Base de données et requêtes SQL

Contexte

Les données complexes sont souvent stockées dans des bases de données relationnelles. Celles-ci distribuent les données en **tables** qui sont **interrogées**.

1 Tables, attributs, domaines

Nous allons utiliser une base de données « jouet » : plutôt que de contenir des dizaines de tables et des centaines de lignes, celles-ci aura trois tables ; chaque table aura moins de vingt lignes. Gardez en mémoire qu'il y a potentiellement des milliers de lignes dans une « vraie » BDD, ce qui rend très intéressantes les techniques développées ici.

Voici la table books :

id	author	title	words
1	J.K. Rowling	Harry Potter and the Philosopher's Stone	79944
2	J.K. Rowling	Harry Potter and the Chamber of Secrets	85141
3	J.K. Rowling	Harry Potter and the Prisoner of Azkaban	107253
4	J.K. Rowling	Harry Potter and the Goblet of Fire	190637
5	J.K. Rowling	Harry Potter and the Order of the Phoenix	257045
6	J.K. Rowling	Harry Potter and the Half-Blood Prince	168923
7	J.K. Rowling	Harry Potter and the Deathly Hallows	197651
8	Stephenie Meyer	Twilight	118501
9	Stephenie Meyer	New Moon	132807
10	Stephenie Meyer	Eclipse	147930
11	Stephenie Meyer	Breaking Dawn	192196
12	J.R.R. Tolkien	The Hobbit	95022
13	J.R.R. Tolkien	Fellowship of the Ring	177227
14	J.R.R. Tolkien	Two Towers	143436
15	J.R.R. Tolkien	Return of the King	134462

Notation

Une table a des **colonnes** appelés aussi **attributs** ou **champs** ; chaque attribut est d'un certain type appelé **domaine**.
Les **lignes** d'une table sont aussi appelées des **enregistrements**.
La valeur NULL désigne une case non remplie.

Attributs	id	author	title	words
Domaines	INT	TEXT	TEXT	INT

Autres domaines possibles : FLOAT, BOOL, DATE.

2 Structure trois tiers

Une base de données est souvent volumineuse, et sert à stocker de l'information. Extraire l'information qui nous intéresse s'appelle **interroger la base**.

On interroge une base à l'aide de commandes écrites dans le langage SQL (structured query language). Ces commandes sont appelées **requêtes SQL**.

Il y a donc deux « agents » dans cette configuration :

- La base de données qui regroupe les informations,
- Vous, qui souhaitez interroger la base.

En pratique, on ajoute souvent une interface utilisateur, ce qui donne

- La base de données qui a plein d'informations, stockée sur un serveur,
- Un serveur intermédiaire qui assure la communication entre vous et la base. Cela permet en particulier d'avoir une interface utilisateur plus conviviale, voire des fonctionnalités en plus ;
- L'utilisateur qui interroge la base.

Cette architecture à trois étages (= « tier » en anglais) se nomme une **architecture trois tiers.**

3 SELECT et WHERE

3.1 SELECT ··· FROM ···

La commande **SELECT** retourne les enregistrements de un ou plusieurs attributs dans un tableau de résultats. Cette commande permet donc de sélectionner une ou plusieurs colonnes d'une table.

Syntaxe :

```
SELECT colonne FROM Table

SELECT colonne1, colonne2, ..., colonneN FROM Table

SELECT * FROM Table
```

Remarque

Le * permet de sélectionner toutes les colonnes, sans avoir à ré-écrire tous les noms.

Exemple

```
SELECT title FROM books
```

donne le résultat :

title
Harry Potter and the Philosopher's Stone
Harry Potter and the Chamber of Secrets
Harry Potter and the Prisoner of Azkaban
Harry Potter and the Goblet of Fire
Harry Potter and the Order of the Phoenix
Harry Potter and the Half-Blood Prince
Harry Potter and the Deathly Hallows
Twilight
New Moon
Eclipse
Breaking Dawn
The Hobbit
Fellowship of the Ring
Two Towers
Return of the King

```
SELECT author, title FROM books
```

donne le résultat :

author	title
J.K. Rowling	Harry Potter and the Philosopher's Stone
J.K. Rowling	Harry Potter and the Chamber of Secrets
J.K. Rowling	Harry Potter and the Prisoner of Azkaban
J.K. Rowling	Harry Potter and the Goblet of Fire
J.K. Rowling	Harry Potter and the Order of the Phoenix
J.K. Rowling	Harry Potter and the Half-Blood Prince
J.K. Rowling	Harry Potter and the Deathly Hallows
Stephenie Meyer	Twilight
Stephenie Meyer	New Moon
Stephenie Meyer	Eclipse
Stephenie Meyer	Breaking Dawn
J.R.R. Tolkien	The Hobbit
J.R.R. Tolkien	Fellowship of the Ring
J.R.R. Tolkien	Two Towers
J.R.R. Tolkien	Return of the King

```
SELECT author FROM books
```

donne le résultat :

author
J.K. Rowling
J.K. Rowling
J.K. Rowling
J.K. Rowling
J.K. Rowling
J.K. Rowling
J.K. Rowling
Stephenie Meyer
Stephenie Meyer
Stephenie Meyer
Stephenie Meyer
J.R.R. Tolkien
J.R.R. Tolkien
J.R.R. Tolkien
J.R.R. Tolkien

Pour éviter des redondances dans les résultats il suffit d'ajouter DISTINCT après le mot SELECT.

```
SELECT DISTINCT author FROM books
```

donne le résultat :

author
J.K. Rowling
Stephenie Meyer
J.R.R. Tolkien

```
SELECT * FROM books
```

sélectionne tout et renvoie toute la table.

3.2 WHERE

3.2.1 Principe

La commande **WHERE** permet d'extraire uniquement les enregistrements d'une base de données qui vérifient une condition.

```
SELECT colonne1|colonne1, colonne2, ..., colonneN|* FROM Table
WHERE conditions logiques sur attributs
```

Exemple

```
SELECT * FROM books
WHERE author='J.K. Rowling'
```

retourne la vue :

id	author	title	words
1	J.K. Rowling	Harry Potter and the Philosopher's Stone	79944
2	J.K. Rowling	Harry Potter and the Chamber of Secrets	85141
3	J.K. Rowling	Harry Potter and the Prisoner of Azkaban	107253
4	J.K. Rowling	Harry Potter and the Goblet of Fire	190637
5	J.K. Rowling	Harry Potter and the Order of the Phoenix	257045
6	J.K. Rowling	Harry Potter and the Half-Blood Prince	168923
7	J.K. Rowling	Harry Potter and the Deathly Hallows	197651

Remarque

C'est l'analogue de la syntaxe $\{x \in E \mid P(x)\}$ lorsque E est un ensemble et $P(x)$ une assertion en la variable x.
Les conditions logiques portent sur des **attributs** (noms des colonnes), mais servent à éliminer/conserver des **enregistrements** (lignes).

3.2.2 Opérateurs dans la commande WHERE

Opérateurs **booléens** (ie Logique)	Description
OR, AND, NOT	Opérateurs logiques entre conditions
=	Égale
<>	Pas égale
!=	Pas égale
IN	Teste si Liste de plusieurs valeurs possibles
IS NULL	Valeur est nulle
IS NOT NULL	Valeur n'est pas nulle
>	Supérieur à
<	Inférieur à
>=	Supérieur ou égale à
<=	Inférieur ou égale à
BETWEEN	Valeur comprise dans un intervalle donné (utile pour les nombres ou dates)
Opérateurs **numériques**	Description
+, -, ×, ÷, ABS	les quatre opérations, la valeur absolue
MIN, MAX	retient le minimum/maximum entre plusieurs attributs numériques ou texte

Exemples

```
SELECT title FROM books
WHERE author='J.K. Rowling' AND words > 120000
```

title
Harry Potter and the Goblet of Fire
Harry Potter and the Order of the Phoenix
Harry Potter and the Half-Blood Prince
Harry Potter and the Deathly Hallows

```
SELECT * FROM books
```

```
WHERE author IN ('J.K. Rowling', 'J.R.R. Tolkien')
```

id	author	title	words
1	J.K. Rowling	Harry Potter and the Philosopher's Stone	79944
2	J.K. Rowling	Harry Potter and the Chamber of Secrets	85141
3	J.K. Rowling	Harry Potter and the Prisoner of Azkaban	107253
4	J.K. Rowling	Harry Potter and the Goblet of Fire	190637
5	J.K. Rowling	Harry Potter and the Order of the Phoenix	257045
6	J.K. Rowling	Harry Potter and the Half-Blood Prince	168923
7	J.K. Rowling	Harry Potter and the Deathly Hallows	197651
12	J.R.R. Tolkien	The Hobbit	95022
13	J.R.R. Tolkien	Fellowship of the Ring	177227
14	J.R.R. Tolkien	Two Towers	143436
15	J.R.R. Tolkien	Return of the King	134462

Exercice 1 Considérons une table **coords** ayant quatre attribut : POINT (TEXT), X(FLOAT), Y(FLOAT) et Z(FLOAT). Que sélectionnent les requètes suivantes ?

```
SELECT  POINT from coords
WHERE X*X+Y*Y+Z*Z=1
```

```
SELECT  POINT from coords
WHERE (X*X+Y*Y+Z*Z >=  1) AND (MAX(X,Y,Z)<=1)
```

```
SELECT  POINT, MIN(X,Y,Z), MAX(X,Y,Z), (X+Y+Z)/3 from coords
```

Exercice 2 Expliquer la séquence et le résultat :

```
SELECT  title, author, MIN(title, author) from books
```

title	author	MIN(title, author)
Harry Potter and the Philosopher's Stone	J.K. Rowling	Harry Potter and the Philosopher's Stone
Harry Potter and the Chamber of Secrets	J.K. Rowling	Harry Potter and the Chamber of Secrets
Harry Potter and the Prisoner of Azkaban	J.K. Rowling	Harry Potter and the Prisoner of Azkaban
Harry Potter and the Goblet of Fire	J.K. Rowling	Harry Potter and the Goblet of Fire
Harry Potter and the Order of the Phoenix	J.K. Rowling	Harry Potter and the Order of the Phoenix
Harry Potter and the Half-Blood Prince	J.K. Rowling	Harry Potter and the Half-Blood Prince
Harry Potter and the Deathly Hallows	J.K. Rowling	Harry Potter and the Deathly Hallows
Twilight	Stephenie Meyer	Stephenie Meyer
New Moon	Stephenie Meyer	New Moon
Eclipse	Stephenie Meyer	Eclipse
Breaking Dawn	Stephenie Meyer	Breaking Dawn
The Hobbit	J.R.R. Tolkien	J.R.R. Tolkien
Fellowship of the Ring	J.R.R. Tolkien	Fellowship of the Ring
Two Towers	J.R.R. Tolkien	J.R.R. Tolkien
Return of the King	J.R.R. Tolkien	J.R.R. Tolkien

Que se passerait-il si on remplaçait MIN par MAX ?

4 Agrégation

Les **fonctions d'agrégation** reçoivent toutes les lignes d'**UN** champ en entrée, et retournent en sortie le résultat d'une opération statistique : le plus petit MIN, le plus grand MAX, la valeur moyenne AVG ou la somme SUM de plusieurs enregistrements. On dispose aussi de COUNT qui donne le nombre de lignes dans la table, et de COUNT DISTINCT qui compte le nombre de lignes dont les enregistrements sont différents.

4.1 Syntaxe

```
SELECT fonction(colonne) FROM Table
```

```
SELECT fonction1(colonne1), fonction2(colonne2), ..., fonctionN(colonneN) FROM Table
```

Exemple

```
SELECT count(title) FROM books
```

count(title)
15

on peut sélectionner d'abord les colonnes vérifiant une condition logique puis appliquer une certaine fonction :

```
SELECT fonction1(colonne1), fonction2(colonne2), ..., fonctionN(colonneN) FROM Table
WHERE conditions logiques sur certains colonnes
```

Exemple

```
SELECT count(title) FROM books
WHERE author='J.R.R. Tolkien'
```

count(title)
4

4.2 Liste des fonctions d'agrégation disponibles

Fonction	Renvoie	Domaine
MIN	minimum des enregistrements du champ	INT ou FLOAT
MAX	maximum des enregistrements du champ	INT ou FLOAT
SUM	somme des enregistrements du champ	INT ou FLOAT
AVG	moyenne des enregistrements du champ	INT ou FLOAT
COUNT	nombre d'enregistrements	quelconque

Remarque
MAX et MIN peuvent aussi bien être utilisées comme fonctions d'agrégation (elles portent alors sur toutes les valeurs d'un champ) que comme opérateurs servant à créer des conditions au sein de la commande WHERE (elles s'appliquent alors ligne par ligne à plusieurs champs).

Exercice 3 Ecrire une requete SQL qui donne le nombre moyen de mots que contient un livre dans la base de données.

4.3 GROUP BY

Il y a une différence entre

- se demander le nombre de titres dans la base, ou le nombre de titres d'un auteur donné : le résultat est un nombre
- se demander le nombre de titres **par auteur** : le résultat est une liste d'entiers, chacune des lignes correspondant à un auteur et contenant le nombre de livres qu'il a écrits.

Dans le premier cas on applique la fonction à toute la table, dans le second on l'applique à plusieurs sélections, puis on regroupe. Pour cela, on doit utiliser la commande GROUP BY.

```
SELECT colonne1|colonne1, colonne2, ..., colonneN|* FROM Table
WHERE conditions logiques sur certaines colonnes
GROUP BY colonneA
```

```
SELECT colonne1|colonne1, colonne2, ..., colonneN|* FROM Table
WHERE conditions logiques sur certains colonnes
GROUP BY colonneA, colonneB, ..., colonneZ
```

Exemple

```
SELECT count(title) FROM books
GROUP BY author
```

count(title)
7
4
4

Pour faire apparaître les auteurs à côté :

```
SELECT author, count(title) FROM books
GROUP BY author
```

author	count(title)
J.K. Rowling	7
J.R.R. Tolkien	4
Stephenie Meyer	4

4.4 GROUP BY ··· HAVING ···

Exemple

Essayons d'afficher les auteurs ayant écrit plus d'un million de mots :

```
SELECT author, sum(words)  FROM books
WHERE sum(words)>1000000
GROUP BY author
```

Ce code génère une erreur, car **WHERE n'accepte pas les fonctions d'agrégation**. L'instruction WHERE sélectionne « en amont », avant que l'on ait groupé par author ; il faut faire ici une sélection après que l'on ait groupé par auteur. On utilise alors la commande HAVING. Ainsi

```
SELECT author, sum(words)  FROM books
GROUP BY author
HAVING sum(words)>1000000
```

et

```
SELECT author, sum(words)   FROM books
GROUP BY author
HAVING  sum(words)>590000;
```

rendent :

author	sum(words)
J.K. Rowling	1086594

et

author	sum(words)
J.K. Rowling	1086594
Stephenie Meyer	591434

La commande HAVING doit toujours être précédée de GROUP BY. Elle fonctionne comme WHERE, mais permet aussi d'écrire des conditions portant sur les valeurs de fonctions d'agégation, une fois les groupes formés.

L'ordre des commandes est, à ce stade du cours, toujours le suivant :

```
SELECT ...
FROM table
WHERE condition
GROUP BY expression
HAVING condition
```

Remarques

1. Il est conseillé de placer un maximum de conditions dans le WHERE, pour des raisons de complexité temporelle de la requete.
2. MAX et MIN jouent :
   - tantôt le rôle de fonctions d'agrégations, ils agissent alors sur toutes les entrées d'une seul champ et rendent une seule sortie ;
   - tantôt, comme dans la partie précédente, un rôle de fonction entre champs : à l'instar de + ou $\times$, ils opèrent un calcul différent pour **chaque** enregistrement de la base de données.

Exercice 4 Une table T dispose de trois champs numériques A, B et C. Ecrire une requète permettant :

1. D'obtenir pour chaque enregistrement, la plus grande valeur et la valeur moyenne entre A, B et C
2. D'obtenir la valeur maximale du champ A
3. D'obtenir la plus grande valeur parmi toutes celles présentes dans les trois colonnes A,B,C.
4. Comparer les résultats de ces deux requètes :

```
SELECT MAX(MIN(A,B,C)) FROM T

ou

SELECT MIN(MAX(A,B,C)) FROM T
```

5 Renommage : AS ···

Les requetes créent de nouvelles tables fictives appelées aussi **vues**. On peut vouloir donner des noms plus explicites aux nouvelles colonnes ainsi créées, et par exemple appeler `Nombre_mots` l'attribut `sum(words)`. Il suffit pour cela d'utiliser le mot clé **AS** dans la commande **SELECT** :

Exemple

```
SELECT author, sum(words) AS Nombre_mots FROM books
GROUP BY author
HAVING  sum(words)>590000;
```

	author	Nombre_mots
rend	J.K. Rowling	1086594
	Stephenie Meyer	591434

L'utilisation de AS :

- s'effectue entre SELECT et FROM
- s'étend à tous les noms d'attributs, agrégation d'attributs, table
- permet d'utiliser le nom à l'intérieur de la requête

Exemple

```
SELECT author AS auteur, sum(words) AS Nombre_mots  FROM books AS livres
GROUP BY auteur
HAVING  Nombre_mots>590000;
```

	auteur	Nombre_mots
rend	J.K. Rowling	1086594
	Stephenie Meyer	591434

Enfin, on peut se passer de **AS** :

```
SELECT author  auteur, sum(words) Nombre_mots  FROM books livres
GROUP BY auteur
HAVING  Nombre_mots>590000;
```

Remarque

Le renommage n'est pas seulement esthétique ou pratique : il est nécessaire dans certaines requêtes complexes, cf plus loin.

6 Tri et troncature

6.1 ORDER BY ... ASC|DESC

La commande ORDER BY permet de trier les lignes dans un résultat d'une requête SQL. on peut trier les données selon une ou plusieurs colonnes, par ordre ascendant ASC ou descendant DESC.

```
SELECT colonne1, colonne2, colonne3
FROM table
ORDER BY colonne1 DESC, colonne2 ASC
```

```
SELECT * FROM books
ORDER BY title
```

id	author	title	words
11	Stephenie Meyer	Breaking Dawn	192196
10	Stephenie Meyer	Eclipse	147930
13	J.R.R. Tolkien	Fellowship of the Ring	177227
2	J.K. Rowling	Harry Potter and the Chamber of Secrets	85141
7	J.K. Rowling	Harry Potter and the Deathly Hallows	197651
4	J.K. Rowling	Harry Potter and the Goblet of Fire	190637
6	J.K. Rowling	Harry Potter and the Half-Blood Prince	168923
5	J.K. Rowling	Harry Potter and the Order of the Phoenix	257045
1	J.K. Rowling	Harry Potter and the Philosopher's Stone	79944
3	J.K. Rowling	Harry Potter and the Prisoner of Azkaban	107253
9	Stephenie Meyer	New Moon	132807
15	J.R.R. Tolkien	Return of the King	134462
12	J.R.R. Tolkien	The Hobbit	95022
8	Stephenie Meyer	Twilight	118501
14	J.R.R. Tolkien	Two Towers	143436

```
SELECT * FROM books
ORDER BY author ASC, words DESC
```

id	author	title	words
5	J.K. Rowling	Harry Potter and the Order of the Phoenix	257045
7	J.K. Rowling	Harry Potter and the Deathly Hallows	197651
4	J.K. Rowling	Harry Potter and the Goblet of Fire	190637
6	J.K. Rowling	Harry Potter and the Half-Blood Prince	168923
3	J.K. Rowling	Harry Potter and the Prisoner of Azkaban	107253
2	J.K. Rowling	Harry Potter and the Chamber of Secrets	85141
1	J.K. Rowling	Harry Potter and the Philosopher's Stone	79944
13	J.R.R. Tolkien	Fellowship of the Ring	177227
14	J.R.R. Tolkien	Two Towers	143436
15	J.R.R. Tolkien	Return of the King	134462
12	J.R.R. Tolkien	The Hobbit	95022
11	Stephenie Meyer	Breaking Dawn	192196
10	Stephenie Meyer	Eclipse	147930
9	Stephenie Meyer	New Moon	132807
8	Stephenie Meyer	Twilight	118501

6.2 LIMIT ··· OFFSET ···

En fin de requête, cette commande restreint l'affichage :

```
LIMIT  # Nombre de lignes a afficher
OFFSET  # en partant de la ligne numero....
```

7 Deux exercices pour manipuler les BDD

Pour interagir avec les bases de données du TP, on utilise DB Browser for SQLite. Pour connecter DB Browser à une base de données, utiliser le menu Ouvrir une base de données (celle-ci doit être installée localement). Pour étudier la structure d'une base on dispose des onglets Structure et Parcourir les données qui permettent de voir les tables existantes et ce qu'elles contiennent. Pour exécuter une requête SQL, on va dans l'onglet correspondant.

Exercice 5 La table Country contient des données sur des pays, distribuées entre les six attributs : Name Code Capital Province Area Population
Ecrire une requete SQL qui permet d'obtenir :

1. la liste des pays dont la population excède 60 000 000 habitants ;
2. la même liste triée par ordre alphabétique ;
3. la liste des pays et de leurs populations respectives, triée par ordre décroissant de population ;
4. le nom des dix pays ayant la plus petite superficie ;
5. le nom des dix suivants.

Exercice 6 La table language possède les attributs suivants : Country Name Percentage
L'attribut Country est un code pour désigner le pays, Name est le nom d'**une** langue parlée dans celui-ci, et Percentage la proportion d'habitants dont c'est la langue maternelle.

1. Donner la liste des dix langues parlées dans le plus de pays différents.
2. Quelles sont les langues parlées dans exactement cinq pays ?
3. Dans un second temps, écrire une requete pour obtenir les codes de ces pays
   Pour les questions suivants, passez par des sous requètes, du genre :

$$\text{SELECT ... FROM } (\underbrace{\text{SELECT ... FROM ...}etc}_{\text{requte ds la requte}})etc$$

4. * effectuer 3) sans passer par la requete intermédiaire 2)
5. * Modifier la requète de la première question de cet exercice, pour trier la liste obtenue par ordre alphabétique des LANGUES.

8 Jointure, interrogation de plusieurs tables

8.1 Principe et exemples

Il arrive que l'on ait besoin d'informations contenues dans plusieurs tables *différentes*. Il faut alors exprimer une requête permettant de les regrouper.

Par exemple, on dispose de trois tables dans notre base de données « jouet » :

Une table `book`, déjà vu ; une table `borrow`, donnant un historique de chaque emprunt de chaque livre ; et enfin une table `readers`, qui donne des informations sur les lecteurs qui peuvent potentiellement emprunter ces livres.

id	author	title	words
1	J.K. Rowling	Harry Potter and the Philosopher's Stone	79944
2	J.K. Rowling	Harry Potter and the Chamber of Secrets	85141
3	J.K. Rowling	Harry Potter and the Prisoner of Azkaban	107253
4	J.K. Rowling	Harry Potter and the Goblet of Fire	190637
5	J.K. Rowling	Harry Potter and the Order of the Phoenix	257045
6	J.K. Rowling	Harry Potter and the Half-Blood Prince	168923
7	J.K. Rowling	Harry Potter and the Deathly Hallows	197651
8	Stephenie Meyer	Twilight	118501
9	Stephenie Meyer	New Moon	132807
10	Stephenie Meyer	Eclipse	147930
11	Stephenie Meyer	Breaking Dawn	192196
12	J.R.R. Tolkien	The Hobbit	95022
13	J.R.R. Tolkien	Fellowship of the Ring	177227
14	J.R.R. Tolkien	Two Towers	143436
15	J.R.R. Tolkien	Return of the King	134462

TABLE 1 – books

reader	book	dayout	dayreturn
1	2	1	9
2	5	2	3
3	10	2	4
4	2	3	8
5	11	5	19
6	7	5	20
7	5	7	39
8	5	7	12
2	1	7	2
1	6	9	12
4	1	10	17
1	1	10	10
6	4	11	11
1	8	14	18
8	6	19	19
1	6	22	23
1	2	17	18
2	1	11	18
1	2	21	65

TABLE 2 – borrow

id	name	age
1	Louise	19
2	Lucas	17
3	Marie	18
4	Erwann	18
5	Rose	22
6	Leo	31
7	Arthur	15
8	Clara	15

TABLE 3 – readers

Remarquez que :

1. Dans la table borrow, les livres sont désignés par leur code et non leur titre ; de même les lecteurs sont désignés par leur code id.
2. Par exemple qui a emprunté le livre Harry Potter and the Philosopher's Stone ? on va voir dans borrow, dans laquelle le livre a pour code **1**. Il a été emprunté par les lecteurs 1, 2 et 4 ; regardons dans la table readers : ce sont Louise, Lucas et Erwann.
3. Pour savoir si un livre a été emprunté, et plus généralement pour toute sorte d'information, il faut donc arriver à faire communiquer les deux tables ensemble.

La commande pour **joindre** deux tables est Table1 JOIN Table2

Exemple

SELECT * FROM readers JOIN borrow

id	name	age	reader	book	dayout	dayreturn
1	Louise	19	1	2	1	9
1	Louise	19	2	5	2	3
1	Louise	19	3	10	2	4
1	Louise	19	4	2	3	8
1	Louise	19	5	11	5	19
1	Louise	19	6	7	5	20
1	Louise	19	7	5	7	39
1	Louise	19	8	5	7	12
1	Louise	19	2	1	7	9
1	Louise	19	1	6	9	12
1	Louise	19	4	1	10	17
1	Louise	19	1	1	10	10
1	Louise	19	6	4	11	11
1	Louise	19	1	8	14	18
1	Louise	19	8	6	19	19
1	Louise	19	1	6	22	23
1	Louise	19	1	2	17	18
1	Louise	19	1	2	21	65
1	Louise	19	2	1	11	
2	Lucas	17	1	2	1	9
2	Lucas	17	2	5	2	3
2	Lucas	17	3	10	2	4
2	Lucas	17	4	2	3	8
2	Lucas	17	5	11	5	19
2	Lucas	17	6	7	5	20
2	Lucas	17	7	5	7	39
2	Lucas	17	8	5	7	12
2	Lucas	17	2	1	7	9

...

Attention

Cette commande est en général inutile. Si la Table1 a n_1 enregistrements et la Table2 a n_2 enregistrements alors Table1 JOIN Table2 admet $n_1 n_2$ enregistrements. Ceux ci sont produits en juxtaposant chaque entrée de Table1 avec chaque entrée de Table2. Cela ne produit aucune information intéressante.
Il faudrait ne garder dans la table de droite que les entrées qui correspondent à la table de gauche.

Pour cela on utilise la commande Table1 JOIN Table2 ON Condition de jointure

La condition de jointure est typiquement une égalité, de la forme Tables1.attribut1=Table2.attribut. On peut néanmoins donner n'importe quelle condition logique.

Exemple

```
1 SELECT * FROM readers JOIN borrow ON id=reader
```

	id	name	age	reader	book	dayout	dayreturn
	1	Louise	19	1	2	1	9
	2	Lucas	17	2	5	2	3
	3	Marie	18	3	10	2	4
	4	Erwann	18	4	2	3	8
	5	Rose	22	5	11	5	19
	6	Leo	31	6	7	5	20
	7	Arthur	15	7	5	7	39
	8	Clara	15	8	5	7	12
	2	Lucas	17	2	1	7	9
renvoie	1	Louise	19	1	6	9	12
	4	Erwann	18	4	1	10	17
	1	Louise	19	1	1	10	10
	6	Leo	31	6	4	11	11
	1	Louise	19	1	8	14	18
	8	Clara	15	8	6	19	19
	1	Louise	19	1	6	22	23
	1	Louise	19	1	2	17	18
	1	Louise	19	1	2	21	65
	2	Lucas	17	2	1	11	NULL

Remarque

Il n'est pas interdit pour deux tables différentes d'avoir des champs de même nom. Pour préciser de quel champ parle, on utilise la syntaxe NomDeLaTable.NomDuChamp plutôt que NomDuChamp. Ici :

```
SELECT * FROM readers JOIN borrow ON readers.id=borrow.reader
```

C'est utilisable aussi en sortie.

```
SELECT readers.*,borrow.book FROM readers JOIN borrow ON readers.id=borrow.reader
```

renvoie ainsi les colonnes numéros 1,2,3 et 5 de la vue précédente.

8.2 Notion de clé primaire d'une table

Une clé primaire est la donnée (d'une ou de plusieurs colonnes de la table) qui permet d'identifier de manière unique un enregistrement dans une table : deux enregistrements distincts de la table ne peuvent pas avoir les mêmes valeurs pour les colonnes qui définissent la clé primaire.

Dans nos trois tables :

1. Pour book et reader, des clés primaires artificielles ont été rajoutées : book.id et reader.id. Une clé primaire naturelle pour la table reader pourrait être le couple Nom, Prénom, par exemple... en supposant qu'il n'y a pas d'homonymie possible.
2. Pour borrow, aucun champ ne peut jouer le rôle de clé primaire. Une clé possible est (reader, book, dayout).

Méthode

Au moment de joindre des tables, posez-vous la question de la clé primaire éventuelle.

9 Requetes complexes : emploi de toutes les commandes

On peut alors appliquer toutes les commandes que l'on a vues précédemment dans une même requete :

Exemple

La requête suivante demande la longueur des différents emprunts de Louise :

```
SELECT  (dayreturn-dayout) AS S FROM readers JOIN borrow
ON readers.id=borrow.reader
WHERE name='Louise'
```

Exemple

Pour chaque livre emprunté, on peut obtenir le nombre de fois où il a été emprunté, et le temps moyen d'emprunt :

```
SELECT books.title, count(*) AS Nombre_Emprunts, AVG(borrow.dayreturn - borrow.dayout) AS
    Temps_Sorti
FROM books join borrow ON books.id=borrow.book
GROUP BY books.id
```

title	Nombre_Emprunts	Temps_Sorti
Harry Potter and the Philosopher's Stone	4	3.0
Harry Potter and the Chamber of Secrets	4	14.5
Harry Potter and the Goblet of Fire	1	0.0
Harry Potter and the Order of the Phoenix	3	12.6666666666667
Harry Potter and the Half-Blood Prince	3	1.33333333333333
Harry Potter and the Deathly Hallows	1	15.0
Twilight	1	4.0
Eclipse	1	2.0
Breaking Dawn	1	14.0

Remarque

On constate que les livres qui n'ont jamais été empruntés ne figurent pas dans la vue obtenue : en effet la jointure ne trouve aucune association entre •d'une part les enregistrements de la base books qui contiennent des livres jamais empruntés, et •d'autre part les enregistrements de la base borrow.

10 Opérations ensemblistes

Supposons que l'on souhaite obtenir tous les livres qui n'ont jamais été empruntés. Il est facile d'obtenir tous les livres empruntés par l'opération de jointure suivante :

```
SELECT DISTINCT books.title FROM books JOIN borrow ON books.id=borrow.book
```

Harry Potter and the Chamber of Secrets
Harry Potter and the Order of the Phoenix
Eclipse
Breaking Dawn
Harry Potter and the Deathly Hallows
Harry Potter and the Philosopher's Stone
Harry Potter and the Half-Blood Prince
Harry Potter and the Goblet of Fire
Twilight

Le fait de joindre les bases books et borrow fournit facilement les livres empruntés, mais fait disparaître ceux qui ne l'ont jamais été. L'opération de jointure, de par sa construction même, ne peut pas répondre au problème.

On aimerait disposer d'une opération de complémentaire, qui serait en quelque sorte : « livres de la table `books` » \ « livres de la requetes `livres empruntés` »

C'est à cela que servent les commandes UNION, INTERSECT, EXCEPT : elles donnent respectivement la réunion, l'intersection, et la différence ensembliste de deux requetes.

Exemple

```
SELECT books.title FROM books
EXCEPT
SELECT books.title FROM books JOIN borrow ON books.id=borrow.book
```

Il faut que ces deux requetes disposent du même nombre de champs, et c'est la seule contrainte : les champs n'ont même pas besoin de disposer du même domaine. Que fait la requete suivante ?

Exemple

```
SELECT title, author FROM books
UNION
 SELECT  title,words FROM books
```

11 Récapitulatif et ordre d'utilisation des commandes

Les opérations possibles sont la sélection, la jointure de tables, la restriction à une condition, l'utilisation de fonctions d'agrégations, la projection, sélection (ou restriction) WHERE, le renommage AS , la jointure JOIN ON, le produit JOIN ; et les fonctions d'agrégation suivantes sont disponibles : minimum, maximum, somme, moyenne, comptage.

```
SELECT *
FROM table JOIN table2 ON condition {JOIN table 3 ON ...}
WHERE condition
GROUP BY enregistrement
HAVING condition
{ UNION | INTERSECT | EXCEPT }
ORDER BY expression
LIMIT count OFFSET start
```

A partir d'ici, je considère que ces questions sont non prioritaires pour votre préparation au concours. En effet lesquestions de SQL parsujet sont au nombre de 3 en moyenne, avec une seule d'entre elles vraiment difficile. Si vous vous sentez à l'aise ou curieux allez-y, mais sinon employez votre temps de préparation à améliorer d'autres points.

12 Imbrication, jointures complexes

12.1 Requête imbriquée

Exemple

On veut savoir quels lecteurs ont d'autres lecteurs du même âge. Il est intéressant pour cela :

1. de connaître d'abord les âges des différents lecteurs, et ceux qui sont représentés plusieurs fois
2. de déterminer ensuite quels lecteurs ont ces âges.

La première requête est :

```
SELECT age ,count(*) AS effectif
FROM readers
GROUP BY age
HAVING  effectif !=1
```

On utilise alors cette requête `sous_requete1` comme une table classique. Il faut lui donner un nom, et effectuer une jointure avec la table `readers` :

```
SELECT readers.name
FROM readers join
(SELECT age, count(*) AS effectif FROM readers GROUP BY age) AS sous_requete1
ON readers.age=sous_requete1.age
WHERE effectif!=1
```

Alternativement, plutôt que de donner un nom à la vue, on peut choisir de donner un nom au champ age, qui est le seul à être ré-appelé :

```
SELECT readers.name
FROM readers join
(SELECT age AS age2 ,count(*) AS effectif FROM readers GROUP BY age HAVING  effectif !=1)
ON readers.age=age2
```

La commande suivante est moins performante, car on ne filtre qu'après la jointure et non avant :

```
SELECT readers.name
FROM readers join
(SELECT age AS age2 ,count(*) AS effectif FROM readers GROUP BY age) on readers.age=age2
WHERE effectif!=1
```

Dans tous les cas, l'utilisation d'une sous-requête est ici incontournable.
En particulier, la commande suivante ne produit pas le résultat voulu :

```
SELECT readers.name
FROM readers
GROUP BY age
HAVING count(*)!=1
```

On obtient une table contenant Arthur et Marie, et non pas les lecteurs que l'on visait.
Remarque : La commande GROUP BY a en effet pris *un* représentant de *chaque* groupe, et non tout le groupe. Ainsi utiliser GROUP BY sans fonction d'agrégation sur le champ est généralement synosyme d'erreur.

12.2 Jointure d'une table avec elle-même

Suppososn que la table T contienne un attribut de tmps `t` de domaine INT, et que l'on veuille regarder l'évolution d'un attribut `A` de domaine INT ou FLOAT au cours du temps : maximum au cours du temps $\max_{1\leqslant i\leqslant t} a_i$, différences successives $(a_{i+1}-a_i)_{i\leqslant t-1}$, moyennes cumulées $(\frac{1}{n}\sum_{i=1}^{n} a_i)_{n\leqslant t}$ par exemple.

On doit alors envisager une condition de jointure de T avec elle-même, du type `A1.t = A2.t -1` ou même `A1.t <= A2.t`.

Ainsi la commande suivante affiche l'évolution du nombre de mots dans les différents tomes de Harry Potter :

```
SELECT b2.words-b1.words , b1.*
FROM books b1 join books b2 ON b1.id=b2.id-1
WHERE b1.id<7
```

13 Exercices

13.1 Basiques

Exercice 7

1. Donner pour chaque livre, le nombre de fois où il a été emprunté.
2. Donner pour chaque lecteur et chaque livre, le nombre d'emprunts
3. * Donner pour chaque lecteur, le plus grand nombre de fois où il a emprunté le meme livre

Exercice 8 Soit le schéma de la base de données Bibliothèque suivante :
Etudiant(**NumEtd**,NomEtd,PrenomEdt,AdresseEtd)
Livre(**NumLivre**,TitreLivre,*NumAuteur*,*NumEditeur*,*NumTheme*,AnneeEdition)
Auteur(**NumAuteur**,NomAuteur,AdresseAuteur)
Editeur(**NumEditeur**,NomEditeur,AdresseEditeur)
Theme(**NumTheme**,IntituléTheme)
Prêt(**NumEtd**,**NumLivre**,**DatePret**,DateRetour)
On a indiqué en **gras** les clés primaires et en italique les clés étrangères.
Donner une requète permettant d'obtenir : 1. Le nom, le prénom et l'adresse de l'étudiant de nom 'Dupont'
2. Le numéro de l'auteur 'Durand'
3. la liste des livres de l'auteur numéro 121
4. les livres de l'auteur nom 'Alami'
5. le numéro de l'auteur du livre 'le Lys dans la vallée'
6. le nom et l'adresse de l'auteur du livre 'le Lys dans la vallée'
7. Les livres de l'auteur 'Dupont' qui édités chez l'éditeur 'EDT'
8. les livres de l'auteur 'Dupont' et ceux de l'auteur 'Dupond'
9. les livres qui n'ont jamais été empruntés

Exercice 9 On se donne la base de données constituée des trois tables suivantes : **Table Avion :**
NA : numéro avion de type entier (clé primaire),
Nom : nom avion de type texte,
Capacite : capacité avion de type entier
Table Pilote :
NP : numéro pilote de type entier,
Nom : nom du pilote de type texte,
Adresse : adresse du pilote de type texte
Table Vol :
NV : numéro de vol de type texte,
NP : numéro de pilote de type entier,
NA : numéro avion de type entier,
VD : ville de départ de type texte,
VA : ville d'arrivée de type texte,
HD : heure de départ de type entier,
HA : heure d'arrivée de type entier

1. Afficher tous les avions et toutes les données les concernant (numéro, nom, capacité)
2. Afficher tous les avions par ordre croissant sur le nom
3. Afficher les noms et les capacités des avions
4. Afficher les différentes villes de départ sans redondance
5. Afficher les vols dont le départ est Paris ou l'arrivée Séoul
6. Afficher la capacité maximale, minimale, et moyenne des avions
7. Afficher les données des avions dont la capacité est supérieure à la capacité moyenne
8. Afficher les numéros des pilotes qui n'ont jamais effectué de vol selon la base de données
9. Afficher les noms des pilotes qui conduisent un AIRBUS

Exercice 10 On a une base de données triangles.db, constituée d'une seule table, dont le modèle logique est :

triangle(idt :integer, ab :integer, ac :integer, bc :integer)

Chaque élément de la table représente les longueurs d'un triangle ABC, ainsi qu'un identificateur unique. Ecrire une requète qui :

1. Compte le nombre de triangles dans la base
2. Donne les identifiants des triangles équilatéraux

3. Sélectionne les identifiants des triangles dont le périmètre vaut 100
4. Donne les identifiants des triangles rectangles
5. Donne la plus petite valeur des produits $AB \times AC \times BC$, parmi les triangles (ABC) de périmètre supérieur ou égal à 10;
6. Donne le(s) identifiant(s) de(s) triangle(s) dont le produit $AB \times AC \times BC$ est le plus petit, parmi les triangles (ABC) de périmètre supérieur ou égal à 10;
7. Donne, pour chacun des triangles dont tous les côtés sont de longueur au moins 2, l'identifiant et le plus petit côté
8. Donne les identifiants des triangles dont le périmètre est minimal
9. Donne, parmi les triangles dont tous les côtés sont de longueur au moins 2, les identifiants de ceux dont le périmètre est minimal
10. Donne les identifiants de ceux dont le périmètre est minimal et dont tous les côtés sont de longueur au moins 2
11. Donne les identifiants, parmi les triangles dont le périmètre est minimal, ceux dont tous les côtés sont de longueur au moins 2.
12. Donne la liste (identifiant, longueur) du plus petit côté
13. Donne le plus petit de tous les côtés de tous les triangles
14. Donne les identifiants des triangles dont l'un des côtés a la longueur la plus petite de toute la base

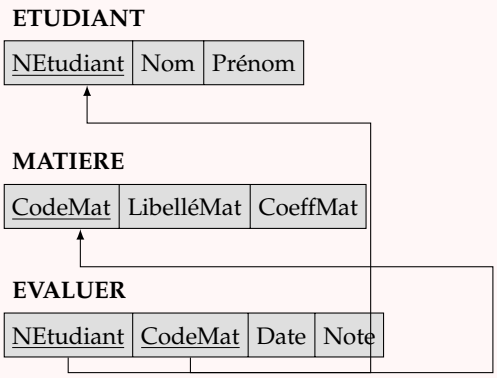
    Deux triangles sont dits **presque égaux** lorsque leur plus gand et leur plus petit côté coincide. Ainsi les deux triangle de longueurs (2,4,5) et (2,5,3) sont presque égaux.
15. Donner une requète qui, pour chaque couple de triangles presque égaux, retourne la moyenne des longueurs du troisième côté

13.2 Standard

Exercice 11 Une base de données gère les notes d'étudiants, dans différentes matières. Elle est constituée d'une seule table :
TABLE(Matiere, Etudiant, Coeff, , Moyenne)
Donner des requètes SQL pour

1. la moyenne générale de chaque étudiant
2. Les dix étudiants les meilleurs dans la matière 'Chimie'
3. Parmi les étudiants qui ont au moins 12 en Chimie, les cinq derniers
4. la moyenne générale de chacune des matières
5. Les étudiants dont la moyenne de Maths est inférieure à 8
6. Les étudiants dont la moyenne générale est inférieure à 8
7. la moyenne de chaque étudiant, en ne tenant compte que des matières où l'étudiant a une moyenne supérieures à 10
8. la moyenne de Maths, en ne tenant compte que des étudiants dont la moyenne de Physique est supérieure à 10
9. la moyenne de Maths, en ne tenant compte que des étudiants dont la moyenne générale est supérieure à 10
10. la moyenne générale de la promotion, en ne tenant compte que des étudiants dont les moyennes dans toutes les matières sont toutes supérieures à 8.
11. la moyenne générale de la promotion, en ne tenant compte que des étudiants dont, dans toute les matières, les moyennes sont supérieures à la moyenne de la classe.

Exercice 12 Une base de données gère les notes d'étudiants, dans différentes matières. Elle est constituée de trois tables :
ETUDIANT(NEtudiant , Nom, Prénom)
MATIERE(CodeMat , LibelléMat, CoeffMat)
EVALUER(NEtudiant, CodeMat , Date, Note)



Description des Tables

Table `ETUDIANT`

Cette table contient les informations sur les étudiants. Elle est définie comme suit :

- **NEtudiant** (clé primaire) : Numéro unique d'identification de l'étudiant.
- **Nom** : Nom de l'étudiant.
- **Prénom** : Prénom de l'étudiant.

Exemple de données :

NEtudiant	Nom	Prénom
1	Dupont	Jean
2	Martin	Claire

Table `MATIERE`

Cette table contient des informations sur les matières. Elle est définie comme suit :

- **CodeMat** (clé primaire) : Code unique de la matière.
- **LibelléMat** : Nom complet de la matière.
- **CoeffMat** : Coefficient de la matière, qui peut influencer le poids de la note dans le calcul de la moyenne.

Exemple de données :

CodeMat	LibelléMat	CoeffMat
M1	Mathématiques	3
M2	Physique	2

Table `EVALUER`

Cette table enregistre les évaluations des étudiants dans différentes matières. Elle est définie comme suit :

- **NEtudiant** (clé étrangère) : Numéro de l'étudiant (référence à `ETUDIANT`).
- **CodeMat** (clé étrangère) : Code de la matière (référence à `MATIERE`).
- **Date** : Date de l'évaluation.
- **Note** : Note obtenue par l'étudiant dans la matière.

Exemple de données :

NEtudiant	CodeMat	Date	Note
1	M1	2024-05-01	15
1	M2	2024-05-02	14
2	M1	2024-05-01	16

Donner une requète pour :

1. le nombre total d'étudiants
2. la note la plus basse et la plus haute, parmi toutes les notes
3. la note la plus basse et la plus haute, pour chaque matière
4. pour chaque étudiant, sa moyenne générale
5. pour chaque matière, la moyenne de la promotion
6. la moyenne de chaque étudiant dans chaque matière
7. la moyenne générale de chaque étudiant
8. Les dix étudiants les meilleurs dans la matière 'Chimie'
9. Parmi les étudiants qui ont au moins 12 en Chimie, les cinq derniers
10. la moyenne générale de chacune des matières
11. Les étudiants dont la moyenne de Maths est inférieure à 8
12. Les étudiants dont la moyenne générale est inférieure à 8
13. la moyenne de chaque étudiant, en ne tenant compte que des matières où l'étudiant a une moyenne supérieures à 10

14. la moyenne de Maths, en ne tenant compte que des étudiants dont la moyenne de Physique est supérieure à 10
15. la moyenne de Maths, en ne tenant compte que des étudiants dont la moyenne générale est supérieure à 10
16. la moyenne générale de la promotion, en ne tenant compte que des étudiants dont les moyennes dans toutes les matières sont toutes supérieures à 8.
17. la moyenne générale de la promotion, en ne tenant compte que des étudiants dont, dans toute les matières, les moyennes sont supérieures à la moyenne de la classe.

1. SELECT COUNT(*) FROM ETUDIANT
2. SELECT MIN(Note), MAX(Note) FROM EVALUER
3. SELECT CodeMat, LibelléMat, MIN(Note), MAX(Note) FROM EVALUER JOIN MATIERE ON EVALUER.CodeMat=MATIERE.CodeMat GROUP BY CodeMat
4. SELECT Netudiant, Nom, Prenom, AVG(CoeffMat*Note) FROM ETUDIANT JOIN EVALUER ON ... JOIN MATIERE ON ... GROUP BY Netudiant
5. SELECT CodeMat, LibelléMat, AVG(Note) FROM EVALUER JOIN MATIERE ON EVALUER.CodeMat=MATIERE.CodeMat GROUP BY Netudiant
6. SELECT Netudiant, CodeMat, Nom, Prenom,LibelléMat, AVG(Note) FROM ETUDIANT JOIN EVALUER ON ... JOIN MATIERE ON ... GROUP BY Netudiant, CodeMat

Exercice 13 Une base de données regroupe des informations sur des nombres entiers positifs :

- NOMBRE(nombre)
- DIVISE(n1,n2) indique les couples tq le nombre n1 divise le nombre n2

On suppose que si un nombre est dans la base, alors tous ses facteurs premiers y sont aussi.

1. Donner tous les multiples de n présents dans la base.
2. Donner tous les diviseurs de n présents dans la base.
3. Donner les nombres composés présents dans la base.
4. Donner les nombres premiers présents dans la base.
5. Donner les facteurs premiers de n.
6. Donner les facteurs communs à n et m présents dans la base.
7. Donner les facteurs premiers communs à n et m.
8. Donner les triplets (a,b,c) tq ab=c.
9. Donner les triplets (a,b,c) tq ab=c, mais c n'est pas présent dans la base
10. Donner les triplets (a,b,c) tq ab=c, mais a n'est pas présent dans la base
11. Donner les nombres non multiples de m
12. Donner les puissances de 2 présentes dans la base.

SELECT n2 FROM DIVISE WHERE n1=n
SELECT n1 FROM DIVISE WHERE n2=n
SELECT DISTINCT n2 FROM DIVISE where n1 !=1 AND n1 !=n2
SELECT DISTINCT n1 FROM DIVISE where n1 NOT IN (SELECT DISTINCT n2 FROM DIVISE where n1 !=1 AND n1 !=n2) AND p !=1
SELECT n1 FROM DIVISE where n1 !=1 AND n2=n AND n1 NOT IN (SELECT n2 FROM DIVISE WHERE n1 !=1 AND n2 !=n1)

Exercice 14 Une base de données regroupe des informations sur des molécules, et est contituée de trois tables :

- MOLECULES(id, nom)
- ATOMES(nom, symbole)
- ASSOCIATION(idmolecule, symboleatome, nombre)

Ecrire des requètes qui

1. donne le nombre de molécules
2. pour chaque molécule, compte le nombre d'atomes qu'elle contient
3. pour chaque molécule, compte le nombre d'atomes distincts qu'elle contient
4. pour la molécule 'Glucose', renvoie une table de deux colonnes composées de ses atomes et du nombre de fois qu'ils sont dans la molécule
5. renvoie tous les composés chimiques isomères à 'Glucose'
6. Trie les molécules par nombre de C croissant et nombre de H décroissant

7. Donne les molécules contenant plus de C que de H, ordonné par nombre croissant des autres atomes présents.
8. donne les molecules contenant le plus grand nombre d'atomes
9. Donne, sur la population des molecules contenant le plus grand nombre d'atomes de Carbone, le nombre moyen d'atomes d'hydrogène.
10. Donne, pour chaque molécule, le nombre d'atomes de carbone, d'hydrogène, et d'oxygène.
11. Donne les molécules ne contenant pas d'Oxygène

Exercice 15 Soit la BDD définie par le modèle logique suivant :

- ACTOR (id, name)
- MOVIE (id, title, year, score, votes, director) où director réfère à ACTOR(id)
- CASTING (movieid, actorid) où movieid réfère à MOVIE(id) et actorid réfère à ACTOR(id)

Attention : certains acteurs différents ont le même nom (ex : Carry Grant et Hugh Grant), certains films différents ont le même titre (ex : Dune de Lynch et Dune de Villeneuve).

1. Donner le nombre d'acteurs.
2. Donner les noms des acteurs distincts (on ne fait apparaître qu'une fois un nom porté par plusieurs acteurs).
3. Donner pour chaque titre, le nombre de films qui le portent
4. Donner le nombre de titres portés par plusieurs films différents.
5. Donner le nom d'acteur le plus porté
6. Donner la liste des acteurs qui le portent
7. Donner la liste des acteurs jouant dans le film F
8. Donner la liste des acteurs en commun entre les films F1 et F2.
9. Donner le nombre des acteurs en commun entre les films F1 et F2.
10. Donner le score moyen des films sortis l'année 2010.
11. Donner, parmi les films de score 5, ceux pour lesquels le public a le moins voté.
12. Donner, parmi les films pour lesquels le public a le moins voté, ceux de score 5.
13. Trouver les réalisateurs dont les films remportent le plus de votes.

    On dit que deux films sont **étrangers entre eux** si aucun acteur jouant dans l'un ne joue dans l'autre.
14. Donner le nombre de de films étrangers entre eux.
15. Pour chaque couple d'acteur, donner le nombre de films dans lesquels les deux ont joué.

Exercice 16 Points(IdPoint, x, y)
Chemin(Nom, IdChemin)
Appartenance(IdChemin,IdPoint, OrdreTrace)

1. Le nombre de points pour chacune des chemins
2. Le périmétre de chacun des chemins
3. Le(s) point(s) le(s) plus traversé(s)
4. Le(s) chemin(s) le(s) plus long(s)

13.3 Avancés

Exercice 17 Au moyen d'une requete SQL, donner le livre de J.K. Rowling qui a le plus grand nombre de mots.

Exercice 18 On dit qu'un livre est de longueur typique si son nombre de mots est compris entre $\pm 10\%$ de la moyenne. Donner le nombre de livres typiques que chaque auteur a écrit.

Exercice 19 Ecrire une requete qui renvoie, pour chaque lecteur :

1. Le nombre de livres différents empruntés
2. Le livre qu'il a emprunté qui contienne le plus de mots

Exercice 20 Donner la liste des paires de livres (distincts) ayant au moins un lecteur en commun. Cette liste devra être sans répétition, et si (a,b) apparaît, alors (b,a) n'apparaît pas.

Exercice 21 La table T contient deux colonnes : temps, qui est un entier naturel ; et A, qui est une valeur (float) mesurée au cours du temps.
Ecrire une requete qui retourne une vue à trois champs : le champ temps t ; le champ $\max_{0 \leqslant i \leqslant t} A(i)$; le nombre de valeurs t' telles que $0 \leqslant t' \leqslant t$ et pour lesquelles $\max_{0 \leqslant i \leqslant t} A(i) = A(t')$.

14 Solutions

Solution

Solution 3.2.2
1. Tous les points de la base points situés sur la sphère unité
2. Tous les points de la base situés hors de la boule unité, mais situés dans un certain cube de côté 2 et centré en O (ie la boule pour la norme infinie)
3. Pour chaque point : son nom, sa plus grande coordonnée, sa plus petite coordonnée, et la moyenne de ses coordonnées.

Solution

Solution 4.2 Le plus simple :

```
SELECT AVG(words) FROM books
```

Solution

Solution 4.4

```

SELECT MAX(A, B, C), (A+B+C)/3 from T


SELECT MAX(A) FROM T


SELECT MAX(MAX(A,B,C)) FROM T
```

Solution

Solution 13.1
1. `SELECT title, count(*) FROM books join borrow on books.id=borrow.book GROUP BY title`
2. `SELECT title, name, count(*) FROM books JOIN borrow ON books.id=borrow.book JOIN readers ON readers.id=borrow.reader GROUP BY title, name`

```
SELECT name, max(nbemprunts) AS Nombre_emprunts_repet_max
FROM
(SELECT reader AS lecteur , book AS livre , count(*) AS nbemprunts FROM borrow
GROUP BY reader, book)
join readers on readers.id=lecteur
join books on books.id=livre
GROUP BY lecteur
```

3.

Solution

Solution 13.1
1. SELECT COUNT(id) FROM triangle
2. SELECT id FROM Triangle WHERE ab=ac AND ac=bc
3. SELECT id FROM Triangle WHERE ab+ac+bc=100
4. SELECT id FROM Triangle WHERE (ab*ab+ac*ac=bc*bc OR bc*bc+ac*ac=ab*ab OR ab*ab+bc*bc=ac*ac)
5. SELECT MIN(ab*bc*ac) FROM Triangle WHERE ab+ac+bc>=10

6. SELECT id FROM Triangle WHERE ab*bc*ac= (SELECT MIN(ab*bc*ac) FROM Triangle WHERE ab+ac+bc>=10)
7. SELECT id, MIN(ab,ac,bc) FROM Triangle where min(ab,ac,bc)>=2
8. select id from Triangle where ab+bc+ac=(select min(ab+bc+ac) from triangle)
9. select id from Triangle where ab+bc+ac=(select min(ab+bc+ac) from triangle where min(ab,bc,ac)>=2)
10. select id from Triangle where ab+bc+ac=(select min(ab+bc+ac) from triangle) and min(ab,bc,ac)>=2
11. la mmême chose!!!
12. select id, min(ab,bc,ac) from Triangle
13. select min(min(ab,bc,ac)) from Triangle
14. select id from Triangle where (select min(min(ab,bc,ac)) from Triangle) in (ab,ac,bc)
15. select AVG(ab+ac+bc- min(ab,ac,bc)-max(ab,ac,bc)) from Triangle group by min(ab,ac,bc),max(ab,ac,bc)

Solution

Solution 13.2

1. SELECT SUM(Coeff*Moyenne)/SUM(Coeff) FROM Table GROUP BY Etudiant
2. SELECT Etudiant FROM TABLE WHERE Matiere='Chimie' ORDER BY Moyenne LIMIT 10
3. SELECT Etudiant FROM TABLE WHERE Matiere='Chimie' AND Moyenne>=12 ORDER BY Moyenne DESC LIMIT 5
4. SELECT AVG(Moyenne) FROM Table GROUP BY Matiere
5. SELECT Etudiant FROM TABLE WHERE Matiere='Maths' AND Moyenne <8
6. SELECT Etudiant FROM TABLE GROUP BY Etudiant HAVING SUM(Coeff*Moyenne)/SUM(Coeff) < 8
7. SELECT SUM(Coeff*Moyenne)/SUM(Coeff) FROM Table WHERE Moyenne>10
8. SELECT Moyenne FROM Table WHERE Matiere='Maths' AND Etudiant IN (SELECT Etudiant FROM TABLE WHERE Matiere='Physique' AND Moyenne>10)
9. SELECT Moyenne FROM Table WHERE Matiere='Maths' AND Etudiant IN (SELECT Etudiant FROM TABLE GROUP BY Etudiant HAVING SUM(Coeff*Moyenne)/SUM(Coeff) >10)
10. SELECT AVG(MoyenneEtudiant) FROM (SELECT SUM(Coeff*Moyenne)/SUM(Coeff) as MoyenneEtudiant FROM Table GROUP BY Etudiant HAVING MIN(Moyenne)>=8)
11. SELECT Etudiant FROM (SELECT * FROM TABLE JOIN (SELECT Matiere, SUM(Coeff*Moyenne)/SUM(Coeff) FROM Table GROUP BY Matiere) as MoyennesMat ON TABLE.Matiere=MoyennesMat.Matiere) HAVING (MIN(TABLE.Matiere-MoyennesMat.Matiere))>=0

Solution

Solution 13.3 SELECT title FROM books join (SELECT max(words) AS M FROM books WHERE author='J.K. Rowling') WHERE books.words=M
SELECT title FROM books WHERE books.words =(SELECT max(words) AS M FROM books WHERE author='J.K. Rowling')
(la requete (SELECT max(words) AS M FROM books WHERE author='J.K. Rowling') rend une unique valeur, et est donc identifiée à un entier)
On peut aussi trier la table et ne retenir que la première entrée :
SELECT title FROM books order by words DESC limit 1

Solution

Solution 13.3 Voici deux solutions :
SELECT author, count(title) FROM books join (SELECT avg(words) AS m FROM books) WHERE (0.9*m <= words) and (words <=1.1*m) GROUP BY author
Ou bien :
SELECT author, count(title) FROM books WHERE (words>= (SELECT 0.9* avg(words) FROM books)) and (words<= (SELECT 1.1 * avg(words) FROM books)) GROUP BY author

Solution

Solution 13.3 1.

```
SELECT readers.name,  count(distinct borrow.book) AS NbLivres
FROM readers JOIN borrow on readers.id=borrow.reader
GROUP BY readers.name
```

2. Première solution : 1/ trouver le max(words) pour chaque lecteur en créant un table contenant lecteur/-max(words) 2/ trouver le couple (lecteur, words) qui réalise le max dans la table.

```
SELECT DISTINCT readers.name, title
FROM readers JOIN borrow on readers.id=borrow.reader
join books on borrow.book=books.id
WHERE (readers.name, words) in
(SELECT readers.name,  max(words) AS Taille
FROM readers JOIN borrow on readers.id=borrow.reader
join books on borrow.book=books.id
GROUP BY readers.name)
```

Seconde solution : fixons un lecteur ; alors il est facile de savoir quel est le plus gros livre lu. Pour chaque lecteur, prenons le/les livre ayant le plsu grand nombre de mots

```
SELECT reader AS lecteur_ext, books.words, books.title
FROM borrow JOIN books on borrow.book=books.id
WHERE books.words = (SELECT max(words)
FROM borrow JOIN books on borrow.book=books.id
WHERE reader=lecteur_ext)
GROUP BY lecteur_ext
```

Cette requete est beaucoup plus longue en temps d'execution, car on ne demande pas la création d'une vue intermédiaire : le SGBD doit se débrouiller pour gérer les « SELECT max(words) FROM borrow JOIN books on borrow.book=books.id WHERE reader=lecteur_ext » pour chaque valeur de lecteur_ext.

Solution

Solution 13.3 SELECT DISTINCT books.title, books2.title FROM borrow AS b1 JOIN borrow AS b2 on b1.reader=b2.reader join books on books.id=b1.book join books AS books2 on books2.id=b2.book WHERE b1.book<b2.book order by books.title, books2.title