SYSTÈME DE GESTION DE BASE DE DONNÉES

Qu’est-ce qu’une base de données ?

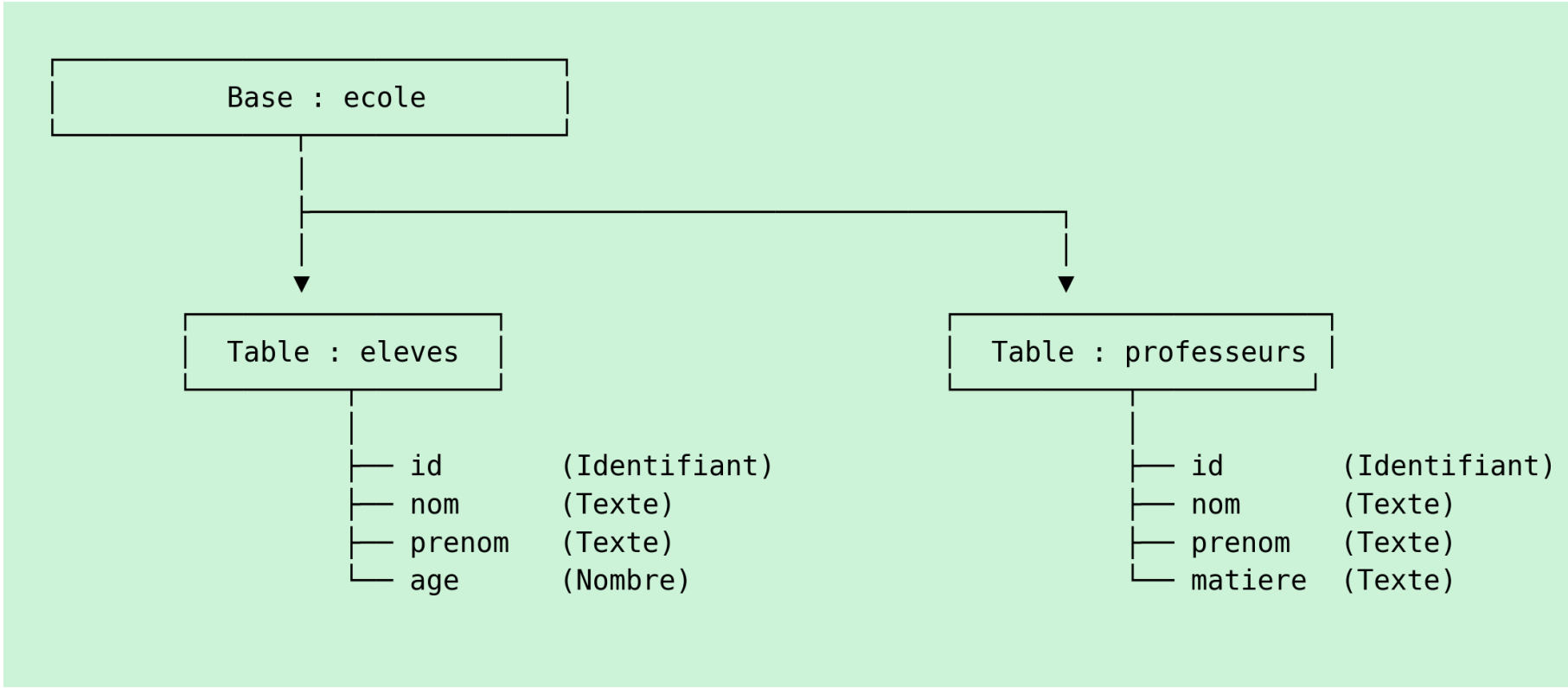
- Une **base de données** est un **ensemble organisé de données**.
- MySQL est un **SGBD** (Système de Gestion de Base de Données) qui permet :
  - de **stocker** des données,
  - de **consulter** les données,
  - de **modifier** les données,
  - de **supprimer** des données.

Exemples d’utilisation : gestion d’un site web, d’un magasin, d’un réseau social, d’une école…

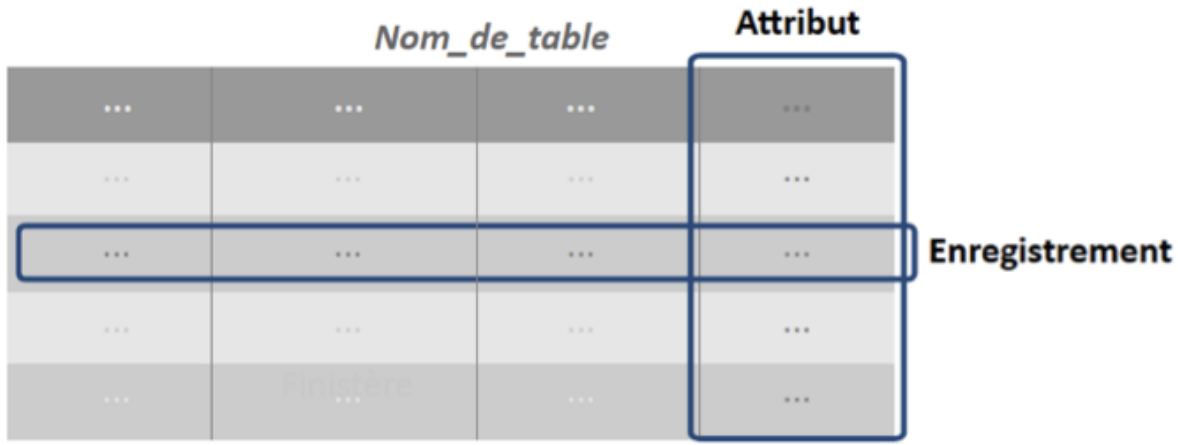
STRUCTURE

Concept	Description	Exemple
Base de données	Contient plusieurs tables	ecole
Table	Contient des données organisées en lignes et colonnes	eleves, professeurs
Ligne (enregistrement)	Une donnée complète	('Dupont', 'Paul', 18)
Colonne (champ)	Un type d’information	nom, prenom, age

STRUCTURE


Base : ecole
Table : eleves
id (Identifiant)
nom (Texte)
prenom (Texte)
age (Nombre)
Table : professeurs
id (Identifiant)
nom (Texte)
prenom (Texte)
matiere (Texte)

Table, attribut et enregistrement



Type de l’attribut

Chimistes

...	...	Naissance	...
...	...	...	...
...	Volta	18-02-1745	...
...	...	...	...
...	Finistere	...	...

date

Chimistes

chaîne de caractères

...	Nom	...	...
...	...	...	...
...	Volta	...	...
...	...	...	...
...	Finistere	...	...

Villes

...	...	...	Population
...	...	...	...
Bordeaux	...	...	239157
...	...	...	...
...	Finistere	...	...

nombre entier

Hydrocarbures

...	...	PM	...
...	...	...	...
...	...	...	...
...	...	...	...
Butane	...	58,123	...

nombre réel

Exemple

Villes

Ville	Departement	Region	Population
Strasbourg	Bas-Rhin	Alsace	271782
Bordeaux	Gironde	Aquitaine	239157
Dijon	Côte-d'Or	Bourgogne	151212
Brest	Finistère	Bretagne	141303
Amiens	Somme	Picardie	133448
...	...	...	...

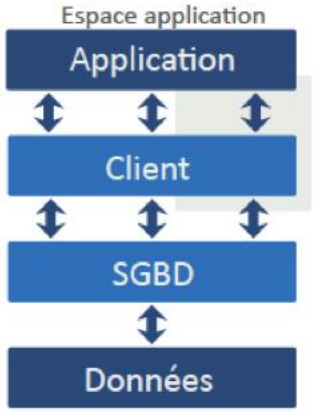
4 attributs

- 3 textuels : `Ville`, `Departement`, `Region`
- 1 numérique : `Population`

261 enregistrements

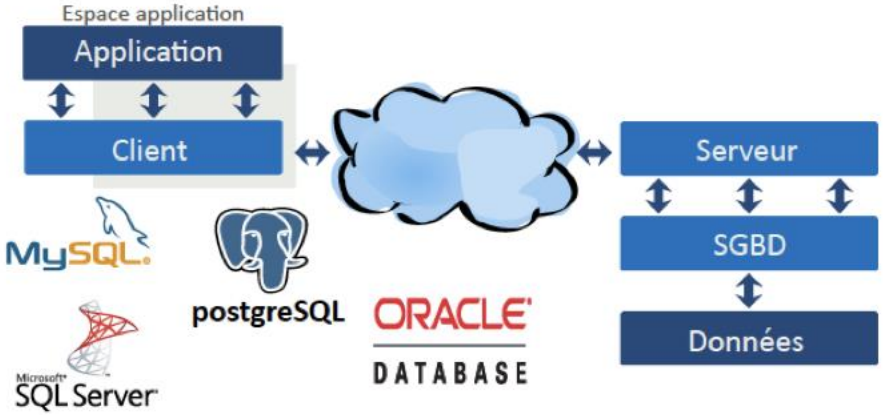
Les différents modèles d'architecture des bases de données

Modèle client-serveur



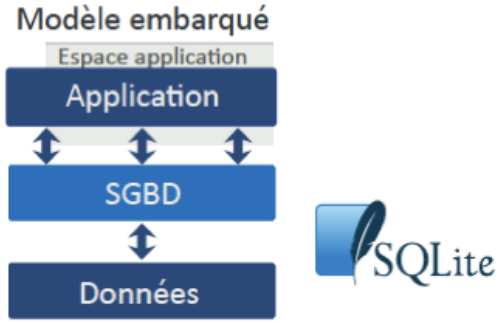
l'application passe par un **client** qui dialogue avec le **SGBD** pour accéder aux **données**.

Modèle client-serveur réseau



l'**application cliente** envoie ses requêtes via le réseau à un **serveur** qui exécute le **SGBD** (MySQL, PostgreSQL, Oracle, SQL Server) pour accéder aux **données**.

Modèle embarqué



l'**application** intègre directement le **SGBD** (comme SQLite) pour gérer en local ses **données**, sans passer par un serveur externe.

CRÉATION, MODIFICATION ET SUPPRESSION DE TABLE

```
CREATE TABLE nom_table(
    colonne1 datatype  PRIMARY KEY ,
    colonne2 datatype,
    colonne3 datatype,
    ...
    colonneN datatype,
);
```

```
CREATE TABLE Etudiants(
    "Numero"          INTEGER PRIMARY KEY NOT NULL,
    "Nom"             TEXT                NOT NULL,
    "Prenom"          TEXT                NOT NULL,
    "Date_naissance"  INTEGER             NOT NULL
);
```

```
ALTER TABLE Villes ADD COLUMN "CodePostal";

ALTER TABLE Villes DROP COLUMN "CodePostal";

ALTER TABLE Produits RENAME TO Ordinateurs;
```

```
DROP TABLE Produits;
```

MODIFICATION, SUPPRESSION D'ENREGISTREMENT

```
UPDATE Nom_Table
       SET champ1 = valeur1 , champ2 = valeur2 ,
           .... , champN = valeurN
       WHERE [condition];
```

```
UPDATE Villes SET Population = 56
              WHERE Ville = "Castelmoron-d'Albret";
```

```
DELETE FROM Nom_Table WHERE [condition];
```

Exemple

```
DELETE FROM Villes WHERE ( Population < 100
        AND Ville IS NOT "Castelmoron-d'Albret");

DELETE FROM Villes WHERE ( Population < 100
        AND Ville != "Castelmoron-d'Albret");
```

INSERTION D'UN ENREGISTREMENT

```
INSERT INTO Nom_Table [(champ1,champ2,...,champN)]
       VALUES (valeur1,valeur2,...,valeurN);
```

```
INSERT INTO Villes (Ville,Departement,Region,
                    Population)
       VALUES ("Castelmoron-d'Albret","Gironde",
               "Aquitaine",54);
```

SÉLECTION D'ENREGISTREMENTS

```
SELECT * FROM Villes
        WHERE Region = "Alsace";
```

Résultat

Ville	Departement	Region	Population
Strasbourg	Bas-Rhin	Alsace	271782
Mulhouse	Haut-Rhin	Alsace	109588
Colmar	Haut-Rhin	Alsace	67615
Haguenau	Bas-Rhin	Alsace	34280
Schiltigheim	Bas-Rhin	Alsace	30952

Sélection de toutes les colonnes pour les enregistrements de la région Alsace

```
SELECT Ville, Population FROM Villes
```

Résultat

Ville	Population
Strasbourg	271782
Bordeaux	239157
Dijon	151212
Brest	141303
Amiens	133448
...	...

Sélection de certaines colonnes

```
SELECT Ville, Population FROM Villes
        WHERE Departement = "Bas-Rhin";
```

Résultat

Ville	Population
Strasbourg	271782
Haguenau	34280
Schiltigheim	30952

Sélection de toutes les colonnes pour certaines valeur enregistrées.

SÉLECTION D'ENREGISTREMENTS

```
SELECT Ville , Population FROM Villes
       ORDER BY Population DESC;
```

Résultat

Ville	Population
Strasbourg	271782
Bordeaux	239157
Dijon	151212
Brest	141303
Amiens	133448
...	...

Sélection et trie par ordre croissant

```
SELECT Ville , Population FROM Villes
       ORDER BY population DESC
       LIMIT 4;
```

Résultat

Ville	Population
Strasbourg	271782
Bordeaux	239157
Dijon	151212
Brest	141303

Sélection, trie par ordre croissant et affichage des 4 premières lignes

```
SELECT Ville , Population FROM Villes
       ORDER BY Population DESC
       LIMIT 4 OFFSET 3;
```

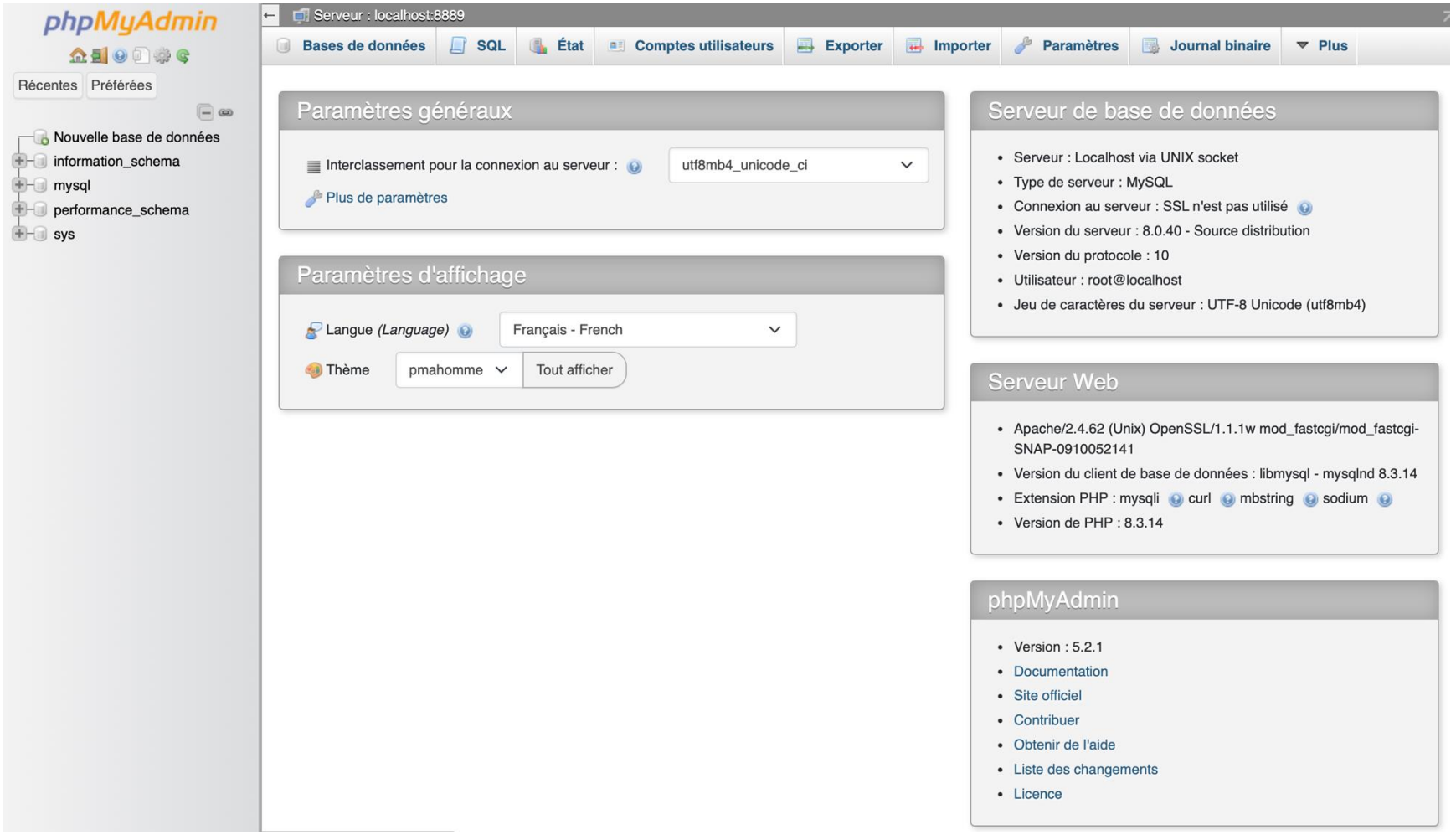
Résultat

Ville	Population
Brest	141303
Amiens	133448
Metz	120738
Orléans	114167

Sélection, trie par ordre croissant et affichage des 4 premières lignes en sautant les 3 premières lignes.

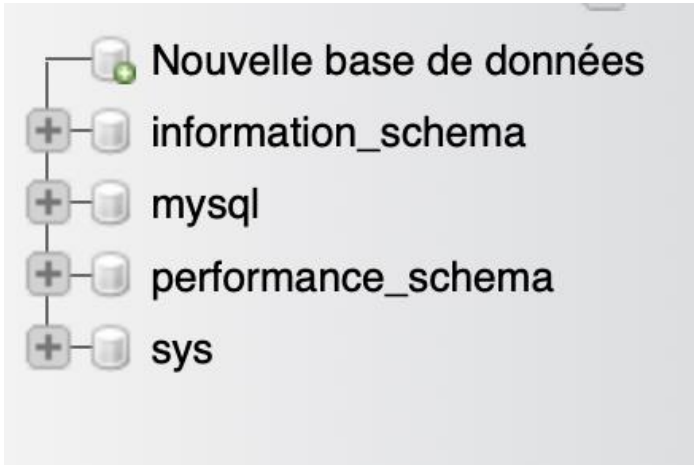
phpMyAdmin

phpMyAdmin est une **application web** (programmée en PHP) qui permet de gérer une base de données **MySQL** ou **MariaDB** à travers une interface graphique, depuis un navigateur.



Bases de données système de MySQL

Ces bases sont **créés automatiquement par MySQL**.
L'utilisateur ne les supprime pas : elles sont nécessaires au bon fonctionnement du serveur.

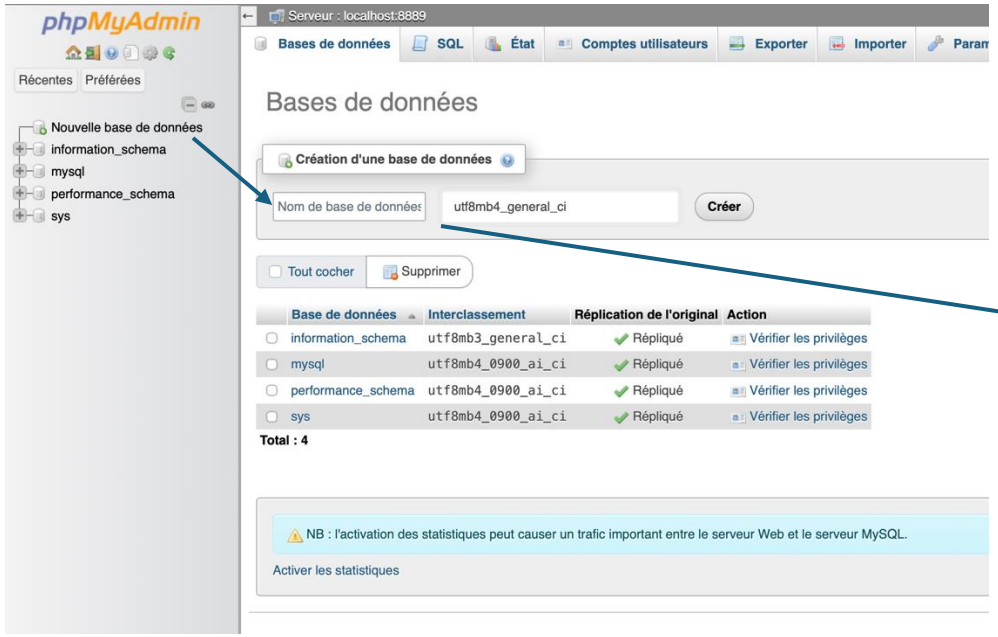


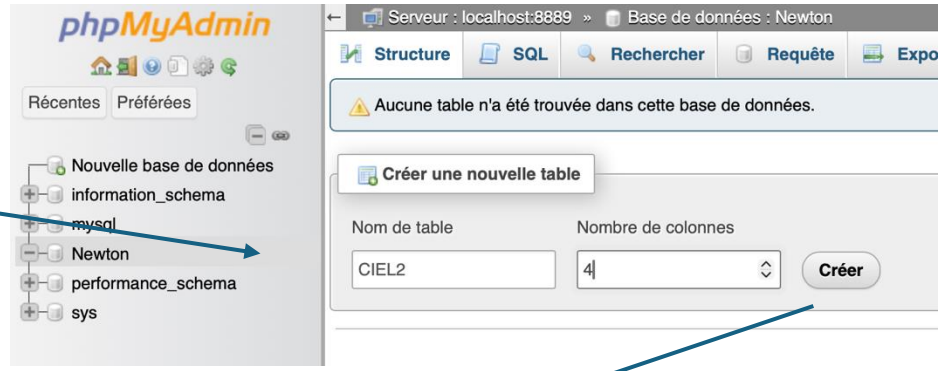
- **information_schema**
  Contient des informations *sur les bases de données elles-mêmes* (métadonnées).
  Exemple : noms des tables, colonnes, types de données, droits des utilisateurs.

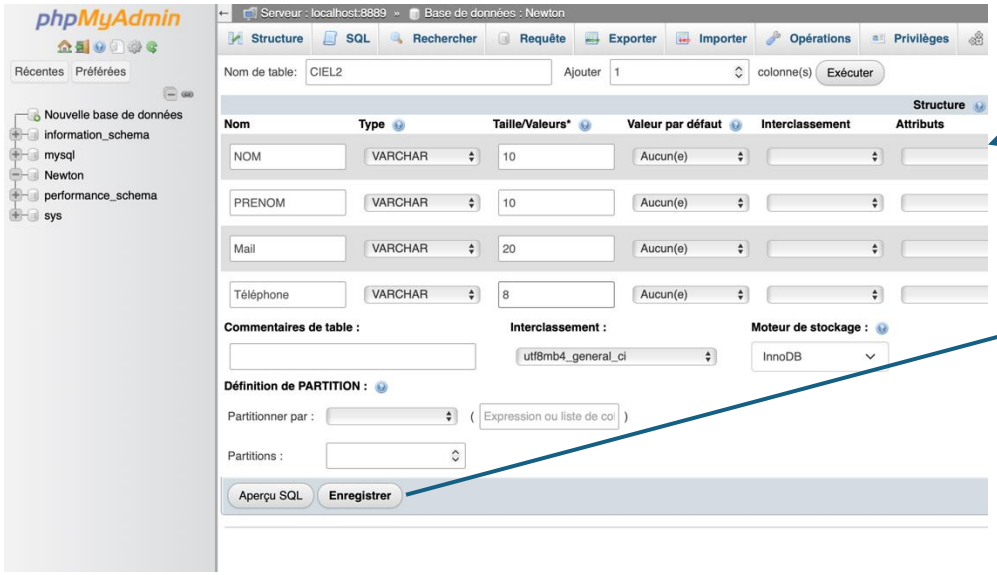
- **mysql**
  C'est la base **système principale** de MySQL.
  Elle gère les **utilisateurs, privilèges et mots de passe**.
  ⚠ On ne doit pas la modifier directement.

- **performance_schema**
  Sert à analyser les **performances** du serveur MySQL.
  Elle contient des statistiques sur l'utilisation des requêtes, du CPU, de la mémoire.

- **sys**
  Fournit une **vue simplifiée** des informations de performance.
  C'est comme un résumé facile à lire de performance_schema.


phpMyAdmin
Récentes
Préférées
Nouvelle base de données
information_schema
mysql
performance_schema
sys
Serveur : localhost:8889
Bases de données
SQL
État
Comptes utilisateurs
Exporter
Importer
Bases de données
Création d'une base de données
Nom de base de donnée
utf8mb4_general_ci
Créer
Tout cocher
Supprimer
Base de données
Interclassement
Réplication de l'original
Action
information_schema utf8mb3_general_ci Répliqué Vérifier les privilèges
mysql utf8mb4_0900_ai_ci Répliqué Vérifier les privilèges
performance_schema utf8mb4_0900_ai_ci Répliqué Vérifier les privilèges
sys utf8mb4_0900_ai_ci Répliqué Vérifier les privilèges
Total : 4
NB : l'activation des statistiques peut causer un trafic important entre le serveur Web et le serveur MySQL.
Activer les statistiques


phpMyAdmin
Récentes
Préférées
Nouvelle base de données
information_schema
mysql
Newton
performance_schema
sys
Serveur : localhost:8889 » Base de données : Newton
Structure
SQL
Rechercher
Requête
Aucune table n'a été trouvée dans cette base de données.
Créer une nouvelle table
Nom de table
Nombre de colonnes
CIEL2
4
Créer


phpMyAdmin
Récentes
Préférées
Nouvelle base de données
information_schema
mysql
Newton
performance_schema
sys
Serveur : localhost:8889 » Base de données : Newton
Structure
SQL
Rechercher
Requête
Exporter
Importer
Opérations
Privilèges
Nom de table: CIEL2
Ajouter 1 colonne(s)
Exécuter
Structure
Nom
Type
Taille/Valeurs*
Valeur par défaut
Interclassement
Attributs
NOM VARCHAR 10 Aucun(e)
PRENOM VARCHAR 10 Aucun(e)
Mail VARCHAR 20 Aucun(e)
Téléphone VARCHAR 8 Aucun(e)
Commentaires de table :
Interclassement :
utf8mb4_general_ci
Moteur de stockage :
InnoDB
Définition de PARTITION :
Partitionner par :
Expression ou liste de col
Partitions :
Aperçu SQL
Enregistrer


phpMyAdmin
Récentes
Préférées
Nouvelle base de données
information_schema
mysql
Newton
Nouvelle table
CIEL2
performance_schema
sys
Serveur : localhost:8889 » Base de données : Newton » Table : CIEL2
Parcourir
Structure
SQL
Rechercher
Insérer
Exporter
Importer
Privilèges
Opérations
Structure de table
Vue relationnelle
Nom Type Interclassement Attributs Null Valeur par défaut Commentaires Extra Action
1 NOM varchar(10) utf8mb4_general_ci Non Aucun(e) Modifier Supprimer Plus
2 PRENOM varchar(10) utf8mb4_general_ci Non Aucun(e) Modifier Supprimer Plus
3 Mail varchar(20) utf8mb4_general_ci Non Aucun(e) Modifier Supprimer Plus
4 Téléphone varchar(8) utf8mb4_general_ci Non Aucun(e) Modifier Supprimer Plus
Tout cocher
Avec la sélection :
Parcourir
Modifier
Supprimer
Primaire
Unique
Index
Spatial

CLÉ PRIMAIRE

```sql
SELECT * FROM Villes WHERE Ville = "Saint-Denis";
```

Résultat

Ville	Departement	Region	Population
Saint-Denis	La Réunion	La Réunion	141303
Saint-Denis	Seine-Saint-Denis	Île-de-France	106785

Problème : Quelle est la population de la ville Saint-Denis ?

CLÉ PRIMAIRE

Rôle d'une clé primaire (PRIMARY KEY)

- **clé primaire** est un **identifiant unique** pour chaque ligne d'une table.
- Elle garantit que **chaque enregistrement est différent** et peut être retrouvé facilement.
- Les valeurs d'une clé primaire ne peuvent pas être :
  - **NULL** (vide),
  - **dupliquées**.

Id	Ville	Departement	Region	Population
20	Saint-Denis	La Réunion	La Réunion	141303
37	Saint-Denis	Seine-Saint-Denis	Île-de-France	106785

CLÉ PRIMAIRE

Auto-incrémentation (AUTO_INCREMENT en MySQL)

- C'est une option qu'on peut associer à une **clé primaire** numérique.
- Elle permet à la base de données de **générer automatiquement une nouvelle valeur unique** pour chaque nouvel enregistrement.
- Ainsi, on n'a pas besoin d'indiquer soi-même l'id quand on insère une ligne : MySQL s'en occupe.

```
CREATE TABLE eleves (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nom VARCHAR(50),
  prenom VARCHAR(50),
  age INT
);
```

Puis si on ajoute un élève :

```
INSERT INTO eleves (nom, prenom, age)
VALUES ('Dupont', 'Paul', 18);
```

La base attribue automatiquement id = 1.
Si on ajoute un autre élève, il aura id = 2, puis id = 3, etc.

Comment choisir une clé primaire ?

1. **Elle doit identifier de façon unique chaque enregistrement.**
    - aucune ligne ne doit partager la même valeur de clé.

2. **Elle ne doit jamais changer.**
    - un identifiant qui évolue (comme une adresse mail, un numéro de téléphone) n'est pas un bon choix.

3. **Elle doit être simple et stable.**
    - éviter les clés trop longues ou qui risquent de se modifier.

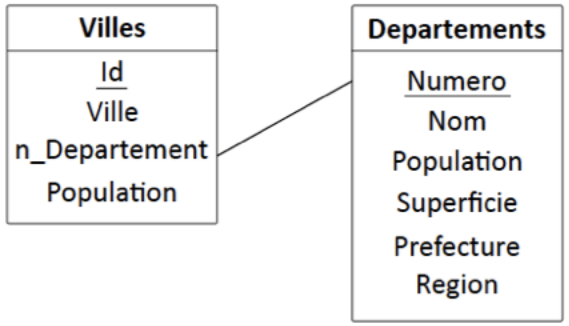
- **Clé artificielle** (souvent id AUTO_INCREMENT)
    - Crée un identifiant qui n'a de sens qu'en base.
    - Simple, rapide, recommandé pour la plupart des cas.

- **Clé naturelle** (issue des données elles-mêmes)
    - Exemple : N° de sécurité sociale, matricule étudiant, ISBN d'un livre.
        - Avantage : déjà unique dans la réalité.
        - Inconvénient : parfois trop long, ou sujet à modification.

CLÉ ÉTRANGÈRE

Définition d'une clé étrangère (FOREIGN KEY)

- Une **clé étrangère** est un **lien** entre deux tables.
- C'est une colonne d'une table qui fait **référence à la clé primaire** d'une autre table.
- Elle permet d'assurer la **cohérence des données** : on ne peut insérer une valeur dans la clé étrangère que si cette valeur existe déjà dans la table référencée.



- Chaque ville appartient à un **département existant**.
- On ne peut pas associer une ville à un numéro de département qui n'existe pas dans la table **Départements**.

- Dans la table **Départements** :
  - Numero est la **clé primaire** (soulignée).
  - Chaque département est identifié de manière unique par son numéro.
- Dans la table **Villes** :
  - Id est la **clé primaire** de la table.
  - n_Departement est une **clé étrangère** : elle fait référence au Numero de la table **Départements**.

CLÉ ÉTRANGÈRE

```sql
CREATE TABLE Villes (
   Id INTEGER PRIMARY KEY,
   Ville VARCHAR(50),
   n_Departement INTEGER,
   FOREIGN KEY(n_Departement) REFERENCES
                        Departements(Numero),
   ...
);
```

Ce code SQL sert à **créer la table Villes** avec une clé primaire et une clé étrangère.

La colonne n_Departement est un entier qui va contenir le **numéro du département** auquel appartient la ville.

CARDINALITÉ

La **cardinalité** décrit **le type de relation entre deux tables**, c'est-à-dire combien d'enregistrements d'une table peuvent être liés à combien d'enregistrements d'une autre table.

Les trois cardinalités principales

- **1–1 (un à un)**
  Chaque enregistrement d'une table correspond à **un seul** enregistrement dans l'autre table.
  Exemple : un **numéro de sécurité sociale** correspond à une **seule personne**.

- **1–N (un à plusieurs)**
  Un enregistrement d'une table peut correspondre à **plusieurs** enregistrements dans l'autre table.
  Exemple : un **département** possède plusieurs **villes**, mais chaque ville appartient à un seul département.

- **N–N (plusieurs à plusieurs)**
  Plusieurs enregistrements d'une table peuvent être associés à plusieurs enregistrements de l'autre table.
  Exemple : un **élève** peut suivre plusieurs **cours**, et chaque cours est suivi par plusieurs élèves.

CARDINALITÉ