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/

: https://www.sqlitetutorial.net/

: https://www.mysqltutorial.org/

2. Fonctions de bases

- Création d'une base de données

SQL	Exemple base de données cours_db
CREATE DATABASE mabase ;	

- Création d'une table « matable »

SQL	Exemple base de données clients
CREATE TABLE matable **(** attribut1 type1, attribut2 type2, **) ;**	

- La norme SQL impose les types de données suivants :

Les types chaîne de caractères

Type	Taille en octet
CHAR	1
CHARACTER(X)	
CHAR(X)	(Chaîne limitée à 255c)
CHAR(X)	4+X
VARCHAR(X)	variable d'un enregistrement à l'autre (chaîne limitée à 255c)
VARCHAR(X)	<4+X
TEXT	4+taille de la chaîne (Sans limite)

Les types numériques

Type	Min	Max	Taille en octet
INT INTEGER	-2147483648	2147483647	4
FLOAT	-3.402823466E+38	3.402823466E+38	4

- Suppression d'une table « matable »

DROP TABLE matable **;**	

- 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.

SQL	A la création de la table **CREATE TABLE** nom_de_la_table **(** id **INT PRIMARY KEY NOT NULL AUTO_INCREMENT,** **[...]** **);** Autre syntaxe **CREATE TABLE** nom_de_la_table **(** id **INT NOT NULL AUTO_INCREMENT,** **[...],** **PRIMARY KEY** (id) **);**
Exemple	

Clés étrangères

SQL	**CREATE TABLE** Orders **(** OrderID **INT NOT NULL PRIMARY KEY,** OrderNumber **INT NOT NULL,** PersonID **INT ,** **FOREIGN KEY** (PersonID **)** **REFERENCES** Persons(PersonID) **);**
Exemple	

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

SQL	Insérer les valeurs des différents enregistrements.
INSERT INTO matable **VALUES** (attribut1,attribut2,attribut3), (attribut1,attribut2,attribut3), **……** **;**	

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

SQL	Mettez à jour l'émail de francois : blaise.francois@gmail.com
UPDATE matable **SET** attribut1 = 'valeur1', attribut2 = 'valeur2' **WHERE** Condition ;	

- Suppression d'un enregistrement dans une table

SQL	Supprimer l'enregistrement de francois	Expression régulière
DELETE FROM matable **WHERE** condition		DELETE FROM artists_backup WHERE name LIKE '%Santana%';

- Modification de la structure de la table

Modifications de la table client en ajoutant une colonne nb_mois et une colonne prix

id_client	nom	prenom	email	type	nb_mois	prix
1	dupond	jean	j.dupond@gmail.com	1	5	50
2	picot	alice	a.picot@laposte.net	3	5	55
3	renan	alix	a.renan@gmail.com	2	10	50
5	francois	blaise	blaise.francois@gmail.com	2	10	50

SQL	
ALTER TABLE matable **ADD** nom_colonne type_donnees - **DROP COLUMN** nom_colonne - **MODIFY** nom_colonne type_donnees - **CHANGE** colonne_ancien_nom colonne_nouveau_nom type_donnees - **ALTER TABLE** table2 **ADD FOREIGN KEY** (id_com) **REFERENCES table1**(id_com) ;	

- 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** matable ; éventuellement **ORDER BY** attribut ; peut être très long expose tous les éléments de la table	

- Sélection d'un ou plusieurs attributs dans une table

SQL	Sélectionner les noms et prénoms de tous les clients.
SELECT attribut1, attribut2 **FROM** matable	

- 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)

SQL	
SELECT attribut **FROM** matable **WHERE** attribut **LIKE** quelquechose ;	

Expressions régulières

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	Sélectionner les noms de plus de 5 caractères
SELECT DISTINCT attribut1,attibut2 …. **FROM** matable **WHERE** condition	

3. Opération d'agrégation

- Ajouter

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

SQL	Faites la somme des abonnements des clients dont la durée excède 5 mois
SELECT SUM (montant) **AS** somme **FROM** matable ; **En option** **GROUP BY** attibut**;**	

ou **WHERE** condition **;**	

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 courts, plus simples et plus facilement compréhensibles.

- Compter

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

L'opérateur « **COUNT()** » est utilisé à cette fin. (on utilise **DISTINCT** pour éviter les doublons)

SQL	Compter le nombre de clients différents
SELECT COUNT (* ou attribut) **AS** nb **FROM** matable **;**	

- Effectuer une moyenne

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

SQL	Trouver la valeur moyenne des totaux de paiement pour un client et l'appeler paiement_moyen
SELECT AVG (attribut) **AS moy** **FROM** matable **;**	

- Trouver les Minimum et Maximum

Les opérateurs **MIN** et **MAX** permettent de le faire.SQL	Sélectionner la valeur le plus faible abonnement

`SELECT MIN(nom_colonne)`
`FROM table`
`SELECT MAX(nom_colonne)`
`FROM table`	