Bases de données

Découverte SQL

Introduction

Différents SGBDR

- SQLite : open source. n'utilise pas de serveur. Stocke la base dans un unique fichier. Très rapide pour des bases "modestes" (< 1 million d'enregistrements)
- MySQL, PostgreSQL, Oracle etc. : (Oracle n'est pas open source). Utilisent un serveur. Très rapides pour des bases "conséquentes".

La majorité de ces logiciels sont écrits en C ou en C++.

- Certains sont spécialisés (graphiques, données numériques précises etc)
- D'autres sont fournis avec un serveur dans le cloud etc.

Tous implémentent les fonctions de base de SQL + quelques fonctions "perso" parfois commodes.

SQLite

Nous utiliserons SQLite qui est programmé en C et qu'on peut utiliser

- en Python (import sqlite3),
- en mode fenêtré avec DB Browser,
- ainsi qu'en ligne (https://inloop.github.io/sqlite-viewer/).
- dans l'interpréteur sqlite :
  - `CREATE TABLE department(id INTEGER, name STRING);`
- mais aussi
  - iOS,
  - Android etc.

À noter, toutes les applications mobiles qui enregistrent *localement* beaucoup de données utilisent SQLite.

Ce n'est pas le seul moyen d'enregistrer quelque chose sur le téléphone : fichiers, "local storage = dictionnaire" etc.

Mise en place du TP

Aller chercher le fichier **decouverte.sql** dans la zone d'échange, **en faire en une copie sur votre zone locale** et ‹ l ›ouvrir avec le logiciel **SQLite Studio**.

→ Si jamais vous modifiez ou supprimez la table, il suffit de la réouvrir avec le logiciel pour la réinitialiser.

1. Extraire des données d'une table avec SELECT

On peut sélectionner toutes les lignes d'une table avec

```
SELECT noms_colonnes_separes_par_virgules
FROM nom_table;
```

Sélectionner des colonnes

A votre tour !

Maintenant à votre tour: effectuez dans SQLite Studio la commande suivante:

```
SELECT name, salary FROM employees ;
```

Exercice 1

Affichez les noms, rôles et commissions des employés.

.....

.....

Sélectionner toute une table

On peut aussi utiliser `*` pour sélectionner toutes les colonnes.

A votre tour !

```
SELECT * FROM employees ;
```

Exercice 2

Ajoutez `LIMIT 5` avant la fin (`;`) de la requête. Qu'est-ce que cela change ?

.....

.....

Avec et sans `DISTINCT`

`DISTINCT` pour sélectionner une seule occurrence de chaque valeur de la colonne en question.

```
SELECT DISTINCT nom_colonne
FROM nom_table;
```

A votre tour !

```
SELECT DISTINCT manager FROM employees;
```

Exercice 3

Qu'obtient-on sans le mot clé DISTINCT ?

.....

Filtrer avec `WHERE`

Sélectionne uniquement les lignes qui respectent la clause du WHERE.

```
SELECT noms_colonnes_separes_par_virgules
FROM nom_table
WHERE nom_colonne op_comp valeur op_bool nom_colonne op_comp valeur;
```

A votre tour !

```
SELECT name, salary, commission FROM employees
WHERE salary > 50000;
```

Exercice 4

Affichez les noms et salaires des employés n'ayant pas reçu de commission.

On peut vérifier qu'un champ n'est pas renseigné avec `IS NULL`.

.....

.....

.....

La clause porte sur les valeurs des colonnes

- Utilisation d'opérateurs de comparaison (`op_comp`) : `=`, `<>`, `!=`, `>`, `>=`, `<`, `<=`
- Utilisation d'opérateurs booléens (`op_bool`) : `AND`, `OR`
  - `AND` : combinaisons de conditions sur des colonnes différentes
  - `OR` : plusieurs valeurs possibles pour une même colonne

A votre tour !

```
SELECT name, salary, commission FROM employees
WHERE salary > 20000 AND commission > 0;
```

```
SELECT name, salary, commission FROM employees
WHERE salary > 20000 OR commission > 0;
```

Exercice 5

Affichez toutes les infos des employés gagnant moins de 40000 ou du département 4.

.....

.....

.....

.....

Renommage de colonne avec `AS`

Change l'affichage et le nommage des données

```
SELECT abrev.nom_colonne AS nom_affiche
FROM nom_table AS abrev
ORDER BY nom_colonne [DESC];
```

- Associé à un nom de colonne : change le nom affiché de la colonne dans le résultat.
- Associé à un nom de table : permet d'abrévier le nom de la table pour préciser de quelle table provient une colonne dont le nom est utilisé par plusieurs tables. Cette abréviation **doit** être utilisée dans le reste de la requête.

A votre tour !

```
SELECT name, manager FROM employees;
SELECT name AS superman, manager AS superpower FROM employees;
```

2. Fonctions de calcul sur les données extraites.

Appliquer une fonction sur les valeurs d'une colonne avec `ORDER BY`

- Trie les données selon la colonne précisée.
- Par défaut, le tri est dans l'ordre croissant, DESC permet d'obtenir l'ordre décroissant.

```
SELECT FONCTION(nom_colonne)
FROM nom_table;
ORDER BY nom_colonne
```

A votre tour !

```
SELECT name, manager FROM employees
ORDER BY salary ASC;
```

Exercice 6

Affichez les employés par département décroissant.

.....

.....

.....

- `COUNT` : compte le nombre de lignes sélectionnées.
- `MIN, MAX` : renvoie la valeur minimum ou maximum de la colonne, parmi les lignes sélectionnées
- `SUM, AVG` : calcule la somme ou la moyenne des valeurs numériques de la colonne, parmi les lignes sélectionnées

A votre tour !

```
SELECT AVG(salary) FROM employees;
```

Exercice 7

Affichez la masse salariale (total du salaire de tout les employés) de l'entreprise avec un nom de colonne pertinent.

.....

.....

.....

Agréger avec `GROUP BY` (Hors programme)

`GROUP BY` : agrège ensemble les valeurs identiques d'une colonne pour appliquer une fonction à chacun des sous-ensembles.

A votre tour !

```
SELECT AVG(salary) FROM employees;
```

Et après

```
SELECT manager, AVG(salary) AS salary FROM employees
GROUP BY manager;
```

Exercice 8

Affichez les départements et leur salaire moyen. On utilisera un nom de colonne pertinent.

.....

.....

.....

3. Modifitcation de données

Ajouter avec `INSERT`

`INSERT` : ajoute une nouvelle ligne de données dans une table

```
INSERT INTO nom_table VALUES (liste_valeurs_dans_ordre_colonnes_table);

INSERT INTO nom_table (liste_nom_colonnes_a_remplir)
VALUES (liste_des_valeurs_a_inserer_dans_ordre_liste_colonnes);
```

A votre tour !

```
SELECT * FROM department;
```

Et après :

```
INTO department VALUES (5,"Comms");
SELECT * FROM department;
```

Exercice 9

Insérer un nouvel employé

id	nom	désignation	encadre	recruté le	salaire	département
8	Turing	Tech	6	2022/05/04	33000	3

.....

.....

Depuis une base vide, les données sont insérées de la même manière :

A votre tour !

```
DROP TABLE IF EXISTS department;

CREATE TABLE department(
id integer,
name text,
PRIMARY KEY("id")
);

INSERT INTO department VALUES (1,"Sales");
INSERT INTO department VALUES (2,"Admin");
INSERT INTO department VALUES (3,"IT");
INSERT INTO department VALUES (4,"Foreign");

SELECT * FROM department;
```

Mettre à jour avec `UPDATE`

`UPDATE` : met à jour la ou les lignes qui respectent la clause du WHERE

```
UPDATE nom_table SET nom_colonne1=valeur1, nom_colonne2=valeur2
WHERE nom_colonne op_comp valeur op_bool nom_colonne op_comp valeur;
```

A votre tour !

```
SELECT * FROM department;
UPDATE department SET name="Foreign Affair" WHERE name="Foreign";
SELECT * FROM department;
```

Exercice 10

Doubler le salaire de tous les employés gagnant moins de 30000$.

.....

.....

.....

.....

Effacer avec `DELETE`

`DELETE` : efface la ou les lignes d'une table qui respectent la clause du WHERE

```
FROM nom_table
WHERE nom_colonne op_comp valeur op_bool nom_colonne op_comp valeur;
```

A votre tour !

```
SELECT * FROM department;
DELETE FROM department WHERE name="Foreign";
SELECT * FROM department;
```

Exercice 11

Effacer tous les employés du département 3 OU gagnant plus de 40000$.

.....

.....

.....

.....

4. Extraction des données de deux tables

Lorsqu'on travaille avec plusieurs table en relation, il faut pouvoir les combiner.

Deux approches différentes :

1. produit cartésien,
2. utiliser les clés étrangères.

Le produit cartésien de deux ensembles est l'ensemble des couples d'éléments de ces ensembles.

P = {1, 2, 3}, Q = {A, B}

P x Q = {(1, A), (1, B), (2, A), (2, B), (3, A), (3, B)}

Produit cartésien

Comme son nom l'indique, génère de façon exhaustive toutes les associations possibles entre les lignes des deux tables

Si on croise une table avec 500 villes avec une contenant 100 000 habitants, le nombre total de la table créée par produit cartésiens sera de 500 * 100 000 = 50 000 000 lignes.

→ Cette approche est à éviter autant que possible.

A votre tour !

```
SELECT * FROM department;
```

Et après

```
SELECT * FROM department, employees;
```

Exercice 12

Affichez le nombre de lignes des tables department, employees et de leur produit cartésien.

On utilisera plusieurs requêtes.

.....

.....

.....

.....

.....

.....

Mettre en relation avec `JOIN ON`

Génère uniquement les associations entre les lignes qui sont liées par des clés primaires et étrangères identiques.

Permet d'associer deux tables qui sont en relation.

→ C'est la notion la plus importante et la plus délicate du chapitre.

A votre tour !

```
SELECT * FROM department
JOIN employees ON employees.dept=department.id;
```

Pour sélectionner des colonnes il convient d'utiliser le noms complets :

A votre tour !

```
SELECT employees.name, department.name FROM department
JOIN employees ON employees.dept=department.id;
```

Problème, deux colonnes issues de tables différentes peuvent porter le même nom. On utilise alors des alias :

A votre tour !

```
SELECT employees.name AS name, department.name AS dept FROM department
JOIN employees ON employees.dept=department.id;
```

Avec un alias sur les tables on écrit la requête plus facilement :

A votre tour !

```
SELECT e.name AS name, d.name AS dept FROM department AS d
JOIN employees AS e ON e.dept=d.id
ORDER BY d.id;
```

Exercice 13

1. Affichez les noms, salaires et département des employés, triés par date d'embauche.
2. Cette fois limiter au 3 premiers résultats.
3. Ne conserver que ceux qui ont touché une commission. La clause WHERE est entre JOIN ... et ORDER BY ...

.....

.....

.....

.....

.....

.....

.....