

Base de données: Qu'est-ce que c'est ?

Histoire et origine des bases de données

Les bases de données, également appelées BBDD ou DB, sont un système qui supporte aujourd'hui le stockage d'informations dans la grande majorité des entreprises du secteur du logiciel. Son origine remonte à 1963, date à laquelle ce terme est apparu pour la première fois.

Au fil du temps, différentes technologies et types ont été développés autour d'elle jusqu'à la naissance des bases de données relationnelles, en particulier, à la fin des années 80, les bases de données SQL ont été créées. Plus précisément, c'est Edgar F. Codd, chercheur chez IBM, qui a défini le concept de SQL qui est aujourd'hui si souvent enseigné dans les universités.

En général, nous n'imaginons pas une société de télécommunications ou une banque avec un fichier de millions de clients, car il y a tellement de personnes qui utilisent ces services que ces informations sont trop limitées. C'est là que nous voyons qu'il doit y avoir un système ou un moyen de stocker les informations de millions de clients.

La première chose à savoir est que les bases de données sont de grands dépôts d'informations et qu'elles sont nées pour organiser l'information. Elles sont d'abord conçues, puis utilisées.

Fonctionnement, exemples et technologies

Comment fonctionne une base de données

En général, une base de données fonctionne dès lors qu'elle est conçue pour être consommée, c'est-à-dire qu'elle est là pour effectuer des opérations sur les informations. Ne vous inquiétez pas, je vais maintenant vous expliquer ce qu'elles sont avec l'exemple d'un guichet de banque :

- **CREATE** Si l'argent est retiré, une nouvelle transaction est générée dans la banque.
- **READ** Pour pouvoir retirer l'argent, le système doit lire et vérifier que le code pin de la carte est correct.
- **UPDATE** Dès que vous retirez de l'argent à un distributeur automatique, le solde de la banque diminue, n'est-ce pas ?
- **DELETE** Pensez au moment où vous perdez votre portefeuille. La carte bancaire est annulée et effacée pour vous en donner une autre.

Création d'une base de données

Il existe des commandes spécifiques pour créer et construire des structures permettant d'accéder ou de créer des informations. Voici un exemple rapide de ce qui se passe lorsque vous créez un compte bancaire avec 1 000 € :

```
CREATE TABLE COMPTE(VINGT_DIGITS, BALANCE);  
INSERT INTO TABLE COMPTE(VINGT_DIGITS,BALANCE)  
VALUES(398249048014324242,1000);
```

C'est le logiciel que l'opérateur bancaire utilise lorsqu'il utilise le logiciel. Ils ne savent pas eux-mêmes ce qui se passe lorsqu'ils cliquent sur le bouton de création de compte.

Types de bases de données

En général, nous pouvons classer les bases de données de nombreuses façons. Cependant, pour savoir ce qu'est une base de données, la classification sera simple : SQL et non-SQL.

1.SQL

Ce sont des bases de données structurées, les SQL sont aussi appelées bases de données relationnelles. Elles sont basées sur les relations entre des entités logiques (tables) structurées pour ensuite utiliser des commandes telles que les opérations CRUD, comme nous l'avons vu dans la section précédente.

Quand utiliser SQL

- Lorsque vous souhaitez que les informations soient robustes et cohérentes au sein de votre organisation.
- Lorsque les données changent au fil du temps
- Lorsque votre entreprise présente des incohérences au niveau des données et qu'elle est appelée à se développer.
- Lorsque vous avez des millions de clients et d'utilisateurs et que le temps de réponse doit être immédiat.

Quand NE PAS utiliser SQL

- Lorsque vous cherchez à analyser des informations massives d'une manière flexible et pas tellement à modifier ou à mettre à jour les informations.
- Lorsque vous êtes une petite entreprise et que vous souhaitez stocker des informations commerciales pertinentes.
- Lorsque vous voulez appliquer l'analyse avancée dans votre entreprise. Les informations sont extraites et exportées vers noSQL très souvent.

Comme nous venons de le voir, les bases de données SQL ne fonctionnent pas toujours et ont leurs inconvénients. C'est là que le noSQL entre en jeu.

2. noSQL

Ces bases de données sont des structures qui nous permettent de stocker des informations de manière massive où, comme leur nom l'indique, elles ne suivent pas un modèle entité-relation où elles ne suivront pas les propriétés SQL. Dans le détail, il existe principalement trois types de bases de données noSQL : document, clé-valeur et graphes. Pour l'instant, sachez simplement que vous devez maîtriser SQL pour être en mesure de gérer noSQL par la suite.

Quand utiliser noSQL

Quand les BD ne sont pas suffisantes pour l'analyse des données

Quand on ne cherche pas à faire des opérations CRUD

Lorsque nous recherchons des projets Big Data avancés

Technologies SQL

Il existe de nombreuses versions de technologies liées aux technologies relationnelles ou SQL. En détail, nous allons mentionner les technologies les plus demandées sur le marché. En général, nous pouvons citer de nombreuses technologies qui utilisent une structure SQL ; cependant, chacune d'entre elles peut avoir de petites différences et/ou des fonctionnalités qui nous feront décider laquelle implémenter dans l'entreprise.

Utilisez [SQLite](#) lui-même pour commencer à programmer à partir de zéro, c'est un système simple et rapide.

Si vous voulez apprendre à utiliser SQL, c'est vraiment suffisant a priori. Maîtriser l'une d'entre elles en général et passer ensuite à une autre technologie est très facile.

Enfin, il faut noter que, comme vous pouvez l'imaginer, la gestion des bases de données est étroitement liée aux projets Big Data, car les informations sont souvent déjà stockées et organisées. C'est pourquoi la maîtrise de SQL est l'un des fondamentaux dans cet univers.

Commandes de base pour les débutants : Exemples de requêtes

Les opérations CRUD, de Create-Read-Update-Delete. Elles sont essentielles pour créer, lire, mettre à jour et supprimer du contenu. Les voici :

Commande de création de données :

```
INSERT INTO table (colonne1, colonne2) VALUES ('valeur1', 'valeur2');
```

Commande permettant de lire les données d'une table :

```
SELECT (colonne1, colonne2) FROM table WHERE colonne=critère;
```

Commande permettant de mettre à jour une table :

```
UPDATE carte SET mdp=1234 WHERE id = '789227421A';
```

Commande permettant de supprimer une ligne d'une table :

```
DELETE FROM compte_courant WHERE vingt_digits =  
27977902348632494135;
```

Si vous voulez maîtriser les piliers et les principes fondamentaux d'une base de données (BD), vous devez apprendre à concevoir une **BD relationnelle** avec le **diagramme entité-relation**.

Glossaire des termes SQL de base

SQL : Fait référence à Structured Query Language, langage de requête. SQL est un langage permettant de stocker des données et d'effectuer des requêtes avec celles-ci de manière organisée.

Ligne : Désigne une liste d'éléments qui composent une base de données, avec des valeurs pour chaque colonne.

Tableau : Renvoie à l'idée des données que nous avons reliées par un contexte. Imaginez une feuille Excel.

Colonne : Un tableau est composé de plusieurs colonnes, comme dans Excel. Cependant, en SQL, nous devons définir comment les données de cette colonne seront à jamais. Exemple : la colonne ID doit comporter 9 chiffres et une lettre majuscule.

SGBDR : Système de gestion de base de données relationnelle. Désigne le système de gestion derrière. *PostGree*, *SQLite*, *SQLServer*... Ces systèmes veillent au bon fonctionnement de la base de données.

BBDD ou DB : Ils viennent de l'acronyme Bases des Données ou DataBase en anglais.

Entité - Relation : C'est la manière dont on conçoit la Base de données avant de la programmer, où l'on situe les entités puis les relations.

Clé primaire : cela fait référence à l'idée que dans une base de données SQL, chaque ligne doit être identifiée de manière unique. Pensez à un numéro de téléphone, il est unique, n'est-ce pas ?

Requête : Le code écrit en SQL qui doit être programmé pour répondre à la question.

CRUD : Create-Read-Update-Delete. Cela fait référence au fait que SQL est préparé pour créer, modifier, supprimer et lire les données des tables.

Transaction : Un groupe d'opérations qui ne peuvent être exécutées que si elles se produisent toutes.

Comment créer une base de données en 3 étapes

1) Définir le schéma entité-relation

Pour maîtriser SQL et créer une base de données, vous devez comprendre ce qu'est une entité, la relation entre elles, ce que signifie la clé primaire, puis la programmer.

2. Écrire les commandes

Une fois que nous avons conçu la base de données, nous commençons à programmer la création de la base de données avec les commandes `CREATE SCHEMA, USER, TABLE...` et les informations populaires avec les commandes `INSERT INTO TABLE....`

3. Commencez à utiliser la base de données

À partir de là, la base de données est activée et prête à être utilisée, par exemple, par Netflix en interne. En d'autres termes, le système de films, d'applications, d'utilisateurs... est en fait conçu en premier, afin qu'il puisse ensuite être connecté et gérer les données de manière évolutive.

Maintenant, passons aux exercices. Mais d'abord, nous vous laissons avec une règle d'or pour comprendre comment le code est écrit et comment SQL fonctionne en interne :

La syntaxe SQL : La règle d'or

Ensuite, n'oubliez pas que vous devez respecter la règle d'or pour votre langage de requête:

```
SELECT fieldsIWantToView - STEP3
```

```
FROM table - STEP1
```

```
WHERE cond1 et cond2 et cond3 - STEP2 ;
```

Vous écrivez toujours SELECT FROM WHERE dans cet ordre. En outre, vous disposez des étapes données en interne par SQL.

EXERCICES

<https://www.programiz.com/sql/online-compiler/>

`SELECT * FROM Customers;` – Retourne toutes les informations de la table

`SELECT * FROM Customers WHERE age<28;`

Where SQL : Conditions multiples

Nous pouvons également placer plusieurs conditions pour filtrer l'information. Nous pouvons combiner autant de critères que nous le souhaitons de la manière suivante :

```
SELECT *  
FROM table  
WHERE condition1 AND/OR condition2 AND/OR condition3...
```

Exercices avec Where en SQL

Exercice 1

En partant de la table CUSTOMERS, trouvez seulement ceux qui peuvent être vendus à des personnes de 25 ans ou plus.

Exercice 2

A partir du tableau ORDERS, trouve les commandes d'une quantité supérieure à 300 éléments.

Exercice 3

Dans le tableau SHIPPINGS, trouve celles qui sont en status PENDING.

Exercice 4

Dans le tableau ORDERS, trouve ceux qui correspondent au client No. 4.

Exercice 5

Dans le tableau ORDERS, trouve ceux qui correspondent au client No. 4 et qui sont des souris (MOUSE)

SQL Where LIKE

Avant d'aller plus loin avec SQL Where LIKE, nous devons préciser que cet article appartient au sujet des Bases de données où nous exposons que, aujourd'hui dans l'industrie, le langage SQL est principalement utilisé. Une fois que nous commençons à travailler sur le

filtrage des opérations SQL avec le WHERE, nous devons manipuler des outils tels que le LIKE.

Comme nous le savons déjà, la programmation et les bases de données sont la base pour connaître, travailler ou étudier dans le monde du Big Data ou pour construire et créer des IA. Sans plus attendre, allons-y.

LIKE est un opérateur extrêmement puissant du langage SQL, qui permet aux utilisateurs de rechercher rapidement des modèles de données. Il s'agit d'une excellente alternative à la recherche standard par mot-clé lors de la recherche d'un ensemble spécifique de données, car il permet aux utilisateurs d'effectuer des recherches variables plutôt que de devoir utiliser des mots ou des phrases exacts. Elle est particulièrement utile pour les recherches de texte qui incluent des modèles.

Par exemple, si vous souhaitez rechercher toutes les polices de caractères commençant par "neo", avec LIKE, vous pouvez facilement faire la distinction entre "neon" et "neoprene". En résumé, l'utilisation de la fonction LIKE pour les motifs peut être un outil fondamental pour trouver ce que vous cherchez ; c'est l'une des parties les plus utiles de Where en SQL.

Comment fonctionne LIKE

Dans le where, like fonctionne en filtrant les informations de la manière suivante : Filtrage par critères, par motif ou par caractère générique. Nous pouvons définir différents modèles... Des exemples très courants que nous avons dans notre vie quotidienne :

- Filtrage par ceux qui commencent par une lettre ou qui se terminent par elle.
- Filtrage par position
- Seulement ceux qui répondent à un critère combiné

Comment utiliser l'application Like ?

Elle est placée à côté du WHERE et vous définissez la colonne à laquelle vous voulez appliquer le modèle de recherche. Disons que l'application LIKE est largement utilisée pour commencer à mieux filtrer les informations que l'on veut obtenir à partir de millions de données déjà organisées et dans des tables.

Caractères génériques ou motifs de recherche dans LIKE (SQL)

1. utilisation de % en SQL

L'utilisation de LIKE avec le motif % représente zéro, un ou plusieurs caractères. Voyons un exemple avec notre table CUSTOMERS :

```
SELECT * FROM Customers WHERE age LIKE ('200');
```

Cette fois, le LIKE sélectionnera tout ce qui commence par 2 en discriminant le reste. Une fois que cela est donné, il filtrera les lignes contenant 0, un ou plusieurs caractères.

2. Utilisation du caractère générique _ dans LIKE

D'autre part, un autre motif ou joker très courant est le _ dans **LIKE**. Nous pouvons appliquer l'utilisation de ce joker pour nous assurer qu'il y a un nombre fixe d'éléments. Imaginez que vous ayez besoin d'obtenir toutes les cartes comportant plus de quatre zéros pour une campagne :

```
SELECT * FROM Orders WHERE amount LIKE ('_____') ; --Il y a  
5 _ s
```

4. Exemple d'utilisation de NOT

Vous pouvez également utiliser l'objet pour filtrer sur la base de la négation des cas que vous recherchez. L'exemple est l'opérateur NOT dans le LIKE.

Imaginez le tableau ci-dessus pour Customers. Pour rechercher ceux dont le nom de famille ne se termine pas par Z, nous pourrions faire ceci :

```
SELECT * FROM customers WHERE NOT last_name LIKE ('%t') ;
```

Exercices SQL Where Like

Exercice 1

Obtenez les personnes de la table Customers dont leur last_name commence par R.

Exercice 2

A partir du tableau Shippings, nous voulons savoir lesquelles n'ont toujours pas été livrées.

Exercice 3

A partir de la table Customers on obtient ceux dont le prénom commence par J.

Exercice 4

Obtenez les comptes de la table Orders où la quantité n'est PAS à 3 chiffres.

Exercice 5

Dans la table Customers, trouve ceux dont le prénom commence par D.

Exercice 6

A partir de la table Customers, récupère tous les clients dont le pays n'est pas USA.

Between SQL : Comment ça marche

Jetons un coup d'œil à l'opérateur BETWEEN en SQL et à la façon dont il peut vous aider à en savoir plus sur la gestion des bases de données. Grâce à cet opérateur, vous serez en mesure de filtrer les données de manière plus intelligente. Alors, sans plus attendre, commençons !

Between : les syntaxes SQL

La première chose que vous devez savoir et qui vous aidera à comprendre cet opérateur est que between fait partie des opérateurs WHERE. La manière dont l'instruction between est utilisée et sa syntaxe en SQL sont les suivantes :

```
SELECT colonne  
FROM table WHERE colonne  
BETWEEN la valeur1 AND valeur2 ;
```

Elle fonctionne en permettant aux programmeurs de sélectionner rapidement des données dans une plage. Par exemple, au lieu d'écrire plusieurs lignes de code spécifiant des conditions pour chaque valeur individuelle de la plage, l'opérateur Between est utilisé pour créer une seule instruction qui effectue une requête dans ces valeurs.

Exemples de code avec Between en SQL

Imaginez que vous préparez une campagne d'assurance vie mais pour les actifs. Nous avons besoin d'obtenir des données entre les personnes âgées mais pas les retraités :

```
SELECT * FROM customers  
  
WHERE age  
  
BETWEEN 18 AND 65;
```

Imaginez que vous ayez besoin de filtrer les informations pour lancer une offre personnalisée par différents noms de famille. Nous avons besoin d'interroger tous les noms de 'JUAN' à 'JOAQUIN' :

```
SELECT * FROM customers  
  
WHERE first_name  
  
BETWEEN 'John' AND 'Robert' ;
```

Exercices avec Between

Exercice 1

Obtenez les noms des Orders détenant une quantité entre 300 et 1000.

Exercice 2

Localisez les clients entre 'DOE' et 'REINHARDT'.

Exercice 3

Localisez les clients dont leur âge est supérieur à 22 et inférieur à 30.

Exercice 4

Localisez les clients dont le pays est entre UK et USA.

SQL Where in : Exemple et utilisation

L'une des requêtes SQL typiques est l'utilisation de l'opérateur IN. Imaginez que vous ayez besoin de collecter dans votre base de données exactement des chaussures ORANGE, BLEU, VERT ou BLANC. L'opérateur IN est utilisé dans ce cas, car contrairement à l'opérateur LIKE, nous ne recherchons pas des modèles, mais des valeurs concrètes possibles.

Syntaxe et utilisation de IN en SQL

L'exemple est la requête suivante :

```
SELECT *  
  
FROM customers  
  
WHERE country IN ('USA', 'UK') ;
```

La principale différence entre LIKE en SQL et IN est que LIKE recherche des modèles, tandis que IN effectue des recherches exactes.

Exercices pour pratiquer IN en SQL dans WHERE

Exercice 1

Du tableau CUSTOMERS, trouver des informations sur Robert ou David, mais uniquement sur ces deux-là.

Exercice 2

Collecter toutes les informations sur les clients non premium. Les clients premium sont Robert et David.

Exercice 3

Collecter toutes les informations sur les orders qui ne soient ni souris ni monitor.

Exercice 4

Collecter toutes les informations sur les shippings qui ne sont pas livrés.

