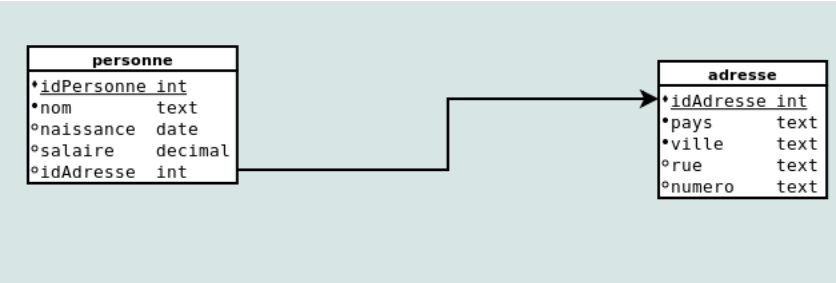
Bases de Données
cours / TD

Lycée Charlotte Perriand — Terminale NSI

INTRODUCTION À SQL

1 Cours

Les exemples de requêtes qui suivent, considèrent les deux tables suivante :



- Chaque personne possède obligatoirement un nom.
- On peut également connaitre sa date de naissance, son salaire, et son adresse, mais ces données sont facultatives : il est possible qu'elles ne soient pas connues.
- Chaque adresse est composée au moins d'un nom de pays et d'un nom de ville, elle peut également comprendre un nom de rue et un numéro.
- Il est possible que certaines adresses ne soient reliées à aucune personne.

1.1 Requêtes simples

1.1.1 Définition

Pour extraire des information depuis les tables d'une base de données relationnelle, on utilise des *requêtes SQL* sous la forme suivante :

```
SELECT expression [,...]
FROM TABLE
WHERE condition;
```

1.2 Jointures

1.2.1 Définition

L'opération de jointure permet de réaliser le lien entre plusieurs tables. On peut effectuer la jointure entre deux tables de la manière suivante :

```
SELECT expression [,...]
FROM table1 JOIN table2 ON table1.colonneX=table2.colonneY;
```

`colonneX` est une colonne de la `table1` et `colonneY` une colonne de la `table2`. `colonneX` et `colonneY` représentent des données "compatibles" : en général elles couvrent la même réalité.

Une manière équivalente d'écrire une opération de jointure est la suivante :

```
SELECT expression [,...]
FROM table1,table2
WHERE table1.colonneX=table2.colonneY;
```

1.2.2 Exemples

Pour extraire les noms des personnes et leur ville, on écrira l'une des deux requêtes suivantes :

```
SELECT personne.nom, adresse.ville
FROM personne JOIN adresse ON personne.idAdresse=adresse.idAdresse
```

ou encore :

```
SELECT personne.nom, adresse.ville
FROM personne, adresse
WHERE personne.idAdresse=adresse.idAdresse
```

1.3 Groupements

1.3.1 Définition

La clause `GROUP BY expression` permet de grouper les lignes ayant la même valeur pour l'expression. Une expression est soit une colonne, soit une expression appliquée à une ou plusieurs colonnes. La clause `GROUP BY` apparaît à la suite de la clause `WHERE` lorsqu'il y en a une. L'écriture générale d'une requête avec regrouprement est donc :

```
SELECT [expression,] [fonction [,...]]
FROM table1
[WHERE expression]
GROUP BY expression
```

Le résultat d'une requête avec groupement fait apparaître autant de lignes qu'il y a de valeurs différentes pour l'expression.

Remarques :

- Généralement, l'expression qui apparait dans le `GROUP BY`, apparait également dans la partie `SELECT` mais ce n'est pas une obligation ;
- Dans la partie SELECT, on trouvera l'expression qui apparait dans le `GROUP BY`, et des expressions utilisant des fonctions d'agrégat (MIN, MAX, AVG, SUM, COUNT, ...).

La clause `WHERE` permet de sélectionner des lignes des tables qui apparaissent dans la partie `FROM`. La clause `HAVING` permet de sélectionner les groupes issus de de la partie `GROUP BY`, elle suit directement la clause `GROUP BY` :

```
SELECT [expression,] [fonction [,...]]
FROM table1
[WHERE expression]
GROUP BY expression
HAVING condition
```

Une condition exprimée dans la partie `HAVING` utilise toujours une fonction d'agrégat.

1.3.2 Exemples

Pour connaitre le nombre d'adresses dont on dispose dans chaque ville, on va regrouper les adresses par ville, et afficher pour chaque groupe ainsi constitué, le nombre de lignes qui le composent.

```
SELECT ville, COUNT(*)
FROM adresse
GROUP BY ville
```

Pour connaitre le salaire moyen dans chaque ville, il faudra joindre les deux tables avant d'effectuer un groupement sur les villes :

```
SELECT ville, AVG(salaire)
FROM adresse JOIN personne ON adresse.idAdresse=personne.idAdresse
GROUP BY ville
```

Pour se limiter aux villes de France, on ajoute une sélection sur le pays. Autrement dit, on sélectionne les lignes avant d'effectuer le regroupement :

```
SELECT ville, AVG(salaire)
FROM adresse JOIN personne ON adresse.idAdresse=personne.idAdresse
WHERE pays='France'
GROUP BY ville
```

Pour se limiter aux villes dans lesquelles on trouve plus de 1000 personnes, on ajoute une sélection sur les groupes constitué. Autrement dit, on constitue les groupes, puis on les sélectionne :

```
SELECT ville, AVG(salaire)
FROM adresse JOIN personne ON adresse.idAdresse=personne.idAdresse
GROUP BY ville
HAVING COUNT(*)>1000
```

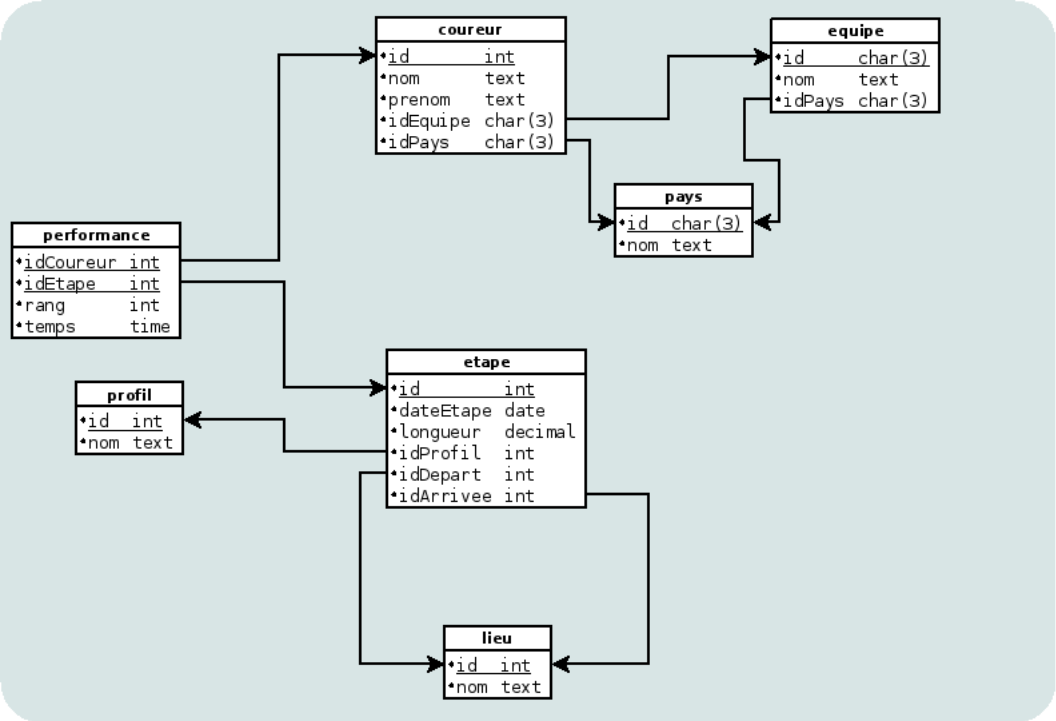
Au final, on peut combiner les deux types de sélection pour obtenir les salaires moyens pour chaque ville française dans laquelle on compte plus de 1000 personnes :

```
SELECT ville, AVG(salaire)
FROM adresse JOIN personne ON adresse.idAdresse=personne.idAdresse
WHERE pays='France'
GROUP BY ville
HAVING COUNT(*)>1000
```

2 Exercices : requêtes SQL

Les exercices qui suivent portent sur une base de données qui contient la description des étapes du tour de France cycliste, les informations concernant les coureurs et leurs résultats aux différentes étapes.

Le modèle physique de la base de données est le suivant :



2.1 Requêtes simples

Ecrivez Les requêtes SQL permettant d'afficher les informations suivantes :

1. Les coureurs dont l'identifiant de pays est "FRA".
2. Les performances de moins de $1h30$
3. Le nom des équipes dont l'identifiant de pays n'est pas "FRA".
4. L'id des coureurs qui ont gagné une étape
5. Les noms de profil qui contiennent la lettre "i".

2.2 Jointures

Ecrivez Les requêtes SQL permettant d'afficher les informations suivantes :

1. Les coureurs des équipes dont l'identifiant de pays est "FRA".
2. Les coureurs des équipes dont l'identifiant de pays est "FRA", et leurs performances(rang et temps) au cours de l'étape n°1.
3. Le nom du vainqueur de l'étape n°1, et le nom de son équipe.
4. La liste des étapes : n° d'étape, nom des villes de départ et d'arrivée.
5. Les coureurs (prénom et nom) qui ont le même nom de famille.

2.3 Groupements

Ecrivez Les requêtes SQL permettant d'afficher les informations suivantes :

1. Les id de profil, et le nombre d'étapes de chacun.
2. Le nom des équipes et le nombre de coureurs de chacune.
3. Les noms des coureurs, avec leur meilleur rang, leur temps total et le nombre d'étapes auxquelles ils ont participé.
4. Les noms des équipes, avec le temps total de leurs coureurs.
5. Les noms des équipes, et le rang de leur meilleur coureur au cours de l'étape n°1.

2.4 Groupements & sélections

Ecrivez Les requêtes SQL permettant d'afficher les informations suivantes :

1. L'id et le nom des coureurs qui ont terminé le tour de France, ainsi que leur temps cumulé la fin de l'épreuve.
2. Les noms des équipes qui comptent encore 9 coureurs à la fin de l'étape 10.
3. Les noms des coureurs qui ont fini au moins deux fois dans les 5 premiers.
4. Les noms des profils qui correspondent à plus de 3 étapes.
5. Les noms des équipes qui comptent plus de 3 coureurs français.

2.5 Tri des résultats

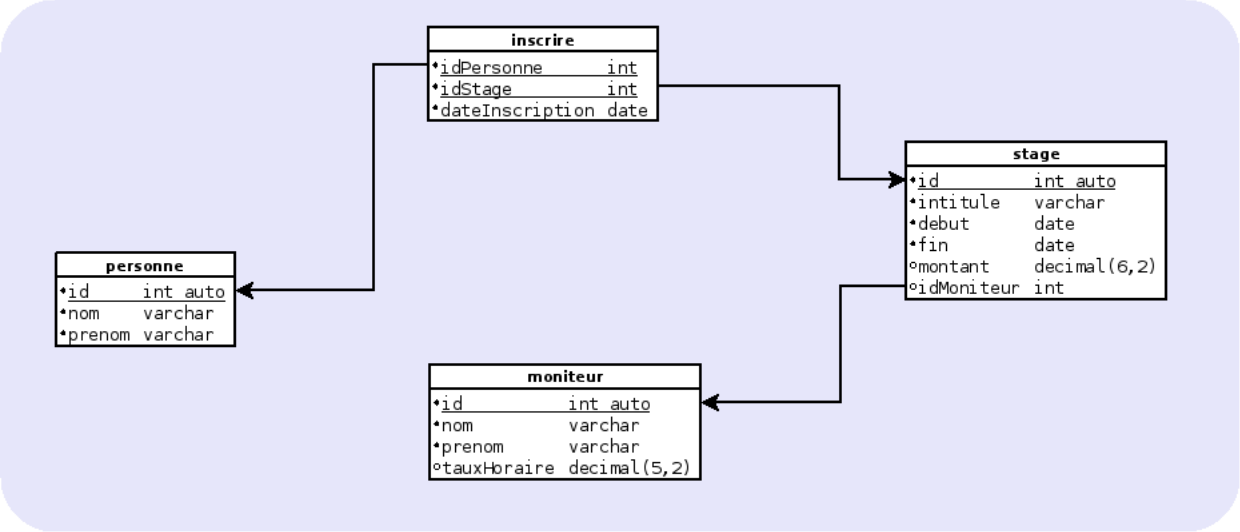
Ecrivez Les requêtes SQL permettant d'afficher les informations suivantes :

1. Le nom et le prénom des coureurs dans l'ordre lexicographique du nom puis du prénom.
2. L'id des étapes, le nom de leur profil, et leur date, dans l'ordre descendant des noms de profil puis ascendant des dates.
3. Le nom des pays et le nombre de coureurs pour chacun, triés depuis le pays qui dispose du plus grand nombre de coureurs à celui qui en dispose du plus petit nombre. En cas d'égalité, on triera dans l'ordre lexicographique inverse.
4. Le classement général(id, nom et prénom des coureurs) à l'issue de l'étape 10 (attention aux abandons).
5. Le nom des coureurs qui n'ont jamais figuré dans les 100 premiers, avec leur meilleur rang. On affichera les coureurs dans l'ordre de ce rang. En cas d'égalité on triera les coureurs en fonction de leur rang moyen.

3 Exercices : gestion des données

3.1 Création des tables

Ecrivez le script SQL permettant de générer les tables correspondant au modèle suivant :



3.2 Insertion des données

Insérez les données suivantes dans les tables :

1.

personne	
nom	prénom
Dubois	Jacques
Legrand	Lise
Dupré	Kévin
Leroux	Maria

2.

moniteur		
nom	prénom	taux horaire
Martin	Mireille	13.5
Lebleu	Alex	12.8
Legris	Naomi	13.5
Delval	Max	12.8

3.

stage				
intitulé	début	fin	montant	id moniteur
Tennis	17 mai 2014	25 mai 2014	250,00	1
Squash	4 juin 2014	11 juin 2014	148,80	1
Volley	8 juin 2014	17 juin 2014	200.00	2
Tennis	30 mai 2014	12 juin 2014	240.00	3
Squash	14 avril 2014	30 avril 2014	210.00	2
Volley	13 juin 2014	17 juin 2014	150,55	3

4.

inscrire		
idpersonne	idstage	date d'inscription
1	1	14 février 2014
1	3	17 mars 2014
2	4	25 avril 2014
3	5	07 février 2014

3.3 Modification des données

Effectuez les opérations suivantes en utilisant une commande SQL *UPDATE*. Il est possible que certaines commandes soient rejetées par le SGBD, auquel cas vous devez vérifier (et surtout comprendre) pourquoi.

1. Le prénom de la personne n°1 est *André*.
2. Le nom du moniteur n°2 est *Lerouge*.
3. Changez le prénom de la personne n°1 : *Jordan*.
4. Changez le nom du moniteur n°2 : *Levert*.
5. Affectez la valeur 1 à tous les id de moniteurs qui apparaissent dans la table *stage*.
6. Affectez la valeur 1000 à tous les id de moniteurs qui apparaissent dans la table *stage*.
7. Affectez une valeur aléatoire à tous les id de moniteurs qui apparaissent dans la table *stage* (voir *http ://dev.mysql.com/doc/refman/5.0/en/mathematical-functions.html*).
8. La date d'inscription de M. Dupré est le 17 février 2014.
9. Reculez d'une journée la date dinscription de M. Dupré.
10. Reculez d'une journée les dates de début et de fin de tous les stages de tennis.
11. Faites en sorte que les noms de personnes et le noms de moniteurs soient écrits en majuscule dans les tables.
12. Effacez tous les id de moniteurs qui apparaissent dans la table *stage*.

3.4 Suppression des données

Effectuez les opérations suivantes en utilisant une commande SQL *DELETE*. Il est possible que certaines commandes soient rejetées par le SGBD, auquel cas vous devez vérifier (et surtout comprendre) pourquoi.

1. Supprimez la personne n°1.
2. Supprimez les inscriptions de la personne n°1.
3. Supprimez le moniteur n°4.
4. Supprimez la personne n°2.
5. Supprimez le moniteur dont le nom est *Martin*.
6. Supprimez toutes les inscriptions.
7. Supprimez tous les stages qui débutent en juin.
8. supprimez les stages qui durent moins de 5 jours.
9. Supprimez toutes les données qui sont encore dans les tables.