

1 Notions exigibles

- relation / table
↪ Tableau de données. Chaque ligne représente un objet. Les objets sont de même nature.
- attribut
↪ Colonne d'une table
- domaine
↪ Ensemble des valeurs que peuvent prendre les attributs
- clef primaire (PRIMARY KEY)
↪ Attribut (ou groupe d'attributs) permettant d'identifier chaque ligne de manière unique
- clef étrangère (FOREIGN KEY)
↪ Attribut (ou groupe d'attributs) étant une clef primaire pour une autre table de la base de données
- schéma relationnel
↪ Mise en relation de plusieurs tables en une base de données

2 Commandes exigibles

- CREATE TABLE nom_de_table
↪ Création d'une table
- SELECT nom_de_champ FROM nom_de_table
↪ Sélection / extraction de données d'une table
- WHERE
↪ Permet de spécifier une condition à respecter
- INSERT INTO nom_de_table
↪ Insertion de données dans une table
- DELETE FROM nom_de_table
↪ Suppression de données d'une table
- UPDATE nom_de_table
↪ Mise à jour de données d'une table
- SET
↪ Permet de mettre à jour un attribut précis lors d'une mise à jour de la table
- SELECT nom_de_champ FROM nom_de_table_1 INNER JOIN nom_de_table_2
ON cle_primaire = cle_etrangere
↪ Réalisation d'une jointure entre deux tables.
- AND, OR et NO
↪ Opérateurs logiques permettant de construire des conditions complexes à partir de plusieurs conditions
- =, <>, <, <=, >, >=
↪ Opérateurs de comparaison
- +, -, *, /
↪ Opérateurs arithmétiques

3 Commandes non exigibles

- Opérateurs ensemblistes : UNION, INTERSECTION, EXCEPT
- Opérateurs spécifiques de l'algèbre relationnelle : projection, renommage (AS), produit cartésien
- Fonctions d'agrégation : MIN, MAX, SUM, AVG (moyenne), COUNT
- DISTINCT, ORDER BY, GROUP BY

4 Exemples d'utilisation des commandes et fonctions SQL

La commande CREATE TABLE. La commande CREATE TABLE permet de créer une table.

Exemple : On souhaite créer deux tables `client` et `commande` où chaque commande est associée à un client.

La requête

```
1 CREATE TABLE client (  
2   id_client INTEGER PRIMARY KEY,  
3   nom TEXT,  
4   prénom TEXT  
5 );
```

permet de créer la table `client` contenant un numéro d'identification unique pour chaque client ainsi que son nom et son prénom. L'attribut `id_client` est la clé primaire de cette table.

La requête

```
1 CREATE TABLE commande (  
2   id_commande INTEGER PRIMARY KEY,  
3   date_commande DATE,  
4   id_client INTEGER,  
5   FOREIGN KEY (id_client) REFERENCES client(id_client)  
6 );
```

permet de créer la table `commande` contenant un numéro d'identification unique pour chaque commande ainsi que la date à laquelle la commande a été effectuée et le numéro d'identification du client ayant effectué la commande. L'attribut `id_commande` est la clé primaire de cette table. L'attribut `id_client` est une clé étrangère pointant vers la clé primaire de la table `client`.

Les commandes UPDATE et SET. Les commandes UPDATE et SET permettent de mettre à jour les données d'une table.

Exemple : On souhaite modifier le nom du client numéro 1 dans l'exemple précédent.

La requête

```
1 UPDATE client  
2 SET nom = 'Martin'  
3 WHERE id_client = 1;
```

remplace son ancien nom par 'Martin'.

Il est possible de modifier plusieurs colonnes simultanément. La requête

```
1 UPDATE client  
2 SET nom = 'Martin',  
3     prénom = 'Paul'  
4 WHERE id_client = 1;
```

remplace son ancien nom par 'Martin' et son ancien prénom par 'Paul'.

La commande SELECT DISTINCT. La commande SELECT DISTINCT, utilisée à la place de la commande SELECT, permet de ne retenir qu'une occurrence de chaque valeur dans une colonne donnée, même si cette colonne comporte des valeurs qui se répètent.

Exemple : Une table `etablissement` contient les données suivantes concernant plusieurs établissements scolaires et la ville où ils se situent.

id	nom	ville
1	Edouart Herriot	Livry-Gargan
2	Diderot	Lyon
3	Edouard Herriot	Lyon
4	Louise Michel	Champigny-sur-Marne
5	Diderot	Paris

La requête

```
1 SELECT DISTINCT nom
2 FROM etablissement;
```

permet d'obtenir les différents noms d'établissements, sans répétition :

Edouard Herriot

Diderot

Louise Michel

La commande SELECT DISTINCT peut éventuellement être combinée avec la commande WHERE pour filtrer une partie des entrées.

La commande ORDER BY. La commande ORDER BY, placée en fin de requête et suivie d'un nom de colonne, permet de trier les résultats demandés dans l'ordre croissant des valeurs de la colonne spécifiée, ou dans l'ordre alphabétique s'il s'agit de données de type TEXT.

On peut ajouter le mot-clé DESC à la fin de la commande pour trier les données dans l'ordre décroissant.

Exemple : Une table `etablissement` contient les données suivantes concernant plusieurs établissements scolaires, leur nombre d'élèves et la ville où ils se situent.

identifiant	nom	nombre_eleves	ville
1	Edouart Herriot	887	Livry-Gargan
2	Diderot	220	Lyon
3	Edouard Herriot	808	Lyon
4	Louise Michel	653	Champigny-sur-Marne
5	Diderot	1200	Paris

- Pour afficher les données en triant les noms d'établissement dans l'ordre alphabétique, on peut utiliser la requête suivante :

```
1 SELECT * FROM etablissement
2 ORDER BY nom;
```

- Pour afficher les noms des établissements dans l'ordre décroissant du nombre d'élèves, on peut utiliser la requête suivante :

```
1 SELECT nom FROM etablissement
2 ORDER BY nom_eleves DESC;
```

La fonction COUNT(). La fonction d'agrégation COUNT() permet de connaître le nombre d'enregistrements d'une table, vérifiant éventuellement une certaine condition.

Nous donnons ci-dessous deux exemples d'utilisation de la fonction COUNT(), en considérant une table nommée `ma_table` comportant deux colonnes `colonne_1` et `colonne_2`.

- La requête suivante renvoie le nombre total d'enregistrements dans `ma_table` :

```
1 SELECT COUNT(*) FROM ma_table;
```

- La requête suivante renvoie le nombre d'enregistrements de `ma_table` vérifiant la condition `cond` :

```
1 SELECT COUNT(*) FROM ma_table
2 WHERE cond;
```

La fonction AVG(). La fonction d'agrégation AVG() permet de calculer la moyenne des valeurs d'une colonne dans une table.

Par exemple, si on considère la table nommée `table` contenant les enregistrements suivants :

colonne_1	colonne_2	colonne_3	colonne_4
1	69	Lyon	4
2	31	Toulouse	8
3	54	Nancy	5
4	64	Saint-Jean-de-Luz	17
5	44	Nantes	6

alors la requête suivante

```
1 SELECT AVG(colonne_4)
2 FROM table
3 WHERE colonne_1 <= 3;
```

affiche la moyenne des valeurs de `colonne_4` des trois premiers enregistrements : 5.6667 c'est-à-dire $\frac{4 + 8 + 5}{3}$.

La commande GROUP BY. La commande GROUP BY permet de regrouper tous les enregistrements dont la valeur d'un attribut donné est identique, en appliquant une fonction à chaque groupe d'enregistrements.

Exemple : Une table `etablissement` contient les données suivantes concernant plusieurs établissements scolaires, leur nombre d'élèves et la ville où ils se situent.

identifiant	nom	nombre_eleves	ville
1	Pierre-Gilles de Gennes	598	Paris
2	Diderot	220	Lyon
3	Edouard Herriot	808	Lyon
4	Paul Valéry	525	Paris
5	Edouard Herriot	887	Livry-Gargan
6	Diderot	1200	Paris

On peut calculer le nombre d'établissements dans chaque ville à l'aide de la requête suivante :

```
1 SELECT ville, COUNT(*)
2 FROM etablissement
3 GROUP BY ville;
```

On obtient alors le résultat suivant :

ville	COUNT(*)
Livry-Gargan	1
Lyon	2
Paris	3

On peut de plus renommer les colonnes de cette nouvelle table :

```

1 SELECT ville, COUNT(*) AS nombre_etab
2 FROM etablissement
3 GROUP BY ville;
```

On obtient alors le résultat suivant :

ville	nombre_etab
Livry-Gargan	1
Lyon	2
Paris	3

La commande INNER JOIN. La commande INNER JOIN permet de réaliser la jointure de deux tables. Cette commande retourne les enregistrements lorsqu'il y a au moins une ligne dans chaque table qui correspond à la condition.

Exemple : Une table **etablissement** contient les données suivantes concernant plusieurs établissements scolaires, leur nombre d'élèves et la ville où ils se situent.

ident	nom	ville
1	Pierre-Gilles de Gennes	Paris
2	Diderot	Lyon
3	Edouard Herriot	Lyon
4	Paul Valéry	Paris
5	Edouard Herriot	Livry-Gargan
6	Diderot	Paris

Une seconde table **effectifs** contient pour plusieurs établissements scolaires leur identifiant et leur nombre d'élèves.

ident	eleves
1	598
2	220
3	808
4	525
5	887
6	1200

On peut alors créer une table contenant pour chaque établissement son nom et son nombre d'élèves :

```

1 SELECT nom,eleves
2 FROM etablissement INNER JOIN effectifs
3 ON etablissement.ident = effectifs.ident;
```

On obtient alors le résultat suivant :

nom	eleves
Pierre-Gilles de Gennes	598
Diderot	220
Edouard Herriot	808
Paul Valéry	525
Edouard Herriot	887
Diderot	1200