

Abréviations

BDD	Base de données
SGBD	Système de gestion de bases de données
SQL	Structured Query Language (<i>en français, langage de requête structurée</i>)

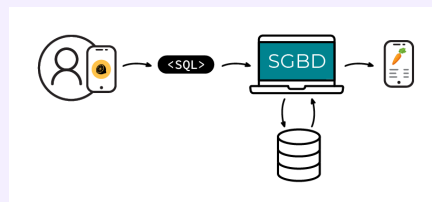
Définitions

SGBD	Logiciel qui va manipuler les données d'une base. C'est ce logiciel qui commande les interactions avec la base pour y récupérer, ajouter, modifier ou supprimer des données.
SQL	Langage informatique qui permet d'interagir avec les bases de données.

Les différents SGBD

MySQL	Le plus connu des SGBD. C'est le plus utilisé, car il était open-source avant son rachat par Oracle Corporation. Connu pour être notamment utilisé par les sites WordPress.
MariaDB	"Copie" open-source de MySQL qui suit les mêmes règles de langage que MySQL.
Oracle Database	Très cher, mais utile pour traiter un très gros volume de données. Ce sont presque exclusivement les grandes entreprises qui l'utilisent. Oracle tend à se faire rattraper par les SGBD open-source type MariaDB ou PostgreSQL. Il est en réelle perte de vitesse sur le marché.
PostgreSQL	Grand SGBD open-source disponible sur le marché. C'est le SGBD qui suit le plus les recommandations du SQL, ainsi que le plus rapide (ces dernières années). Il est notamment utilisé par Instagram ou par Spotify.
SQLite	Il stocke toute la base de données dans un seul et unique fichier. Peu propice à l'utilisation sur un grand nombre de données. Recommandé pour développer une base de données "en local".

Quel est le lien entre le SGBD et le langage SQL ?



Un utilisateur arrive sur Foodly. Il scanne un aliment présent dans son supermarché pour connaître ses caractéristiques nutritionnelles. Que va faire l'application ?

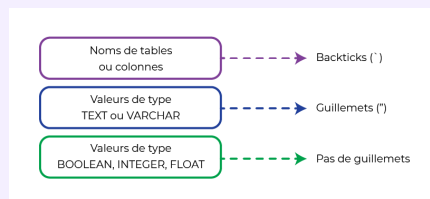
L'application va traduire cette recherche en SQL et l'envoyer **au SGBD**, qui va **récupérer l'aliment** en question dans le stockage de la base de données, pour ensuite le **redonner à l'application**. L'utilisateur retrouvera ainsi son aliment avec toutes ses caractéristiques.



Commandes terminal MySQL

<code>mysql -u root -p</code>	Accéder à MySQL.
<code>mysql -u root -p nom_de_la_base_de_donnees < nom_du_fichier.sql</code>	Permet de "charger" une base de données en une seule commande, en chargeant le fichier.
<code>use nom_de_la_base_de_donnees;</code> <code>source nom_du_fichier.sql;</code>	Permet de "charger" les données d'une bdd en chargeant le fichier.
<code>exit;</code>	Retourner dans le terminal en mode "normal".

Rédaction des valeurs selon leur type



Effectuer des opérations

<code>SELECT COUNT(*)</code> <code>FROM utilisateur</code> <code>WHERE email LIKE " %gm ail.co m";</code>	<code>COUNT (*)</code> permet de compter le nombre de lignes répondant au filtre parmi toutes les colonnes .
<code>SELECT COUNT(nom)</code> <code>FROM utilisateur</code> <code>WHERE email LIKE " %gm ail.co m";</code>	<code>COUNT (nom)</code> permet de compter le nombre de lignes répondant au filtre dans la colonne nom .
<code>SELECT COUNT(DIS TINCT nom)</code> <code>FROM utilisateur</code> <code>WHERE email LIKE " %gm ail.co m";</code>	<code>COUNT (DIS TINCT nom)</code> permet d'éviter les doublons.
<code>SELECT COUNT(DIS TINCT nom) AS " Ali as"</code>	<code>AS</code> permet de créer un alias à la colonne précédemment citée.
<code>SELECT MAX(sucre) AS "taux de sucre maximum"</code> <code>FROM aliment;</code>	Nous donne le maximum de la colonne sur la sélection.
<code>MIN</code>	Nous donne le minimum de la colonne sur la sélection.
<code>AVG</code>	Nous donne la moyenne de la colonne sur la sélection.
<code>SUM</code>	Nous donne la somme de la colonne sur la sélection.
<code>SELECT ROUND(AVG (ca lor ies)) AS " cal ories moyennes des aliments > 30g"</code> <code>FROM aliment</code> <code>WHERE calories > 30;</code>	<code>ROUND(AVG (ca lor ies))</code> permet d'arrondir la valeur retournée.
<code>UPPER</code>	Transformer le texte en majuscules.
<code>NOW</code>	Retourne la date actuelle.

Filtres SQL

WHERE colonne = valeur	WHERE peut s'exécuter avec SELECT, mais aussi avec n'importe quelle autre commande : vous pouvez l'utiliser avec UPDATE ou DELETE pour ne mettre à jour ou supprimer qu'un objet spécifique.
>	Supérieur à
<	Inférieur à
>=	Supérieur ou égal à
<=	Inférieur ou égal à
LIKE = " tex te"	Permet de sélectionner les objets dont le texte d'une colonne répond à un modèle spécifique.
" %gm ail.co m"	Texte se terminant par "gmail.com".
" gma il.c om %"	Texte commençant par "gmail.com".
" %gm ail.co m%"	Texte comprenant "gmail.com" au début ou à la fin.
AND	Permet d'ajouter un filtre supplémentaire avec la condition ET
OR	Permet d'ajouter un filtre supplémentaire avec la condition OU

Le caractère % à un rôle très spécifique. Il va permettre de faire correspondre des schémas spécifiques, on parle parfois de pattern.

Types et options de champs

PRIMARY KEY (option)	Champ spécial obligatoire dans toutes les tables. Indique à MySQL que ce champ sera l'identifiant permettant d' identifier les objets.
NOT NULL (option)	Ce champ ne peut pas être nul .
AUTO_INCREMENT (option)	Ce champ sera créé par MySQL automatiquement , pas besoin de s'en soucier ! MySQL va utiliser l'id précédent et y ajouter +1 lors de l'ajout d'un nouvel objet.
UNIQUE (option)	Ce champ ne peut pas avoir la même valeur en double .
DEFAULT value (option)	Ce champ sert à indiquer une valeur par défaut . Utile pour ne pas avoir à spécifier une valeur tout le temps.
INTEGER (type)	Champ numérique sous forme de nombre entier.
VARCHAR (100) (type)	Champ sous forme de texte , limité à 100 caractères
FLOAT (type)	Ce champ contiendra des chiffres décimaux .
BOOLEAN (type)	Ce champ ne peut stocker que les valeurs true (vrai) ou false (faux).

Une option dans un champ est un attribut *optionnel* qui va **modifier le comportement** de ce champ. Le type lui est obligatoire !



Commandes SQL

<code>CREATE DATABASE nomdel abase;</code>	Création d'une nouvelle base de données.
<code>USE nomdel abase;</code>	Utiliser une base de données.
<code>SHOW DATABASES;</code>	Montre toutes les bases de données.
<code>CREATE TABLE utilis ateur (id INTEGER NOT NULL AUTO_INCREMENT PRIMARY KEY, nom VARCHAR(100), prenom VARCHAR(100), email VARCHAR(255) NOT NULL UNIQUE);</code>	Création d'une table.
<code>SHOW TABLES;</code>	Affiche les tables existantes.
<code>SHOW COLUMNS FROM nomdel atable;</code>	Affiche le schéma d'une table.
<code>FROM nomdel atable</code>	Spécifie la table de la base de donnée qui nous intéresse
<code>INSERT INTO `nomde lat able` (`nom`, `prenom`, `email`) VALUES ('Doe', 'John', 'john.doe@gmail.com'), ('Smith', 'Jane', 'jane@hotmail.com');</code>	Permet d'insérer des données dans une table.
<code>SELECT * FROM `nomde lat able`;</code>	<code>SELECT</code> indique à MySQL que nous souhaitons récupérer de la donnée ; * indique que l'on souhaite récupérer toutes les colonnes (ou champs) présents dans cette table ; <code>FROM</code> permet à MySQL de comprendre depuis quelle table nous souhaitons récupérer de la donnée.
<code>SELECT `nom`, `prenom`, `email` FROM `nomde lat able`;</code>	Indiquer les colonnes à afficher permet de choisir les champs que MySQL va montrer.
<code>UPDATE `nomdelatable` SET `email` = 'iloverammstein@gmail.com' WHERE `id` = '1';</code>	<code>UPDATE</code> table mettre à jour de la donnée en indiquant la table dans laquelle se trouve(nt) le ou les objets que l'on veut modifier. <code>SET</code> colonne = valeur sert à indiquer quelles sont la ou les colonnes à modifier, et quelles sont la ou les valeurs qu'elles doivent désormais prendre.
<code>DELETE FROM `nomdelatable` WHERE `id` = '1';</code>	Supprime une ligne de table en fonction du filtre.
<code>DROP TABLE `nomde lat able`;</code>	Supprime toutes les données d'une table et la table elle-même.
<code>DROP DATABASE `nomde lab ase`;</code>	Supprime entièrement et de façon irréversible la base de données.

Commandes SQL (cont)

```
ORDER BY DATABASE `nom` ASC;
```

Ce mot clé vous permet d'ordonner une colonne par ordre croissant (*ascending* en anglais, d'où le mot clé SQL **ASC**), ou décroissant (*descending* en anglais, soit le mot clé **DESC**).

```
CREATE VIEW utilis_ate urs_gmail_vw AS
( SELECT *
  FROM utilisateur
 WHERE email LIKE "%gmail.com"
);
```

Crée une **vue**. MySQL a un système de "vues" qui permet de créer **tables temporaires** à partir d'une commande SQL. Il est donc possible de "sauvegarder" une commande SQL.

```
SELECT * FROM utilis_ate urs_gmail_vw;
```

Utilisation d'une vue. À noter que **_vw** permet de faire la distinction des "vraies" tables.

```
SELECT *
FROM utilisateur
JOIN langue
ON utilis_ate_urs_langue_id = langue.id;
```

La table `langue` a été jointe grâce à **JOIN langue**. Pour pouvoir faire cette jointure, il faut préciser à MySQL la correspondance entre la table `langue` et la table `utilisateur`. Ici, cette correspondance est effectuée via la clé `langue_id` pour la table `utilisateur` et la table `langue`. Cela se fait grâce à `ON `utilisateur`.`langue_id` = `langue`.`id``.

```
SELECT *
FROM utilisateur
JOIN utilis_ate_urs_aliment
ON (utilisateur.id = utilisateur_aliment.utilisateur_id)
JOIN aliment
ON (aliment.id = utilis_ate_urs_aliment.aliment_id);
```

- MySQL sélectionne tous les utilisateurs avec **SELECT * FROM utilisateur**
- On joint la table `utilis_ate_urs_aliment` avec **JOIN utilis_ate_urs_aliment**
- On relie en considérant que l'id de l'utilisateur est stocké en tant que `utilisateur_id` dans la table `utilis_ate_urs_aliment` **ON (utilisateur.id = utilis_ate_urs_aliment.utilisateur_id)**
- À ce **JOIN**, on lie de la donnée de la table `aliment`, soit un `nouvel aliment` avec **JOIN aliment**
- On précise que l'id de l'aliment est stocké dans `aliment_id` avec **ON (aliment.id = utilis_ate_urs_aliment.aliment_id)**

```
ALTER TABLE nomdel_atable ADD colonne FLOAT;
```

Modification de la table en **ajoutant** une colonne de type `FLOAT`.

```
ALTER TABLE nomdel_atable DROP colonne;
```

Modification de la table en **supprimant** une colonne.

```
ALTER TABLE nomdel_atable MODIFY colonne;
```

Modification de la table en **changeant le type** de la colonne.

```
ALTER TABLE nomdel_atable CHANGE truc colonne;
```

Modification de la table en **changeant le nom** de la colonne.



Commandes SQL (cont)

```
ALTER TABLE aliment
```

```
ADD FOREIGN KEY (famille_id)
```

```
REFERENCES famille (id)
```

```
ON DELETE CASCADE;
```

FOREIGN KEY (famille_id) indique que la colonne "famille_id" est une **clé étrangère**

REFERENCES famille (id) on indique ensuite ce à quoi cette clé fait référence

ON DELETE permet de savoir quel comportement avoir en cas de suppression de la référence

Options :

- RESTRICT ou NO ACTION empêche la suppression sous conditions
- SET NULL supprime la référence et remplace les objets par NULL
- CASCADE supprime tous les objets reliés

```
ALTER TABLE aliment
```

```
ADD FOREIGN KEY (famille_id)
```

```
REFERENCES famille (id)
```

```
ON DELETE CASCADE ON UPDATE CASCADE;
```

Options de ON UPDATE :

- RESTRICT ou NO ACTION empêche la mise à jour sous conditions
- SET NULL met à jour la référence et remplace les objets par NULL
- CASCADE met à jour tous les objets reliés



By Piotezaza (Piotezaza)
cheatography.com/piotezaza/

Published 24th January, 2024.
Last updated 24th January, 2024.
Page 6 of 6.

Sponsored by [Readable.com](https://readable.com)
Measure your website readability!
<https://readable.com>