2 Le choix du Système de Gestion de Base de Données

Les concepts présentés dans ces pages ne sont pas liés à un SGBDR en particulier. Ils sont applicables en l'état sur la majorité des systèmes que l'on peut trouver de nos jours. Toutefois, pour les mettre en pratique, il faut bien faire le choix de l'une de ces solutions.

Ce chapitre n'a pas la prétention de faire une liste exhaustive des différents SGBDR, mais de présenter rapidement quatre des plus courants d'entre eux.

2.1 Microsoft® SQL Server

C'est le système de base de données relationnelle de la firme américaine MICROSOFT®.

Avantages	Inconvénients
• Ce moteur est capable de supporter un grand nombre de bases de données. • Il est l'un des plus performants en environnement Microsoft® Windows®. • L'administration et l'optimisation sont facilitées par des outils conviviaux. • Il est disponible en version « Express » gratuite mais pouvant être bridée selon les cas.	• Il fonctionne uniquement sous un environnement Microsoft® Windows®. • Il impose l'acquisition de licences pour le déploiement d'une application professionnelle.

2.2 ORACLE® DATABASE

C'est le système de base de données relationnelle de la firme américaine ORACLE®.

Avantages	Inconvénients
• Ce moteur est capable de supporter un grand nombre de bases de données par instance. • Il est multiplateforme. • Il offre une grande richesse fonctionnelle.	• Le coût des licences. • Son administration est complexe par la richesse fonctionnelle disponible. • Il est un grand consommateur de ressources.

2.3 PostgreSQL

C'est un système de base de données à la fois relationnelle et objet, disponible sous licence BSD, c'est-à-dire sous licence libre permettant la réutilisation de tout ou partie du logiciel sans restriction. Il n'est détenu par aucune entreprise car il repose intégralement sur une communauté.

Avantages	Inconvénients
• Son mode de licence. • Il est multiplateforme. • Son comportement très stable est reconnu. • Il offre des possibilités de programmation avancée, directement dans la base de données. • Il supporte les bases à gros volume et à fort taux de connexion.	• Son déploiement. • Sa stricte conformité à la norme, mais est-ce un inconvénient ?

2.4 MYSQL

C'est un système de base de données relationnelle distribué sous une double licence GPL, licence d'utilisation des logiciels libre sous condition, et propriétaire, détenue par ORACLE® qui ainsi possède deux des SGBD les plus installés du marché.

Avantages	Inconvénient
• Il est multiplateforme. • Il est natif dans la majorité des Framework web. • Il offre des performances élevées en restitution (lecture).	• Il est peu réputé pour les bases à gros volumes, jugement à tempérer avec les dernières évolutions et la notion de « gros » volumes.

2.5 Pour conclure

Une fois évalués les potentiels candidats, le choix du SGBDR peut se faire selon quatre axes :

- Le budget alloué à l'outil et son déploiement.

- La plate-forme sur laquelle il doit fonctionner.
- Le volume de données stockées.
- Le ratio écriture / lecture de ces données.

Pour illustrer les exemples de cet ouvrage, le choix s'est porté sur MySQL, disponible sous licence GPL. Il bénéficie également de modules d'installation « au clic » comme pour la majorité des plateformes, et fait toujours partie du quatuor de déploiement des plateformes web de base :

- **LAMP** : Linux, Apache, MySQL, Php.
- **WAMP** : Windows®, Apache, MySQL, Php.
- **MAMP** : MacOs, Apache, MySQL, Php.

3 Requêtes : interrogations simples

Ce chapitre présente les bases du langage SQL : l'interrogation des données. Une base de données gérant des marathons illustre ce chapitre et les suivants.

3.1 Requêtes de sélection : SELECT

La sélection est ce que l'on fait le plus couramment en SQL. Il s'agit ici d'interroger la base de données afin d'en extraire un ou plusieurs champs, en précisant dans quelle table ils se trouvent.

Voici une requête de sélection présentant l'ordre d'utilisation des commandes SQL.

Syntaxe SQL

```
SELECT <champ1>, <champ2>, …
FROM <table>
WHERE <conditions>
GROUP BY <champ1>, <champ2>, …
HAVING <conditions>
ORDER BY <champ1>, <champ2>, …
LIMIT;
```

☑ ***Exemple 1***

Ci-dessous, le contenu de la table COUREUR de la base de données MARATHON.

Table COUREUR

CO_ID	CO_NOM	CO_PRENOM	CO_NAISSANCE	CO_SEXE 1 Homme / 2 Femme
1	Aldrin	Buzz	1930-01-20	1
2	El Karoui	Nicole	1944-05-29	2
3	Hawking	Stephen	1942-01-08	1
4	Moyo	Dambisa	1969-02-02	2
5	Dawkins	Richard	1941-03-26	1
6	Krauss	Lawrence	1954-05-27	1
7	Lordon	Frédéric	2002-01-15	1
8	Blum	Lenore	1942-12-18	2
9	Laplace	Pierre	1999-03-23	1
10	Cooper	Sheldon	1998-03-26	1
11	Klein	Etienne	1958-04-01	1
12	Boyd Granville	Evelyn	1924-05-01	2
13	Villani	Cédric	1973-10-05	1
14	Bell	Jocelyn	1943-07-15	2
15	Juliot-Curie	Irène	1995-03-04	2
16	Turing	Alan	1990-05-16	1
17	Cood	Edgar Frank	1991-05-18	1

Cette table recense l'intégralité des coureurs, liste longue et lourde à manipuler. Il est donc nécessaire de réaliser une **projection** de certains champs pour rendre cette liste plus utilisable. Cette opération consiste à ne garder que certaines données pour l'affichage.

La **projection** consiste à ne sélectionner que certaines colonnes pour l'affichage.

Pour connaître les noms, prénoms et dates de naissances des coureurs, la requête de sélection est rédigée de la manière suivante :

```
SELECT CO_NOM, CO_PRENOM, CO_NAISSANCE
FROM COUREUR;
```

Le résultat obtenu se présente comme suit :

CO_NOM	CO_PRENOM	CO_NAISSANCE
Aldrin	Buzz	2010-01-20
El Karoui	Nicole	1944-05-29
Hawking	Stephen	1942-01-08
Moyo	Dambisa	1969-02-02
Dawkins	Richard	1941-03-26
Krauss	Lawrence	1954-05-27
Lordon	Frédéric	2002-01-15
Blum	Lenore	1942-12-18
Laplace	Pierre	1999-03-23
Cooper	Sheldon	1998-03-26
Klein	Etienne	1958-04-01
Boyd Granville	Evelyn	1924-05-01
Villani	Cédric	1973-10-05
Bell	Jocelyn	1943-07-15
Juliot-Curie	Irène	1995-03-04
Turing	Alan	1990-05-16
Cood	Edgar Frank	1991-05-18

Pour afficher la totalité d'une table, le caractère joker « * » est utilisé.

Ainsi la requête ci-dessous affiche l'intégralité de la table COUREUR sans faire de projection :

```
SELECT *
FROM COUREUR;
```

3.2 Eviter les doublons : DISTINCT

L'utilisation du mot clé **DISTINCT** permet d'éviter les redondances de lignes dans le résultat d'une requête. Il se place juste après la commande SELECT.

Syntaxe SQL

```
SELECT DISTINCT <champ1>, <champ2>, …
FROM <table> ;
```

☑ ***Exemple 2***

Ci-dessous, le contenu de la table CLUB de la base de données MARATHON.

Table CLUB

CL_ID	CL_NOM	CL_VILLE
1	La foulée bordelaise	Bordeaux
2	Union Athlétique du médoc	Paulliac
3	La foulée arcachonnaise	Arcachon
4	Les coureurs du bassin	Lège
5	Les scientifiques font du sport	Bordeaux
6	Les coureurs de la dune	Lège
7	Les galopins de guyenne	Soulac sur Mer

Pour afficher le nom des villes en évitant les doublons, la requête SQL s'écrit :

```
SELECT DISTINCT CL_VILLE
FROM CLUB;
```

Le résultat obtenu est :

CL_VILLE
Bordeaux
Paulliac
Arcachon
Lège
Soulac sur Mer

3.3 Renommer une colonne : AS

Par soucis de lisibilité du résultat d'une requête, il est possible de renommer temporairement une colonne en utilisant la commande **AS**. Cette commande permet de créer un alias.