



**Sommaire**

[TYPES DES ATTRIBUTS 2](#_heading=h.30j0zll)

[1.1.1. TYPES DES ATTRIBUTS : ENTIERS 2](#_heading=h.1fob9te)

[1.1.2. TYPES DES ATTRIBUTS : REELS 2](#_heading=h.3znysh7)

[TYPES DES ATTRIBUTS : Chaine de caracteres 3](#_heading=h.2et92p0)

[1.1.3. TYPES DES ATTRIBUTS : Dates et heures 3](#_heading=h.tyjcwt)

[1.1.4. TYPES DES ATTRIBUTS : ENUMERATIONS 3](#_heading=h.3dy6vkm)

[1.1.5. TYPES DES ATTRIBUTS : ENSEMBLES 4](#_heading=h.1t3h5sf)

[1. Interrogation d’une BD 4](#_heading=h.4d34og8)

[Sélection de lignes ou restriction 5](#_heading=h.17dp8vu)

[Classer le RÉSULTAT d'une interrogation 6](#_heading=h.35nkun2)

[LES FONCTIONS DE CALCUL OU DE GROUPE 6](#_heading=h.1ksv4uv)

[Sélection des groupes 7](#_heading=h.2jxsxqh)

[Les sous interrogations (REQUÊTES IMBRIQUÉES) 7](#_heading=h.z337ya)

[LES VUES 8](#_heading=h.3j2qqm3)

[Droits d'accès aux tables 9](#_heading=h.1y810tw)

[SQL : CRÉATION et suppression d’une table 10](#_heading=h.4i7ojhp)

[1. Création (Définition) de table 10](#_heading=h.2xcytpi)

[2. Suppression de table 11](#_heading=h.1ci93xb)

[3. Modification de la DÉFINITION : Ajout et retrait d’une colonne de table 11](#_heading=h.3whwml4)

[SQL : Modifier une base 11](#_heading=h.2bn6wsx)

MySQL

* **Système de Gestion de Bases de Données Relationnelles** (abrégé SGBDR), c'est-à-dire un logiciel qui permet de gérer des bases de données.
* Il utilise **le langage SQL**.
* SGBDR 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.

Syntaxe de MySQL

# TYPES DES ATTRIBUTS

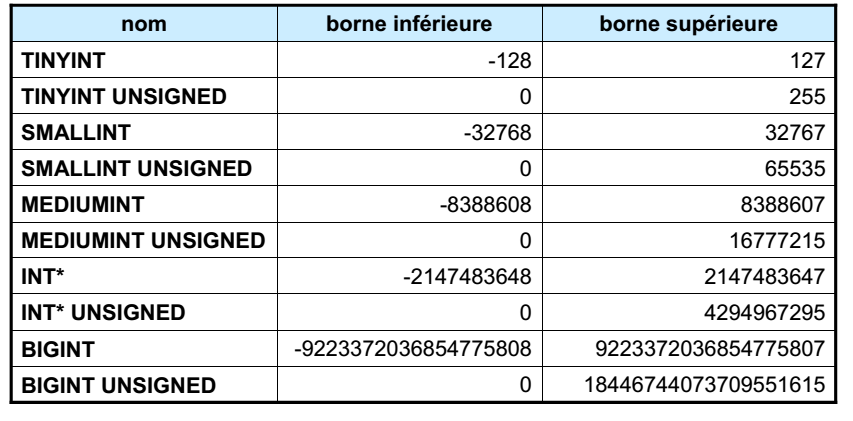
Les propriétés de vos objets peuvent être de types très différents :

* **Nombre entier** signé ou non (température, quantité commandée, âge)
* **Nombre à virgule** (prix, taille)
* **Chaîne de caractères** (nom, adresse, article de presse)
* **Date et heure** (date de naissance, heure de parution)
* **Énumération** (une couleur parmi une liste prédéfinie)
* **Ensemble** (une ou des monnaies parmi une liste prédéfinie)

Il s’agit de choisir le plus adapté à vos besoins.

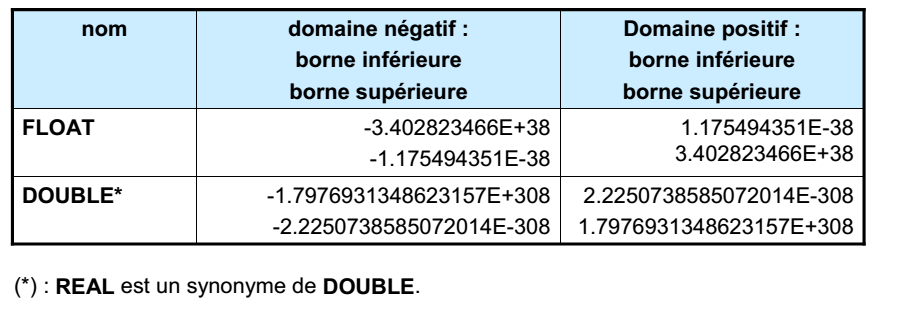
Ces types vont occuper une plus ou moins grande quantité de mémoire, il faut choisir le type le plus approprié à la nature des données à stocker.

## TYPES DES ATTRIBUTS : ENTIERS

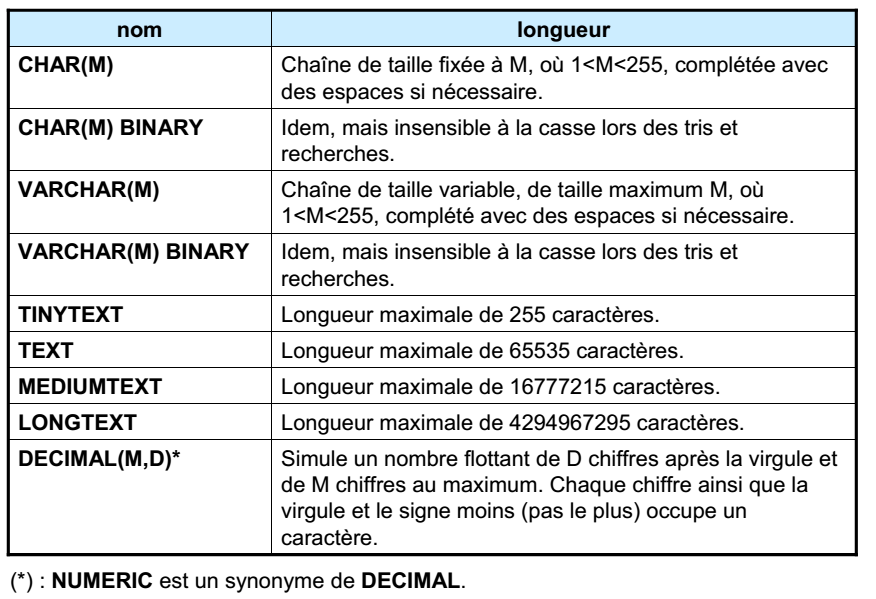


(\*) :**INTEGER**est un synonyme de**INT**.  
**UNSIGNED**permet d’avoir un type non signé.  
**ZEROFILL**: remplissage des zéros non significatifs.

* + 1. ***TYPES DES ATTRIBUTS :*** *REELS*



## TYPES DES ATTRIBUTS : Chaine de caractères

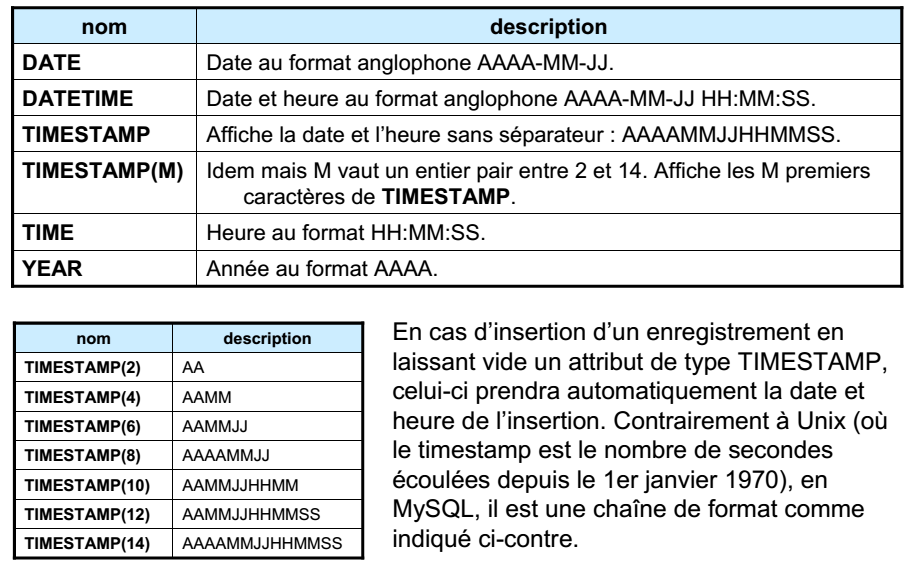


- Les types TINYTEXT, TEXT, MEDIUMTEXT et LONGTEXT peuvent êtrejudicieusement remplacés respectivement par TINYBLOB, BLOB, MEDIUMBLOB et LONGBLOB.

- Ils ne diffèrent que par la sensibilité à la casse qui caractérise la famille desBLOB. Alors que la famille des TEXT sont insensibles à la casse lors des tris etrecherches.

- Les BLOB peuvent être utilisés pour stocker des données binaires.Les VARCHAR, TEXT et BLOB sont de taille variable. Alors que les CHAR etDECIMAL sont de taille fixe.

* + 1. ***TYPES DES ATTRIBUTS :*** *Dates et heures*



* + 1. ***TYPES DES ATTRIBUTS :*** *ENUMERATIONS*

Un attribut de type **ENUM** peut prendre une valeur parmi celles définies lors dela création de la table plus la chaîne vide ainsi que **NULL** si la définition le permet.

Ces valeurs sont exclusivement des chaînes de caractères. Uneénumération peut contenir 65535 éléments au maximum.

Définition d’un tel attribut :  
nom\_attribut ENUM(‘’valeur 1’’,’’valeur 2’’…) NOT NULL  
nom\_attribut ENUM(‘’valeur 1’’,’’valeur 2’’…) NULL

* + 1. ***TYPES DES ATTRIBUTS :*** *ENSEMBLES*

Un attribut de type**SET**peut prendre pour valeur la chaîne vide,**NULL**ou unechaîne contenant une liste de valeurs qui doivent être déclarées lors de ladéfinition de l’attribut lors de la création de la relation.

Par exemple, un attribut déclaré comme ci :**SET(‘‘voiture’’, ‘’moto’’, ’’vélo’’) NOT NULL**  
peut prendre les valeurs suivantes :**‘’’’** (chaîne vide), **‘’voiture,moto’’, ‘’vélo,voiture,moto’’**  
et toutes autres combinaisons de listes des trois valeurs définie plus haut.

Un attribut déclaré comme suit :**SET(‘‘voiture’’, ‘’moto’’, ’’vélo’’) NULL**  
peut prendre, en plus ce celles précédentes, la valeur**NULL**.

Il ne peut être défini que 64 éléments maximum.

**Le Langage SQL**

Le SQL est un langage normalisé haut niveau de dialogue avec un SGBD relationnel (Système de Gestion de Base de données). Il a été inventé par IBM.

IL permet de :

* Créer une base de données (CREATE TABLE…)
* Modifier une base de données (INSERT, UPDATE, DELETE,…)
* Interroger une BD (**SELECT**…)
* Créer des vues sur une base de données (CREATE VIEW…)
* Spécifier des droits d’accès à une BD (GRANT…)
* D’optimiser les performances d’accès aux données (INDEX…)
* De définir des contraintes d’intégrité (COMMIT…)

Au travers d’exemple, nous allons voir les principales possibilités de recherche d’informations dans une BD.

# Interrogation d’une BD

La clause **SELECT** permet d’interroger une BD. C’est l’application sur la BD des opérateurs de l’algèbre relationnelle : projection, restriction et jointure.

Le résultat d’une requête SELECT est une table temporaire avec les champs demandés.

**Syntaxe :**

**SELECT** <clause d’unicité: DISTINCT><liste de champs>

**FROM** <liste de tables>

**WHERE** <critère de sélection des lignes>

**GROUP BY** <liste de champs>

**HAVING** <critère de sélection des groupes >

**ORDER BY** <Liste de champs><critère d’ordre>

### Sélection de lignes ou restriction

La clause **WHERE** permet de spécifier quelles sont les lignes à sélectionner. Elle est suivie d'un prédicat qui sera évalué pour chaque ligne de la table. Les lignes pour lesquelles le prédicat est vrai seront sélectionnées.

La syntaxe est la suivante

**SELECT \***

**FROM nom\_table**

**WHERE predicat** ;

Les prédicats, qu'ils soient simples ou composés, sont constitués à partir d'expressions que l'on compare entre elles.

Les expressions peuvent être de trois types : numérique, chaîne de caractères ou date. A chacun de ces types correspond un format de constante :

Constante numérique

Nombre contenant éventuellement un signe, un point décimal et une puissance de dix. Ex : -10, 2.5, 1.2 E-10

Constante chaîne de caractères

Une chaîne de caractères entre apostrophes. Ex :'MARTIN' (attention, une lettre en majuscules n'est pas considérée comme égale à la même lettre en minuscule).

Constante date

Une chaîne de caractères entre apostrophes au format suivant : jour-mois-année où le jour est sur deux chiffres, le mois est désigné par les trois premières lettres de son nom en anglais, l'année est sur deux chiffres. Ex : '01-FEB-85'

#### Prédicat simple

Un prédicat simple est le résultat de la comparaison de deux expressions au moyen d'un opérateur de comparaison qui peut être :

=   égale

!= différent

<  inférieur

<= inférieur ou égal

>  supérieur

>= supérieur ou égal

Il faut ajouter à ces opérateurs arithmétiques classiques les opérateurs suivants :

expr1 BETWEEN expr2 AND expr3

vrai si expr1 est compris entre expr2 et expr3, bornes incluses

expr1 IN (expr2, expr3, ...)

vrai si expr1 est égale à l'une des expressions de la liste entre parenthèses

expr LIKE chaine

oùchaine est une chaîne de caractères pouvant contenir l'un des caractères jokers :

\_   remplace exactement 1 caractère

% remplace une chaîne de caractères de longueur quelconque, y compris de longueur nulle.

#### Prédicats COMPOSES

Les opérateurs logiques AND (et) et OR(ou inclusif) peuvent être utilisés pour combiner entre eux plusieurs prédicats. L'opérateur NOT placé devant un prédicat en inverse le sens.

L'opérateur AND est prioritaire par rapport à l'opérateur OR. Des parenthèses peuvent être utilisées pour imposer une priorité dans l'évaluation du prédicat, ou simplement pour rendre plus claire l'expression logique.

*Exemple : Quels sont les romans dont le prix est supérieur à 80 ?*

*SELECT titre*

*FROM Livre*

*WHERE (prix>80) AND (genre = ‘roman’)*

#### Valeurs NULL

Pour sql, une valeur NULL est une valeur non définie. Il est possible d'ajouter une ligne à une table sans spécifier de valeur pour les colonnes non obligatoires : ces colonnes absentes auront la valeur NULL.

Par exemple, les employés dont la rémunération ne prend pas en compte de commission auront une valeur NULL, c'est-à-dire indéfinie, comme commission.

L'opérateur IS NULL permet de tester la valeur NULL : le prédicat expr IS NULL est vrai si l'expression a la valeur NULL (c'est-à-dire si elle est indéfinie).

**Exemple :** Quels sont les livres en cours d’emprunt ?

*SELECT numlivre*

*FROM Livre*

*WHERE retour IS NULL;*

### Classer le RÉSULTAT d'une interrogation

Les lignes constituant le résultat d'un [SELECT](about:blank) sont obtenues dans un ordre indéterminé. On peut, dans un [SELECT](about:blank), demander que le résultat soit classé dans un ordre ascendant ou descendant, en fonction du contenu d'une ou plusieurs colonnes (jusqu'à 16 critères de classement possibles). Les critères de classement sont spécifiés dans une clause ORDER BY dont la syntaxe est la suivante :

ORDER BY {nom\_col1 | num\_col1 [DESC] [, nom\_col2 | num\_col2 [DESC],...]}

### LES FONCTIONS DE CALCUL OU DE GROUPE

**AVG** ([DISTINCT | ALL] expression)

Renvoie la **moyenne** des valeurs d' expression.

**COUNT** (\* | [DISTINCT | ALL] expression)

renvoie**le nombre de lignes du résultat** de la requête. Si expression est présent, on ne compte que les lignes pour lesquelles cette expression n'est pas NULL.

**MAX** ([DISTINCT | ALL] expression)

Renvoie **la plus petite des valeurs** d'expression.

**MIN** ([DISTINCT | ALL] expression)

Renvoie **la plus grande des valeurs** d'expression.

**SUM** ([DISTINCT | ALL] expression)

Renvoie **la somme** des valeurs

**DISTINCT**

Indique à la fonction de groupe de ne prendre en compte que des valeurs distinctes.

**ALL**

Indique à la fonction de groupe de prendre en compte toutes les valeurs, c'est la valeur par défaut.

**Exemple 1**: Afficher le prix moyen des livres

*SELECT AVG(Prix)*

*FROM Livre*

**Exemple 2** : Afficher pour chaque auteur le prix moyen de ses livres

*SELECT Auteur, AVG(Prix)*

*FROM Livre*

*GROUP BY Auteur*

### Sélection des groupes

De la même façon qu'il est possible de sélectionner certaines lignes au moyen de la clause WHERE, il est possible dans une requête comportant une fonction de groupe de sélectionner par la clause **HAVING** des groupes qui satisfont certaines conditions.

**Exemple**: Afficher le nombre de livres de tous les auteurs qui ont écrit plus de 2 livres

*SELECT Auteur, COUNT (\*)*

*FROM Livre*

*GROUP BY Auteur*

***HAVIN****G COUNT(\*) > 2*

## Les sous interrogations (REQUÊTES IMBRIQUÉES)

Une caractéristique puissante de SQL est la possibilité qu'un critère de recherche employé dans une clause WHERE (expression à droite d'un opérateur de comparaison) soit lui-même le résultat d'un SELECT ; c'est ce qu'on appelle une sous-interrogation.

#### Sous interrogation ramenant une seule valeur

**Exemple :** Quelles sont les livres dont le prix est supérieur au prix moyen?

SELECT \*

FROM Livre

WHERE Prix > (SELECT AVG(prix)

FROM Livre)

#### Sous interrogation ramenant plusieurs lignes

Une sous interrogation peut ramener plusieurs lignes à condition que l'opérateur de comparaison admette à sa droite un ensemble de valeurs. Les opérateurs permettant de comparer une valeur à un ensemble de valeurs sont :

* l'opérateurIN
* les opérateurs obtenus en ajoutant ANY ou ALL à la suite d'un opérateur de comparaison classique (=, !=, >, >=, <, <=)
  + ANY: la comparaison est vraie si elle est vraie pour au moins un des éléments de l'ensemble.
  + ALL: la comparaison sera vraie si elle est vraie pour tous les éléments de l'ensemble.

**Exemple :** Quelles sont les personnes qui habitent dans la même ville que «Durand »

*SELECT nom, Prenom*

*FROM Personne*

*WHERE Ville IN (SELECT Ville*

*FROM Personne*

*WHERE Nom = ‘Durand’);*

**Exemple** : Afficher tous les romans dont le prix est inférieur à tous les livres de Zola

*SELECT titre*

*FROM Livre*

*WHERE Prix < ALL (SELECT Prix*

*FROM Livre*

*WHERE Auteur = ‘Zola’);*

#### Sous interrogation ramenant plusieurs colonnes

Il est possible de comparer le résultat d'un [SELECT](about:blank) ramenant plusieurs colonnes à une liste de colonnes. La liste de colonnes figurera entre parenthèses à gauche de l'opérateur de comparaison.

**Exemple :** Quelles sont les personnes qui habitent la même ville que ‘Durand’ et possèdent le même prénom ?

*SELECT nom, Prenom*

*FROM Personne*

*WHERE (Ville,Prenom) = (SELECT Ville, Prenom*

*FROM Personne*

*WHERE Nom = ‘Durand’);*

# LES VUES

**CREATE VIEW** : permet de créer une vue, soit **une table virtuelle produite par une requête de sélection**. Une vue est constituée par une requête SELECT prédéfinie appliquée à une ou plusieurs tables d’une base de données.

Les vues peuvent servir à recueillir des données régulièrement consultées ou mises à jour.

Les vues sont souvent employées pour des raisons de sécurité. En effet, il devient possible de ne montrer qu’une partie des données et ainsi cacher les informations confidentielles.

Le créateur de la vue en détient la propriété et des privilèges, il peut accorder des droits aux différents utilisateurs de la base de données.

#### Créer une vue

La commande CREATE VIEW permet de créer une vue en spécifiant le [SELECT](about:blank) constituant la définition de la vue :

CREATE VIEW nom\_vue [(nom\_col1,...)]

AS SELECT ...

WITH CHECK OPTION ;

La spécification des noms de colonnes de la vue est facultative. Par défaut, les noms des colonnes de la vue sont les mêmes que les noms des colonnes résultantes du [SELECT](about:blank) (si certaines colonnes résultat du [SELECT](about:blank) sont des expressions, il faut renommer ces colonnes dans le [SELECT](about:blank), ou spécifier les noms de colonne de la vue).

Une fois créée, une vue s'utilise comme une table. Il n'y a pas de duplication des informations mais stockage de la définition de la vue.

**Exemple :** Création d'une vue constituant une restriction de la table emp aux employés du departement 10.

CREATE VIEW emp10 AS

SELECT \*

FROM emp

WHERE n\_dept = 10 ;

Le CHECK OPTION permet de vérifier que la mise à jour ou l'insertion faite à travers la vue ne produisent que des lignes qui font partie de la sélection de la vue.

Ainsi donc, si la vue emp10 a été créée avec CHECK OPTION on ne pourra à travers cette vue ni modifier, ni insérer des employés ne faisant pas partie du département 10.

Il est possible d'effectuer des [INSERT](about:blank) et des [UPDATE](about:blank) à travers des vues, sous deux conditions :

* le[SELECT](about:blank) définissant la vue ne doit pas comporter de jointure,
* les colonnes résultat du [SELECT](about:blank) doivent être des colonnes réelles et non pas des expressions.

**Exemple :** Modification des salaires du département 10 à travers la vue emp10.

UPDATE emp10

SET sal = sal \*1.1;

Toutes les lignes de la table emp, telles que le contenu de la colonne n\_dept est égal à 10 seront modifiées.

#### Supprimer une vue

Une vue peut être détruite par la commande :

DROP VIEWnom\_vue;

#### Renommer une vue

On peut renommer une vue par la commande :

RENAME ancien\_nom TO nouveau\_nom;

## Droits d'accès aux tables

La protection des objets d'une base de données est décentralisée : c'est le créateur d'un objet qui possède tous les droits de lecture, de modification et de suppression de cet objet.

Les autres utilisateurs n'ont aucun droit d'accès à cet objet, à moins que le créateur ne les leur accorde explicitement par une commande **GRANT**:

Syntaxe : **GRANT privilege ON {nom\_table | nom\_vue} TO nom\_utilisateur**

**WITH GRANT OPTION** ;

Les privilèges pouvant être accordés sont entre autres les suivants : [update(col,...) ]

SELECT = consultation

INSERT = rajout des lignes

UPDATE(col,...) = modification des lignes (peut-être restreint à certaines colonnes)

DELETE = suppression des lignes

ALTER = modification de la définition de la table

ALL = tous les droits ci-dessus

Un utilisateur ayant reçu un privilège avec l'option GRANT peut le transmettre à son tour avec la clause WITH GRANT OPTION).

**Exemple :** L'utilisateur scott peut autoriser l'utilisateur douglas à lire sa table emp.

GRANT SELECT ON emp TO douglas ;

* Les droits peuvent être accordés à tous les utilisateurs en employant le mot réservé PUBLIC à la place du nom d'utilisateur.

GRANT SELECT, UPDATE ON emp TO PUBLIC;

* Un utilisateur ayant accordé un privilège peut le reprendre à l'aide de l'ordre **REVOKE**.

REVOKE PRIVILèGE ON {nom\_table | nom\_vue} FROM nom\_utilisateur;

* Le propriétaire d'une table peut donner des droits de lecture non pas sur la table entière, mais sur une vue basée sur la table. Ainsi, seules les informations faisant partie de la vue seront accessibles.

CREATE VIEW emp1 AS

SELECT nom, fonction, embauche

FROM scott.emp

WHERE n\_dept != 10 ;

GRANT SELECT ON emp1 TO PUBLIC;

Tous les utilisateurs pourront sélectionner les employés à travers la vue emp1 : ils ne verront que les colonnes nom, fonction, embauche de la table emp et n'auront pas accès aux employés du département 10. Avec l'option de contrôle (CHECK OPTION), les vues peuvent servir à réaliser les contrôles lors des mises à jour.

## SQL : CRÉATION et suppression d’une table

La clause SQL CREATE TABLE permet de créer une table de base de données en spécifiant le nom des champs, leur type et les contraintes relatives à ces champs. Il faut aussi spécifier les contraintes relatives à la table (clés primaires, clés étrangères, etc.)

### 1. Création (Définition) de table

**CREATE TABLE**<nom de la table>

**(**  <champ 1><type du champ 1><contraintes sur champ 1>,

<champ 2><type du champ 2><contraintes sur champ 2>,

<champ n><type du champ n><contraintes sur champ n>,

<Contraintes sur la table>,

<Contraintes sur la table>

**)**

#### Les principaux types de DONNÉES admis par SQL sont :

* SMALLINT : entiers signés courts (16 bits),
* INTEGER : entiers signés longs (32 bits)
* CHAR (N), VARCHAR (N) : chaines de caractères fixe et variable de N caractères
* DECIMAL (M,N) : chaine de caractères qui simule un décimal de M chiffres avant la virgule et N aprés
* FLOAT : réels flottants simple précision,
* DOUBLE PRECISION : réel flottants

#### Les contraintes DIFFÉRENTES que l'on peut DÉCLARER sont les suivantes :

* NOT NULL : La colonne ne peut pas contenir de valeurs NULL.
* UNIQUE : Chaque ligne de la table doit avoir une valeur différente ou NULL pour cette (ou ces) colonne.
* PRIMARY KEY : Chaque ligne de la table doit avoir une valeur différente pour cette (ou ces) colonne. les valeurs NULL sont rejetées.
* FOREIGN KEY : Cette colonne fait référence à une colonne clé d'une autre table.
* CHECK : Permet de spécifier les valeurs acceptables pour une colonne.

### 2. Suppression de table

**DROP TABLE**<nom de la table>

### 3. Modification de la DÉFINITION : Ajout et retrait d’une colonne de table

* Ajout :

**ALTER TABLE**<nom de la table>

ADD <champ ><type du champ ><contraintes sur champ 1>

* Suppression

ALTER TABLE <nom de la table>

DROP Genre

## SQL : Modifier une base

Le langage de manipulation de données est le langage permettant de modifier les informations contenues dans une base de données.

L'unité manipulée est la ligne. Il existe trois commandes SQL permettant d'effectuer les trois types de modifications des données : **ajout, modification et suppression.**

**Ajout de lignes**

**INSERT INTO** nom\_table(nom\_col1, nom\_col2, ...)

VALUES (val1, val2...)

**Modification de lignes**

**UPDATE** nom\_table

SET nom\_col1 = {expression1 | ( SELECT ...) },

nom\_col2 = {expression2 | ( SELECT ...) }

nom\_colp = {expressionp | ( SELECT ...) }

WHERE condition

**Suppression de lignes**

**DELETE FROM** nom\_table

WHERE predicat ;