

# Les bases de données et le langage SQL

Formation : BTS | Module : SQL



## Objectifs

### 0. Introduction

0.1 **Base de données? SQL?** Qu'est ce que c'est?

0.2 Pourquoi ce cours est-il utile?

### 1. Les bases de données :

1.1 Comprendre le rôle des bases de données dans une application

1.2 Connaitre les différents types de base de données

1.3 Comprendre les concepts d'une base de données **relationnelle**

1.4 *TP*

### 2. Le langage **SQL** :

2.1 Comprendre l'intérêt et le rôle du langage **SQL** ?

2.2 Apprendre la syntaxe du langage SQL

2.3 *TP : DQL, DML, DDL*

### 3. Ressources



# 0. INTRODUCTION



# Les bases de données et le langage SQL → 0. Introduction

## 0.1 Base de données? SQL? Qu'est ce que c'est?

- **Base de données (Database) :**

*Nous parlerons de **SGBDR***

*Permet de stocker et de retrouver l'intégralité de données brutes ou d'informations en rapport avec un thème ou une activité ; celles-ci peuvent être de natures différentes et plus ou moins reliées entre elles.*

- **SQL (Structured Query Language) :**

*Le SQL est un langage informatique normalisé servant à exploiter des bases de données relationnelles. La partie langage de manipulation des données de SQL permet de **rechercher**, **d'ajouter**, de **modifier** ou de **supprimer** des données dans les bases de données relationnelles.*

*Nous parlerons de **CRUD***

# Les bases de données et le langage SQL → 0. Introduction

## 0.1 Base de données? Stocker des données?

### ■ Différentes utilisations des fichiers:

- Fichier programme
- Bibliothèque de fonctions
- Fichier périphérique
- Fichier texte brut
- **Fichier de données**

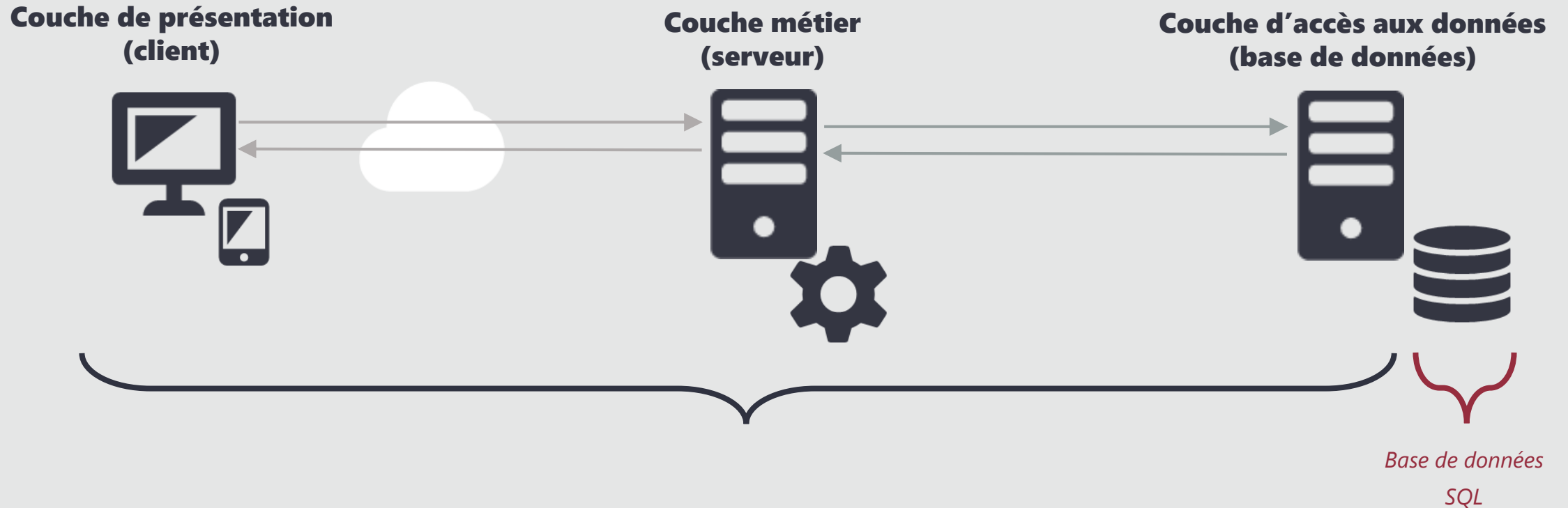


Type	Observation
Fichier programme	contient du code exécutable par le Système d'exploitation
Bibliothèque de fonctions	contient du code exécutable pouvant être utilisé par un programme
Fichier texte	contient des caractères ASCII
Fichier périphérique	adressage de périphériques (Unix)
<b>Fichier de données</b>	<b>stockage d'informations d'une application</b>

# Les bases de données et le langage SQL → 0. Introduction

## 0.2 Pourquoi ce cours est-il utile ?

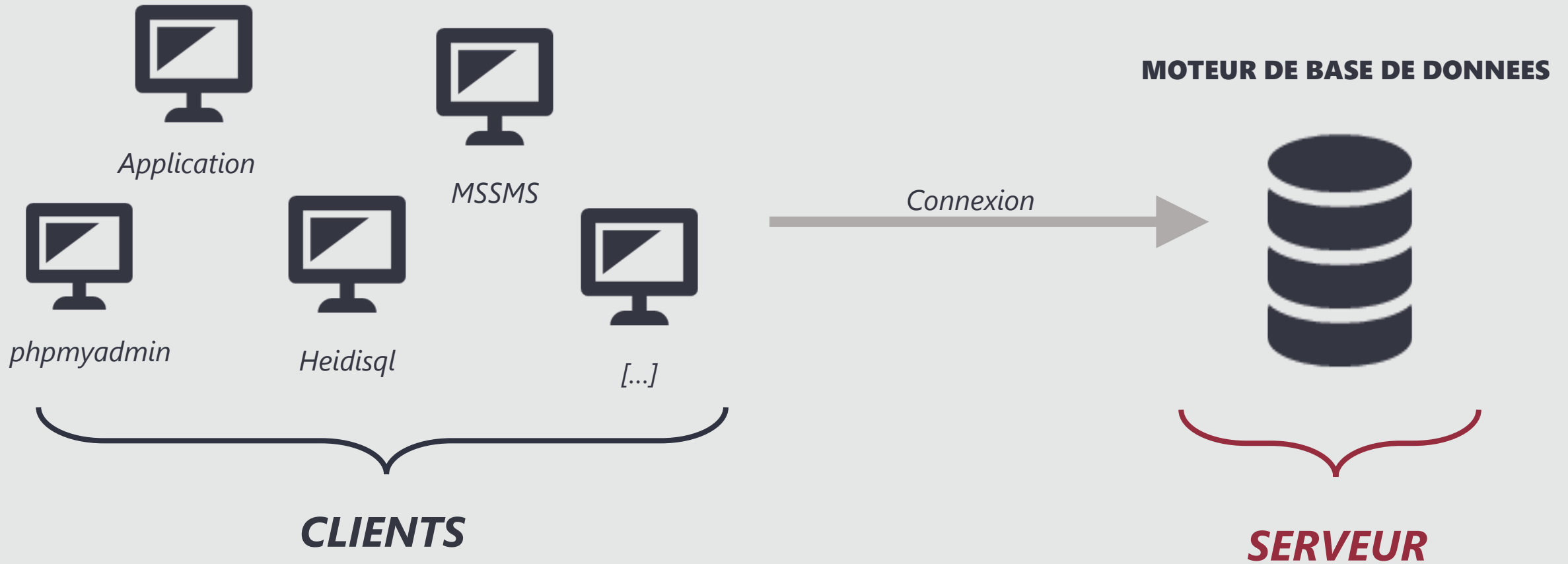
Architecture type d'une application :



# Les bases de données et le langage SQL → 0. Introduction

## 0.2 Pourquoi ce cours est-il utile ?

Ne pas confondre CLIENT et SERVEUR :



# 1. LES BASES DE DONNÉES





# Les bases de données et le langage SQL → 1. Les bases de données

## 1.0 Comprendre le rôle des bases de données dans une application

### **Une base de données permet de :**

- Stocker les données de l'application (informations utilisateurs, contenus, paramètres etc.)
- Organiser, hiérarchiser, structurer les données pour faciliter :
  - La cohérence des données
  - la manipulation (lecture, écriture, mise à jour et suppression),
  - la maintenance,
  - les performances
- Sécuriser les données (contrôles d'accès, sauvegardes, ...)
- Partager les données

### **Une base de données relationnel permet aussi de :**

- Consolider les données stockées
- Eviter la redondance d'informations
- Créer des relations entre les différentes données

# Les bases de données et le langage SQL → 1. Les bases de données

## 1.1 Comprendre le fonctionnement

### **Les types de données :**

- Les données permanentes (données de base) :
  - Données de base de l'application
  - Exemple : fiches clients d'un magasin
- Les données de mouvement :
  - Données se référant aux données permanentes en apportant des informations supplémentaires et ponctuelles
  - Exemple : factures des clients (qté, date, etc.)
- Les données de travail :
  - Données créées par l'application (données pouvant être volumineuse)
  - Exemple : dates d'éditions des factures
- Les données d'archive
  - Données permanentes, de mouvement et de travail non modifiables à des fins de sauvegarde
  - Exemple : factures des années précédentes

# Les bases de données et le langage SQL → 1. Les bases de données

## 1.1 Comprendre le fonctionnement

### ■ Les règles appliquées aux relations

#### ■ La cohérence

- Toute valeur prise par un attribut doit appartenir au domaine sur lequel il est défini

#### ■ Unicité

- Tous les enregistrements d'une relation doivent être distincts

#### ■ Identifiant

- Attributs ou ensemble d'attributs permettant de caractériser de manière unique chaque élément de la relation
  - Clé primaire
  - Clés secondaires

#### ■ Intégrité référentielle

- Situation dans laquelle pour chaque information d'une table A qui fait référence à une information d'une table B, l'information référencée existe dans la table B
  - Clé étrangère

#### ■ Contrainte d'entité

- Toute valeur participant à une clé primaire doit être non NULL

# Les bases de données et le langage SQL → 1. Les bases de données

## 1.1 Comprendre le fonctionnement

Il existe 5 formes normales (3 dans les faits) pour s'assurer qu'un schéma est conforme au modèle relationnel **RI 10-11**

- **Première forme normale:**

- Une table est dite en première forme normale lorsque toutes les colonnes contiennent des valeurs simples

- **Deuxième forme normale**

- Une table est dite en deuxième forme normale si elle est en première forme normale et si toutes les colonnes non clés dépendent fonctionnellement de la clé primaire

- **Troisième forme normale**

- Une table est dite en troisième forme normale si elle est en deuxième forme normale et s'il n'existe pas de dépendance fonctionnelle entre deux colonnes non clé

Les bases de données et le langage SQL → 1. Les bases de données

## 1.1 Comprendre le fonctionnement

### **Les opérations sur une base de données :**

**C**reate / Créer

**R**ead / Lire

**U**pdate / Mettre à jour

**D**elete / Supprimer

Les bases de données et le langage SQL → 1. Les bases de données

## 1.2 Connaitre les différents types de base de données

### 4 grands types de base de données :

Date	Type	Observation
~1970	Modèle relationnel	Modèle le plus populaire encore aujourd'hui
~1969	Modèle réseau	Modèle de base de données complexe. Rend l'application dépendante de la structure des données
~1990	Modèle hiérarchique (objet)	Modèle abandonné très rapidement
~2009	NoSQL ( <i>not only SQL</i> )	Modèle très populaire pour le traitement de données volumineuses (Big Data par exemple)

# Les bases de données et le langage SQL → 1. Les bases de données

## 1.2 Connaître les différents types de base de données

### Les bases de données les plus populaires :

- Modèle NoSQL :

- Cassandra
- MongoDB
- Oracle NoSQL
- [...]

The Oracle NoSQL logo, consisting of the word "ORACLE" in a white, sans-serif font inside a red rectangular box.

- Modèle relationnel :

- PostgreSQL
- MySQL
- MariaDB (Fork GNU de MySQL)
- Oracle Database
- **SQL Server (Microsoft)**
- [...]

The Oracle Database logo, consisting of the word "ORACLE" in a white, sans-serif font inside a red rectangular box.

Les bases de données et le langage SQL → 1. Les bases de données

TD : Manipuler une base de données SQL Server

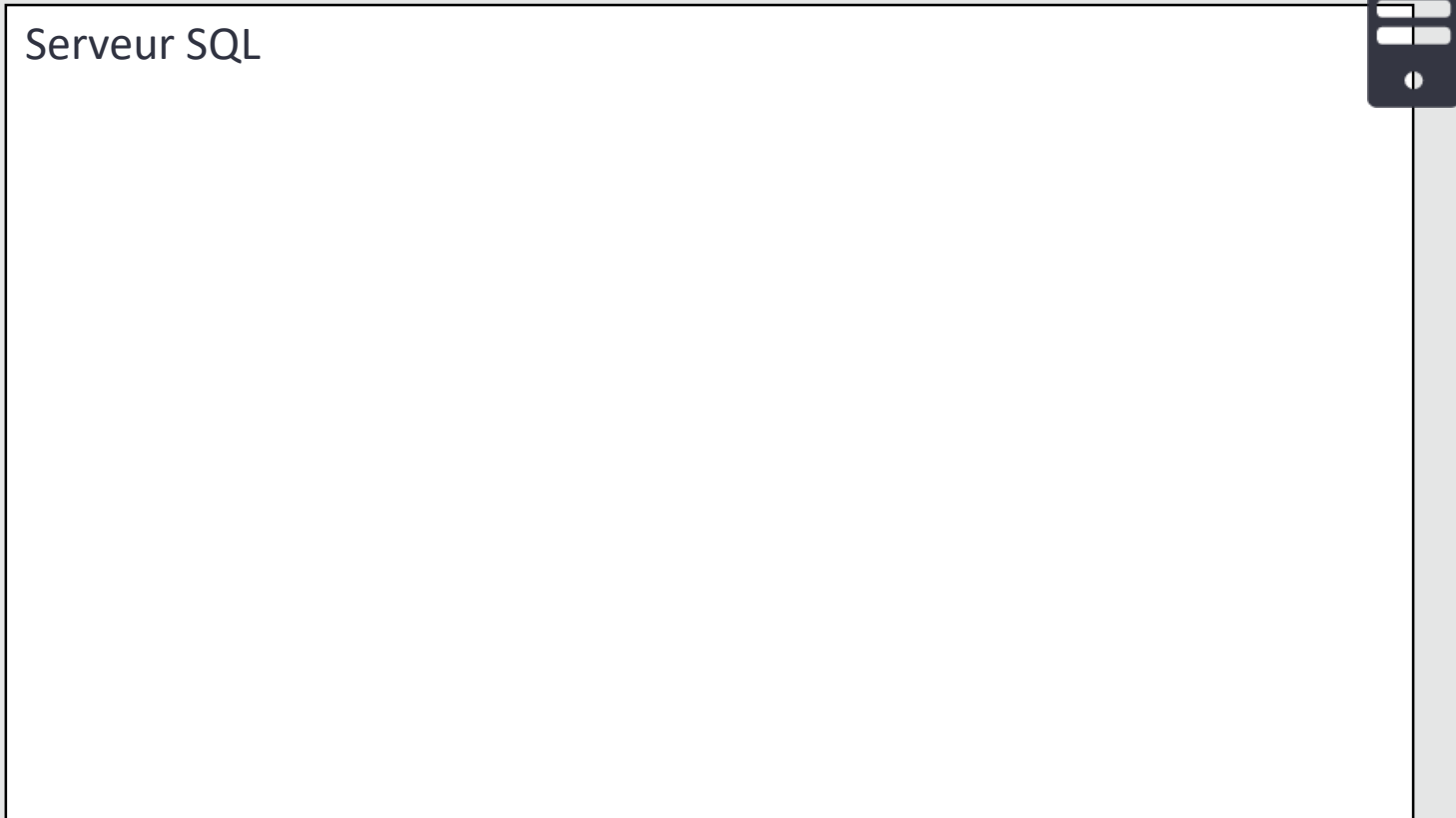
- 1. Installer et configurer SSMS**
2. Découverte de l'environnement
3. Créer la base de données UNE\_ENTREPRISE
4. Créer le schéma FACTURATION
5. Créer un utilisateur COMPTABLE1 et attribuer les droits, connexion avec COMPTABLE1
6. Créer la table FACTURES et ses attributs dans le schéma FACTURATION
7. Réaliser un CRUD sur la table FACTURES
8. Modifier la table FACTURES pour ajouter une clé primaire
9. Créer la table CLIENTS avec des contraintes
10. Créer la relation 1,\* : table FACTURES ajouter une clé étrangère
11. Réaliser un INSERT
12. Créer un graphique UML des deux tables



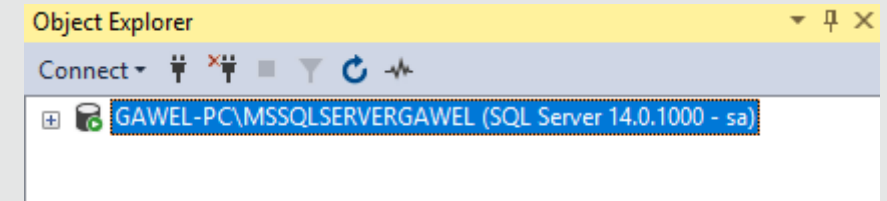
# Les bases de données et le langage SQL → 1. Les bases de données

## 1.3 Comprendre les concepts d'une base de données relationnelle

### Structure d'une base de données :



### Correspondance dans SSMS :



### Rôle :

**Moteur** de base de données.

C'est le composant logiciel central du système de gestion de base de données

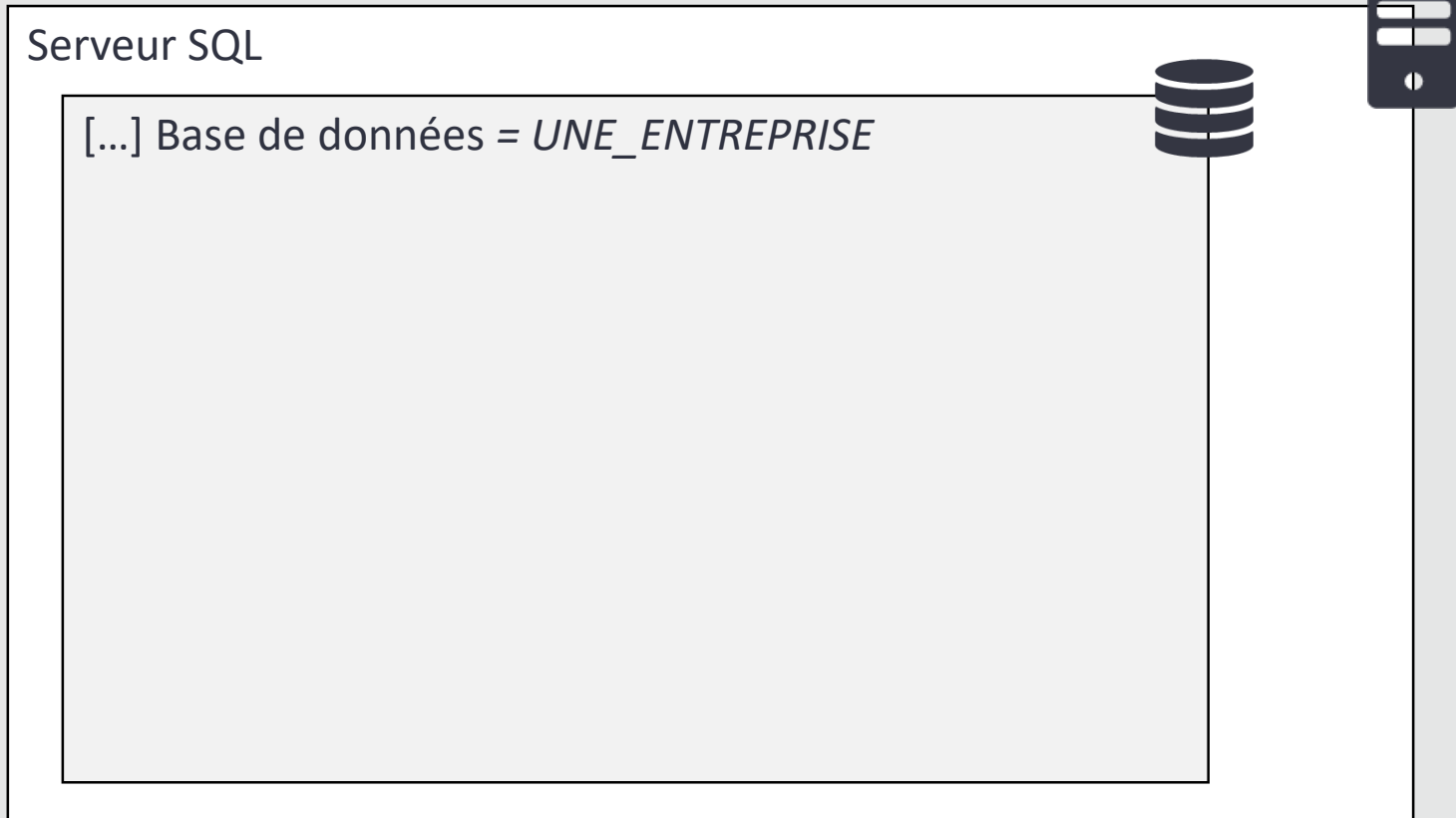
### Exemple :

SQL Server, MySQL, PostgreSQL...

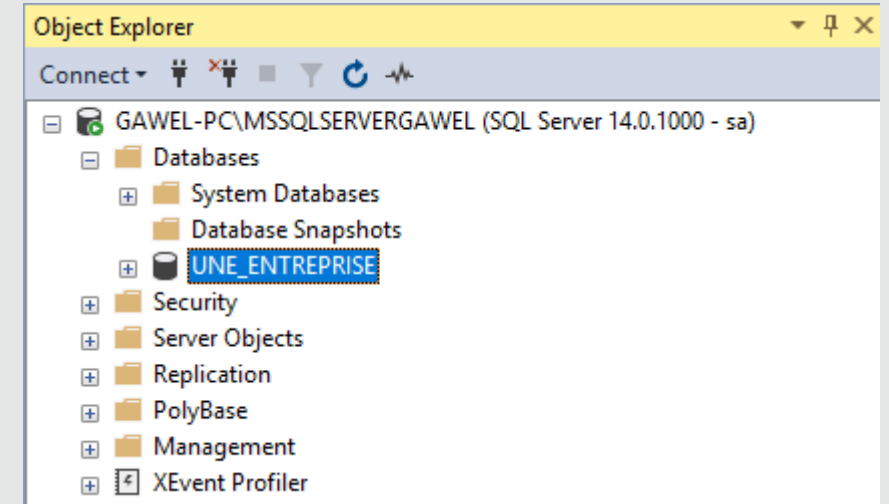
Les bases de données et le langage SQL → 1. Les bases de données

## 1.3 Comprendre les concepts d'une base de données relationnelle

### Structure d'une base de données :



### Correspondance dans SSMS :



### Rôle :

Base de données d'une application.

C'est le lieux de stockage des données pour l'application.

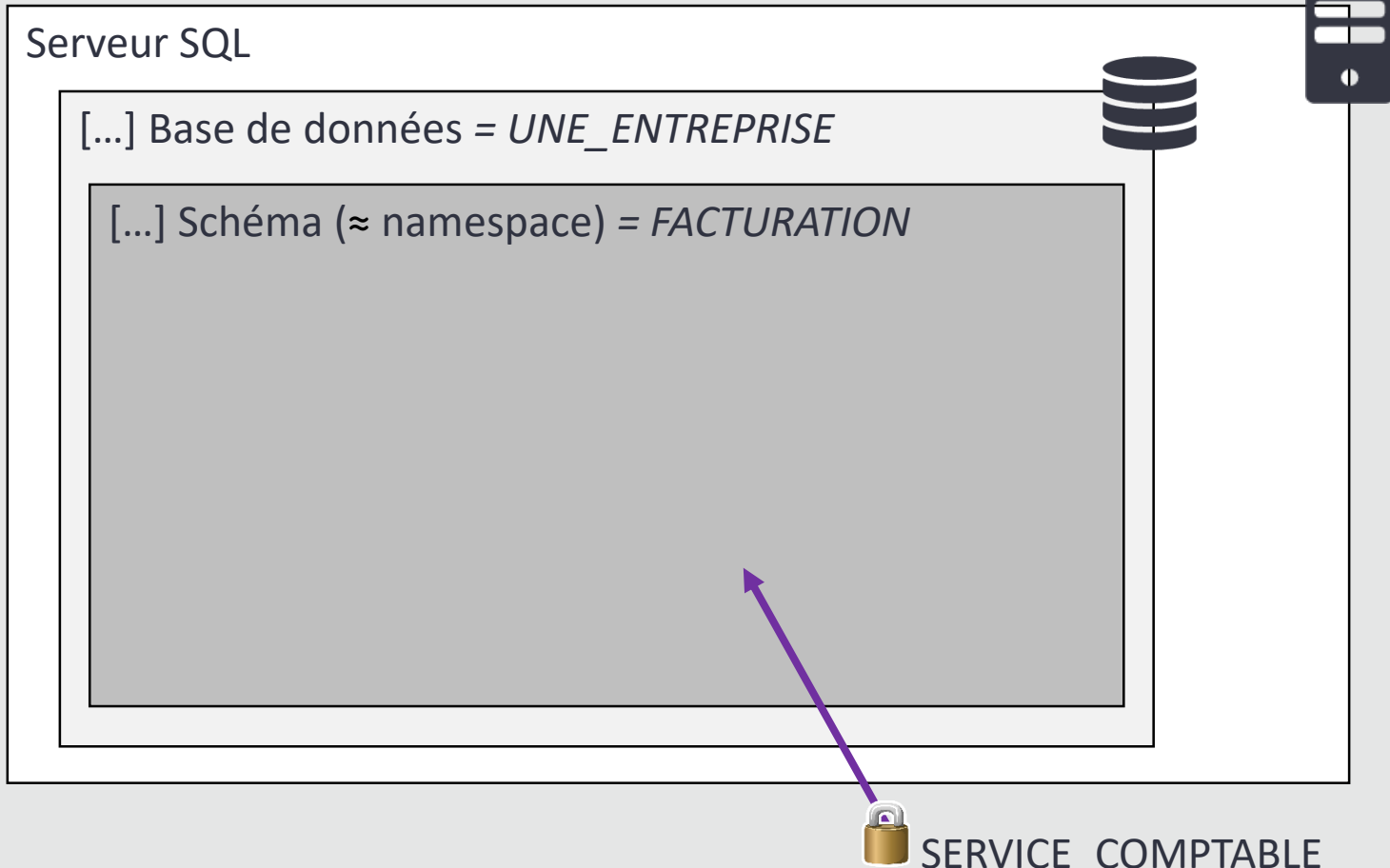
### Exemple :

La base de données d'une ENTREPRISE.

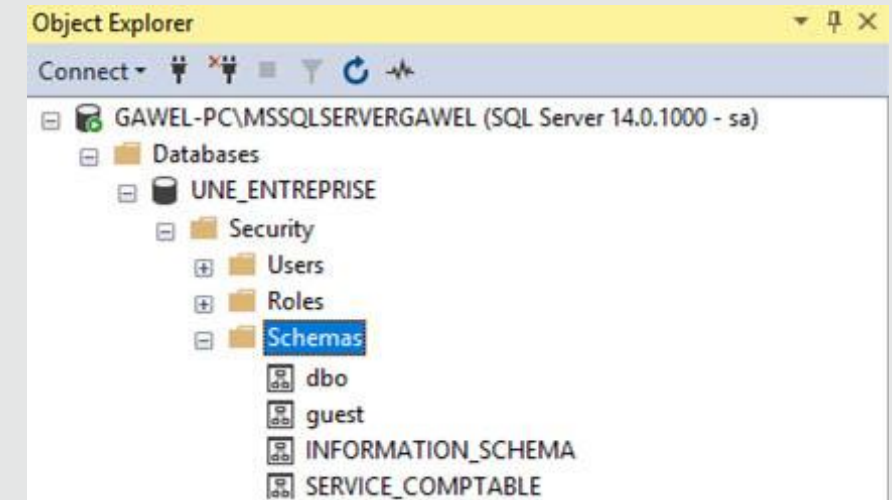
# Les bases de données et le langage SQL → 1. Les bases de données

## 1.3 Comprendre les concepts d'une base de données relationnelle

### Structure d'une base de données :



### Correspondance dans SSMS :



### Rôle :

**Schémas** d'une base de données.

C'est un regroupement logique d'objets et de fonctions (regroupement de données, créations de règles de sécurités...).

### Exemple :

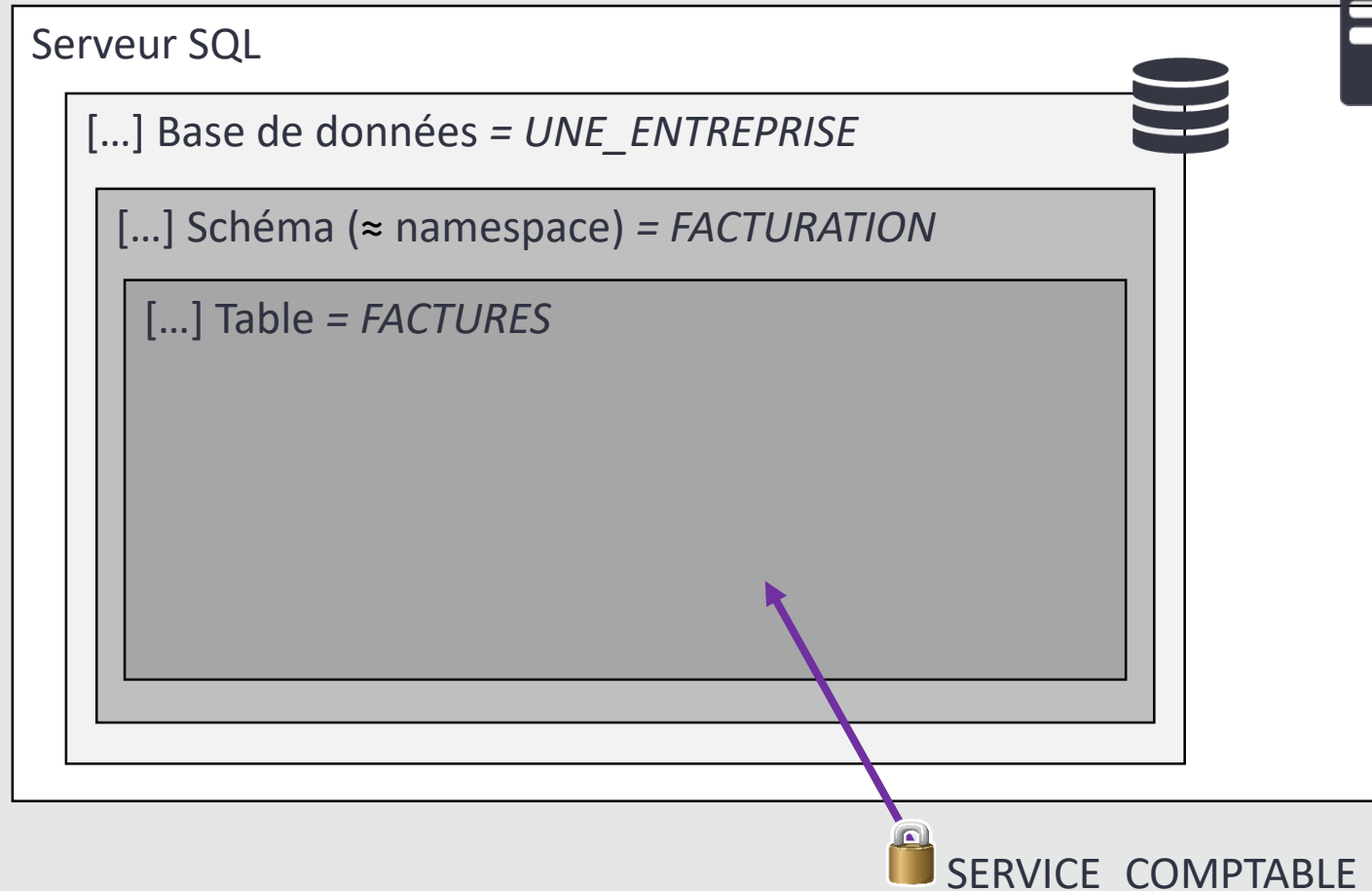
Règle : Seulement le SERVICE\_COMPTABLE est autorisé à accéder aux données de FACTURATION de l'ENTREPRISE.



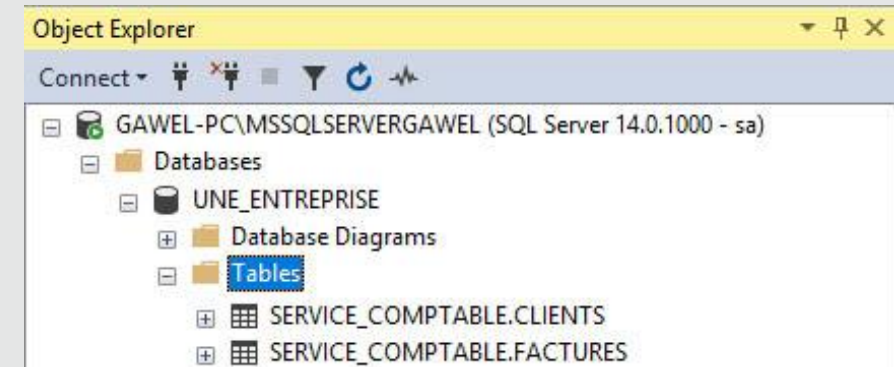
# Les bases de données et le langage SQL → 1. Les bases de données

## 1.3 Comprendre les concepts d'une base de données relationnelle

### Structure d'une base de données :



### Correspondance dans SSMS :



### Rôle :

Tables de la base de données.

Les données d'une base de données sont organisées par table. Une table représente un type de données.

### Exemple :

Une table *FACTURES* dans une base de données *ENTREPRISE*.

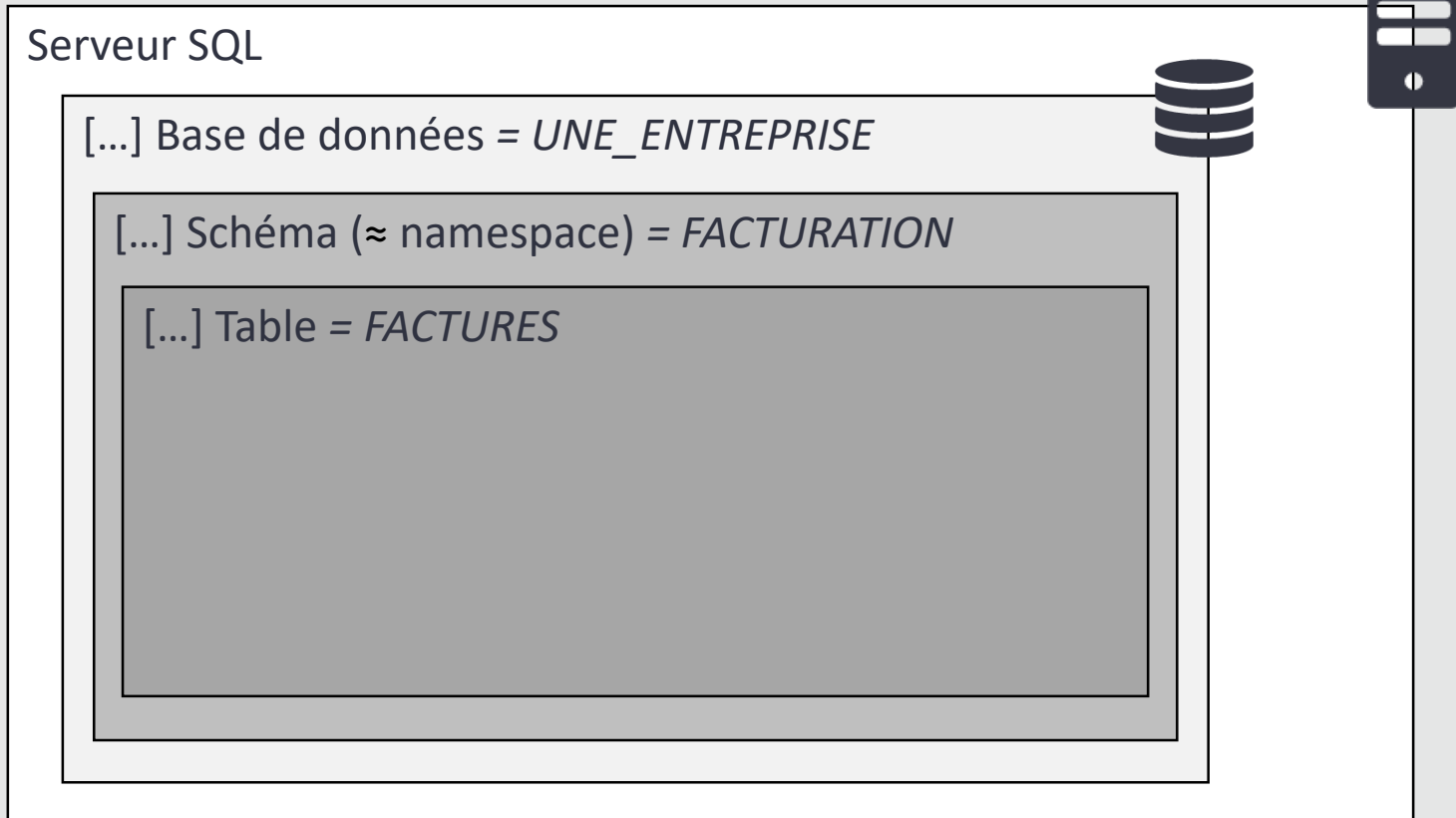
Note : La table *FACTURES* est dans le schéma *FACTURATION* pour des raisons de sécurité.



# Les bases de données et le langage SQL → 1. Les bases de données

## 1.3 Comprendre les concepts d'une base de données relationnelle

### Structure d'une base de données :



### Composition d'une table :

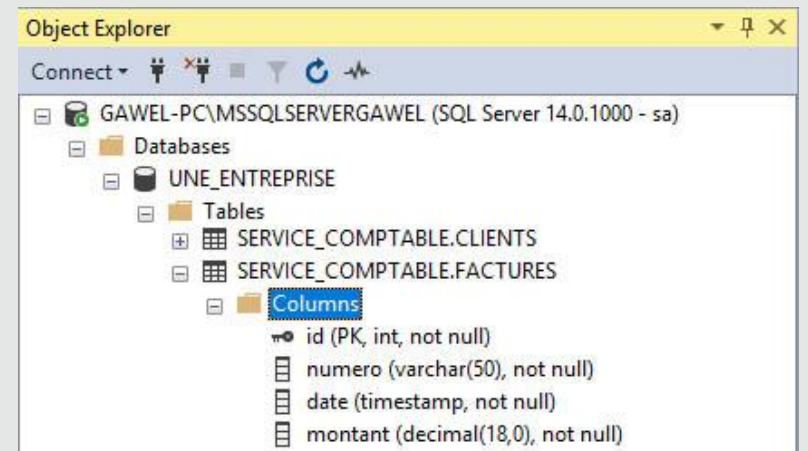
Une table est composée **d'attributs**.

Un attribut est une «colonne» permettant de stocker le « détail » d'un **enregistrement**.

Exemple :

Attributs de la table *FACTURES* :

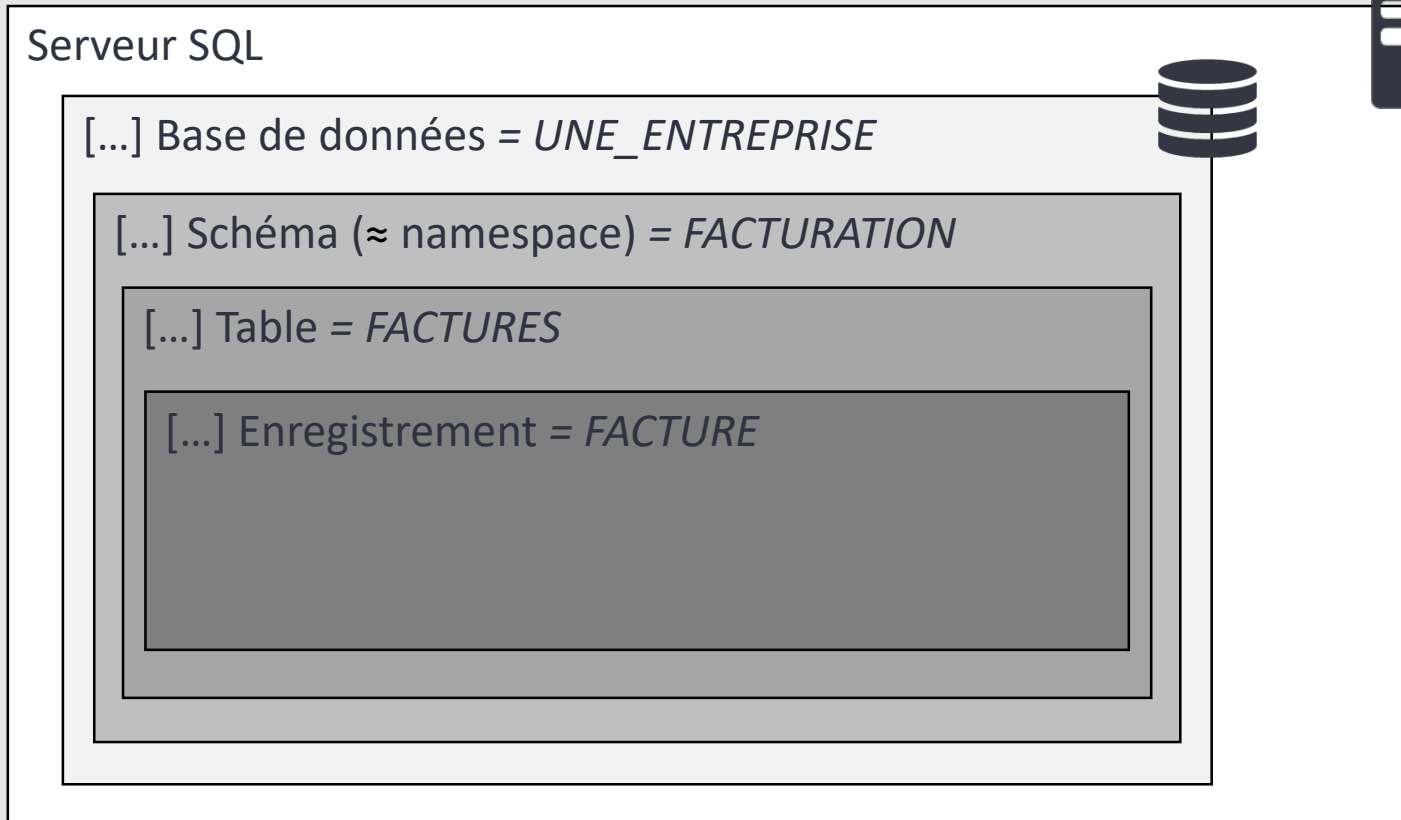
- numéro de facture,
- date,
- montant total,
- etc.



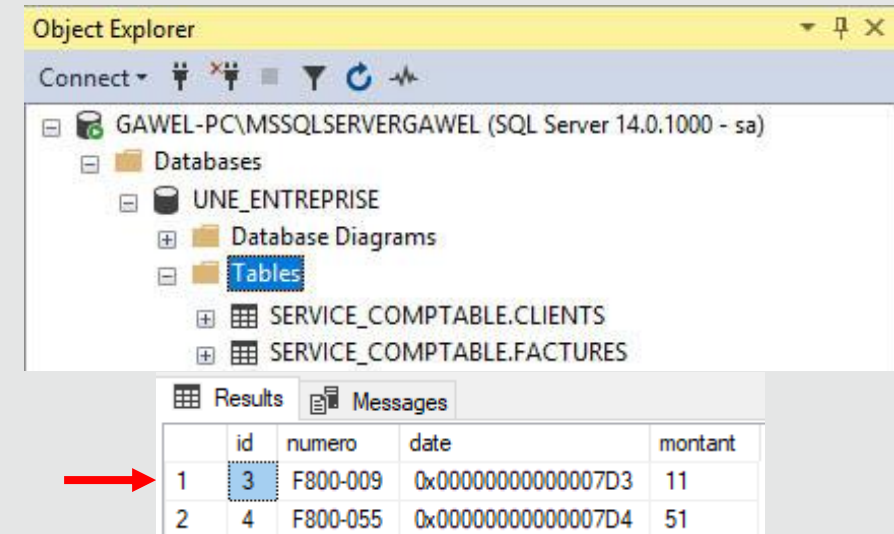
Les bases de données et le langage SQL → 1. Les bases de données

## 1.3 Comprendre les concepts d'une base de données relationnelle

### Structure d'une base de données :



### Correspondance dans SSMS :



### Rôle :

Données brutes d'un **enregistrement**.

C'est une ligne d'enregistrement de données dans une table.

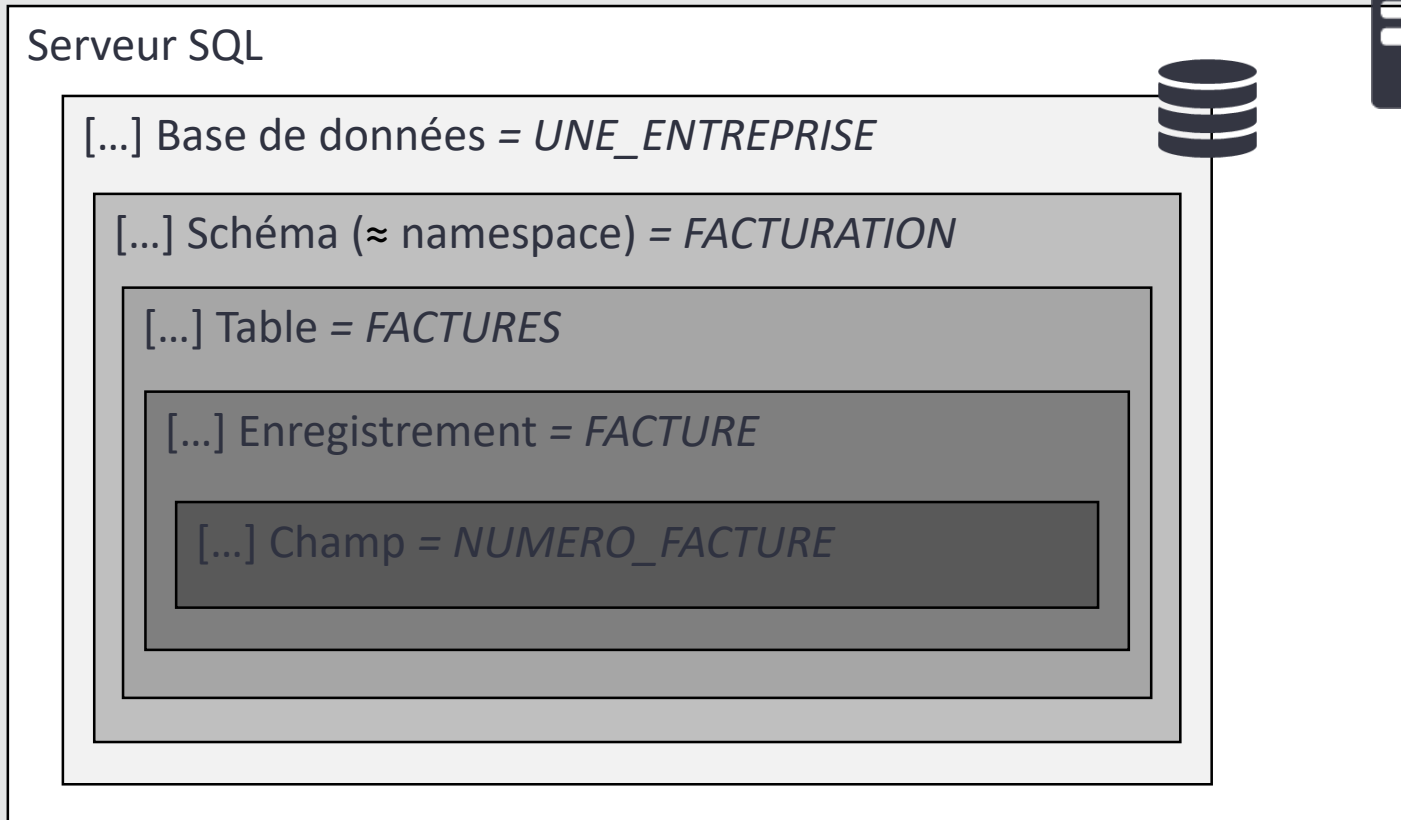
### Exemple :

Un enregistrement d'une FACTURE dans la table FACTURES d'une ENTREPRISE.

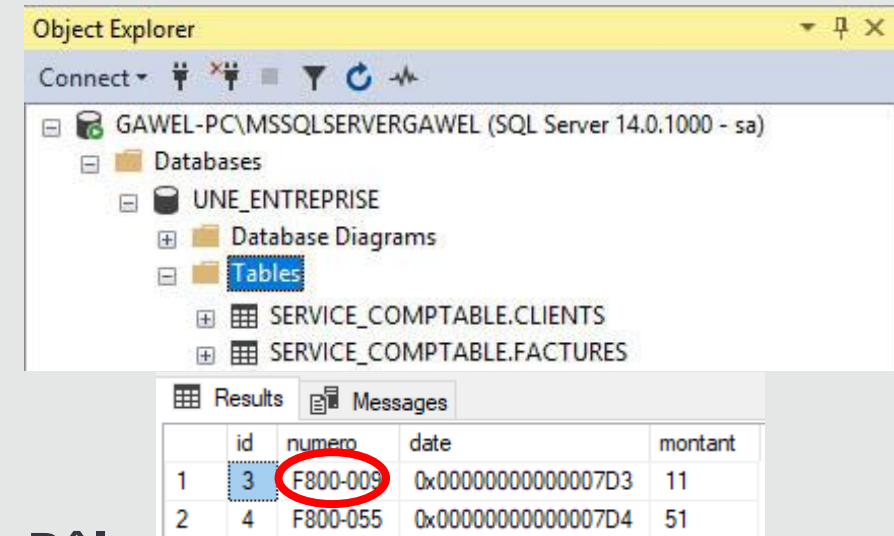
Les bases de données et le langage SQL → 1. Les bases de données

## 1.3 Comprendre les concepts d'une base de données relationnelle

### Structure d'une base de données :



### Correspondance dans SSMS :



### Rôle :

Champ d'un enregistrement.

C'est une donnée brute d'un enregistrement.

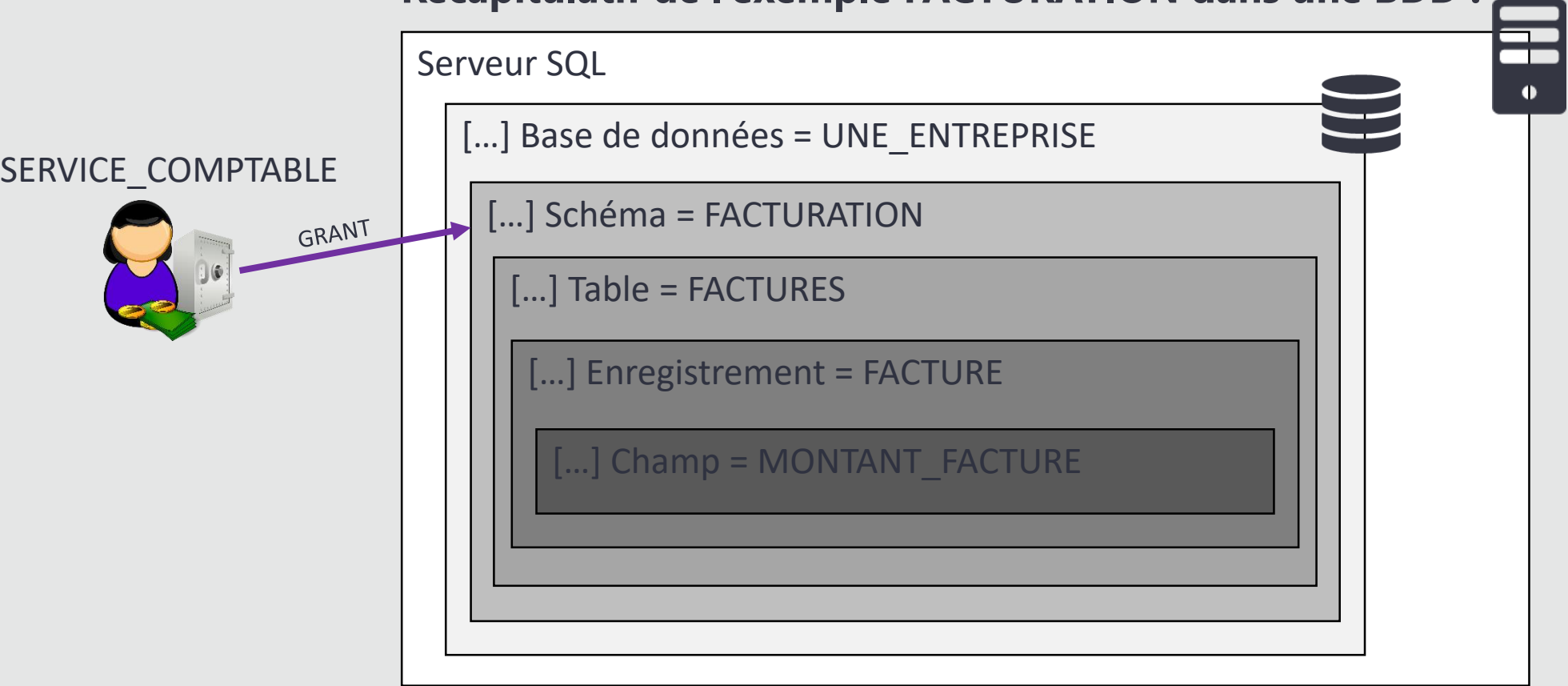
### Exemple :

Le numéro (unique) d'une FACTURE.

# Les bases de données et le langage SQL → 1. Les bases de données

## 1.3 Comprendre les concepts d'une base de données relationnelle

### Récapitulatif de l'exemple FACTURATION dans une BDD :





# Les bases de données et le langage SQL → 1. Les bases de données

## TD : Manipuler une base de données SQL Server

1. Installer et configurer SSMS
- 2. Découverte de l'environnement**
- 3. Créer la base de données UNE\_ENTREPRISE**
- 4. Créer le schéma FACTURATION**
- 5. Créer un utilisateur COMPTABLE1 et attribuer les droits, connexion avec COMPTABLE1**
6. Créer la table FACTURES et ses attributs dans le schéma FACTURATION
7. Réaliser un CRUD sur la table FACTURES
8. Modifier la table FACTURES pour ajouter une clé primaire
9. Créer la table CLIENTS avec des contraintes
10. Créer la relation 1,\* : table FACTURES ajouter une clé étrangère
11. Réaliser un INSERT
12. Créer un graphique UML des deux tables

Les bases de données et le langage SQL → 1. Les bases de données

1.3 Comprendre les concepts d'une base de données relationnelle

Représentation d'une table SQL :

TABLE FACTURES

ATTRIBUT NUMERO\_FACTURE de la table FACTURE

id_facture	numero_facture	date_facture	montant_facture	nb_article	paye	client_id	...
1	F-900-08	2005-01-12 12:36:20.120	120,12€	3	false	55	...
2	Z-500-02	2019-05-05 00:00:00.000	1650,00€	100	true	3	...
3	F-900-09	null	56,31€	1	true	55	...
...	...	...	...			...	...

ENREGISTREMENT FACTURE

CHAMP NUMERO\_FACTURE de l'enregistrement FACTURE N°3

# Les bases de données et le langage SQL → 1. Les bases de données

## 1.3 Comprendre les concepts d'une base de données relationnelle

### Les types de données SQL :

- Les types de données permettent :
  - d'optimiser la place que prennent les données sur le serveur
  - d'optimiser la rapidité des traitements et donc les performances
- Pas de type de données standard
- On trouve en général les types suivants :

Type	Syntaxe SQL
Chaine de caractères	char(n), varchar(n), text, blob
Entier	int
Décimal	decimal(x,y), float
Date	date, dateTime, « timestamp »
Boolean	bit,bool, boolean

*La fonction **getdate()**  
permet de récupérer le  
**dateTime** à l'instant **T***



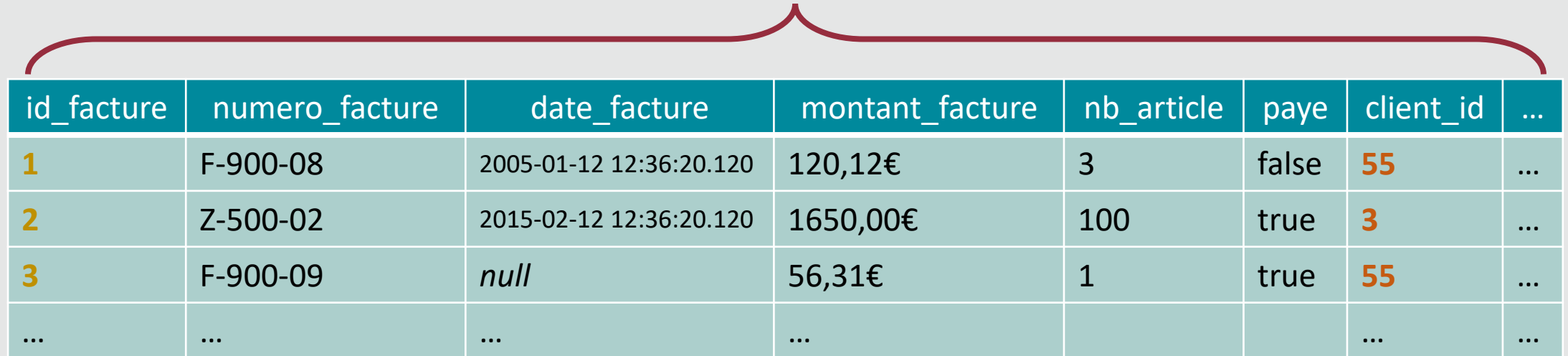
SQL serveur : [documentation sur les types de données](#)

# Les bases de données et le langage SQL → 1. Les bases de données

## 1.3 Comprendre les concepts d'une base de données relationnelle

### Les types de données SQL : Exemple

TABLE FACTURES



id_facture	numero_facture	date_facture	montant_facture	nb_article	paye	client_id	...
1	F-900-08	2005-01-12 12:36:20.120	120,12€	3	false	55	...
2	Z-500-02	2015-02-12 12:36:20.120	1650,00€	100	true	3	...
3	F-900-09	<i>null</i>	56,31€	1	true	55	...
...	...	...	...	...	...	...	...

**VARCHAR** Chaîne caractères

**DATETIME** Date

**DECIMAL** Décimal

**INT** Type entier

**BIT** Boolean

Les bases de données et le langage SQL → 1. Les bases de données

TD : Manipuler une base de données SQL Server

1. Installer et configurer SSMS
2. Découverte de l'environnement
3. Créer la base de données UNE\_ENTREPRISE
4. Créer le schéma FACTURATION
5. Créer un utilisateur COMPTABLE1 et attribuer les droits, connexion avec COMPTABLE1
- 6. Créer la table FACTURES et ses attributs dans le schéma FACTURATION**
- 7. Réaliser un CRUD sur la table FACTURES**
8. Modifier la table FACTURES pour ajouter une clé primaire
9. Créer la table CLIENTS avec des contraintes
10. Créer la relation 1,\* : table FACTURES ajouter une clé étrangère
11. Réaliser un INSERT
12. Créer un graphique UML des deux tables

# Les bases de données et le langage SQL → 1. Les bases de données

## 1.3 Comprendre les concepts d'une base de données relationnelle

### Les contraintes SQL :

- Les contraintes SQL permettant :
  - de consolider les données en apposant des règles (règle d'unicité, valeur non null, etc.)
  - d'optimiser les performances de recherche
  - de créer des relations entre les tables (base de données relationnelle !)

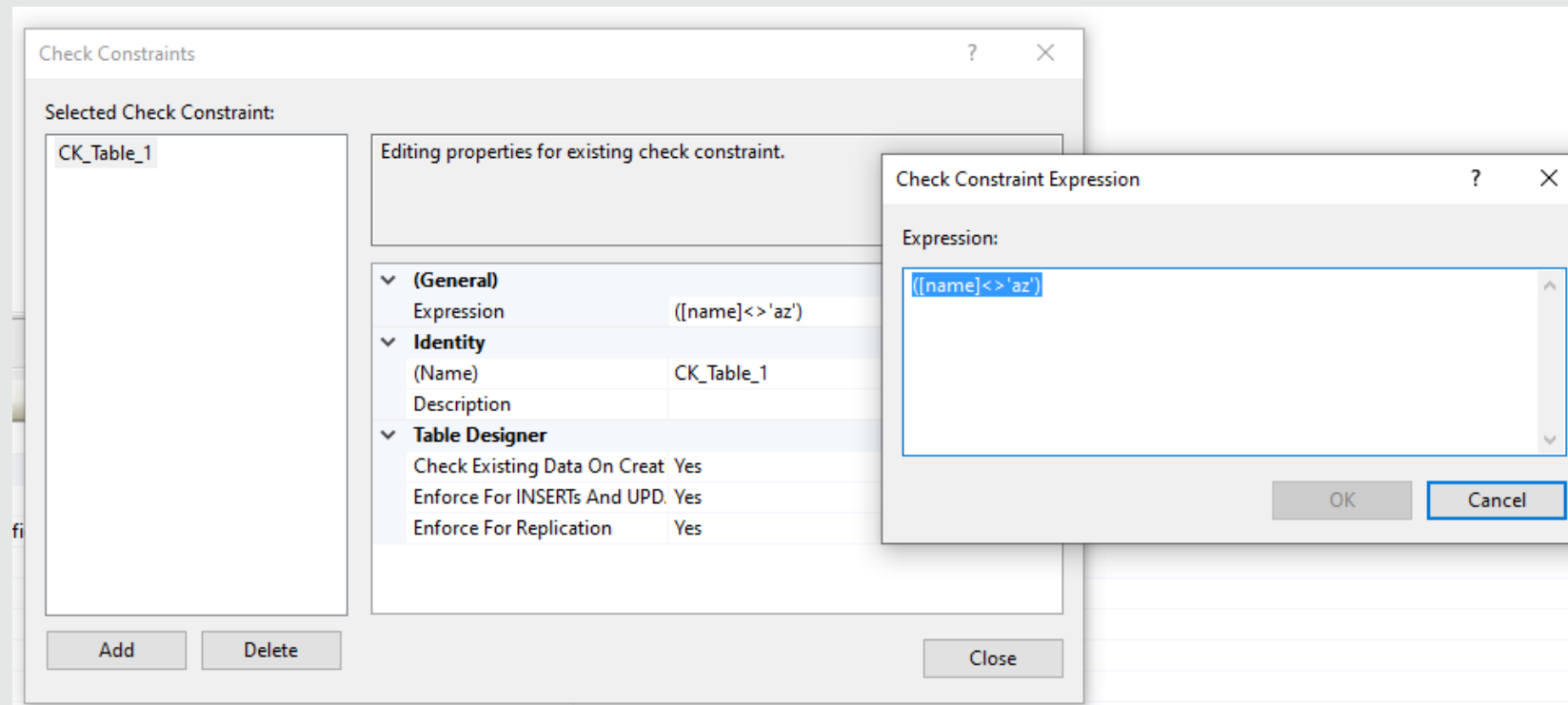
Contrainte	Description
index	Structure facilitant la recherche, le tri et le regroupement d'informations d'une collection
Clé primaire	Champ ou combinaison de champs dont les valeurs sont différentes pour chaque enregistrement de la collection
Clé étrangère:	Champ ou combinaison de champs d'une table A qui sont en relation avec une clé primaire dans une table B
Unique	Créer une contrainte d'unicité sur le champ d'un attribut
NOT NULL	Trefuser les champs = null
INDEX	Indexer l'attribut pour accélérer les requêtes
DEFAULT	Attribuer une valeur par défaut à un champ lors de la création de l'enregistrement
[...]	<i>Et bien d'autres encore..</i>

# Les bases de données et le langage SQL → 1. Les bases de données

## 1.3 Comprendre les concepts d'une base de données relationnelle

### Les contraintes SQL :

- Créer une contrainte personnalisée : (presque) tout est possible!

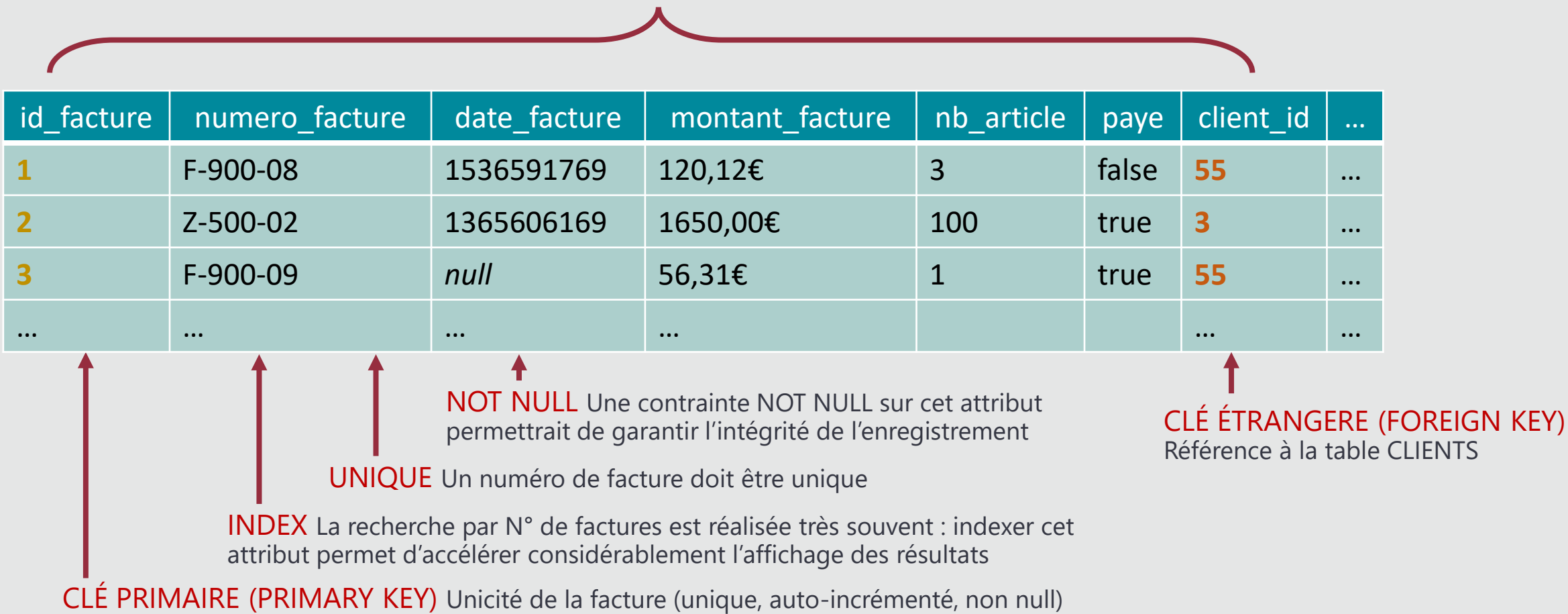


# Les bases de données et le langage SQL → 1. Les bases de données

## 1.3 Comprendre les concepts d'une base de données relationnelle

### Les contraintes SQL : Exemple

### TABLE FACTURES



id_facture	numero_facture	date_facture	montant_facture	nb_article	paye	client_id	...
1	F-900-08	1536591769	120,12€	3	false	55	...
2	Z-500-02	1365606169	1650,00€	100	true	3	...
3	F-900-09	null	56,31€	1	true	55	...
...	...	...	...			...	...

**CLÉ PRIMAIRE (PRIMARY KEY)** Unicité de la facture (unique, auto-incrémenté, non null)

**INDEX** La recherche par N° de factures est réalisée très souvent : indexer cet attribut permet d'accélérer considérablement l'affichage des résultats

**UNIQUE** Un numéro de facture doit être unique

**NOT NULL** Une contrainte NOT NULL sur cet attribut permettrait de garantir l'intégrité de l'enregistrement

**CLÉ ÉTRANGÈRE (FOREIGN KEY)**  
Référence à la table CLIENTS



# Les bases de données et le langage SQL → 1. Les bases de données

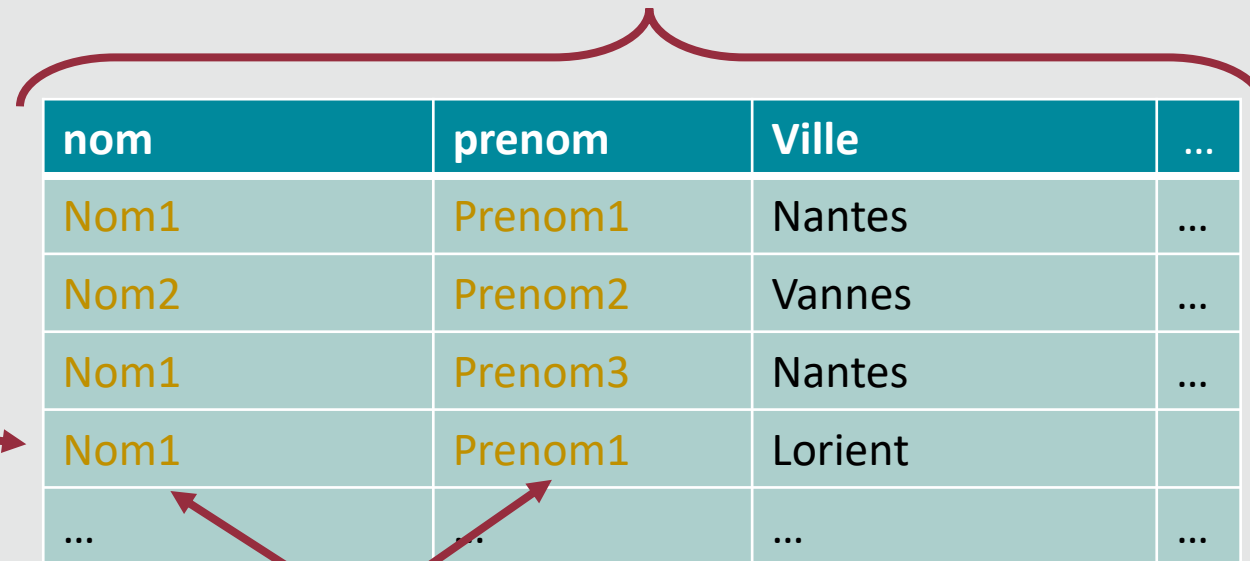
## 1.3 Comprendre les concepts d'une base de données relationnelle

### Les clés primaires composées

Dans certain cas il est souhaitable de disposer d'une clé primaire non pas

Sur un unique ID mais sur plusieurs ID. Nous parlons ici d'une clé primaire composée.

**TABLE PERSONNE**



nom	prenom	Ville	...
Nom1	Prenom1	Nantes	...
Nom2	Prenom2	Vannes	...
Nom1	Prenom3	Nantes	...
Nom1	Prenom1	Lorient	
...	...	...	...

Enregistrement impossible →

CLÉ PRIMAIRE COMPOSEE (PRIMARY KEY)

# Les bases de données et le langage SQL → 1. Les bases de données

## TD : Manipuler une base de données SQL Server

1. Installer et configurer SSMS
2. Découverte de l'environnement
3. Créer la base de données UNE\_ENTREPRISE
4. Créer le schéma FACTURATION
5. Créer un utilisateur COMPTABLE1 et attribuer les droits, connexion avec COMPTABLE1
6. Créer la table FACTURES et ses attributs dans le schéma FACTURATION
7. Réaliser un CRUD sur la table FACTURES
- 8. Modifier la table FACTURES pour ajouter une clé primaire**
- 9. Créer la table CLIENTS avec des contraintes**
10. Créer la relation 1,\* : table FACTURES ajouter une clé étrangère
11. Réaliser un INSERT
12. Créer un graphique UML des deux tables

Les bases de données et le langage SQL → 1. Les bases de données

## 1.3 Comprendre les concepts d'une base de données relationnelle

### Les relations SQL :

- Permet de faire référence à une donnée déjà existante dans la base de données (relations entre les données)
- Permet d'éviter la redondance de données
- Permet d'organiser les différents types de données (par exemple : les factures, les clients, etc.)

### Exemple (UML) :



L'**UML** ou encore le **MERISE**  
sont des représentations  
graphiques normées destinées  
à représenter (entre autre) les  
bases de données. Très utiles,  
elles simplifient grandement la  
compréhension des systèmes  
complexes !

# Les bases de données et le langage SQL → 1. Les bases de données

## 1.3 Comprendre les concepts d'une base de données relationnelle

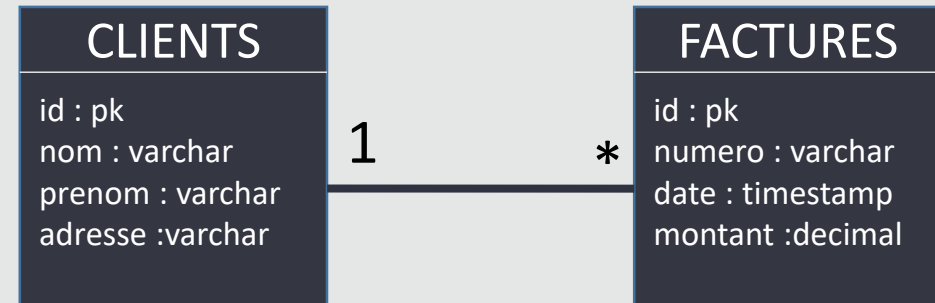
### Les relations SQL :

Le type de la relation entre 2 tables s'appelle une **cardinalité**.

Il existe 4 types de cardinalités :

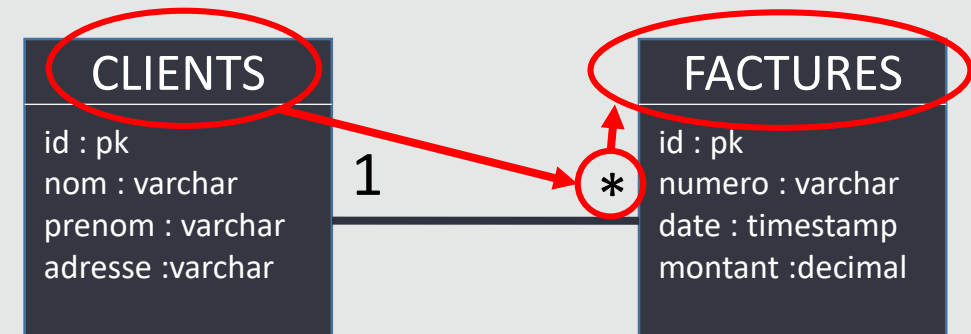
Cardinalité	Description
1	Un seul
*	Plusieurs
1..*	Au moins un
0..*	Un nombre indéterminé

### Exemple (UML) :



**1 client** possède **plusieurs factures**.  
**1 facture** ne peut avoir que **1 client**.

### Sens de lecture :



# Les bases de données et le langage SQL → 1. Les bases de données

## 1.3 Comprendre les concepts d'une base de données relationnelle

### Comment traduire une relation \*.\* (ManyToMany)

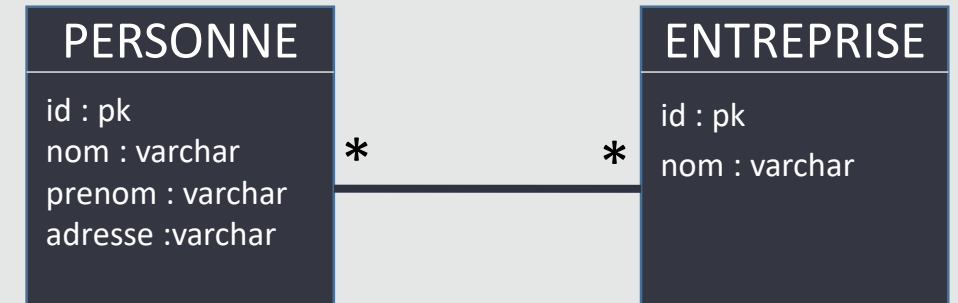
Les relations n,n ne sont pas faisables en base de données relationnelle!

Ainsi nous utilisons des tables intermédiaires : des tables « relationnelles ».

A noter que les ternaires sont à éviter autant que possible !

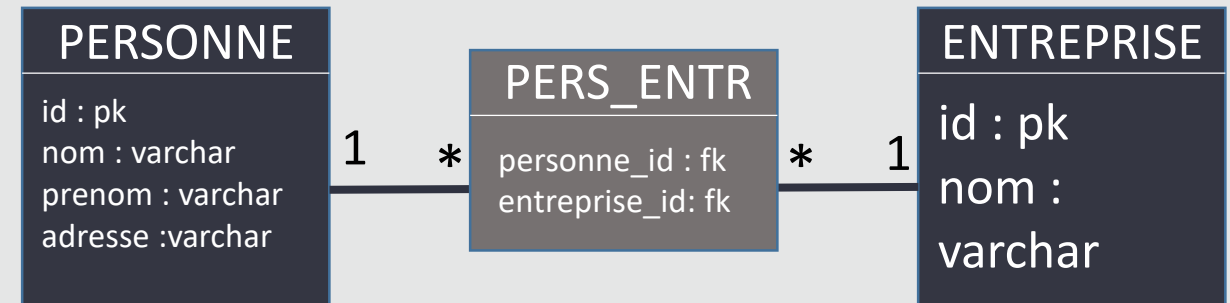
Cardinalité	Description
*.*	Plusieurs à Plusieurs

### Exemple (POO) :



**plusieurs personnes** possède **plusieurs entreprises**.  
**plusieurs entreprises** possède **plusieurs personnes**.

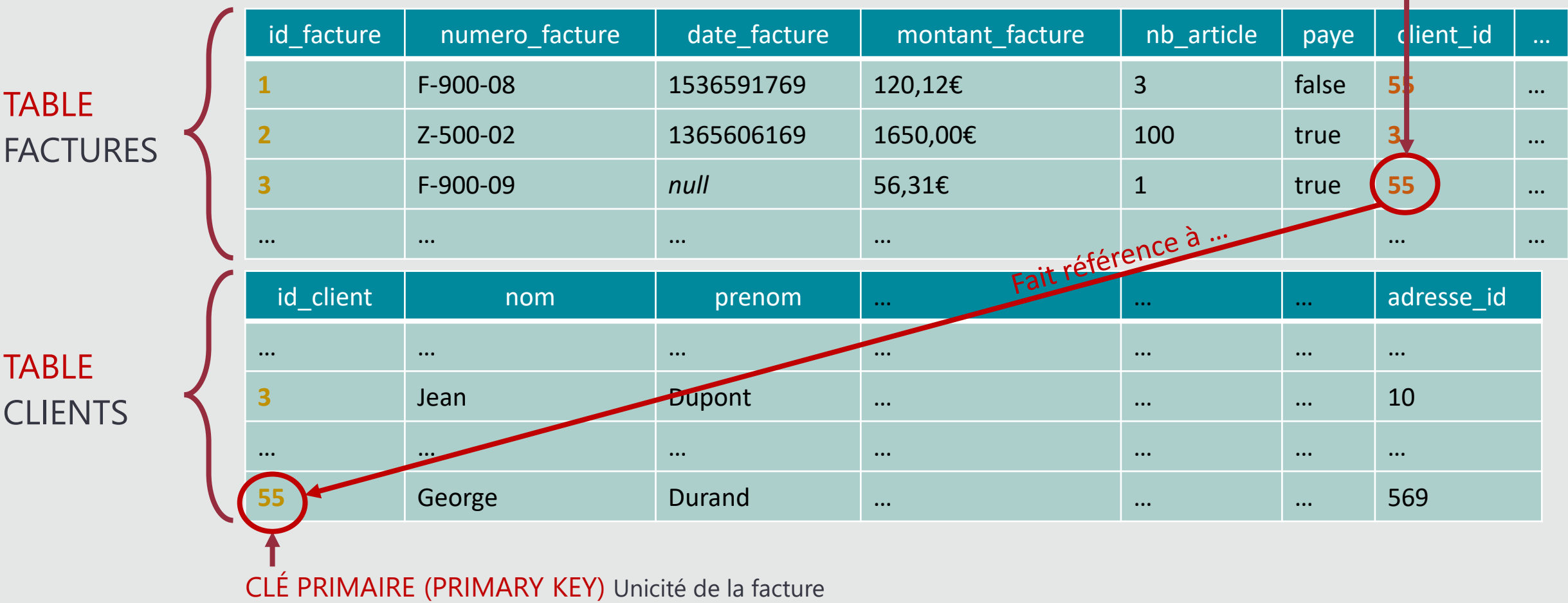
### Exemple (SQL) :



Les bases de données et le langage SQL → 1. Les bases de données

1.3 Comprendre les concepts d'une base de données relationnelle

Les relations SQL :



# Les bases de données et le langage SQL → 1. Les bases de données

## TD : Manipuler une base de données SQL Server

1. Installer et configurer SSMS
2. Découverte de l'environnement
3. Créer la base de données UNE\_ENTREPRISE
4. Créer le schéma FACTURATION
5. Créer un utilisateur COMPTABLE1 et attribuer les droits, connexion avec COMPTABLE1
6. Créer la table FACTURES et ses attributs dans le schéma FACTURATION
7. Réaliser un CRUD sur la table FACTURES
8. Modifier la table FACTURES pour ajouter une clé primaire
9. Créer la table CLIENTS avec des contraintes
- 10. Créer la relation 1,\* : table FACTURES ajouter une clé étrangère**
- 11. Réaliser un INSERT**
- 12. Créer un graphique UML des deux tables**
- 13. Sensibilisation à l'utilisation des commandes inconnues...**

# Les bases de données et le langage SQL → 1. Les bases de données

## 1.5 Travaux pratiques

1. Prérequis : être connecté en « Super Administrateur » sur SQL Server
2. Créer l'utilisateur « gaston\_lagaffe » (dans le moteur de base de données) :
3. Créer la base de données « editions\_dupuis » :
  1. Encodage : « french\_ci\_as »
  2. Propriétaires : « Super Administrateur », et « gaston\_lagaffe »
4. Se connecter en tant que « gaston\_lagaffe » avec SSMS
5. Créer un schéma « agences »





# Les bases de données et le langage SQL → 1. Les bases de données

## 1.4 Travaux pratiques

5. Créer la table « bureaux » dans le schéma « agences ». Attributs de la table :

Name	Type	Contrainte
Id	Int	PRIMARY KEY, NOT NULL, AUTO-INCREMENT
nom	varchar(50)	NOT NULL, INDEX, UNIQUE
Ville	varchar(50)	INDEX
codepostal	varchar(5)	DEFAULT = 44000

6. (CREATE) Peupler la table avec les enregistrements suivantes :

	Bureaux							
id	null	null	null	1	null	null	null	
Nom	Bureau 1	Bureau 2	Bureau 3	Bureau 4	null	Bureau 4	Bureau 6	
Ville	Paris	Vannes	Nantes	Niort	Rennes	Brest	Paris	
Code postal	75000	56130	null	null	35000	29200	750707	

Et corriger les éventuelles problèmes...

7. (Read) Vérifier l’insertion

Les bases de données et le langage SQL → 1. Les bases de données

1.4 Travaux pratiques

8. Créer la table « gerants » dans le schéma « agences ». Attributs de la table :

gerants			
	Column Name	Data Type	Allow Nulls
🔑	id	int	<input type="checkbox"/>
	nom	varchar(50) - UNIQUE	<input type="checkbox"/>
	prenom	varchar(60)	<input type="checkbox"/>
	age	int	<input checked="" type="checkbox"/>
	taille	decimal(18, 2)	<input checked="" type="checkbox"/>
			<input type="checkbox"/>

9. (CREATE) Peupler la table avec les enregistrements suivantes :

	Gérants							
Nom	Lagaffe	Dupuis	Fantasio	Lebrac	null	Spirou	Prunelle	Prunelle
Prénom	Gaston	Anne	Doisy	George	null	Roger	Amélie	Sophie
Âge	32	null	35	23	32	45,5	32	31
Taille	null	null	1,80	1,32	1,65	1,72	1,78	1

Et corriger les éventuelles problèmes...

10. (Read) Vérifier l’insertion

# Les bases de données et le langage SQL → 1. Les bases de données

## 1.4 Travaux pratiques

11. Créer et observer le graphique UML des 2 tables

12. Créer la relation entre les bureaux et leurs gérants

Relation : 1 bureau ne peut avoir que 1 gérant. 1 gérant possède plusieurs bureaux.

13. (Update) Peupler la table « bureaux » comme suite :

	Bureaux							
id	1	2	3	4	5	6	7	
Nom	Bureau 1	Bureau 2	Bureau 3	Bureau 4	<i>null</i>	Bureau 4	Bureau 6	
Ville	Paris	Vannes	Nantes	Niort	Rennes	Brest	Paris	
Code postal	75000	56130	<i>null</i>	<i>null</i>	35000	29200	750707	
Gérant	1	1	<i>null</i>	<i>null</i>	<i>null</i>	George	4	

*Et corriger les éventuelles problèmes...*

14. (Read) Vérifier les insertions réalisées

12. Créer et observer le graphique UML des 2 tables

# 1. LES BASES DE DONNEES ETUDE DE CAS



# Les bases de données et le langage SQL → 1. Les bases de données

## TD : Etude de cas

1. Sur papier, créer le diagramme UML complet de notre exemple FACTURE.
2. Le client est sympa et nous donne le template de la facture à développée :

Facture FR-001

Joanna Binet

48 Coubertin  
31400 Paris

LOGO

Facturé à

Envoyé à

Date

29/01/2019

Cendrillon Ayot  
69 rue Nations  
22000 Paris

Cendrillon Ayot  
46 Rue St Ferréol  
92360 Île-de-France

Commande n°

1630/2019

Échéance

24/05/2019

Qté	Désignation	Prix Unit. HT	Montant HT
1	Grand brun escargot pour manger	100.00	100.00
2	Petit marinière uniforme en bleu	15.00	30.00
3	Facile à jouer accordéon	5.00	15.00

Total HT

145.00

TVA 20.0%

29.00

Total

174.00 €

## 2. LE LANGUAGE SQL



# Les bases de données et le langage SQL → 2. Le langage SQL

## 2.1 Comprendre l'intérêt et le rôle du langage SQL ?

### Qu'est ce que le SQL (Structured Query Language) ?

- C'est un langage normalisé pour les bases de données relationnelles
- Il permet d'interagir avec la structure de la base de données
- Il permet d'interagir avec les données de la base de données
- Il est indépendant du moteur SQL

### Qui l'utilise ?



# Les bases de données et le langage SQL → 2. Le langage SQL

## 2.1 Comprendre l'intérêt et le rôle du langage SQL ?

### Qu'est ce que le SQL (Structured Query Language) ?

- C'est un ensemble d'instructions et de fonctions pour la gestion des données
  - Il fait
    - L'administration de la base (les utilisateurs, les droits d'accès...)
    - La gestion des structures (les objets)
    - La gestion du contenu (les données)
  - Il ne fait pas
    - La présentation des résultats
    - La gestion des erreurs



# Les bases de données et le langage SQL → 2. Le langage SQL

## 2.1 Comprendre l'intérêt et le rôle du langage SQL ?

### Qui l'utilise?

- Le SQL est utilisé par les différents utilisateurs:
  - L'utilisateur final
    - Gestion des lignes (le contenu)
  - Le développeur
    - Gestion des requêtes complexes
    - Gestion de l'intégrité des données
    - Gestion de la structure (le schéma des tables)
  - L'administrateur
    - Création de la base
    - Démarrage et arrêt de la base
    - Sécurité des accès
    - Optimisation et surveillance
    - Maintenance (les fichiers physiques, la mémoire)

# Les bases de données et le langage SQL → 2. Le langage SQL

## 2.2 Apprendre la syntaxe du langage SQL

**Le SQL est structuré en 3 parties :**

Acronyme	Signification	Opérations
<b>DDL</b>	Data Definition Language	<ul style="list-style-type: none"><li>• <b>CREATE</b> : Création de tables, index, schéma, etc...</li><li>• <b>ALTER</b> : Modification de tables, index, etc...</li><li>• <b>DROP</b> : Suppression de tables, index, etc...</li></ul>
<b>DQL</b>	Data Query Language	<ul style="list-style-type: none"><li>• <b>SELECT</b> : Extraction des données</li></ul>
<b>DML</b>	Data Manipulation Language	<ul style="list-style-type: none"><li>• <b>INSERT</b> : Insertion d'un enregistrement</li><li>• <b>UPDATE</b> : Modification d'enregistrements</li><li>• <b>DELETE</b> : Suppression d'enregistrements</li></ul>

Remember ! {  
C<sub>reate</sub> / Créer  
R<sub>ead</sub> / Lire  
U<sub>pdate</sub> / Mettre à jour  
D<sub>ele</sub>te / Supprimer

## 2. LE LANGUAGE SQL : DLL



# Les bases de données et le langage SQL → 2. Le langage SQL

## 2.2 Apprendre la syntaxe du langage SQL

### DDL → Manipulation d'une base de données

Les DDL sont sensiblement différents selon le moteur SQL !

CRUD	Déclaration	Syntaxe
READ	SELECT	<code>SELECT * FROM sys.databases;</code>
	Exemple : X	
CREATE	CREATE	<code>CREATE DATABASE nom_de_la_bdd COLLATE &lt;encoding_name&gt;;</code>
	Exemple :	<code>CREATE DATABASE une_entreprise COLLATE French_CI_AS;;</code>
DELETE	DROP	<code>DROP DATABASE nom_de_la_bdd;</code>
	Exemple :	<code>DROP DATABASE une_entreprise;</code>
UPDATE	ALTER	<code>ALTER DATABASE nom_de_la_bdd SET NOM_CONSTANTE = 'valeur';</code>
	Exemple :	<code>-- Version SQL Server ALTER DATABASE une_entreprise SET COMPATIBILITY_LEVEL = 140;</code>

# Les bases de données et le langage SQL → 2. Le langage SQL

## 2.2 Apprendre la syntaxe du langage SQL

### DDL → Manipulation d'une base de données

CRUD	Déclaration	Syntaxe	Action
X	USE	<code>USE nom_de_la_bdd;</code>	<code>USE</code> master; Utiliser une base de données définie
X	GO	<code>GO</code>	« Commiter » les modifications

Les bases de données et le langage SQL → 2. Le langage SQL

## 2.2 Apprendre la syntaxe du langage SQL

### DDL → Manipulation les schémas

CRUD	Déclaration	Syntaxe
CREATE	CREATE	<code>CREATE SCHEMA schema_name_clause;</code>
DELETE	DROP	<code>DROP SCHEMA schema_name_clause;</code> (Attention un schéma utilisé par une table ne peut pas être supprimé)

# Les bases de données et le langage SQL → 2. Le langage SQL

## 2.2 Apprendre la syntaxe du langage SQL

### DDL → Manipulation des tables

CRUD	Déclaration	Syntaxe
READ	SELECT	<code>SELECT * FROM nom_bdd.INFORMATION_SCHEMA.TABLES WHERE TABLE_TYPE='BASE TABLE'</code>
	Exemple :	<code>SELECT * FROM une_entreprise.INFORMATION_SCHEMA.TABLES WHERE TABLE_TYPE='BASE TABLE'</code>
CREATE	CREATE	<code>CREATE TABLE nom_bdd.nom_schema.nom_table(nom_attribut type_attribut contraintes, [...]);</code>
	Exemple :	<code>CREATE TABLE UNE_ENTREPRISE.FACTURATION.facture(id int NOT NULL, nom varchar(50));</code>
DELETE	DROP	<code>DROP TABLE nom_bdd.nom_schema.nom_table;</code>
	Exemple :	<code>DROP TABLE UNE_ENTREPRISE.FACTURATION.facture;</code>
UPDATE	ALTER	<code>ALTER TABLE nom_bdd.nom_schema.nom_table ACTION nom_attribut type contraintes, [...];</code>
	Exemple :	<code>ALTER TABLE UNE_ENTREPRISE.FACTURATION.facture ADD nom_client VARCHAR(20), nb_article INT NOT NULL ;</code> <code>ALTER TABLE UNE_ENTREPRISE.FACTURATION.facture ALTER COLUMN nom_client VARCHAR(50);</code>

# Les bases de données et le langage SQL → 2. Le langage SQL

## 2.2 Apprendre la syntaxe du langage SQL

### DDL → Manipulation des tables → Créer des contraintes et relations

Description	Syntaxe
Créer une table avec des contraintes	<pre>CREATE TABLE ma_table(   -- Créer une clé primaire auto-incrémentée   col1      INT IDENTITY(1,1) CONSTRAINT pk_nom PRIMARY KEY,   -- Interdir les valeurs null   col2      VARCHAR(30) NOT NULL,   -- Vérifier l'intervall d'une valeur   col5      CHAR(5) NOT NULL CONSTRAINT ck_matable_col5 CHECK(col5 BETWEEN 1000 AND 95999),   -- Valeur par défaut   col6      VARCHAR(80) DEFAULT 'Nantes' NOT NULL,   -- Vérifier une valeur selon une liste de critères   col7      CHAR(2) DEFAULT 'VALEUR1' NOT NULL ck_matable_col7 col7 IN('VALEUR1', 'VALEUR2')),   -- Valeur unique dans la table   col8      VARCHAR(80) NOT NULL CONSTRAINT ck_matable_col8 UNIQUE   -- Créer une clé secondaire   fk_nom    INT CONSTRAINT fk_mon_fk REFERENCES ma_table2(mn_id)),   -- Clé primaire composée (impossible ici car une clé existe déjà (col1) donc en commentaire   -- CONSTRAINT pk_col_fk_compo PRIMARY KEY(col1, col2), );</pre>



Les bases de données et le langage SQL → 2. Le langage SQL

## 2.2 Apprendre la syntaxe du langage SQL

### DDL → Manipulation les index

Description	Syntaxe
Créer une table avec des contraintes	<pre>-- Créer un index CREATE INDEX my_index_name ON my_table (col1);  -- Supprimer un index DROP INDEX index_name ON my_table;</pre>

/!\ Des INDEX sont automatiquement créés pour les cas suivant :

- CLE PRIMAIRE
- CLE SECONDAIRE
- Contrainte d'UNICITE

# Les bases de données et le langage SQL → 2. Le langage SQL

## 2.2 Apprendre la syntaxe du langage SQL

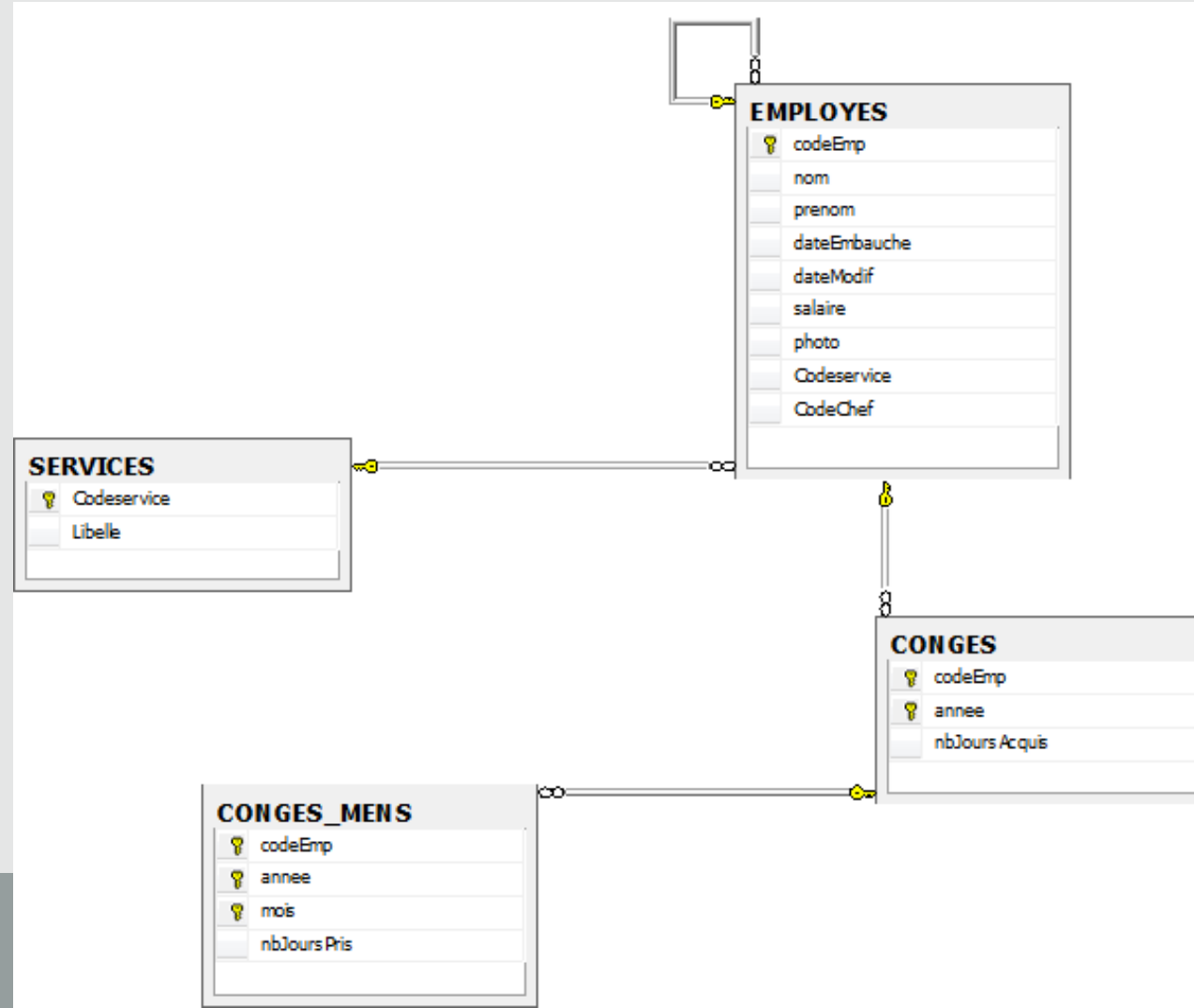
### DDL → Manipulation des utilisateurs

CRUD	Déclaration	Syntaxe
READ	SELECT	<code>SELECT * FROM master.sys.server_principals;</code>
	Exemple :	X
CREATE	CREATE	<code>CREATE LOGIN nom_user WITH PASSWORD='passwd', DEFAULT_DATABASE=nom_bdd [...];</code>
	Exemple :	<code>CREATE LOGIN toto WITH PASSWORD='mon!pass', DEFAULT_DATABASE=une_entreprise;</code>
DELETE	DROP	<code>DROP LOGIN nom_user;</code>
	Exemple :	<code>DROP LOGIN toto;</code>
UPDATE	ALTER	<code>ALTER LOGIN nom_user ACTION [...];</code>
	Exemple :	<code>ALTER LOGIN toto WITH PASSWORD='new!passwd';</code>

# Les bases de données et le langage SQL → 1. Les bases de données

## 2.2 Travaux pratiques

- 1, Créer la base de données « GESTIONEMPLOYES » selon le schéma suivant (attention aux types)
- 2, Vérifier l'intégrité de la base de données avec le jeu de données



## 2. LE LANGUAGE SQL : DML



Les bases de données et le langage SQL → 2. Le langage SQL

2.2 Apprendre la syntaxe du langage SQL

DML → INSERT : Insertion d'un enregistrement

CRUD	Déclaration	Syntaxe	Exemple
CREATE	INSERT INTO	INSERT INTO table_name(col1, ...) VALUES ('data', ...);	INSERT INTO factures(numero_facture) VALUES ('F800-01');

Syntaxe : INSÉRER DANS la table(colonnes à insérer) LE/LES VALEURS (valeurs à insérer)

Exemple : INSERER DANS LES factures : UNE facture AVEC le numéro de facture 'F800-01'

Les bases de données et le langage SQL → 2. Le langage SQL

## 2.2 Apprendre la syntaxe du langage SQL

### **DML → INSERT : Insertion d'un enregistrement**

#### **Un INSERT doit respecter :**

- Les contraintes (non null, clé primaire et secondaire, etc.)
- Le type de données des champs (varchar, decimal, boolean, etc.)
- Dans le cas d'une clé primaire auto-incrémenter : la clé ne doit pas être stipulée

#### **Un INSERT peut aussi s'écrire :**

```
INSERT INTO table_name VALUES ('data', ...);
```

#### **Dans ce cas l'INSERT doit respecter :**

- L'ordre des colonnes

Les bases de données et le langage SQL → 2. Le langage SQL

2.2 Apprendre la syntaxe du langage SQL

DML → INSERT : Insertion d'un enregistrement

Exemple :

1) Table de données brutes :

id	numero_facture	date_facture	montant_facture	nb_article	paye	client_id
1	F-900-08	1536591769	120,12€	3	false	55

2) `INSERT INTO factures(numero_facture, nb_article, paye, client_id) VALUES ('F500-05', 5, 0, 22);`

id	numero_facture	date_facture	montant_facture	nb_article	paye	client_id
1	F-900-08	1536591769	120,12€	3	false	55
2	F500-05	null	null	5	false	22

Les bases de données et le langage SQL → 2. Le langage SQL

2.2 Apprendre la syntaxe du langage SQL

DML → UPDATE : Modification d'enregistrements

CRUD	Déclaration	Syntaxe	Exemple
UPDATE	UPDATE	UPDATE table_name SET col1='data', ... WHERE condition;	UPDATE factures SET numero_facture='F999' WHERE numero_facture = 'F800-05';

Syntaxe : MODIFIER LA table : LES COLONNES col1 SERONT EGALS A 'data' QUAND l'on respect cette condition

Exemple : MODIFIER LES factures : LE numéro de facture SERA EGAL A 'F999' QUAND le numéro de facture EST ÉGALE A 'F800-05'



Les bases de données et le langage SQL → 2. Le langage SQL

## 2.2 Apprendre la syntaxe du langage SQL

### **DML → UPDATE : Modification d'enregistrements**

#### **Un UPDATE est capable :**

- De modifier un ou plusieurs champs en même temps
- De modifier un ou plusieurs enregistrements en même temps /!\

#### **Un UPDATE doit respecter :**

- Les contraintes (non null, clé primaire et secondaire, etc.)
- Le type de données des champs (varchar, decimal, boolean, etc.)

#### **Attention :**

- Ne JAMAIS modifier une clé primaire ! (possible sur certaine configuration de BDD) /!\

# Les bases de données et le langage SQL → 2. Le langage SQL

## 2.2 Apprendre la syntaxe du langage SQL

### DML → UPDATE : Modification d'enregistrements

#### Exemple :

1) Table de données brutes :

id	numero_facture	date_facture	montant_facture	nb_article	paye	client_id
1	F-900-08	1536591769	120,12€	3	false	55
2	F400-15	1536591769	1200,12€	5	false	10
3	F500-05	null	null	5	false	22

2) `UPDATE factures SET paye=1 WHERE montant_facture <> null;`

id	numero_facture	date_facture	montant_facture	nb_article	paye	client_id
1	F-900-08	1536591769	120,12€	3	true	55
2	F400-15	1536591769	1200,12€	5	true	10
3	F500-05	null	null	5	false	22

Les bases de données et le langage SQL → 2. Le langage SQL

2.2 Apprendre la syntaxe du langage SQL

DML → DELETE : Suppression d'enregistrements

CRUD	Déclaration	Syntaxe	Exemple
DELETE	DELETE	DELETE FROM table_name WHERE condition;	DELETE FROM factures WHERE client_id=55;

Syntaxe : SUPPRIMER DANS LA table LES ENREGISTREMENTS QUI RESPECTE cette condition

Exemple : SUPPRIMER LES factures QUAND l'identifiant du client EST ÉGALE A 55

Les bases de données et le langage SQL → 2. Le langage SQL

## 2.2 Apprendre la syntaxe du langage SQL

### **DML → DELETE : Suppression d'enregistrements**

#### **Un DELETE est capable :**

- De supprimer un ou plusieurs enregistrements en même temps /\

#### **Attention :**

- Le DELETE peut être traître! /\

Bien vérifier la requête avant de l'exécuter ou mieux : utilisez les transactions !

# Les bases de données et le langage SQL → 2. Le langage SQL

## 2.2 Apprendre la syntaxe du langage SQL

### DML → DELETE : Suppression d'enregistrements

#### Exemple :

1) Table de données brutes :

id	numero_facture	date_facture	montant_facture	nb_article	paye	client_id
1	F-900-08	1536591769	120,12€	3	false	55
2	F400-15	1536591769	1200,12€	5	false	10
3	F500-05	null	null	5	false	22

2) `DELETE FROM factures WHERE date_facture >= 1536591769;`

id	numero_facture	date_facture	montant_facture	nb_article	paye	client_id
3	F500-05	null	null	5	false	22

# Les bases de données et le langage SQL → 2. Le langage SQL

## 2.2 Apprendre la syntaxe du langage SQL

### Récapitulatif des requêtes CRUD (DML) :

CRUD	Syntaxe	Exemple
CREATE	<code>INSERT INTO table_name(col1, ...) VALUES ('data', ...);</code>	<code>INSERT INTO factures(numero_facture) VALUES ('F800-01');</code>
UPDATE	<code>UPDATE table_name SET col1='data', ... WHERE condition;</code>	<code>UPDATE factures SET numero_fact='F999' WHERE numero_facture='F800-05';</code>
DELETE	<code>DELETE FROM table_name WHERE condition;</code>	<code>DELETE FROM factures WHERE client_id=55;</code>

Les bases de données et le langage SQL → 2. Le langage SQL  
2.3 Travaux pratiques



**3 - TP - DLL+INSERT - Location – enonce.pdf**

# Les bases de données et le langage SQL → 2. Le langage SQL

## 2.3 Travaux pratiques

### TP Location

#### Questions :

- 1) Modifier la désignation du premier article en « Ecran Samsung »
- 2) Les articles avec le codeGam EG doivent être au format « EG -*désignation* »
- 3) Modifier le codeGam EG en ER d'une gamme





# Les bases de données et le langage SQL → 2. Le langage SQL

## 2.3 Travaux pratiques

### Questions :

- 1) Supprimer le client « boutaud sabine»
- 2) Supprimer toutes les fiches avec une date de payée non renseignée
- 3) Supprimer les articles avec le codeGam EG
- 4) Vite le fisc. arrive! Supprimer la totalité des clients!
- 5) Observer la table FICHE : pourquoi?



## 2. L'ALGÈBRE RELATIONNELLE : DQL



Les bases de données et le langage SQL → 2. Le langage SQL

2.2 L'algèbre relationnelle → Les opérateurs

*L'algèbre relationnelle est la logique d'extraction des données.*

# Les bases de données et le langage SQL → 2. Le langage SQL

## 2.2 L'algèbre relationnelle → Les opérateurs → La théorie

OP	OPERATEUR	DESCRIPTION
U	UNION	Ensemble des éléments DISTINCTS des deux relations initiales
$\cap$	INTERSECTION	Ensemble des éléments COMMUNS aux deux relations initiales
-	DIFFERENCE	Ensemble des éléments de la première relation QUI NE SONT PAS dans la deuxième
$\div$	DIVISION	Ensemble des éléments PRÉSENT dans le dividende MAIS PAS le diviseur
$\Sigma$ (sigma )	SELECTION/RESTRICTION	Éléments de la relation initiale qui répondent à une <b>CONDITION</b>
$\Pi$ (Pi)	PROJECTION	<b>Schéma D'ATTRIBUTS et de VALEURS</b>
X	PRODUIT CARTESIEN	Association de CHAQUE LIGNE de la première table avec chaque ligne de la seconde
JOIN	JOINTURE	<b>RESTRICTION sur le produit catésien</b>
	CALCUL ELEMENTAIRE	Calcul portant sur CHAQUE LIGNE
	CALCUL AGREGAT	Calcul portant sur un REGROUPEMENT de lignes

Les bases de données et le langage SQL → 2. Le langage SQL

2.2 L'algèbre relationnelle → Les opérateurs → La théorie

**Exemple d'utilisation :**

```
// Filtrer les CLIENTS par code postal = 44
```

```
R1 =  $\sigma$ (code_postal commence par '44') Clients
```

```
// Jointure de la données CLIENT avec la données FICHE
```

```
R2 = R1 JOIN (R1.client_id=Fiches.client_id) Fiche
```

```
// Préserver la donnée : id de la fiche, nom du client, prénom du client
```

```
ResC =  $\pi$ R2(fiche_id, etat, nom, prenom)
```

Les bases de données et le langage SQL → 1. Les bases de données  
2.2 Travaux pratiques

## 5 - TP - ALGEBRE - LOCATION

## 2. LE LANGUAGE SQL : DQL



Les bases de données et le langage SQL → 2. Le langage SQL

2.2 Apprendre la syntaxe du langage SQL

DQL → SELECT : Extraction des données

CRUD	Déclaration	Syntaxe	Exemple
READ	SELECT	SELECT col1, col2, ... FROM table_name;	SELECT numero_facture FROM factures;

Syntaxe : SÉLECTIONNER les colonnes par leurs noms DANS la table (nom de la table)

Exemple : SÉLECTIONNER le numéro des factures DES factures



Les bases de données et le langage SQL → 2. Le langage SQL

2.2 Apprendre la syntaxe du langage SQL

DQL → SELECT : Extraction des données

Exemple :

1) Table de données brutes :

id	numero_facture	date_facture	montant_facture	nb_article	paye	client_id	...
1	F-900-08	1536591769	120,12€	3	false	55	...
2	Z-500-02	1365606169	1650,00€	100	true	3	...
3	F-900-09	null	56,31€	1	true	55	...
...	...	...	...			...	...

2) `SELECT numero_facture, paye FROM factures;`

numero_facture	paye
F-900-08	false
Z-500-02	true
F-900-09	true
...	

# Les bases de données et le langage SQL → 2. Le langage SQL

## 2.2 L'algèbre relationnelle → Les opérateurs → Projection

Clause	Description	Syntaxe	Exemple
<b>*</b>	Sélectionner toutes les colonnes	<code>SELECT * FROM table_name;</code>	<code>SELECT * FROM factures;</code>
<b>x</b>	Sélectionner un schema	<code>SELECT col1, col2 FROM table_name;</code>	<code>SELECT id, name FROM factures;</code>
<b>AS</b>	Permet de « renommer » un attribut	<code>SELECT col1 AS "ma_col" FROM table_name;</code>	<code>SELECT name AS "nom" FROM factures;</code>
<b>ORDER BY</b>	Tri des résultats	<code>SELECT * FROM table_name ORDER BY col1 ASC DESC</code>	<code>SELECT id, name FROM factures ORDER BY name ASC</code>

# Les bases de données et le langage SQL → 2. Le langage SQL

## 2.2 L'algèbre relationnelle → Les opérateurs → Restriction

Clause	Description	Syntaxe	Exemple
<b>WHERE</b>	Réaliser un filtre des données (condition). Opérateurs possibles : =, >, <, >=, <=, <>	<code>SELECT * FROM table_name WHERE col1 = 'text';</code>	<code>SELECT * FROM factures WHERE numero_facture = 'F800-009';</code>
<b>AND</b> <b>OR</b> <b>NOT</b>	AND = ET logique OR = OU logique NOT = NON logique	<code>SELECT * FROM table_name WHERE condition1 AND condition2;</code> <code>SELECT * FROM table_name WHERE condition1 OR condition2;</code> <code>SELECT * FROM table_name WHERE NOT condition1</code>	<code>SELECT * FROM factures WHERE numero_facture='F800' AND date=2018;</code> <code>SELECT * FROM factures WHERE numero_facture='F800' OR date=2018;</code> <code>SELECT * FROM factures WHERE NOT numero_facture= 'F800';</code>
<b>TOP(x)</b>	Limiter le nombre de résultats	<code>SELECT TOP(qté) * FROM table_name;</code>	<code>SELECT TOP(10) * FROM factures;</code>
<b>LIKE</b>	La clause LIKE est un <u>opérateur</u> « Ressemble à »	<code>SELECT * FROM table_name WHERE col1 LIKE 'text%';</code>	<code>SELECT * FROM factures WHERE numero_facture LIKE 'F800%9';</code>
<b>DISTINCT</b>	Extraire seulement les champs différents	<code>SELECT DISTINCT * FROM table_name;</code>	<code>SELECT DISTINCT client_id FROM factures;</code>
<b>GROUP BY</b>	Grouper les données par champs identiques	<code>SELECT col1, ... FROM table_name GROUP BY col1, ...;</code>	<code>SELECT client_id FROM factures GROUP BY client_id;</code>

Les bases de données et le langage SQL → 2. Le langage SQL

2.2 L'algèbre relationnelle → Les opérateurs → Calcul élémentaire

Clause	Description	Syntaxe	Exemple
()	Réaliser un calcul élémentaire	<code>SELECT (col1 + col2 * col3 / col4), col5 FROM table_name;</code>	<code>SELECT numero, (mntHt * 1,20) FROM factures;</code>

Opérateur	Description
+	Addition
-	Soustraction
*	Multiplication
/	Division
%	Modulo

# Les bases de données et le langage SQL → 2. Le langage SQL

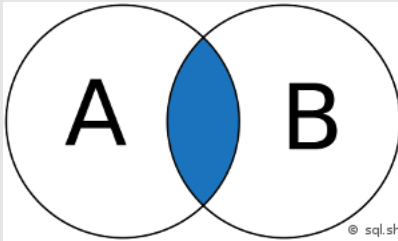
## 2.2 L'algèbre relationnelle → Les opérateurs → Calcul agrégat

Clause	Description	Syntaxe	Exemple
<b>COUNT()</b>	Fonction pour compter le nombre d'occurrences	SELECT COUNT(*) FROM table_name;	SELECT COUNT(*), client_id FROM factures;
<b>AVG()</b>	Fonction pour faire une moyenne	SELECT AVG(col1) FROM table_name;	SELECT AVG(mntNet), client_id FROM factures;
<b>SUM()</b>	pour calculer la somme sur un ensemble d'enregistrement	SELECT SUM(col1) FROM table_name;	SELECT SUM(mntNet), client_id FROM factures;
<b>MAX()</b>	pour récupérer la valeur maximum d'une colonne sur un ensemble de ligne.	SELECT MAX(col1) FROM table_name;	SELECT MAX(mntNet), client_id FROM factures;
<b>MIN()</b>	pour récupérer la valeur minimum de la même manière que MAX()	SELECT MIN(col1) FROM table_name;	SELECT MIN(mntNet), client_id FROM factures;

# Les bases de données et le langage SQL → 2. Le langage SQL

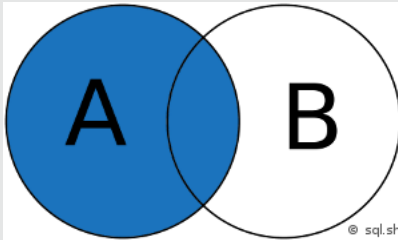
## 2.2 Apprendre la syntaxe du langage SQL → les principales jointures

- Les jointures permettent d'associer 2 tables ensemble (par exemple une table « A » et une table « B ») :
  - INNER JOIN** : jointure interne pour retourner les enregistrements quand la condition est vrai dans les 2 tables. C'est l'une des jointures les plus communes.



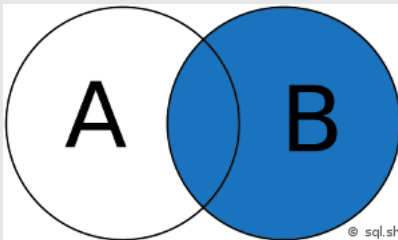
```
SELECT * FROM A INNER JOIN B ON A.key = B.key;
```

- LEFT JOIN** : jointure externe pour retourner tous les enregistrements de la table de gauche même si la condition n'est pas vérifiée dans l'autre table.



```
SELECT * FROM A LEFT JOIN B ON A.key = B.key;
```

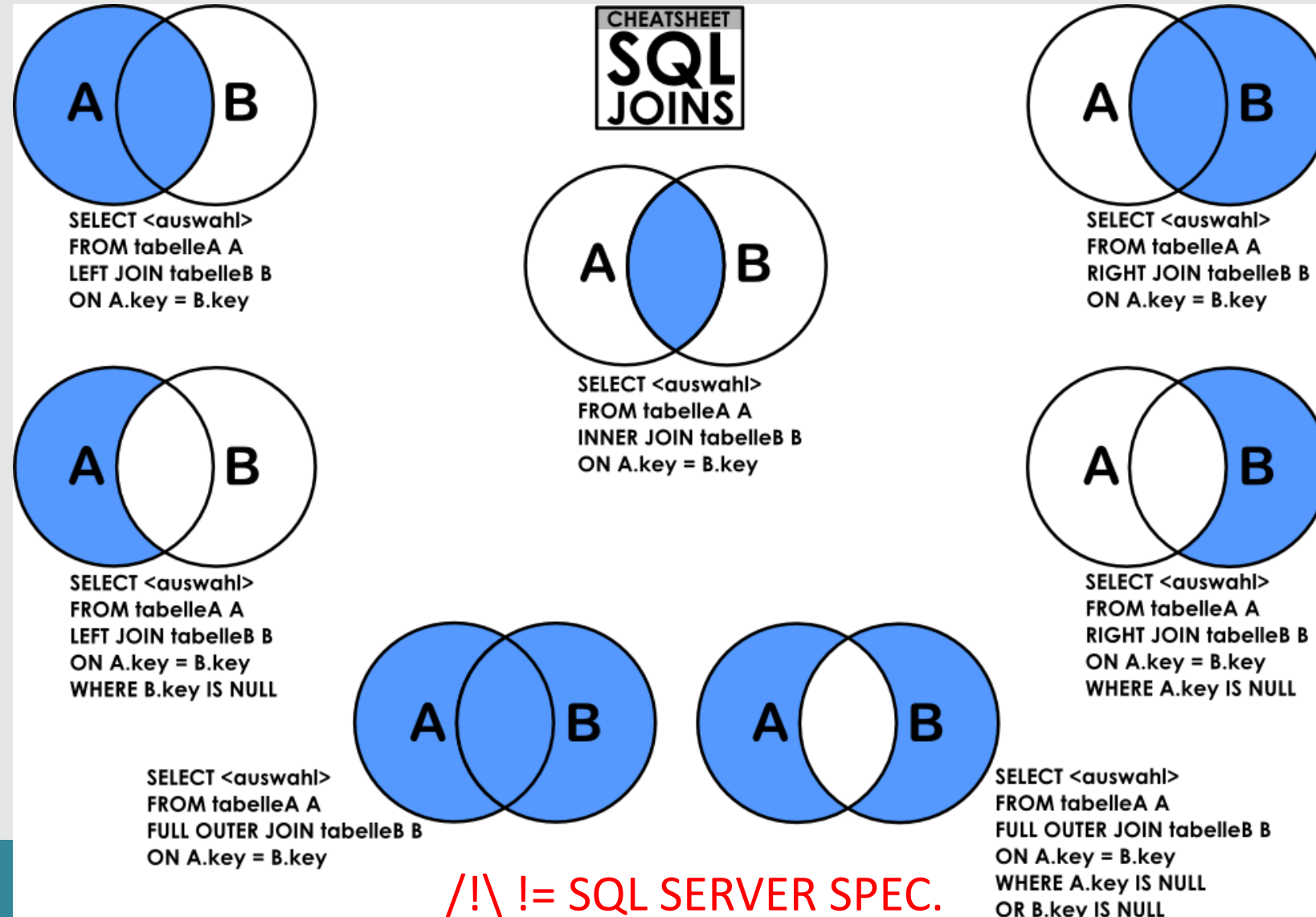
- RIGHT JOIN** : jointure externe pour retourner tous les enregistrements de la table de droite même si la condition n'est pas vérifiée dans l'autre table.



```
SELECT * FROM A RIGHT JOIN B ON A.key = B.key;
```

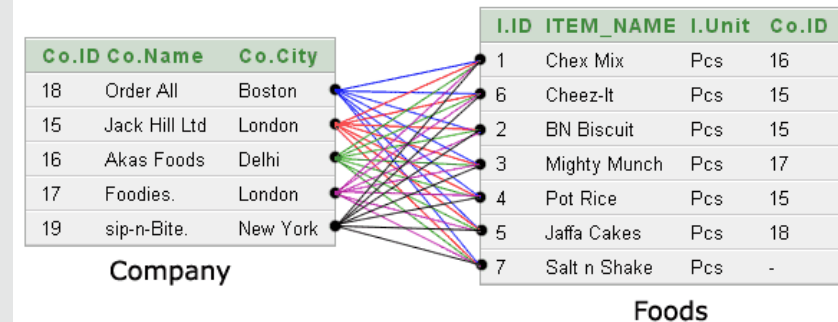
# Les bases de données et le langage SQL → 2. Le langage SQL

## 2.2 Apprendre la syntaxe du langage SQL → toutes les jointures



### produit cartésien : CROSS JOIN

```
SELECT foods.item_name,foods.item_unit,  
company.company_name,company.company_city  
FROM foods  
CROSS JOIN company;
```



#!/ != SQL SERVER SPEC.

# Les bases de données et le langage SQL → 2. Le langage SQL

## 2.2 Apprendre la syntaxe du langage SQL → Les sous requêtes

- Imbriquer une requête dans une autre :

CRUD	Déclaration	Syntaxe
READ	SELECT	<code>SELECT col1, col2, ... FROM table_name WHERE col1 &gt; (SELECT AVG(col1) FROM table_name);</code>
READ	SELECT	<code>SELECT col2, (SELECT AVG(col1) FROM table_name), ... FROM table_name;</code>



Les bases de données et le langage SQL → 1. Les bases de données  
2.2 Travaux pratiques

## 6 - TP - DQL - location

## 2. LE LANGUAGE SQL : ++



# Les bases de données et le langage SQL → 1. Les bases de données

## 2.2 Apprendre la syntaxe du langage SQL : Les notions avancées à connaître

- **Les vues (Views) :**

Les vues permettent de créer des « tables virtuelles ». Une vue permet ainsi de proposer une agrégation de tables afin de simplifier et d'optimiser les performances de requêtes avec jointures.

- **Les transactions :**

Les transactions permettent de « maîtriser les conséquences » des requêtes SQL exécutées. De plus, la transaction permet l'optimisation des performances de le cas de requêtes (très) nombreuses.

- **BONUS : Les procédures stockées (Stored Procedures) :**

Fonction « métier » prenant et retournant des paramètres ou un ensemble de résultats, Les procédures permettent de réaliser des traitements sur les données.

- **BONUS : Les triggers (Database Triggers) :**

Routine exécutée lors de la détection d'une requête (listener) sur la base de données par une personne ou une application. Ainsi, les triggers permettent l'exécution automatique des procédures stockées.

- **BONUS : Les Cluster (de base de données) :**

Le cluster est une redondance de base de données (installation de plusieurs moteur de base de données pour une seule BDD). Le cluster permet d'augmenter considérablement les performances de la base de données.

# Les bases de données et le langage SQL → 2. Le langage SQL

## 2.2 Apprendre la syntaxe du langage SQL : les variables

### Créer et utiliser une variable

```
-- Créer la variable

SET @monId =9

-- Utiliser la variable

INSERT INTO ma_table VALUES (@monId, 'Toto', '10/10/2009');

-- Incrémenter la variable

SET @monId = @monId + 1;
```

# Les bases de données et le langage SQL → 2. Le langage SQL

## 2.2 Apprendre la syntaxe du langage SQL : les vues

Les vues permettent de masquer la complexité des requêtes mais aussi et surtout de proposer un gain de performance pour la requête :

```
-- Créer la vue
CREATE VIEW nom_de_la_vue
AS
-- Début du contenu de la vue
SELECT * FROM ma_table1 INNER JOIN ma_table2 ON ma_table1.ma_table2_id = ma_table2.id
-- Fin du contenu de la vue
-- Commit de la vue
GO
```

# Les bases de données et le langage SQL → 2. Le langage SQL

## 2.2 Apprendre la syntaxe du langage SQL : les transactions

Les transactions permettent d'exécuter une suite d'instructions, de vérifier l'impact sur les données, puis de valider (ou annuler) les actions.

```
-- Ouverture de la transaction
BEGIN TRAN nom_de_la_transaction;

-- suite de requêtes encapsulées dans la transaction
UPDATE factures SET numero_facture = 'nouveau_numero' WHERE numero_facture = ancien_numero';
SELECT * FROM factures WHERE numero_facture = 'nouveau_numero';

-- les requêtes fonctionnent → valider les modifications
COMMIT TRAN nom_de_la_transaction;

-- les requêtes ne fonctionnent pas → annuler les modifications
-- attention : le ROLLBACK est inopérant si le COMMIT est réalisé précédemment!
ROLLBACK TRAN nom_de_la_transaction;
```

# Les bases de données et le langage SQL → 2. Le langage SQL

## 2.2 Apprendre la syntaxe du langage SQL : les triggers

Les triggers permettent l'exécution de « procédure » SQL. Le déclencheur est généralement une action réalisée sur la base de données (INSERT/UPDATE/DELETE/...). Exemple :

```
-- Creation d'un trigger
CREATE TRIGGER nom_trigger
-- Déclenchement du trigger
ON ma_table AFTER INSERT, UPDATE AS
    -- Action d'un trigger
    UPDATE ma_table SET ma_col_2 = (SELECT ma_col_1 FROM INSERTED) * 2;
GO
```

# Les bases de données et le langage SQL → 2. Le langage SQL

## 2.2 Apprendre la syntaxe du langage SQL : les procédures stockées

Les procédures stockées permettent de réaliser/variabiliser des séquences « d'actions ». Très souvent utilisées dans les « routines » SQL.

```
-- Créer une procédure stockée
CREATE PROCEDURE my_first_proc
    @lastame varchar(20),
    @firstname char(20)
AS
    -- Réaliser LES opérations souhaitées
    SELECT *
    FROM employe
    WHERE nom = @firstname AND prenom = @lastame;
GO
-- Executer une procédure stockée
EXECUTE my_first_proc @lastame='Arthur', @firstname='Raimbaud';
```



Les bases de données et le langage SQL → 1. Les bases de données  
2.2 Travaux pratiques

**TP FINAL :DVD**

# 3. RESSOURCES



## Les bases de données et le langage SQL → 3. Ressources

Type	Ressource
W3C – SQL (officiel)	<a href="https://www.w3schools.com/sql/">https://www.w3schools.com/sql/</a>
Syntaxes SQL (non officiel)	<a href="https://sql.sh">https://sql.sh</a>
Télécharger SSMS	<a href="https://www.microsoft.com/fr-fr/download/details.aspx?id=8961">https://www.microsoft.com/fr-fr/download/details.aspx?id=8961</a>
Documentation SSMS	<a href="https://docs.microsoft.com/en-us/sql/ssms/sql-server-management-studio-ssms?view=sql-server-2017">https://docs.microsoft.com/en-us/sql/ssms/sql-server-management-studio-ssms?view=sql-server-2017</a>
Aller plus loin	<a href="https://openclassrooms.com/fr/courses/4449026-initiez-vous-a-lalgebre-relationnelle-avec-le-langage-sql">https://openclassrooms.com/fr/courses/4449026-initiez-vous-a-lalgebre-relationnelle-avec-le-langage-sql</a>