Lycée Lakanal

2025- 2026

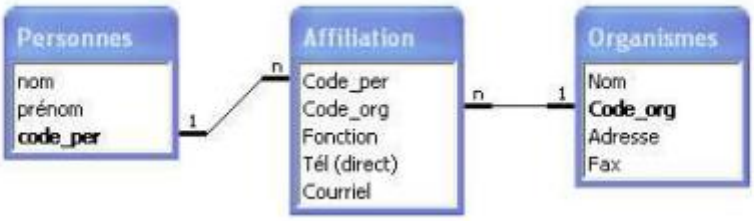
PC

# TD 1 – Bases de données

## *Introduction*

On fait appel aux bases de données lorsque la gestion des informations sous forme d'une unique tableau devient trop fastidieuse. Imaginons par exemple une grande entreprise, disposant de nombreux départements : service clients, service marketing, comptabilité, ressources humaines… chacun d'entre eux a ses propres données, qu'il peut stocker dans son propre tableau (que nous appellerons ici : table). Il est souvent nécessaire de croiser des informations provenant de plusieurs de ces tables : c'est le rôle d'une base de données de permettre de le faire. Une base de données est donc un regroupement de tables. Pour pouvoir croiser les données, il est nécessaire qu'elles comportent ce que l'on appelle des clés primaires : une clé primaire est un attribut d'une table permettant d'identifier de façon unique une entrée (une ligne), d'une table : par exemple, le service client va attribuer à chaque client un numéro qui lui est propre : ce numéro sera une clé primaire de la table client. Le service des ventes, lui, notera que l'achat N°2010213 a été effectué parle client N° 3546. Ce service des ventes n'a aucune donnée sur les clients, leur numéro mis à part. Mais, en croisant les deux tables, on pourra par exemple trouver le nom et l'adresse du client ayant effectué l'achat N°2010213. Dans cet exemple, on dira que le numéro du client est une clé étrangère dans la table Ventes : c'est une colonne qui renvoie à la clé primaire d'une autre table. Dresser le schéma relationnel d'une base de données, c'est décrire les données contenues dans ses différentes tables, en indiquant quelles sont les clés primaires et étrangères que l'on y trouve. Il est important, pour pouvoir gérer sereinement cette base de données, de commencer par donner son schéma relationnel. Par exemple :

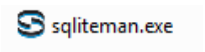
**Modèle relationnel d'une base de données associant du personnel à des organisations**



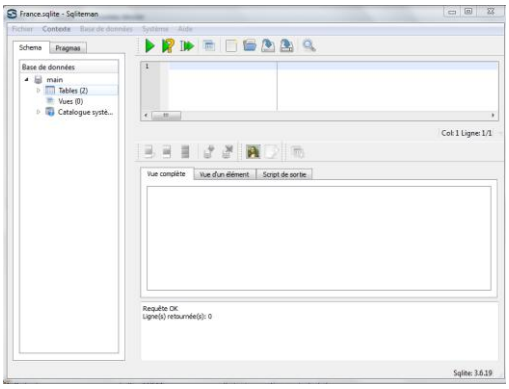

Ici, la table personnes contient 3 colonnes (nom, prénom, cod_per). Code_per est une clé primaire de cette table : chaque personne a un code unique, qui lui est propre. Ce code se retrouve dans la table Affiliation, c'est donc une clé étrangère pour cette table.

## *Démarrage*

**1.** Allez dans la zone d'échange de votre classe, puis dans le dossier informatique, puis bases_de_données,
Puis sqliteman. Cliquez enfin sur l'icône :




**2.** La fenêtre suivante apparaît :



Sélectionner "Fichier", "ouvrir" (ou CTRL+O), et indiquer le dossier où trouver votre base de données, à savoir la zone d'échange de la classe, dossier informatique, puis bases_de_donnees et enfin bdd.

Sélectionnez alors la base de données movies.

## Exercice 1

Parcourir les différentes tables de cette base de données, et dresser son schéma relationnel.

Dans les exercices 2 et 3, on travaillera avec la base de données France, que l'on ouvrira suivant la même procédure que ci-dessus. Nous n'utiliserons que sa table Villes2, dont le schéma relationnel est le suivant :

### Villes2

| Attribut | Type |
|---|---|
| Numéro | Entier |
| Dept | Chaîne de caractères *merci la Corse* ! |
| Nom | Chaîne de caractères |
| Code postal | Chaîne de caractère (*pourquoi* ?) |
| Pop_2010 | Entier |
| densité | Réel |

Clé primaire

## Exercice 2

**On répondra aux questions ci-dessous en écrivant des « requêtes » (c'est-à-dire des instructions), sur le modèle de celles données en annexe. On doit répondre à chaque question ou sous-question grâce à une seule et unique requête.**

1. Y a – t – il des villes de densité nulle ? Sont – ce les mêmes que celles qui avaient une population nulle en 2010 ? Explication ?
2. Déterminer le nombre de villes pour chaque département.
3. Donner la population de chaque département.
4. Donner une estimation de la superficie de la ville de Paris (vous devez trouver environ $105\ km^2$).
5. Sortir les données de la ville (ou des villes) de densité (non nulle) minimale.
6. Donner la liste des noms et la population 2010 des villes dont le nom commence par B ou dont la population 2010 était supérieure à 50 000 .

## Exercice 3

1. Que fait la stupide requête suivante ? Qu'aurait – on pu écrire d'autre, avec le même résultat ?

```
SELECT Nom , Pop_2010 FROM villes2
WHERE Nom IN
```

(SELECT Nom FROM villes2
WHERE Nom LIKE 'B%')

**2.** Donner la liste des villes dont la population est supérieure (ou égale) à celle de toutes les villes dont le nom commence par A .

**3.** On souhaite déterminer la population moyenne (en 2010) de l'ensemble des villes dont le nom commence par 'A', département par département.
Que renvoient les requêtes suivantes ?

```
SELECT AVG(nb) FROM
(SELECT Dept, Nom, COUNT(*) as nb
FROM villes2
WHERE Nom LIKE 'A%'
GROUP BY Dept)
As result
```
(*on reçoit pour réponse :* 19,77)

```
SELECT AVG(nb) FROM
(SELECT Dept, Nom, COUNT(Pop_2010) as nb
FROM villes2
GROUP BY Dept)
As result
```
(*on reçoit pour réponse :* 359,8039)

```
SELECT nb FROM
(SELECT Dept, Nom, AVG(Pop_2010) as nb
FROM villes2
WHERE Nom Like ('A%')
GROUP BY Dept)
As result
```
(*on reçoit pour réponse :* 1637,19)

**4.** Que fait la requête suivante ?

```
SELECT Nom , Pop_2010 FROM villes2
WHERE Nom IN
(SELECT Nom FROM villes2
WHERE Pop_2010 >50000)
```
*On pourra comparer son résultat avec celui de l'instruction suivante :*

```
SELECT Nom , Pop_2010 FROM villes2
WHERE Pop_2010 >50000
```

---

## Exercice 4

On s'occupe ici de la base de données mondial, située au même endroit que la base précédente. Cette base de données contient de nombreuses tables de données géographiques. Parmi celles – ci, on trouve une table nommée Country, qui possède 6 attributs :

Name Code Capital Province Area Population.

Les quatre premiers sont des chaînes de caractères, le cinquième un nombre flottant et le sixième un entier. Le code du pays est une clé primaire de la table, son unicité est donc garantie.

1. Rédiger une requête SQL permettant d'obtenir la liste des noms de pays dont la population excède 60 000 000 d'habitants.
2. Rédiger une requête SQL permettant d'obtenir la même liste, triée par ordre alphabétique.
3. Rédiger une requête SQL permettant d'obtenir la même liste, triée par ordre décroissant de population.
4. Rédiger une requête SQL permettant d'obtenir la liste des noms dix plus petits pays (en terme de surface).
5. Rédiger une requête SQL permettant d'obtenir la liste des quinze suivants.
6. En utilisant les tables Country et Island, dresser la liste des noms de pays ou d'iles (*une seule liste,merci !*).
7. Dresser la liste des noms des pays ayant le même nom qu'une île.

---

## Exercice 5

On revient à la base mondial. On y on trouve une table nommée encompasses possédant trois attributs :

Country Continent Percentage.

Le premier attribut est le code du pays, le deuxième le nom du continent, et le dernier la portion du pays présente sur le continent. La clé primaire de cette table est le couple (Country, Continent), et la valeur du troisième argument ne peut pas être nulle. Cette seconde table possède un attribut en commun avec la première table : l'attribut Country de la table encompasses renvoie en effet à l'attribut Code de la table country (c'est une clé étrangère pour la table encompasses). Ceci va nous permettre de croiser les informations de ces deux tables.

**Q1.** Rédiger une requête SQL permettant d'obtenir la liste des pays dont une partie au moins du territoire est en Europe.

**Q2.** Rédiger une requête SQL permettant d'obtenir le nombre de pays qui sont à cheval sur plusieurs continents.

**Q3.** Rédiger une requête SQL permettant d'obtenir (super intéressant) le nombre moyen d'habitants des pays du continent américain qui comptent moins de dix habitants par kilomètre carré.

---

## Exercice 6

Dans la même base de données figure une table nommée city qui possède les attributs suivants :

Name Country Province Population Longitude Latitude.

L'attribut Coutry est toujours le code du pays : c'est, à nouveau, une clé étrangère pour la table city.

1. Déterminer les capitales européennes situées à une latitude supérieure à 60, ainsi que les noms des pays correspondants.
   Cela commence à se compliquer… il faut croiser les tables country, city et encompasses :

2. La table Language posssède les attributs suivants :

   Country Name Percentage.

   L'attribut Country est le code du pays, Name le nom d'une langue parlée dans ce pays, et Percentage le pourcentage d'habitants dont c'est la langue maternelle.

   **a.** Donner la liste ordonnée (dans le sens décroissant), des 10 langues parlées dans le plus de pays.
   **b.** Quelles sont, suivant cette base de données, les 5 langues les plus parlées dans le monde ? On précisera pour chacune d'entre elles le nombre de personnes qui la parlent.
   **c.** Quel est le pays partiellement francophone dans lequel la langue française est la plus minoritaire ?
   **d.** Quelle est le pourcentage de personnes francophones dans le monde ?

***Nous ne travaillerons dans tous les exercices qui suivent qu'avec une seule base de données : la base de données eph2, qui se trouve à l'endroit habituel.***

---

## Exercice 7

Parcourir les différentes tables, puis donner le schéma relationnel de cette base de données.

## Exercice 8

On ordonne les corps par masse décroissante. Parmi les corps suivants : Titan, Terre, Lune, Ganymede, quels sont ceux dont le rang est situé entre la 13ème et la 22ème position ?

## Exercice 9

Donner la liste des noms des satellites de Neptune présents dans cette base de données.

## Exercice 10

Deux corps n'ont aucune donnée présente dans la table etat. Quel est leur nom ?

## Exercice 11

De quelle(s) planète(s) les deux corps ci – dessus sont – ils satellites ?

## Exercice 12

On néglige les deux corps ci – dessus. Quelle est (s'il y en a une), la première date à laquelle tous les corps ont des positions établies dans la table etat ?

## Exercice 13

Donner le nombre d'entrées enregistrées dans la table etat concernant le système planétaire de Neptune.
Si vous êtes joueurs, vous pouvez prolonger en donnant le nom de la planète dont le système planétaire détient le record d'entrées enregistrées dans la table etat,

# ANNEXE : QUELQUES ÉLÉMENTS DE SYNTAXE SQL

| **SELECT … FROM** | |
|---|---|
| SELECT * FROM Truc | Affiche toute la table Truc. |
| SELECT Bidule FROM Truc | Affiche la colonne bidule de la table Truc. |
| SELECT Bidule, machin FROM Truc | Affiche les colonnes machin et bidule de la table Truc. |

| **SELECT … FROM … WHERE** | |
|---|---|
| SELECT * FROM Villes2<br>WHERE Pop_2010>100000 | Affiche toutes les entrées (lignes) de la table Villes2 correspondant à des villes dont la population était supérieure à 100000 habitants en 2010 . |
| SELECT Nom FROM Villes2<br>WHERE Pop_2010>100000<br>AND NOT Nom LIKE 'P%' | Affiche les villes dont la population était supérieure à 100000 habitants en 2010 et dont le nom ne commence pas par la lettre P. |

| **SELECT … FROM … ORDER BY** | |
|---|---|
| SELECT * FROM Villes2<br>ORDER BY POP_2010 | Affiche les lignes de la table Villes2 ordonnées par population 2010 croissante (*par défaut, c'est l'ordre croissant qui est adopté*). |
| SELECT * FROM Villes2<br>ORDER BY POP_2010 DESC | Affiche les lignes de la table Villes2 ordonnées par population 2010 décroissante. |
| SELECT * FROM Villes2<br>ORDER BY POP_2010 DESC, Nom ASC | Affiche les lignes de la table Villes2 ordonnées par population 2010 décroissante, et, en cas d'égalité, par nom croisant (au sens lexicographique). |

| **LIMIT et OFFSET** | |
|---|---|
| SELECT * FROM Villes2<br>ORDER BY POP_2010 DESC LIMIT 10 | Affiche les 10 villes les plus peuplées en 2010. |
| SELECT * FROM Villes2 ORDER BY POP_2010<br>DESC LIMIT 5 OFFSET 10 | Affiche les 5 suivantes. |

| **FONCTION D'AGREGATION : MIN, MAX, SUM, COUNT, AVG** | |
|---|---|
| SELECT MAX(Pop_2010) FROM Villes2<br>WHERE Code_Postal LIKE '92%' | Renvoie la population de la ville des Hauts-de-Seine la plus importante. |
| SELECT COUNT(*) FROM Villes2<br>WHERE Code_Postal LIKE '%9%' | Renvoie le nombre de villes situées dans un département dont le numéro comporte le chiffre 9. |

| **GROUP BY** | |
|---|---|
| SELECT * FROM Villes2 GROUP BY Dept | Renvoie une entrée (*la première possible de la table*) par département |
| SELECT Dept, Max(Pop_2010) FROM Villes2<br>GROUP BY Dept | Renvoie la liste des départements, avec, pour chacun d'entre eux, la population de sa ville la plus peuplée. |

| **DISTINCT** | |
|---|---|
| SELECT DISTINCT TRUC FROM BIDULE | Renvoie toutes les entrées de la colonne Truc de la table Bidule, en évitant les doublons. |

| **GROUP BY ...HAVING** | |
|---|---|
| SELECT Dept, Max(Pop_2010) AS N<br>FROM Villes2 GROUP BY Dept<br>HAVING N BETWEEN 100000 AND 200000 | Contrairement à WHERE, HAVING effectue la restriction après que le regroupement a été réalisé. |
| **ALIAS : AS** | |
| SELECT TRUC AS MACHIN FROM BIDULE<br>ou<br>SELECT TRUC MACHIN FROM BIDULE | Sélectionne l'attribut (colonne) "Truc" de la table Bidule, en le renommant "Machin". Ceci est souvent nécessaire lors de sélections imbriquées.<br>Le « AS » n'est pas nécessaire, mais aide à la lisibilité. |

| **UNION** | |
|---|---|
| SELECT COL1, COL2 FROM TABLE 1<br>UNION<br>SELECT COL3, COL4 FROM TABLE 2 | Réunion des colonnes COL1, COL2 de la Table 1 et des colonnes COL3, COL4 de la Table2.<br>Les données de COL1 et COL3 doivent être du même type, de même pour les entrées de COL2 et COL4 |
| **INTERSECT** | |
| SELECT COL1 FROM TABLE1<br>INTERSECT<br>SELECT COL2 FROM TABLE2 | No comment. |

| **SELECTIONS IMBRIQUEES (Exemples)** | |
|---|---|
| SELECT AVG(N) FROM<br>(SELECT COL1 AS N FROM TABLE1<br>WHERE …)<br>WHERE … | Des alias (renommages) sont le plus souvent nécessaires lors de requêtes imbriquées du type SELECT … FROM (SELECT …) WHERE … |
| SELECT COL1 FROM TABLE 1<br>WHERE COL1><br>(SELECT AVG(COL2) FROM TABLE2<br>WHERE …) | SELECT … FROM … WHERE (SELECT …) est un autre type de sélections imbriquées. |

| **JOINTURES : JOIN … ON** | |
|---|---|
| SELECT X, Y, Z, T FROM Table 1<br>JOIN Table2, Table3<br>ON X=Z | Une jointure sans conditionnement « ON » se référant à la clé primaire de l'une des tables qui est clé étrangère pour les autres tables n'a généralement aucune utilité. |
| SELECT Table1.X, Table2.X, Table3.X, T<br>FROM Table 1 JOIN Table2, Table3<br>ON Table1.X=Table3.X | Lorsque des colonnes des différentes tables mises en jeu portent le même nom, on précise… |

| **Quelques opérateurs** | |
|---|---|
| + ; – ; / ; * | Opérateurs arithmétiques usuels. |
| = ; < ; > ; <= ; >= ; != | Opérateurs de comparaison usuels (pas de = = en SQL !). |
| NOT | Négation. |
| AND | ET |
| OR | OU |
| LIKE | X LIKE '%truc' est vrai ssi la chaîne de caractère se termine par 'Truc'.<br>X LIKE "%truc%' est vrai ssi la chaîne de caractère contient le mot 'Truc'. |