SQL v2.0 20/07/15

DATABASE

CREATE DATABASE nomdatabase;
Crée une database
SHOW DATABASES;
Montre les databases existantes
DROP DATABASE nomdatabase;
Supprime la database nommé
USE nombase;
Utilise la database nommé

TABLE

CREATE TABLE Nomtable (
Nomcolonne TYPE**(**option**)**,
Nomcolonne2 TYPE**(**option**)**,
PRIMARY KEY (id));
Création de la table, nom et option
SHOW TABLES;
Liste des tables existantes dans la database
DROP TABLE Nomtable;
Efface la table nommée
TRUNCATE TABLE table_name
Vide la table (Efface les données, pas la table)
SHOW COLUMNS FROM Nomtable ;
Montre les colonnes dans la table nommée
OPTIMIZE TABLE Relation ;

- Si la table contient des lignes effacées ou des lignes fragmentées, la table est compactée.
- Si les pages d'index ne sont pas triées, `OPTIMIZE TABLE` les trie. Si les statistiques ne sont pas à jour (et que la table n'a pas pu effectuer de réparation en triant l'index), elles sont mises à jour

OPERATEURS

Opérateurs de la clause WHERE

=	Equal
<>	Not equal. **Note:** In some versions of SQL this operator may be written as !=
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal
BETWEEN	Between an inclusive range
LIKE	Search for a pattern
IN	To specify multiple possible values for a column

WHERE Country='Germany'
AND City='Berlin';
Opérateur AND

WHERE City='Berlin'
OR City='München';
Opérateur OR

WHERE Country='Germany'
AND (City='Berlin' **OR** City='München');
Opérateur AND et OR

SELECT *column_name(s)*
FROM *table_name*
WHERE *column_name* **LIKE** *pattern*;
Ex : **SELECT** * **FROM** Customers
WHERE City **LIKE** 's%'; //Qui commence par s
'%s'; //Qui finit par s
'%land%'; //Qui contient land
'%land%'; //Qui ne contient pas land
'_erlin'; //Commence par un caractère quelconque puis erlin
'[bsp]%'; //Qui commence par b, s ou p
'[!bsp]%'; // Qui ne commence PAS par b, s ou p
'[a-c]%'; //Commence par a, b ou c

SELECT *column_name(s)*
FROM *table_name*
WHERE *column_name* **IN** (*value1*,*value2*,...);
EX : **SELECT** * **FROM** Customers
WHERE City **IN** ('Paris','London');
Lister en spécifiant plusieurs valeurs dans le WHERE

SELECT *column_name(s)*
FROM *table_name*
WHERE *column_name* **BETWEEN** *value1* **AND** *value2;*
EX : **SELECT** * **FROM** Products
WHERE Price **BETWEEN** 10 **AND** 20;
EX : **SELECT** * **FROM** Products
WHERE Price **NOT BETWEEN** 10 **AND** 20;
EX : **SELECT** * **FROM** Products
WHERE (Price **BETWEEN** 10 **AND** 20)
AND NOT CategoryID **IN** (1,2,3);
EX : **SELECT** * **FROM** Orders
WHERE OrderDate **BETWEEN** #07/04/1996# **AND** #07/09/1996#;
Lister en spécifiant plusieurs valeurs dans le WHERE

MODIFICATIONS STRUCTURE TABLE

ALTER TABLE "nom de table"
[alter spécifications];
Ex : **ALTER TABLE** nomtable
ADD nomcolonne TYPE(option);
Ajouter une colonne sur la table nommée
ALTER TABLE nomtable **DROP** nomcolonne;
Supprimer la colonne dans la table nommée
ALTER TABLE nomtable

CHANGE anciennomcolonne nouveaunom TYPE(option);
Changer le nom de la colonne dans la table nommée
ALTER TABLE nomtable
MODIFY nomcolonne char(30);
Modifier le type de donnée de la colonne dans la table nommée

LISTER LES DONNEES

SELECT Nomcolonne **FROM** Nomtable **WHERE** Condition;
Ex : **SELECT** Nom, DateNaissance **FROM** Personne **WHERE** DateNaissance >= '1980-01-01';
Ex : **SELECT** * **FROM** produit;
Ex : **SELECT** * **FROM** personne **WHERE** datenaissance **LIKE** "1995%" ;
Lister des données dans la table nommée

SELECT DISTINCT *column_name,column_name*
FROM *table_name*;
EX : **SELECT DISTINCT** City **FROM** Customers;
Lister des données uniquement distinctes (différentes)

SELECT *column_name(s)*
FROM *table_name*
LIMIT *number*;
EX : **SELECT** * **FROM** Persons **LIMIT** 5;
Lister un nombre précis de données

SELECT *column_name* **AS** *alias_name*
FROM *table_name;*
Ou
SELECT *column_name(s)*
FROM *table_name* **AS** *alias_name;*
//Alias

SELECT CustomerName, Address+', '+City+', '+PostalCode+', '+Country **AS** Address
FROM Customers;
// Afficher plusieurs champs dans une cellule

Address
Obere Str. 57, Berlin, 12209, Germany

Commentaire

```
SELECT * -- tout sélectionner
//ou
SELECT * # tout sélectionner
//ou
SELECT * /* tout selectionner */
```

EXIST (SQL)

```
SELECT *
FROM commande
WHERE EXISTS (
SELECT *
FROM produit
WHERE c_produit_id = p_id)
```

// La requete externe s'executera (SQL) uniquement si la requete interne retourne au moins un resultat.

Condition ALL (SQL)

```
SELECT *
FROM table1
WHERE condition > ALL (
SELECT *
FROM table2
WHERE condition2)
```

//ALL permet de comparer une valeur dans l'ensemble de valeurs
d'une sous-requete.

ANY / SOME (SQL)

```
SELECT *
FROM table1
WHERE condition > ANY (
SELECT *
FROM table2
WHERE condition2 )
```

// ANY (ou SOME) permet de comparer une valeur avec le resultat d'une sous-requête.

CASE (SQL)

```
SELECT id, nom, marge_pourcentage, prix_unitaire
CASE
WHEN marge_pourcentage=1 THEN 'Prix ordinaire'
WHEN marge_pourcentage>1 THEN 'Prix superieur a la normale'
ELSE 'Prix inferieur a la normale'
END
FROM `achat`
//OU
UPDATE `achat`
SET `quantite` = (
CASE
WHEN `surcharge` < 1 THEN `quantite` + 1
WHEN `surcharge` > 1 THEN `quantite` - 1
ELSE quantite
END
)
```

EXPLAIN

```
EXPLAIN SELECT *
FROM `user`
ORDER BY `id` DESC
```

// permet d'afficher
le plan d'execution d'une requete SQL.

GESTION DES DONNEES

INSERT INTO "nom de table" ("colonne 1", "colonne 2", ...)
VALUES ("valeur 1", "valeur 2", ...);
Ex : **INSERT INTO** Personne (Nom, Prenom, DateNaissance, Email) **VALUES** ('DUVAL', 'Guillaume', '2013-09-17', 'guillaumeduval93@gmail.com'); Fin Ex
Insertion de valeurs dans les colonnes de la table nommée

UPDATE *table_name*
SET *column1=value1,column2=value2,...*
WHERE *some_column=some_value*;
Ex : UPDATE Customers
SET ContactName='Alfred Schmidt', City='Hamburg'
WHERE CustomerName='Alfreds Futterkiste';
Modifier les données contenues dans une table
DELETE FROM *table_name*
WHERE *some_column=some_value*;
EX : **DELETE FROM** Customers
WHERE CustomerName='Alfreds Futterkiste'
AND ContactName='Maria Anders';
Supprimer des données dans la colonne d'une table à un endroit précis

AGGREGATE FUNCTIONS

AVG() - Returns the average value
SELECT AVG(column_name) **FROM** table_name
EX : **SELECT** ProductName, Price **FROM** Products
WHERE Price>(**SELECT AVG**(Price) **FROM** Products);

COUNT() - Returns the number of rows
SELECT COUNT(column_name) **FROM** table_name;
EX : **SELECT COUNT**(**DISTINCT** CustomerID) **AS** NumberOfCustomers **FROM** Orders;

FIRST() - Returns the first value
SELECT *column_name* **FROM** *table_name*
ORDER BY *column_name* **ASC**
LIMIT 1;
The FIRST() function is only supported in MS Access

LAST() - Returns the last value
EX : **SELECT** *column_name* **FROM** *table_name*
ORDER BY *column_name* **DESC**
LIMIT 1;
The LAST() function is only supported in MS Access.

MAX() - Returns the largest value
SELECT MAX(column_name) **FROM** table_name;
EX : **SELECT MAX**(Price) **AS** HighestPrice **FROM** Products;

MIN() - Returns the smallest value
SELECT MIN(column_name) **FROM** table_name;
EX : **SELECT MIN**(Price) **AS** SmallestOrderPrice **FROM** Products;

SUM() - Returns the sum
SELECT SUM(column_name) **FROM** table_name;
EX : **SELECT SUM**(Quantity) **AS** TotalItemsOrdered
FROM OrderDetails;

GROUP BY

SELECT column_name,
aggregate_function(column_name)
FROM table_name
WHERE column_name operator value
GROUP BY column_name;
EX : **SELECT**
Shippers.ShipperName,**COUNT**(Orders.OrderID) **AS** NumberOfOrders **FROM** Orders
LEFT JOIN Shippers
ON Orders.ShipperID=Shippers.ShipperID
GROUP BY ShipperName;
EX : **SELECT** Shippers.ShipperName,
Employees.LastName,
COUNT(Orders.OrderID) **AS** NumberOfOrders
FROM ((Orders
INNER JOIN Shippers
ON Orders.ShipperID=Shippers.ShipperID)
INNER JOIN Employees
ON Orders.EmployeeID=Employees.EmployeeID)
GROUP BY ShipperName,LastName;

HAVING BY

column_name, aggregate_function(column_name)
FROM table_name
WHERE column_name operator value
GROUP BY column_name
HAVING aggregate_function(column_name) operator value;
EX : **SELECT** Employees.LastName,
COUNT(Orders.OrderID) **AS** NumberOfOrders **FROM** (Orders
INNER JOIN Employees
ON Orders.EmployeeID=Employees.EmployeeID)
GROUP BY LastName
HAVING COUNT(Orders.OrderID) > 10;
EX : **SELECT** Employees.LastName,
COUNT(Orders.OrderID) **AS** NumberOfOrders **FROM** Orders
INNER JOIN Employees
ON Orders.EmployeeID=Employees.EmployeeID
WHERE LastName='Davolio' **OR** LastName='Fuller'
GROUP BY LastName
HAVING COUNT(Orders.OrderID) > 25;

SCALAR FUNCTIONS

UCASE() - Converts a field to upper case
SELECT UCASE(column_name) **FROM** table_name;
EX : **SELECT UCASE**(CustomerName) **AS** Customer, City
FROM Customers;

LCASE() - Converts a field to lower case
SELECT LCASE(column_name) **FROM** table_name;
EX : **SELECT LCASE**(CustomerName) **AS** Customer, City
FROM Customers;

MID() - Extract characters from a text field
SELECT MID(column_name,start[,length]) **AS** *some_name*
FROM table_name;
EX : **SELECT MID**(City,1,4) **AS** ShortCity
FROM Customers;

LEN() - Returns the length of a text field
SELECT LEN(column_name) **FROM** table_name;
EX : **SELECT** CustomerName,**LEN**(Address) as LengthOfAddress
FROM Customers;

ROUND() - Rounds a numeric field to the number of decimals specified
SELECT ROUND(column_name,decimals) **FROM** table_name;
EX : **SELECT** ProductName, **ROUND**(Price,0) **AS** RoundedPrice
FROM Products;

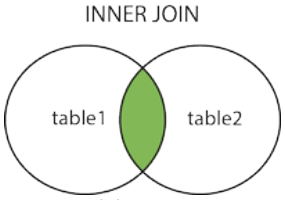
NOW() - Returns the current system date and time
SELECT NOW() **FROM** table_name;
EX : **SELECT** ProductName, Price, Now() **AS** PerDate
FROM Products;

FORMAT() - Formats how a field is to be displayed
SELECT FORMAT(column_name,format) **FROM** table_name;
EX : **SELECT** ProductName, Price,
FORMAT(Now(),'YYYY-MM-DD') **AS** PerDate
FROM Products;

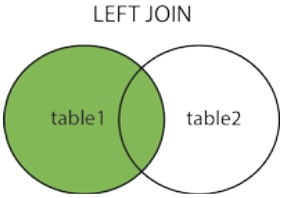
FOREIGN KEY

CREATE TABLE Nomtable (
Nomcolonne **TYPE**(option),
PRIMARY KEY (id),
FOREIGN KEY (colonnetableFK) **REFERENCES** tablePK (colonne)); Créer une ForeignKey directement avec la table
ALTER TABLE Nomtable
ADD FOREIGN KEY (colonnetableFK)
REFERENCES tablePK (colonne);
Ajouter une ForeignKey à une table déjà crée

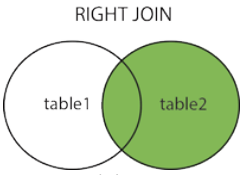
JOIN



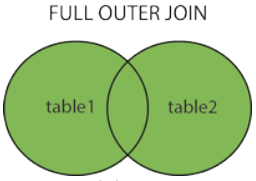
SELECT table1.colonne, table2.colonne
FROM table1
JOIN table2 **ON** table1.colonne = table2.colonne;
EX : **SELECT** Produit.id, Produit.libelle, Categorie.libelle
FROM Produit
JOIN Categorie **ON** produit.categorieid = Categorie.id;
EX: **SELECT** Orders.OrderID, Customers.CustomerName, Orders.OrderDate
FROM Orders
INNER JOIN Customers
ON Orders.CustomerID=Customers.CustomerID;
Afficher plusieurs colonnes venant de plusieurs tables
(**JOIN = INNER JOIN**)



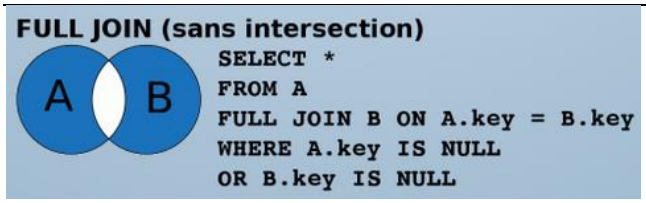
SELECT *column_name(s)*
FROM *table1*
LEFT JOIN *table2*
ON *table1.column_name=table2.column_name*;
EX : **SELECT** Customers.CustomerName, Orders.OrderID
FROM Customers
LEFT JOIN Orders
ON Customers.CustomerID=Orders.CustomerID
ORDER BY Customers.CustomerName;
LEFT JOIN = LEFT OUTER JOIN



SELECT *column_name(s)*
FROM *table1*
RIGHT JOIN *table2*
ON *table1.column_name=table2.column_name*;
EX :**SELECT** Orders.OrderID, Employees.FirstName
FROM Orders
RIGHT JOIN Employees
ON Orders.EmployeeID=Employees.EmployeeID
ORDER BY Orders.OrderID;
RIGHT JOIN = RIGHT OUTER JOIN



SELECT *column_name(s)*
FROM *table1*
FULL OUTER JOIN *table2*
ON *table1.column_name=table2.column_name*;
EX : **SELECT** Customers.CustomerName, Orders.OrderID
FROM Customers
FULL OUTER JOIN Orders
ON Customers.CustomerID=Orders.CustomerID
ORDER BY Customers.CustomerName;



UNION

SELECT *column_name(s)* **FROM** *table1*
UNION
SELECT *column_name(s)* **FROM** *table2*;
L'opérateur **UNION** sélectionne uniquement des valeurs distinctes par défaut.

SELECT *column_name(s)* **FROM** *table1*
UNION ALL
SELECT *column_name(s)* **FROM** *table2*;
Pour permettre des valeurs en double, utilisez le mot clé **ALL** avec **UNION**

SELECT City, Country **FROM** Customers
WHERE Country='Germany'
UNION ALL
SELECT City, Country **FROM** Suppliers
WHERE Country='Germany'
ORDER BY City;
Utilise **UNION ALL** pour sélectionner toutes les valeurs de duplicata Ex :les villes allemandes des tables " Customers " et " Suppliers "

SELECT INTO

L'instruction SELECT INTO sélectionne les données d'une table et l'insère dans une nouvelle table.
SELECT *
INTO CustomersBackup2013
FROM Customers;
Créer une copie backup d'une table

SELECT *
INTO CustomersBackup2013 **IN** 'Backup.mdb'
FROM Customers;
Créer une copie dans une autre DB

SELECT CustomerName, ContactName
INTO CustomersBackup2013
FROM Customers;
Copier seulement quelques colonnes dans une nouvelle table

SELECT Customers.CustomerName, Orders.OrderID
INTO CustomersOrderBackup2013
FROM Customers
LEFT JOIN Orders
ON Customers.CustomerID=Orders.CustomerID;
Copier les données de plus d'une table

INSERT INTO SELECT

Le INSERT INTO SELECT copie les données d'une table et l'insère dans une table existante.
INSERT INTO *table2*
SELECT * **FROM** *table1;*

INSERT INTO Customers (CustomerName, Country)
SELECT SupplierName, Country **FROM** Suppliers
WHERE Country='Germany';

CONSTRAINT

- **NOT NULL** - Indique qu'une colonne ne peut pas stocker la valeur NULL.
- **UNIQUE** - Veille à ce que chaque rangée pour une colonne doit avoir une valeur unique.
- **PRIMARY KEY** - Une combinaison d'un NOT NULL et UNIQUE. Assure qu'une colonne (ou une combinaison de deux ou plusieurs colonnes) a une identité unique qui aide à trouver un enregistrement particulier dans un tableau plus facilement et rapidement.
- **FOREIGN KEY** - Veiller à l'intégrité référentielle des données dans une table pour correspondre à des valeurs d'une autre table.
- **CHECK** - Veiller à ce que la valeur dans une colonne répond à une condition spécifique.
- **DEFAULT** - Définit une valeur par défaut lorsque spécifié aucun pour cette colonne.

CREATE TABLE Persons
(
P_Id int **NOT NULL**,
LastName varchar(255) **NOT NULL**,
FirstName varchar(255),
Address varchar(255),
City varchar(255),
CONSTRAINT uc_PersonID **UNIQUE**
(P_Id,LastName)
)
Créer une table avec contrainte

ALTER TABLE Persons
ADD CONSTRAINT uc_PersonID **UNIQUE**
(P_Id,LastName)
Ajouter une contrainte

ALTER TABLE Persons
DROP INDEX uc_PersonID
Supprimer une contrainte

INDEX

Un index peut être créé dans une table pour trouver des données plus rapidement et efficacement.

CREATE INDEX index_name
ON table_name (column_name)

CREATE UNIQUE INDEX index_name
ON table_name (column_name)

EX : **CREATE INDEX** PIndex
ON Persons (LastName)
EX : **CREATE INDEX** PIndex
ON Persons (LastName, FirstName)

ALTER TABLE table_name
DROP INDEX index_name

VIEW

Dans SQL, une vue est une table virtuelle basée sur le résultat - set d'une instruction SQL .

Une vue contient des lignes et des colonnes, tout comme une vraie table. Les champs de vue sont des champs d'une ou plusieurs tables dans la base de données réelles.

CREATE VIEW view_name **AS**
SELECT column_name(s)
FROM table_name
WHERE condition

CREATE OR REPLACE VIEW view_name **AS**
SELECT column_name(s)
FROM table_name
WHERE condition

EX : **CREATE VIEW** [Current Product List] **AS**
SELECT ProductID,ProductName
FROM Products
WHERE Discontinued=No

EX : **CREATE OR REPLACE VIEW** [Current Product List] **AS**
SELECT ProductID,ProductName,Category
FROM Products
WHERE Discontinued=No

DROP VIEW view_name

MYSQL DATE FUNCTIONS

NOW()	Returns the current date and time
CURDATE()	Returns the current date
CURTIME()	Returns the current time
DATE()	Extracts the date part of a date or date/time expression
EXTRACT()	Returns a single part of a date/time
DATE_ADD()	Adds a specified time interval to a date
DATE_SUB()	Subtracts a specified time interval from a date
DATEDIFF()	Returns the number of days between two dates
DATE_FORMAT()	Displays date/time data in different formats

SQL Date Data Types:

- **DATE**
  - format YYYY-MM-DD
- **DATETIME**
  - format: YYYY-MM-DD HH:MI:SS
- **TIMESTAMP**
  - format: YYYY-MM-DD HH:MI:SS
- **YEAR**
  - format YYYY or YY
- **TIME**
  - format HH :MM :SS