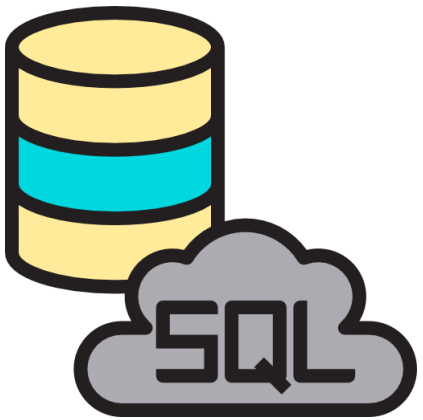
Implémentation de la base de données

420-1B2-SW Développement avec bases de données

Implémentation de la base de données

- SQL (Structured Query Language) est le langage de base de données à l'aide duquel nous pouvons effectuer certaines opérations sur la base de données(BDD) existante et nous pouvons également utiliser ce langage pour créer une BDD.

- SQL utilise certaines commandes/instructions telles que Create, Drop, Insert, etc. pour effectuer les tâches requises.



Implémentation de la base de données

- Ces commandes/instructions SQL sont principalement classées en cinq catégories :

- DDL (**D**ata **D**efinition **L**anguage) Langage de définition de données
- DQL (**D**ata **Q**uery **L**anguage) Langage de requête de données
- DML (**D**ata **M**anipulation **L**anguage) Langage de manipulation de données
- DCL (**D**ata **C**ontrol **L**anguage) Langage de contrôle des données
- TCL (**T**ransaction **C**ontrol **L**anguage) Langage de contrôle des transactions

Implémentation de la base de données

DDL (Data Definition Language)

- DDL se compose des commandes/instructions SQL qui peuvent être utilisées pour **définir** la base de données.
- Il traite simplement des descriptions du schéma de la base de données(relationnel) et est utilisé pour créer et modifier la structure des objets de la base de données(tables, contraintes d'intégrité '*clé primaire, clé étrangère*', etc).

Implémentation de la base de données

DDL (Data Definition Language)

Exemples de commandes DDL :

- **CREATE** est utilisé pour créer la base de données ou ses objets (comme la table)
- **DROP** est utilisé pour supprimer la base de données et ses objets.
- **ALTER** est utilisé pour modifier la structure de la base de données (comme modifier les attributs d'une table)

Définition de la table avec les instructions DDL

420-1B2-SW Développement avec bases de données

Définition de la base de données

Les instructions DDL(Data Definition Language) permettent de définir la structure des données, CREATE DATABASE permettra de créer une base de données :

```
-- Syntaxe simplifiée la plus fréquente lorsque les options avancées ne sont pas utilisées
CREATE DATABASE nom_base_données;
```

exemples :

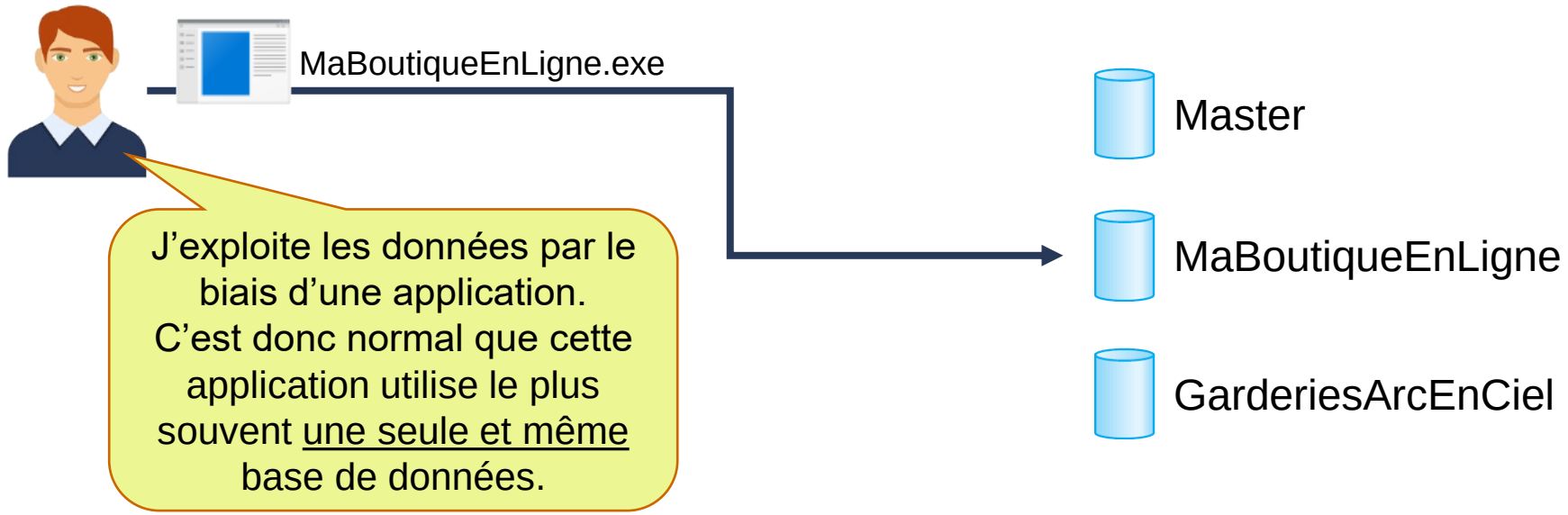
```
CREATE DATABASE MaBoutiqueEnLigne;

CREATE DATABASE GarderiesArcEnCiel;

CREATE DATABASE Resto_LesGrosBurgers;
```

Définition de la base de données

Un serveur pouvait héberger plusieurs bases de données. La connexion d'un utilisateur exploite cependant **une seule base de données à la fois**. Cela signifie que toutes les instructions SQL lancées par un utilisateur ciblent uniquement la base de données assignée à son **contexte**.



Définition de la base de données

– En appliquant l'intégrité référentielle (la contrainte de clé étrangère) le SGBD s'assure que la clé primaire référencée par la clé étrangère existe faute de quoi il (le SGBD) soulèvera une erreur :

Code	Client	NoProvince
CHOPS	Chop-suey Chinese	2
COMMI	Comércio Mineiro	4
ERNSH	Ernst Handel	2
GODOS	Godos Cocina Típica	8
GREAL	Great Lakes Food Market	1
KOENE	Königlich Essen	6
LEHMS	**Lehmanns Marktstand**	**15**
MEREP	Mère Paillarde	1
…	…	

NoProvince	Province
1	Alberta
2	Colombie-Britannique
3	Île-du-Prince-Édouard
4	Manitoba
5	Nouveau-Brunswick
6	Nouvelle-Écosse
7	Nunvut
8	Ontario
…	…

Définition de la base de données

– L’instruction DDL DROP DATABASE vous permettra éventuellement de supprimer une base de données :

```
DROP DATABASE nom_base_données;
```

```
DROP DATABASE MaBoutiqueEnLigne;
```

Définition de la table avec les instructions DDL

420-1B2-SW Développement avec bases de données

Définition de la table

– L'instruction DDL CREATE TABLE permet de créer une table :

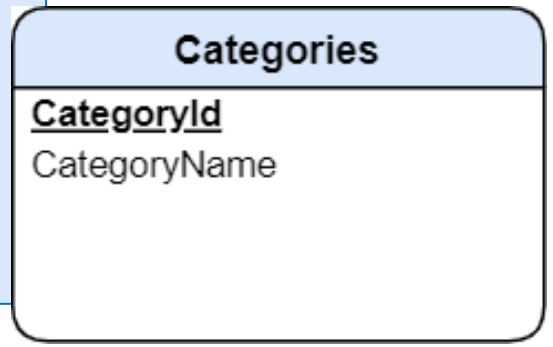
```
-- Syntaxe simplifiée la plus fréquente lorsque les options avancées ne sont pas utilisées
CREATE TABLE nom_table (
        { <definition_colonne> }
        [ ,...n ]
);
```

<definition_colonne> = nom_colonne <type_donnee> [contrainte]

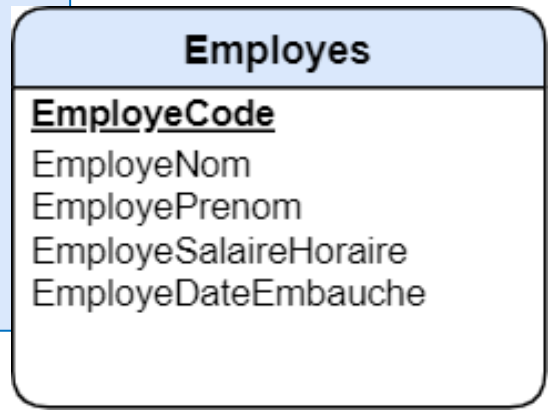
Définition de la table

– Exemples simples de l'instruction CREATE TABLE :

```
USE MaBoutiqueEnLigne
CREATE TABLE Categories (
      CategoryId int PRIMARY KEY,
      CategoryName varchar(50) NOT NULL
);
```




```
USE GarderiesArcEnCiel
CREATE TABLE Employes (
      EmployeCode char(5) PRIMARY KEY,
      EmployetNom  varchar(50) NOT NULL,
      EmployePrenom  varchar(50) NOT NULL,
      EmployeSalaireHoraire  float NULL,
      EmployeDateEmbauche  datetime NULL
);
```



Définition de la table

– Une colonne peut accepter les **valeurs nulles** ou non en adjoignant le modificateur NULL ou NOT NULL à la définition de la colonne :

```
CREATE TABLE nom_table (
    nom_colonne   <type_donnee>  NOT NULL,
    nom_colonne   <type_donnee>  NULL,
    nom_colonne   <type_donnee>
);
```

NULL par défaut si non-spécifié.

```
CREATE TABLE Products (
        ProductId int PRIMARY KEY,
        ProductName nvarchar(40) NOT NULL,
        UnitPrice int NULL,
        etc…
);
```

Une clé primaire ne peut pas être *nullable*.

Définition de la table

– Une colonne peut constituer la **clé primaire** de la table en adjoignant le modificateur PRIMARY KEY à la définition de la colonne :

```
CREATE TABLE nom_table (
    nom_colonne   <type_donnee>   PRIMARY KEY
);
```

Une clé primaire ne peut pas être *nullable*.

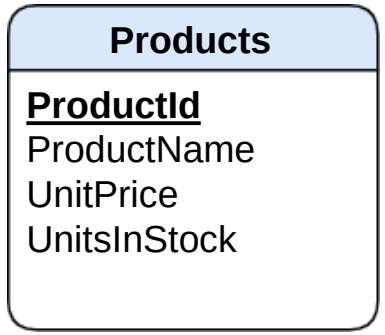
```
CREATE TABLE Products (
        ProductId int PRIMARY KEY,
        ProductName nvarchar(40) NOT NULL,
        UnitPrice int NULL,
        UnitsInStock smallint NULL,
        etc…
);
```

Définition de la table

– Une autre syntaxe permettant d'assigner une colonne pour clé primaire, utilisant le mot-clé PRIMARY KEY à la fin de la définition de la table :

```
CREATE TABLE nom_table (
      nom_colonne   <type_donnee>,
      PRIMARY KEY (  nom_colonne, […n] )
);
```

```
CREATE TABLE Products (
            ProductId int,
            ProductName varchar(40) NOT NULL,
            UnitPrice int NULL,
            UnitsInStock smallint NULL,
            etc… ,
            PRIMARY KEY ( ProductId )
);
```



Définition de la table

– Cette même syntaxe permet la création d'une clé primaire composite, en indiquant le nom des diverses colonnes entre les parenthèses du mot-clé PRIMARY KEY :

```
CREATE TABLE OrderDetails (
        OrderId int,
        ProductId int,
        UnitPrice int not null,
        Quantity smallint not null,
        Discount real not null,
        PRIMARY KEY ( OrderId, ProductId )
);
```

Définition de la table

– Pour créer un champ **auto incrémenté**, très utile pour les clés artificielles qui doivent se voir attribuer des valeurs séquentielles :

```
CREATE TABLE Products (
        ProductId int PRIMARY KEY auto_increment,
        ProductName nvarchar(40) NOT NULL,
        UnitPrice int NULL,
        UnitsInStock smalli... NULL,
        etc
);
```

Racine = 1
attribuée à la première valeur insérée

Incrément = 1
à chaque nouvelle valeur attribuée

– Notez que cette syntaxe n'est pas normalisée et diffère largement d'un manufacturier à un autre.

Définition de la table

– D'autres parts, il est possible de créer une colonne qui agit en **clé étrangère** à l'aide du modificateur FOREIGN KEY auquel il suffit de spécifier le nom de la table et de sa colonne référencée :

```
CREATE TABLE Products (
        ProductId int PRIMARY KEY,
        ProductName nvarchar(40) NOT NULL,
        SupplierId int FOREIGN KEY REFERENCES Suppliers( SupplierId ),
        CategoryId int FOREIGN KEY REFERENCES Categories( CategoryId ),
        UnitPrice int NULL,
        UnitsInStock smallint NULL,
        etc…
);
```

Définition de la table

– Une colonne peut posséder la définition d'une **valeur par défaut** qui sera assigné à tout nouvel enregistrement pour lequel aucune valeur n'est explicitement spécifiée :

```
CREATE TABLE nom_table (
    nom_colonne <type_donnee> [ NULL | NOT NULL ]  DEFAULT expression
);
```

```
CREATE TABLE ProductsPriceByCustomer (
        CustomerCode char(10) NOT NULL,
        ProductId int NOT NULL,
        Ratio float NOT NULL DEFAULT 1.0,
);
```

Ratio par défaut est 1.0,
à moins qu'une autre valeur
ne soit spécifiée

Définition de la table

– L'instruction DDL DROP TABLE vous permettra éventuellement de supprimer une table.

```
DROP TABLE nom_table;
```

```
DROP TABLE Invoices;
```

– Une table ne pourra pas être supprimée si l'un de ses attributs est référencé par une relation d'une autre table.

Implémentation de la base de données

Exercices

420-1B2-SW Développement avec bases de données

Exercice

Fournir le schéma relationnel puis implémenter le modèle relationnel:

BDD: Location d'appartements