Objectifs :

Etre capable de comprendre comment fonctionne une BDD MariaDB.

Etre capable de configurer une BDD MariaDB.

Matériel à avoir :

Une machine Debian 10 sur laquelle vous avez préalablement installé LAMP, ainsi qu’une machine cliente Windows pour accéder à la base de données par l’interface de Adminer.

Consignes :

Dans un premier temps, vous allez rechercher des commandes MariaDB sur internet, pour connaitre leur fonctionnement et leur syntaxe. Ensuite, dans un second document, d’exercice, en suivant les instructions, vous utiliserez ces commandes pour mettre en place une base de données simple afin de tester réellement les commandes et leurs syntaxes.

Table des matières

[Se connecter à MariaDB 3](#_Toc57123117)

[Gestion de la BDD en elle-même 3](#_Toc57123118)

[Visualiser le nom des BDD existantes 3](#_Toc57123119)

[Créer une base de données 3](#_Toc57123120)

[Créer un utilisateur administrateur de cette base de données 3](#_Toc57123121)

[Utiliser une base de données 4](#_Toc57123122)

[Supprimer une BDD entière 4](#_Toc57123123)

[Gestion de la structure des tables 4](#_Toc57123124)

[Visualiser le nom des tables existantes dans la BDD 4](#_Toc57123125)

[Visualiser la structure d’une table existante dans la BDD 4](#_Toc57123126)

[Créer une table avec ses colonnes et leurs caractéristiques 5](#_Toc57123127)

[Modifier une table (rajouter, supprimer et modifier une colonne) 6](#_Toc57123128)

[Rajouter une colonne dans une table 6](#_Toc57123129)

[Modifier une colonne dans une table 6](#_Toc57123130)

[Supprimer une colonne dans une table 7](#_Toc57123131)

[Supprimer une table entière 7](#_Toc57123132)

[Gestion des données contenues dans une table 8](#_Toc57123133)

[Visualiser les enregistrements présents dans une table 8](#_Toc57123134)

[Visualiser tous les enregistrements dans une table avec toutes les colonnes 8](#_Toc57123135)

[Visualiser certains enregistrements dans une table avec toutes les colonnes 9](#_Toc57123136)

[Visualiser tous les enregistrements dans une table avec des colonnes spécifiques 9](#_Toc57123137)

[Insérer une ligne de données (un enregistrement) dans une table avec toutes les valeurs 10](#_Toc57123138)

[Insérer une ligne de données (un enregistrement) dans une table en spécifiant uniquement les valeurs à remplir 11](#_Toc57123139)

[Insérer plusieurs lignes dans une table en une seule commande 11](#_Toc57123140)

[Modifier une donnée spécifique dans une ligne spécifique 12](#_Toc57123141)

[Supprimer une donnée spécifique dans une ligne spécifique 12](#_Toc57123142)

[Supprimer une ligne d’enregistrement entière 13](#_Toc57123143)

[Éléments complémentaires 14](#_Toc57123144)

[Les différentes structures conditionnelles possibles 14](#_Toc57123145)

[Listes des types de données que vous pouvez utiliser sur les colonnes 14](#_Toc57123146)

[Listes des contraintes/options que vous pouvez utiliser sur les colonnes 15](#_Toc57123147)

[Clé primaire 17](#_Toc57123148)

[Clé étrangère 19](#_Toc57123149)

[Mise en place d’une clé étrangère à la création de la table 19](#_Toc57123150)

[Mise en place de la clé étrangère avec modification d’une table existante 21](#_Toc57123151)

[Table d’association 24](#_Toc57123152)

[Insertion des enregistrements avec les clés étrangères 27](#_Toc57123153)

Travail à faire :

**Chercher les commandes MariaDB et leurs syntaxes pour :**

## Se connecter à MariaDB

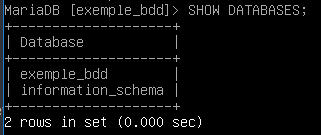
En tant que super admin : sudo mariadb

En tant qu’user avec droits sur une BDD : mariadb -u nomuser -p

# Gestion de la BDD en elle-même

## Visualiser le nom des BDD existantes

Contexte : Vous voulez visualiser le nom de toutes les BDD existantes auxquelles vous avez accès avec votre utilisateur connecté.

SHOW DATABASES ;

## Créer une base de données

CREATE DATABASE nomBDD ;

## Créer un utilisateur administrateur de cette base de données

GRANT ALL ON nomBDD.\* TO 'nomuser'@'localhost' IDENTIFIED BY 'azerty' WITH GRANT OPTION;

FLUSH PRIVILEGES ;

## Utiliser une base de données

USE nomBDD;

## Supprimer une BDD entière

Contexte : Vous avez créé une BDD de tests, vous avez fini avec, donc vous voulez la supprimer.

Syntaxe :

DROP DATABASE nombdd ;

Vous remarquerez que DATABASE est au singulier, car on ne souhaite supprimer qu’une seule DATABASE.

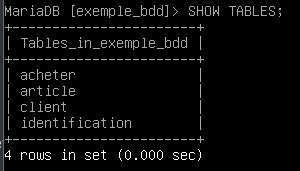
Exemple :

DROP DATABASE test ;

# Gestion de la structure des tables

## Visualiser le nom des tables existantes dans la BDD

Contexte : Vous voulez visualiser le nom de toutes les tables existantes dans la BDD que vous utilisez.

SHOW TABLES ;

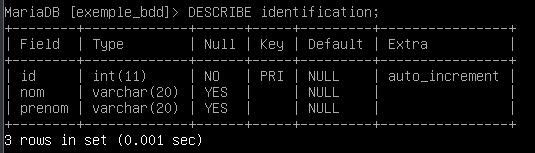
## Visualiser la structure d’une table existante dans la BDD

Contexte : Vous voulez visualiser la structure d’une table précise existante dans la base de données. Cela vous permet de voir le nom de ses colonnes (colonne Field), leurs types (colonne type !), leurs options (colonnes Null et Extra), si elles sont configurées en tant que clé primaire (PRI dans la colonne Key) ou en tant que clé étrangère (MUL dans la colonne Key).

Syntaxe :

DESCRIBE nomdelatable ;

Exemple :

DESCRIBE identification ;

## Créer une table avec ses colonnes et leurs caractéristiques

Syntaxe : CREATE TABLE nomdelatable (dans les parenthèses se configurent chaque colonne sous cette forme : ( nom\_colonne1 type\_colonne1 [options], nom\_colonne2 type\_colonne2 [options]) ;

Le nom de la colonne et le type de la colonne sont des renseignements **obligatoires** pour construire une colonne.

CREATE TABLE nomdelatable (id\_nomdelatable INT(5) NOT NULL PRIMARY KEY AUTO\_INCREMENT, nomcolonne2\_nomdelatable VARCHAR(30), nomcolonne3\_nomdelatable VARCHAR(30));

Exemple :

CREATE TABLE identification (id\_identification INT(5) NOT NULL PRIMARY KEY AUTO\_INCREMENT, nom\_identification VARCHAR(30), prenom\_identification VARCHAR(30));

## Modifier une table (rajouter, supprimer et modifier une colonne)

### Rajouter une colonne dans une table

Contexte : Vous voulez une nouvelle information sur votre BDD, par exemple l’âge de vos utilisateurs, il faut donc rajouter une colonne.

Syntaxe :

ALTER TABLE nomdelatable ADD COLUMN nomdelanouvellecolonne typedelanouvellecolonne ;

Vous voulez altérer la table (= la modifier) => ALTER TABLE nomdelatable, pour y ajouter une colonne => ADD COLUMN configuration d’une colonne. Je rappelle qu’une colonne doit être obligatoirement composée d’un nom et d’un type. Si vous le souhaitez, vous pouvez en plus spécifier des options, comme lorsque vous déclarez les colonnes à la création de la table.

Exemple :

ALTER TABLE identification ADD COLUMN age INT ; => Cela créer dans ma table identification une nouvelle colonne nommée age et de type INT (nombre entier).

### Modifier une colonne dans une table

#### Modifier le nom d’une colonne

Contexte : Vous vous êtes trompés sur le nom d’une colonne, vous voulez la renommer. Lors de cette commande, vous allez changer le nom de la table, mais vous allez devoir aussi saisir à nouveau les informations de configuration de votre colonne, à savoir le type et les options s’il y en a.

Syntaxe :

ALTER TABLE nomdelatable CHANGE nomcolonne nouveaunomcolonne typecolonne [options] ;

Exemple :

Vous n’avez pas respecté les conventions de nommage pour votre nouvelle colonne age, il vous fait donc la renommer en respectant nomcolonne\_nomtable.

ALTER TABLE identification CHANGE age age\_identification INT ;

#### Modifier le type ou les options d’une colonne

Contexte : Vous vous êtes trompés sur le type d’une colonne ou sur ses options, vous voulez donc pouvoir la modifier. Attention, cette commande ne permet pas de changer le nom de la table. Pour cela, regarder la section juste au-dessus.

Syntaxe :

ALTER TABLE nomdelatable MODIFY nomcolonne typecolonne [options] ;

Exemple :

Vous voulez rendre la colonne prénom obligatoire à la saisie d’un utilisateur dans votre BDD, vous allez donc rajouter l’option NOT NULL sur la colonne prenom.

ALTER TABLE identification MODIFY prenom VARCHAR(20) NOT NULL;

Nous venons de rajouter une option, vous pouvez évidemment utiliser cette commande pour changer le type de la colonne, définir ou modifier la taille du champ, rajouter ou supprimer une option …

### Supprimer une colonne dans une table

Contexte : Vous avez une information dans votre BDD qui ne vous est plus nécessaire, vous voulez supprimer la colonne concernée.

Syntaxe :

ALTER TABLE nomdelatable DROP COLUMN nomcolonne;

Vous voulez modifier la table nomdelatable pour y supprimer la colonne nommée nomcolonne.

Exemple :

ALTER TABLE identification DROP COLUMN age ;

## Supprimer une table entière

Contexte : Vous avez créé une table qui ne sert plus aujourd’hui, ou bien vous avez trop de modifications à apporter dessus, il vaut donc mieux la supprimer pour mieux la recréer ensuite.

Syntaxe :

DROP TABLE nomdelatable ;

Exemple :

DROP TABLE identification ;

# Gestion des données contenues dans une table

## Visualiser les enregistrements présents dans une table

### Visualiser tous les enregistrements dans une table avec toutes les colonnes

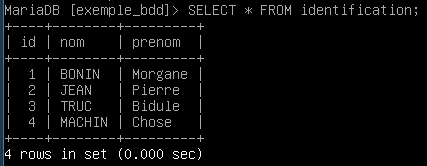
Contexte : Vous avez réalisé des enregistrements dans votre table, ou vous en avez modifié … et vous voulez donc vérifier que vous avez bien fait les choses.

Syntaxe :

SELECT \* FROM nomdelatable ;

Vous voulez sélectionner => SELECT toutes les colonnes => \* de la table nomdelatable => FROM nomdelatable ;

Exemple :

SELECT \* FROM identification ;

À noter que s’il n’y a pas d’enregistrement dans votre table, vous aurez le message suivant : 

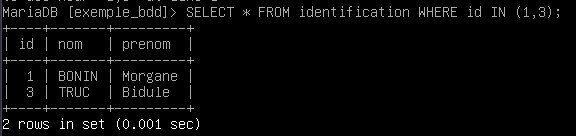
### Visualiser certains enregistrements dans une table avec toutes les colonnes

Syntaxe :

SELECT \* FROM nomdelatable WHERE id IN (x,y,z);

Vous voulez sélectionner => SELECT toutes les colonnes => \* de la table nomdelatable => FROM nomdelatable pour les lignes respectant la condition id IN (x,y,z) c’est-à-dire à la fois id=x et id=y et id=z.

Exemple :

SELECT \* FROM identification WHERE id IN (1,3);

### Visualiser tous les enregistrements dans une table avec des colonnes spécifiques

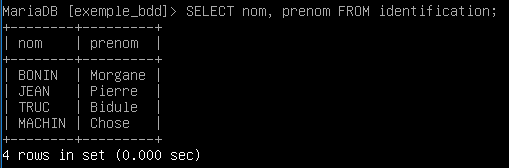
Syntaxe :

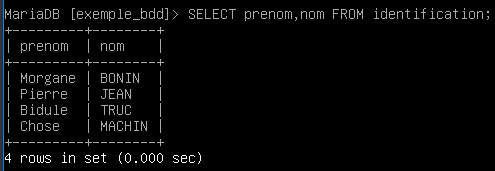
SELECT nomcolonne1, nomcolonne2 FROM nomdelatable;

Vous voulez sélectionner => SELECT nomcolonne1, nomcolonne2=> les colonnées nommées nomcolonne1, nomcolonne2 de la table nomdelatable => FROM nomdelatable.

Exemple :

SELECT nomcolonne1, nomcolonne2 FROM identification ;



À noter, que les colonnes s’affichent dans le sens où vous les avez écrites dans la commande :

## Insérer une ligne de données (un enregistrement) dans une table avec toutes les valeurs

Contexte : Cette commande est à utiliser dans le cas où on remplit notre table avec une ligne qui va contenir une valeur par colonne, c’est-à-dire que chaque colonne sera complétée.

Syntaxe : INSERT INTO nomdelatable VALUES (On écrit les valeurs pour notre enregistrement dans l’ordre d’existence de nos colonnes. La valeur1 correspond à la colonne 1, la valeur 2 à la colonne 2 …)

INSERT INTO nomdelatable VALUES ('valeur 1', 'valeur 2', ...)

Exemple :

INSERT INTO identification VALUES (NULL, 'Morgane', ...)

Pour la valeur 1, j’ai mis NULL afin que la colonne correspondante qui est id\_identification qui correspond à notre clé primaire s’auto incrémente toute seule.

## Insérer une ligne de données (un enregistrement) dans une table en spécifiant uniquement les valeurs à remplir

Contexte : Cette commande est à utiliser dans le cas où on remplit notre table avec une ligne qui va contenir uniquement certaines valeurs par colonne. C’est-à-dire que certaines colonnes seront donc sans valeur, à condition bien sûr que ces colonnes ne soient pas obligatoires, c’est-à-dire qu’elles n’ont pas été configurées comme NOT NULL.

Syntaxe : INSERT INTO nomdelatable (On liste les colonnes auxquelles on veut ajouter des données : nomcolonneA, nomcolonneB …) VALUES (On écrit les valeurs pour notre enregistrement dans l’ordre des colonnes qu’on a déclaré juste avant. La valeurA correspond à la colonne A, la valeur B à la colonne B …)

INSERT INTO nomdelatable (nomcolonneA, nomcolonneB) VALUES ('valeur A', 'valeur B', ...)

Exemple :

INSERT INTO identification(prenom\_identification, nom\_identification) VALUES ('Morgane', 'BONIN' ...)

Cela permet de remplir uniquement les colonnes que l’on souhaite. Les autres colonnes sont remplies par auto-incrémentation s’il elle est paramétrée, ou bien obtienne la valeur NULL.

À noter que pour la saisie des valeurs :

- si on saisit une chaine de caractère, il faut la placer obligatoirement entre ' ' ou entre «  ».

- si on saisit un chiffre, on l’écrit tout simplement : 12

## Insérer plusieurs lignes dans une table en une seule commande

Contexte : C’est la même que les deux commandes précédentes, en fonction de si vous remplissez toutes les colonnes, ou si vous remplissez seulement certaines d’entre elles. La différence ce jour à la fin. C’est-à-dire que vous mettez les informations de votre ligne entre parenthèses, pour saisir une deuxième ligne, il suffit de mettre une virgule après la fin de votre parenthèse puis d’en ouvrir une nouvelle paire.

Syntaxe : INSERT INTO nomdelatable VALUES (mes valeurs pour la 1re ligne), (mes valeurs pour la seconde ligne), (mes valeurs pour la 3ème ligne) ;

Exemple :

INSERT INTO identification VALUES (NULL, 'Morgane', 'BONIN'), (NULL, 'Tom', 'JERY'), (NULL, 'Harry', 'POTTER');

## Modifier une donnée spécifique dans une ligne spécifique

Contexte : Vous vous êtes trompés sur le nom de l’un de vos enregistrements, donc vous devez rectifier.

Syntaxe :

UPDATE nomdelatable SET nomcolonne=nouvellevaleur WHERE id=x ;

Après SET, vous devez spécifier la colonne que vous voulez modifier et indiquer la nouvelle valeur à mettre à la place, cela écrasera l’ancienne, et pour éviter que cette modification soit faite sur toute votre table, vous spécifiez la ligne concernée et ce grâce à l’égalité : id = N°deligne. De cette façon, seule la ligne concernée par cette condition sera modifiée.

UPDATE nomdelatable SET nomcolonne=nouvellevaleur WHERE id=x;

Exemple :

UPDATE identification SET nom=’JERRY’ where id=2; => Cela changera le nom en JERRY de la ligne d’id 2. À faire à condition d’avoir regardé avant l’id de la ligne qu’on veut modifier bien sûr !

Ou bien

UPDATE identification SET nom=’JERRY’ where prenom=’Tom’; => Cela changera le nom en JERRY de la ligne ayant le prénom Tom. Mais si plusieurs lignes ont le prénom Tom, cela les changera le nom en JERRY sur chacune d’entre elles. C’est pour cela que la méthode avec l’id est celle qui est recommandée.

## Supprimer une donnée spécifique dans une ligne spécifique

Contexte : À condition que la colonne accepte le fait que sa donnée soit NULL, c’est-à-dire qu’elle n’ait pas était configurée en NOT NULL, vous pouvez supprimer une donnée sur une ligne en utilisant la même commande que pour modifier une donnée, vue juste au-dessus, mais au lieu de mettre une valeur dans le champ, vous mettez NULL.

Syntaxe :

UPDATE nomdelatable SET nomcolonne=NULL WHERE id=x;

Exemple :

UPDATE identification SET nom=NULL where id=3; => Cela supprimera le nom de l’enregistrement d’id 3.

## Supprimer une ligne d’enregistrement entière

Contexte : Une personne se désinscrit et ne souhaite plus apparaitre dans votre BDD, vous devez donc supprimer la ligne qui lui est associée.

Syntaxe :

DELETE FROM nomdelatable WHERE id=x ;

Exemple :

DELETE FROM identification WHERE id=2 ; => Cela va supprimer la ligne qui correspond à id=2 ;

# Éléments complémentaires

## Les différentes structures conditionnelles possibles

Dans quelques commandes vues précédemment, il y a des conditions qui suivent le mot WHERE.

Il est conseillé de toujours baser ces conditions sur la colonne id, car c’est la colonne qui est la clé primaire de vos tables, c’est donc la seule qui est assurée de ne pas avoir deux fois la même valeur. Mais si vous le souhaitez, vous pouvez vous baser sur n’importe quelle autre commande en suivant la même syntaxe d’écriture.

Nous avons vu que vous pouviez utiliser l’égalité pour appliquer la commande sur une ligne spécifique : **id=x**, et nous avons aussi vu que vous pouviez spécifier plusieurs lignes spécifiques avec **id IN (x, y, z)**.

Vous avez d’autres syntaxes possibles :

**id != x ;** <= Exclure la ligne x en particulier

**id < x ;** <= Inférieur à x

**id <= x ;** <= Inférieur ou égale à x

**id > x ;** <= Supérieur à x

**id >= x ;** <= Supérieur ou égale à x

**id BETWEEN x AND z ;** <= Compris entre x et z (x et z étant compris dans l’intervalle)

## Listes des types de données que vous pouvez utiliser sur les colonnes

À la création des colonnes, vous devez définir obligatoirement un type de donnée pour votre colonne.

À noter que lorsque vous saisissez des données dans un champ, c’est-à-dire que vous donnez une valeur à un champ lors d’un enregistrement, vous devez entourer votre valeur par des ‘ ‘ ou des « » s’il s’agit d’une chaine de caractères. S’il s’agit d’un nombre, vous n’en avez pas besoin.

Les types de colonnes les plus courants sont :

INT : **Nombre entier**. Vous pouvez spécifier entre parenthèses le nombre de chiffres qui seront affichés en BDD, pour un âge mettez INT(2) ou (3) en fonction ;) Pour un téléphone, mettez INT(10).

FLOAT : **Nombre à virgule**

VARCHAR : **Champs de texte avec taille variable**, peut contenir jusqu’à 255 caractères, mais vous pouvez le restreindre en spécifiant la taille entre parenthèses VARCHAR(20) pour une colonne prénom par exemple.

DATE : **Date** sous la forme AAAA-MM-JJ

DATETIME : **Date + heure** sous la forme AAAA-MM-JJ HH:MM:SS

Il y en a d’autres souvent utilisé :

CHAR : **Champs texte dédiés à tes tailles de texte qui ne varient pas** par exemple un code postal si vous le stocké sous forme de chaine de caractères, dans ce cas-là, spécifier la taille entre parenthèses CHAR(5).

TEXT : **Champs de texte pouvant contenir beaucoup plus de caractères**.

TIME : Heure sous la forme HH:MM:SS

Il en existe plein d’autres, mais beaucoup moins usités.

Ces éléments **doivent** être mis en place à la création de la colonne en même temps que la création de la table correspondante (CREATE TABLE), mais ils peuvent être modifiés en modifiant la colonne (ALTER TABLE … MODIFY).

## Listes des contraintes/options que vous pouvez utiliser sur les colonnes

Sur MariaDB vous pouvez mettre en place des options sur vos colonnes, pour les configurer plus spécifiquement. Ces options s’appellent des contraintes.

Il existe différentes contraintes. Les plus importantes sont :

* **NOT NULL** : empêche d’enregistrer une valeur nulle pour la colonne, c’est-à-dire que cela rend obligatoire le fait de saisir une valeur dans cette colonne
* **PRIMARY KEY** : indiquer si cette colonne est considérée comme clé primaire
* **FOREIGN KEY** : permets d’indiquer que cette colonne fait référence à des valeurs préexistantes dans une colonne d'une autre table. C’est la clé étrangère.

En voici d’autres intéressantes :

* **DEFAULT** : attribuer une valeur par défaut si aucune donnée n’est indiquée pour cette colonne lors de l’ajout d’une ligne dans la table. Cette valeur par défaut est NULL.
* **UNIQUE** : les valeurs de la colonne doivent être unique ou NULL, c'est-à-dire qu'à l'exception du marqueur NULL, il ne doit jamais y avoir plus d'une fois la même valeur (pas de doublon possible)

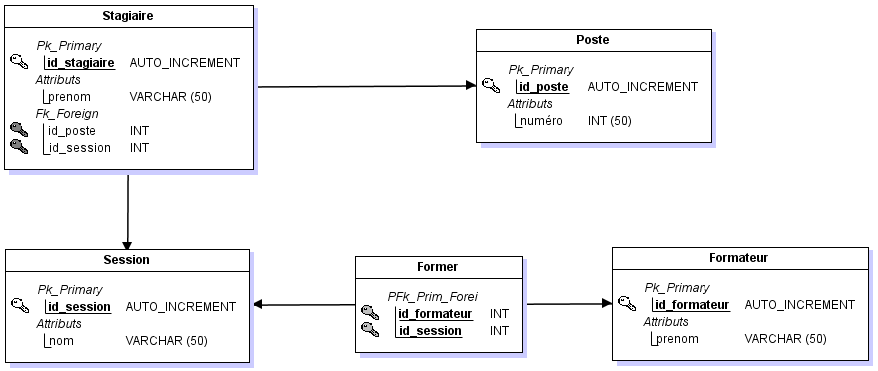
Ces éléments **peuvent** être mis en place à la création de la colonne en même temps que la création de la table correspondante (CREATE TABLE), ou bien en modifiant la colonne (ALTER TABLE … MODIFY).

# Clé primaire

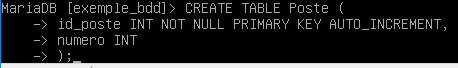
Lors de la conception d’une BDD vous avez découvert le concept de **clé primaire** quand vous avez créé vos MLD. Cet élément est à la base des relations entre nos tables. Il permet d’identifier de manière unique une ligne dans notre BDD.

Nous créons les BDD en nous basant sur les MLD créées.

Prenons l’exemple du cours sur la conception.



Créons la table stagiaire avec les colonnes « id\_poste » et « numero » et indiquons que la colonne « id\_poste » est la clé primaire de notre table, avec tous les paramètres que cela implique. Par convention et lisibilité ont créé la colonne étant la clé primaire en 1er.

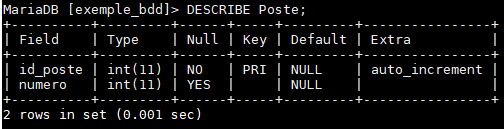
Création table POSTE :

Vous constatez que pour la déclaration de la colonne id\_poste on a indiqué son type « INT » et ensuite ses autres paramètres (**obligatoires** dans notre cas de clé primaire), qu’on appelle des « contraintes », à savoir :

- NOT NULL : pour indiquer que cette colonne id\_poste est obligatoire, donc elle ne peut pas être null

- PRIMARY KEY : Évidemment, pour indiquer que c’est cette colonne qui fait office de clé primaire

- AUTO\_INCREMENT : pour faire en sorte que la valeur numérique (INT) de notre id\_poste s’incrémente automatiquement à chaque nouvel enregistrement, de cette façon ce n’est pas à nous de gérer cet id mais au système. Comme cela, on est certains de ne pas avoir deux fois le même id.

Chaque colonne d’id dans chaque table que vous devez créer doit contenir ces contraintes afin de la déclarer correctement comme clé primaire.

# Clé étrangère

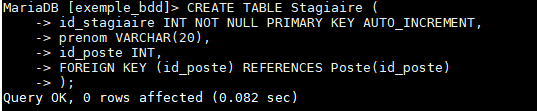
Vous avez une table basique « Poste » qui possède deux colonnes, dont une qui est sa clé primaire.

Dans le cadre de notre MLD, cette table est en relation avec Stagiaire par le biais d’une clé étrangère. C’est la combinaison entre la clé primaire et la clé étrangère qui permet de mettre en lien deux tables.

Par convention et lisibilité ont créé les colonnes faisant office de clé étrangère en dernier.

## Mise en place d’une clé étrangère à la création de la table

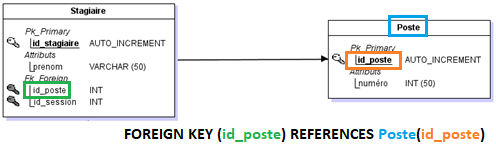
Étant donné que Poste est déjà créé, nous allons créer la table Stagiaire en indiquant la clé étrangère qui fait référence à la table Poste dès la création de la table.

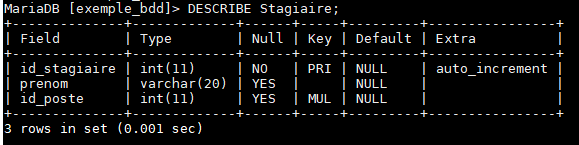
Création Table Stagiaire :

On créer nos trois colonnes, id\_stagiaire en clé primaire, prenom et id\_poste. Car il faut en premier lieu créer la colonne qui va recevoir la relation, ici « id\_poste ». Ensuite, on définit cette clé étrangère par la dernière ligne :

FOREIGN KEY (id\_poste) REFERENCES Poste(id\_poste)

Qui veut dire qu’on déclare en tant que clé étrangère (Foreign key) la colonne id\_poste présente sur la table qu’on est en train de créer, qui fait référence à la clé primaire id\_poste de la table Poste.

En image et en couleur, ça donne :

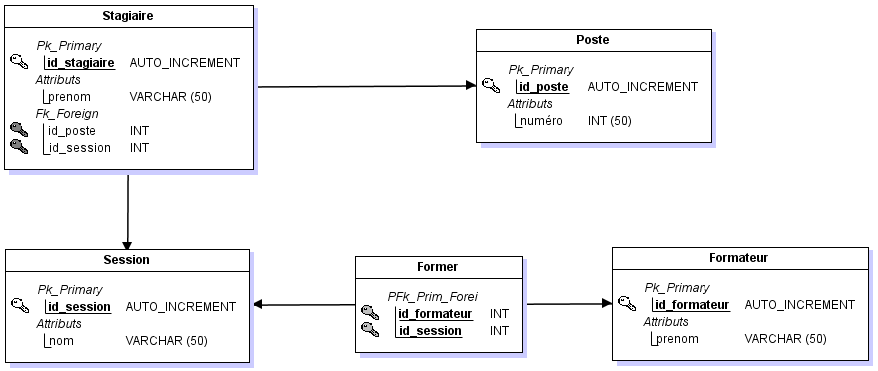
Pour vérifier que cette clé étrangère à bien était mise en place, faites une description de votre table Stagiaire :

Vous constatez que la colonne id\_poste est déclarée en tant que « MUL » dans la colonne « KEY », MUL = MULTIPLE, cela veut dire que dans notre table la colonne MUL permet d’avoir de multiples références vers d’autre table, autrement dit, il s’agit de notre clé étrangère.

Un second point qui peut être amené à votre réflexion et qui dépend de votre gestion, votre MLD et donc votre vision des choses, il s’agit du fait que la colonne servant de clé étrangère ait le droit d’être NULL ou non. À vous de définir si le champ id\_poste est obligé d’être remplis quand on créer un poste. Si la réponse est oui, alors vous devez définir la colonne comme étant NOT NULL. Si la réponse est non, comme dans notre cas, vous pouvez la laisser en NULL.

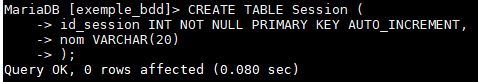
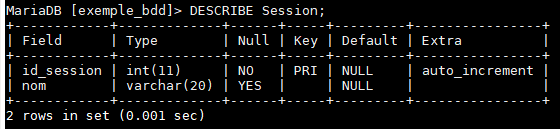
En effet, lorsqu’on créer un stagiaire, il n’y a pas obligation de lui attribuer un poste (ce à quoi la colonne id\_poste correspond).

## Mise en place de la clé étrangère avec modification d’une table existante



Reprenons notre MLD et poursuivons sa création.

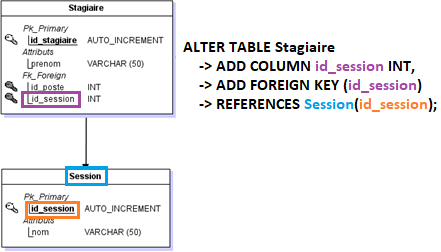
Nos stagiaires doivent appartenir à une session.

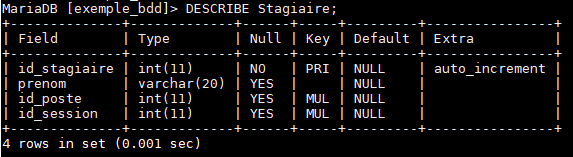
Créons la table Session :

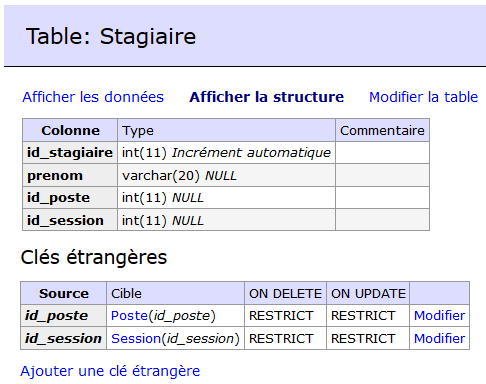
Maintenant que la table Session est créée, on peut modifier notre table Stagiaire pour créer la nouvelle clé étrangère, celle qui fait le lien avec la Session.

Modification table Stagiaire :

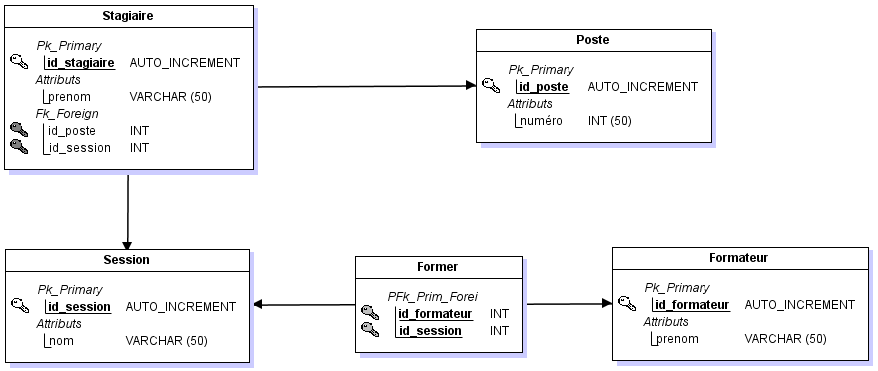
On modifie la table Stagiaire,  
=> On ajoute la colonne id\_session en type INT  
=> On déclare cette nouvelle colonne en tant que clé étrangère qui fait référence à la clé primaire nommée id\_session de la table Session.

En image et en couleur, ça donne :

À présent votre table Stagiaire est correctement paramétrée avec ses deux clés étrangères :

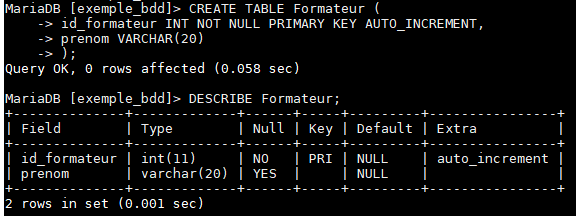
Pour le voir de manière plus certaine, connectez-vous à Adminer et allez sur votre table Stagiaire.

# Table d’association



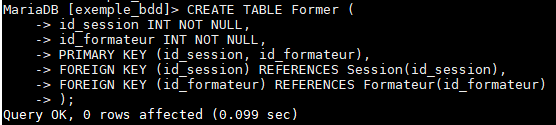
Nous savons à présent créer et paramétrer une clé étrangère à la création même de la table, mais aussi en l’ajoutant après lors d’une modification.

Attaquons-nous à un cas légèrement plus complexe, à savoir la table d’association.

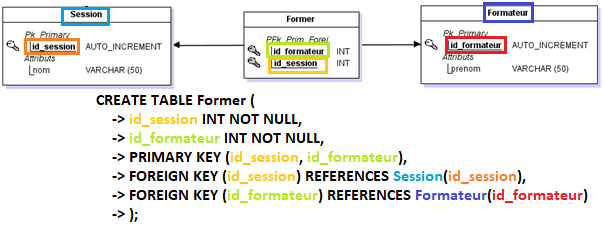
En premier, créons la table Formateur :

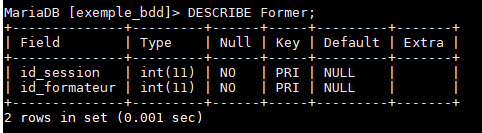
Maintenant, nous avons nos deux tables pour notre relation, à savoir Session et Formateur. Mais grâce à notre MLD on sait qu’on doit mettre en place une table d’association, car une session peut avoir plusieurs formateurs et un formateur peut avoir plusieurs sessions. Comme on est dans une relation multiple (n étant la cardinalité maximum des deux côtés dans notre MCD), cela donne lieu à la table d’association Former.

Mettons en place cette table d’association en créant ses colonnes, en paramétrant sa clé primaire et ses clés étrangères. La spécificité d’une table d’association, comparer à ce que nous venons de faire, c’est que sa clé primaire est composée des deux clés étrangères.

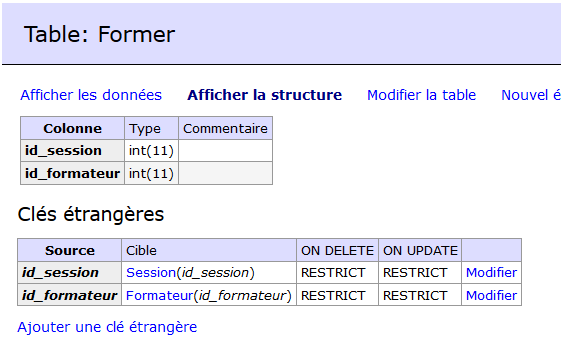


On créer la table d’association Former  
=> on créer la colonne id\_session  
=> on créer la colonne id\_formateur  
=> on met en place la clé primaire qui est composée de id\_session et de id\_formateur  
=> on met en place une clé étrangère sur la colonne id\_session qui référence la clé primaire id\_session de la table Session  
=> on met en place la seconde clé étrangère sur la colonne id\_formateur qui référence la clé primaire id\_formateur de la table Formateur.

En image et en couleur :

Pour vérifier la bonne mise en place, décrivez votre table :

Vous voyez qu’id\_session et id\_formateur sont des clés primaires, mais on ne voit pas le fait qu’elles soient clés étrangères. Pour cela, connectez-vous sur Adminer :



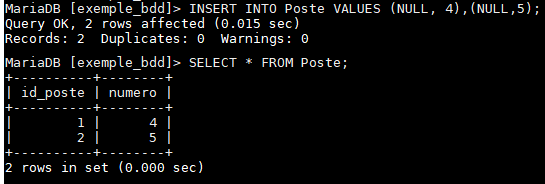
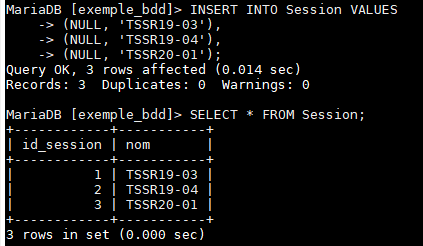
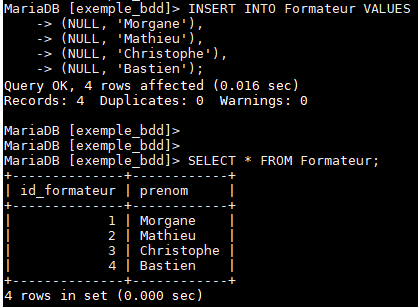
On constate bien que les colonnes id\_session et id\_formateur sont des clés étrangères.

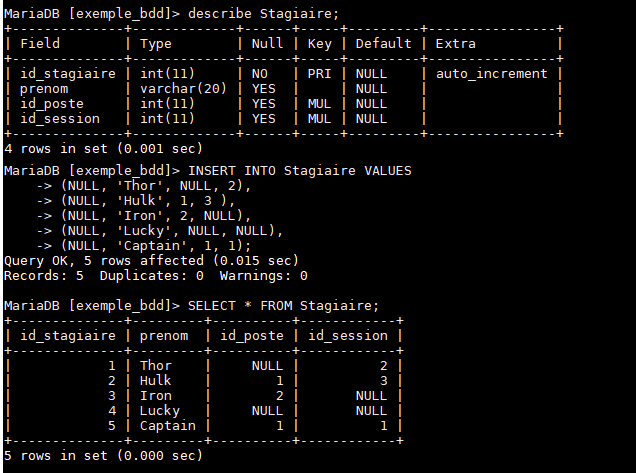
Vous avez donc bien construit votre BDD en suivant votre MLD.

# Insertion des enregistrements avec les clés étrangères

À présent, pour finaliser, il faut que vous mettiez des valeurs dans vos tables, c’est-à-dire que vous mettiez en place des enregistrements dans vos tables.

Cela se passe exactement comme avant, c’est-à-dire que les colonnes servant de clés étrangères sont des colonnes comme les autres, elles attendent donc une saisie de valeur. La valeur doit être l’id correspondant à la ligne que vous voulez mettre en lien.

Dans un premier temps, faisons deux enregistrements sur la table Poste, sur la table Session et sur la table Formateur, car ces tables n’ont pas de clés étrangères.

Créons des stagiaires, qui sont dans une session et possèdent peut-être des pc :

Pour vous faciliter le travail, je vous conseille de faire comme moi. En premier, décrivez votre table, cela permet de voir dans quel ordre les colonnes sont déclarées, de cette façon vous pouvez créer plusieurs stagiaires en une seule ligne de commande, sans avoir à déclarer les colonnes avant.

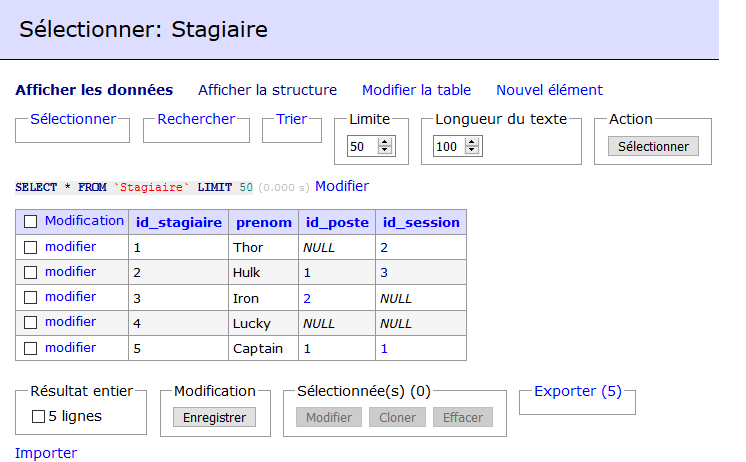
Puis affichez vos saisies.

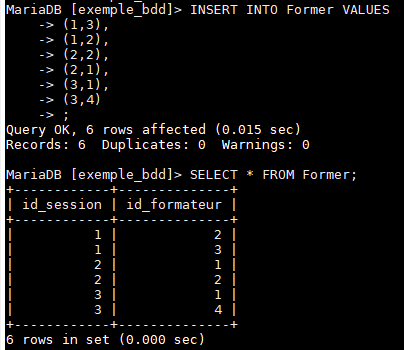
Moi j’ai 5 stagiaires :

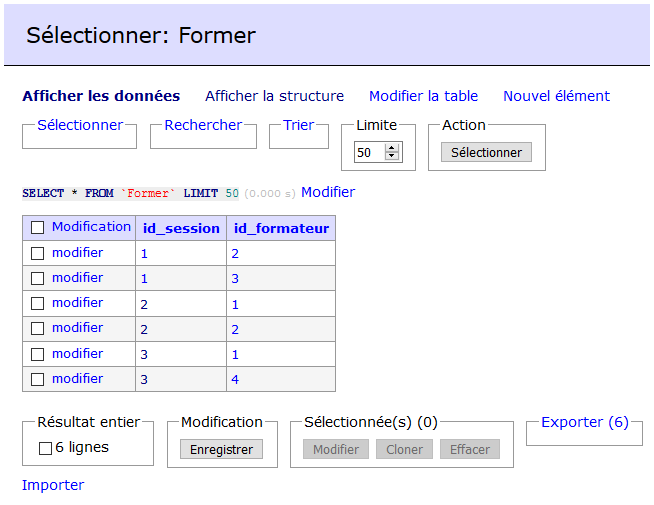
- le 1er, Thor, n’a pas de poste (NULL), et est dans la session d’id 2, je regarde le contenu de la table Session et je vois que celle d’id 2 = TSSR19-04.

- le 2nd, Hulk, possède un poste, le numéro 4 et appartient à la session d’id 3 = TSSR20-01.

Ainsi de suite, je suis la valeur de mes clés étrangères en suivant leur lien vers les clés primaires associées dans les tables respectives pour voir à qu’elle valeur cela correspond.

Sur Adminer, si vous visualisez la table Stagiaire, vous voyez les données, et les valeurs saisies dans les colonnes servant de clés étrangères sont cliquables, afin de vous amener directement sur la valeur correspondante. Cela permet de vérifier que votre clé étrangère fonctionne.

Testons avec la table d’association !

Sur Adminer, toutes nos valeurs sont bien cliquables et renvoi donc bien aux données réelles.

Évidemment, les données vues comme ça, c’est un peu barbare, suivre le lien à la main pour savoir qui est associé à qui. Mais souvent une BDD est utilisée dans le cadre d’un site web, qui va permettre cet affichage simple des informations.

Voici le contenu d’un site internet basique reprenant les infos que nous avons mis dans notre BDD.

Vous verrez le contenu de chaque table, mais aussi ce qu’on peut faire grâce à cela, c’est-à-dire obtenir une table avec directement le nom des éléments liées et non pas leurs ID. C’est ce qui nous intéresse.

