RÉSURGENCES INSERTION

ACCATONE

SQL
Structured Query Language
SELECT

Objectifs de ce cours

Ce cours va vous permettre de comprendre le fonctionnement d'une base de donnée et les principales requêtes de base du langage SQL.

Objectif(s) pédagogique(s) :

- Découvrir les subtilités du SELECT

Version 19 mars 2021, Alexis Brou.

Plan du cours

- Formuler des requêtes SQL pour extraire des données
- Les prédicats
- WHERE
- AS / ORDER BY

Formuler des requêtes SQL pour extraire des données

SELECT

La clause SELECT décrit la relation résultat, <attributs> désigne :

- soit une liste d'attributs ;
- soit une expression obtenue à l'aide des fonctions statistiques SUM (somme), AVG (moyenne), COUNT (compte), MIN et MAX ;
- soit une expression

Formuler des requêtes SQL pour extraire des données

SELECT

SELECT champ1, champ2 FROM table

Affiche les champs choisis de la table.

On peut utiliser * à la place de l'énumération des champs, pour récupérer TOUS les champs d'un coup.

FROM permet de sélectionner une ou plusieurs tables

Formuler des requêtes SQL pour extraire des données

SELECT

La clause SELECT et la clause FROM sont liées. En effet, lors de chaque opération de sélection,
le mot-clé FROM permet d'identifier la ou les tables nécessaires à la construction du résultat de la
requête.

Formuler des requêtes SQL pour extraire des données

SELECT

Table Clients

nom	ville
Claudio	Marseille
Jean	Paris
Alfonso	Paris
Marie	Lyon
Nathalie	Marseille
Jo	Paris

ville
Marseille
Paris
Paris
Lyon
Marseille
Paris

SELECT * FROM Clients
SELECT ville FROM Clients

le symbole * dans la clause SELECT signifie qu'on désire récupérer TOUS les champs de la table.

Formuler des requêtes SQL pour extraire des données

SELECT DISTINCT

SELECT DISTINCT champ1 FROM table

DISTINCT permet de ne pas afficher les doublons de lignes lors de l'affichage des résultats.

Formuler des requêtes SQL pour extraire des données

LIMIT

- SELECT * FROM table **LIMIT** x,y

permet d'afficher un nombre limité de résultat.

On commence par le résultat d'indice x et on affiche y résultat. (en sachant que les résultats sont indexés en commençant par 0)

Les prédicats

Une condition est appelée prédicat en SQL. Un prédicat permet de comparer 2 expressions de valeurs :

- la première expression contenant des spécifications de colonnes est appelée terme ;
- la seconde expression contenant seulement des spécifications de constantes est appelée constante.

Il existe une grande diversité de prédicats en SQL, on trouve en effet :

Les prédicats

- un prédicat de comparaison permettant de comparer un terme à une constante à l'aide des opérateurs suivants :

  - = : égal

  - <> ou != : différent

  - > : plus grand que

  - >= : plus grand ou égal

  - < : plus petit que

  - <= : plus petit ou égal

Les prédicats

Ces opérateurs sont valables pour les types INT, FLOAT,DECIMAL ,CHAR, VARCHAR, DATE et valent NULL si un des termes de la comparaison est NULL.

- un prédicat d'intervalle BETWEEN permettant de tester si la valeur d'un terme est comprise entre la valeur de 2 constantes

Les prédicats

- Un prédicat de comparaison de texte noté LIKE permettant de tester si un terme de type chaîne de caractères contient une ou plusieurs sous-chaînes (ce prédicat est donc réservé aux types CHAR et VARCHAR). Le caractère % remplace une chaîne de caractères quelconque y compris la chaîne vide.
- Un prédicat de test de nullité qui permet de tester si un terme a une valeur convenue NULL,signifiant que sa valeur est inconnue ou que le champ n'a pas été renseigné. Pour tester la nullité, on utilise IS NULL, et la non nullité par IS NOT NULL ;

Les prédicats

- Un prédicat d'appartenance noté IN qui permet de tester si la valeur d'un terme appartient à une liste de valeurs constantes

WHERE

La commande WHERE dans une requête SQL permet d'extraire les lignes d'une base de données qui respectent une condition. Cela permet d'obtenir uniquement les informations désirées.

- **Vous pouvez utiliser plusieurs conditions dans le** WHERE **en les séparant par** AND ou OR **selon le résultat voulu.**

WHERE

Pour les nombres, on utilise les comparateurs.

WHERE salaire >= 1887.50

WHERE id = 128

WHERE club_id != 6

Pour les varchar, on peut utiliser LIKE à la place de =, cela vous permet d'utiliser le caractère %.

WHERE

Opérateur de comparaison :

<, <=, >,>=,!

Comparaison NULL

WHERE colonne IS NULL(ou IS NOT NULL)

WHERE

BETWEEN AND :

SELECT prenom FROM employe

WHERE salaire BETWEEN 14000 AND 21500;

IN

SELECT prenom FROM employe

WHERE salaire IN (10000, 20000);

WHERE

LIKE :

WHERE nom LIKE 'M%' commence par M

WHERE nom LIKE '%Z' finit par z

WHERE nom LIKE '%P%' contient P

AND ET OR

WHERE sexe = 'M' AND salaire > 12000

WHERE sexe = 'M' OR salaire > 12000;;

AS / ORDER BY

.**Renommer un champ ou une table (faire un alias)**

.**Utile pour renommer un champ qui provient d'une fonction (sinon le nom du champ est le nom de la fonction) mais marche avec tous les champs.**

.SELECT ville, AVG(age) as 'moyenne' from clients GROUP BY ville

AS / ORDER BY

Permet aussi de faire un alias (raccourci) de table

Dans les jointures, on peut renommer temporairement une table A pour n'avoir à écrire que A au lieu du nom complet dans les requêtes

- `SELECT C.ville, AVG(age) as 'moyenne' from clients as C GROUP BY C.ville`

AS / ORDER BY

.SELECT * FROM table **ORDER BY** champ DESC *(ou ASC)*

Permet de classer les résultats en classant par ordre croissant le champ sélectionné.

Il faut rajouter DESC après le champ pour avoir l'ordre décroissant (ASC est implicite).

On peut trier avec plusieurs critères en les séparant par des virgules

ORDER BY champ1 DESC,champ2 *ASC*