0. Introduction SQL

SQL (sigle de *Structured Query Language*, en français **langage de requête structurée**) est un langage informatique normalisé servant à exploiter des bases de données relationnelles. La partie langage de manipulation des données de SQL permet de rechercher, d'ajouter, de modifier ou de supprimer des données dans les bases de données relationnelles.

En juin 1970, Edgar Frank Codd publia l'article *A Relational Model of Data for Large Shared Data Banks* (« Un référentiel de données relationnel pour de grandes banques de données partagées ») dans la revue *Communications of the ACM* (*Association for Computing Machinery*). Ce référentiel relationnel fondé sur la logique des prédicats du premier ordre a été rapidement reconnu comme un modèle théorique intéressant, pour l'interrogation des bases de données, et a inspiré le développement du langage *Structured English QUEry Language* (*SEQUEL*) (« langage d'interrogation structuré en anglais »), renommé ultérieurement SQL pour cause de conflit de marque déposée.

Développée chez IBM en 1970 par Donald Chamberlin et Raymond Boyce, cette première version a été conçue pour manipuler et éditer des données stockées dans la base de données relationnelle à l'aide du système de gestion de base de données IBM System R. Le nom SEQUEL, qui était déposé commercialement par l'avionneur Hawker Siddeley pour un système d'acquisition de données, a été abandonné et contracté en SQL en 1975[1]. SQL était censé alors devenir un élément clé du futur projet FS.

En 1979, *Relational Software, Inc.* (actuellement *Oracle Corporation*) présenta la première version commercialement disponible de SQL, rapidement imité par d'autres fournisseurs.

SQL a été adopté comme recommandation par l'Institut de normalisation américaine (ANSI) en 1986, puis comme norme internationale par l'ISO en 1987 sous le nom de *ISO/CEI 9075 - Technologies de l'information - Langages de base de données - SQL*.

1. Chaque SGDB possède son SQL « maison » on utilisera les plus courants :

SQL « standard » : https://sql.sh/

SQLite : https://www.sqlitetutorial.net/

MySQL ORACLE : https://www.mysqltutorial.org/

2. Fonctions de bases

- Création d'une table « matable »

SQL	SQL Lite	MySQL
	CREATE TABLE matable (attribut1 INTEGER, attribut2 TEXT, attribut3 BOOLEAN, attribut4 BLOB, attribut5 FLOAT);	CREATE TABLE `commande` (`attribut1` INT, `attibut2` DATE, `attribut3` VARCHAR(20), `attribut4` TEXT);

- La norme SQL impose les types de données suivants :

CHAR(n) : limités aux caractères de l'ASCII sur 8 octets n indiquant la taille de la zone qui est fixe et sera complétée par des "blancs".

VARCHAR(n) : limités aux caractères de l'ASCII sur 1 octet n indiquant la taille maximale de zone qui est variable.

Le type TEXT n'existe pas dans la norme SQL et reste spécifique à certains SGBDR (voir leurs doc respectives)

- Clés primaires et étrangères

Les clé primaires et étrangères sont définies lors de la création des tables.

La clé primaire est un attribut ou un ensemble d'attributs qui identifie de manière unique chaque ligne de la table. La clé primaire suit ces règles :

- Une clé primaire doit contenir des valeurs uniques. Si la clé primaire se compose de plusieurs attributs, la combinaison de valeurs dans ces colonnes doit être unique.
- Une colonne de clé primaire ne peut pas avoir de valeurs NULL. Toute tentative d'insertion ou de mise à jour de NULL dans les colonnes de clé primaire entraînera une erreur.
- Une table ne peut avoir une ou une seule clé primaire.

SQL	A la création de la table Autre syntaxe

SQL Lite	CREATE TABLE countries (
	  country_id INTEGER PRIMARY KEY,
	  name TEXT NOT NULL
	);
		CREATE TABLE languages (
		  language_id INTEGER,
		  name TEXT NOT NULL,
		  PRIMARY KEY (language_id)
		);

MySQL	CREATE TABLE users(
	  user_id INT AUTO_INCREMENT PRIMARY KEY,
	  username VARCHAR(40),
	  password VARCHAR(255),
	  email VARCHAR(255)
	);
		CREATE TABLE roles(
		  role_id INT AUTO_INCREMENT,
		  role_name VARCHAR(50),
		  PRIMARY KEY(role_id)
		);

Clés étrangères

SQL	
SQL Lite	CREATE TABLE track(trackid INTEGER, trackname TEXT, trackartist INTEGER, FOREIGN KEY(trackartist) REFERENCES artist(artistid));

MySQL	CREATE TABLE Orders (
	  OrderID int NOT NULL,
	  OrderNumber int NOT NULL,
	  PersonID int,
	  PRIMARY KEY (OrderID),
	  FOREIGN KEY (PersonID) REFERENCES Persons(PersonID)
	);

- Suppression d'une table « matable »

- Insertion d'une ou plusieurs valeur dans une table « matable »

SQL	SQL Lite	MYSQL
	INSERT INTO matable VALUES (attribut1,attribut2,attribut3), (4 , "Jean", 50);	Idem Guillemets simple ou double

- Modification d'une ou plusieurs valeurs d'attributs :

SQL	SQL Lite	MYSQL
	UPDATE employees SET lastname = 'Smith' WHERE employeeid = 3;	UPDATE employees SET lastname = 'Hill', email = 'mary.hill@classicmodelcars.com' WHERE employeeNumber = 1056;

- Suppression d'un enregistrement dans une table

SQL	SQL Lite	MYSQL
	DELETE FROM artists_backup WHERE artistid = 1; DELETE FROM artists_backup WHERE name LIKE '%Santana%';	DELETE FROM employees WHERE officeCode = 4;

- Sélection de l'ensemble d'une table

Le SELECT est la commande de base du SQL destinée à **extraire des données** d'une base ou **calculer de nouvelles données à partir d'existantes**...

Voici la syntaxe générale d'une commande SELECT :

SELECT [DISTINCT ou ALL] * ou liste de colonnes

FROM nom de table

[WHERE prédicats]

[GROUP BY ordre des groupes]

[HAVING condition]

[ORDER BY] liste de colonnes

- Sélection de toute une table

	SELECT * FROM employees;

- Sélection d'un ou plusieurs attributs dans une table

SQL	SQL Lite	MYSQL
	SELECT trackid, name, composer, unitprice FROM tracks;	SELECT lastname, firstname, jobtitle FROM employees;

- WHERE et LIKE et =

= et LIKE permettent d'écrire la condition de sélection mais il différent dans leur usage.

= aura une égalité stricte avec l'opérande de droite

LIKE permet d'utiliser :

- % - Représente zéro, un ou plusieurs caractères

- _ - Représente un seul caractère (MS Access utilise un point d'interrogation (?) À la place)

SELECT * FROM Customers WHERE CustomerName LIKE 'a%';	SELECT * FROM Customers WHERE CustomerName LIKE '%a';	SELECT * FROM Customers WHERE CustomerName LIKE 'a_f_%';
CustomerName Alfreds Futterkiste Ana Trujillo Emparedados y helados Antonio Moreno Taquería Around the Horn	**CustomerName** Antonio Moreno Taquería Centro comercial Moctezuma Godos Cocina Típica Que Delícia	**CustomerName** Alfreds Futterkiste

- Opérateurs disponibles sur le prédicat WHERE ……

opérateurs de comparaisons	= <> < <= > >=
connecteurs logiques	{OR | AND}
opérateur de négation	NOT
parenthèses	(...)
opérateurs mathématiques	+ - * /
comparaison logique	IS [NOT] {TRUE | FALSE | UNKNOWN}
comparaison avec valeur	IS [NOT] NULL
intervalle	valeur BETWEEN borne_basse AND borne_haute
comparaison partielle de chaîne de caractères	valeur LIKE motif [ESCAPE echappement]
comparaison à une liste de valeur	valeur [NOT] IN (liste)

- Suppression des doublons dans le résultat de la requête

SQL	MYSQL
	SELECT DISTINCT state, city FROM customers WHERE state IS NOT NULL ORDER BY state , city;

3. Opération d'agrégation

- Ajouter

L' opérateur « **SUM** » permet d'ajouter toutes les entrées d'un attribut/colonne.

SQL	SQLite	MySQL

	`SELECT SUM(milliseconds) FROM tracks;`
	`----------`
	SELECT
	AlbumId,
	SUM(milliseconds)
	FROM
	tracks
	GROUP BY
	AlbumId;
		SELECT
		SUM(quantityOrdered * priceEach) orderTotal
		FROM
		orderdetails
		WHERE
		orderNumber = 10100;

orderNumber	quantityOrdered	priceEach
10100	30	136.00
10100	50	55.09
10100	22	75.46
10100	49	35.29

AS (alias)

Dans le langage SQL il est possible d'utiliser des **alias** pour renommer temporairement une colonne ou une table dans une requête. Cette astuce est particulièrement utile pour faciliter la lecture des requêtes.

- renommer le nom d'une colonne dans les résultats d'une requête SQL. C'est pratique pour avoir un nom facilement identifiable dans une application qui doit ensuite exploiter les résultats d'une recherche.
- attribuer un autre nom à une table dans une requête SQL. Cela peut aider à avoir des noms plus court, plus simple et plus facilement compréhensible.

- Compter

Il peut être utile de compter le nombre d'entrées dans une colonne/attribut.

L'opérateur « **COUNT()** » est utilisé à cette fin.

SQL	SQLite exemples	MySQL idem pour les plus simples

	`SELECT COUNT(*) FROM t1;`
	
	`SELECT COUNT(DISTINCT c) FROM t1;`
	
	`SELECT COUNT(*) FROM tracks WHERE albumid = 10;`
		`SELECT productLine, COUNT(*) FROM products GROUP BY productLine;`

- Effectuer une moyenne

L'opérateur « AVG() » est utilisé à cette fin.

SQL	SQLite exemples	MySQL idem pour les plus simples
	`SELECT avg(DISTINCT val) FROM avg_tests; SELECT ---------- SELECT avg(val) FROM avg_tests WHERE rowid < 5;`	`SELECT AVG(buyprice) 'Average Price' FROM products;`

- Trouver les Minimum et Maximum

Les opérateurs **MIN** et **MAX** permettent de le faire.

SQL	SQLite exemples	MySQL idem pour les plus simples
	`SELECT AlbumId, MAX(bytes) FROM tracks GROUP BY AlbumId;`	`SELECT MIN(buyPrice) FROM products WHERE productline = 'Motorcycles';`