Afpa
se former, avancer

LE LANGAGE SQL

STRUCTURED QUERY LANGUAGE

Concevoir et utiliser une base de données relationnelle.

Concepteur Développeur d'Applications

TABLE DES MATIÈRES

INTRODUCTION

L'OUTIL BASE DE DONNÉES (BDD)

Historique

Face à l'augmentation d'informations que les entreprises doivent gérer et partager, il apparaît dans les années 1960, la notion de concept de base de données (BDD).

Définition

Une BDD est un ensemble structuré de données enregistré sur un support accessibles pour satisfaire simultanément plusieurs utilisateurs de façon sélective et en un temps opportun.

La structure de l'ensemble requiert une description rigoureuse appelée SCHEMA.

LE SYSTÈME DE GESTION DE BASE DE DONNÉES - SGBD

LE SYSTÈME DE GESTION DE BASE DE DONNÉES

Historique

Les SGBD ont près de 50 ans d'histoires.

A la fin des années 1960, apparaissent les 1er SGBD (structure de type graphe et langage navigationnels).

Dans les années 1970, basée sur la théorie mathématique des relations, apparaissent les 1er SGBD/R (R pour relation).

Objectifs de l'approche BDD

4 objectifs principaux :

- Intégration et corrélation
- Flexibilité
- Disponibilité
- Sécurité

12 règles de Cood

- Les 12 règles de Cood sont un ensemble de règles édictées par Edgard F. Cood conçues pour définir ce qui est exigé d'un SGBD afin qu'il puisse être considéré comme SGDB/R

LES RÈGLES DE COOD

DÉTAILS DES 4 OBJECTIFS PRINCIPAUX

Intégration et corrélation

un "réservoir" commun (intégration) est constitué, représentant une modélisation (corrélation) aussi fidèle que possible de l'organisation réelle de l'entreprise.

Toutes les applications puisent dans ce réservoir les données les concernant, évitant les duplications.

Flexibilité

Dans le cas des BDD, cette notion porte généralement le nom d'indépendance.

- Indépendance physique : le changement de support n'aura pas d'impact que l'accès aux données.
- Indépendance logique : Les changements au niveau logique (tables, colonnes, rangées, etc) ne doivent pas exiger un changement dans l'application basée sur les structures.
- Indépendance des stratégies d'accès : on ne doit pas prendre en charge l'écriture des procédures d'accès aux données.

DÉTAILS DES 4 OBJECTIFS

Disponibilité

Le choix d'une approche BDD ne doit pas se traduire par des temps de traitement plus longs que ceux des systèmes antérieurs.

En fait, tout utilisateur doit (ou devrait !) pouvoir ignorer l'existence d'utilisateurs concurrents.

Ici, on parle de performance.

Sécurité

La sécurité couvre les aspects :

- L'intégrité ou protection des accès invalide (erreurs ou pannes) et contre l'incohérence des données dans la BDD.
- La confidentialité ou protection contre l'accès non autorisé ou la modification illégale des données.

LES DIFFÉRENTS LANGAGES D'UN SGBD

DIFFERENTS LANGAGES D'UN SGBD

Langage de Description de Données

Il permet de décrire précisément la structure de la base et le mode de stockage des données.

Dans une approche Base de Données, on effectue la description de toutes les données une fois pour toutes :

- elle est constituée de l'ensemble des tables et dictionnaires de la base, son schéma.

En particulier, le LDD précise :

- la structure logique des données (nom, type, contraintes spécifiques...),
- la structure physique (mode d'implantation sur les supports, mode d'accès), la définition des sous-schémas ou "vues".

Langage de Manipulation de Données

Il convient de rappeler que l'utilisation d'une BDD suppose un grand nombre d'utilisateurs (souvent non informaticiens) ayant tous des tâches et des besoins variés auxquels le LMD doit pouvoir répondre.

On peut répertorier :

- le langage d'interrogation ou langage de requête : syntaxe souple, accessible aux non-spécialistes, permet la formulation de demandes utilisant des critères variés et combinés. SQL en un exemple typique.
- le langage hôte : Pour les traitements réguliers ou d'importants volumes d'informations : une interface permettant l'utilisation de la base à l'aide des langages généraux (COBOL, C, Basic, Java...).

RÔLE DU SGBD

Rappelons succinctement les différentes fonctions assumées par un SGBD :

- description de la structure de la base (schéma)
- organisation du stockage physique
- manipulation des informations (sélection, extraction, mise à jour)
- protection (sécurité)

Pour personnaliser de façon fiable les accès à la base, il convient d'identifier l'utilisateur (login et mot de passe) et de vérifier qu'il est autorisé à effectuer sur les données les traitements qu'il demande (contrôle des droits d'accès par des ACL [Access Control List].

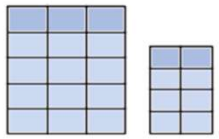
L'essentiel de la mise en œuvre de ces fonctions revient à une personne (ou une équipe) appelée administrateur de la BDD ou DataBase Administrator.

Les LMD se répartissent en 2 catégories principales :

- **Langages navigationnels :**
  - On les rencontre avec les SGBD '**hiérarchiques**' ou '**réseaux**'. Les requêtes du langage (ou questions) décrivent les chemins d'accès aux différentes données, celles-ci étant généralement chaînées entre elles.
- **Langages algébriques** (ex : SQL)
  - On les rencontre avec les SGBD relationnels. Ils utilisent, pour fournir des résultats aux requêtes, les opérateurs de l'algèbre relationnelle.

LANGAGES ALGÉBRIQUES

- **le modèle relationnel** (**SGBDR**, *Système de gestion de bases de données relationnelles*) : les données sont enregistrées dans des tableaux à deux dimensions (lignes et colonnes). La manipulation de ces données se fait selon la théorie mathématique des relations

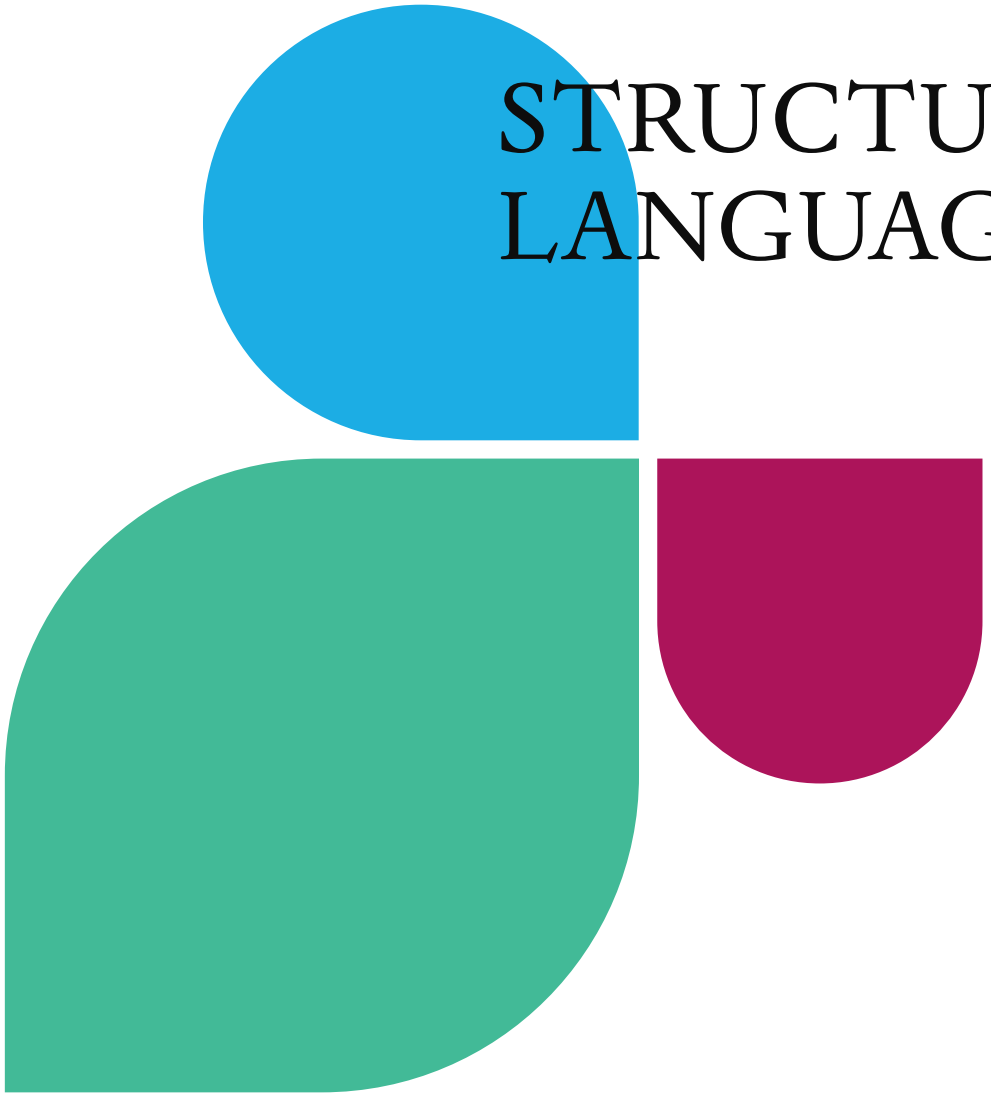


Ce sera ce type de modèle que nous allons étudier et utiliser.

Dans ce modèle :

- une relation est un ensemble de tuples (collections non ordonnées de valeurs connues avec des noms) ou uplets (collections ordonnées de valeurs connues avec des noms) dont l'ordre est sans importance : chaque tuple est unique.
- les colonnes de la table sont appelées attributs ou champs. Leur ordre est défini lors de la création de la table.
- une clé est un ensemble ordonné d'attributs qui caractérise un tuple. Une clé primaire le caractérise de manière unique, à l'inverse d'une clé secondaire.

STRUCTURED QUERY LANGUAGE - SQL



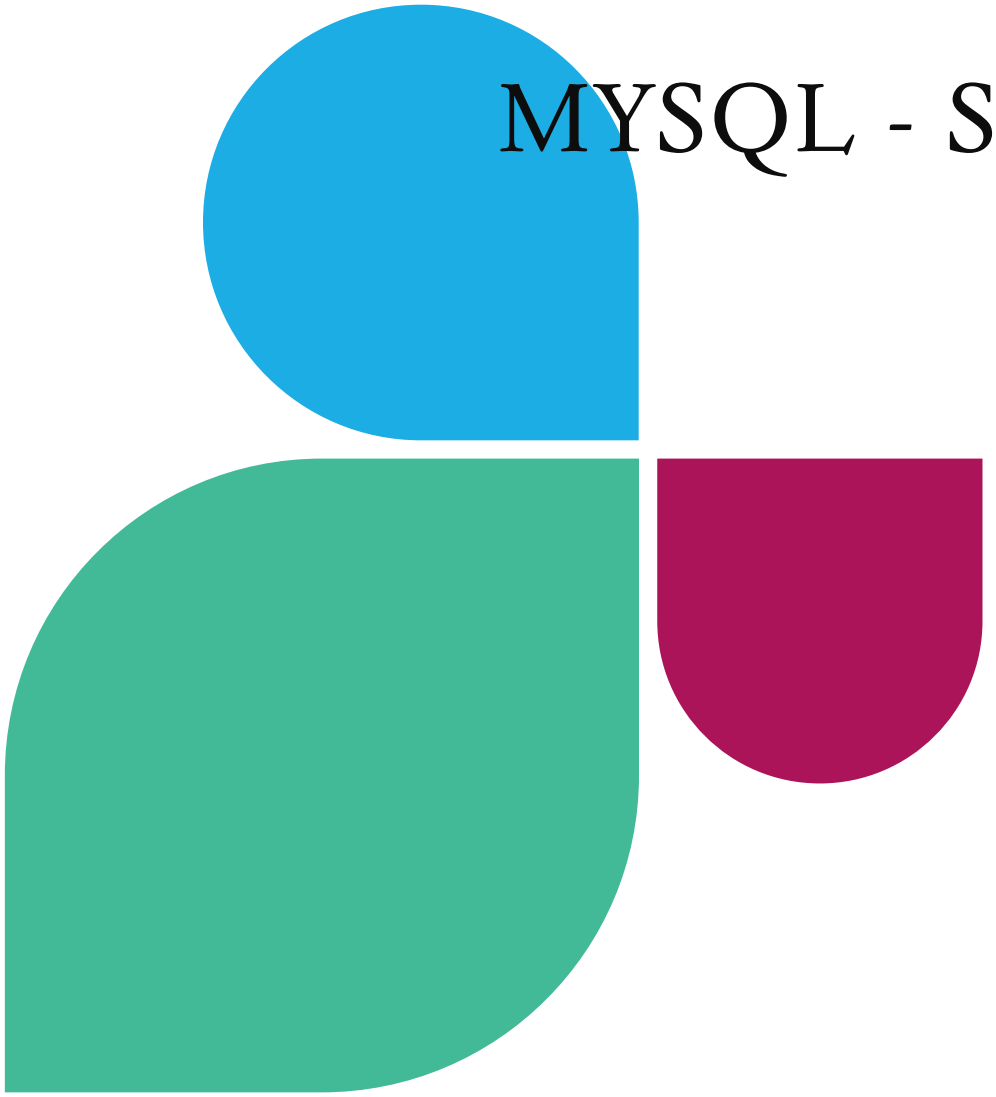
COMPOSITION

SQL est un langage déclaratif, il n'est donc pas à proprement parlé un langage de programmation, mais plutôt une interface standard pour accéder aux bases de données.

Il est composé de quatre sous ensemble :

- Le LDD (Langage de Définition de Données) pour créer et supprimer des objets dans la base de données (tables, contraintes d'intégrité, vues, etc..).
  - exemple : CREATE, DROP, ALTER
- Le LCD (Langage de Contrôle de Données) pour gérer les droits sur les objets de la base (création des utilisateurs et affectations de leurs droits.
  - exemple : GRANT, REVOKE
- Le LMD (Langage de Manipulation de Données) pour la recherche, l'insertion, la mise à jour et la suppression de données.
  - exemple : INSERT, UPDATE, DELETE, SELECT
- Le LCT (langage de Contrôle de Transaction) pour les gestions des transaction ou annulation de modifications de données.
  - exemple : COMMIT, ROLLBACK

MYSQL - SGBD/R



MYSQL

MySQL est un Système de Gestion de Bases de Données Relationnelles (abrégé SGBD/R)

- logiciel qui permet de gérer des bases de données, et donc de gérer de grosses quantités d'informations.
- utilise pour cela le langage SQL.
- Un SGBD/R les plus connus et les plus utilisés.
- MySQL peut donc s'utiliser seul, mais est la plupart du temps combiné à un autre langage de programmation : PHP, par exemple, pour de nombreux sites web, mais aussi Java, Python, C++, et beaucoup, beaucoup d'autres.

- MySQL peut s'utiliser en ligne de commande ou avec une interface graphique (MySql Workbench).
- Pour se connecter à MySQL en ligne de commande, on utilise :
  - mysql -u utilisateur [-h hôte] -p
- Pour terminer une instruction SQL, on utilise le caractère ;
- En SQL, les chaînes de caractères doivent être entourées de guillemets simples ' texte '
- Lorsque l'on se connecte à MySQL, il faut définir l'encodage utilisé, soit directement dans la connexion avec l'option --default-character-set, soit avec la commande : SET NAMES 'utf8';
- Ensuite, on peut utiliser le langage SQL pour manipuler les bases.

MYSQL

Fichiers de configuration

Si l'on veut garder la même configuration en permanence malgré les redémarrages de serveur et pour toutes les sessions, il existe une solution plus simple que de démarrer chaque fois le logiciel avec les options désirées : utiliser les fichiers de configuration.

Emplacement	Commentaire
WINDIR\my.ini, WINDIR\my.cnf	WINDIR est le dossier de Windows. Généralement, il s'agit du dossier C:\Windows. Pour vérifier, il suffit d'exécuter la commande suivante (dans la ligne de commande Windows) : echo %WINDIR%
C:\my.ini ou C:\my.cnf	
INSTALLDIR\my.ini ou INSTALLDIR\my.cnf	INSTALLDIR est le dossier dans lequel MySQL a été installé.

MYSQL : LES MOTEURS DU SGBD

MyISAM

Le moteur historique de MySQL, il est d'ailleurs utilisé par défaut.

- Rapide tant en lecture qu'en écriture, et il est très bien intégré à MySQL
- Ne gère pas les relations, c'est-à-dire qu'on ne peut pas définir de contrainte d'intégrité référentielle (clé étrangère / foreign key).

InnoDB

Apparu par la suite, la plus importante différence avec MyISAM

- Moteur relationnel : il permet de créer des contraintes d'intégrité, tout comme d'autres SGBD comme PostgreSQL, SQL Server ou Oracle.

À première vue, on pourrait se dire qu'il vaut mieux utiliser InnoDB que MyISAM

- Point important avant de se décider : La Performance

MyISAM et InnoDB

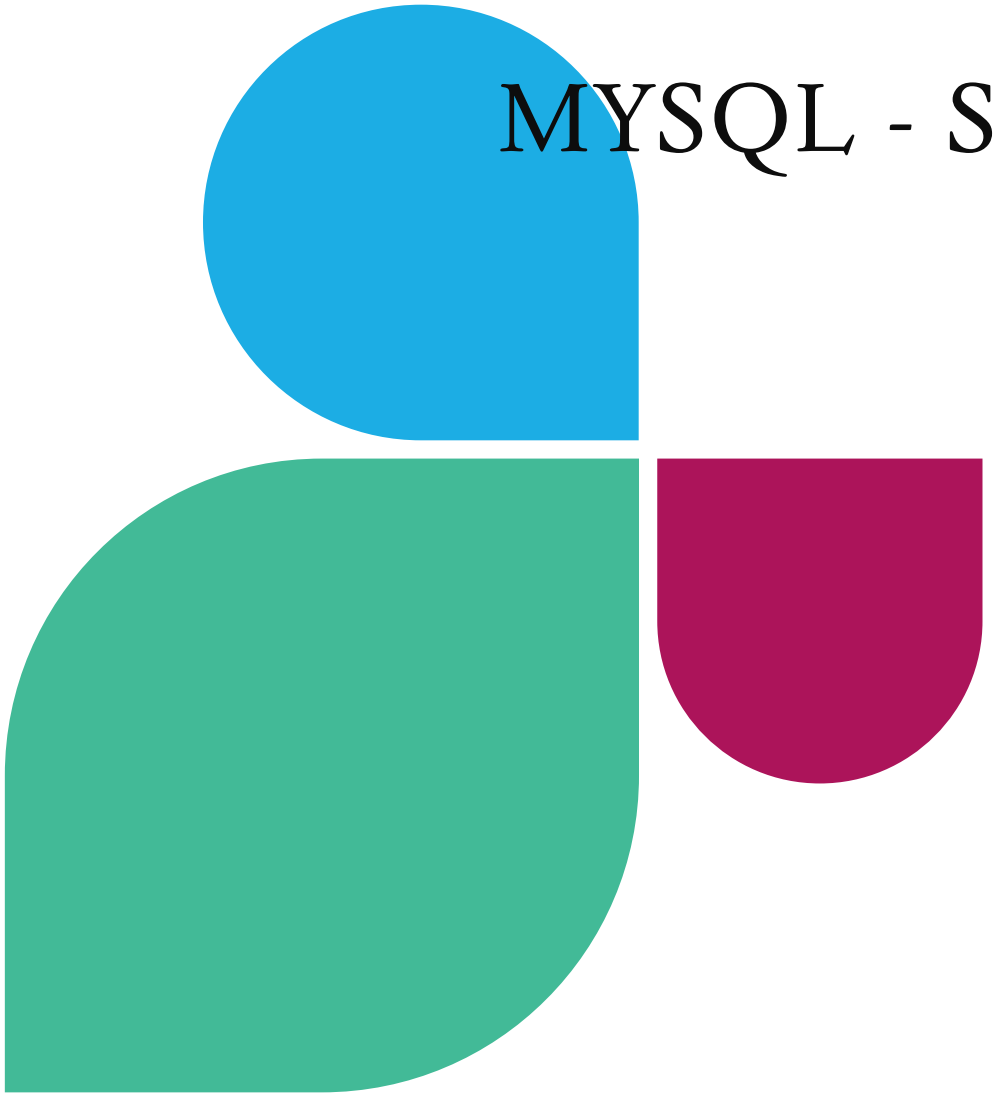
- sensiblement la même performance en lecture,
- en écriture, InnoDB est plus lent que MyISAM.
- InnoDB occupent plus de place de stockage que MyISAM.

TYPE DE DONNÉES DE MYSQL

- MySQL définit plusieurs types de données
  - des numériques entiers,
  - des numériques décimaux,
  - des textes alphanumériques,
  - des chaînes binaires alphanumériques
  - des données temporelles.
- Il est important de toujours utiliser le type de donnée adapté à la situation.
- SET et ENUM sont des types de données qui n'existent que chez MySQL. Il vaut donc mieux éviter de les utiliser.

```
TINYINT (1o : -127+128)
SMALLINT (2o : +-65 000)
MEDIUMINT (3o : +-16 000 000)
INT (4o : +- 2 000 000 000)
BIGINT (8o : +- 9 trillions)
  Intervalle précis : -(2^(8*N-1)) -> (2^8*N)-1
  /!\ INT(2) = "2 chiffres affichés" -- ET NON PAS "nombre à 2 chiffres"

FLOAT(M,D) DOUBLE(M,D) FLOAT(D=0->53)
  /!\ 8,3 -> 12345,678 -- PAS 12345678,123!

TIME (HH:MM)
YEAR (AAAA)
DATE (AAAA-MM-JJ)
DATETIME (AAAA-MM-JJ HH:MM; années 1000->9999)
TIMESTAMP (comme date, mais 1970->2038, compatible Unix)

VARCHAR(ligne)
TEXT (multi-lignes; taille max=65535)
BLOB (binaire; taille max=65535)

Variantes :
TINY (max=255)
MEDIUM (max=~16000)
LONG (max=4Go)
  Ex : TINYTEXT, LONGBLOB, MEDIUMTEXT

ENUM ('valeur1', 'valeur2', ...) -- (default NULL, ou '' si NOT NULL)
```

MYSQL - SYNTAXE



MOTS RÉSERVÉS ET CONVENTION DE NOMMAGES

Mots réservés

Il existe tous un ensemble de mots réservés que nous pouvons donc pas utiliser comme nom de table, nom d'attributs, variables etc…

Liste des mots réservés

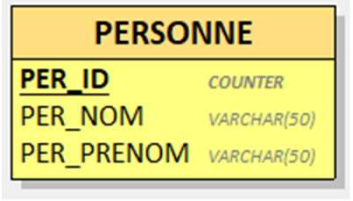
Dans tous les cas, le SGBD vous l'indiquera lors de l'écriture vos requêtes. Il y a un moyen d'échappement mais si on peut s'éviter des complications…

Convention de nommages.

Comme pour le code, vous devez convenir d'une convention de nommage pour l'écriture de vos tables et de leurs attributs.

Les conventions que nous avons déjà utilisées, soit PascalCase, camelCase etc… sont à conserver.

Cependant, une convention que j'aime utiliser pour avoir une facilité de lecture de mes requêtes SQL :



LES VARIABLES UTILISATEUR

Vous avez la possibilité d'utiliser, de déclarer des variables.

Il existe trois types de variables dans le langage SQL.

La commande SHOW VARIABLES; permet l'affichage dans MySQL Workbench, de l'ensemble des variables et de leurs valeurs. (également disponible en ligne de commandes).

- Variables locales
  - Forcément de type scalaires (entier, décimale, chaînes, booléen)
  - Pour un tableau, il faut créer une table temporaire ou concaténer chaque ligne…
  - Elles sont déclarées après DECLARE avec leur nom, leur type, et éventuellement la valeur par défaut.
    - DECLARE MaVariable INT DEFAULT 1;
- Variables de session
  - Leur nom commence par @ et dure le temps du thread.
  - Déclarer par SET, ou SELECT
    - SET @date = 'date'; SELECT @test := 2;
  - Une variable définie dans la liste de champs ne peut être utilisé comme une condition.
- Variables globales
  - Visible pour tous les utilisateurs et est précédée de @@
  - peuvent modifier les fichiers de configurations pendant la session, donc nécessaire de préciser le critère définitif ou éphémère avec SET GLOBAL ou SET SESSION

LES ALIAS

Les Alias

Une expression ou une colonne peut être baptisée avec AS.

Cet alias est utilisé comme nom de colonne et peut donc être nommé dans les clauses des requêtes.

Il peut servir comme raccourcie à un nom de table (utile pour les jointures)

Ces Alias fonctionnent avec ORDER BY, GROUP BY, HAVING mais pas WHERE

Exemple :

```
SELECT
    p.nom AS parent,
    e.nom AS enfant,
    MIN((TO_DAYS(NOW())-TO_DAYS(e.date_naissance))/365) AS agemini
FROM
    personne AS p
LEFT JOIN
    personne AS e
ON
    p.nom=e.parent WHERE e.nom IS NOT NULL
GROUP BY
    parent HAVING agemini > 50 ORDER BY p.date_naissance;
```

LA VALEUR NULL

SQL possède une valeur pour représenter l'absence de valeur : NULL.

Il peut être assigné à des colonnes TEXT, INTEGER ou autres.

Attention, une colonne déclarée NOT NULL ne pourra pas en contenir.

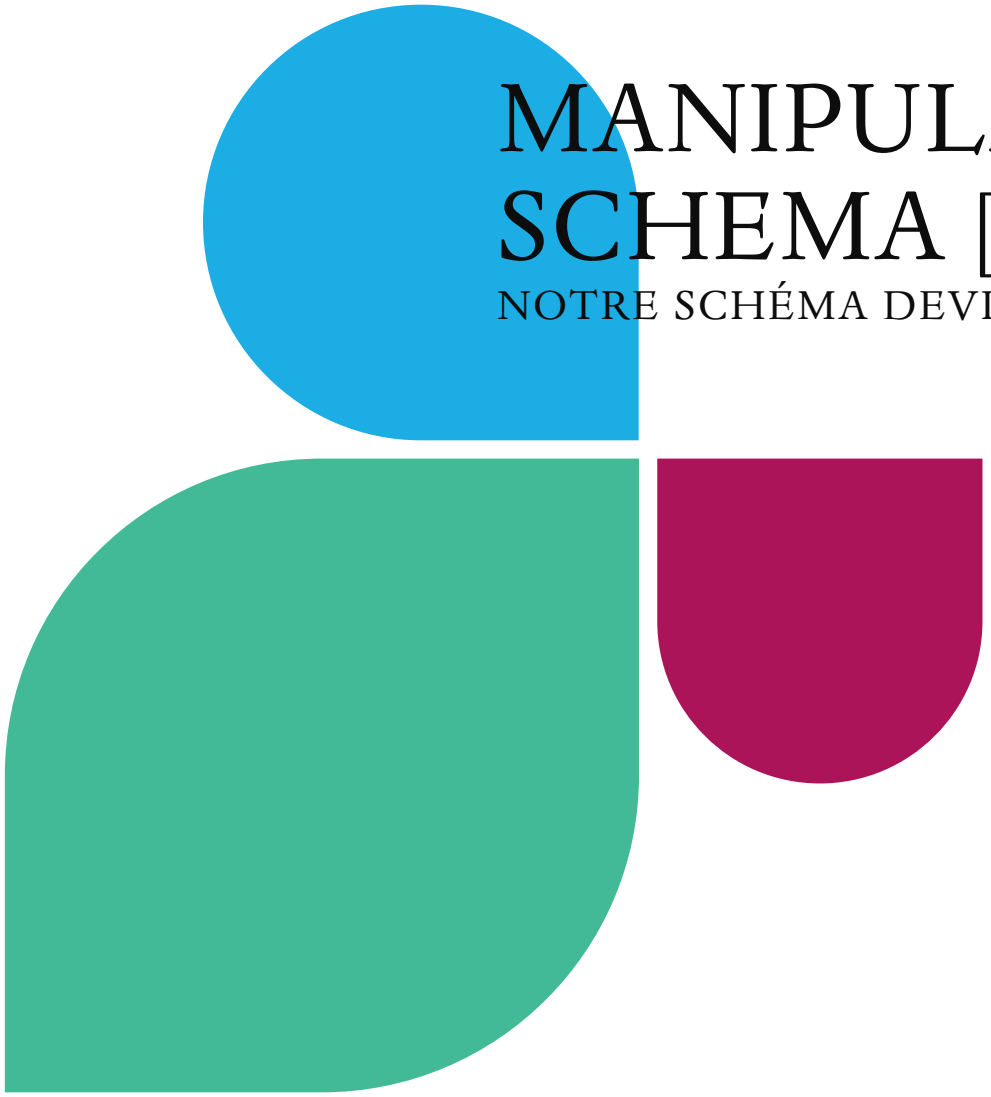
NULL ne doit pas être entouré d'apostrophes ou de guillemets, ou bien il désignera une chaine de caractères.

Il faudra donc dans votre futur codage prendre en compte le renvoi de la valeur NULL dans les résultats de vos requêtes.

```
INSERT into Singer
       (F_Name, L_Name, Birth_place, Language)
       values
       ("", "Homer", NULL, "Greek"),
       ("", "Sting", NULL, "English"),
       ("Jonny", "Five", NULL, "Binary");
```

MANIPULATION DE SCHEMA [BASE]

NOTRE SCHÉMA DEVIENT UNE BASE DE DONNÉES



CRÉATION, SUPPRESSION, COPIE ET BACKUP

Création et Suppression

- CREATE DATABASE Nom_de_la_base;
  - Permet de créer un nouveau schéma soit une nouvelle base de données.
- DROP DATABASE Nom_de_la_base;
  - Permet de détruire une nouvelle base de données.

Copie et Backup

- mysqldump
  - Peut sauvegarder les bases. Il suffit de réinjecter son résultat dans une autre base

Commande exécuter en console en étant déconnecter de la base de donnée [backup + restauration] :

```
mysqldump -u user -p pass nom_de_la_base > sauvegarde.sql

mysql nom_base < chemin_fichier_de_sauvegarde.sql
```

Depuis MySQL Workbench ou PHP MyAdmin, il suffit d'utiliser la fonction d'export où vous pouvez dés lors sauvegarder la base avec ou sans les données.

UTILISATION ET WARNINGS

Utilisation d'une base

Il faut penser avant tout de s'assurer que vous êtes sur le bon schéma :

```
USE nom_de_la_base;
```

Dans MySQL Worbench, un clic droit sur le schéma, puis set as default schema permet de sélectionner le schéma sur lequel on souhaite travailler.

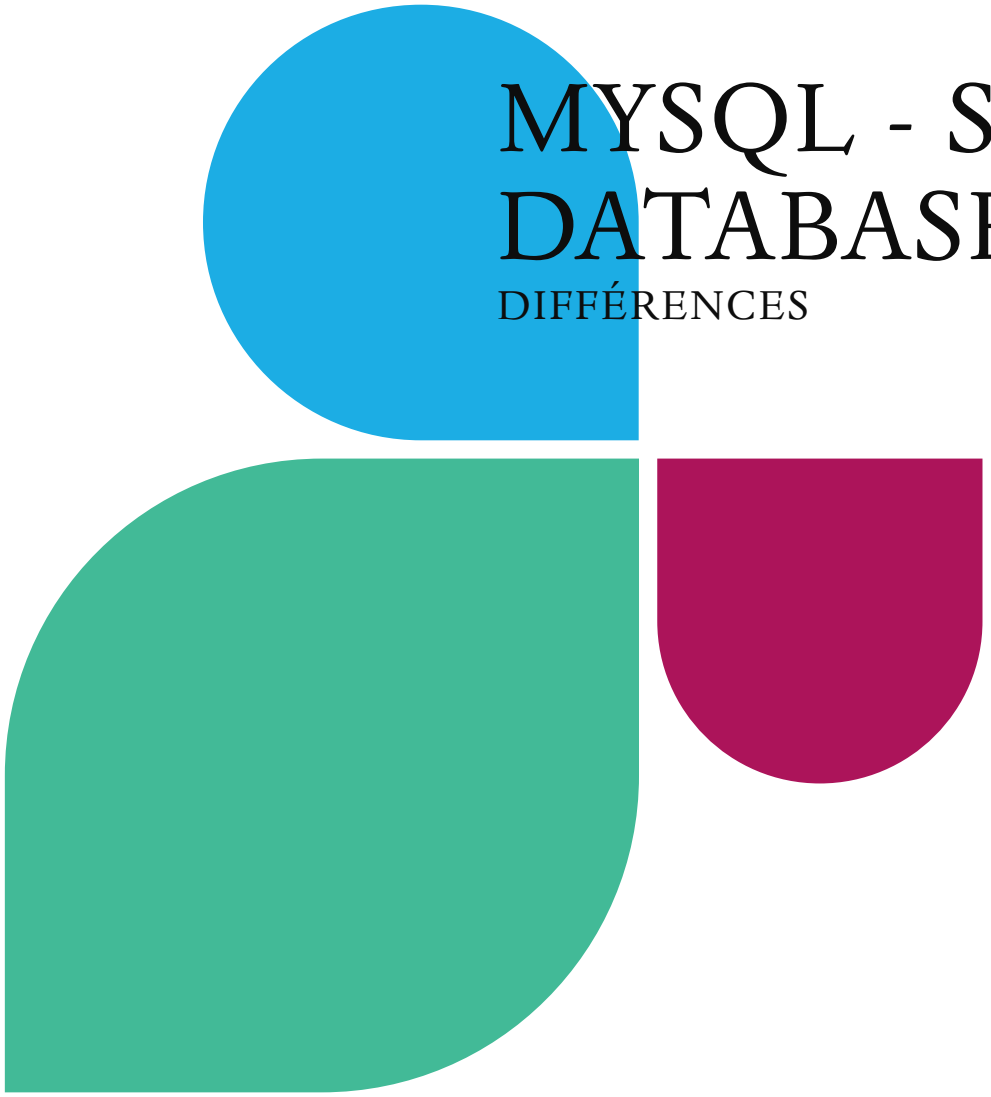
Affichage des warnings

Que ce soit sur Workbench ou en ligne de commandes, on a la possibilité d'afficher les warnings :

- `SHOW WARNINGS;`

MYSQL - SCHÉMA ET DATABASE

DIFFÉRENCES



SCHEMA VS DATABASE

La différence fondamentale entre DATABASE et SCHEMA est que la base de données est manipulée régulièrement tandis que schéma n'est pas modifié fréquemment.

- SCHEMA est la définition structurelle de la base de données tandis que la DATABASE est la collection de données organisées et interdépendantes.
- DATABASE contient le schéma et les enregistrements des tables, mais SCHEMA inclut les tables, le nom de l'attribut, le type d'attribut, les contraintes, etc.
- L'instruction DDL (Data Definition Language) est utilisée pour générer et modifier le schéma tandis que DML (Data Manipulation Language) est utilisé pour la manipulation des données dans la database.
- Le schema n'utilise pas de mémoire à des fins de stockage, alors que la database le fait.

SCHEMA CONTRE DATABASE

MYSQL

Dans MySQL, DATABSE et SCHEMA peuvent être utilisés de manière interchangeable.

Vous pouvez utiliser SCHEMA au lieu de DATABASE et vice versa lors de l'écriture de requêtes SQL dans MySQL.

Voir l'exemple suivant - les deux requêtes créeront une database.

- CREATE DATABASE database_name_one;
- CREATE SCHEMA database_name_two;

Oracle - PostgreSQL

Un schéma contient un groupe de tables.

Une base de données contient un groupe de schémas.

MYSQL- MANIPULATION DE TABLES

NOS ENTITÉS DEVENUES TABLES

CRÉATION

Lorsqu'on crée une table, on doit l'identifier par son nom et définir sa structure soit les colonnes qui la composent.

La clause PRIMARY KEY est utilisée pour identifier la clé primaire de la table.

- obligatoires et uniques.

Cette contrainte permet de s'assurer du respect de ces caractéristiques. À noter qu'il y a toujours une seule clé primaire par table.

L'expression NOT NULL signifie qu'à l'ajout d'une nouvelle ligne dans la table, la colonne doit obligatoirement posséder une valeur. Le terme NULL dans cette colonne ne sera pas tolérée.

```
CREATE TABLE [IF NOT EXISTS] Nom_table (
    colonne1 description_colonne1,
    [colonne2 description_colonne2,
    colonne3 description_colonne3,
    ...,]
    [PRIMARY KEY (colonne_clé_primaire)]
)
[ENGINE=moteur];
// MyISAM = moteur par défaut

exemple :

CREATE TABLE Animal (
    id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
    espece VARCHAR(40) NOT NULL,
    sexe CHAR(1),
    date_naissance DATETIME NOT NULL,
    nom VARCHAR(30),
    commentaires TEXT,
    PRIMARY KEY (id)
)
ENGINE=MyISAM;
```

LES CONTRAINTES

Les contraintes sont les règles appliquées aux colonnes de données d'une table.

Celles-ci sont utilisées pour limiter le type de données pouvant aller dans une table. Cela garantit l'exactitude et la fiabilité des données de la base de données.

Les contraintes peuvent être au niveau de la colonne ou de la table.

NOT NULL

- Ne peut être null

CHECK

- Vérifie la valeur saisie dans un enregistrement

DEFAULT

- Prend la valeur par défaut si non fournie par INSERT

UNIQUE

- Empêche deux enregistrements identiques dans une colonne

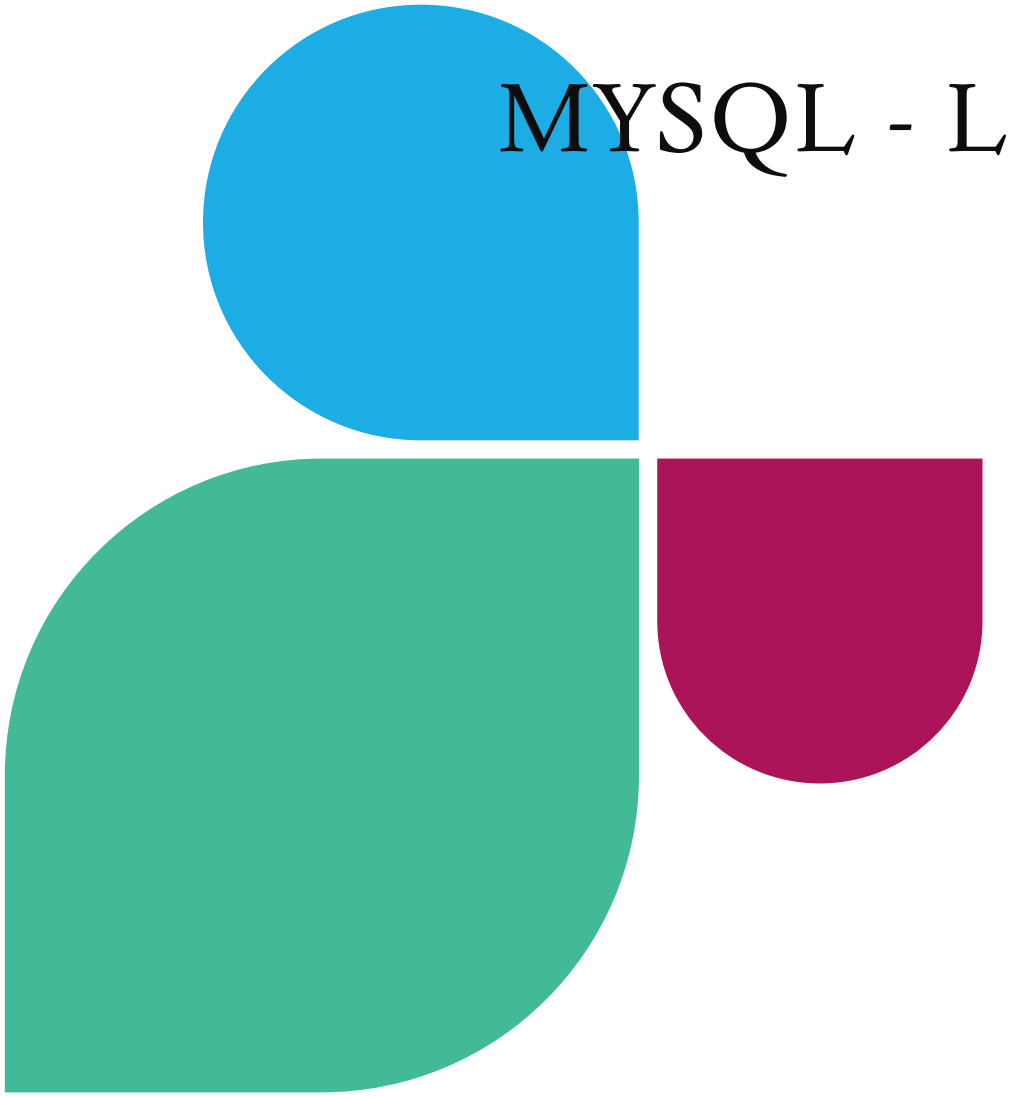
PRIMARY KEY

- La clé primaire

FOREIGN KEY

- La clé étrangère

MYSQL - LES VUES



LES VUES

Les vues sont des objets de la base de données, constitués d'**un nom** et d'une **requête de sélection**.

La requête SELECT stockée dans une vue :

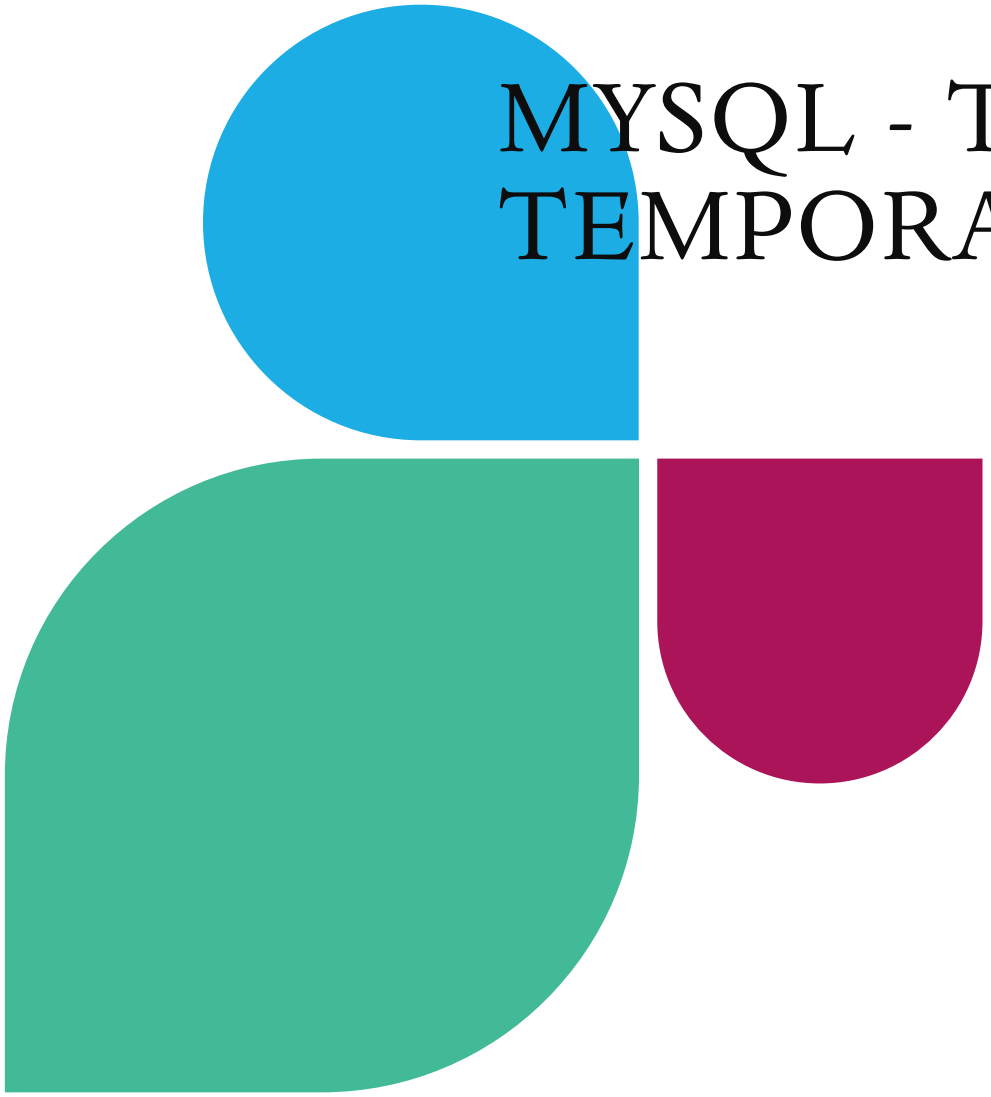
- peut utiliser des jointures, des clauses WHERE, GROUP BY, des fonctions (scalaires ou d'agrégation), etc.
- L'utilisation de DISTINCT et LIMIT est cependant déconseillée.

On peut sélectionner les données à partir d'une vue de la même manière qu'on le fait à partir d'une table. On peut donc utiliser des jointures, des fonctions, des GROUP BY, des LIMIT…

Les vues permettent de simplifier les requêtes, de créer une interface entre l'application et la base de données, et/ou de restreindre finement l'accès en lecture des données aux utilisateurs.

```sql
-- Syntaxe
CREATE [OR REPLACE] VIEW nom_vue
AS requete_select;

-- Creation d'une vue V_Chien_Race
CREATE OR REPLACE VIEW V_Chien_race
AS SELECT id, sexe, date_naissance, nom, commentaires, espece_id, race_id,
mere_id, pere_id, disponible
    FROM Chien
    WHERE race_id IS NOT NULL;

-- Consultation
SELECT * FROM V_Chien_Race;
```

MYSQL - TABLES TEMPORAIRES



TABLES TEMPORAIRES

Principe

Une table temporaire :

- n'existe que pour la session dans laquelle elle a été créée. Dès que la session se termine, les tables temporaires sont supprimées.
- créée de la même manière qu'une table normale. Il suffit d'ajouter le mot-clé TEMPORARY avant TABLE.
- créer une table (temporaire ou non) à partir de la structure d'une autre table avec CREATE [TEMPORARY] TABLE nouvelle_table LIKE ancienne_table;
- créer une table à partir d'une requête SELECT avec CREATE [TEMPORARY] TABLE SELECT ...;

Objectifs

1. Elles permettent de gagner en performance lorsque dans une session, on doit exécuter plusieurs requêtes sur un même set de données.
2. On peut les utiliser pour créer des données de test.
3. On peut les utiliser pour stocker un set de résultats d'une procédure stockée.

COPIE ET MODIFICATION

Copier une table

Pour obtenir la même structure (noms et types des champs, index, mais aucun enregistrement) puis dupliquer le contenu :

```sql
CREATE TABLE `new1` LIKE `old1`;

INSERT INTO `new1` SELECT * FROM `old1`;
```

```sql
ALTER TABLE Test_tuto
    MODIFY prenom nom VARCHAR(30) NOT NULL;
-- Changement du type + changement du nom

ALTER TABLE Test_tuto
    MODIFY id id BIGINT NOT NULL;
-- Changement du type sans renommer

ALTER TABLE Test_tuto
    MODIFY id BIGINT NOT NULL AUTO_INCREMENT;
-- Ajout de l'auto-incrémentation

ALTER TABLE Test_tuto
    MODIFY nom VARCHAR(30) NOT NULL DEFAULT 'Blabla';
-- Changement de la description (même type mais ajout d'une valeur par défaut)
```

ALTER TABLE

ALTER TABLE nom_table ADD ...

- permet d'ajouter quelque chose (une colonne par exemple)

ALTER TABLE nom_table DROP ...

- permet de retirer quelque chose

ALTER TABLE nom_table CHANGE

- Peux modifier le nom, la définition (ou les deux) d'une colonne

ALTER TABLE nom_table MODIFY

- Peut **seulement** modifier la définition de la colonne.

RENOMMER ET SUPPRESSION

RENAME

Pour renommer une table, il faut préalablement retirer ses privilèges avec ALTER TABLE (DROP), puis ALTER TABLE (ADD) pour remettre les privilèges à attribuer à la nouvelle table.

Rename ne peut pas renommer les tables temporaires.

```
Renommage :

ALTER TABLE `old` RENAME `new`

Raccourci :

RENAME TABLE `old_name` TO `new_name`
```

DROP

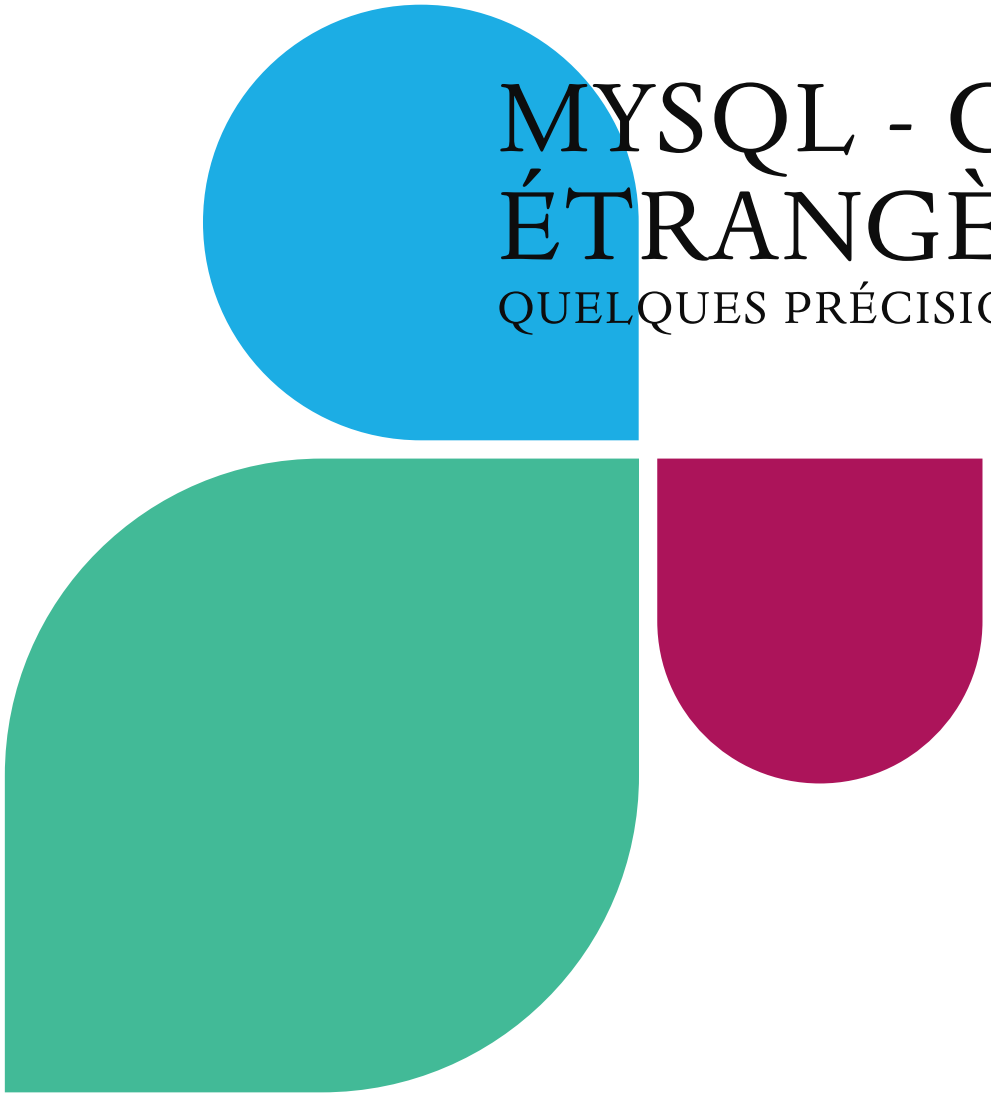
Pour supprimer une table (enregistrements et structure), il faut "drop"

En option, on peut y ajouter une vérification d'existence.

```
Plusieurs :

DROP TABLE `table1`, `table2`, ... ;

Avec vérification :

DROP TABLE `table` IF EXISTS;
```

MYSQL - CLÉS ÉTRANGÈRES

QUELQUES PRÉCISIONS



PRINCIPE ET SYNTAXE

Les clés étrangères permettent de gérer des relations entre plusieurs tables, et garantissent la cohérence des données.

```
CREATE TABLE [IF NOT EXISTS] Nom_table (
    colonne1 description_colonne1,
    [colonne2 description_colonne2,
    colonne3 description_colonne3,
    ...,]
    [ [CONSTRAINT [symbole_contrainte] ]
      FOREIGN KEY (colonne(s)_clé_étrangère)
      REFERENCES table_référence (colonne(s)_référence)]
);

CREATE TABLE Commande (
    numero INT UNSIGNED PRIMARY KEY AUTO_INCREMENT,
    client INT UNSIGNED NOT NULL,
    produit VARCHAR(40),
    quantite SMALLINT DEFAULT 1,
    CONSTRAINT fk_client_numero          -- On donne un nom à notre clé
        FOREIGN KEY (client)             -- Colonne sur laquelle on crée la clé
        REFERENCES Client(numero)        -- Colonne de référence
);

-- on peut utiliser ALTER Table pour modifier un table
ALTER TABLE Commande
   ADD CONSTRAINT fk_client_numero
   FOREIGN KEY (client)
   REFERENCES Client(numero);

ALTER TABLE nom_table
   DROP FOREIGN KEY symbole_contrainte;
```

ON DELETE ET ON UPDATE

Quand on crée une clé étrangères, il existe deux options à mettre en place afin de gérer les suppressions et les mises à jours :

- ON DELETE : comportement que devra avoir le SGBD qui va supprimer un enregistrement qui est référencé dans une autre table.
- ON UPDATE : même chose mais dans le cas d'une mise à jour qui est référencé

Ces deux options acceptent un paramètre à choisir parmi 4 ci-dessous :

- RESTRICT ou NO ACTION * : ne va rien faire. C'est à vous de faire en sorte de respecter la contrainte d'intégrité

** Propre à MySQL donc attention si vous utilisez un autre SGBD*

- SET NULL : la clé étrangère reçoit la valeur NULL. Peut être utile dans le cas d'un DELETE.
- CASCADE : Mise à jour en cascade. *Attention : Il mettra à jour / supprimera automatiquement les enregistrements qui référencent l'enregistrement qui a été modifié / supprimé*

```
ALTER TABLE REPERTOIRE
  ADD CONSTRAINT FK_REPERTOIRE FOREIGN KEY (GRO_ID)
  REFERENCES GROUPE (GRO_ID) ON DELETE restrict ON UPDATE restrict;
```