# 1- Introduction

Le langage SQL (Structured Query Language) est un langage de base de données conçu pour gérer les données stockées dans un système de gestion de base de données relationnelle.

De nombreux systèmes de gestion de bases de données relationnelles actuels, tels qu'Oracle Database, Microsoft SQL Server, MySQL, IBM DB2, IBM Informix et Microsoft Access, utilisent SQL.

Dans un système de gestion de base de données (SGBD), le langage SQL est utilisé pour :

- Créer la structure de la base de données et des tables.
- Effectuer des tâches de gestion de base de données élémentaires (ajout, suppression et modification).
- Exécuter des requêtes complexes pour transformer les données brutes en informations utiles.

On peut utiliser le language SQL pour :

- créer la structure de la base de données et des tables => l'utiliser entant que langage de définition de données (DDL).
- supprimer, sélectionner et mettre à jour des données dans les tables de la base de données => l'utiliser en tant que langage de manipulation de données (DML).
- Les principales instructions DDL (Data Definition Language) SQL sont :
  - CREATE DATABASE
  - CREATE TABLE
  - DROP TABLE
  - ALTER TABLE.
- La majorité des requêtes SQL seront compatibles avec le SGBD Microsoft SQL Server.

# 2- Créer une base de données

Exemple : `CREATE DATABASE SW;`

Une nouvelle base de données nommée SW est créée à l'aide de l'instruction SQL `CREATE DATABASE SW;`.
Une fois la base de données créée, la prochaine étape consiste à créer les tables de la base de données.

# 3- Créer une table

Le format général de la commande CREATE TABLE est le suivant :

```
CREATE TABLE <nom_table> (
    NomColonne1 TypeDeDonnées1 ContrainteOptionnelle1,
    NomColonne2 TypeDeDonnées2 ContrainteOptionnelle2,
    ...
    ContraintesOptionnellesDeTable
);
```

- `<nom_table>` est le nom de la table de la base de données, par exemple, `Employee`.
- Chaque champ dans CREATE TABLE a trois parties : NomColonne, TypeDeDonnées, ContrainteOptionnelle.

# 3-1 Types de données

Le type de données, tel que décrit ci-dessous, doit être un type de données système ou un type de données défini par l'utilisateur. De nombreux types de données ont une taille spécifiée, telle que CHAR(35) ou Numeric(8,2).

- **bit :** Données entières avec une valeur de 1 ou 0.
- **int :** Données entières (nombre entier) de -2^31 (-2,147,483,648) à 2^31 – 1 (2,147,483,647).
- **smallint :** Données entières de 2^15 (-32,768) à 2^15 – 1 (32,767).
- **tinyint :** Données entières de 0 à 255.
- **decimal :** Données numériques à précision fixe et échelle, de -10^38 -1 à 10^38.
- **numeric :** Synonyme de decimal.
- **timestamp :** Numéro unique dans toute la base de données.
- **uniqueidentifier :** Identifiant global unique (GUID).
- **money :** Valeurs monétaires de -2^63 (-922,337,203,685,477.5808) à 2^63 – 1 (+922,337,203,685,477.5807), avec une précision jusqu'à la dix millième d'une unité monétaire.
- **smallmoney :** Valeurs monétaires de -214,748.3648 à +214,748.3647, avec une précision jusqu'à la dix millième d'une unité monétaire.
- **float :** Données numériques à virgule flottante de -1.79E + 308 à 1.79E + 308.
- **real :** Données numériques à virgule flottante de -3.40E + 38 à 3.40E + 38.
- **datetime :** Données de date et d'heure du 1er janvier 1753 au 31 décembre 9999, avec une précision d'une centaine de secondes ou 3.33 millisecondes.
- **date:** date, de 0001-01-01 jusqu'à 9999-12-31
- **time:** temps, de 00:00:00.0000000 through 23:59:59.9999999
- **smalldatetime :** Données de date et d'heure du 1er janvier 1900 au 6 juin 2079, avec une précision d'une minute.

- **char :** Données de caractères non Unicode de longueur fixe avec une longueur maximale de 8,000 caractères.
- **varchar :** Données de caractères variables non Unicode avec une longueur maximale de 8,000 caractères.
- **text :** Données de caractères variables non Unicode avec une longueur maximale de 2^31 – 1 (2,147,483,647) caractères.
- **binary :** Données binaires de longueur fixe avec une longueur maximale de 8,000 octets.
- **varbinary :** Données binaires de longueur variable avec une longueur maximale de 8,000 octets
- **image :** Données binaires de longueur variable avec une longueur maximale de 2^31 – 1 (2,147,483,647) octets.

# 3-2 Contraintes de Colonnes

Les contraintes de colonnes sont: NULL, NOT NULL, UNIQUE, PRIMARY KEY, IDENTITY et DEFAULT.
la contrainte NULL indique que les valeurs NULL sont autorisées.
La contrainte NOT NULL indique que la valeur doit être fournit quand une nouvelle ligne est créée.

Exemple:

```
USE SW
CREATE TABLE EMPLOYEES
(
EmployeeNo CHAR(10) NOT NULL PRIMARY KEY,
DepartmentName CHAR(30) NOT NULL DEFAULT 'Human Resources',
FirstName CHAR(25) NOT NULL,
LastName CHAR(25) NOT NULL UNIQUE,
BirthDate DATE NOT NULL,
);
```

## 3-2-1 Contrainte IDENTITY

- Nous pouvons utiliser la contrainte de colonne facultative IDENTITY pour fournir une valeur unique et incrémentielle pour cette colonne.
- Les colonnes d'identité sont souvent utilisées avec les contraintes PRIMARY KEY pour servir d'identifiant de ligne unique pour la table.
- La propriété IDENTITY peut être attribuée à une colonne de type de données tinyint, smallint, int, decimal ou numeric.
- Cette contrainte :
    - Génère des numéros séquentiels

- N'impose pas l'intégrité de l'entité
- Seule une colonne peut avoir la propriété IDENTITY
- Doit être définie comme un type de données entier, numérique ou décimal
- Ne peut pas mettre à jour une colonne avec la propriété IDENTITY
- Ne peut pas contenir de valeurs NULL
- Ne peut pas lier des valeurs par défaut et des contraintes par défaut à la

Pour IDENTITY[(seed, increment)] :

- Seed - la valeur initiale de la colonne d'identité
- Increment - la valeur à ajouter à la dernière colonne incrémentée

Nous utiliserons un autre exemple de base de données pour illustrer davantage les instructions SQL DDL en créant la table tblHotel dans cette base de données HOTEL.

```
USE HOTEL;
CREATE TABLE tblHotel (
   HotelNo Int not NULL IDENTITY (1,1) PRIMARY KEY,
   Name Char(50) NOT NULL,
   Address Char(50) NULL,
   City Char(25) NULL,
);
```

# 3-3 Contraintes de Table

Les contraintes de table sont identifiées par le mot-clé CONSTRAINT et peuvent être utilisées pour implémenter diverses contraintes telles que :

- **UNIQUE constraint :** Empêche l'entrée de valeurs en double dans une colonne.
- **PRIMARY KEY constraint**: Définit une colonne, ou une combinaison de colonnes comme clé primaire.
- **FOREIGN KEY constraint :** Définit une colonne, ou une combinaison de colonnes, dont les valeurs correspondent à la clé primaire d'une autre table.
- **CHECK constraint :** Restreint les valeurs pouvant être saisies dans une table.

## 3-3-2 Contrainte UNIQUE

- La contrainte UNIQUE empêche l'entrée de valeurs en double dans une colonne.
- Les contraintes PK et UNIQUE sont utilisées pour imposer l'intégrité de l'entité.
- Plusieurs contraintes UNIQUE peuvent être définies pour une table.
- Lorsqu'une contrainte UNIQUE est ajoutée à une table existante, les données existantes sont toujours validées.
- Une contrainte UNIQUE peut être placée sur des colonnes qui acceptent des valeurs nulles.

- Une seule ligne peut être NULL.
- Une contrainte UNIQUE crée automatiquement un index unique sur la colonne sélectionnée.

Syntaxe générale pour la contrainte UNIQUE :

```
[CONSTRAINT constraint_name] UNIQUE [CLUSTERED | NONCLUSTERED] (col_name [,
col_name2 […, col_name16]]) [ON segment_name]
```

Exemple d'utilisation de la contrainte UNIQUE :

```
CREATE TABLE EMPLOYEES (
   EmployeeNo CHAR(10) NOT NULL,
   firstname char(50),
   lastname char(50),
   CONSTRAINT emp_unique UNIQUE(EmployeeNo,lastname)
);
```

## 3-3-3 Contrainte PRIMARY KEY

```
USE SW
CREATE TABLE EMPLOYEES
(
EmployeeNo CHAR(10) NOT NULL UNIQUE,
DepartmentName CHAR(30) NOT NULL DEFAULT 'Human Resources',
FirstName CHAR(25) NOT NULL,
LastName CHAR(25) NOT NULL,
BirthDate DATE NOT NULL,
CONSTRAINT Employee_PK PRIMARY KEY(EmployeeNo,LastName)
);


USE SW
CREATE TABLE PROJECT
(
ProjectID Int NOT NULL IDENTITY (1000,100),
ProjectName Char(50) NOT NULL,
Department Char(35) NOT NULL,
MaxHours Numeric(8,2) NOT NULL DEFAULT 100,
StartDate DateTime NULL,
EndDate DateTime NULL,
CONSTRAINT ASSIGNMENT_PK PRIMARY KEY(ProjectI)
);
```

## 3-3-4 Contrainte FOREIGN KEY

- La contrainte FOREIGN KEY (FK) définit une colonne ou une combinaison de colonnes dont les valeurs correspondent à la PRIMARY KEY (PK) d'une autre table.
- Les valeurs dans une FK sont automatiquement mises à jour lorsque les valeurs PK dans la table associée sont mises à jour/changées.
- Les contraintes FK doivent faire référence à la PK ou à la contrainte UNIQUE d'une autre table.
- Le nombre de colonnes pour FK doit être le même que pour la PK ou la contrainte UNIQUE.
- Si l'option WITH NOCHECK est utilisée, la contrainte FK ne validera pas les données existantes dans une table.
- Aucun index n'est créé sur les colonnes qui participent à une contrainte FK.

Syntaxe générale pour la contrainte FOREIGN KEY :

```
[CONSTRAINT constraint_name] [FOREIGN KEY (col_name [, col_name2 […,
col_name16]])] REFERENCES [owner.]ref_table [(ref_col [, ref_col2 […,
ref_col16]])]
```

- Dans cet exemple, le champ HotelNo dans la table tblRoom est une FK vers le champ HotelNo dans la table tblHotel précédemment montrée.

```
USE HOTEL
CREATE TABLE tblRoom (
   HotelNo Int NOT NULL  IDENTITY(1,1),
   RoomNo Int NOT NULL,
   Type Char(50) NULL,
   Price  Float NULL,
   CONSTRAINT pk PRIMARY KEY (HotelNo, RoomNo),
   CONSTRAINT fk FOREIGN KEY (HotelNo) REFERENCES tblHotel(HotelNo)
);
```

## 3-3-5 Contrainte CHECK

- La contrainte CHECK restreint les valeurs pouvant être entrées dans une table.
- Elle peut contenir des conditions de recherche similaires à une clause WHERE.
- Elle peut faire référence à des colonnes dans la même table.
- La règle de validation des données pour une contrainte CHECK doit évaluer une expression booléenne.
- Elle peut être définie pour une colonne qui a une règle liée.

Syntaxe générale pour la contrainte CHECK :

```
[CONSTRAINT constraint_name] CHECK [NOT FOR REPLICATION] (expression)
```

Dans cet exemple, le champ Type est restreint à avoir uniquement les types 'Single', 'Double',

```
USE HOTEL;

CREATE TABLE tblRoom2 (
   HotelNo Int NOT NULL,
   RoomNo Int NOT NULL,
   Type Char(50) NULL,
   Price Float NULL,
   CONSTRAINT prim_key PRIMARY KEY (HotelNo, RoomNo),
   CONSTRAINT f_key FOREIGN KEY (HotelNo) REFERENCES tblHotel(HotelNo),
   CONSTRAINT Valid_Type CHECK (Type IN ('Single', 'Double'))
);
```

Dans ce deuxième exemple, la date d'embauche de l'employé doit être antérieure au 1er janvier 2004 ou avoir une limite de quota de 300 000.

```
CREATE TABLE SALESREPS (
   Empl_num Int Not Null
   CHECK (Empl_num BETWEEN 101 and 199),
   Name Char (15),
   Age Int CHECK (Age >= 21),
   Quota Float CHECK (Quota >= 0.0),
   HireDate DateTime,
   CONSTRAINT QuotaCap CHECK ((HireDate < '2004-01-01') OR (Quota <=300000))
);
```

# 4- ALTER TABLE

- Vous pouvez utiliser des instructions ALTER TABLE pour ajouter et supprimer des contraintes.
- ALTER TABLE permet aussi d'ajouter et de supprimer, et de modifier des colonnes.
- Lorsqu'une contrainte est ajoutée, toutes les données existantes sont vérifiées pour d'éventuelles violations.

## 4-1- Ajouter une contrainte

Dans cet exemple, nous utilisons l'instruction ALTER TABLE pour ajouter la propriété IDENTITY à une colonne appelée Name.

- On suppose que la contrainte "unqName" n'existe pas encore dans la base de données hotel:

```
USE HOTEL;
ALTER TABLE tblHotel ADD CONSTRAINT unqName UNIQUE (Name);
```

- On suppose que la contrainte "df_city" n'existe pas encore dans la base de données hotel. Et qu'il n'y a pas de contrainte **default** définit sur le champ "City" de la table "tblHotel"

```
USE HOTEL ALTER TABLE tblHotel
Add CONSTRAINT df_city DEFAULT 'Vancouver' FOR City;
```

- On suppose que la contrainte "pk_key" n'existe pas encore dans la base de données hotel. Et qu'il n'y a pas de contrainte **primary** définit sur la table

```
USE HOTEL ALTER TABLE tblHotel
Add CONSTRAINT pk_key PRIMARY KEY(HotelNo);
```

- On suppose que la contrainte "fk_key" n'existe pas encore dans la base de données hotel.

```
USE HOTEL ALTER TABLE tblHotel
Add CONSTRAINT fk_key foreing KEY(HotelNo) references  category(id_category)
```

- On suppose que la contrainte "ck_key" n'existe pas encore dans la base de données hotel.

```
USE HOTEL ALTER TABLE category
Add CONSTRAINT ck_key check (id_category < 100 and id_category > 0)
```

- ajouter les contraintes colonne NULL et NOT NULL:
- on suppose que la colonne id_category existe déja dans la table category:

```
alter table category
alter column id_category int not null;
```

- on suppose que la colonne description existe déja dans la table category, et qu'il n'y a pas de contrainte de clé primaire sur cette colonne:

```
alter table category
alter column description char(25) null;
```

# 4-2- Supprimer une contrainte

- On suppose que la contrainte "unqName" existe (dans la table tblHotel de la base de données Hotel):

```
USE HOTEL;
ALTER TABLE tblHotel DROP CONSTRAINT unqName;
```

## 4-3- Ajouter une colonne

- Dans l'exemple suivant, nous utiliserons l'instruction ALTER TABLE pour ajouter la colonne NumBeds avec la contrainte NOT NULL dans la table tblRoom
- On suppose que la colonne "tblRoom" n'existe pas encore dans la table tblRoom de la bd hotel:

```
use hotel;
ALTER TABLE tblRoom
ADD NumBeds int NOT NULL;
```

## 4-4- Supprimer une colonne

- On suppose que la colonne "NumBeds" existe (dans la table tblRoom de la base de données hotel):
- On suppose aussi que la colonne NumBeds n'est impliquée dans aucune contrainte.

```
use hotel;
ALTER TABLE tblRoom
DROP Column NumBeds;
```

## 4-6- Changer le type d'une colonne

- On suppose que la colonne "BirthDate" existe (dans la table Employees de la base de données sw):

```
USE sw;
ALTER TABLE Employees
ALTER COLUMN BirthDate date;
```

# 5- Remarques sur certaines contraintes de colonne

- une colonne ne peut avoir une contrainte colonne UNIQUE et une contrainte de colonne PRIMARY KEY au même moment (dans la même ligne).
- la contrainte IDENTITY ne peut être utilisée qu'avec des colonnes de type: int, bigint, smallint, tinyint, decimal et numeric avec s=0

- la contrainte de colonne PRIMARY KEY ne peut être attribuée qu'à une seule colonne. Pour pouvoir sélectionner plusieurs colonnes, il faut utiliser la contrainte de table (PRIMARY KEY)
- on ne peut pas utiliser à la fois la contrainte PRIMARY KEY de colonne et de table en même temps.

# 6- DROP TABLE

- La commande DROP TABLE permet de supprimer une table de la base de données.
- Assurez-vous d'avoir sélectionné la base de données correcte.

```
DROP TABLE employees;
```

L'exécution de l'instruction SQL DROP TABLE ci-dessus supprimera la table employees de la base de données.

# Références


- Watt, Adrienne, and Nelson Eng. *Database design*. 2nd Edition. BCcampus, 2014
- Microsoft 2024, Data types (Transact-SQL), accédé le 4 Novembre 2024, https://learn.microsoft.com/en-us/sql/t-sql/data-types/data-types-transact-sql?view=sql-server-ver16.