

## 1) créer une database

```
mysql> create database test;
Query OK, 1 row affected (0,01 sec)

mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| sys |
| test |
+-----+
5 rows in set (0,01 sec)
```

Figure 1: creation database

## 2) créer des tables

```
mysql> use test;
Database changed
mysql> create table employe(nom varchar(30),
-> prenom varchar(30),
-> salaire int NOT NULL,
-> adresse varchar(30));
```

Figure 2: création de la table



```
mysql> insert into employe values('djily','goumballa',1000000,'dakar');
Query OK, 1 row affected (0,01 sec)
```

Figure 3: remplissage du tableau

# Algèbre relationnelle

## Sélection (select)

Le SELECT instruction est utilisée pour sélectionner des données dans une base de données.

```
mysql> select * from employe;
+-----+-----+-----+-----+
| nom   | prenom | salaire | adresse |
+-----+-----+-----+-----+
| djily | goumballa | 1000000 | dakar |
| laye  | mbengue  | 10000   | thies  |
| top   | daour    | 200000  | dakar  |
| serigne | modou  | 50000   | saint-louis |
+-----+-----+-----+-----+
```

Figure 5: selection de tous les éléments de la table

```
mysql> select nom from employe;
+-----+
| nom |
+-----+
| djily |
| laye |
| top |
| serigne |
+-----+
4 rows in set (0,00 sec)
```

Figure 4: selection d'un élément du tableau

## Projection

La projection en algèbre entiténelle est un opérateur qui permet de sélectionner des colonnes spécifiques d'une table. La projection est réalisée à l'aide de la commande SELECT.

```
mysql> select nom,prenom from employe;
+-----+-----+
| nom    | prenom |
+-----+-----+
| djily   | gouballa |
| laye    | mbengue |
| top     | daour   |
| serigne | modou   |
+-----+-----+
4 rows in set (0,00 sec)
```

Figure 6: Projection

## Produit cartésien

Le produit cartésien en algèbre entiténelle est un opérateur qui permet de combiner chaque tuple (ligne) d'une entité avec chaque tuple d'une autre entité.

```
mysql> SELECT * FROM employe, clients;
+-----+-----+-----+-----+-----+-----+-----+
| nom    | prenom | salaire | adresse | nom    | prenom | telephone |
+-----+-----+-----+-----+-----+-----+-----+
| djily   | gouballa | 1000000 | dakar    | diouf   | poo    | 787899009 |
| djily   | gouballa | 1000000 | dakar    | seck    | mbakhal | 781545454 |
| djily   | gouballa | 1000000 | dakar    | saar    | abdou  | 778789890 |
| laye    | mbengue  | 10000   | thies    | diouf   | poo    | 787899009 |
| laye    | mbengue  | 10000   | thies    | seck    | mbakhal | 781545454 |
| laye    | mbengue  | 10000   | thies    | saar    | abdou  | 778789890 |
| top     | daour    | 200000  | dakar    | diouf   | poo    | 787899009 |
| top     | daour    | 200000  | dakar    | seck    | mbakhal | 781545454 |
| top     | daour    | 200000  | dakar    | saar    | abdou  | 778789890 |
| serigne | modou    | 50000   | saint-louis | diouf   | poo    | 787899009 |
| serigne | modou    | 50000   | saint-louis | seck    | mbakhal | 781545454 |
| serigne | modou    | 50000   | saint-louis | saar    | abdou  | 778789890 |
+-----+-----+-----+-----+-----+-----+-----+
```

Figure 7: Produit cartésien

## Union

L'union est un opérateur qui permet de combiner les tuples (lignes) de deux entités en une seule entité contenant tous les tuples uniques de chaque entité.

```
mysql> select nom from employe
-> union
-> select prenom from clients;
+-----+
| nom    |
+-----+
| djily   |
| laye    |
| top     |
| serigne |
| abdou   |
| mbakhal |
| poo     |
+-----+
```

Figure 8: union

## Différence

La différence est un opérateur qui permet de sélectionner les tuples (lignes) d'une entité qui ne sont pas présents dans une autre entité. La différence est réalisée à l'aide de la commande SELECT, en utilisant l'opérateur MINUS ou EXCEPT entre deux requêtes SELECT.

```
mysql> select nom from clients
-> except
-> select nom from employe;
+-----+
| nom   |
+-----+
| saar  |
| seck  |
| diouf |
+-----+
```

Figure 9: Différence

## Intersection

L'intersection est un opérateur qui permet de sélectionner les tuples (lignes) communs à deux entités. L'intersection est réalisée à l'aide de la commande SELECT, en utilisant l'opérateur INTERSECT entre deux requêtes SELECT.

```
mysql> select nom,prenom from employe
-> INTERSECT
-> select nom,prenom from clients;
Empty set (0,01 sec)
```

Figure 10: Intersection

Les deux tables n'ont pas d'éléments en commun.

## JOINTURE

La jointure est un opérateur qui permet de combiner deux ou plusieurs entités en utilisant une ou plusieurs colonnes communes.

### INNER JOIN

Il renvoie les enregistrements dont les valeurs correspondent dans les deux tables.

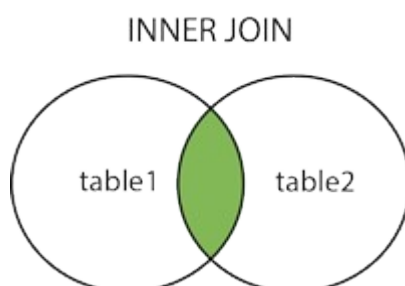
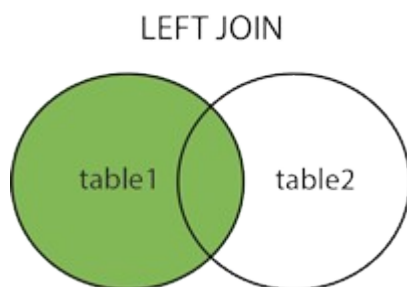


Figure 11: INNER JOIN

## LEFT JOIN

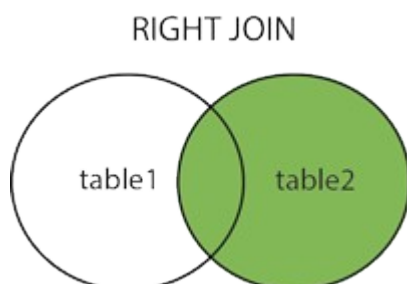
Il renvoie tous les enregistrements de la table de gauche et les enregistrements correspondants de la table de droite.



*Figure 12: LEFT JOIN*

## RIGHT JOIN:

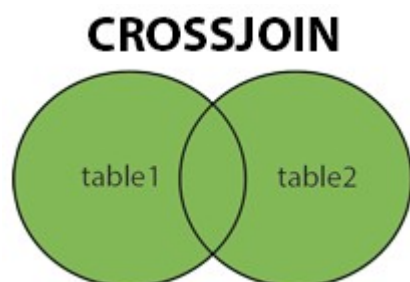
Il renvoie tous les enregistrements de la table de droite et les enregistrements correspondants de la table de gauche.



*Figure 13: RIGHT JOIN*

## CROSS JOIN:

Il renvoie tous les enregistrements des deux tables.




*Figure 14: CROSS JOIN*

# Langage de définition des données

## CREATE

La fonction permet de créer une base de données, une nouvelle table...

Cliquez pour voir un exemple  [creation database](#)

## DROP

La fonction permet de supprimer une base de données, une table...

Exemple: drop table employe; # supprime la table employe

## ALTER

La commande ALTER permet de modifier une table existante. Idéal pour ajouter une colonne, supprimer une colonne ou modifier une colonne existante, par exemple pour changer le type.

```
mysql> alter table clients
-> add id int;
Query OK, 0 rows affected (0,02 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> select * from clients;
+-----+-----+-----+-----+
| nom   | prenom | telephone | id   |
+-----+-----+-----+-----+
| saar  | abdou  | 778789890 | NULL |
| seck  | mbakhal | 781545454 | NULL |
| diouf | poo    | 787899009 | NULL |
+-----+-----+-----+-----+
```

Figure 15: Alter

## RENAME

RENAME est une instruction utilisée pour renommer une table existante, une colonne ou une contrainte dans une base de données.

```
mysql> rename table clients to personnes;
Query OK, 0 rows affected (0,01 sec)

mysql> show tables;
+-----+
| Tables_in_test |
+-----+
| employes       |
| personnes      |
+-----+
```

Figure 16: RENAME

## TRUNCATE

La commande TRUNCATE permet de supprimer toutes les données d'une table sans supprimer la table en elle-même.

Exemple: TRUNCATE table employes;

## Langage de manipulation de données

### INSERT

INSERT est une instruction utilisée pour insérer une ou plusieurs lignes dans une table existante dans une base de données.

Cliquez pour voir un exemple  remplissage du tableau

### UPDATE

UPDATE est une instruction utilisée pour modifier les données existantes dans une table dans une base de données.

```
mysql> update employes
-> set nom='bien'
-> where adresse='dakar';
Query OK, 2 rows affected (0,01 sec)
Rows matched: 2  Changed: 2  Warnings: 0

mysql> select * from employes;
```

nom	prenom	salaire	adresse
bien	goumballa	1000000	dakar
laye	mbengue	10000	thies
bien	daour	200000	dakar
serigne	modou	50000	saint-louis

*Figure 17: UPDATE*


### DELETE

DELETE est une instruction utilisée pour supprimer des données existantes dans une table dans une base de données.


Exemple: Delete from employes where salaire<=50000; supprime les lignes ayant un salaire inférieur à 50000

# Langage d'interrogation de donnée

## Jointure

Vous pouvez lz retrouver ici.  JOINTURE

## Projection

Vous pouvez lz retrouver ici.  Projection

## Sous requête

Une sous-requête (aussi appelé “requête imbriquée” ou “requête en cascade”) consiste à exécuter une requête à l’intérieur d’une autre requête.

```
mysql> SELECT nom, salaire
-> FROM employes
-> where salaire<=50000;

+-----+-----+
| nom    | salaire |
+-----+-----+
| laye    | 10000   |
| serigne | 50000   |
+-----+-----+
```

Figure 18: SOUS REQUETE

## Alias (AS)

Un alias est un nom alternatif donné à une table ou à une colonne dans une requête.


```
mysql> select nom as noms
-> from `employes`;

+-----+
| noms  |
+-----+
| bien  |
| laye  |
| bien  |
| serigne |
+-----+
```

Figure 19: ALIAS

## Commande

### UNION

Vous pouvez le retrouver ici.  Union

### GROUP BY

La commande GROUP BY est utilisée pour regrouper les enregistrements d'une table en fonction des valeurs d'une ou plusieurs colonnes

```
mysql> select adresse, sum(salaire)
-> from employes
-> group by adresse;
```

adresse	sum(salaire)
dakar	1200000
thies	10000
saint-louis	50000

Figure 20: GROUP BY

### ORDER BY

La commande ORDER BY est utilisée pour trier(ordre croissant (ASC) ou décroissant (DESC)) les résultats d'une requête en fonction des valeurs d'une ou plusieurs colonnes d'une table.

```
mysql> select * from employes
-> order by salaire desc;
```

nom	prenom	salaire	adresse
bien	goumballa	1000000	dakar
bien	daour	200000	dakar
serigne	modou	50000	saint-louis
laye	mbengue	10000	thies

Figure 21: ORDER BY



## HAVING

La commande HAVING en SQL est utilisée pour filtrer les résultats d'une requête qui ont été regroupés.

```
mysql> select adresse, sum(salaire)
-> from employes
-> group by adresse
-> having sum(salaire)>50000;
+-----+-----+
| adresse | sum(salaire) |
+-----+-----+
| dakar   |      1200000 |
+-----+-----+
```

Figure 22: HAVING

## Fonction

### DISTINCT

La commande DISTINCT est utilisée pour retourner des résultats uniques dans une requête SELECT.

```
mysql> select distinct adresse
-> from employes;
+-----+
| adresse |
+-----+
| dakar   |
| thies   |
| saint-louis |
+-----+
```

Figure 23: DISTINCT

### COUNT

La fonction d'agrégation COUNT() permet de compter le nombre d'enregistrement dans une table.

```
mysql> select count(*) as nb_personnes
-> from employes;
+-----+
| nb_personnes |
+-----+
|          4   |
+-----+
```

Figure 24: COUNT

## LIMIT

La commande LIMIT est utilisée pour limiter le nombre de résultats renvoyés dans une requête SELECT.

```
mysql> select nom,prenom
-> from employes
-> limit 2;
+-----+-----+
| nom   | prenom  |
+-----+-----+
| bien  | gouballa|
| laye  | mbengue |
+-----+-----+
```

Figure 25: LIMIT

## LIKE %% (B%A)

La commande LIKE est utilisée pour rechercher des enregistrements dans une table qui correspondent à un modèle de chaîne de caractères spécifique.

```
mysql> select nom,prenom
-> from employes
-> where nom like '%e';
+-----+-----+
| nom   | prenom  |
+-----+-----+
| laye  | mbengue |
| serigne | modou  |
+-----+-----+
```

Figure 26: LIKE %% (B%A)

## IN & NOT IN

Les commandes IN et NOT IN sont utilisées pour rechercher des enregistrements dans une table où la valeur d'une colonne donnée correspond à une liste de valeurs spécifiées ou ne correspond pas à cette liste.

```
mysql> select nom,prenom
-> from employes
-> where adresse in ('thies','saint-louis');
+-----+-----+
| nom   | prenom  |
+-----+-----+
| laye  | mbengue |
| serigne | modou  |
+-----+-----+
```

Figure 28: IN

```
mysql> select nom,prenom
-> from employes
-> where adresse not in ('thies','saint-louis');
+-----+-----+
| nom   | prenom  |
+-----+-----+
| bien  | gouballa|
| bien  | daour   |
+-----+-----+
```

Figure 27: NOT IN

## BETWEEN

La commande BETWEEN est utilisée pour rechercher des enregistrements dans une table où la valeur d'une colonne donnée est comprise dans une plage de valeurs spécifiée.

```
mysql> select prenom,salaire
-> from employes
-> where salaire between 200000 and 1000000;
+-----+-----+
| prenom | salaire |
+-----+-----+
| gouballa | 1000000 |
| daour | 200000 |
+-----+-----+
```

Figure 29: BETWEEN

## AVG

La commande AVG est utilisée pour calculer la moyenne des valeurs numériques d'une colonne dans une table.

```
mysql> select avg(salaire) as moyenne
-> from employes;
+-----+
| moyenne |
+-----+
| 315000.0000 |
+-----+
```

Figure 30: AVG

## MIN

La commande MIN est utilisée pour trouver la plus petite valeur dans une colonne donnée d'une table.

```
mysql> select min(salaire) as mini
-> from employes;
+-----+
| mini |
+-----+
| 10000 |
+-----+
```

Figure 31: MIN

## MAX

La commande MAX est utilisée pour trouver la plus grande valeur dans une colonne donnée d'une table.

```
mysql> select max(salaire) as maxi  
-> from employes;  
  
+-----+  
| maxi   |  
+-----+  
| 1000000 |  
+-----+
```

*Figure 32: MAX*

## SUM

La commande SUM est utilisée pour calculer la somme des valeurs numériques d'une colonne dans une table.

```
mysql> select sum(salaire) as somme  
-> from employes;  
  
+-----+  
| somme   |  
+-----+  
| 1260000 |  
+-----+
```

*Figure 33: SUM*

## Index des figures

Figure 1: creation database.....	1
Figure 2: création de la table.....	1
Figure 3: remplissage du tableau.....	1
Figure 4: selection d'un élément du tableau.....	1
Figure 5: selection de tous les éléments de la table.....	1
Figure 6: Projection.....	2
Figure 7: Produit cartésien.....	2
Figure 8: union.....	2
Figure 9: Différence.....	3
Figure 10: Intersection.....	3
Figure 11: INNER JOIN.....	3
Figure 12: LEFT JOIN.....	4
Figure 13: RIGHT JOIN.....	4
Figure 14: CROSS JOIN.....	4
Figure 15: Alter.....	5
Figure 16: RENAME.....	5
Figure 17: UPDATE.....	6
Figure 18: SOUS REQUETE.....	7
Figure 19: ALIAS.....	7
Figure 20: GROUP BY.....	8
Figure 21: ORDER BY.....	8
Figure 22: HAVING.....	9
Figure 23: DISTINCT.....	9
Figure 24: COUNT.....	9
Figure 25: LIMIT.....	10
Figure 26: LIKE %% (B%A).....	10
Figure 27: NOT IN.....	10
Figure 28: IN.....	10
Figure 29: BETWEEN.....	11
Figure 30: AVG.....	11
Figure 31: MIN.....	11
Figure 32: MAX.....	12
Figure 33: SUM.....	12