Manipulation d'une bases de données

Partie 2

Créer une base de données

Dans cette partie vous allez apprendre

- Créer la base de données
- Moteur de stockage
- Création des tables.
- Manipuler les tables
- Les contraintes d'intégrité

Créer la base de données

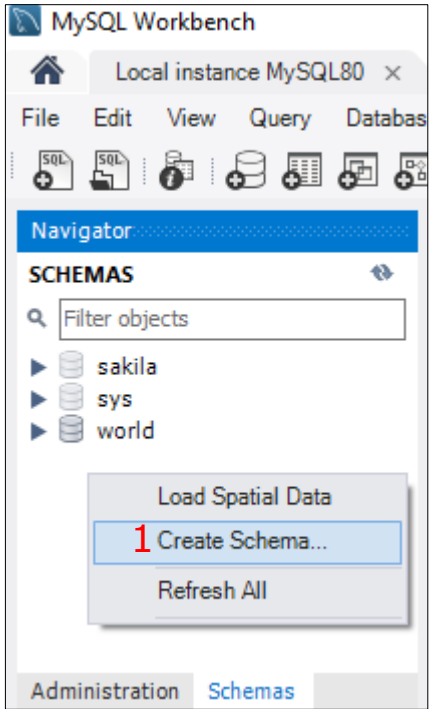
- Base de données et SGBD
  - Une base de données est un ensemble d'informations stockées de manière organisée et structurée afin de pouvoir facilement consulter et modifier leur contenu.
  - Un SGBD (Système de gestion de bases de données) est un logiciel permettant d'interagir avec une base de données : sélectionner, ajouter, modifier et supprimer des données de la base.
  - On regroupe généralement ces opérations sous l'acronyme **CRUD** pour (**Create**, **Read**, **Update** and **Delete**).
  - MySQL est bien évidemment un SGBD.

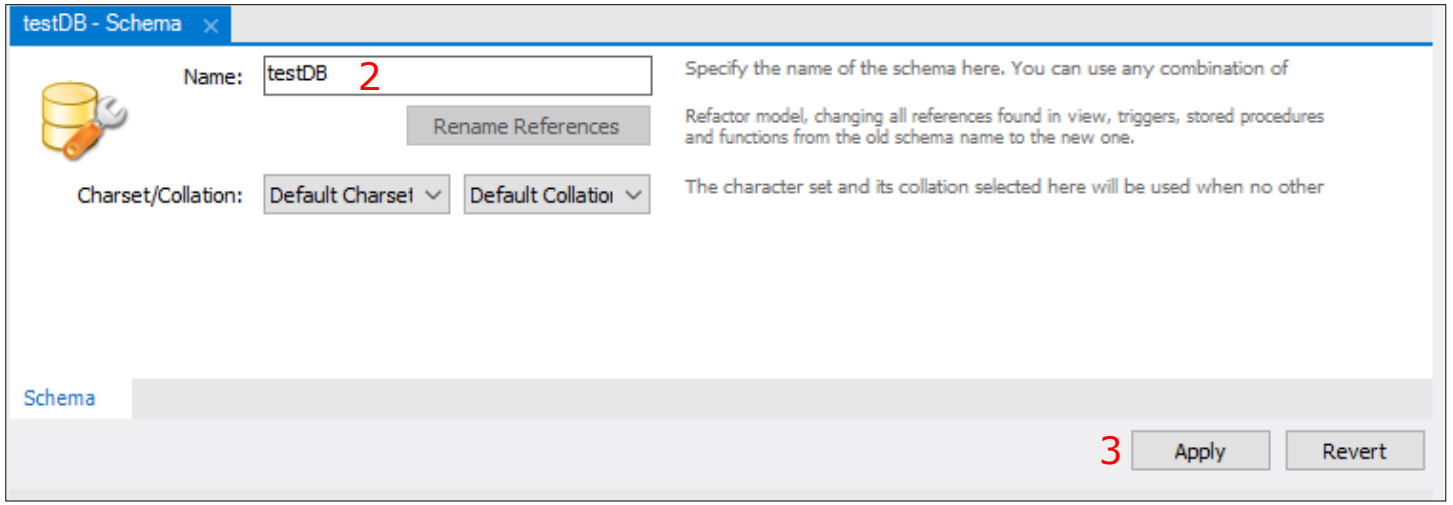
Créer la base de données

- Langage de requêtes SQL
  - Le SQL (**Structured Query Language**) est un langage d'interrogation des B.D relationnelles qui permet :
    - Définir et modifier la structure de la B.D.
    - Interroger la B.D.
    - Contrôler l sécurité et d'intégrité de la B.D.
    - Sauvegarde et restauration La B.D.
  - Le SQL peut être réparti en quatre catégories :
    - **LDD** : langage de définition de données qui manipule la structure de la B.D.
    - **LID** : langage d'interrogation de données qui permet de rechercher les informations dans la B.D.
    - **LMD** : langage de manipulation de données qui permet de manipuler les données de la B.D et de les mettre à jour.
    - **LCD** : langage de contrôle de données qui permet de définir les droits d'accès pour les utilisateurs de la B.D.

Créer la base de données

- Créer une base de données avec l'outil graphique
  - Pour créer la B.D graphiquement dans **MySQL Workbench** :
    - 1- Un clic droit dans une emplacement vide de l'espace **SCHEMAS.**
    - 2- Sélectionner **Create Schema**.
    - 3- Taper le nom de la base de données : **testDB** par exemple.
    - 4- Cliquer sur le bouton **Apply**.





Créer la base de données

- Créer une base de données à l'aide de SQL
  - Syntaxe SQL pour créer une B.D :

```
CREATE DATABASE databasename;
```

  - L'exemple suivant permet de créer la base de données **testDB** :

```
CREATE DATABASE testDB;
```

  - Vérifier la création de la B.D :

```
SHOW CREATE DATABASE testDB;
```

Database	Create Database
testDB	CREATE DATABASE `testDB` /*!40100 DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci */ /*!80016 DEFAULT ENCRYPTION='N' */

  - Afficher les bases de données existantes :

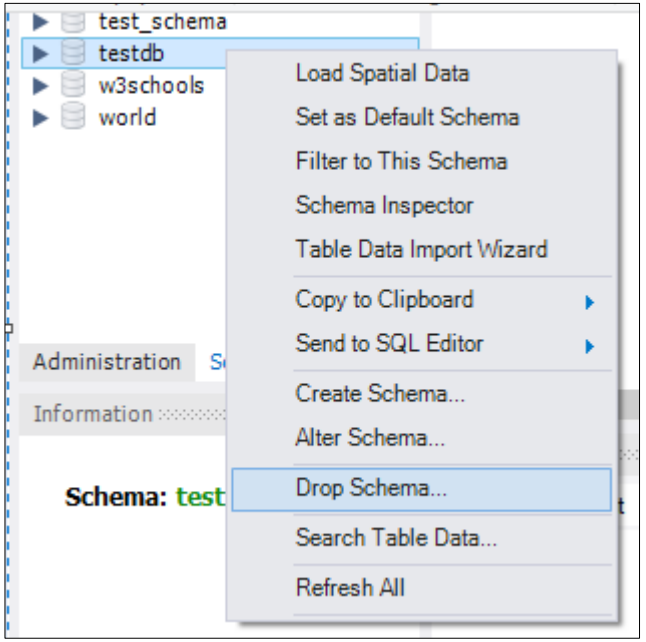
```
SHOW DATABASES;
```

  - Pour utiliser la B.D crée :

```
USE testDB;
```

Créer la base de données

- Supprimer une base de données
  - Outil graphique :
    - 1- Un clic droit sur la B.D.
    - 2- Sélectionner **Drop Schema**.
    - 3- Cliquer sur **Drop Now**.



  - Syntaxe SQL pour supprimer une B.D :

```
DROP DATABASE databasename;
```

  - L'exemple suivant permet de supprimer la base de données **testDB** :

```
DROP DATABASE testDB;
```

Moteur de stockage

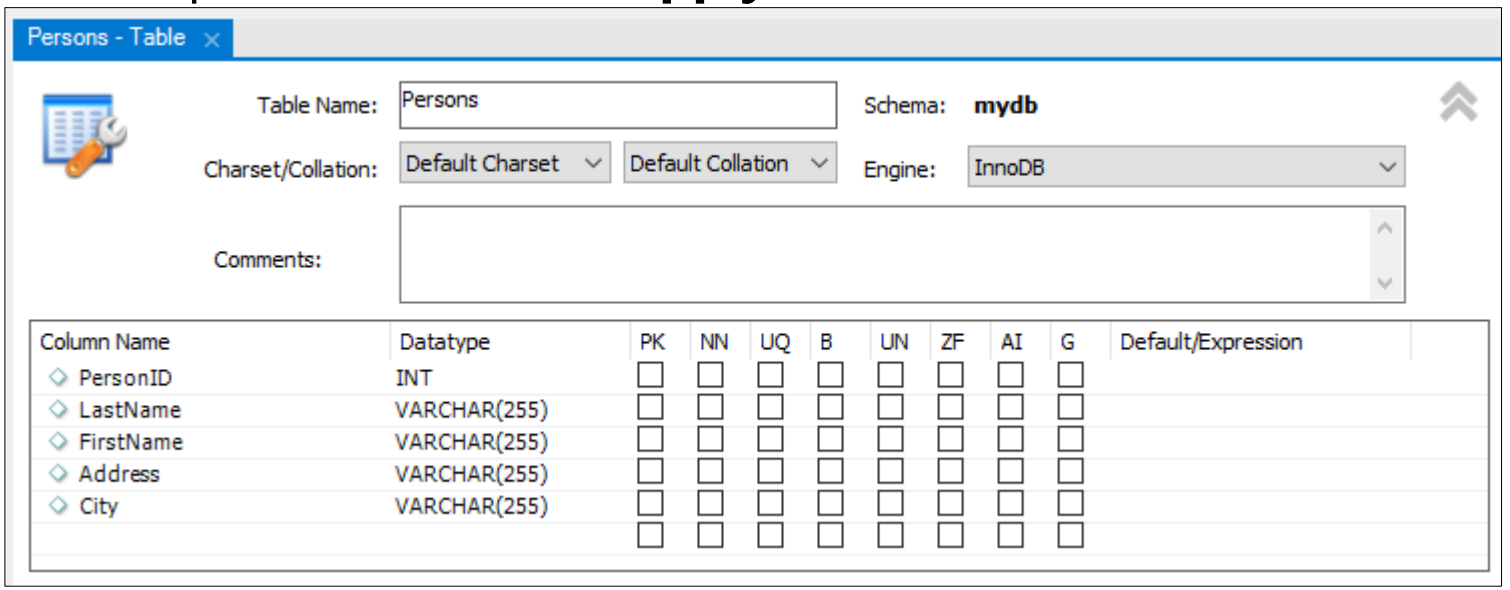
- Moteurs de stockage
  - On appelle moteur de stockage l'ensemble des algorithmes utilisés par un SGBDR pour stocker les informations et y accéder au moyen d'une requête SQL.
  - Un moteur de stockage est aussi appelé moteur de table.
  - Voici la syntaxe SQL pour afficher les engins disponible :

```
SHOW ENGINES;
```

Engine	Support	Comment	Transactions	XA	Savepoints
MEMORY	YES	Hash based, stored in memory, useful for temp...	NO	NO	NO
MRG_MYISAM	YES	Collection of identical MyISAM tables	NO	NO	NO
CSV	YES	CSV storage engine	NO	NO	NO
FEDERATED	NO	Federated MySQL storage engine	NULL	NULL	NULL
PERFORMANCE_SCHEMA	YES	Performance Schema	NO	NO	NO
MyISAM	YES	MyISAM storage engine	NO	NO	NO
InnoDB	DEFAULT	Supports transactions, row-level locking, and fo...	YES	YES	YES
BLACKHOLE	YES	/dev/null storage engine (anything you write to ...	NO	NO	NO
ARCHIVE	YES	Archive storage engine	NO	NO	NO

Création des tables

- Créer une table avec l'outil graphique
  - Les étapes :
    - 1- Sélectionner la base de données et cliquer sur la petite flèche pour développer les objets de la base de données.
    - 2 – Cliquer avec le bouton droit sur l'objet **Tables** puis sélectionner **Create Table**.
    - 3- Saisir le nom de la tables et les colonnes.
    - 4- Cliquer sur le bouton **Apply**.



Création des tables

- Créer une table à l'aide de SQL
  - Syntaxe SQL pour créer une table :

```
CREATE TABLE table_name (
    column1 datatype constraint,
    column2 datatype constraint,
    ....
);
```

  - L'exemple suivant permet de créer la table **Persons** :

```
CREATE TABLE Persons (
    PersonID INT,
    LastName VARCHAR(255),
    FirstName VARCHAR(255),
    Address VARCHAR(255),
    City VARCHAR(255)
);
```

Création des tables

- Typage des colonnes
  - Type de données numériques :

Type	Taille	Utilisation
TINYINT	1 octet	petites valeurs entières/ Boolean
SMALLINT	2 octets	valeur entière
MEDIUMINT	3 octets	valeur entière
INT ou INTEGER	4 octets	valeur entière
BIGINT	8 octets	Valeur maximale entier
FLOAT	4 octets	Simple précision valeurs à virgule flottante
DOUBLE	8 octets	Double-précision valeurs à virgule flottante
DECIMAL	De DECIMAL (M, D), si M> D, M + 2 est par ailleurs D + 2	valeur décimale

Création des tables

- Typage des colonnes
  - Type de données caractères :

Type	Taille (octets)	Utilisation
CHAR	0-255	chaîne longueur fixe
VARCHAR	0-65535	chaînes de longueur variable
TINYBLOB	0-255	Pas plus de 255 caractères dans une chaîne binaire
TINYTEXT	0-255	Courtes chaînes de texte
BLOB	0-65535	données textuelles longues sous forme binaire
TEXTE	0-65535	Longue données de texte
MEDIUMBLOB	0-16777215	forme binaire de longueur moyenne des données de texte
MEDIUMTEXT	0-16777215	longueur moyenne des données de texte
LONGBLOB	0-4294967295	Grands données de texte sous forme binaire
LONGTEXT	0-4294967295	Grande données de texte

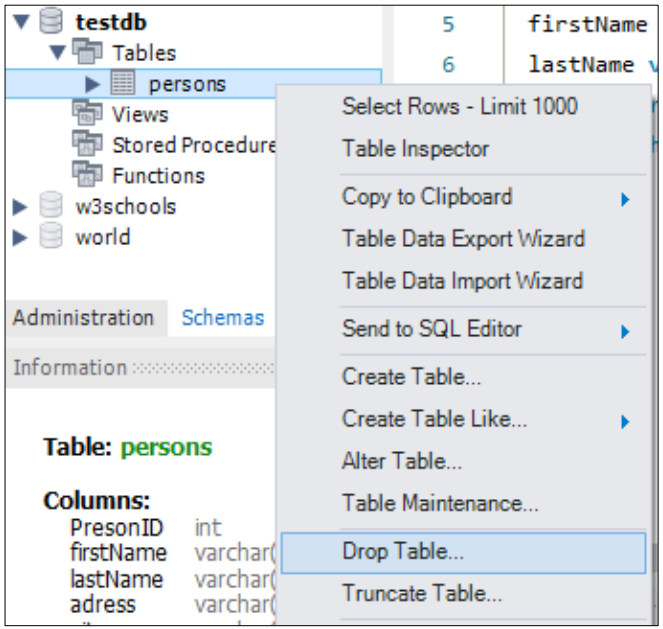
Création des tables

- Typage des colonnes
  - Type de données date/heure :

Type	Taille (Byte)	Format	Utilisation
DATE	3	AAAA-MM-JJ	Les valeurs de date
TIME	3	HH: MM: SS	Valeur temps ou la durée
ANNÉE	1	AAAA	Année Valeur
DATETIME	8	AAAA-MM-JJ HH: MM: SS	Mixage valeurs date et heure
TIMESTAMP	4	AAAAMMJJ HHMMSS	Date de mélange et la valeur temps, un horodatage

Manipuler les tables

- Supprimer une table avec l'outil graphique
  - Les étapes :
    - 1- Sélectionner la base de données et cliquer sur la petite flèche pour développer les objets de la base de données.
    - 2- Sélectionner l'objet **Tables** et cliquer sur la petite flèche pour afficher toutes les tables disponibles.
    - 3 – Cliquer avec le bouton droit sur la table que vous voulez supprimer puis sélectionner **Drop Table**.
    - 4- Cliquer sur le bouton **Drop Now**.



Manipuler les tables

- Supprimer une table à l'aide de SQL
  - Syntaxe SQL pour supprimer une table :

```
DROP TABLE table_name;
```

  - L'exemple suivant permet de supprimer la table **Persons** :

```
DROP TABLE Persons;
```

  - Syntaxe SQL pour supprimer les données à l'intérieur d'une table, mais pas la table elle-même :

```
TRUNCATE TABLE table_name;
```

  - L'exemple suivant permet de vider la table **Shippers** :

```
TRUNCATE TABLE Shippers;
```

Manipuler les tables

- Ajouter une colonne à une table
  - Syntaxe SQL pour ajouter une colonne à une table :

```
ALTER TABLE table_name
ADD column_name datatype;
```

  - L'exemple suivant ajoute la colonne **Email** à la table **Customers** :

```
USE W3schools;
ALTER TABLE Customers
ADD Email VARCHAR(255);
```

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country	Email
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany	NULL
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico	NULL
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico	NULL
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK	NULL
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden	NULL

Manipuler les tables

- Modifier le type d'une colonne
  - Syntaxe SQL pour modifier le type d'une colonne :

```
ALTER TABLE table_name
MODIFY COLUMN column_name datatype;
```

  - L'exemple suivant modifie le type de la colonne **Email** de la table **Customers** :

```
USE W3schools;
ALTER TABLE Customers
MODIFY COLUMN Email VARCHAR(50);
```

Manipuler les tables

- Modifier le nom d'une colonne
  - Syntaxe SQL pour modifier le nom d'une colonne :

```
ALTER TABLE table_name
CHANGE old_column new_column datatype;
```

  - L'exemple suivant modifie le nom de la colonne **Email** de la table **Customers** :

```
USE W3schools;
ALTER TABLE Customers
CHANGE Email Adress_email VARCHAR(50);
```

Manipuler les tables

- Supprimer une colonne
  - Syntaxe SQL pour modifier le nom d'une colonne :

```
ALTER TABLE table_name
DROP COLUMN column_name;
```

  - L'exemple suivant supprime la colonne **Adress_email** de la table **Customers** :

```
USE W3schools;
ALTER TABLE Customers
DROP COLUMN Adress_email;
```

Les contraintes d'intégrité

- Contraintes d'intégrités
  - Une contrainte d'intégrité est une règle qui définit la cohérence d'une donnée de la B.D.
  - Il existe trois types de contraintes d'intégrités :
    - 1- Contrainte d'intégrité de domaine :
      - **`NOT NULL`** : garantit qu'une colonne ne peut pas avoir une valeur **`NULL`**.
      - **`UNIQUE`** : garantit que toutes les valeurs d'une colonne sont différentes.
      - **`CHECK`** : Garantit que les valeurs d'une colonne satisfont à une condition spécifique.
      - **`DEFAULT`** : Définit une valeur par défaut pour une colonne si aucune valeur n'est spécifiée.
    - 2- Contrainte d'intégrité des entités :
      - **`PRIMARY KEY`** : Une combinaison de **`NOT NULL`** et **`UNIQUE`**.
    - 3- Contrainte d'intégrité référentielle :
      - **`FOREIGN KEY`** : assure l'intégrité des relations entre les clés primaires et las clés étrangères.

Les contraintes d'intégrité

- La contrainte NOT NULL
  - Par défaut, une colonne peut contenir des valeurs **NULL**.
  - Le **NOT NULL** impose qu' une colonne n'accepte pas **NULL**.
  - L'exemple suivant spécifie la contrainte **NOT NULL** au moment de la création de la table :

```
CREATE TABLE Persons (
    ID INT NOT NULL,
    LastName VARCHAR(255) NOT NULL,
    FirstName VARCHAR(255) NOT NULL,
    Age INT
);
```

  - Après la création de la table :

```
ALTER TABLE Persons
MODIFY Age INT NOT NULL;
```

Les contraintes d'intégrité

- La contrainte UNIQUE
  - La contrainte **UNIQUE** garantit que toutes les valeurs d'une colonne sont différentes.
  - L'exemple suivant spécifie la contrainte **UNIQUE** au moment de la création de la table :

```
CREATE TABLE Persons (
    ID INT NOT NULL UNIQUE,
    LastName VARCHAR(255) NOT NULL,
    FirstName VARCHAR(255) NOT NULL,
    Age INT
);
```

Les contraintes d'intégrité

- La contrainte UNIQUE
  - Il y'a aussi la possibilité de définir la contrainte **UNIQUE** à la fin dans la syntaxe de la création de la table :

```
CREATE TABLE Persons (
    ID INT NOT NULL,
    LastName VARCHAR(255) NOT NULL,
    FirstName VARCHAR(255) NOT NULL,
    Age INT,
    UNIQUE (ID)
);
```

Les contraintes d'intégrité

- La contrainte UNIQUE
  - Il y'a aussi la possibilité de nommer la contrainte **UNIQUE** à l'aide de l'instruction **CONSTRAINT** au moment de la création de la table :

```
CREATE TABLE Persons (
    ID INT NOT NULL,
    LastName VARCHAR(255) NOT NULL,
    FirstName VARCHAR(255) NOT NULL,
    Age INT,
    CONSTRAINT UC_Person UNIQUE (ID,LastName)
);
```

Les contraintes d'intégrité

- La contrainte UNIQUE
  - Syntaxe pour ajouter la contrainte **UNIQUE** après la création de la table :

```
ALTER TABLE Persons
ADD UNIQUE (ID);
```

  - Syntaxe pour nommer la contrainte **UNIQUE** après la création de la table :

```
ALTER TABLE Persons
ADD CONSTRAINT UC_Person UNIQUE (ID,Lastname);
```

  - Pour supprimer la contrainte **UNIQUE** :

```
ALTER TABLE Persons
DROP CONSTRAINT UC_Person;
```

  - Ou bien :

```
ALTER TABLE Persons DROP INDEX UC_Person;
```

Les contraintes d'intégrité

- La contrainte PRIMARY KEY
  - La contrainte **PRIMARY KEY** identifie de manière unique chaque enregistrement d'une table.
  - L'exemple suivant spécifie la contrainte **PRIMARY KEY** au moment de la création de la table :

```
CREATE TABLE Persons (
    ID INT PRIMARY KEY,
    LastName VARCHAR(255) NOT NULL,
    FirstName VARCHAR(255) NOT NULL,
    Age INT
);
```

Les contraintes d'intégrité

- La contrainte PRIMARY KEY
  - Il y'a aussi la possibilité de définir la contrainte **PRIMARY KEY** à la fin dans la syntaxe de la création de la table :

```
CREATE TABLE Persons (
    ID INT,
    LastName VARCHAR(255) NOT NULL,
    FirstName VARCHAR(255) NOT NULL,
    Age INT,
    PRIMARY KEY (ID)
);
```

Les contraintes d'intégrité

- La contrainte PRIMARY KEY
  - Il y'a aussi la possibilité de nommer la contrainte **PRIMARY KEY** à l'aide de l'instruction **CONSTRAINT** au moment de la création de la table :

```
CREATE TABLE Persons (
    ID INT NOT NULL,
    LastName VARCHAR(255) NOT NULL,
    FirstName VARCHAR(255) NOT NULL,
    Age INT,
    CONSTRAINT PK_Person PRIMARY KEY (ID,Lastname)
);
```

Les contraintes d'intégrité

- La contrainte PRIMARY KEY
  - Syntaxe pour ajouter la contrainte **PRIMARY KEY** après la création de la table :

```
ALTER TABLE Persons
ADD PRIMARY KEY (ID);
```

  - Syntaxe pour nommer la contrainte **PRIMARY KEY** après la création de la table :

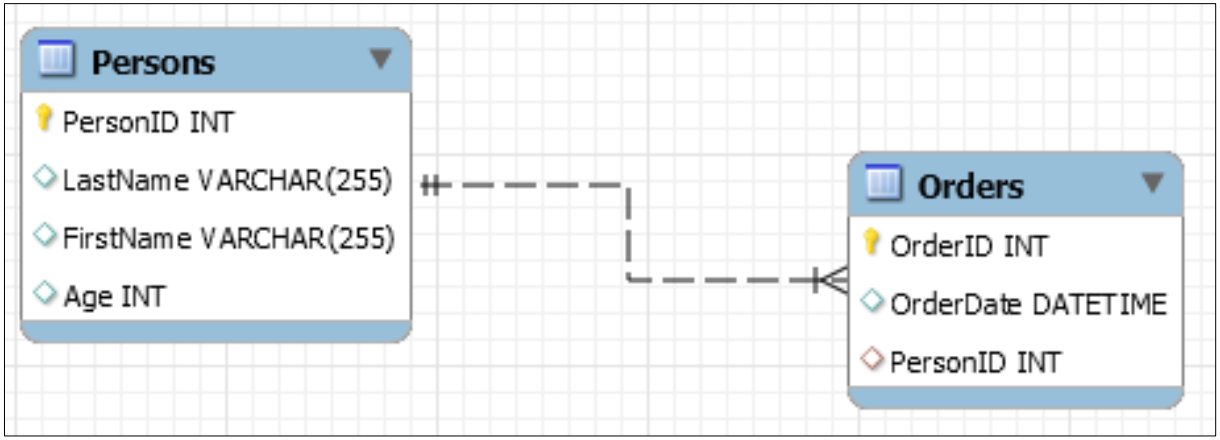
```
ALTER TABLE Persons
ADD CONSTRAINT PK_Person PRIMARY KEY (ID,Lastname);
```

  - Pour supprimer la contrainte **PRIMARY KEY** :

```
ALTER TABLE Persons
DROP PRIMARY KEY;
```

Les contraintes d'intégrité

- La contrainte FOREIGN KEY
  - La contrainte **`FOREIGN KEY`** assure les relations entre les clés primaires et las clés étrangères.
  - Une clé étrangère est un champ dans une table, qui fait référence à la clé primaire dans une autre table :



	PersonID	LastName	FirstName	Age
▶	1	Hansen	Ola	30
	2	Svendson	Tove	23
	3	Pettersen	Kari	20
*	NULL	NULL	NULL	NULL

	OrderID	OrderDate	PersonID
▶	1	2020-07-15	3
	2	2021-01-21	3
	3	2019-09-10	2
	4	2022-06-02	1
*	NULL	NULL	NULL

Les contraintes d'intégrité

- La contrainte FOREIGN KEY
  - L'exemple suivant spécifie la contrainte **FOREIGN KEY** au moment de la création de la table :

```
CREATE TABLE Persons (
    PersonID INT PRIMARY KEY,
    LastName VARCHAR(255) NOT NULL,
    FirstName VARCHAR(255) NOT NULL,
    Age INT
);
CREATE TABLE Orders (
    OrderID INT PRIMARY KEY,
    OrderDate DATE,
    PersonID INT,
    FOREIGN KEY (PersonID) REFERENCES Persons(PersonID)
);
```

Les contraintes d'intégrité

- La contrainte FOREIGN KEY
  - Il y'a aussi la possibilité de nommer la contrainte **FOREIGN KEY** à l'aide de l'instruction **CONSTRAINT** au moment de la création de la table :

```
CREATE TABLE Orders (
    OrderID INT PRIMARY KEY,
    OrderDate DATE,
    PersonID INT,
    CONSTRAINT FK_PersonOrder FOREIGN KEY (PersonID)
    REFERENCES Persons(PersonID)
);
```

Les contraintes d'intégrité

- La contrainte FOREIGN KEY
  - Syntaxe pour ajouter la contrainte **FOREIGN KEY** après la création de la table :

```
ALTER TABLE Orders
ADD FOREIGN KEY (PersonID) REFERENCES Persons(PersonID);
```

  - Syntaxe pour nommer la contrainte **PRIMARY KEY** après la création de la table :

```
ALTER TABLE Orders
ADD CONSTRAINT FK_PersonOrder
FOREIGN KEY (PersonID) REFERENCES Persons(PersonID);
```

  - Pour supprimer la contrainte **PRIMARY KEY** :

```
ALTER TABLE Orders
DROP FOREIGN KEY FK_PersonOrder;
```

Les contraintes d'intégrité

- La contrainte CHECK
  - La contrainte **CHECK** est utilisée pour limiter la plage de valeurs qui dans une colonne.
  - L'exemple suivant spécifie la contrainte **CHECK** au moment de la création de la table : ici **CHECK** garantit **Age >= 18**

```
CREATE TABLE Persons (
    ID INT PRIMARY KEY,
    LastName VARCHAR(255) NOT NULL,
    FirstName VARCHAR(255) NOT NULL,
    Age INT CHECK (Age>=18),
    City VARCHAR(255)
);
```

Les contraintes d'intégrité

- La contrainte CHECK
  - Il y'a aussi la possibilité de définir la contrainte **CHECK** à la fin dans la syntaxe de la création de la table :

```
CREATE TABLE Persons (
    ID INT PRIMARY KEY,
    LastName VARCHAR(255) NOT NULL,
    FirstName VARCHAR(255) NOT NULL,
    Age INT CHECK (Age>=18),
    City VARCHAR(255) ,
    CHECK (Age>=18)
);
```

Les contraintes d'intégrité

- La contrainte CHECK
  - Il y'a aussi la possibilité de nommer la contrainte **CHECK** à l'aide de l'instruction **CONSTRAINT** au moment de la création de la table :

```
CREATE TABLE Persons (
    ID INT PRIMARY KEY,
    LastName VARCHAR(255) NOT NULL,
    FirstName VARCHAR(255) NOT NULL,
    Age INT,
    City VARCHAR(255),
    CONSTRAINT CHK_Person CHECK (Age>=18 AND City='Rio')
);
```

Les contraintes d'intégrité

- La contrainte CHECK
  - Syntaxe pour ajouter la contrainte **CHECK** après la création de la table :

```
ALTER TABLE Persons
ADD CHECK (Age BETWEEN 18 AND 25);
```

  - Syntaxe pour nommer la contrainte **CHECK** après la création de la table :

```
ALTER TABLE Persons
ADD CONSTRAINT CHK_Person CHECK (Age BETWEEN 18 AND 25);
```

  - Pour supprimer la contrainte **CHECK** :

```
ALTER TABLE Persons
DROP CHECK CHK_Person;
```

Les contraintes d'intégrité

- La contrainte DEFAULT
  - La contrainte **DEFAULT** est utilisée pour définir une valeur par défaut pour une colonne.
  - L'exemple suivant spécifie la contrainte **DEFAULT** au moment de la création de la table : ici **DEUFAULT** définit la valeur **'Sandnes'** comme valeur par défaut du champ **City**

```
CREATE TABLE Persons (
    ID INT PRIMARY KEY,
    LastName VARCHAR(255) NOT NULL,
    FirstName VARCHAR(255) NOT NULL,
    Age INT CHECK (Age>=18),
    City VARCHAR(255) DEFAULT 'Sandnes'
);
```

Les contraintes d'intégrité

- La contrainte DEFAULT
  - Voici un autre exemple d'utilisation de la contrainte **DEFAULT** avec la fonction **CURRENT_DATE()** :

```
CREATE TABLE Orders (
    ID INT PRIMARY KEY,
    OrderNumber INT NOT NULL,
    OrderDate DATE DEFAULT CURRENT_DATE()
);
```

Les contraintes d'intégrité

- La contrainte DEFAULT
  - Syntaxe pour ajouter la contrainte **DEFAULT** après la création de la table :

```
ALTER TABLE Persons
ALTER CITY SET DEFAULT 'Sandnes';
```

  - Pour supprimer la contrainte **DEFAULT** :

```
ALTER TABLE Persons
ALTER CITY DROP DEFAULT;
```