TP n° 15 – Bases de données 2

Tous les fichiers utiles sont placés sur le serveur de base de données. On se connecte au serveur de base de données comme expliqué au TP précédent.

I Prénoms

La base de données `prenoms` contient une table `prenoms` contenant les prénoms enregistrés à l'état civil de Paris depuis 2004 jusqu'en 2013.

1. Écrire sur papier le schéma relationnel de la table de cette base.
2. Pour chacune des requêtes SQL suivantes, donner la traduction « en français » et vérifier la vraisemblance du résultat depuis MySQL Workbench [1].
   (a) `SELECT DISTINCT prenom FROM prenoms ;`
   (b) `SELECT SUM(nombre) FROM prenoms ;`
   (c) `SELECT COUNT(DISTINCT prenom) FROM prenoms ;`
   (d) `SELECT annee, SUM(nombre) FROM prenoms GROUP BY annee ;`
3. Combien de fois votre prénom a-t-il été donné à Paris pendant les années concernées ? On donnera le résultat trié par années croissantes.
4. Même question avec le prénom des professeurs présents.
5. Donner, pour chaque année, le nombre de prénoms différents qui ont été enregistrés.
6. (a) Quels prénoms ont été donnés exactement 100 fois sur au moins une année ?
   (b) Quels prénoms ont été donnés exactement 100 fois sur la période complète ?
7. (a) Que fait la requête suivante ?

```
SELECT MAX(nombre) as maxi, annee FROM prenoms GROUP BY annee
```

   On peut donner un nom (ou alias) à la table qui est créée par la requête précédente. Ainsi :

```
(SELECT MAX(nombre) as maxi, annee FROM prenoms GROUP BY annee) liste_max
```

   permet de nommer `liste_max` la table issue de la requête entre parenthèses. En procédant ainsi on peut l'utiliser dans une requête plus complexe.

   Par exemple, à l'aide d'une requête en jointure utilisant les tables `prenoms` et `liste_max`, on peut déterminer le prénom qui a été le plus donné pour chaque année (en affichant le prénom, l'année et le nombre) :

```
SELECT P.annee, P.prenom, P.nombre FROM
prenoms P JOIN (SELECT MAX(nombre) as max, annee FROM prenoms GROUP BY annee) liste_max
ON liste_max.annee=P.annee
WHERE liste_max.max=P.nombre
ORDER BY P.annee;
```

   (b) En vous inspirant de l'exemple précédent, déterminer le prénom féminin qui a été le plus donné pour chaque année (donner le prénom, l'année et le nombre) ?
   (c) Analyser et comprendre ce que fait la requête suivante :

```
SELECT prenom, somme
FROM (SELECT prenom, SUM(nombre) AS somme FROM prenoms GROUP BY prenom) liste_somme
WHERE liste_somme.somme=(
SELECT MAX(somme) FROM (SELECT prenom, SUM(nombre) AS somme
FROM prenoms GROUP BY prenom) liste_somme
)
```

   Traduire le but de cette requête en français.

1. Voir Annexe pour les fonctions d'agrégation `COUNT` et `SUM` et la commandes `GROUP BY`.

II Notes de colles

La base de données `colles` contient trois tables (`colles`, `eleves`, `profs`) décrivant les colles virtuelles données par des agrégés de la promotion 1930 à des agrégés de la promotion 1950.

II.1 Structure de la base de données

1. Écrire le schéma relationnel des différentes tables. Comprendre le lien entre les différentes tables.

II.2 Requêtes élémentaires

2. Déterminer la liste des professeurs.
3. Déterminer celle des élèves.
4. Déterminer le nombre de « 20 » qui ont été attribués, ainsi que le nombre de notes majorant 6.

II.3 Requêtes avec jointures

5. Déterminer les notes de Jacques-Louis Lions (triées selon les semaines croissantes).
6. Refaire la même chose avec cette fois le nom des colleurs associés.
7. Déterminer les quadruplets (élève, prof, note, semaine) pour toutes les colles où la note était supérieure ou égale à 19.

II.4 Utilisation d'agrégats

8. Déterminer la moyenne des notes de colle de Jacques-Louis Lions.
9. Déterminer la liste des couples (élève, moyenne).
10. Analyser et essayer de comprendre ce que fait la requête suivante :
```
SELECT eleves.nom, COUNT(*) AS nombre_de_plantages
FROM eleves JOIN colles ON ide=eleve
WHERE note < 10
GROUP BY eleves.nom
HAVING nombre_de_plantages > 10
```
Traduire le but de cette requête en français.
11. Parmi tous les élèves, afficher ceux ayant eu au moins 6 notes strictement supérieures à 18.
12. Déterminer le nom du colleur qui tend à donner les meilleurs notes en moyenne.
13. (a) Déterminer pour chaque colleur la moyenne des notes données et l'écart-type. On utilisera cette expression de la variance : $\sigma = \sqrt{\frac{1}{n}\left(\sum_{i=1}^{n} x_i^2\right) - \bar{m}^2}$ où $\bar{m}$ est la moyenne.

    (b) Quel est le colleur qui donne le plus souvent les mêmes notes ?
14. Écrire (sans tricher) une requête permettant de calculer la moyenne des moyennes des élèves.

III Annexe

III.1 Fonction d'agrégation

Dans tous les exemples on utilise la table nommée `donnees`, ci -dessous :

attribut1	attribut2	attribut3
0	0	0
1	1	20
2	2	40
3	2	60

III.1.1 Définitions

On appelle *fonction d'agrégation*, une fonction qui s'applique sur un groupe de valeurs (appelé *agrégat*). Les différentes fonctions d'agrégation sont :

- `COUNT(agregat)` : permet de compter le nombre d'éléments de l'agrégat ;
- `MAX(agregat)`et `MIN(agregat)` : permettent de déterminer le maximum, le minimum de l'agrégat ;
- `AVG(agregat)` : permet de déterminer la moyenne de l'agrégat ;
- `SUM(agregat)` : permet de déterminer la somme de l'agrégat ;

Les agrégats sont la plupart du temps les valeurs d'un attribut de la table (c'est-à-dire une colonne), sauf pour `COUNT()` qui peut s'appliquer à la totalité de la table (`COUNT(*)`).
Exemple : `SELECT AVG(attribut1) FROM donnees` renvoie la moyenne de toutes les valeurs de l'attribut 1.

III.1.2 GROUP BY

Il est possible d'appliquer les fonctions sur des sous-agrégats plus petits qu'un attribut en groupant certaines valeurs sur un critère donné grâce à `GROUP BY critere`. Si le critère est un attribut (une colonne), alors les sous-agrégats sont formés par valeurs distinctes de cet attribut. Si le critère est une condition, alors les sous-agrégats sont (dans l'ordre booléen) les lignes qui ne respectent pas la condition et celles qui la respectent.
`SELECT AVG(attribut3) FROM donnees GROUP BY attribut2=0` renvoie :

AVG(attribut3)
40
0

`SELECT AVG(attribut3) FROM donnees GROUP BY attribut2` renvoie :

AVG(attribut3)
0
20
50

Pour mieux expliciter le résultat, on peut aussi demander d'afficher l'attribut2, qui sert à sélectionner les sous-agrégats.
`SELECT attribut2, AVG(attribut3) FROM donnees GROUP BY attribut2` renvoie :

attribut2	AVG(attribut3)
0	0
1	20
2	50

III.1.3 Sélection en amont et en aval

On peut restreindre les agrégats sur lesquels s'appliquent les fonctions d'agrégation, par la clause `WHERE critere` (sélection en amont, c'est-à-dire avant de réaliser la fonction d'agrégation) ou bien restreindre ce qu'affiche la fonction d'agrégation par la clause `HAVING critere` (sélection en aval, c'est-à-dire après avoir réalisé la fonction d'agrégation). `HAVING` doit toujours être situé après un `GROUP BY`.

Sélection en amont La sélection s'effectue avec la clause `WHERE` sur les lignes initiales de la table avant de réaliser l'agrégation.
`SELECT AVG(attribut3) FROM donnees WHERE attribut2=1` renvoie

AVG(attribut3)
20

Sélection en aval La sélection s'effectue avec la clause `HAVING` sur les lignes qui restent après la réalisation de l'agrégation, ce qui n'a de sens que si on a effectué au préalable un regroupement grâce à `GROUP BY`.
`SELECT AVG(attribut3) FROM donnees GROUP BY attribut2 HAVING attribut2=2` renvoie

AVG(attribut3)
50

III.2 ORDER BY

La clause `ORDER BY attribut` permet de trier les résultats de la requête selon un attribut donné. Par défaut le tri est ascendant. L'option `DESC` permet de trier par ordre descendant.
`ORDER BY nom` trie la requête par ordre ascendant des valeurs de l'attribut `nom`
`ORDER BY annee DESC` trie la requête par ordre descendant des valeurs de l'attribut `annee`
Elle est toujours située à la fin de la requête.