

# Manipulation Des Bases de Données



Formatrice : Meriem Onzy

E-mail : [Meriem.onzy@ofppt.ma](mailto:Meriem.onzy@ofppt.ma)

# Plan du Rappel

1

SGBD et  
Langage SQL

2

Bases de  
données  
Relationnelles

3

Niveaux de  
description des  
données

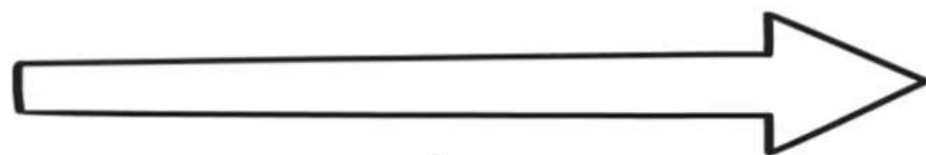
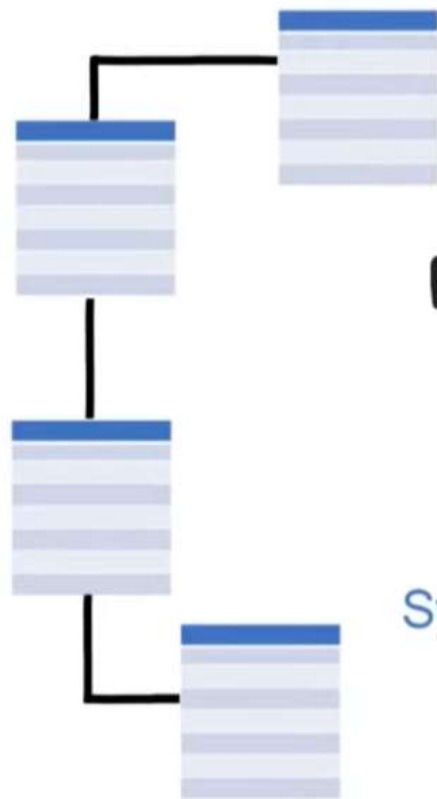
4

Objectifs des  
SGBD

5

Architecture  
Client-Serveur

Données



Système de Gestion de Bases de Données  
(SGBD)

Support de Stockage



# SGBD

Un système de Gestion de Base de Données est un logiciel système qui permet de stocker nos données dans un support de stockage.

Stocker nos  
Créer  
Gérer  
Manipuler



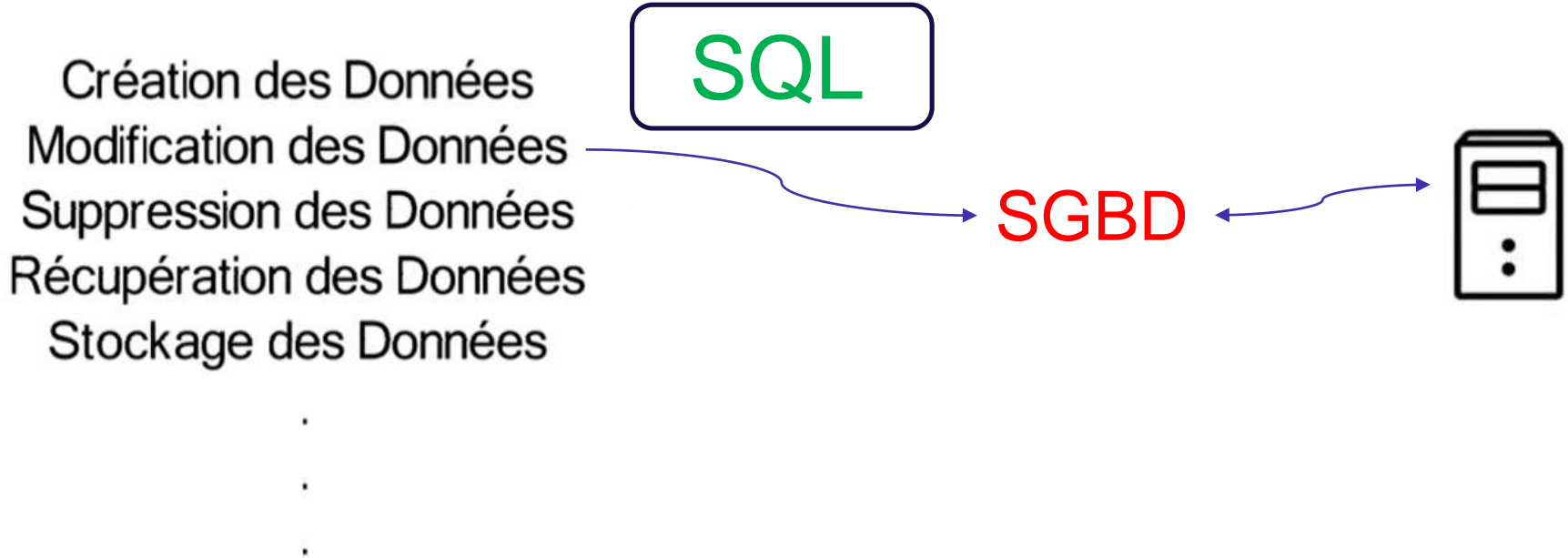
Prénom  
Nom  
Date Naissance  
Genre  
Adresse E-mail  
Mot de Passe



SGBD



# Langage SQL



# SQL

C'est un langage informatique normalisé servant à gérer les bases de données.

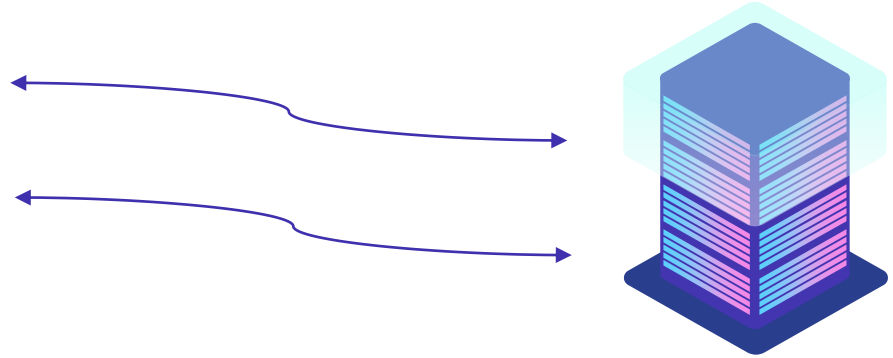
Le SQL nous permet de communiquer avec le SGBD.

Grâce au langage SQL, on pourra manipuler nos données et réaliser les différentes opérations: Recherche, Ajout, Suppression et Modification des données

# Bases de données relationnelles

Les données déjà insérées lors de l'inscription

- Prénom
- Nom
- Adresse e-mail
- Mot de Passe
- Date de naissance



# Bases de données

Une base de données est un ensemble de données stockées de manière structurée, organisée et avec le moins de redondance possible.

## Pourquoi utilise-t-on les bases de données ?

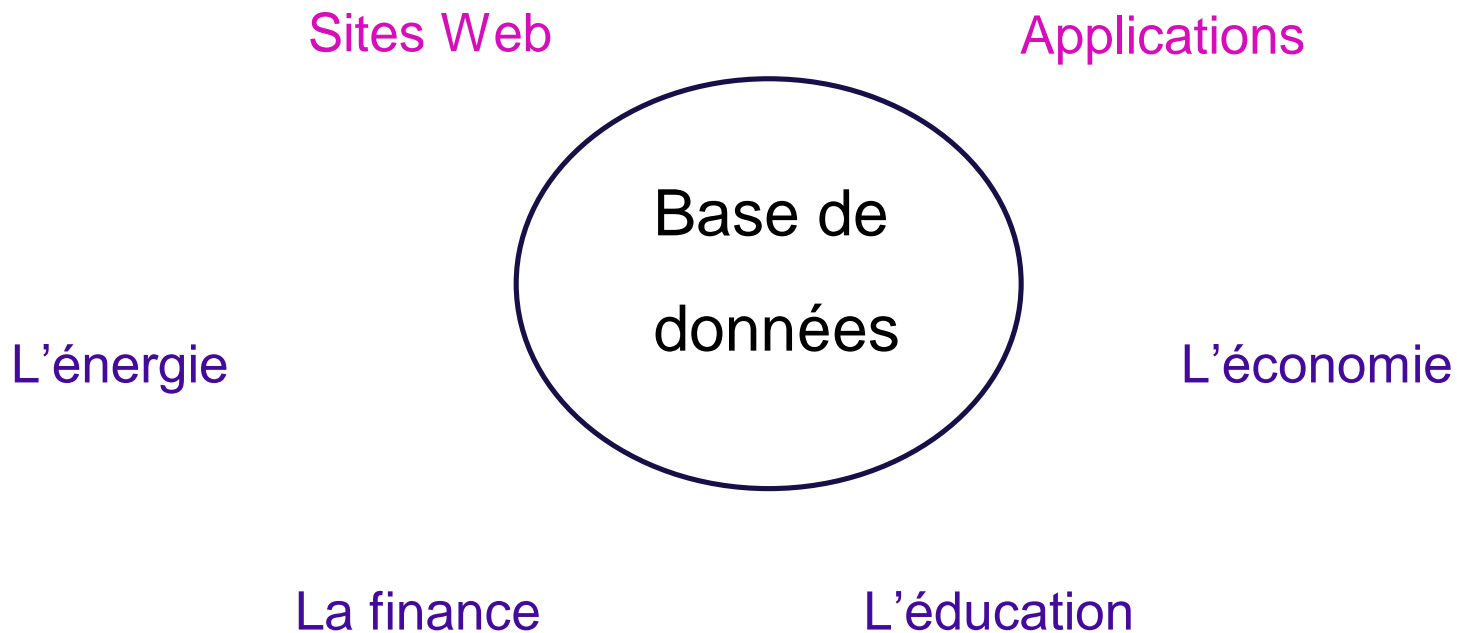
- Assurer la qualité des données (Structurées, Bien Organisées)
- Exploitation des données (Simplement, Efficacement)



# Des Bases De Données



# Des Bases De Données



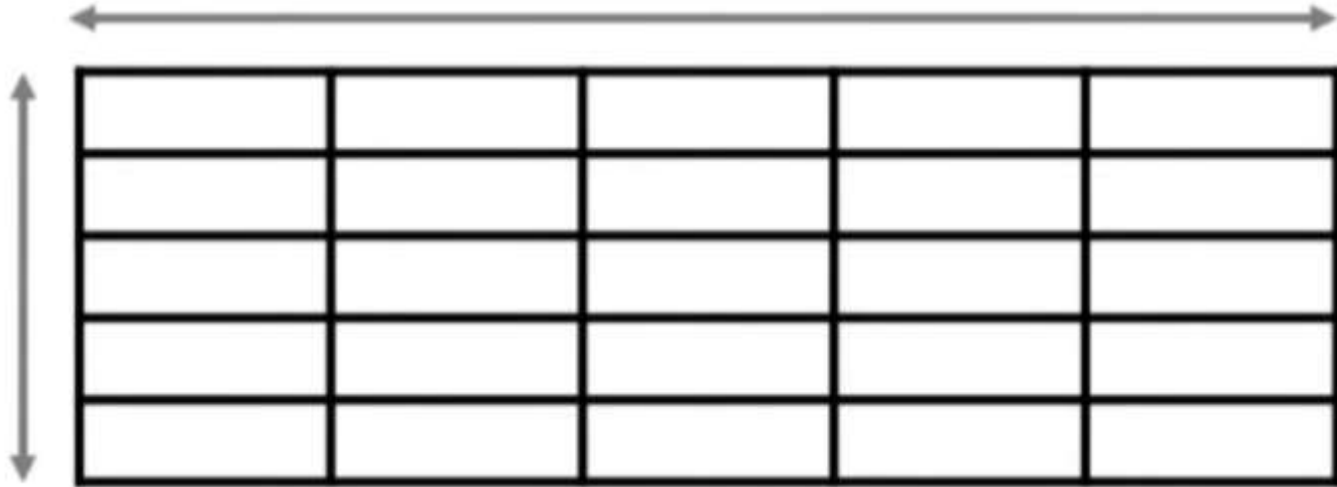


# Bases de données Relationnelles

## Types de Bases de données

# BASE DE DONNEES RELATIONNELLES

Une base de données relationnelle est composée de Tables à 2 Dimensions



A diagram illustrating a 2D table structure. It consists of a grid with 5 rows and 5 columns. Above the grid, a horizontal double-headed arrow spans the width of the table. To the left of the grid, a vertical double-headed arrow spans the height of the table.


Une table est appelée une relation

# BASE DE DONNEES RELATIONNELLES

Enregistrement / Entité




# BASE DE DONNEES RELATIONNELLES



**Table: Membres inscrits**

# BASE DE DONNEES RELATIONNELLES

Attributs



Enregistrement / Entité




# BASE DE DONNEES RELATIONNELLES

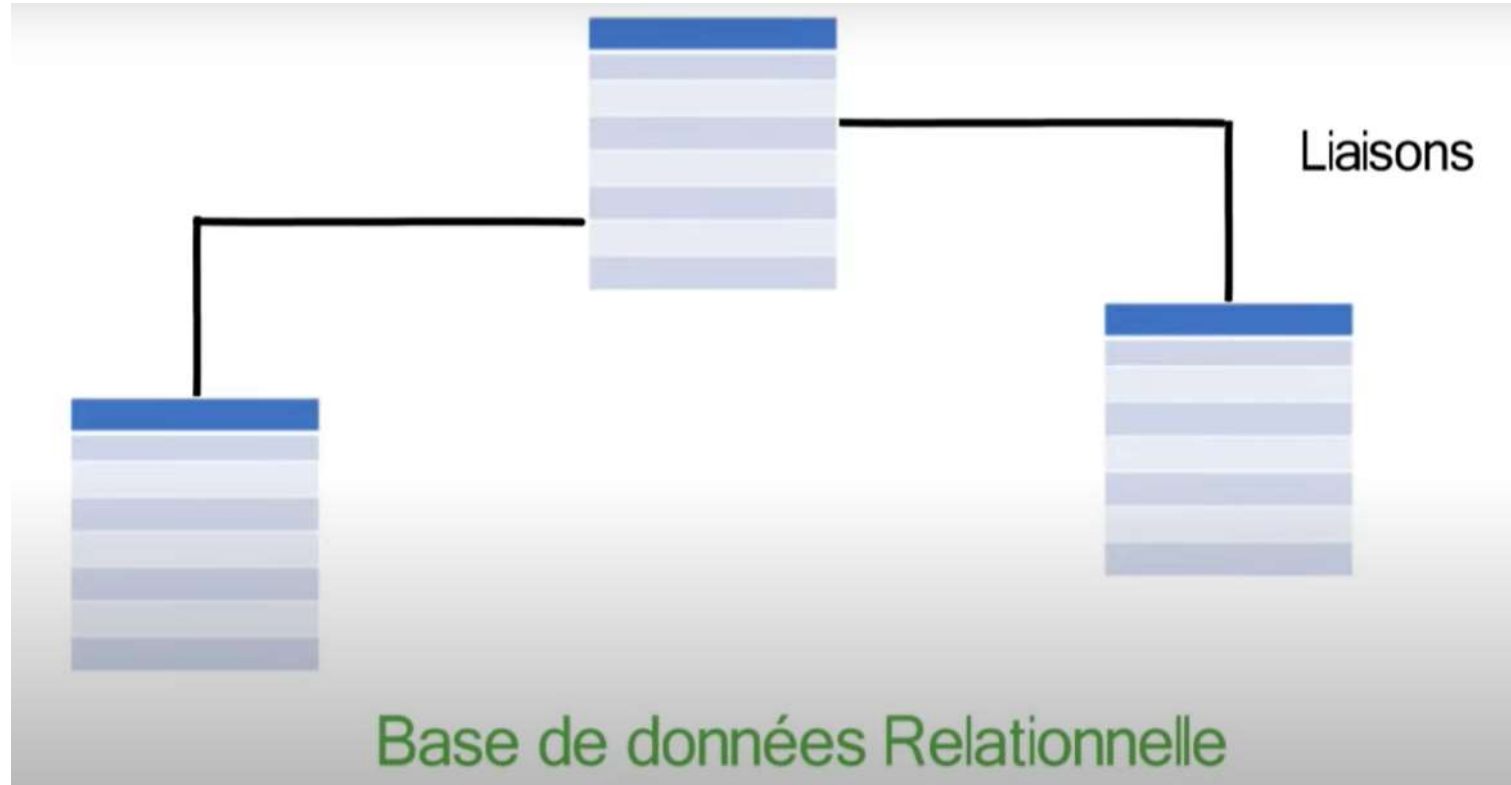
[illegible]



# BASE DE DONNEES RELATIONNELLES

Prénom	Nom	Date Naissance	Genre	Email	Mot de Passe
Peter	PARKER	01/01/1987	M	<a href="mailto:peter.parker@gmail.com">peter.parker@gmail.com</a>	<u>XXXXXXXXXXXXXXXX</u>
Santa	CANADAY	07/12/1998	F	<a href="mailto:scanaday@gmail.com">scanaday@gmail.com</a>	<u>XXXXXXXXXXXXXXXX</u>
Thomas	HOGAN	28/05/1997	M	<a href="mailto:thomas.hogan@gmail.com">thomas.hogan@gmail.com</a>	<u>XXXXXXXXXXXXXXXX</u>
Samuel	Derry	14/10/1995	M	samuel.derry@yahoo.fr	*****

# BASE DE DONNEES RELATIONNELLES



# Bases de données NoSQL

Toute Base de Données ne respectant pas le modèle relationnel est une  
Base de Données NoSQL

- Base de donnée orientée colonnes
- Base de donnée orientée Document
- Base de donnée orientée Clé - Valeur

# Base de données



**BASES DE DONNEES = DONNEES STRUCTUREES + SGBD**



**UN SGBD PERMET DE MANIPULET ET GERER LES DONNEES**



**C'EST UN LOGICIEL QUI PREND EN CHARGE :**

CREATION, DESCRIPTION ET STRUCTURATION DES DONNEES

MODIFICATION ET MISA A JOUR DES DONNEES

INTERROGATION ET RECHERCHE DES DONNEES

ADMINISTRATION DE LA BASE DE DONNEES

# Notion du SGBD

LES SGBD SONT DES SYSTEMES PROPRIETAIRES.  
ILS APPARTIENNENT A DES CONSTRUCTEURS D'ORDINATEURS  
ET EDETEURS DE LOGICIELS

**ORACLE®**



**ORACLE**



**MICROSOFT**



**ORACLE**

# ENTREPRISE ABC

## DÉPARTEMENT ACHATS

## DÉPARTEMENT VENTES

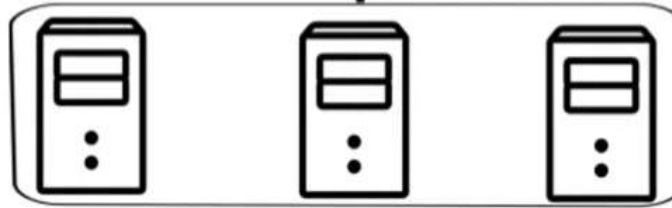
## DÉPARTEMENT FINANCE

## DÉPARTEMENT RH



SGBD

BASE DE DONNÉES  
DE L'ENTREPRISE



- DONNEES ACHATS
- DONNEES VENTES
- DONNEES FINANCE
- DONNEES RH

# NIVEAUX DE DESCRIPTION DES DONNEES

## SCHEMAS DE DONNEES

NIVEAU EXTERNE



NIVEAU CONCEPTUEL



NIVEAU INTERNE (PHYSIQUE)

# Schéma Externe



DECRIE COMMENT UN UTILISATEUR/ PROGRAMME PERCOIT LES DONNEES  
AUXQUELLES IL A ACCES



**NOTION DE VUE** : SOUS-ENSEMBLE D'UNE BASE DE DONNEES



**VUE 1: DONNÉES RH**



**VUE 2: DONNÉES FINANCE**



ASSURER LA SECURITE DES DONNEES



# Schéma Conceptuel / Modélisation Conceptuelle



**TRADUIT LES NOTIONS DU MONDE REEL EN UN LANGAGE DE HAUT NIVEAU**

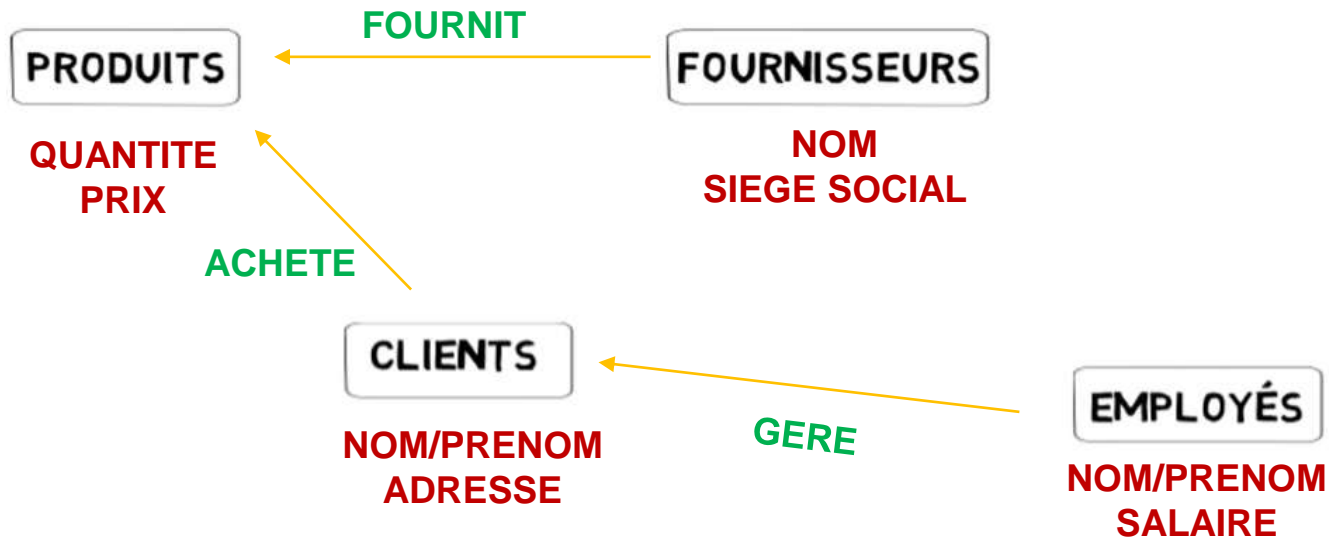


**LE SCHEMA CONCEPTUEL PERMET DE DEFINIR :**

LES OBJETS DU MONDE REEL

LES PROPRIETES DE CES OBJETS

LES LIAISONS ENTRE EUX



# Schéma Interne



**CORRESPOND A LA STRUCTURE MISE EN PLACE POUR ASSURER LE STOCKAGE DES DONNEES**



**FOURNIR LES INFORMATIONS :**

L'ESPACE DE STOCKAGE RESERVE A CHAQUE INFORMATION

LES FICHIERS CONTENANT LES DONNEES ET LEUR EMPLACEMENT

# Modélisation des données

## Merise

	Communication	Données	Traitement
Conceptuel	MCC: Modèle Conceptuel de Communication	<u>MCD: Modèle Conceptuel de Données</u>	MCT: Modèle Conceptuel de Traitement
Organisationnel	MOC: Modèle Organisationnel de Communication	MOD: Modèle Organisationnel de Données	MOT: Modèle Organisationnel de Traitement
Logique	MLC: Modèle Logique de Communication	<u>MLD: Modèle Logique de Données</u>	MLT: Modèle Logique de Traitement
Physique	MPC: Modèle Physique de Communication	<u>MPD: Modèle Physique de Données</u>	MPT: Modèle Physique de Traitement

# Modélisation des données



## Dictionnaire des données

Le dictionnaire de données est un document qui permet de recenser, de classer et de trier toutes les données relevées du cahier des charges

Nom de la donnée	Format	Longueur	Type		Règle de calcul	Règle de gestion	Document
			Élémentaire	Calculé			

1. *Nom de la donnée* : Nom client par exemple ;
2. *Format* : alphabétique, alphanumérique, numérique, date, logique;
3. *Longueur* : approximative ou exacte.
4. *Type* : on met une X pour préciser est ce que c'est une donnée élémentaire ou calculée.
5. *Règle de calcul* : la formule de calcul pour une donnée de type calculée.
6. *Règle de gestion* : on indique (si nécessaire) la règle de gestion relative à la donnée.
7. *Document* : on mentionne le document dans lequel a été trouvée la donnée

# Modélisation des données



## Dictionnaire des données

Le dictionnaire de données est un document qui permet de recenser, de classer et de trier toutes les données relevées du cahier des charges

Nom	Format	Longueur	Type		Règle de calcul	Règle de gestion	Document
			E	C			
Numéro	Numérique		X				Fiche
Nom	Alphabétique	30	X				//
Prénom	Alphabétique	30	X				//
Adresse	Alphabétique	50	X				//
Code Postal	Alphanumérique	10	X				//
Ville	Alphabétique	50	X				//
Téléphone	Alphanumérique	15	X				//
Mail	Alphanumérique	50	X				//
Date d'adhésion	Date		X				//

# Modélisation des données



## Dictionnaire des données

Le dictionnaire de données est un document qui permet de recenser, de classer et de trier toutes les données relevées du cahier des charges

Code donnée	Désignation	Type	Taille	Observation
numCINEtu	Numéro CIN	Alphanumérique	9	Identifiant de l'étudiant
nomEtu	Nom de l'étudiant	Alphabétique	30	
prenomEtu	Prénom de l'étudiant	Alphabétique	30	
dateNaissEtu	Date de naissance	Date		
niveauEtu	Niveau scolaire	Alphanumérique	15	
nomvilleEtu	Nom de la ville	Alphabétique	15	
AdresseEtu	Adresse de l'étudiant	Alphanumérique	90	

# Modélisation des données



## Dépendances Fonctionnelles

Code de Formation	Titre de Formation	Durée	Prix
ID01	Introduction au développement	3 mois	2500
CCP01	C/C++	30 jours	3000
ID02	Introduction au développement	3 mois	2700
BD001	Base de données	30 jours	2500

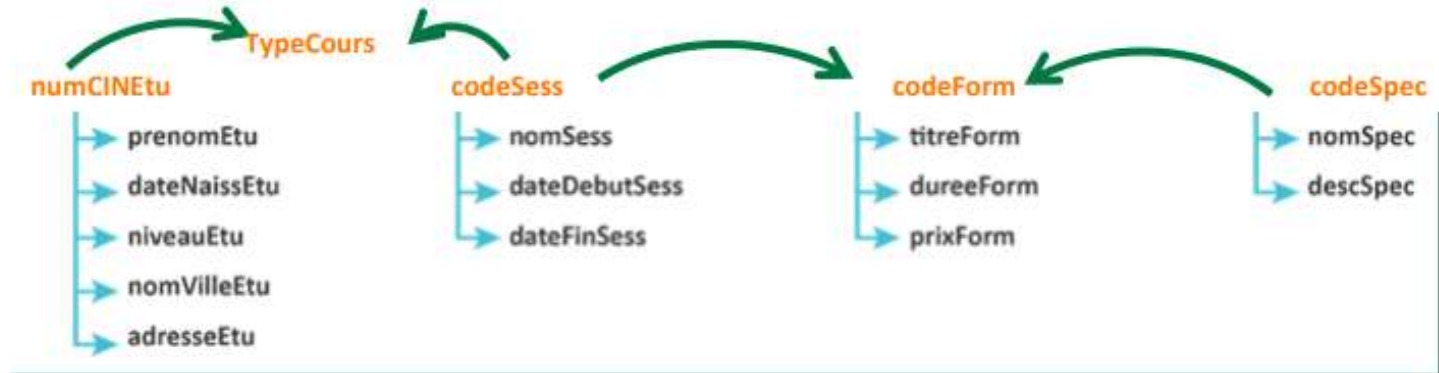
- Le titre de formation dépend du code de formation
- Cette relation est symbolisée sous cette forme :  
codeForm -> titreForm



# Modélisation des données



## Graphe des Dépendances Fonctionnelles



# Modélisation des données



## Modèle Conceptuel des données

Le modèle conceptuel de données est un modèle qui nous permet de concevoir le schéma de données utilisables dans notre SI.

Il décrit de façon formelle les données utilisées par le SI.

### Les éléments de base d'un MCD :

- ☐ Les propriétés
- ☐ Les entités
- ☐ Les relations

# Modélisation des données



## Modèle Conceptuel des données

**Les propriétés** : sont les informations de base qui décrivent les éléments (les entités).

Exemple : Le numéro client, nom clients, prénom client, adresse client

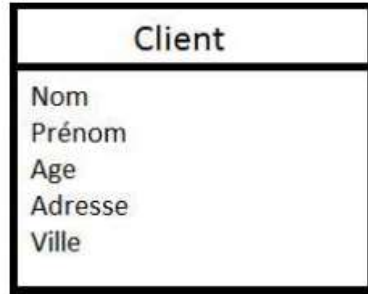
Chaque propriété dispose d'un type(alphabétique, alphanumérique, numérique, date, logique..)

# Modélisation des données



## Modèle Conceptuel des données

**Une entité :** Une entité est la représentation d'un élément dans un SI. Chaque entité regroupe un ensemble de propriétés.



# Modélisation des données



## Modèle Conceptuel des données

**Un identifiant:** Une propriété qui permet de connaître de façon unique et sûre les occurrences d'une entité donnée.

Client
<u>Numéro</u>
Nom
Prénom
Age
Adresse
Ville

Client 1
<u>Numéro</u> :1
Nom : Amin
Prénom : Jamil
Age : 33
Adresse : Rue Farah
Ville : Nador

Client 2
<u>Numéro</u> : 2
Nom : Mohamed
Prénom : Salim
Age : 45
Adresse : Rue M5
Ville : Oujda

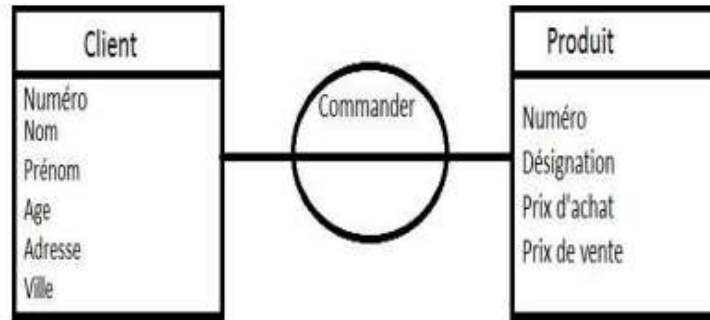
Client 3
<u>Numéro</u> : 3
Nom : Tarik
Prénom : Tribek
Age : 26
Adresse : Rue Maarif
Ville : Casa

# Modélisation des données



## Modèle Conceptuel des données

**Les associations :** Une relation ou association est la liaison qui lie entre les entités du modèle de donnée.

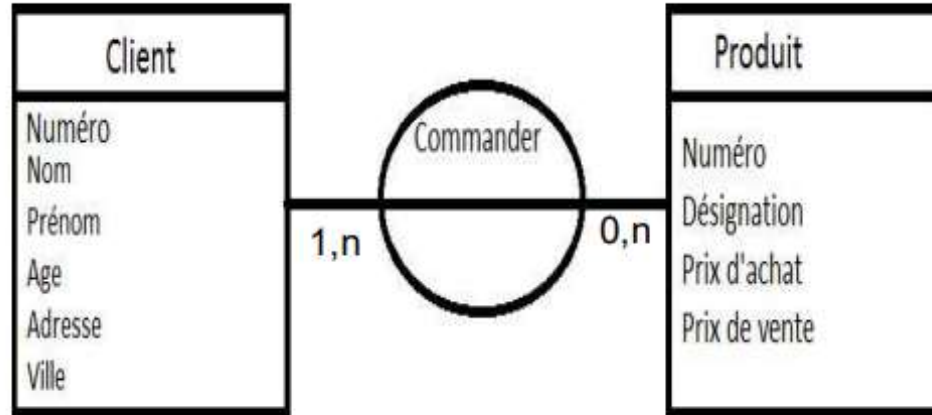


# Modélisation des données



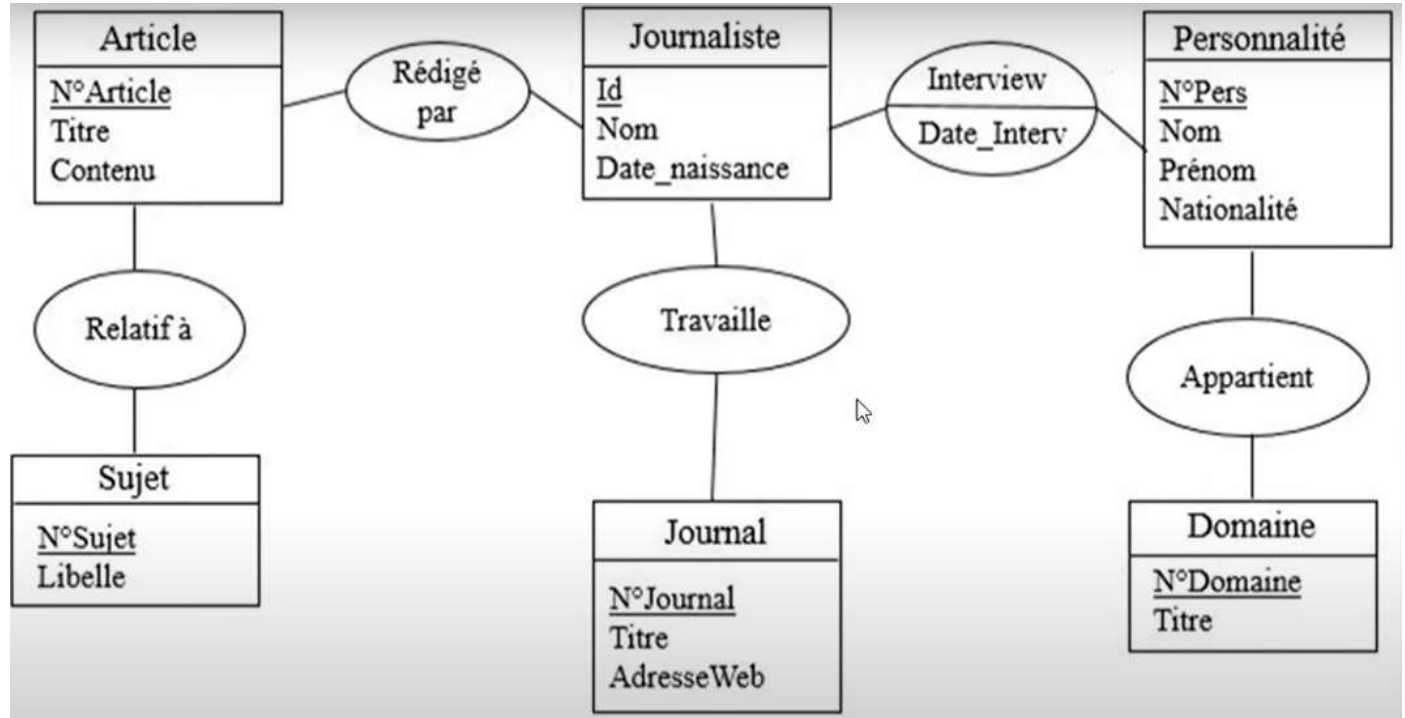
## Modèle Conceptuel des données

### Les cardinalités



# Activité

On considère le modèle conceptuel suivant

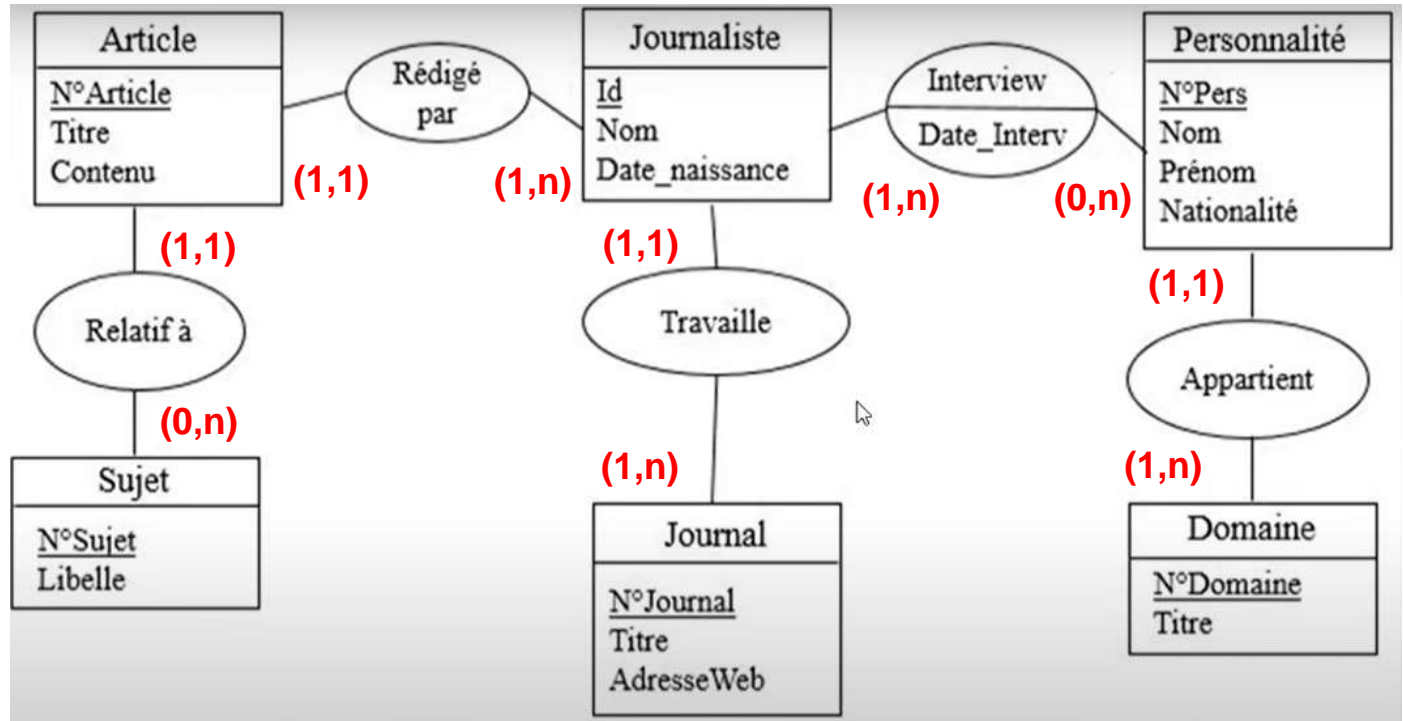


Ajouter les cardinalités selon les règles de gestion



# Activité

On considère le modèle conceptuel suivant



Ajouter les cardinalités selon les règles de gestion

## Exercice 2

« Les restaurateurs achètent des produits alimentaires ».

En considérant que ces achats sont effectués dans le cadre de commandes, les règles de gestion suivantes ont été retenues :

Une commande est passée par un seul restaurateur et elle concerne des produits alimentaires. Un restaurateur peut passer plusieurs commandes et un produit alimentaire peut faire partie de plusieurs commandes.

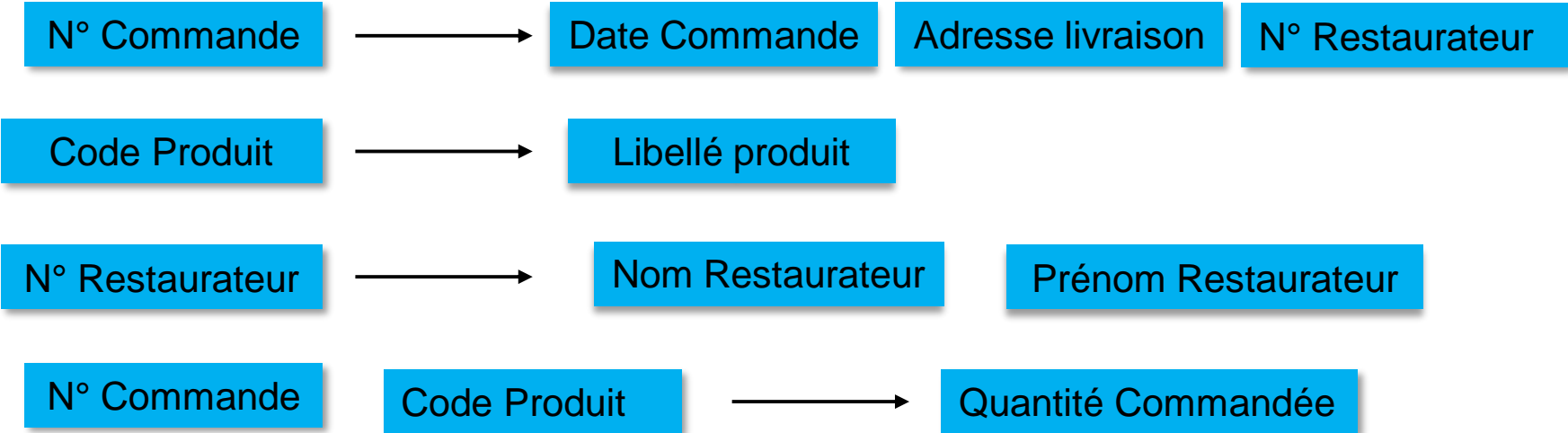
En retenant la liste des propriétés ci-dessous :

- Code produit alimentaire
- Numéro restaurateur
- Numéro commande
- Libellé produit alimentaire
- Date commande
- Nom restaurateur
- Prénom restaurateur
- Quantité commandée
- Adresse livraison

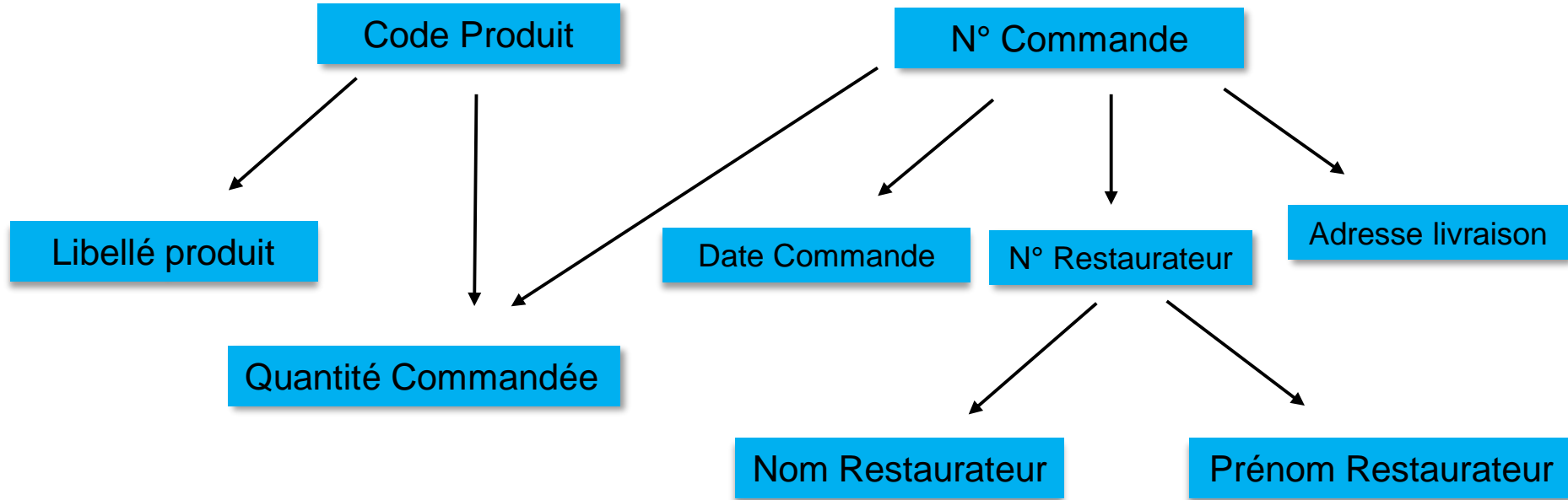
# Solution

Une commande est passée par un seul restaurateur et elle concerne des produits alimentaires. Un restaurateur peut passer plusieurs commandes et un produit alimentaire peut faire partie de plusieurs commandes.

Code produit	N° restaurateur	N° commande	Libellé produit	Date commande
Nom restaurateur	Prénom restaurateur	Quantité commandée	Adresse livraison	



# Solution



# Modélisation des données



## La normalisation

L'objectif de la normalisation est de construire un modèle de données cohérent.

Un MCD incohérent implique un mauvais modèle logique qui peut conduire à des anomalies lors de la phase de manipulation des données

# Modélisation des données



## La normalisation

## Intérêt de la normalisation

Nom Fournisseur	Adresse Fournisseur	Produit	Prix
Brahimi	10, Rue Allal Fassi – Rabat-	Chaise	105,00
		Table	255,00
Filali	28, Rue les iris – Marrakech-	Bureau	1250,00
Soufiani	25, Rue les roses – Tanger -	Lit	1800,00
		Chaise	100,00
Brahimi	1, Rue Fassi – Rabat-	Bureau	1500,00

## Problèmes

Pas d'identifiant

L'adresse

Nom Fournisseur	Adresse Fournisseur	Produit	Prix
Brahimi	10, Rue Allal Fassi – Rabat-	Chaise	105,00
		Table	255,00
Filali	28, Rue les iris – Marrakech-	Bureau	1250,00
Soufiani	25, Rue les roses – Tanger -	Lit	1800,00
		Chaise	100,00
Brahimi	1, Rue Fassi – Rabat-	Bureau	1500,00

- L'adresse du fournisseur sera dupliquée dans chaque enregistrement (redondance)
- Si on souhaite modifier l'adresse d'un fournisseur, il faudra rechercher et mettre à jour tous les enregistrements correspondants à ce fournisseur
- Si on insère un nouveau produit pour un fournisseur déjà référencé, il faudra vérifier que l'adresse est identique

# Modélisation des données



## La normalisation

## Intérêt de la normalisation

La normalisation garantit la cohérence et élimine les redondances, ce qui permet :

- **Une diminution de la taille des données.**
- **Une diminution des risques d'incohérence.**
- **D'éviter une mise à jour multiple des mêmes données.**





# La normalisation

## 1 ère Forme Normale

**Une entité ou une association est considéré est dit de 1 ère Forme Normale (1NF) si toutes ses propriétés sont valides:**

- **Atomique** : Non subdivisible
- **Non répétitives** : à savoir que deux ou plusieurs propriétés ne doivent pas stocker la même information
- **Significatives**
- Chaque entité possède un **identifiant**



# La normalisation

## 2<sup>e</sup> Forme Normale

**Une association est dite en 2<sup>e</sup> Forme Normale (2FN) si :**

- Elle est en 1<sup>e</sup> Forme Normale
- Toutes ses propriétés différentes de l'identifiant sont en **dépendance fonctionnelle avec l'identifiant**



# La normalisation

## 3<sup>e</sup> Forme Normale

**Une association est dite en 3<sup>e</sup> Forme Normale (3FN) si :**

- Elle est en 2<sup>e</sup> Forme Normale
- Il n'existe **pas de dépendance** fonctionnelle entre les propriétés **non-identifiants**



## Dérivation d'un MLD à partir d'un MCD

### Entité Faible

Une entité est dite faible dans l'association si elle possède un maximum de 1  
**(0,1) (1,1)**

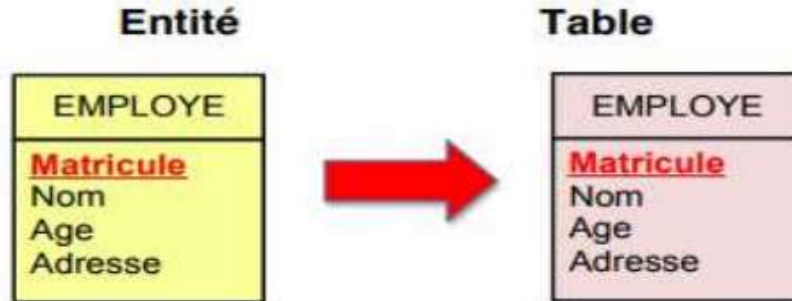
### Entité Forte

Une entité est dite forte dans l'association si elle possède un maximum de n  
**(1,n) (0,n)**



## Dérivation d'un MLD à partir d'un MCD

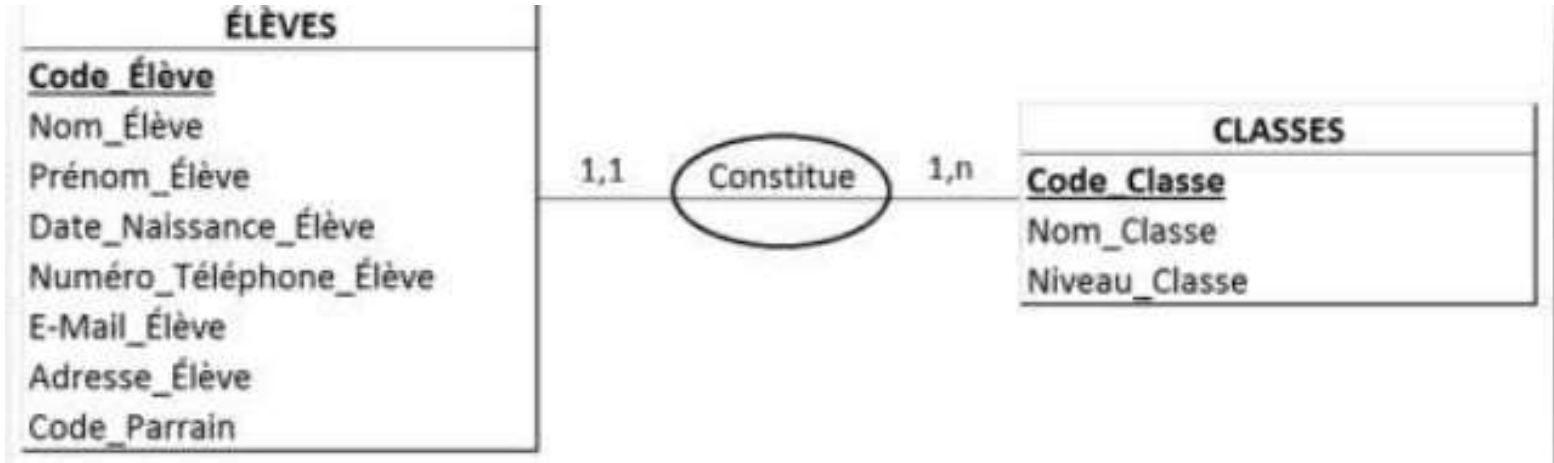
Toute entité devient **une relation** dans laquelle les propriétés deviennent **les attributs**. L'identifiant de l'entité constitue alors la **clé primaire** de la relation.





## Dérivation d'un MLD à partir d'un MCD

### Relation binaire entre une entité forte et une entité faible





## Dérivation d'un MLD à partir d'un MCD

### Relation binaire entre une entité forte et une entité faible

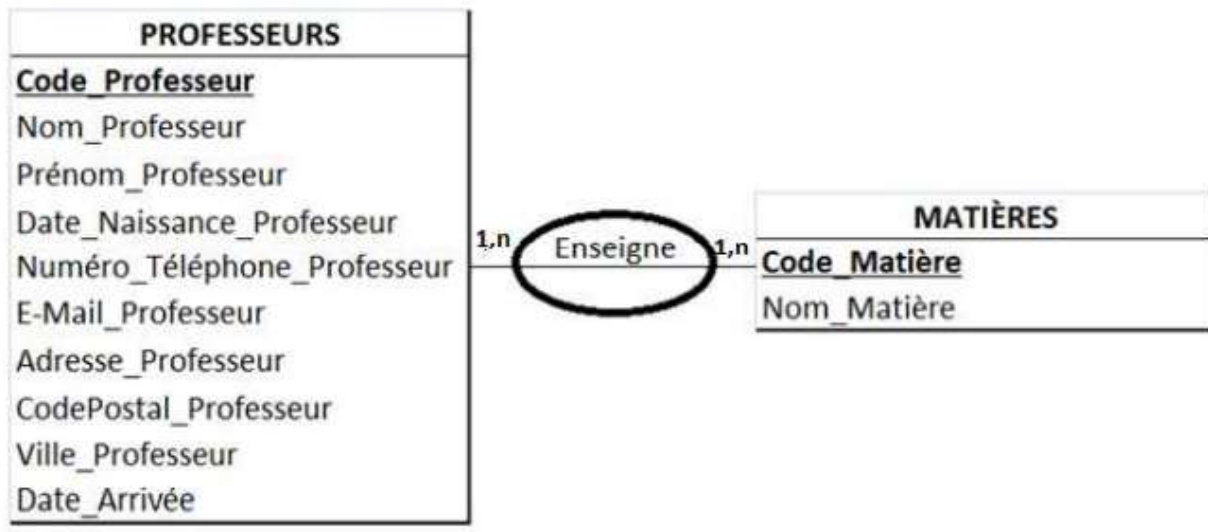
ÉLÈVES
<u>Code Élève</u>
Nom_Élève
Prénom_Élève
Date_Naissance_Élève
Numéro_Téléphone_Élève
E-Mail_Élève
Adresse_Élève
CodePostal_Élève
Ville_Élève
Code_Parrain
#Code_Classe

CLASSES
<u>Code Classe</u>
Nom_Classe
Niveau_Classe



## Dérivation d'un MLD à partir d'un MCD

### Une association entre 2 entités fortes

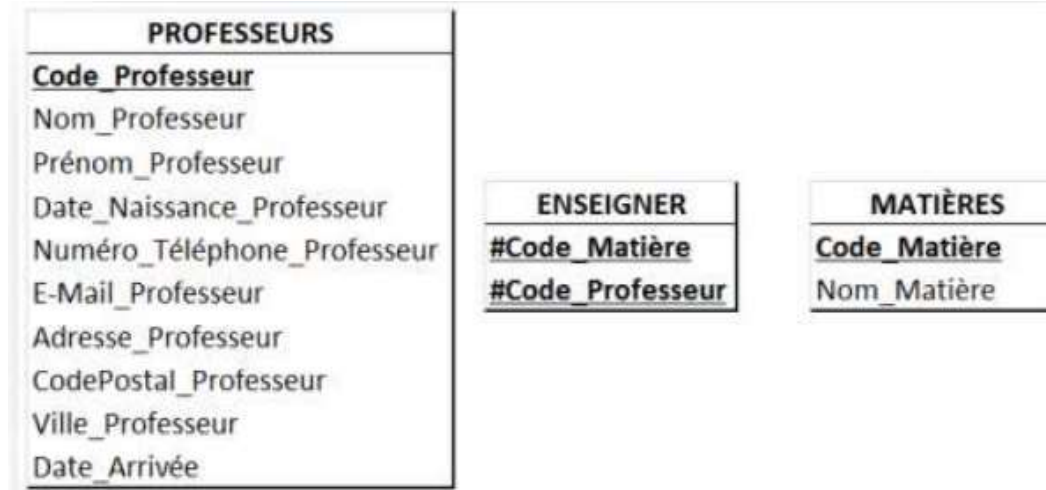






## Dérivation d'un MLD à partir d'un MCD

### Une association entre 2 entités fortes



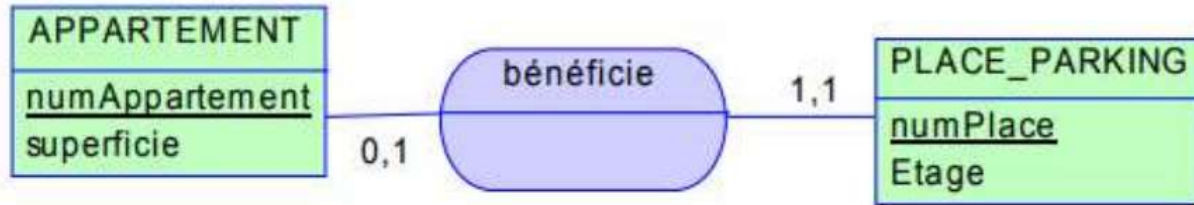
L'association se transforme en une relation et absorbe les identifiants des 2 entités liées.  
L'identifiant devient alors l'union des 2 clés étrangères



## Dérivation d'un MLD à partir d'un MCD

### Une association entre 2 entités faibles

CAS 1 : La cardinalité minimale de E1 est 0 et celle de E2 est 1



L'identifiant de E1 est absorbé par E2

APPARTEMENT (numAppartement, superficie)

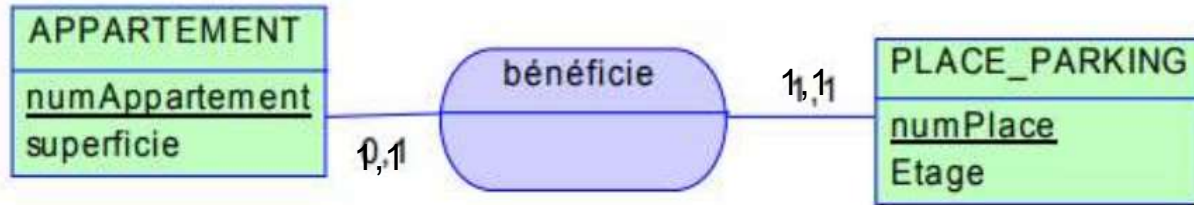
PLACE\_PARKING(num Place, Etage, #numAppartement)



## Dérivation d'un MLD à partir d'un MCD

### Une association entre 2 entités faibles

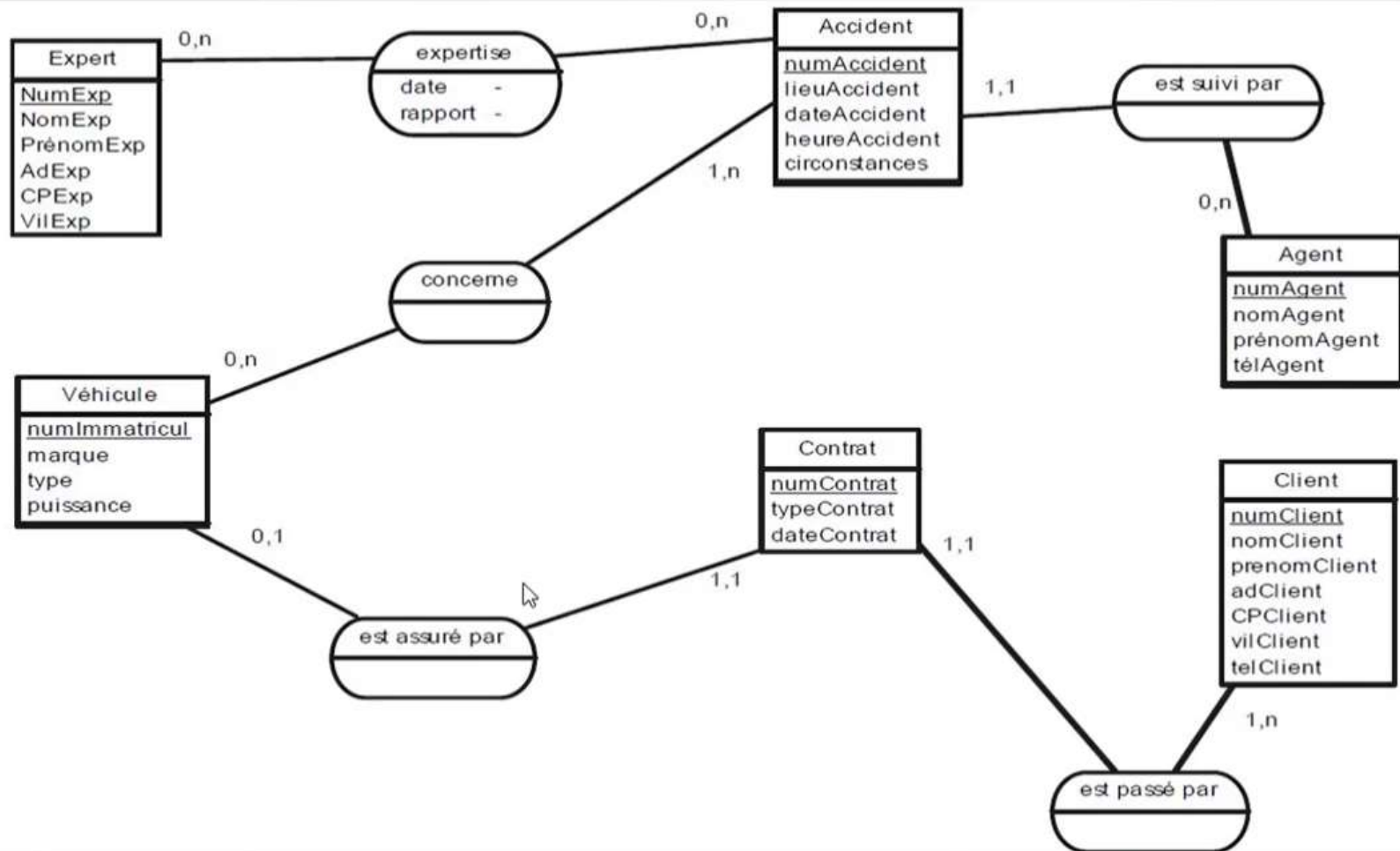
CAS 1 : La cardinalité minimale de E1 et E2 est 1

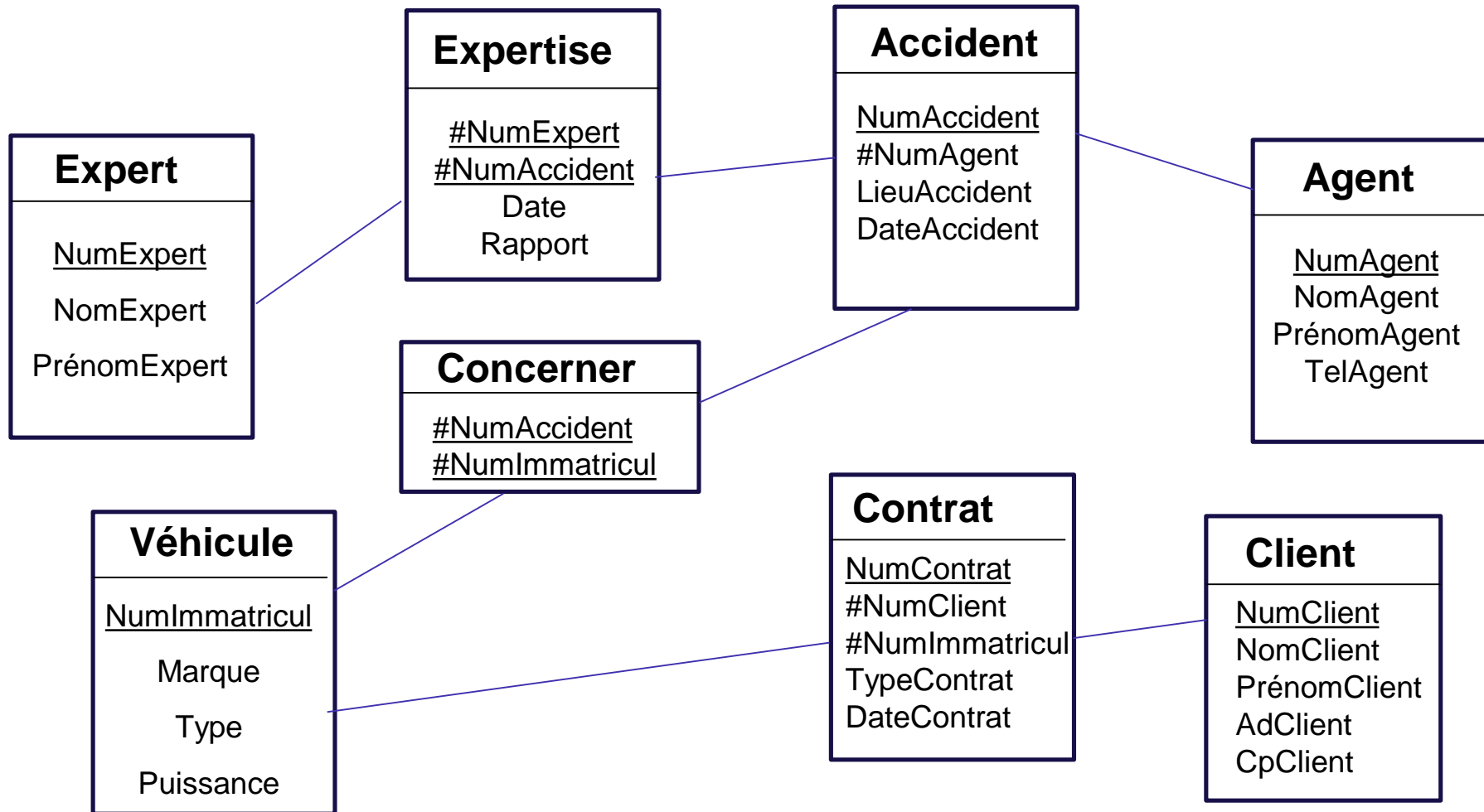


Les deux entités échangent leurs identifiants

APPARTEMENT (numAppartement, superficie, #numPlace)

PLACE\_PARKING(num Place, Etage, #numAppartement)







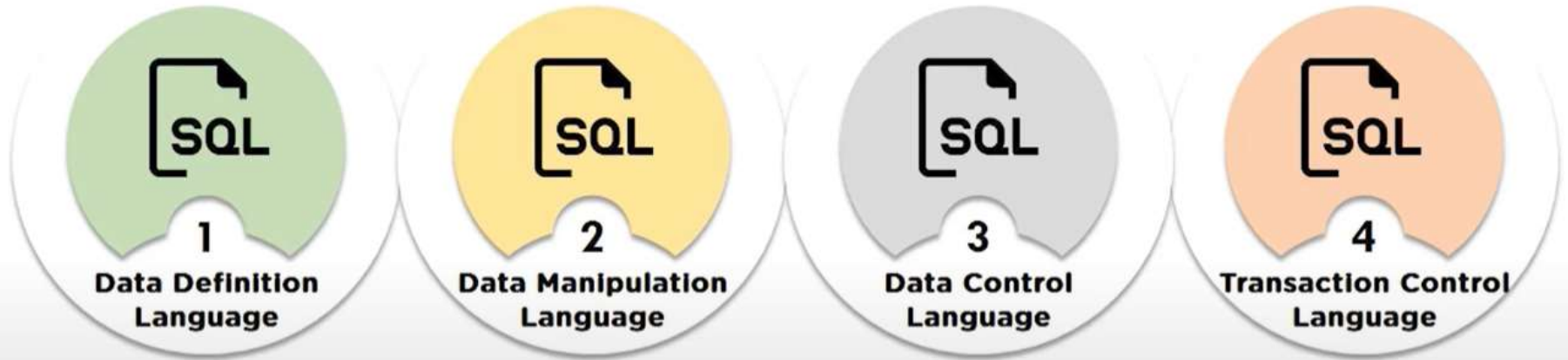
# SQL

## Structured Query Language



# Rappel

## Les types de requêtes en SQL



# Les requêtes en SQL

Data Definition Language(DDL)



CREATE

DROP

TRUNCATE

ALTER



# Les requêtes en SQL

Data Manipulation Language(DML)



INSERT INTO

UPDATE

DELETE

SELECT

# Rappel

Qu'est ce qu'une **Base de donnée** ?

Une Base de donnée est un conteneur qui nous aide à organiser les données.

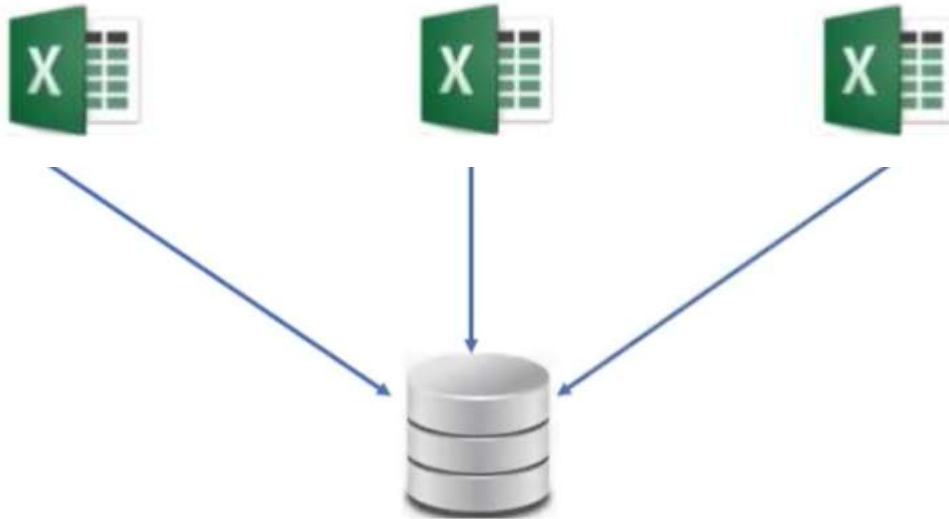
Elle permet de stocker et de retrouver l'intégralité des données dans la BDD.

Cela sera beaucoup plus facile pour interroger les données, les Mettre à jour, supprimer les Données obsolètes etc...



## Rappel

- Si on a plusieurs fichiers Excel, ou toutes les données sont éparpillées sur ces différents fichiers, pourquoi ne pas centraliser toutes ces données dans une même BDD?



# Rappel

Une table est constituée de lignes et colonnes

Colonne

Ligne

	city_id [PK] integer	city character varying (50)	country character varying (50)
1	1	Los Angeles	United States
2	2	New York	United States
3	3	London	United Kingdom

Chaque ligne correspond donc à un **enregistrement de données**

## Création d'une table

```
CREATE TABLE shops (  
    coffeeshop_id INT PRIMARY KEY,  
    coffeeshop_name VARCHAR(50),  
    city_id INT  
);
```

# Insertion de données

Insère-moi la ligne

La/ les valeurs

```
INSERT INTO locations VALUES(2, 'New York', 'United States');  
INSERT INTO locations VALUES(3, 'London', 'United Kingdom');
```

A l'intérieur de

Ma table

Valeurs Souhaitées

# SELECTIONNER LES DONNEES

```
SELECT * FROM employees;  
SELECT * FROM shops;  
SELECT * FROM locations;  
SELECT * FROM suppliers;
```



Toutes les colonnes

# DISTINCT

```
SELECT DISTINCT Pays FROM Clients ;
```



## COUNT

```
SELECT COUNT(*) FROM Products ;
```

## ORDER BY

```
SELECT * FROM Produits  
ORDER BY Prix;
```

```
SELECT * FROM Products  
ORDER BY Price DESC;
```

```
SELECT * FROM Clients  
ORDER BY Pays ASC, NomClient DESC;
```

# Les Opérateurs

## AND

```
SELECT * FROM Clients  
WHERE Pays = 'Maroc'  
AND Ville = 'Tanger';
```

# Les Opérateurs

**OR**

```
SELECT * FROM Clients  
WHERE Pays = 'Maroc' OR Pays = 'USA';
```

# Les Opérateurs

**NOT**

```
SELECT * FROM Clients  
WHERE NOT Pays = 'Maroc' ;
```

# SELECT et WHERE

**WHERE** : Filtrer les données

```
SELECT *  
FROM employees  
WHERE coffeeshop_id = 1 AND gender = 'M';
```

La valeur

Le nom de mes/ma colonnes

Ou sont mes données ?

Ou sont mes données ?

Le nom de mes/ma colonnes

# Mettre à jour / Modifier

```
UPDATE employees SET employee_id = 1;
```

Mets à jour la ligne

La table

Fixe-moi la colonne

Cette nouvelle valeur

# Environnement de travail



PostgreSQL est un **système de gestion de base de données** relationnelle orienté objet puissant et open source qui est capable de prendre en charge en toute sécurité les charges de travail de données les plus complexes.

Prise en charge étendue des types de données  
et des fonctionnalités orientées objet.



# Exercice d'application

Soit la base de données relationnelle des vols quotidiens d'une compagnie aérienne qui contient les tables **Avion**, **Pilote** et **Vol**.

**Table Avion** (**NA** : numéro avion de type entier (clé primaire),  
**Nom** : nom avion de type texte (12),  
**Capacité** : capacité avion de type entier,  
**Localité** : ville de localité de l'avion de type texte (10)  
)

**Table Pilote** (**NP** : numéro pilote de type entier,  
**Nom** : nom du pilote de type texte (25),  
**Adresse** : adresse du pilote de type texte (40)  
)

**Table Vol** (**NV** : numéro de vol de type texte (6),  
**NP** : numéro de pilote de type entier,  
**NA** : numéro avion de type entier,  
**VD** : ville de départ de type texte (10),  
**VA** : ville d'arrivée de type texte (10),  
**HD** : heure de départ de type entier,  
**HA** : heure d'arrivée de type entier )

# Exercice d'application

1) Insérer les avions suivants dans la table Avion :

**(100, AIRBUS, 300, RABAT), (101,B737,250,CASA), (102, B737,220,RABAT)**

# Exercice d'application

1) Insérer les avions suivants dans la table Avion :

**(100, AIRBUS, 300, RABAT), (101,B737,250,CASA), (101, B737,220,RABAT)**

2) Afficher tous les avions

# Exercice d'application

1) Insérer les avions suivants dans la table Avion :

**(100, AIRBUS, 300, RABAT), (101,B737,250,CASA), (102, B737,220,RABAT)**

2) Afficher tous les avions

3) Afficher tous les avions par ordre croissant sur le nom

# Exercice d'application

1) Insérer les avions suivants dans la table Avion :

**(100, AIRBUS, 300, RABAT), (101,B737,250,CASA), (102, B737,220,RABAT)**

2) Afficher tous les avions

3) Afficher tous les avions par ordre croissant sur le nom

4) Afficher les noms et les capacités des avions

# Exercice d'application

1) Insérer les avions suivants dans la table Avion :

**(100, AIRBUS, 300, RABAT), (101,B737,250,CASA), (102, B737,220,RABAT)**

2) Afficher tous les avions

3) Afficher tous les avions par ordre croissant sur le nom

4) Afficher les noms et les capacités des avions

5) Afficher les localités des avions sans redondance

# Exercice d'application

1) Insérer les avions suivants dans la table Avion :

**(100, AIRBUS, 300, RABAT), (101,B737,250,CASA), (102, B737,220,RABAT)**

2) Afficher tous les avions

3) Afficher tous les avions par ordre croissant sur le nom

4) Afficher les noms et les capacités des avions

5) Afficher les localités des avions sans redondance

6) Afficher les avions dont la localité est Rabat ou Casa

# Exercice d'application

1) Insérer les avions suivants dans la table Avion :

**(100, AIRBUS, 300, RABAT), (101,B737,250,CASA), (102, B737,220,RABAT)**

2) Afficher tous les avions

3) Afficher tous les avions par ordre croissant sur le nom

4) Afficher les noms et les capacités des avions

5) Afficher les localités des avions sans redondance

6) Afficher les avions dont la localité est Rabat ou Casa

7) Modifier la capacité de l'avion numéro 101, la nouvelle capacité est 220



# Exercice d'application

1) Insérer les avions suivants dans la table Avion :

**(100, AIRBUS, 300, RABAT), (101,B737,250,CASA), (102, B737,220,RABAT)**

2) Afficher tous les avions

3) Afficher tous les avions par ordre croissant sur le nom

4) Afficher les noms et les capacités des avions

5) Afficher les localités des avions sans redondance

6) Afficher les avions dont la localité est Rabat ou Casa

7) Modifier la capacité de l'avion numéro 101, la nouvelle capacité est 220

8) Supprimer les avions dont la capacité est inférieure à 200

9) Afficher la capacité **maximale**, **minimale**, **moyenne** des avions

## Insérer dans la table VOL

('IT100', 1, 100, 'Casablanca', 'Marrakech', 1200, 1400),  
('IT101', 2, 101, 'Rabat', 'Casablanca', 1330, 1500),  
('IT102', 3, 101, 'Casablanca', 'Rabat', 1400, 1530),  
('IT103', 1, 100, 'Marrakech', 'Casablanca', 1500, 1700),  
('IT104', 2, 101, 'Casablanca', 'Marrakech', 1600, 1800);

## Dans la table Pilote

(1, 'Karim IDRISSE', '12 Rue Atlas Casablanca'),  
(2, 'Fatima Benali', '789 Sahara Avenue Marrakech'),  
(3, 'Ahmed Oujdaoui', '42 Rif Road Rabat')  
(4, 'Amal', 'Tanger');

# WHERE

La commande **WHERE** dans une requête **SQL** permet d'extraire les lignes d'une base de données qui respectent **une condition**.

```
SELECT nom_colonnes  
FROM nom_table  
WHERE condition
```

# WHERE

La commande **WHERE** dans une requête **SQL** permet d'extraire les lignes d'une base de données qui respectent **une condition**.

## Opérateur de Comparaison

Opérateur	Description
=	Égale
<>	Pas égale
!=	Pas égale
>	Supérieur à
<	Inférieur à
>=	Supérieur ou égale à
<=	Inférieur ou égale à
IN	Liste de plusieurs valeurs possibles
BETWEEN	Valeur comprise dans un intervalle donnée (utile pour les nombres ou dates)
LIKE	Recherche en spécifiant le début, milieu ou fin d'un mot.
IS NULL	Valeur est nulle
IS NOT NULL	Valeur n'est pas nulle

# IS NULL

id	nom	date_inscription	fk_adresse_livraison_id	fk_adresse_facturation_id
23	Grégoire	2013-02-12	12	12
24	Sarah	2013-02-17	NULL	NULL
25	Anne	2013-02-21	13	14
26	Frédérique	2013-03-02	NULL	NULL

```
SELECT *  
FROM `utilisateur`  
WHERE `fk_adresse_livraison_id` IS NULL
```

id	nom	date_inscription	fk_adresse_livraison_id	fk_adresse_facturation_id
24	Sarah	2013-02-17	NULL	NULL
26	Frédérique	2013-03-02	NULL	NULL

# IS NOT NULL

id	nom	date_inscription	fk_adresse_livraison_id	fk_adresse_facturation_id
23	Grégoire	2013-02-12	12	12
24	Sarah	2013-02-17	NULL	NULL
25	Anne	2013-02-21	13	14
26	Frédérique	2013-03-02	NULL	NULL

```
SELECT *  
FROM `utilisateur`  
WHERE `fk_adresse_livraison_id` IS NOT NULL
```

id	nom	date_inscription	fk_adresse_livraison_id	fk_adresse_facturation_id
23	Grégoire	2013-02-12	12	12
25	Anne	2013-02-21	13	14

## L'opérateur LIKE

```
SELECT * FROM employes  
WHERE first_name LIKE 'A%'
```

```
SELECT * FROM employes  
WHERE first_name LIKE '%A%'
```

```
SELECT * FROM employes  
WHERE first_name LIKE '%en'
```

# L'opérateur LIKE

```
SELECT * FROM employes  
WHERE first_name LIKE '_e_n__'
```

```
SELECT * FROM employes  
WHERE first_name LIKE 'A%'
```

L'opérateur LIKE est utilisé dans une clause **WHERE** pour rechercher un motif spécifique dans une colonne.

Deux caractères génériques utilisés avec l'opérateur **LIKE** :

**%** Le signe de pourcentage représente zéro, un ou plusieurs caractères.

**\_** Le signe de soulignement représente un seul caractère.



# L'opérateur BETWEEN

```
SELECT * FROM Products  
WHERE Price BETWEEN 10 AND 15;
```

```
SELECT * FROM commande  
WHERE date_commande BETWEEN '2023-04-12' AND '2023-05-05';
```

- L'opérateur BETWEEN sélectionne des valeurs dans un intervalle.
- Les valeurs peuvent être des nombres, du texte ou des dates.
- Les valeurs de début et de fin sont incluses.

## AS (Alias)

```
SELECT commande_id AS id  
FROM commandes;
```

```
SELECT product_name AS "Mes Produits"  
FROM products;
```

# LIMIT

La clause LIMIT est à utiliser pour spécifier le nombre maximum de résultats que l'on souhaite obtenir.

```
SELECT *  
FROM table  
LIMIT 10
```

Cette requête permet de récupérer seulement les 10 premiers résultats d'une table

# LIMIT OFFSET

Permet d'effectuer un **décalage** sur le jeu de résultat.

```
SELECT *  
FROM table  
LIMIT 10 OFFSET 5
```

Cette requête permet de récupérer les résultats 6 à 15  
(car l'OFFSET commence toujours à 0).

Afficher tous les vols dont le numéro de vol se termine par "01"

## Exercice d'application (Suite)

- Afficher les deux premiers enregistrements de la table VOL.
- Afficher le nom et l'adresse de tous les pilotes.
- Afficher les vols dont l'heure de départ est après 14h.
- Afficher les pilotes dont le nom commence par "A".
- Afficher les avions basés à Rabat, avec la localité renommée en 'Emplacement'.
- Afficher tous les vols triés par la ville de départ.
- Afficher tous les vols dont le numéro de vol se termine par "01"

# GROUP BY

La commande **GROUP BY** est utilisée pour grouper plusieurs résultats et utiliser une fonction sur un groupe de résultat.

```
SELECT colonne1, fonction(colonne2)
FROM table
GROUP BY colonne1
```

id	client	tarif	date
1	Pierre	102	2012-10-23
2	Simon	47	2012-10-27
3	Marie	18	2012-11-05
4	Marie	20	2012-11-14
5	Pierre	160	2012-12-03

```
SELECT client, SUM(tarif)
FROM achat
GROUP BY client
```

client	SUM(tarif)
Pierre	262
Simon	47
Marie	38

# GROUP BY

**EMPLOYEES**

DEPARTMENT_ID	SALARY
10	4400
20	13000
20	6000
50	5800
50	3500
50	3100
50	2500
50	2600
60	9000
60	6000
60	4200
80	10500
80	8600
80	11000
90	24000
90	17000

4400

9500

3500

6400

10033

**Average salary  
in EMPLOYEES  
table for each  
department**

DEPARTMENT_ID	AVG(SALARY)
10	4400
20	9500
50	3500
60	6400
80	10033.3333
90	19333.3333
110	10150
	7000

## GROUP BY

```
SELECT department_id , AVG(salary)
FROM employees
GROUP BY department_id;
```

DEPARTMENT_ID	AVG(SALARY)
	7000
90	19333.3333
20	9500
110	10150
50	3500
80	10033.3333
60	6400
10	4400



# HAVING

Similaire à WHERE à la seule différence que **HAVING** permet de filtrer en utilisant des fonctions telles que **SUM()**, **COUNT()**, **AVG()**, **MIN()** ou **MAX()**.

id	client	tarif	date_achat
1	Pierre	102	2012-10-23
2	Simon	47	2012-10-27
3	Marie	18	2012-11-05
4	Marie	20	2012-11-14
5	Pierre	160	2012-12-03

```
SELECT client, SUM(tarif)
FROM achat
GROUP BY client
HAVING SUM(tarif) > 40
```

## Exercice d'application (Suite)

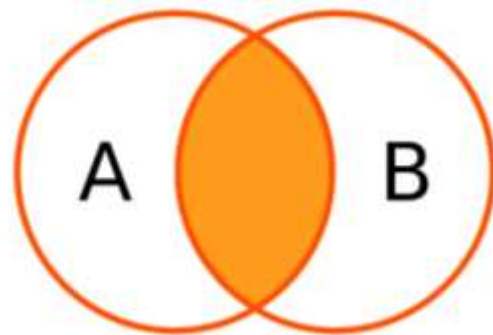
- Nombre de vols par ville de départ
- Nombre de vols par pilote

## Exercice d'application (Suite)

- Nombre de vols par ville de départ
- Nombre de vols par pilote
- Somme de la durée totale des vols par pilote

# JOINTURES

## INNER JOIN



```
SELECT *  
FROM table_1  
INNER JOIN table_2 ON table_1.une_colonne = table_2.autre_colonne;
```

Afficher les enregistrements des tables **table\_1** et **table\_2**

Lorsque les données de la colonne **une\_colonne** de la table **table\_1** est égal aux données de la colonne **autre\_colonne** de la table **table\_2**.

# INNER JOIN

id	prenom	nom	email
1	Marine	Leroy	mleeroy@example.com
2	Jean	René	jrene@example.com
3	Ted	Bundy	tbundy@example.com
4	Paul	Bismuth	pbismuth@example.com
5	Caroline	Rodriguez	crodriguez@example.com

Table : clients

id_commande	id_client	date_ajout	transporteur
1	1	2019-04-01	Mondial Relay
2	2	2019-04-02	Colissimo
3	2	2019-04-05	Colissimo
4	5	2019-04-08	Colissimo
5	9	2019-04-10	Colissimo

Table : commandes

```
SELECT id_client, email, id_commande, date_ajout, transporteur
FROM clients
INNER JOIN commandes ON clients.id = commandes.id_client;
```

# INNER JOIN

```
SELECT id_client, email, id_commande, date_ajout, transporteur  
FROM clients  
INNER JOIN commandes ON clients.id = commandes.id_client;
```

id_client	email	id_commande	date_ajout	transporteur
1	mleeroy@example.com	1	2019-04-01	Mondial Relay
2	jrene@example.com	2	2019-04-02	Colissimo
2	jrene@example.com	3	2019-04-05	Colissimo
5	crodriguez@example.com	4	2019-04-08	Colissimo

## Exercice d'application

10) Afficher le nom et l'adresse des pilotes assurant les vols IT100 et IT104

## Exercice d'application

- 10) Afficher le nom et l'adresse des pilotes assurant les vols IT100 et IT104
- 11) Afficher le nombre total de vols effectués par chaque pilote avec le nom du pilote



# Corrigé

```
SELECT Nom, Adresse  
FROM Pilote, Vol  
WHERE Pilote.NP = Vol.NP  
AND (NV = 'IT100' OR NV = 'IT104');
```

```
SELECT Nom, Adresse  
FROM Pilote  
INNER JOIN Vol ON Pilote.NP = Vol.NP  
WHERE NV = 'IT100' OR NV = 'IT104';
```

## Exercice d'application

- 10) Afficher le nom et l'adresse des pilotes assurant les vols IT100 et IT104
- 11) Afficher le nombre total de vols effectués par chaque pilote avec le nom du pilote
- 12) Afficher les noms des pilotes qui conduisent un AIRBUS

## Corrigé

```
SELECT Pilote.nom ,COUNT(VOL.NV) AS NombreDeVols  
FROM Pilote, Vol  
WHERE Pilote.NP = VOL.NP  
GROUP BY Pilote.Nom;
```

```
SELECT Pilote.nom, COUNT(VOL.NV) AS NombreDeVols  
FROM Pilote  
INNER JOIN VOL ON Pilote.NP = VOL.NP  
GROUP BY Pilote.NP;
```

## Corrigé

```
SELECT DISTINCT Pilote.nom  
FROM Pilote  
JOIN VOL ON Pilote.NP = VOL.NP  
JOIN Avion ON VOL.NA = Avion.NA  
WHERE Avion.nom = 'AIRBUS';
```

```
SELECT DISTINCT Pilote.nom  
FROM Pilote,VOL,Avion  
WHERE Pilote.NP = VOL.NP AND VOL.NA = Avion.NA  
AND Avion.nom = 'AIRBUS';
```

# SQL ALTER TABLE

Ajouter une colonne

```
ALTER TABLE nom_table  
ADD COLUMN nom_column type_donnees
```

```
ALTER TABLE utilisateur  
ADD COLUMN adresse VARCHAR(255)
```



Nom\_Colonne



Type

# SQL ALTER TABLE

Ajouter une **Clé Etrangère**

```
ALTER TABLE table_enfant  
ADD FOREIGN KEY (colonne_enfant) REFERENCES table_parent(colonne_parent);
```

# SQL ALTER TABLE

Ajouter une **clé Primaire**

```
ALTER TABLE nom_de_la_table  
ADD PRIMARY KEY (colonne);
```

```
ALTER TABLE nom_de_la_table  
ADD PRIMARY KEY (colonne1, colonne2);
```

# SQL ALTER TABLE

Modifier une colonne

**MYSQL**

```
ALTER TABLE nom_table  
MODIFY nom_colonne type_donnees
```

**POSTGRESQL**

```
ALTER TABLE nom_table  
ALTER COLUMN nom_colonne TYPE nouveau_type;
```



# RENOMMER UNE COLONNE

## MYSQL

```
ALTER TABLE nom_table  
CHANGE colonne_ancien_nom colonne_nouveau_nom type_donnees
```

## POSTGRESQL

```
ALTER TABLE nom_table  
RENAME COLUMN colonne_ancien_nom TO colonne_nouveau_nom
```

# SQL ALTER TABLE

Supprimer une colonne

```
ALTER TABLE nom_table  
DROP COLUMN nom_colonne
```

# Gestion des cafés

Table Employes

employee_id (PK)	first_name	last_name	e-mail	hire_date	gender	salary	coffeeshop_id
501599	Carson	Mosconi	cmosconi0@census.gov	29/08/2015	M	32973	1
144108	Khalil	Corr	kcorr@github.io	23/04/2014	M	52802	1

# Gestion des cafés

## Fournisseurs

coffeeshop_id (PK)	supplier_name (PK)	coffee_type
1	Beans and Barley	Arabica
1	Cool Beans	Robusta

## Locations

city_id (PK)	city	country
1	Los Angeles	United States
2	New York	United States
3	London	United Kingdom

## shops

coffeeshop_id (PK)	coffeeshop_name	city_id (FK)
1	Common Grounds	1
2	Early Rise	2
3	Ancient Bean	3
4	Urban Grind	1
5	Trembling Cup	2

# Gestion des cafés

## Fournisseurs

coffeeshop_id (PK)	supplier_name (PK)	coffee_type
1	Beans and Barley	Arabica
1	Cool Beans	Robusta
2	Vanilla Bean	Liberica
2	Beans and Barley	Arabica
2	Cool Beans	Robusta
3	Bean Me Up	Excelsa
3	Vanilla Bean	Liberica
3	Cool Beans	Robusta
3	Beans and Barley	Arabica
4	Vanilla Bean	Liberica
4	Bean Me Up	Excelsa
5	Beans and Barley	Arabica

## Locations

city_id (PK)	city	country
1	Los Angeles	United States
2	New York	United States
3	London	United Kingdom

### shops

coffeeshop_id (PK)	coffeeshop_name	city_id (FK)
1	Common Grounds	1
2	Early Rise	2
3	Ancient Bean	3
4	Urban Grind	1
5	Trembling Cup	2

# Questions

## Création des tables:

- Créer la table Employés
- Créer la table Shops
- Ajouter la clé étrangère à la table Employés
- Créer la table locations
- Ajouter la clé étrangère à la table shops
- Créer la table fournisseurs

## Clé Etrangère de la table shops

```
-- Ajouter la clé étrangère à la table shops
ALTER TABLE shops
ADD FOREIGN KEY (city_id)
REFERENCES locations(city_id)
ON DELETE SET NULL;
```

# CREATION DE LA TABLE FOURNISSEURS

```
-- Create suppliers table
CREATE TABLE fournisseurs (
  coffeeshop_id INT,
  supplier_name VARCHAR(40),
  coffee_type VARCHAR(20),
  PRIMARY KEY (coffeeshop_id, supplier_name),
  FOREIGN KEY (coffeeshop_id) REFERENCES shops(coffeeshop_id)
  ON DELETE CASCADE
);
```



# CREATION DE LA TABLE EMPLOYES

```
CREATE TABLE employes (  
    employee_id INT PRIMARY KEY,  
    first_name VARCHAR(50),  
    last_name VARCHAR(50),  
    email VARCHAR(50),  
    hire_date DATE,  
    gender VARCHAR(1), -- "M"/"F" (male/female)  
    salary INT,  
    coffeeshop_id INT  
);
```

# CREATION DE LA TABLE SHOPS

```
CREATE TABLE shops (  
    coffeeshop_id INT PRIMARY KEY,  
    coffeeshop_name VARCHAR(50),  
    city_id INT  
);
```

## AJOUTER LA CLE ETRANGERE A LA TABLE EMPLOYES

```
ALTER TABLE employes  
ADD FOREIGN KEY (coffeeshop_id)  
REFERENCES shops(coffeeshop_id)  
ON DELETE SET NULL;
```

## CREATION DE LA TABLE Locations

```
-- Creation de la table locations
CREATE TABLE locations (
    city_id INT PRIMARY KEY,
    city VARCHAR(50),
    country VARCHAR(50)
);
```

# Questions

## Insertion des données

- Insérer les 2 premiers enregistrements de la table employés
- Insérer le premier enregistrement de la table shops
- Mettre à jour les 2 premiers enregistrements de la table employés
- Insérer la première ligne de la table locations
- Modifier la table shops
- Insérer les deux premiers enregistrements de la table fournisseurs
- Insérer le reste des enregistrements

# Manipulation

Insérer les 2 premiers enregistrements de la table employees

employee_id (PK)	first_name	last_name	email	hire_date	gender	salary	coffeeshop_id (FK)
501559	Carson	Mosconi	cmosconi0@census.gov	29/08/2015	M	32973	1
144108	Khalil	Corr	kcorr1@github.io	23/04/2014	M	52802	1

```
INSERT INTO employees VALUES (501559, 'Carson', 'Mosconi', 'cmosconi0@census.gov', '2015/08/29', 'M', 32973, NULL);
INSERT INTO employees VALUES (144108, 'Khalil', 'Corr', 'kcorr1@github.io', '2014/04/23', 'M', 52802, NULL);
```

## Insérer le 1<sup>er</sup> enregistrement de la table shops

shops		
coffeeshop _id (PK)	coffeeshop_name	city_id (FK)
1	Common Grounds	1
2	Early Rise	2
3	Ancient Bean	3
4	Urban Grind	1
5	Trembling Cup	2

## Insérer le 1<sup>er</sup> enregistrement de la table shops

shops		
coffeeshop _id (PK)	coffeeshop_name	city_id (FK)
1	Common Grounds	1
2	Early Rise	2
3	Ancient Bean	3
4	Urban Grind	1
5	Trembling Cup	2

```
INSERT INTO shops VALUES(1, 'Common Grounds', NULL);
```



## On met à jours les 2 premiers enregistrements

```
UPDATE employes  
SET coffeeshop_id = 1
```

## Insertion de la première ligne de la table locations

locations		
city_id (PK)	city	country
1	Los Angeles	United States
2	New York	United States
3	London	United Kingdom

```
INSERT INTO locations VALUES(1, 'Los Angeles', 'United States');
```

## Modifier la table shops

```
UPDATE shops  
SET city_id = 1  
WHERE coffeeshop_id = 1;
```

## Insertion des deux 1<sup>er</sup> enregistrements de la table fournisseurs

coffeeshop _id (PK)	supplier_name (PK)	coffee_type
1	Beans and Barley	Arabica
1	Cool Beans	Robusta
2	Vanilla Bean	Liberica
2	Beans and Barley	Arabica
2	Cool Beans	Robusta
3	Bean Me Up	Excelsa

```
INSERT INTO suppliers VALUES(1, 'Beans and Barley', 'Arabica');  
INSERT INTO suppliers VALUES(1, 'Cool Beans', 'Robusta');
```

# Gestion des cafés

Table Employes

employee_id (PK)	first_name	last_name	e-mail	hire_date	gender	salary	coffeeshop_id
501599	Carson	Mosconi	cmosconi0@census.gov	29/08/2015	M	32973	1
144108	Khalil	Corr	kcorr@github.io	23/04/2014	M	52802	1
782284	William	Rayman	Rayman@nasa.com	17/08/2015	M	48048	2
225709	Carol	Tarpey	Tarp@harvard.edu	22/12/2021	F	15235	3
614903	Melissa	Lili	LiliM@mynews.com	14/09/2016	F	66566	3
590293	Mary	Ellen	Ellen@gmail.com	27/01/2020	F	41159	4
243999	Jeremy	Smith		03/07/2014	M	23772	5
599230	Simon	Joe	Joe@dev.com	17/08/2015	M	15083	5

# Gestion des cafés

Table Employes

employee_id (PK)	first_name	last_name	e-mail	hire_date	gender	salary	coffeeshop_id
501599	Carson	Mosconi	cmosconi0@census.gov	29/08/2015	M	32973	1
144108	Khalil	Corr	kcorr@github.io	23/04/2014	M	52802	1
782284	William	Rayman	Rayman@nasa.com	17/08/2015	M	48048	2
225709	Carol	Tarpey	Tarp@harvard.edu	22/12/2021	F	15235	3
614903	Melissa	Lili	LiliM@mynews.com	14/09/2016	F	66566	3
590293	Mary	Ellen	Ellen@gmail.com	27/01/2020	F	41159	4
243999	Jeremy	Smith		03/07/2014	M	23772	5
599230	Simon	Joe	Joe@dev.com	17/08/2015	M	15083	5

# Insérer le reste des enregistrements

```
-- Insert into la table locations
INSERT INTO locations VALUES(2, 'New York', 'United States');
INSERT INTO locations VALUES(3, 'London', 'United Kingdom');

-- Insert into la table shops
INSERT INTO shops VALUES(2, 'Early Rise', 2);
INSERT INTO shops VALUES(3, 'Ancient Bean', 3);
INSERT INTO shops VALUES(4, 'Urban Grind', 1);
INSERT INTO shops VALUES(5, 'Trembling Cup', 2);

-- Insert into la table fournisseurs
INSERT INTO fournisseurs VALUES(2, 'Vanilla Bean', 'Liberica');
INSERT INTO fournisseurs VALUES(2, 'Beans and Barley', 'Arabica');
INSERT INTO fournisseurs VALUES(2, 'Cool Beans', 'Robusta');
INSERT INTO fournisseurs VALUES(3, 'Bean Me Up', 'Excelsa');
INSERT INTO fournisseurs VALUES(3, 'Vanilla Bean', 'Liberica');
INSERT INTO fournisseurs VALUES(3, 'Cool Beans', 'Robusta');
INSERT INTO fournisseurs VALUES(3, 'Beans and Barley', 'Arabica');
INSERT INTO fournisseurs VALUES(4, 'Vanilla Bean', 'Liberica');
```

# Manipulation

**De la table fournisseurs Sélectionner :**

- les lignes où le fournisseur est Beans and Barley
- les lignes où le type de café n'est ni 'Robusta' ni 'Arabica'
- Nombre de fournisseurs par type de café (coffee\_type)



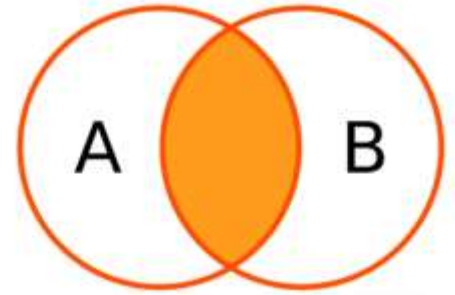
# Manipulations

- Sélectionner toutes les colonnes de chaque table

## **De la table employees Sélectionner :**

- Les 3 premières colonnes de la table employees
- Les employés qui travaillent pour le coffeeshop Common Grounds
- Les employés qui ont un salaire plus que 50 000
- Les employés qui travaillent pour le coffeeshop Common Grounds ET qui ont un salaire plus que 50 000

# Rappel



La jointure interne retourne uniquement les lignes qui ont au moins une correspondance dans les deux tables.

C'est l'une des jointures les plus communes.

**INNER JOIN == JOIN**

# INNER JOIN

id	prenom	nom	email
1	Marine	Leroy	mleeroy@example.com
2	Jean	René	jréné@example.com
3	Ted	Bundy	tbundy@example.com
4	Paul	Bismuth	pbismuth@example.com
5	Caroline	Rodriguez	crodriguez@example.com

Table : clients

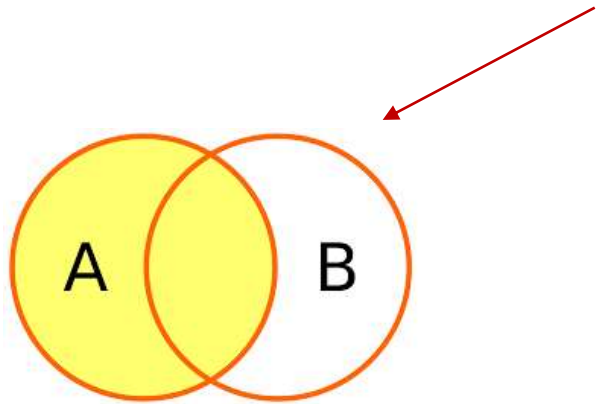
id_commande	id_client	date_ajout	transporteur
1	1	2019-04-01	Mondial Relay
2	2	2019-04-02	Colissimo
3	2	2019-04-05	Colissimo
4	5	2019-04-08	Colissimo
5	9	2019-04-10	Colissimo

Table : commandes

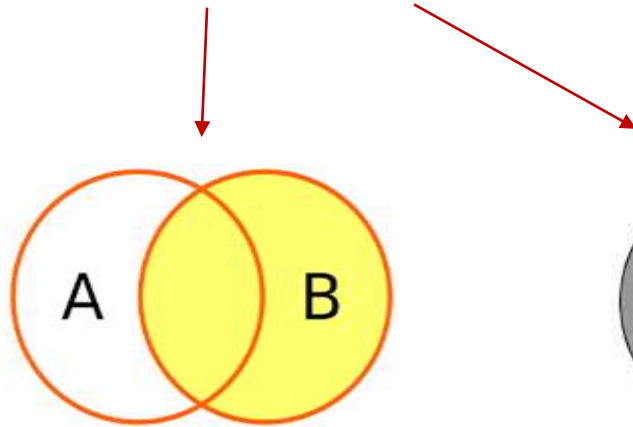
```
SELECT id_client, email, id_commande, date_ajout, transporteur
FROM clients
INNER JOIN commandes ON clients.id = commandes.id_client;
```

Les informations des clients **qui ont passé** des commandes, et les détails de ces commandes.

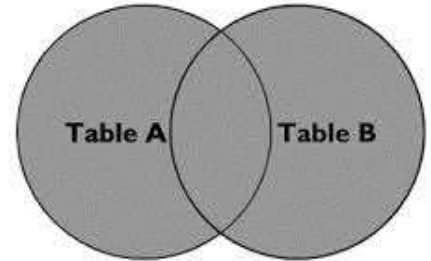
## Jointures externes



**LEFT JOIN**

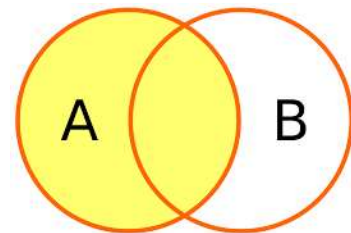


**RIGHT JOIN**



**FULL JOIN**

# LEFT JOIN (Jointure Externe Gauche)



Cette requête est particulièrement intéressante pour récupérer les informations de tableA tout en récupérant les données associées, même s'il n'y a pas de correspondance avec tableB.

A savoir, s'il n'y a pas de correspondance les colonnes de table2 vaudront toutes NULL.

```
SELECT id, prenom, nom, email , id_commande  
FROM clients  
LEFT JOIN commandes ON clients.id = commandes.id_client;
```

```
SELECT id, prenom, nom, email , id_commande  
FROM clients  
LEFT OUTER JOIN commandes ON clients.id = commandes.id_client;
```

# LEFT JOIN

id	prenom	nom	email
1	Marine	Leroy	mleroy@example.com
2	Jean	René	jrene@example.com
3	Ted	Bundy	tbundy@example.com
4	Paul	Bismuth	pbismuth@example.com
5	Caroline	Rodriguez	crodriguez@example.com

Table : clients

id_commande	id_client	date_ajout	transporteur
1	1	2019-04-01	Mondial Relay
2	2	2019-04-02	Colissimo
3	2	2019-04-05	Colissimo
4	5	2019-04-08	Colissimo
5	9	2019-04-10	Colissimo

Table : commandes

```
SELECT id, prenom, nom, email , id_commande
FROM clients
LEFT JOIN commandes ON clients.id = commandes.id_client;
```

Toutes les lignes de la table des clients, et les détails des commandes correspondantes **si elles existent**

# LEFT JOIN

id	prenom	nom	email
1	Marine	Leroy	mleeroy@example.com
2	Jean	René	jrene@example.com
3	Ted	Bundy	tbundy@example.com
4	Paul	Bismuth	pbismuth@example.com
5	Caroline	Rodriguez	crodriguez@example.com

Table : clients

id_commande	id_client	date_ajout	transporteur
1	1	2019-04-01	Mondial Relay
2	2	2019-04-02	Colissimo
3	2	2019-04-05	Colissimo
4	5	2019-04-08	Colissimo
5	9	2019-04-10	Colissimo

Table : commandes

id	prenom	nom	email	id_commande
1	Marine	Leroy	mleeroy@example.com	1
2	Jean	René	jrene@example.com	2
2	Jean	René	jrene@example.com	3
3	Ted	Bundy	tbundy@example.com	NULL
4	Paul	Bismuth	pbismuth@example.com	NULL
5	Caroline	Rodriguez	crodriguez@example.com	4

# LEFT JOIN

id	prenom	nom	email	ville
1	Aimée	Marechal	aime.marechal@example.com	Paris
2	Esmée	Lefort	esmee.lefort@example.com	Lyon
3	Marine	Prevost	m.prevost@example.com	Lille
4	Luc	Rolland	lucrolland@example.com	Marseille

**Table Utilisateurs**

**Table Commandes**

utilisateur_id	date_achat	num_facture	prix_total
1	2013-01-23	A00103	203.14
1	2013-02-14	A00104	124.00
2	2013-02-17	A00105	149.45
2	2013-02-21	A00106	235.35
5	2013-03-02	A00107	47.58

```
SELECT *  
FROM Utilisateurs  
LEFT JOIN commandes ON utilisateurs.id = commandes.utilisateur_id;
```

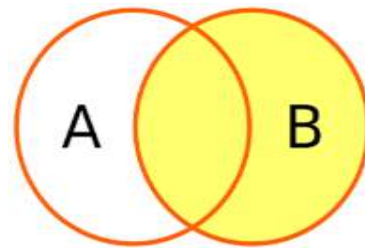


# LEFT JOIN

```
SELECT *  
FROM Utilisateurs  
LEFT JOIN commandes ON utilisateurs.id = commandes.utilisateur_id;
```

id	prenom	nom	date_achat	num_facture	prix_total
1	Aimée	Marechal	2013-01-23	A00103	203.14
1	Aimée	Marechal	2013-02-14	A00104	124.00
2	Esmée	Lefort	2013-02-17	A00105	149.45
2	Esmée	Lefort	2013-02-21	A00106	235.35
3	Marine	Prevost	NULL	NULL	NULL
4	Luc	Rolland	NULL	NULL	NULL

# RIGHT JOIN



Ce type de jointure permet de retourner tous les enregistrements de la table B de droite (right) même s'il n'y a pas de correspondance avec la table de gauche.

```
SELECT id, prenom, nom, email , id_commande  
FROM clients  
RIGHT JOIN commandes ON clients.id = commandes.id_client;
```

```
SELECT id, prenom, nom, email , id_commande  
FROM clients  
RIGHT OUTER JOIN commandes ON clients.id = commandes.id_client;
```

# RIGHT JOIN

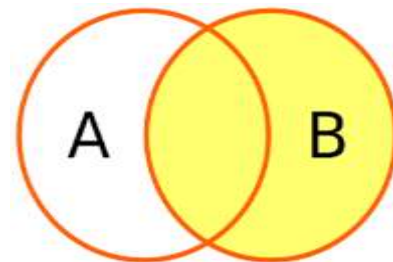
id	prenom	nom	email
1	Marine	Leroy	mleroy@example.com
2	Jean	René	jrene@example.com
3	Ted	Bundy	tbundy@example.com
4	Paul	Bismuth	pbismuth@example.com
5	Caroline	Rodriguez	crodriguez@example.com

Table : clients

id_commande	id_client	date_ajout	transporteur
1	1	2019-04-01	Mondial Relay
2	2	2019-04-02	Colissimo
3	2	2019-04-05	Colissimo
4	5	2019-04-08	Colissimo
5	9	2019-04-10	Colissimo

Table : commandes

```
SELECT id, prenom, nom, email , id_commande
FROM clients
RIGHT JOIN commandes ON clients.id = commandes.id_client;
```



Toutes les commandes avec les détails des clients correspondants, même si certains clients n'ont pas passé de commandes

# RIGHT JOIN

id	prenom	nom	email
1	Marine	Leroy	mleroy@example.com
2	Jean	René	jrene@example.com
3	Ted	Bundy	tbundy@example.com
4	Paul	Bismuth	pbismuth@example.com
5	Caroline	Rodriguez	crodriguez@example.com

Table : clients

id_commande	id_client	date_ajout	transporteur
1	1	2019-04-01	Mondial Relay
2	2	2019-04-02	Colissimo
3	2	2019-04-05	Colissimo
4	5	2019-04-08	Colissimo
5	9	2019-04-10	Colissimo

Table : commandes

id	prenom	nom	email	id_commande
1	Marine	Leroy	mleroy@example.com	1
2	Jean	René	jrene@example.com	2
2	Jean	René	jrene@example.com	3
5	Caroline	Rodriguez	crodriguez@example.com	4
NULL	NULL	NULL	NULL	5

# RIGHT JOIN

id	prenom	nom	email	ville
1	Aimée	Marechal	aime.marechal@example.com	Paris
2	Esmée	Lefort	esmee.lefort@example.com	Lyon
3	Marine	Prevost	m.prevost@example.com	Lille
4	Luc	Rolland	lucrolland@example.com	Marseille

**Table Utilisateurs**

**Table Commandes**

utilisateur_id	date_achat	num_facture	prix_total
1	2013-01-23	A00103	203.14
1	2013-02-14	A00104	124.00
2	2013-02-17	A00105	149.45
3	2013-02-21	A00106	235.35
5	2013-03-02	A00107	47.58

```
SELECT *  
FROM Utilisateurs  
RIGHT JOIN commandes ON utilisateurs.id = commandes.utilisateur_id;
```

# RIGHT JOIN

id	prenom	nom	email	ville
1	Aimée	Marechal	aime.marechal@example.com	Paris
2	Esmée	Lefort	esmee.lefort@example.com	Lyon
3	Marine	Prevost	m.prevost@example.com	Lille
4	Luc	Rolland	lucrolland@example.com	Marseille

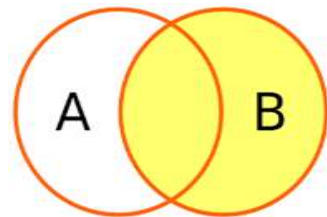
Table Utilisateurs

Table Commandes

utilisateur_id	date_achat	num_facture	prix_total
1	2013-01-23	A00103	203.14
1	2013-02-14	A00104	124.00
2	2013-02-17	A00105	149.45
3	2013-02-21	A00106	235.35
5	2013-03-02	A00107	47.58

id	prenom	nom	utilisateur_id	date_achat	num_facture
1	Aimée	Marechal	1	2013-01-23	A00103
1	Aimée	Marechal	1	2013-02-14	A00104
2	Esmée	Lefort	2	2013-02-17	A00105
3	Marine	Prevost	3	2013-02-21	A00106
NULL	NULL	NULL	5	2013-03-02	A00107

# RIGHT JOIN



id	prenom	nom	email
1	Marine	Leroy	mleeroy@example.com
2	Jean	René	jrene@example.com
3	Ted	Bundy	tbundy@example.com
4	Paul	Bismuth	pbismuth@example.com
5	Caroline	Rodriguez	crodriguez@example.com

Table : clients

id_commande	id_client	date_ajout	transporteur
1	1	2019-04-01	Mondial Relay
2	2	2019-04-02	Colissimo
3	2	2019-04-05	Colissimo
4	5	2019-04-08	Colissimo
5	9	2019-04-10	Colissimo

Table : commandes

```
SELECT id, prenom, nom, email , id_commande
FROM commandes
LEFT JOIN clients
ON clients.id = commandes.id_client;
```

```
SELECT id, prenom, nom, email , id_commande
FROM clients
RIGHT JOIN commandes
ON clients.id = commandes.id_client;
```

# FULL JOIN

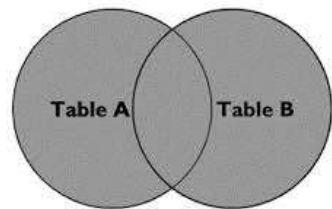
id	prenom	nom	email
1	Marine	Leroy	mleroy@example.com
2	Jean	René	jrene@example.com
3	Ted	Bundy	tbundy@example.com
4	Paul	Bismuth	pbismuth@example.com
5	Caroline	Rodriguez	crodriguez@example.com

Table : clients

id_commande	id_client	date_ajout	transporteur
1	1	2019-04-01	Mondial Relay
2	2	2019-04-02	Colissimo
3	2	2019-04-05	Colissimo
4	5	2019-04-08	Colissimo
5	9	2019-04-10	Colissimo

Table : commandes

```
SELECT id, prenom, nom, email , id_commande
FROM clients
FULL JOIN commandes ON clients.id = commandes.id_client;
```



Liste des clients et commandes, y compris les clients sans commandes et les commandes sans clients.



# FULL JOIN

id	prenom	nom	email
1	Marine	Leroy	mleroy@example.com
2	Jean	René	jrene@example.com
3	Ted	Bundy	tbundy@example.com
4	Paul	Bismuth	pbismuth@example.com
5	Caroline	Rodriguez	crodriguez@example.com

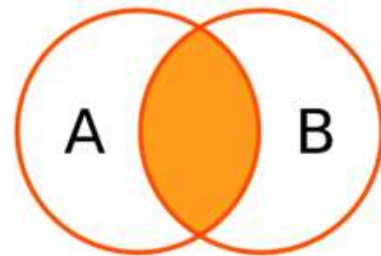
Table : clients

id_commande	id_client	date_ajout	transporteur
1	1	2019-04-01	Mondial Relay
2	2	2019-04-02	Colissimo
3	2	2019-04-05	Colissimo
4	5	2019-04-08	Colissimo
5	9	2019-04-10	Colissimo

Table : commandes

id	prenom	nom	email	id_commande	id_client
1	Marine	Leroy	mleroy@example.com	1	1
2	Jean	René	jrene@example.com	2	2
2	Jean	René	jrene@example.com	3	2
3	Ted	Bundy	tbundy@example.com	NULL	NULL
4	Paul	Bismuth	pbismuth@example.com	NULL	NULL
5	Caroline	Rodriguez	crodriguez@example.com	4	5
NULL	NULL	NULL	NULL	5	9

# NATURAL JOIN



Le '**NATURAL JOIN**' est une opération de jointure qui compare automatiquement toutes les colonnes portant le même nom dans les tables jointes

Le "**NATURAL JOIN**" automatise la jointure en utilisant toutes les colonnes portant le même nom dans les tables jointes sans utiliser le ON

**NATURAL JOIN** est un type de **INNER JOIN**

# NATURAL JOIN

user_id	user_prenom	user_ville	pays_id
1	Jérémie	Paris	1
2	Damien	Lyon	2
3	Sophie	Marseille	NULL
4	Yann	Lille	9999
5	Léa	Paris	1

Table Utilisateurs

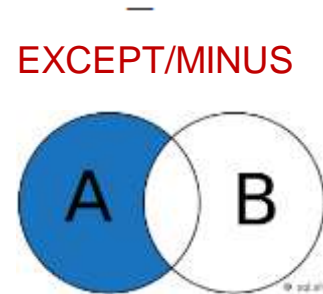
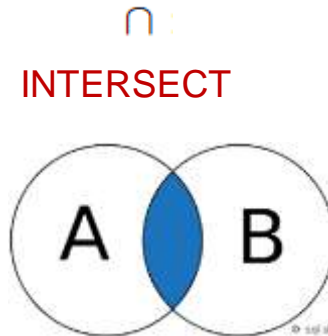
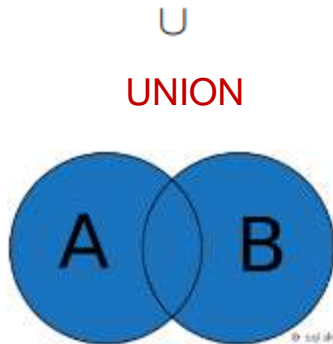
pays_id	pays_nom
1	France
2	Canada
3	Belgique
4	Suisse

Table Pays

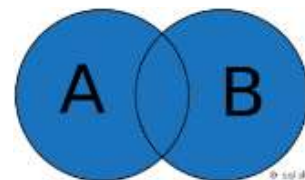
```
SELECT *  
FROM Utilisateurs  
NATURAL JOIN Pays;
```

# Opérateurs Ensemblistes

- Permettent de combiner les résultats de plusieurs select
- Pas de doublons (distinct implicite)
- Les select doivent contenir le même nombre d'attributs



# UNION

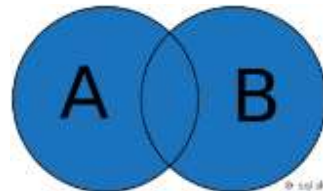


prenom	nom	ville	date_naissance	total_achat
Léon	Dupuis	Paris	1983-03-06	135
Marie	Bernard	Paris	1993-07-03	75
Sophie	Dupond	Marseille	1986-02-22	27
Marcel	Martin	Paris	1976-11-24	39

prenom	nom	ville	date_naissance	total_achat
Marion	Leroy	Lyon	1982-10-27	285
Paul	Moreau	Lyon	1976-04-19	133
Marie	Bernard	Paris	1993-07-03	75
Marcel	Martin	Paris	1976-11-24	39

```
SELECT * FROM magasin1_client
UNION
SELECT * FROM magasin2_client
```

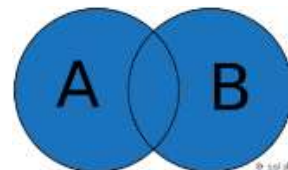
# UNION



```
SELECT * FROM magasin1_client  
UNION  
SELECT * FROM magasin2_client
```

prenom	nom	ville	date_naissance	total_achat
Léon	Dupuis	Paris	1983-03-06	135
Marie	Bernard	Paris	1993-07-03	75
Sophie	Dupond	Marseille	1986-02-22	27
Marcel	Martin	Paris	1976-11-24	39
Marion	Leroy	Lyon	1982-10-27	285
Paul	Moreau	Lyon	1976-04-19	133

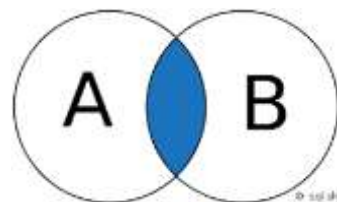
# UNION ALL



```
SELECT * FROM magasin1_client
UNION ALL
SELECT * FROM magasin2_client
```

prenom	nom	ville	date_naissance	total_achat
Léon	Dupuis	Paris	1983-03-06	135
Marie	Bernard	Paris	1993-07-03	75
Sophie	Dupond	Marseille	1986-02-22	27
Marcel	Martin	Paris	1976-11-24	39
Marion	Leroy	Lyon	1982-10-27	285
Paul	Moreau	Lyon	1976-04-19	133
Marie	Bernard	Paris	1993-07-03	75
Marcel	Martin	Paris	1976-11-24	39

# INTERSECT

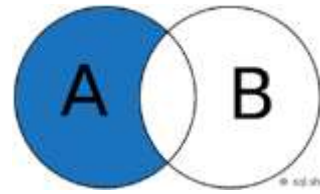


```
SELECT * FROM magasin1_client  
INTERSECT  
SELECT * FROM magasin2_client
```

prenom	nom	ville	date_naissance	total_achat
Marie	Bernard	Paris	1993-07-03	75
Marcel	Martin	Paris	1976-11-24	39



# MINUS/EXCEPT



Permet de récupérer les enregistrements de la première instruction sans inclure les résultats de la seconde requête.

- EXCEPT : PostgreSQL
- MINUS : MySQL et Oracle

# EXCEPT

id	prenom	nom	date_inscription
1	Lionel	Martineau	2012-11-14
2	Paul	Cornu	2012-12-15
3	Sarah	Schmitt	2012-12-17
4	Sabine	Lenoir	2012-12-18

Table « clients inscrits »

id	prenom	nom	date_inscription
1	Paul	Cornu	2013-01-27
2	Manuel	Guillot	2013-01-27
3	Sabine	Lenoir	2013-01-29
4	Natalie	Petitjean	2013-02-03

Table « clients refus email »

Sélectionner uniquement le prénom et le nom des utilisateurs qui accepte de recevoir des emails informatifs.

# EXCEPT

```
SELECT prenom, nom FROM clients_inscrits  
EXCEPT  
SELECT prenom, nom FROM clients_refus_email
```

prenom	nom
Lionel	Martineau
Sarah	Schmitt