la requête correspondante en s'appuyant sur les balises `SELECT`,`FROM` et `WHERE` de la façon suivante :

`SELECT` : indiquer ici les attributs à afficher (dans notre cas : `client.nom`, `client.prenom` et `compte.idcompte`)
`FROM` : indiquer dans la balise les tables à utiliser (ici : `client` et `compte`)
`WHERE` : poser les contraintes, puis les conditions de jointure.

On obtient donc la requête :

▷ Les commentaires en langage SQL sont précédés d'un dièse et vont jusqu'à la fin de la ligne

```
SELECT client.nom,client.prenom,compte.idcompte #attributs a afficher
FROM client, compte                             #tables necessaires
WHERE client.idclient=compte.idproprietaire     #jointure
  AND compte.type='Livret A'                    #contrainte
```

■ Exécuter cette requête. On doit obtenir 72 résultats. On remarque qu'il y a des doublons de `nom` et `prenom` : la base de données autorise un client à posséder plusieurs Livrets A.

On veut maintenant avoir un listing des opérations effectuées sur chaque compte pour un montant compris entre 0 et 100 euros. On demande à avoir l'information de l'identificateur du propriétéaire, l'identificateur du compte, le type de compte et le montant et l'information de l'opération.

■ Faire le bilan des arguments à afficher.

Les attributs à afficher sont `idproprietaire`, `idcompte` *et* `type` *de la table* `compte` *et* `montant` *et* `informations` *de la table* `operation`.

■ Faire le bilan des tables à utiliser pour obtenir ces arguments.

On doit donc faire appel aux deux tables `compte` *et* `operation`.

■ Faire le bilan des contraintes.

Les deux contraintes sont `montant>=0` *et* `montant<=100`.

■ Faire le bilan des jointures (quel est l'attribut qui va permettre de relier une table à une autre ?).

Après la requête, chaque ligne affichée contient de l'information sur un compte et de l'information sur une opération. Il faut que ces deux informations coïncident : le lien est l'attribut idcompte présent dans les deux tables. On aura donc comme condition de jointure de la forme `compte.idcompte = operation.idcompte`

client

idclient	nom	prenom	ville
1	Moulin	Sylvie	Paris
2	Legrand	Anne	Paris
3	Dubois	Emile	Neuilly
4	Leroy	Marie	Neuilly

compte

idcompte	idproprietaire	type
1	1	Compte Courant
2	1	Livret A
3	1	Assurance Vie
4	2	Compte Courant
5	3	Compte Courant
6	3	Assurance Vie
7	4	Assurance Vie
8	4	Plan Epargne Actions
9	4	Livret A

operation

idop	idcompte	montant	informations
1	1	2000.00	Salaire
2	1	-121.53	Courses
3	1	-75.92	Essence
4	1	-150.00	VIR Livret A
5	2	150.00	VIR du Compte Courant
6	4	3000.00	Salaire
7	3	10000.00	VIR initial
8	3	537.00	Interets
9	5	500.00	VIR initial

TABLE 1 – Exemple de base de données

■ En déduire la requête à effectuer et la tester sur l'interface. On pourra essayer de la tester d'abord *à la main* sur la base simplifiée de la figure 1. Sur l'interface, vous devez obtenir 162 résultats.

```
SELECT compte.idproprietaire, compte.idcompte,
       operation.montant,operation.informations
FROM compte,operation
WHERE compte.idcompte=operation.idcompte
  AND operation.montant>=0
  AND operation.montant<=100
```

Un problème se pose maintenant : on ne veut plus afficher l'`idcompte` et à la place de l'identificateur du propriétaire, on voudrait avoir son nom et son prenom.

■ Suivre de nouveau la même méthodologie pour construire la requête qui permettrait ce calcul. En particulier, vous devez avoir plusieurs conditions de jointure : lesquelles ?

On suit la même démarche et en particulier pour les jointures : l'information d'opération doit toujours coincider avec le compte, donc on récupère l'a condition de jointure `operation.idcompte=compte.idcompte`. *Mais cette fois, le proprietaire affiché doit être le proprietaire du compte et on récupère la condition :* `client.idclient=compte.idproprietaire`

```
SELECT client.nom, client.prenom,
       operation.montant, operation.informations
FROM client, operation, compte
WHERE client.idclient=compte.idproprietaire
  AND compte.idcompte=operation.idcompte
  AND operation.montant>=0
  AND operation.montant<=100
```

Si votre requête est bien menée, vous devriez obtenir le même nombre d'enregistrements, à savoir 162. Vous venez de réaliser la jointure la plus compliquée qu'il soit possible de faire sur cette base de données...

■ Modifier la requête pour afficher uniquement les montants sur les Livrets A avec une opération faite au guichet. Vous devriez obtenir 8 résultats au total.

On peut evidemment rajouter plus de contraintes sur différentes tables, notamment la table compte qui pourtant n'a aucun attribut affiché :

```
SELECT client.nom, client.prenom,
       operation.montant, operation.informations
FROM client, operation, compte
WHERE client.idclient=compte.idproprietaire
  AND compte.idcompte=operation.idcompte
  AND operation.montant>=0
  AND operation.montant<=100
        AND compte.type="Livret A"
        AND operation.informations="Guichet"
```

4 Exploiter les résultats

4.1 Tri des données

Le langage SQL offre de nombreuses possibilités. On peut par exemple trier les résultas obtenus. Graphiquement, dans l'interface, cela se fait en cliquant sur le nom d'une colonne.

- Choisir une des requêtes précédemment utilisées, l'exécuter et cliquer sur l'entête d'une des colonnes. Comment évolue la requête ?

La modification se fait en fin de requête avec les mots clef `ORBER BY attribut ASC` *signifiant que l'on trie selon l'attribut dans l'ordre ascendant.*

- Cliquer de nouveau sur l'entête : comment évolue la requête ?

le terme ASC est transformé en DESC : ordre descendand.

- Quelle est la syntaxe pour effectuer un classement du résultat d'une requête ?
- Écrire une requête, et la tester, permettant de trier la liste selon un autre critère au choix.

4.2 Renommage

Le langage SQL permet par ailleurs de créer des alias, c'est-à-dire de donner des noms aux différents éléments manipulés. Exécuter par exemple la requête suivante :

```
SELECT cl.nom,cl.prenom,c.idcompte   #attributs a afficher renommes
FROM client as cl, compte as co      #tables necessaires renommees
WHERE cl.idclient=co.idproprietaire  #jointure
  AND co.type='Livret A'             #contrainte
```

Son résultat est le même que celui d'une requête précédente. On a simplement gagné en confort d'écriture. Grâce au renommage, il est possible d'utiliser deux fois la même table. Par exemple, si l'on veut afficher les couples de noms de personnes ayant le même prénom – cette requête doit afficher 6 résultats :

```
SELECT cl1.nom, cl2.nom,cl1.prenom  #attributs a afficher
FROM client as cl1, client as cl2   #on utilise 2 fois la table client
WHERE cl1.prenom = cl2.prenom       #jointure
  AND cl1.nom != cl2.nom            #contrainte
```

Le renommage permet notamment de renommer les colonnes de résultat. On peut par exemple lancer la requête :

```
SELECT cl1.nom as nom1,             #attributs a afficher
       cl2.nom as nom2,
       cl1.prenom as prenom
FROM client as cl1, client as cl2   #on utilise 2 fois la table client
WHERE cl1.prenom = cl2.prenom       #jointure
  AND cl1.nom != cl2.nom            #contrainte
```

4.3 Agrégation

Le langage SQL propose des fonctions dites d'agrégation qui permettent notamment d'effectuer des calculs sur les résultats. Nous allons nous concentrer sur les fonctions d'agrégation qui sont explicitement au programme : min, max, somme, moyenne, comptage.

Une fonction d'agrégation prend en argument l'ensemble des résultats d'une requête et renvoie, dans notre cas, un réel. Le résultat sera donc a priori une table possédant une unique ligne.

4.3.1 Compter le nombre de résultats

La fonction `COUNT` permet de calculer le nombre de résultats et de l'afficher. Le `COUNT` se positionne dans la clause `SELECT` comme suit :

```
SELECT COUNT(*)
FROM compte
WHERE type='Livret A'
```

Il y a 72 comptes de type Livret A. Si l'on veut maintenant savoir le nombre de comptes par proprietaire, la démarche est un peu différente. Par exemple si l'on veut connaître le nombre de comptes du client 42 :

```
SELECT compte.idproprietaire , COUNT(*)
FROM compte
WHERE compte.idproprietaire = 42
```

Mais comment connaître ce nombre pour chacun des clients ? En langage SQL, on peut grouper les résultats selon les valeurs d'un attribut.

■ Comparer les résultats des deux requêtes suivantes :

```
SELECT compte.idproprietaire
FROM compte
```

```
SELECT compte.idproprietaire
FROM compte
GROUP BY idproprietaire
```

Chaque valeur d'`idproprietaire` apparaît une unique fois : les résultats ont été regroupés selon la valeur de leur `idproprietaire`. Si l'on applique une fonction d'agrégation, elle sera calculée groupe par groupe.

■ Analyser le résultat de la requête suivante en vérifiant notamment que le résultat est cohérent pour le client 42 :

```
SELECT compte.idproprietaire,COUNT(*)
FROM compte
GROUP BY compte.idproprietaire
```

■ Modifier la requête pour avoir, à la place de l'identificateur, le nom et le prénom de chaque client (il faudra passer par une jointure). Donner aussi à la colonne de comptage un nom adapté.

```
SELECT client.nom, client.prenom ,COUNT(*) as nombrecomptes
FROM compte,client
WHERE compte.idproprietaire=client.idclient
GROUP BY compte.idproprietaire
```

4.3.2 Calcul de somme, moyenne, min et max

L'instruction `SUM` permet de sommer tous les résultats d'une colonne. On peut par exemple écrire :

```
SELECT SUM(operation.montant)
FROM operation
```

pour obtenir la somme des montants des opérations.

■ Modifier la requête pour obtenir la somme des montants des opérations positives (dépôts).

```
SELECT SUM(operation.montant)
FROM operation
WHERE operation.montant>=0
```

■ Modifier la requête pour obtenir la somme des dépôts sur des comptes de type Livret A (jointure!!).

```
SELECT SUM(operation.montant)
FROM operation,compte
WHERE operation.idcompte=compte.idcompte
  AND compte.type="Livret A"
        AND operation.montant>=0
```

■ En vous inspirant de ce qui a été fait dans les paragraphes précédents, afficher pour chaque client, identifié par son nom et son prénom, la somme des montants de ses opérations faites au guichet.

```
SELECT client.nom, client.prenom, SUM(operation.montant)
FROM client, compte,operation
WHERE client.idclient=compte.idproprietaire
  AND compte.idcompte = operation.idcompte
        AND operation.informations = "Guichet"
GROUP BY client.idclient
```

On peut par exemple effectuer un filtre sur le résultat d'une opération. Par exemple, on pourrait vouloir afficher les clients, identifiés par leur nom et prénom, dont le montant des opérations total dépasse 500. On affiche aussi cette somme lorsque c'est le cas.

■ Écrire une requête qui permet d'afficher le `nom`, `prenom` et la somme des montants des opérations de tous les clients, en renommant cette somme `somme`

```
SELECT client.nom, client.prenom, SUM(operation.montant) as somme
FROM client, compte,operation
WHERE client.idclient=compte.idproprietaire
  AND compte.idcompte = operation.idcompte
GROUP BY client.idclient
```

Si l'on veut effectuer un filtrage sur le résultat d'une opération sur des groupes, il faut ajouter, en fin de requête, une clause `HAVING`. La clause `HAVING` ne peut apparaître qu'après la constitution de groupe (*i.e.* le `GROUP BY` : elle sert de filtrage sur les groupes comme le `WHERE` sert de filtrage sur les attributs. On ne peut effectuer un filtrage de type `HAVING` que sur une valeur sur laquelle on a construit le regroupement ou bien sur une valeur agrégée.

■ Ajouter en fin de requête précédente la ligne : `HAVING somme>500`. On devrait obtenir 20 résultats.

■ Que se passe-t-il si l'on décide de mettre la clause `HAVING` avant la clause `GROUP BY` ?

Cela ne marche tout simplement pas, la syntaxe n'étant pas respectée.

La fonction `AVG` fonctionne de la même façon que la fonction `SUM` mais calcule la moyenne des valeurs à la place de la somme de valeurs. Il en est de même pour la fonction `MIN` et la fonction `MAX`. Vous pouvez reprendre ce qui vient d'être fait avec ces fonctions.

■ D'après vous, que fait la requête suivante ?

```
SELECT op.idop,op.montant as maximum
FROM operation as op
WHERE op.montant = (SELECT MAX(op2.montant)
                    FROM operation as op2)
```

La tester et comprendre son comportement.

Entre parenthèse, on obtient une table à une ligne et une colonne contenant le montant maximal d'une operation. On affiche donc toutes els opérations dont le montant est égal à ce montant maximal.