

---

# Bases de données 2 (SQL, PL/SQL-ORACLE)

---

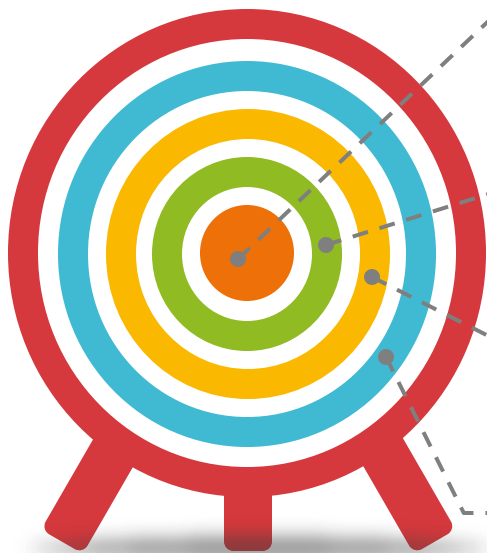
Présenté par:

**Dr. Maria EL HAIBA (EMSI)**



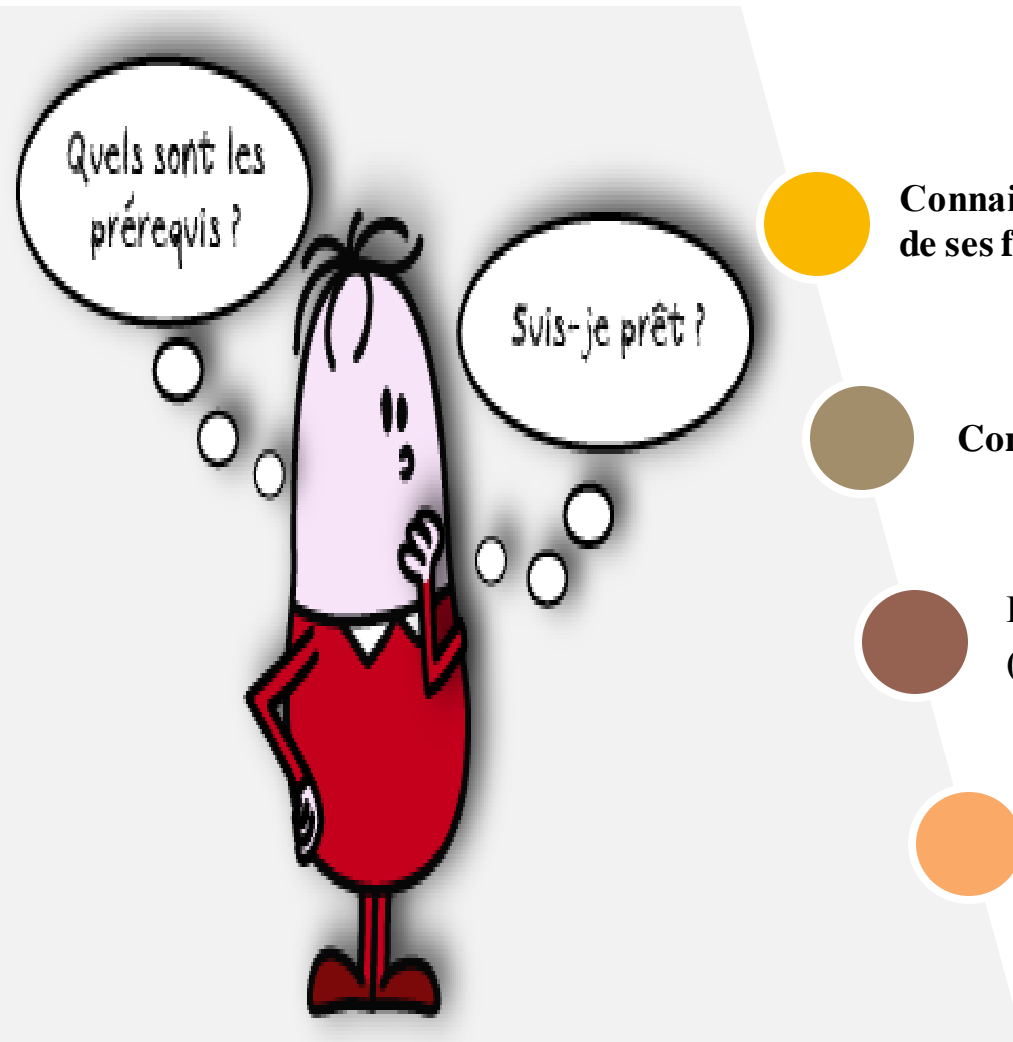
*m.elhaiba@emsi.ma*

# Objectifs de ce cours



- **Rappeler les concepts clés d'une base de données relationnelle et du langage SQL**
- **Se familiariser avec le Système de Gestion de Bases de Données proposé par Oracle**
- **Découvrir le langage PL/SQL et s'initier à ses différentes requêtes**
- **Maîtriser les éléments du Langage PL/SQL: Procédure, Fonctions, Curseurs,...**

# Prérequis



**Connaissances de base du système d'exploitation Microsoft Windows et de ses fonctionnalités principales**

**Conception de BD : Modèle entité-association, Modèle relationnel**

**BD relationnelle, Programmation SGBD, Requêtes SQL,...  
(On fera un Rappel, Ne vous inquiétez pas)**

**Quelques notions d'anglais (Autant de commandes en anglais)**

## **Introduction: Mise en Situation**

1. Introduction aux Bases de données
2. Systèmes de Gestion de Bases de Données SGBD
3. SGBD Relationnelles
4. SGBD Oracle

## **Partie 1 - Langage SQL**

1. Introduction au langage SQL
2. Langage de Définition des Données (LDD)
3. Langage de Manipulation des Données (LMD)

## **Partie 2 - Langage PL/SQL**

1. Concepts de base de PL/SQL
2. Transactions & Curseurs
3. Gestion des exceptions, Fonctions, Procédures et Packages
4. Déclencheurs (Triggers)

# Sommaire



# Introduction: Mise en Situation

1. Introduction aux Bases de Données BD
2. Systèmes de Gestion de Bases de Données SGBD
3. SGBD Relationnelles
4. SGBD Oracle
5. Démarche de construction d'une BD



## Base de donnée: Concept de base

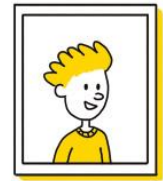
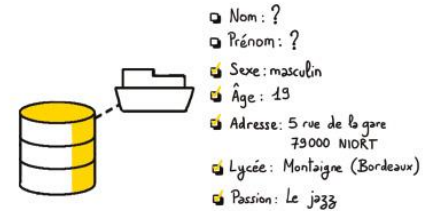


Pour comprendre ce qu'est une base de données, il est important de comprendre d'abord ce que sont véritablement **les données**.

Il s'agit **d'informations brutes** pouvant être liées à n'importe quel objet. Elles peuvent prendre de nombreuses **formes** différentes : nombres, images, fichiers de texte...

# Exemple: Données Personnelles

- Une « **donnée personnelle** » est « toute information se rapportant à une personne physique.
- Une personne peut être identifiée : **directement** (Ex. nom, prénom) ou **indirectement** (Ex. par un identifiant : N° client, un numéro : de téléphone, une donnée biométrique, ...).
- L'identification d'une personne peut être réalisée :
  - A partir d'une **seule donnée** (Ex. N° Carte nationale, N° Immatriculation)
  - A partir du **croisement d'un ensemble de données** (Ex. : Un jeune vivant à telle adresse, né tel jour, étudiant à tel lycée et ayant une telle passion)



Marc PELLETIER



Je suis une base  
de données personnelles

# Base de donnée: Concept de base

Une **base de données** (*son abréviation est BD, en anglais DB, database*) est une **collection organisée de données structurées**. Généralement stockée sur un ordinateur (ou support informatique), elle permet d'accéder, de gérer, de manipuler et de mettre à jour facilement les données (*Data Management*).





# Exemple: Base de Données



En guise d'exemple, on peut évoquer **un annuaire téléphonique en ligne**:

Une telle plateforme utilise une base de données pour stocker les données sur les personnes, le nom, l'adresse, les numéros de téléphone et autres informations de contact.

Autres applications utilisant des bases de données:

- *Gestion du personnel, étudiants, cours, inscriptions*
- *Gestion de comptes clients des banques et des transactions*
- *E-commerce: vendeurs, acheteurs, produits,...*
- ...
- *Tous les domaines de la vie quotidienne (Economie, éducation, transport,...).*

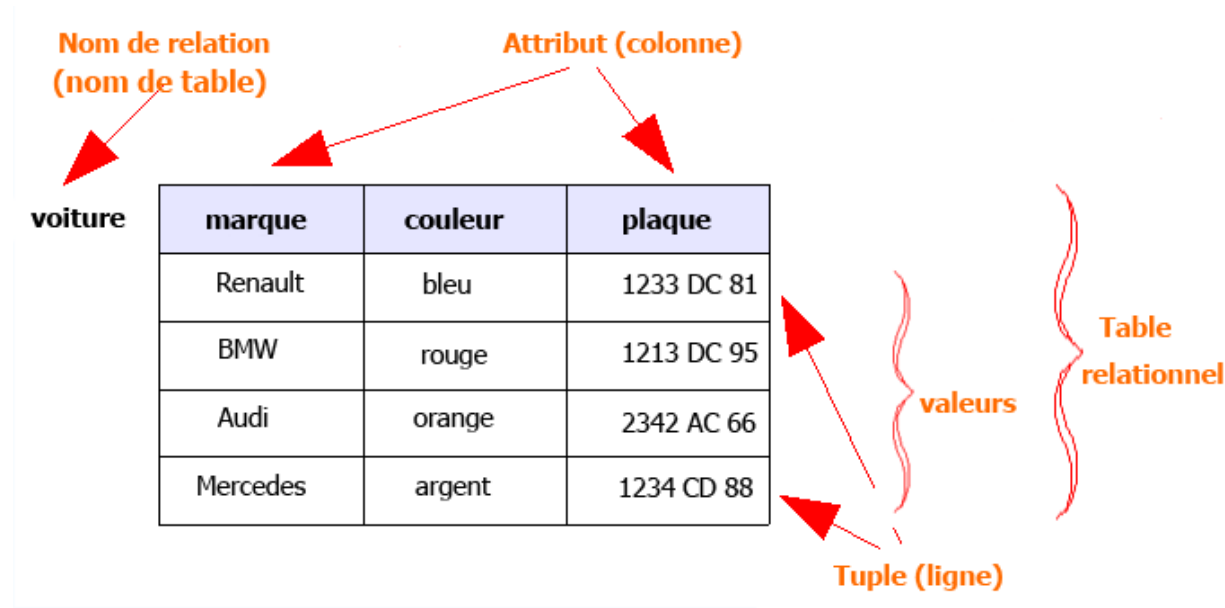
# Types de modèles d'une BD

Evolution depuis le début des années 60

- **Base de données hiérarchique**
  - Lie les enregistrements dans une structure arborescente où chaque enregistrement n'a qu'un seul possesseur.
- **Base de données en réseau**
  - Est une base hiérarchique mais permet en plus d'établir des relations transverses.
- **Base de données relationnelle**
  - Stocke les informations décomposées et organisées dans des matrices appelées relations ou tables.
- **Base de données orientée objet**
  - Stocke les informations groupées sous forme de collections d'objets persistants.
- Plus récemment, les **bases de données NoSQL**
  - Réponse à la croissance d'Internet (immenses volumes de données), ainsi qu'à la nécessité d'une vitesse supérieure et d'un traitement de données non structurées (Big Data).
- Aujourd'hui les **bases de données Cloud**
  - Stocker, de gérer et de récupérer des données dans le Cloud (Microsoft Azure, AWS, Google Cloud,...).

## Qu'est qu'une BD relationnelle?

- Présentées par CODD en 1970, les **BDs relationnelles** ont dominé les années 80.
- Une base de données relationnelle permet d'organiser les données en un ensemble de **tables** (appelés **relations**) comportant des **lignes** (appelées **tuples, enregistrements**) et des **colonnes** (appelés **attributs**).



# BD relationnelle (Modélisation)

- La **modélisation** d'une base de données relationnelle peut être faite selon plusieurs modèles, le plus simple est le modèle de données **entité-association (MCD)**

**Entité:** 'Objet', unité élémentaire pourvue d'une existence propre

– Client, Fournisseur, Commande, Produit sont des entités.

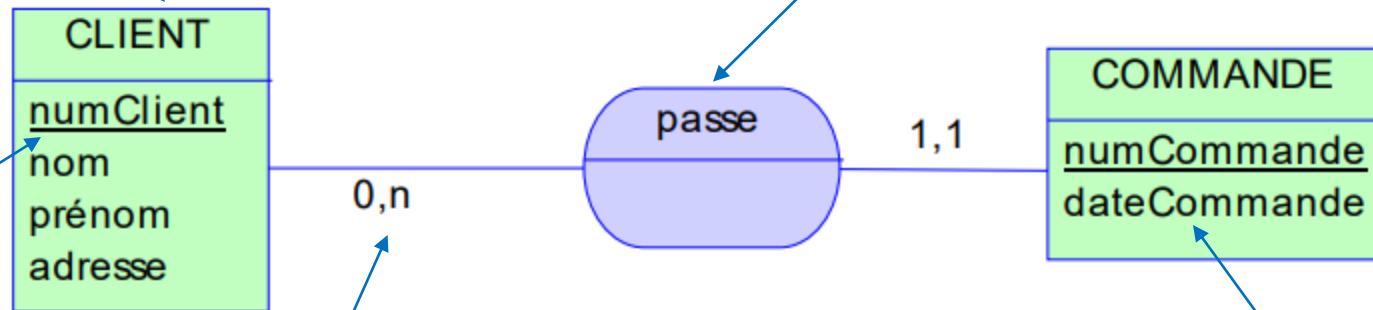
**Association:** Relation entre entités, dépourvue d'existence propre.

– 'passe' est une association entre l'entité Client et l'entité Commande.

**Identifiant:** C'est une (ou plusieurs) propriété(s) permettant d'identifier les occurrences d'une entité d'une manière unique

**Propriété :** Donnée élémentaire caractérisant partiellement une entité ou une association.  
– Nom, Prénom, dateCommande, etc... sont des propriétés

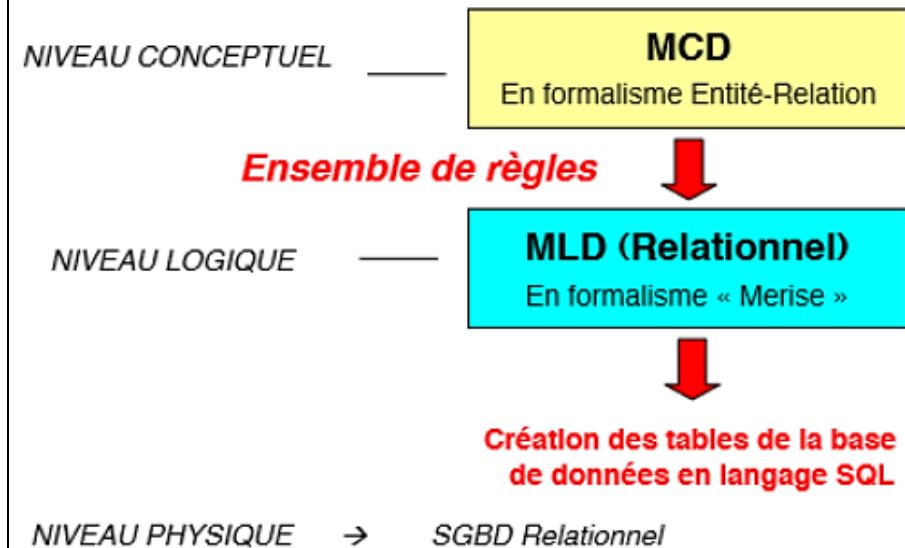
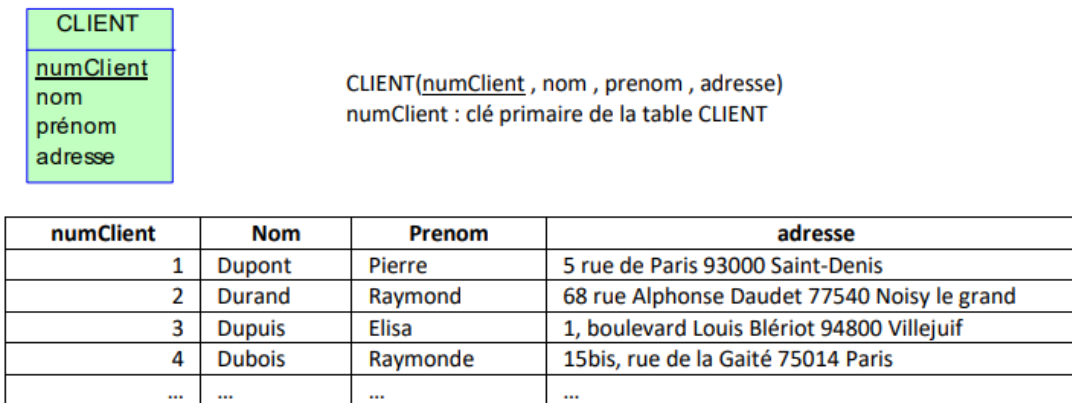
**Cardinalité:** Nombre minimum et maximum d'occurrences d'une association pour une occurrence d'entité



# BD relationnelle (Modélisation)

## Passage au Modèle relationnel

### Conversion des entités :



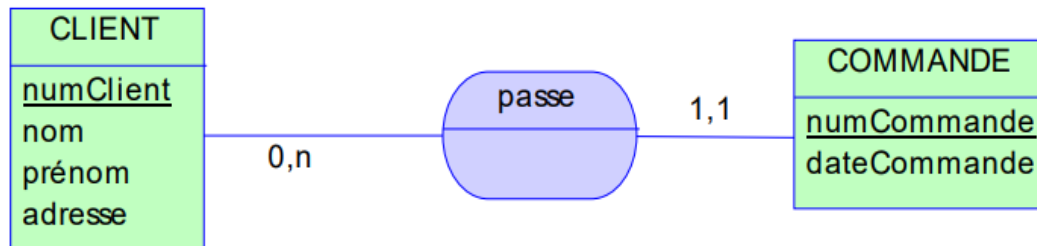
- Une **entité** du MCD devient une **relation**, c'est à dire une **table**.
- Son **identifiant** devient la **clé primaire** de la relation.
- Les **autres propriétés** deviennent les **attributs** de la relation.

# BD relationnelle (Modélisation)

## Passage au Modèle relationnel

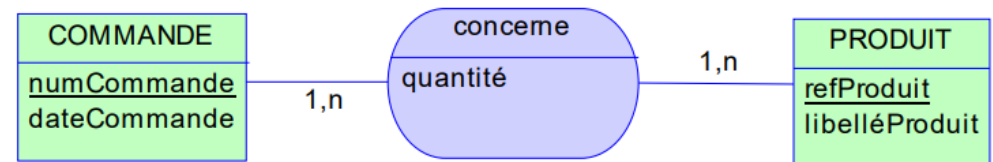
### Conversion des associations :

#### ➤ Association binaire de type $(*, 1 - *, N)$



CLIENT(numClient , nom , prénom , adresse)  
numClient : clé primaire de la table CLIENT  
COMMANDE(numCommande , dateCommande , #numClient)  
numCommande : clé primaire de la table COMMANDE  
#numClient : clé étrangère qui référence numClient de la table CLIENT

#### ➤ Association de type $(*, N), (*, N)$

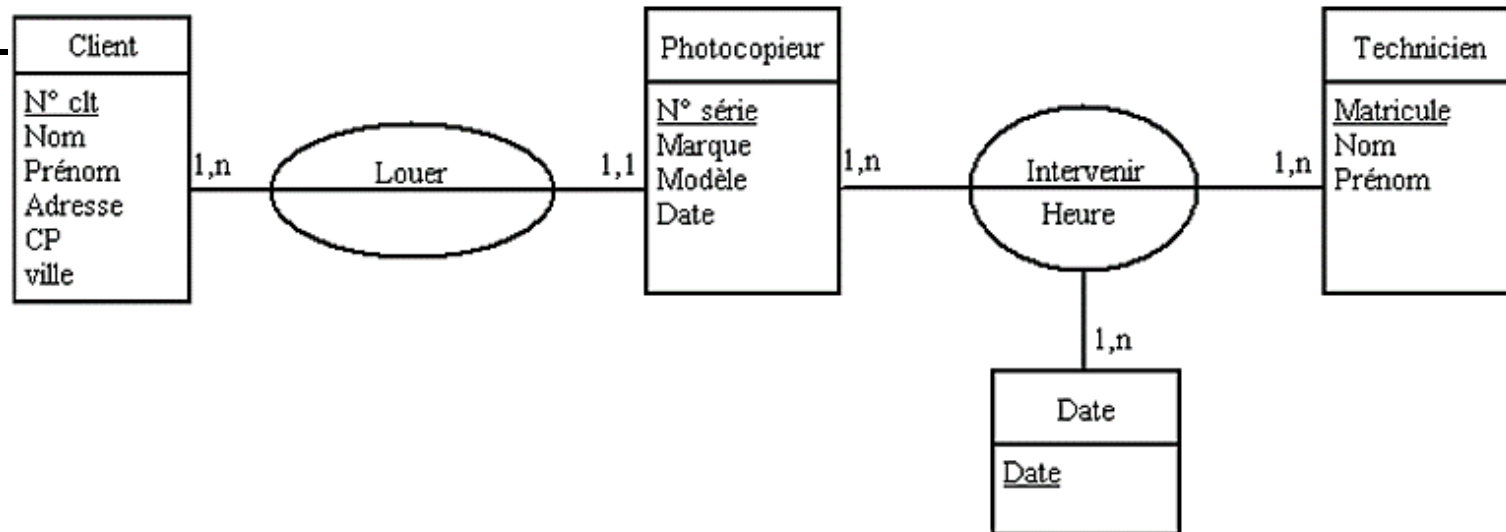


COMMANDE(numCommande , dateCommande)  
numCommande : clé primaire de la table COMMANDE  
PRODUIT(refProduit , libelléProduit)  
refProduit : clé primaire de la table PRODUIT  
CONCERNE(#numCommande , #refProduit , quantité)  
#numCommande , #refProduit : clé primaire composée de la table CONCERNE  
#numCommande : clé étrangère qui référence numCommande de la table COMMANDE  
#refProduit : clé étrangère qui référence refProduit de la table PRODUIT

# Exercice

## Enoncé (1/2):

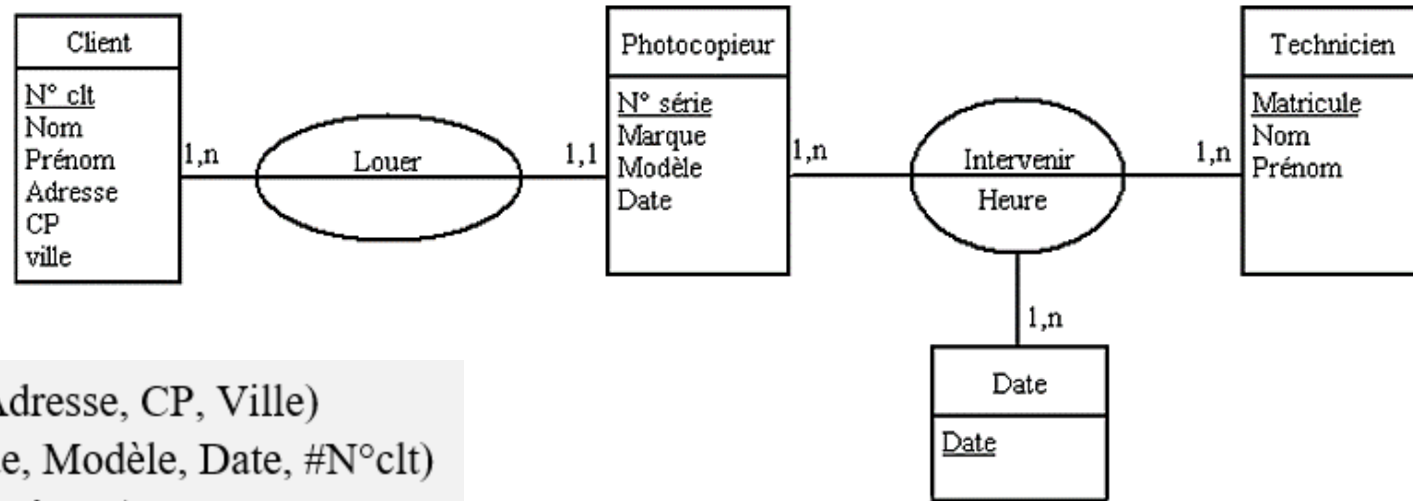
La société Hiez loue des **photocopieurs** à ses **clients**. La location d'un photocopieur est obligatoirement assortie d'un contrat de maintenance, qui prévoit l'intervention d'un **technicien** dans les 24 h. Pour améliorer la gestion des interventions, le propriétaire souhaite recourir à une base de données. Son fils, en a élaboré le modèle conceptuel des données (MCD ou Modèle E-A) suivant.



# Exercice

## Enoncé (2/2):

- a. Citez une entité et une association.
- b. Expliquez les cardinalités Client -1,n- Louer et Photocopieur -1,1- Louer.
- c. Écrivez le modèle relationnel correspondant au modèle conceptuel réalisé.



## Réponse :

**Client** (N°clt, Nom, Prénom, Adresse, CP, Ville)

**Photocopieur** (N°série, Marque, Modèle, Date, #N°clt)

**Technicien** (Matricule, Nom, Prénom)

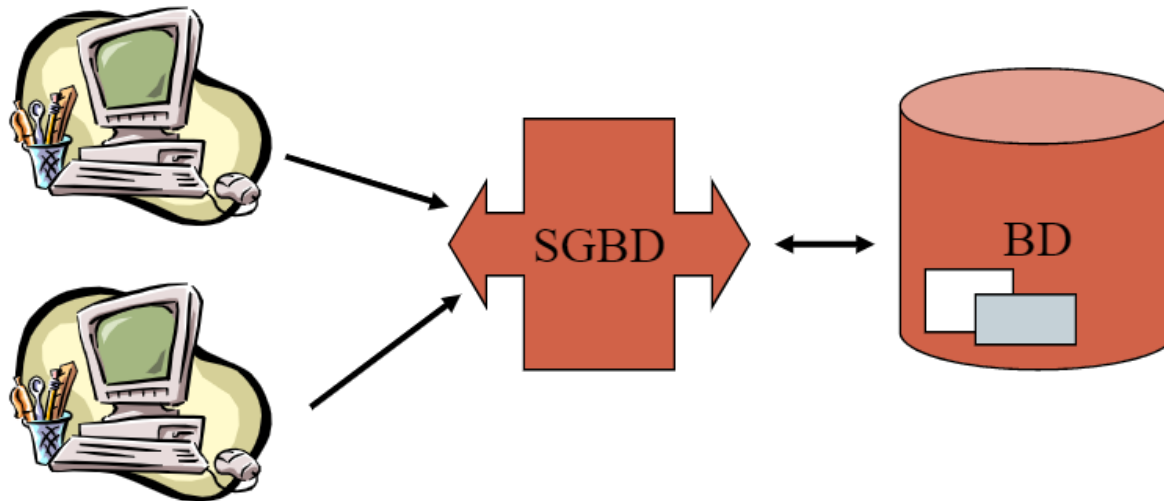
**Intervenir** (#N°série, #Matricule, #Date, Heure)

**Date** (Date)



## SGBD: C'est quoi?

- Afin de pouvoir contrôler les données ainsi que les utilisateurs, la gestion de la base de données se fait grâce à un système appelé SGBD (Système de Gestion des Bases de Données).
- Simplement, Un **SGBD** est le **logiciel** qui permet d'interagir avec une **base de données**.



C'est l'**interface** entre les **utilisateurs** et l'**information brute**.

# Objectifs d'un SGBD



## Manipulation des données

*Interrogation, mise à jour, suppression, ...*

Objectif

01

Objectif

02

## Indépendance données/programme

*Une modification de l'organisation des données n'a pas d'impact sur l'utilisateur*



## Partage et sécurité des données

*Utilisation simultanée des données par différentes applications  
Données protégées contre les accès non-autorisés*

Objectif

06



Objectif

03

## Accès efficace aux données

*Garantie un bon débit et un bon temps de réponse*



## Cohérence des données

*Assurer le respect des règles (contraintes d'intégrité, Ex. âge d'une personne doit être un entier positif)*

Objectif

05

Objectif

04

## Redondance contrôlée des données

*Diminution du volume de stockage, Pas de données multiples ou dupliquées*



# Qu'est qu'un SGBD relationnel?

- Un **SGBDR** est un système de gestion de base de données relationnelles ;
- Les SGBD relationnels mettent au premier plan les relations entre les données. Celles-ci sont organisées en tables à deux dimensions. On parle alors de ligne et de colonnes ;
- Un SGBDR propose les trois principales fonctions suivantes :
  - *La définition des données sous forme de **relations** (Utilisation de LDD);*
  - *La manipulation des données par un **langage** déclaratif (Utilisation de LMD);*
  - *Le **contrôle** et l'**administration** des données (Utilisation de LCD).*

# Exemples de SGBDRs

- Les SGBD **relationnels** sont les plus courants des SGBD ;
- Les plus répandus sont :



- Toutefois et avec l'arrivée des géants du web (Google, Amazon ou Facebook,...) gérant des quantités énormes de données, un type de bases de données **non-relationnelles** (également appelées *NoSQL*), s'est développé (*MongoDB, Cassandra, Oracle NoSQL,...*).

## SGBD Oracle

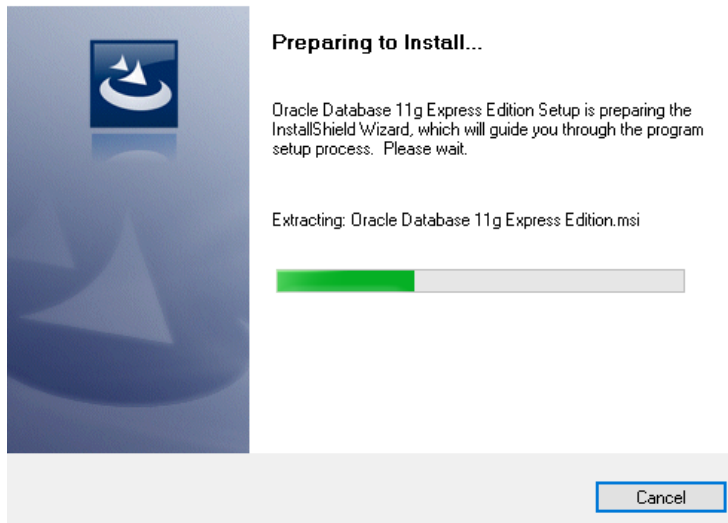
- Oracle est un SGBDR édité et commercialisé par **Oracle Corporation** ;
- Oracle est capable de gérer un grand volume de données, il dispose d'un langage procédurale très puissant qui est **PL/SQL**.



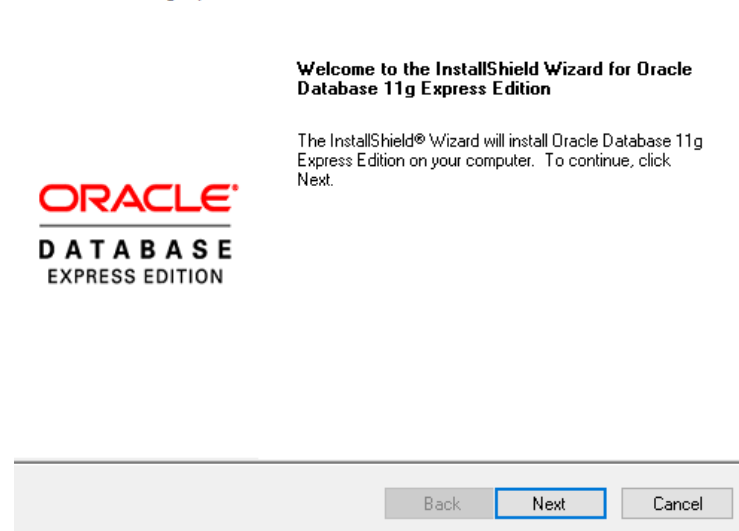
# Installation du SGBD Oracle

1. Télécharger la version du « **Oracle Database Express Edition 11g Release 2** » que vous pouvez trouver sur le lien officiel (selon votre système d'exploitation) : <https://www.oracle.com/database/technologies/xe-downloads.html>
2. Dé-zipper le fichier téléchargé, puis ouvrir le dossier. Double-cliquez sur **setup.exe**. L'écran d'installation apparaît :

Oracle Database 11g Express Edition - InstallShield Wizard



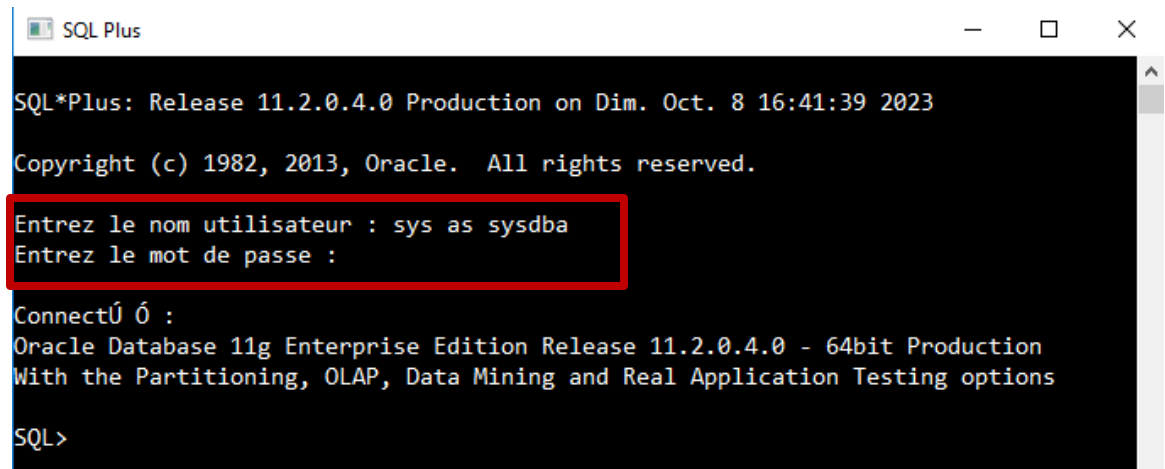
Oracle Database 11g Express Edition - Install Wizard



# Installation du SGBD Oracle

3. Configurer votre installation en **acceptant les termes et les conditions** d'utilisation tout en **choisissant le répertoire** d'installation.
4. À la fin de l'installation, vous pouvez **commencer à utiliser** « Oracle Database XE » normalement, tous les programmes en relation apparaissent dans le menu de démarrage.
5. Vous pouvez ainsi cliquer sur « **Run SQL Command Line** » **ou** « **SQL PLUS** » **selon la version installée**, entrer le nom d'utilisateur et le mot de passe que vous avez donné lors de l'installation.

Fenêtre de l'utilitaire Oracle SQL\* Plus

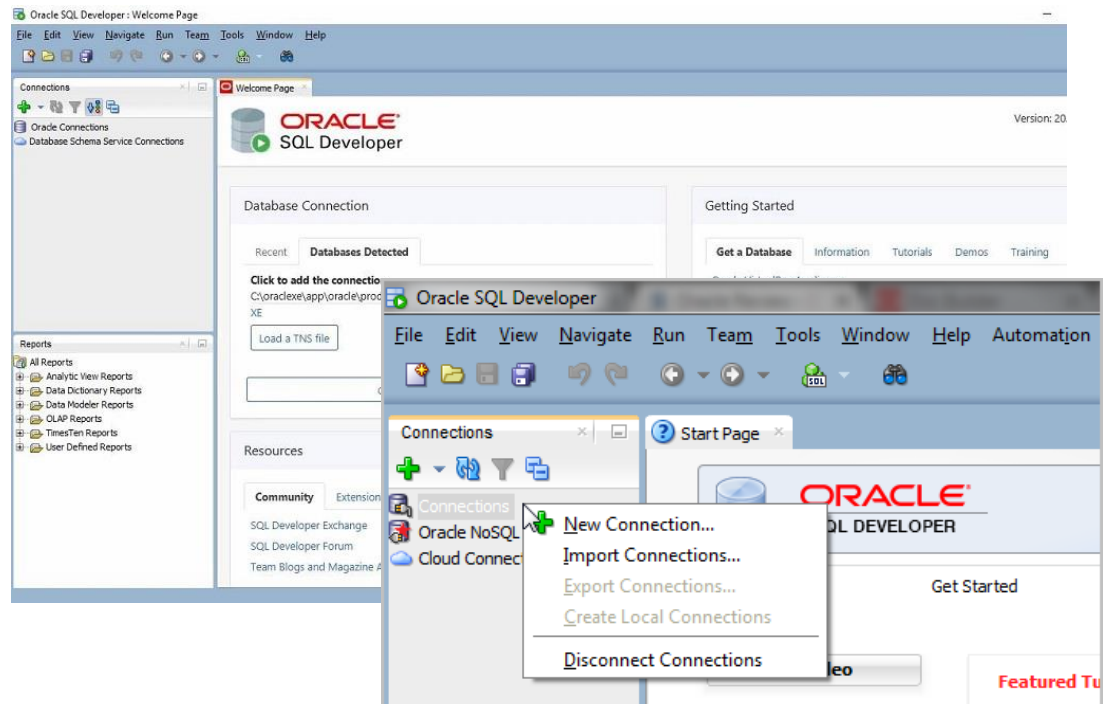


```
SQL Plus
SQL*Plus: Release 11.2.0.4.0 Production on Dim. Oct. 8 16:41:39 2023
Copyright (c) 1982, 2013, Oracle. All rights reserved.
Entrez le nom utilisateur : sys as sysdba
Entrez le mot de passe :
Connecté à :
Oracle Database 11g Enterprise Edition Release 11.2.0.4.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
SQL>
```

# Installation du SGBD Oracle

6. Ou bien d'y accéder via l'IDE « **SQL Developer** », que vous pouvez télécharger à partir du site officiel d'ORACLE ([Lien de téléchargement](#))
7. En créant une **nouvelle connection**, il vous serait possible de commencer votre travail.

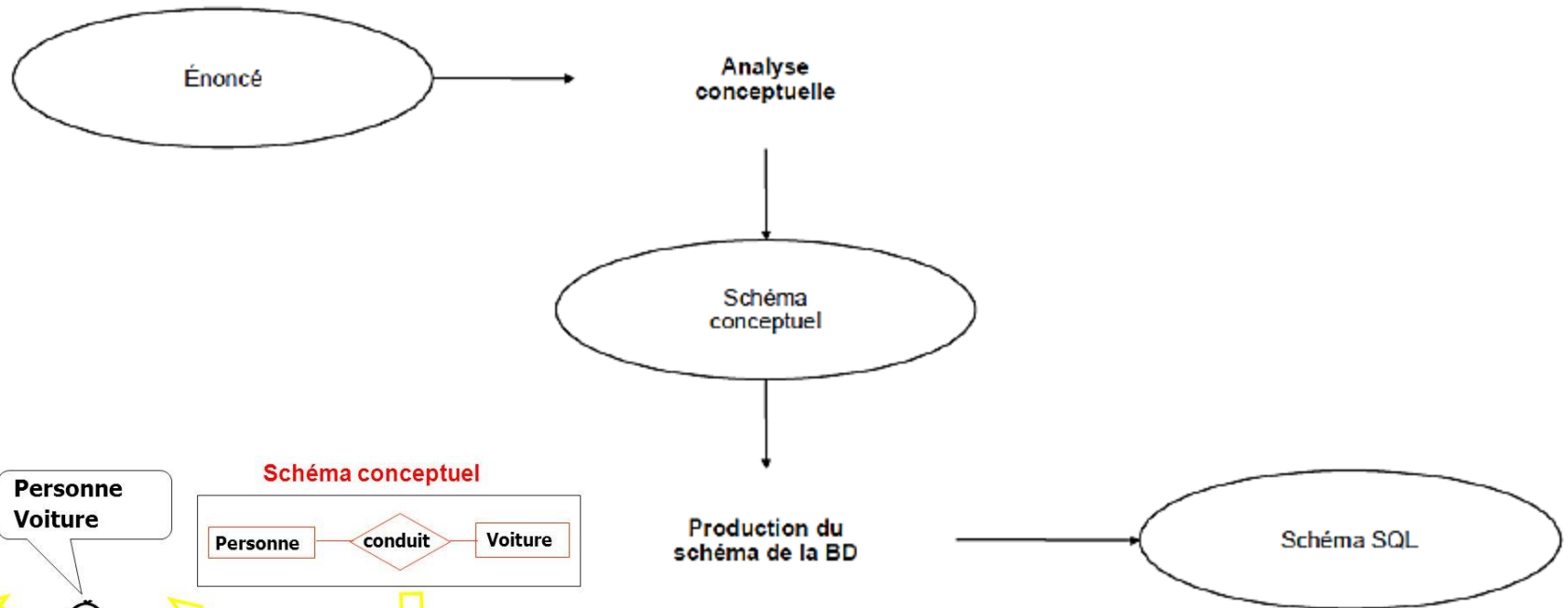
Interface de l'outil Oracle SQL Developer



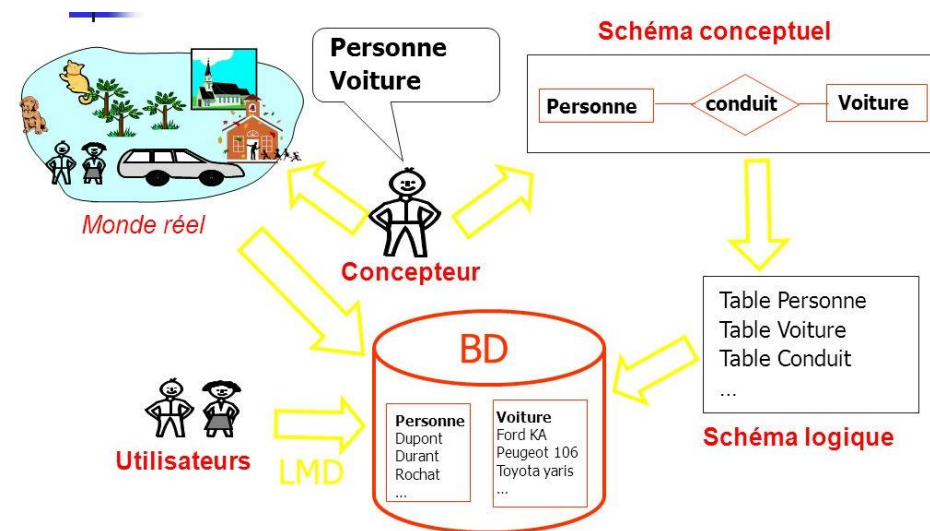


# Démarche de construction d'une BD

## Synthèse



## Exemple



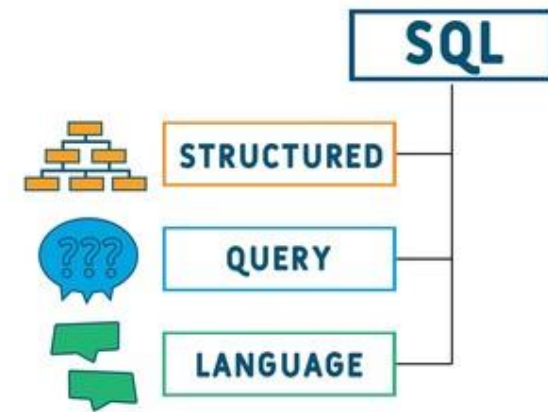
# Partie 1 : Le langage SQL

1. Introduction au langage SQL
2. Langage de Définition des Données (LDD)
3. Langage de Manipulation des Données (LMD)
4. Exercices



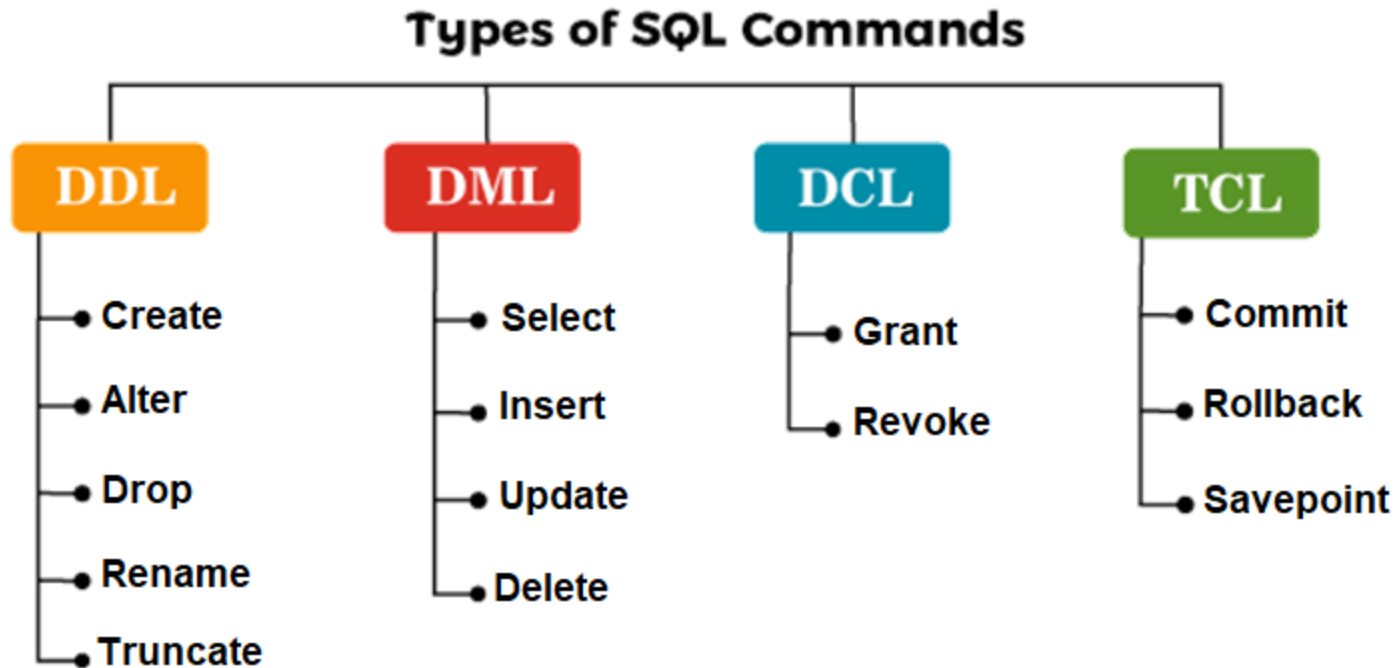
## Brièvement, c'est quoi SQL?

- SQL (Structured Query Language, en français: Langage de requête structurée) est le langage de référence pour **interroger** les bases de données ;
- Il permet de **modifier la structure** de la base de données: Ajouter, modifier, supprimer, mettre à jour des tables, des utilisateurs, gérer les droits,...



# Commandes SQL

SQL est essentiellement un **ensemble de commandes** permettant d'exploiter une base de données relationnelles. Ces commandes peuvent-être regroupées comme suit:



## Introduction LDD

Le Langage de Définition des Données (**LDD**) : permet de créer, modifier ou supprimer des objets dans une BD relationnelle (tables, vues, indexes, procédures, fonctions...).

**CREATE** : Création des objets.

**ALTER** : Modification de la structure des objets.

**DROP** : Suppression des objets.

## Syntaxe de création d'une table : CREATE TABLE

```
CREATE TABLE nom Table  
(  
  colonne type [contrainte de la colonne]  
  [, colonne type [contrainte de la colonne]]  
  ...  
  [, contrainte de la table] ... ) ;
```

### Exemple simple de création :

```
Create table Véhicule  
(  
  NumVeh NUMBER(10),  
  Matricule varchar(15),  
  Marque varchar(20),  
  IDCLT int);
```



Une **contrainte d'intégrité** est une règle qui définit la cohérence d'une donnée ou d'un ensemble de données de la base de données.

## Contrainte sur une colonne

```
[ CONSTRAINT <nom de la contrainte> ]  
    [ NOT NULL | Pour les colonnes obligatoires  
    PRIMARY KEY | Pour les clés primaires (= UNIQUE + NOT NULL)  
    UNIQUE | Pour les identifiants secondaires  
    CHECK (condition) | Pour vérifier une condition lors de l'insertion.  
    REFERENCES <nom de la table> (colonne) : Pour annoncer une clé étrangère.  
]
```

## Contrainte sur une table

```
[ CONSTRAINT <nom de la contrainte>  
    [ UNIQUE (liste de colonnes) |  
    PRIMARY KEY (liste de colonnes) |  
    CHECK (condition) |  
    FOREIGN KEY (liste de colonnes)  
    REFERENCES <nom de la table> (liste colonnes) [<mode>]  
    ]  
]
```

```
[<mode>::=[ON DELETE {CASCADE|SET DEFAULT|SET NULL}]  
| [ON UPDATE {CASCADE| SET DEFAULT| SET NULL}]  
]
```

## Exemple de création d'une table avec contraintes :

```
Create table Véhicule  
(  
  NumVeh NUMBER(10) primary key,  
  Matricule varchar(15) not null unique,  
  Marque varchar(20) not null,  
  IDCLT int references Clients (NumClit)  
);
```

Ou bien, sur Oracle

```
Create table Véhicule  
(  
  NumVeh NUMBER(10),  
  Matricule varchar(15) not null unique,  
  Marque varchar(20) not null,  
  IDCLT int,  
  constraint pk_vehicule primary key (NumVeh),  
  constraint fk_vehicule foreign key (IDCLT) references Clients (NumClit)  
);
```



## REMARQUE


Sur la colonne ou la table: si la contrainte ne fait intervenir qu'un **SEUL ATTRIBUT**.

Sur la table: si la contrainte fait intervenir **PLUSIEURS ATTRIBUTS**.

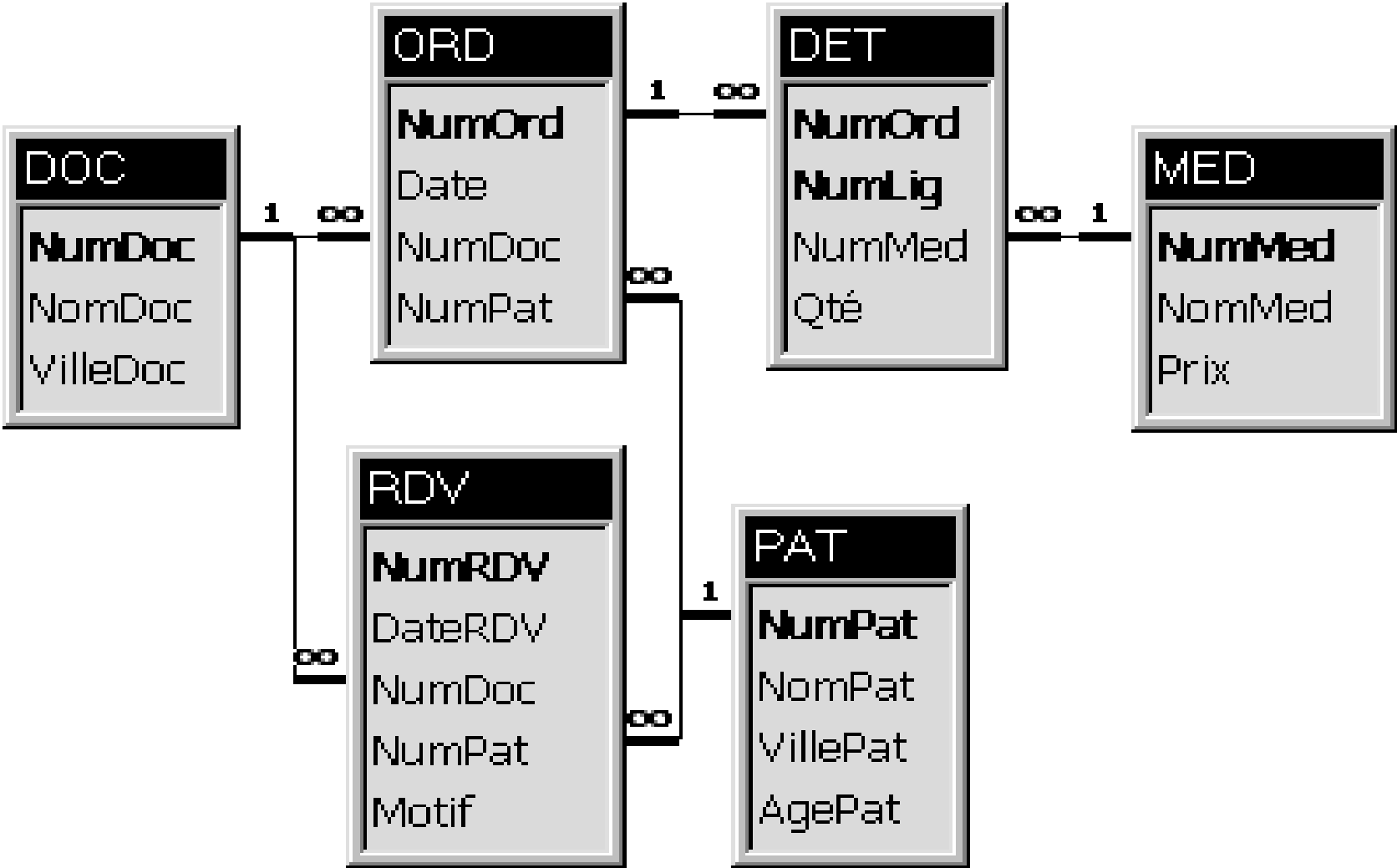
### Exemple : Clé primaire multiple/composée

LigneCommande ( <u>NumCde</u> , <u>NumProd</u> , QteCde)
--

```
Create table LigneCommande(  
  NumCde NUMBER(8),  
  NumProd NUMBER(6),  
  QteCde NUMBER(7,3),  
  constraint pk_LigneCde primary key(NumCde, NumProd),  
  constraint fk_Commande foreign key(NumCde) references Commande (NumCde),  
  constraint fk_Produit foreign key(NumProd) references Produit (NumProd));
```

 Dans le cas de clé primaire multiple, la clé primaire doit être créée comme **contrainte de table**.

**Exemple :** Schéma d'une base de données, Gestion d'un Cabinet



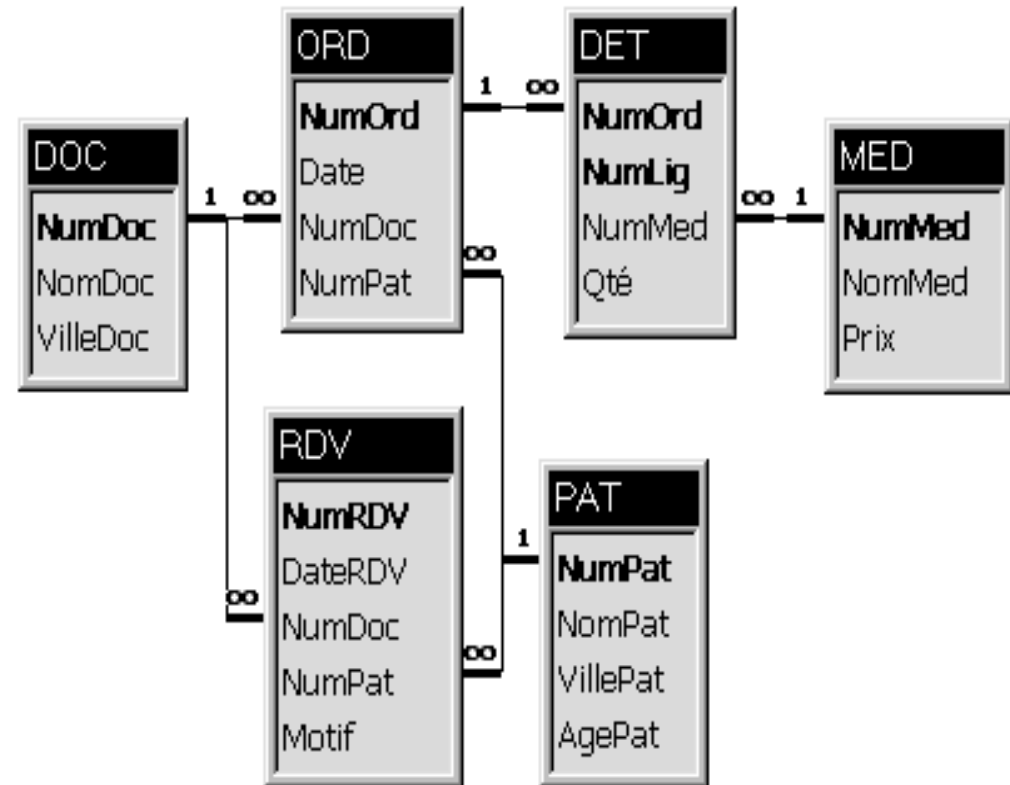
## Création de Tables

### Méthode 1 :

```
Create Table DOC(  
    NumDOC integer PRIMARY KEY,  
    NomDOC VARCHAR2(20),  
    VilleDOC VARCHAR2(20)  
);
```

### Méthode 2 :

```
Create Table DOC(  
    NumDOC integer,  
    NomDOC VARCHAR2(20),  
    VilleDOC VARCHAR2(20),  
  
Constraint PK_DOC Primary Key (NumDOC)  
);
```



## Méthode 3 :

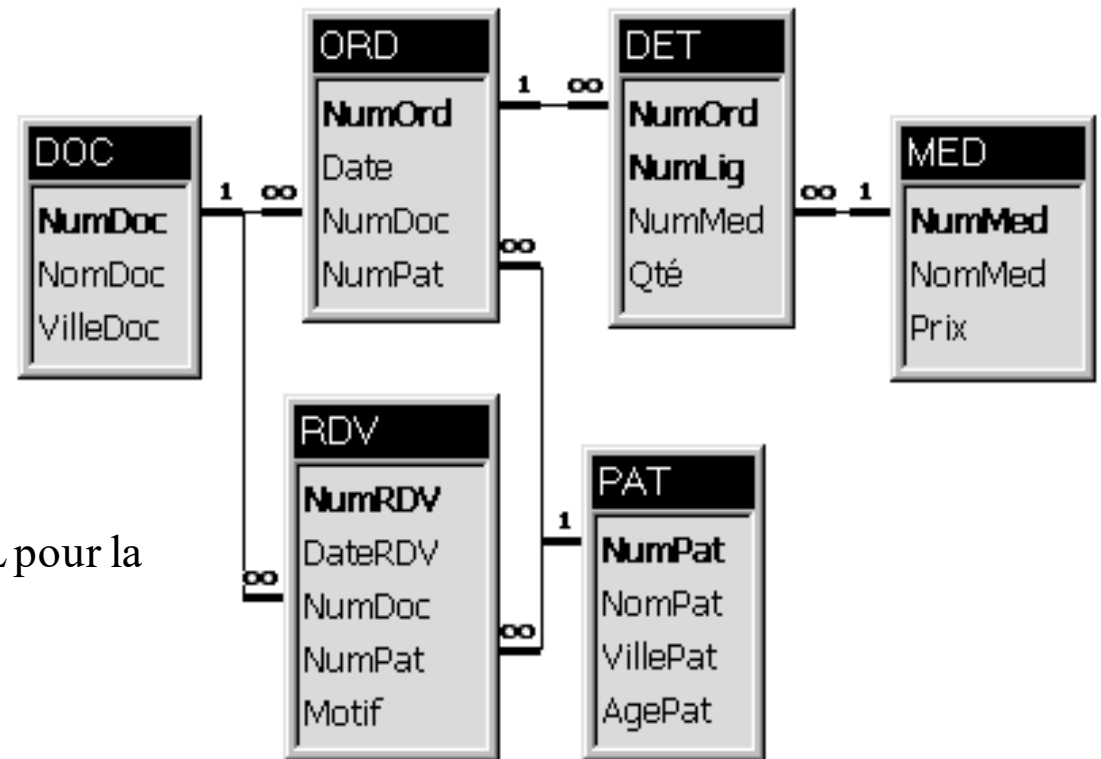
**Create Table DOC(**

NumDOC integer **Constraint PK\_DOC PRIMARY KEY,**

NomDOC VARCHAR2(20),

VilleDOC VARCHAR2(20)

);



**Application :** Donnez l'expression SQL pour la création de la table RDV.

## Exemple de Création de la Table DET

**Create Table DET(**

NumORD integer,

NumLigne integer,

NumMED integer,

QTE integer **Not Null**,

**Constraint PK\_DET Primary Key (NumORD, NumLigne),**

**Constraint NbMaxMed Check (NumLigne < 5),**

**Constraint Ref\_ORD**

**Foreign Key (NumORD) References ORD(NumORD)**

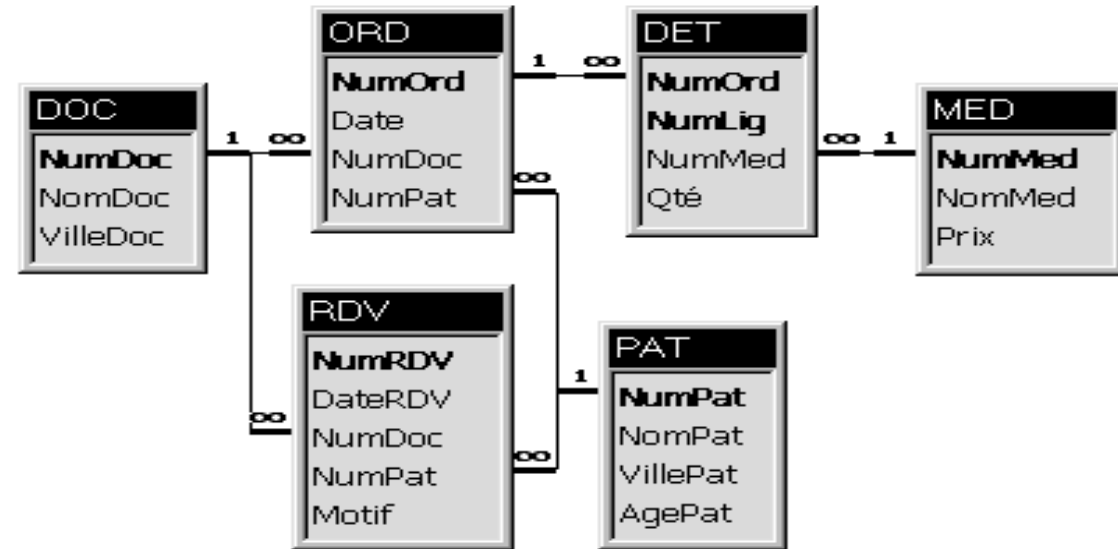
**on update cascade,**

**Constraint Ref\_MED**

**Foreign Key (NumMED) References MED(NumMED)**

**on delete cascade**

**);**



## Syntaxe CREATE DOMAIN

Il se peut que l'utilisateur ait parfois besoin de créer un nouveau **domaine** : qui est essentiellement un type de données avec des **contraintes** optionnelles (pouvant servir à typer des attributs).

```
CREATE DOMAIN <nom domaine> <type> [valeur]
[CONSTRAINT nom_contrainte CHECK (condition) ]
```

### Exemple :

```
CREATE DOMAIN TypeNomDOC VARCHAR2(20);
```

```
CREATE DOMAIN DATE_RDV DATE
  DEFAULT (CURRENT_DATE)
  CHECK (VALUE >= CURRENT_DATE)
  NOT NULL ;
```

### Utilisation :

```
Create Table RDV(
  NumRDV int PRIMARY KEY,
  DateRDV DATE_RDV,
  NumDOC int,
  ...
);
```

## Syntaxe de modification d'une table : ALTER TABLE

```
ALTER TABLE <nom de la Table>
{
    ADD COLUMN <def Colonne> |
    DROP COLUMN <nom Colonne> [RESTRICT|CASCADE] |
    MODIFY<nouvelle déf. Colonne>
    ADD CONSTRAINT <def Contrainte> |
    DROP CONSTRAINT <nom Contrainte> [RESTRICT|CASCADE] |
    RENAME COLUMN Nom_Colonne TO Nouveau_Nom
}
```



**RESTRICT** : Pas de destruction si l'objet est référencé ou utilisé ailleurs

**CASCADE** : Propage la destruction

# Modification d'une Table et de ses colonnes

## Exemples

```
ALTER TABLE DOC ADD COLUMN TEL NUMBER NOT NULL;
```

```
ALTER TABLE DOC MODIFY Ville VARCHAR2(10);
```

```
ALTER TABLE DOC DROP COLUMN TEL;
```

```
ALTER TABLE DOC ADD CONSTRAINT NN_NOM NomDoc NOT NULL;
```

```
ALTER TABLE DOC ADD CONSTRAINT Ville_valide  
CHECK(Ville = 'Rabat' OR ville='Casa');
```

```
ALTER TABLE DOC DROP CONSTRAINT NN_NOM;
```



## Syntaxe de suppression d'une table : DROP TABLE

**DROP TABLE** <Nom de la table> ;

Exemple:

**DROP table** DOC ;

Il est notamment possible de **vider une table sans la supprimer** en utilisant :

**TRUNCATE TABLE** <nom de la Table> ;

Exemple:

**TRUNCATE table** DOC ;

# LES VUES

## Définition :

Table virtuelle calculée/définie par une requête à partir d'autres tables ou vues

Pas d'existence physique mais recalculée à chaque fois qu'elle est invoquée

Vue mono table/Vue multi-tables

**Une vue = une requête de sélection nommée et réutilisée en cas de besoin.**

## Intérêts :

- Indépendance application/données
- Personnalisation des données selon les besoins des utilisateurs
- Confidentialité : restreindre la visibilité des données (on ne donne accès qu'à la vue)
- Rapidité des requêtes

## Utilisation :

Pour les sélections (**Select**), comme une **table ordinaire**.

Pour les mäj. (**Insert, Update, Delete**), avec des **restrictions** selon le type de vue.

## Syntaxe de création d'une vue

**CREATE VIEW** <nom vue> [(liste des attributs)]

**AS** <requête de sélection>

**[WITH CHECK OPTION]**

### **WITH CHECK OPTION**

Permet de vérifier que les mises à jour ou les insertions faites à travers la vue ne produisent que des lignes qui feront partie de la sélection de la vue.

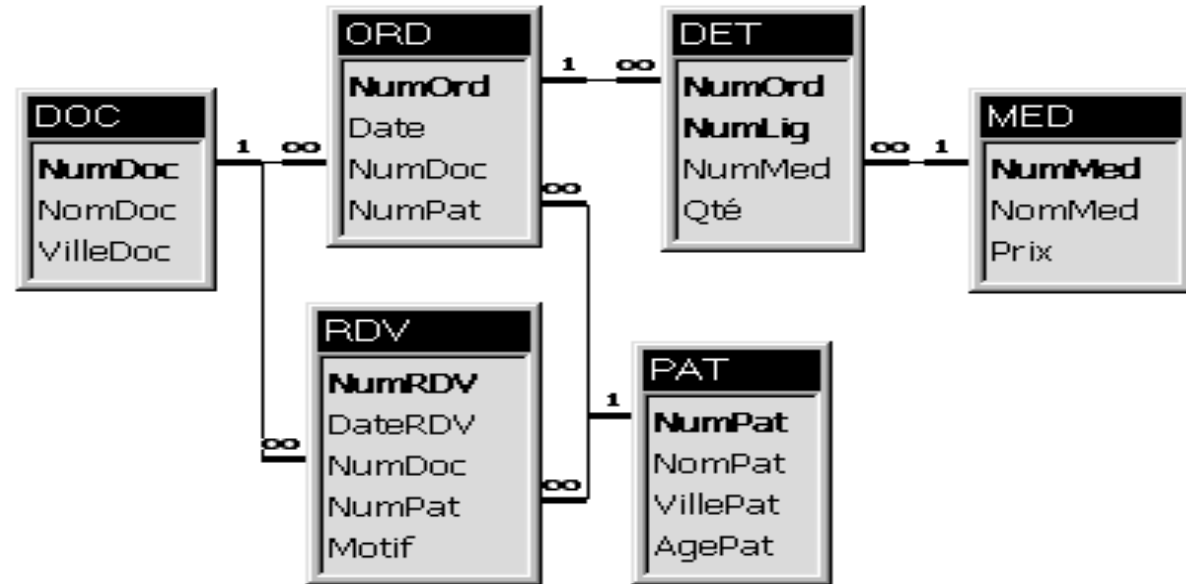
Notons que pour supprimer une vue, nous procédons ainsi :

**DROP VIEW** <nom vue>



**La suppression d'une vue n'entraîne pas la suppression des données.**

## Exemples :



```
CREATE VIEW MedecinsDeRabat AS
Select * From DOC Where VilleDoc='Rabat'
WITH CHECK OPTION ;
```

```
CREATE VIEW DocPat AS
Select NomDoc, NomPat FROM DOC D, RDV R, PAT P
WHERE D.NumDoc=R. NumDoc and R.NumPat=P.NumPat;
```

*Alias*

## Règles d'utilisations des VUES

Le SELECT principal de la vue contient	SELECT	UPDATE	DELETE	INSERT
Plusieurs tables	OUI	NON	NON	NON
GROUP BY	OUI	NON	NON	NON
DISTINCT	OUI	NON	NON	NON
Fonction de groupe/agrégation	OUI	NON	NON	NON
Attribut calculé	OUI	NON	OUI	NON
Attribut NOT NULL pas dans le SELECT	OUI	OUI	OUI	NON
UNION, INTERSECT, MINUS	OUI	NON	NON	NON

## Introduction LMD

Le Langage de Manipulation des Données (**LMD**) : permet d'insérer, de modifier, supprimer ou chercher des lignes dans une BD relationnelle.

**SELECT** : Définition de la liste des colonnes que l'on peut obtenir.

**INSERT** : Ajout des lignes à une table.

**UPDATE** : Mise à jour des colonnes d'une table.

**DELETE** : Suppression d'un ou de plusieurs enregistrements.

## Syntaxe simplifiée du SELECT

**SELECT** [ **DISTINCT** ] \* | **expr**[, **expr**...]

**FROM** **table** -- spécifie la table ou les tables à utiliser

[**WHERE** **condition**] -- filtre les lignes selon une condition donnée

[**ORDER BY** **expr**| **position** [**ASC**| **DESC**]] ; --spécifie l'ordre d'apparition des données dans le résultat



- ORDER BY est toujours la **dernière clause** sur une instruction SELECT.
- Ordonne l'affichage uniquement.
- **ASC** pour ascendant (Le choix par défaut)
- **DESC** pour descendant.



## Condition : Opérateurs de Comparaison

Permettent de **comparer** une colonne ou une expression à une autre colonne ou expression

– Comparaison de valeurs =, >, <, >=, <=, <>

**exp op\_relationnel exp**

– Intervalle BETWEEN

**exp [NOT] BETWEEN exp AND exp**

– Liste de valeurs IN

**exp [NOT] IN (liste\_de\_valeurs)**

– Comparaison avec filtre LIKE

**char\_exp [NOT] LIKE «chaine»** ( \_ un car; % n caractère)

– Indétermination IS NULL

**colonne IS [NOT] NULL**

## Exemples

```
SELECT * FROM Personnes ;
```

```
SELECT DISTINCT ville FROM Personnes ;
```

-- Afin d'afficher les villes des personnes sans répétitions (Unique ou Distinct)

```
SELECT * FROM Personnes WHERE ville != 'Rabat';
```

```
SELECT nom FROM Personnes  
WHERE nom Like 'R_v%';
```

```
SELECT nom FROM Personnes WHERE ville IN ( 'Rabat', 'Casa', 'Agadir' ) ;
```

```
SELECT nom, prenom FROM Personnes  
WHERE taille > 180  
ORDER BY nom ASC, naissance DESC ;
```

-- Pour obtenir un résultat trié (Croissant ou Décroissant)

## Fonctions de groupe (ou d'agrégation)

Permettent de réaliser des calculs sur des ensembles de données.

Fonction	Description
<b>Count</b> (*[ <b>DISTINCT</b>   <b>ALL</b> ] <b>expr</b> )	Le nombre de lignes de <b>expr</b>
<b>Avg</b> ( [ <b>DISTINCT</b>   <b>ALL</b> ] <b>expr</b> )	Valeur moyenne de <b>expr</b> , en ignorant les valeurs NULL
<b>Min</b> ( [ <b>DISTINCT</b>   <b>ALL</b> ] <b>expr</b> )	Valeur minimale de <b>expr</b> , en ignorant les valeurs NULL
<b>Max</b> ( [ <b>DISTINCT</b>   <b>ALL</b> ] <b>expr</b> )	Valeur maximale de <b>expr</b> , en ignorant les valeurs NULL
<b>Sum</b> ( [ <b>DISTINCT</b>   <b>ALL</b> ] <b>expr</b> )	Somme des valeurs de <b>expr</b> , en ignorant les valeurs NULL

## Exemples

*Alias*

```
SELECT Count(*) FROM Personnes ;
```

```
SELECT COUNT(telephone) as NombreTEL FROM Personnes ;
```

```
SELECT Avg(salaire), Sum(salaire), Min(salaire), Max(salaire) FROM Personnes;
```

```
SELECT Min(naissance), Max(naissance) FROM Personnes ;
```

```
SELECT count(DISTINCT Ville) FROM Personnes;
```

```
SELECT Count(telephone), COUNT(*)  
FROM personnes WHERE ville = 'Rabat';
```

## Question

**Salaire moyen pour chaque département de la table EMP ?**

**Table EMP**

<b>EMP</b>	<b>Deptno</b>	<b>job</b>	<b>sal</b>
	10	Dir technique	5000
	10	Chef projet	1500
	10	Programmeur	1300
	20	Chef projet	2975
	20	Analyste	3000
	20	Programmeur	1100
	30	Chef projet	2850
	30	Commercial	1250
	30	Commercial	1600
	30	Commercial	1500
	30	Programmeur	950
	30	Commercial	1250



## La clause GROUP BY

**SELECT** **column, group\_function(expression)**

**FROM** **table**

**[WHERE** **condition]**

**[ GROUP BY** **expr** ] *--forme des groupes de lignes de même valeur de colonne*

**[ORDER BY** **expr** | **position** [ASC | DESC]] ;

## REMARQUE

Les attributs du SELECT ne peuvent être que :

- L'attribut qui crée le groupe
- Une fonction de groupe.

## Exemples

Réponse :

Salaire moyen pour chaque département de la table EMP

```
SELECT deptno, AVG( sal )  
FROM emp  
GROUP BY deptno ;
```

EMP	Deptno	AVG(sal)
	10	2600
	20	2175
	30	1566.7

**Question**    Somme des salaires pour chaque poste (job), regroupés par département

**Table EMP**

EMP	Deptno	job	sal
	10	Dir technique	5000
	10	Chef projet	1500
	10	Programmeur	1300
	20	Chef projet	2975
	20	Analyste	3000
	20	Programmeur	1100
	30	Chef projet	2850
	30	Commercial	1250
	30	Commercial	1600
	30	Commercial	1500
	30	Programmeur	950
	30	Commercial	1250





## Exemples

Réponse :

Somme des salaires pour chaque poste (job), regroupés par département

```
SELECT deptno, job, SUM( sal )  
FROM emp  
GROUP BY deptno, job ;
```

Deptno	job	SUM(sal)
10	Programmeur	1300
10	Chef projet	1500
10	Dir technique	5000
20	Analyste	3000
20	Programmeur	1100
20	Chef projet	2975
30	Programmeur	950
30	Chef projet	2850
30	Commercial	5600

## GROUP BY avec HAVING

```
SELECT column, group_function(expression)

FROM table

[WHERE condition]

[ GROUP BY group_by_expr ]

[ HAVING condition]; --filtre les groupes sujets à une certaine condition

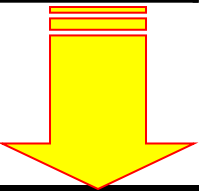
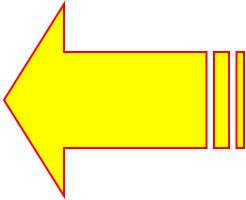
[ORDER BY expr| position [ASC| DESC]] ;
```

# Exemple

Maximum des salaires pour chaque département qui soit supérieur à 2900

```
SELECT deptno, MAX( sal )  
FROM emp  
GROUP BY deptno  
HAVING MAX( sal ) > 2900;
```

Deptno	MAX(sal)
10	5000
20	3000
30	2850



Deptno	MAX(sal)
10	5000
20	3000

EMP	Deptno	job	sal
	10	Dir technique	5000
	10	Chef projet	1500
	10	Programmeur	1300
	20	Chef projet	2975
	20	Analyste	3000
	20	Programmeur	1100
	30	Chef projet	2850
	30	Commercial	1250
	30	Commercial	1600
	30	Commercial	1500
	30	Programmeur	950
	30	Commercial	1250

# Synthèse

## Forme générale du SELECT :

**SELECT** [ **DISTINCT** ] \* | **expr**[, **expr**...]

**FROM** **tables** -- spécifie la table ou les tables à utiliser

[ **WHERE** **condition** ] -- filtre les lignes selon une condition donnée

[ **GROUP BY** **expr** ] --forme des groupes de lignes de même valeur de colonne

[ **HAVING** **condition** ]; --filtre les groupes sujets à une certaine condition

[ **ORDER BY** **expr**| **position** [**ASC**| **DESC**] ] --spécifie l'ordre d'apparition des données dans le résultat

## Requêtes sur plusieurs tables



1. Les opérateurs de **jointures**
2. L'**imbrication** de requêtes
3. Les opérateurs **ensemblistes**

## 1. Requêtes sur plusieurs tables : Les jointures

Elles permettent de **combiner** des enregistrements issues à partir de deux ou plusieurs tables en vue de retrouver des données associées.

- 1 Equijointure (*jointure naturelle/interne*)
- 2 Auto-jointure (*jointure sur la même table*)
- 3 Jointure externe
- 4 Non-équijointure (*jointure par non égalité, théta jointure*)

# Requêtes sur plusieurs tables : Les jointures

## 1 Equijointure (*jointure naturelle*)

```
SELECT expr  
FROM table1, table 2  
WHERE table1.col1= table2.col2
```



Jointure **naturelle** entre 2 tables s'il y a **au moins** une colonne qui porte le même nom entre les 2 tables.



Jointure **interne** retourne les enregistrements quand la condition est **vrai** dans les 2 tables

Clé étrangère col2 dans la table2 qui est  
la clé primaire col1 dans la table1

tab1			col1

tab2			col2

## Equijointure ( jointure interne)

### Exemple 1 :

Liste des RDV avec le docteur 'Alaoui' :

SELECT  
FROM  
WHERE

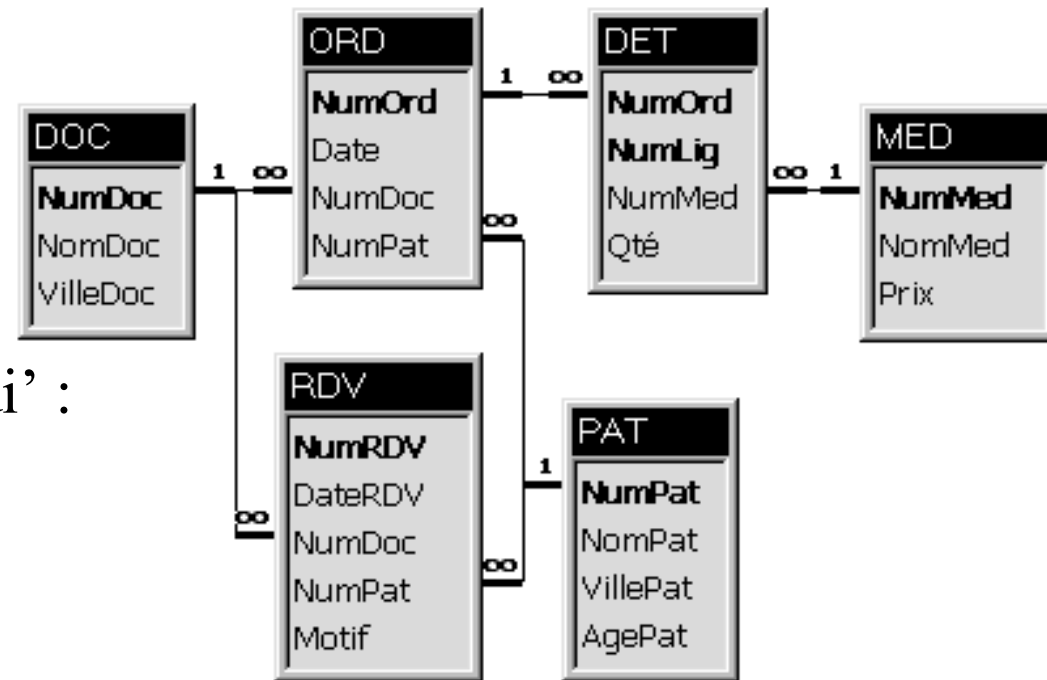
NumRDV  
RDV R, DOC D

**R.NumDoc = D.NumDoc**

**and D.NomDoc = 'Alaoui';**

Jointure pour associer deux tables

Critère de sélection



Ou bien

**SELECT** NumRDV **FROM** RDV R **INNER JOIN** DOC D  
**ON** R.NumDoc = D.NumDoc **WHERE** D.NomDoc = 'Alaoui';



## Equijointure ( jointure interne)

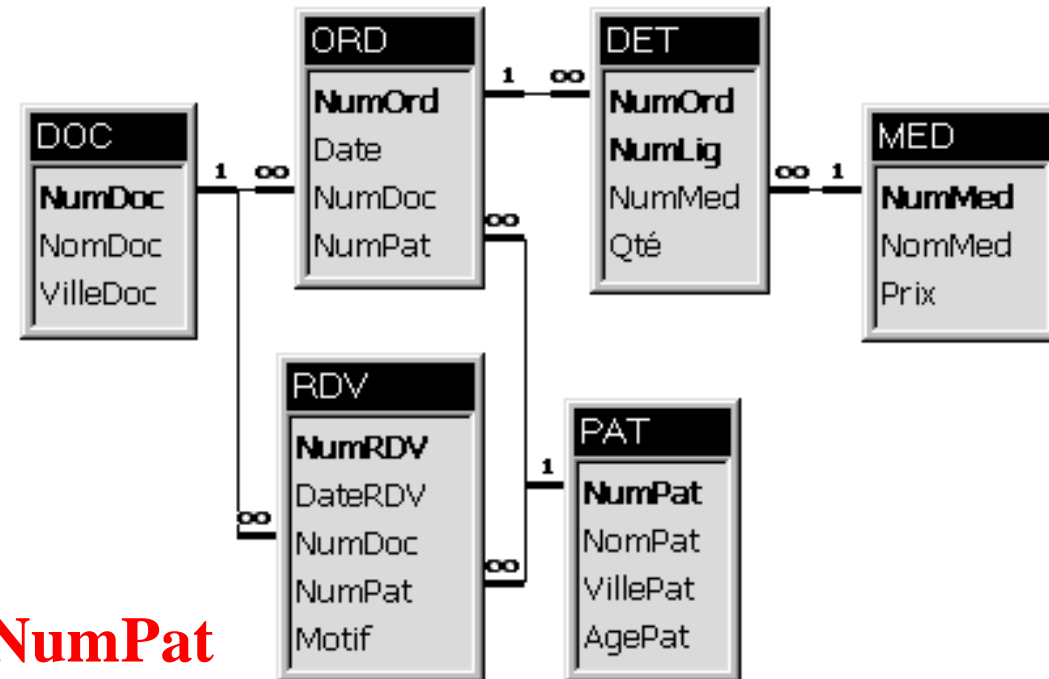
### Exemple 2 :

Liste des patients ayant un RDV avec le docteur 'Alaoui':

```
SELECT      PAT.NomPat
FROM        PAT , RDV , DOC
WHERE       PAT.NumPat = RDV.NumPat
           and RDV.NumDoc = DOC.NumDoc
           and DOC.NomDoc = 'Alaoui';
```

Ou bien

```
SELECT P.NomPat FROM PAT P INNER JOIN RDV R ON P.NumPat = R.NumPat
      INNER JOIN DOC D ON R.NumDoc = D.NumDoc
      WHERE D.NomDoc = 'Alaoui';
```




## Requêtes sur plusieurs tables : Les jointures

### 2 Auto-jointure (*jointure sur une même table*)

```
SELECT expr  
FROM table1 Alias1, table1 Alias 2  
WHERE Alias1.col1= Alias2.col1
```

tab1			col1



! L'auto-jointure implique une utilisation de la même table avec deux alias différents.

# Autojointure

## Exemple 1 :

Liste des employés ayant un salaire égale à celui de «Azhari» :

```
SELECT      E2.Nom
FROM        EMP E1, EMP E2
WHERE       E1.Sal=E2.Sal
and E1.Nom ='Azhari';
```

EMP	Deptno	Nom	Sal
	10	Alaoui	5000
	10	Filali	1500
	10	Rachidi	1250
	20	Tahiri	2975
	20	Rochdi	3000
	20	Ouazzani	1100
	30	Zohri	2850
	30	Azhari	1250
	30	Taouil	1600
	30	Rbati	1500
	30	Andaloussi	950
	30	Soussi	1250

## Autojointure

### Exemple 2 :

Liste des employés ayant un salaire  
≤ à celui de « Azhari »

```
SELECT  E2.Nom  
FROM    EMP E1, EMP E2  
WHERE   E2.Sal ≤ E1.Sal  
and E1.Nom = 'Azhari';
```

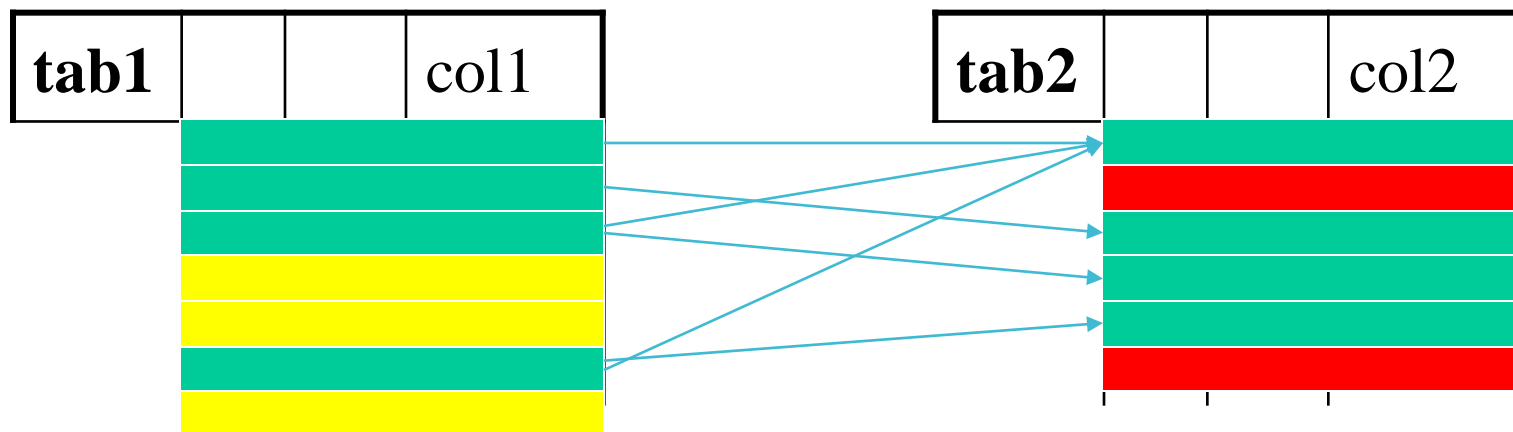
EMP	Deptno	Nom	Sal
	10	Alaoui	5000
	10	Filali	1500
	10	Rachidi	1250
	20	Tahiri	2975
	20	Rochdi	3000
	20	Ouazzani	1100
	30	Zohri	2850
	30	Azhari	1250
	30	Taouil	1600
	30	Rbati	1500
	30	Andaloussi	950
	30	Soussi	1250

# Requêtes sur plusieurs tables: La jointure externe

## 3 Jointure Externe

Les jointures externes permettent de visualiser des lignes qui ne répondent pas à la condition de jointure.

- Jointure externe
- Jointure externe à gauche
- Jointure externe à droite



 Résultats de la jointure **externe à gauche**

 Résultats de la jointure **externe à droite**

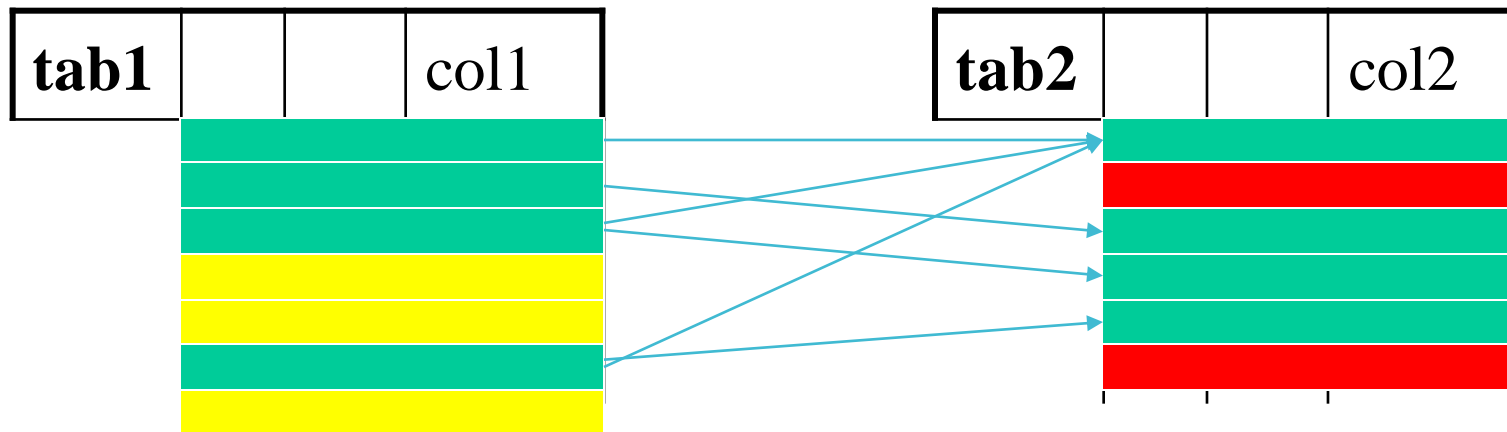
 Résultats de la jointure **externe**

# Requêtes sur plusieurs tables: La jointure externe

## Jointure Externe

```
SELECT table1.colonne, table2.colonne  
FROM table1 FULL OUTER JOIN table2  
ON table1.colonne1 = table2.colonne2 ;
```

SQL standard



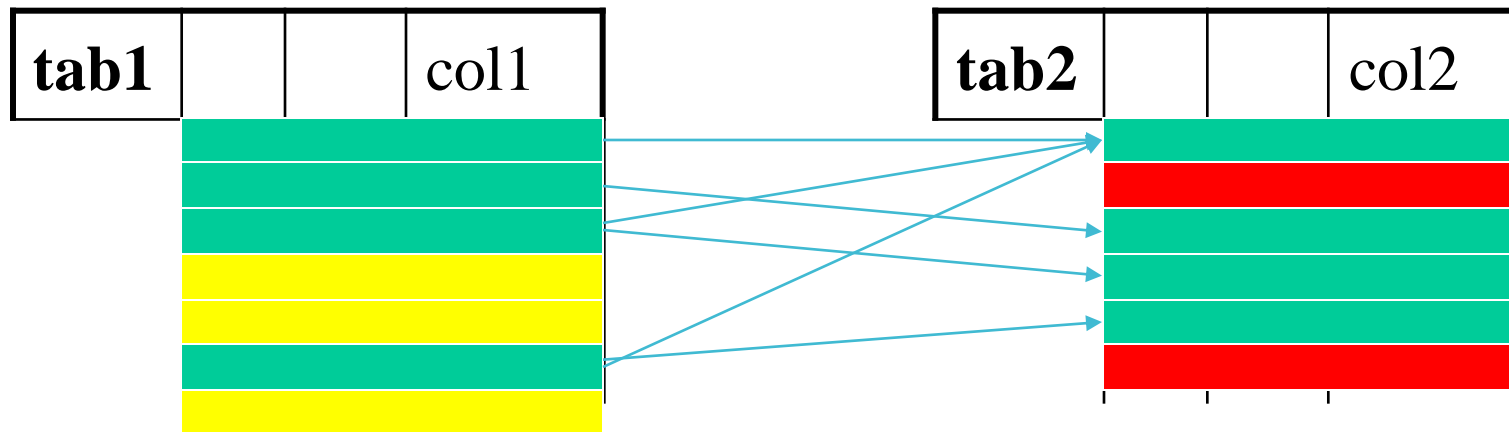
 **Résultats de la jointure externe**

# Requêtes sur plusieurs tables: La jointure externe

## Jointure Externe à gauche

```
SELECT table1.colonne, table2.colonne  
FROM table1 LEFT OUTER JOIN table2  
ON table1.colonne1 = table2.colonne2 ;
```

SQL standard



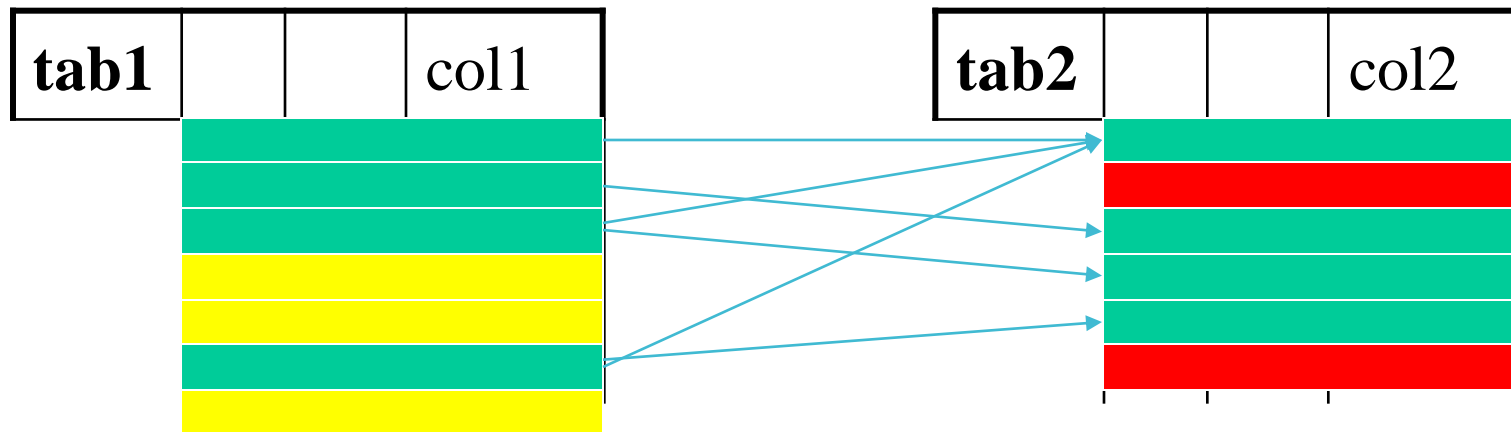
 Résultats de la jointure externe à gauche

# Requêtes sur plusieurs tables: La jointure externe

## Jointure Externe à gauche

```
SELECT table1.colonne, table2.colonne  
FROM table1, table2  
WHERE table1.colonne = table2.colonne (+);
```

SQL Oracle



 Résultats de la jointure externe à gauche

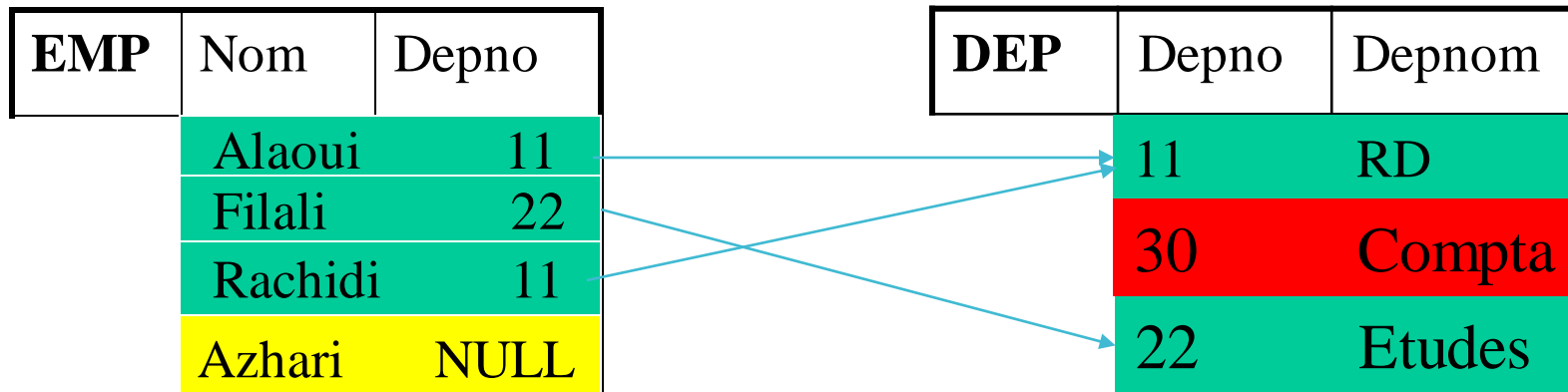


## Requêtes sur plusieurs tables: la jointure externe

### Exemple : Jointure Externe à gauche

```
SELECT EMP.Nom, DEP.Depno  
FROM EMP, DEP  
WHERE EMP.Depno = DEP.Depno (+) ;
```

SQL Oracle



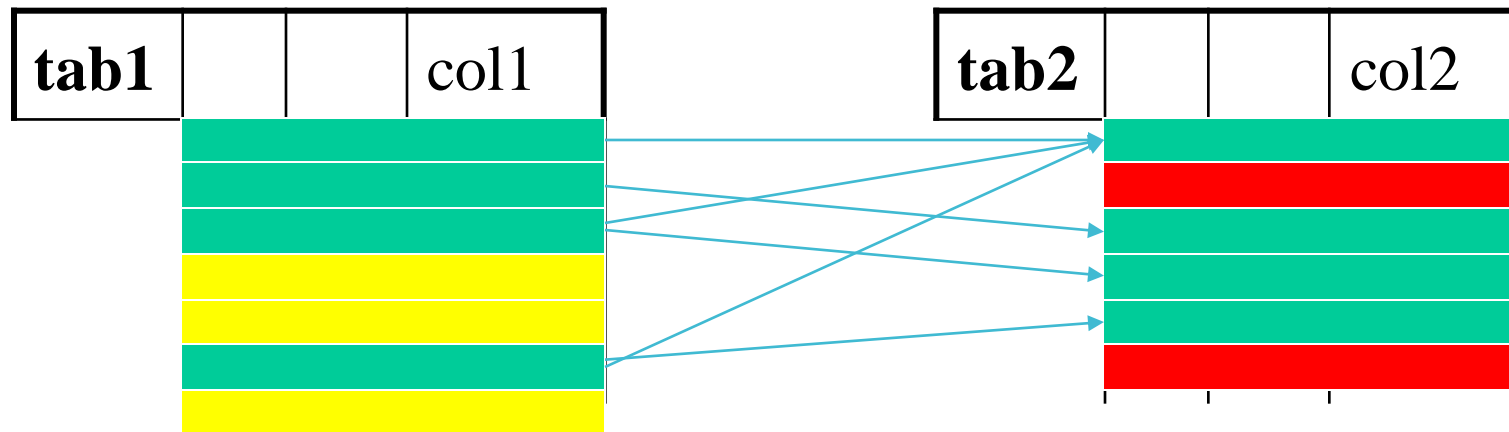
 Résultats de la jointure externe à gauche

# Requêtes sur plusieurs tables: la jointure externe

## Jointure Externe à droite

```
SELECT table1.colonne, table2.colonne  
FROM table1 RIGHT OUTER JOIN table2  
ON table1.colonne1 = table2.colonne2 ;
```

SQL standard



 Résultats de la jointure externe à droite

# Requêtes sur plusieurs tables: la jointure externe

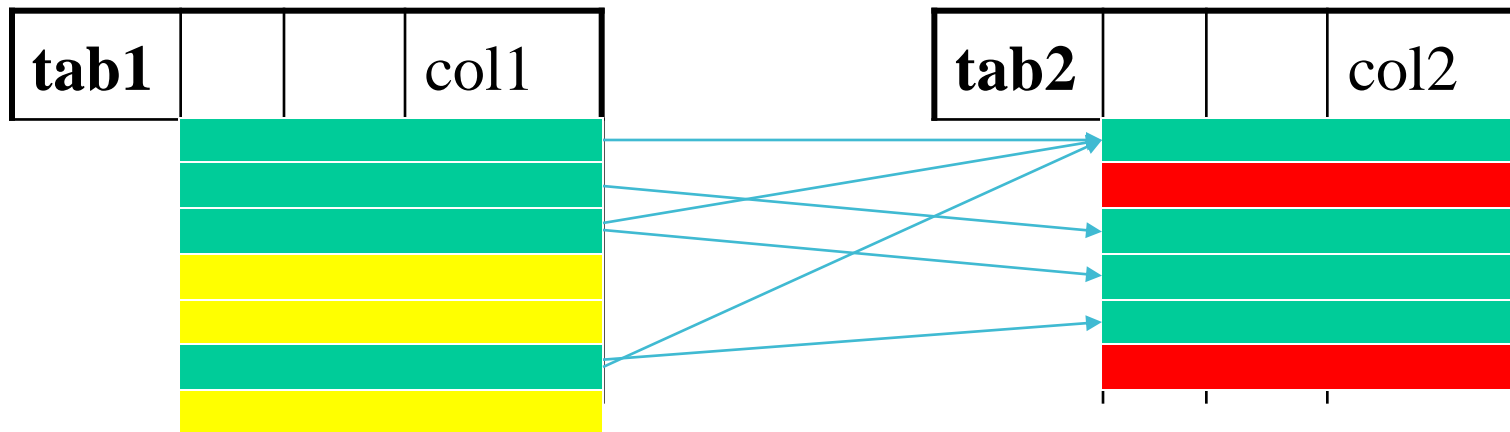
## Jointure Externe à droite

```
SELECT table1.colonne, table2.colonne
```

```
FROM table1, table2
```

```
WHERE table1.colonne (+) = table2.colonne ;
```

SQL Oracle



 **Résultats de la jointure externe à droite**

## Requêtes sur plusieurs tables: la jointure externe

### Exemple : Jointure Externe à droite

```
SELECT EMP.Nom, DEP.Depnom  
FROM EMP, DEP  
WHERE EMP.depno (+) = DEP.depno ;
```

SQL Oracle

EMP	Nom	Depno	DEP	Depno	Depom
	Alaoui	11		11	RD
	Filali	22		30	Compta
	Rachidi	11		22	Etudes
	Azhari	NULL			

 Résultats de la jointure externe à droite

## Requêtes sur plusieurs tables : Les jointures

### 4 Non Equijointure (*thêta jointure*)

La liste des employés et leurs grades :

```
SELECT      EMP.nom, SAL.gra
FROM        EMP, SAL
WHERE       EMP.salemp BETWEEN SAL.salmin and SAL.salmax
```

EMP	nom	salemp	SAL	salmin	salmax	GRA
	Alaoui	100		50	100	1
	Filali	220		101	220	2
	Rachidi	110		221	300	3
	Azhari	200		301	500	4

# Synthèse : Types des jointures SQL

Plusieurs méthodes pour associer 2 tables. Voici la liste des techniques les plus utilisées :

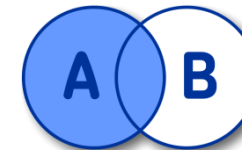
Inner Join

Left Join | Left Join (sans intersection)

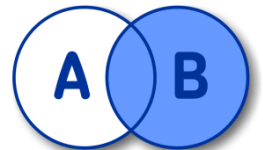
Right Join | Right Join (sans intersection)

Full Outer Join | Full Outer Join (sans intersection)

## SQL JOINS



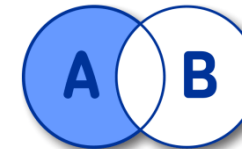
SELECT \* FROM  
A **LEFT** JOIN B  
ON A.KEY = B.KEY



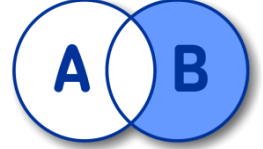
SELECT \* FROM  
A **RIGHT** JOIN B  
ON A.KEY = B.KEY



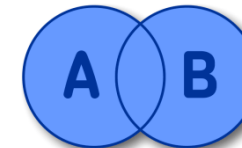
SELECT \* FROM  
A **INNER** JOIN B  
ON A.KEY = B.KEY



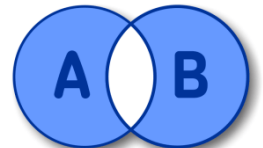
SELECT \* FROM A  
**LEFT** JOIN B  
ON A.KEY = B.KEY  
WHERE B.KEY IS NULL



SELECT \* FROM A  
**RIGHT** JOIN B  
ON A.KEY = B.KEY  
WHERE A.KEY IS NULL



SELECT \* FROM A  
**FULL OUTER** JOIN B  
ON A.KEY = B.KEY



SELECT \* FROM A  
**FULL OUTER** JOIN B ON A.KEY =  
B.KEY WHERE A.KEY IS  
NULL OR B.KEY IS NULL



*Notons que le choix d'une méthode ou d'une autre dépend du contexte et de l'énoncé fourni.*

## 2. Requêtes sur plusieurs tables : Requêtes imbriquées

- Les requêtes imbriquées sont des requêtes à l'intérieur d'autres requêtes. Ceci se produit lorsque la clause WHERE contient elle-même une **sous requête**.
- La sous requête (ou requête/bloc interne) :
  - Peut retourner une seule colonne.
  - Peut retourner des données : depuis une ou plusieurs lignes.

# Syntaxe générale

SELECT colonnes\_de\_projection  
FROM table  
WHERE expr operator (

← Requête principale

SELECT colonnes\_de\_projection  
FROM table  
WHERE .....  
);

← Sous-requête

Sens d'exécution

## REMARQUE

- Une sous-requête est **obligatoirement** entre parenthèses.
- La colonne du WHERE de la première requête est obligatoirement la colonne du SELECT de la sous-requête.
- Plusieurs opérateurs de comparaison sont possible (Ex. < , <= , > IN,...).
- Pas de ORDER BY ou UNION dans la sous requête



## Type de sous requêtes et opérateurs possibles

Type de sous requête	Opérateur
Ramène une seule ligne (une seule valeur)	=, >, >=, <, <=, <>
Ramène plusieurs lignes (plusieurs valeurs)	<b>IN</b> : appartenance <b>ALL</b> : à tous <b>ANY</b> : au moins un <b>EXISTS</b> : non vide
Plusieurs lignes avec plusieurs colonnes	<b>EXISTS</b> : non vide

## Exemples

Les noms des employés qui gagnent plus que 'Filali' ?

```
SELECT nom
FROM EMP
WHERE salemp > (SELECT salemp
                 FROM EMP
                 WHERE nom='Filali');
```

EMP	nom	salemp
	Alaoui	115
	Filali	105
	Rochdi	100
	Fatimi	200

Les employés ayant un salaire supérieur à la moyenne?

```
SELECT nom
FROM EMP
WHERE salemp > (SELECT AVG(salemp)
                 FROM EMP);
```

## Exemples

Les noms des employés qui ne sont pas les moins payés ?

```
SELECT nom
FROM EMP
WHERE salemp > ANY(SELECT salemp
                    FROM EMP);
```

EMP	nom	salemp
	Alaoui	115
	Filali	105
	Rochdi	100
	Fatimi	200

Le nom de l'employé le mieux payé?

```
SELECT nom
FROM EMP
WHERE salemp >= ALL (SELECT salemp
                    FROM EMP);
```

## Explication des Opérateurs

**IN** : la condition est vraie si EXP **appartient** à la liste des valeurs retournées par la sous-requête

**ANY** : la condition est vraie si la comparaison est vraie pour **AU MOINS** une des valeurs retournées par la sous-requête

**ALL** : la condition est vraie si la comparaison est vraie pour **TOUTES** les valeurs retournées par la sous-requête

**EXISTS** (sous-requête)  **FAUX** si  $\text{Resultat}(\text{Sous-requête}) = \emptyset$   
**VRAIE** si  $\text{Resultat}(\text{Sous-requête}) \neq \emptyset$

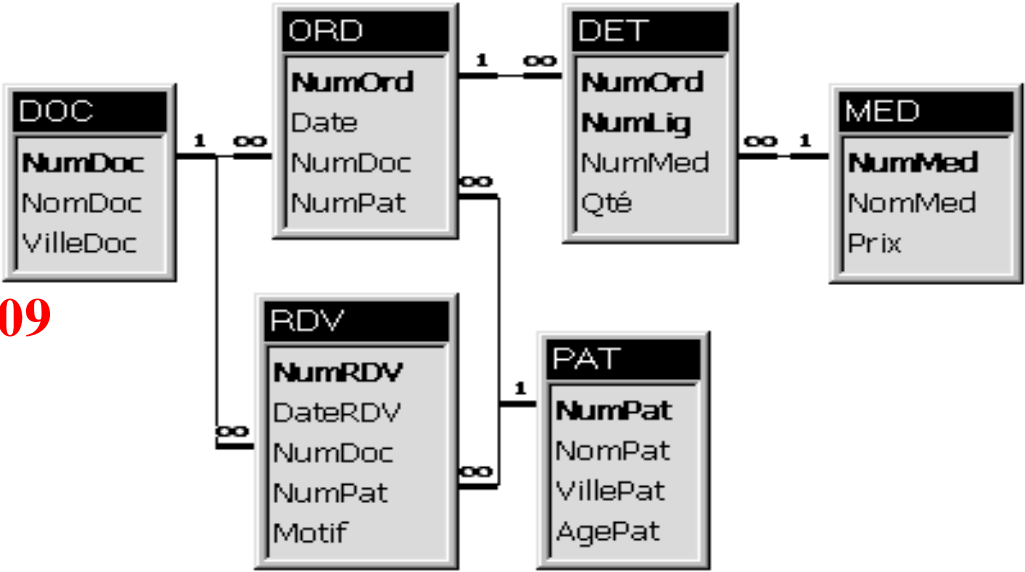
# Exemples

Q : Les docteurs n'ayant pas de RDV en 2009

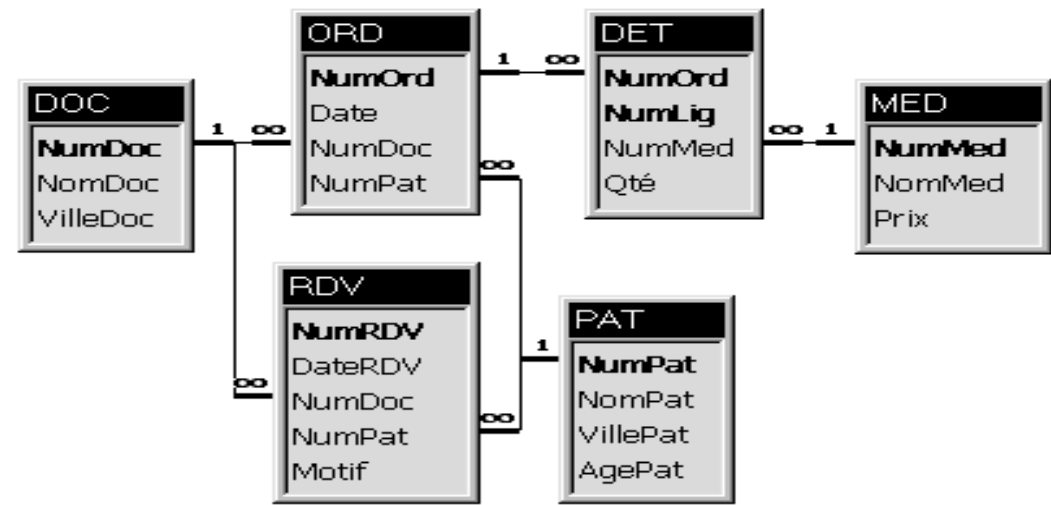
Select \* from DOC  
Where NumDOC NOT IN (select numdoc  
from RDV  
where dateRDV Between  
'01/01/2009' and '31/12/2009'  
);

Avec EXISTS :

Select \* from DOC **D**  
Where NOT EXISTS (select \*  
from RDV R  
where dateRDV Between  
'01/01/2009' and '31/12/2009'  
AND R.NumDOC=**D**.NumDoc  
);



## Exercice



**Q1 :** Les numéros d'ordonnances et leur montant total.

```
Select O.NumORD, SUM(QTE*Prix) AS "Total"
From ORD O, DET D, MED M
Where O.NumORD=D.NumORD and D.NumMED=M.NumMED
Group by O.NumORD;
```

**Q2 :** Les noms des patients ayant pris au moins un médicament de prix supérieur à 150 DH.

```
Select NomPAT
From PAT P, ORD O, DET D, MED M
Where P.NumPAT=O.NumPAT and O.NumORD=D.NumORD and
D.NumMED=M.NumMED
And Prix>=150;
```

**Q2' : Les noms des patients n'ayant pas pris un médicament de prix supérieur à 150 DH.**

```
Select NomPAT
From PAT P
Where P.NumPAT NOT IN (Select O.NumPAT From ORD O, DET D, MED M
Where O.NumORD=D.NumORD and D.NumMed=M.NumMED
And Prix>150);
```

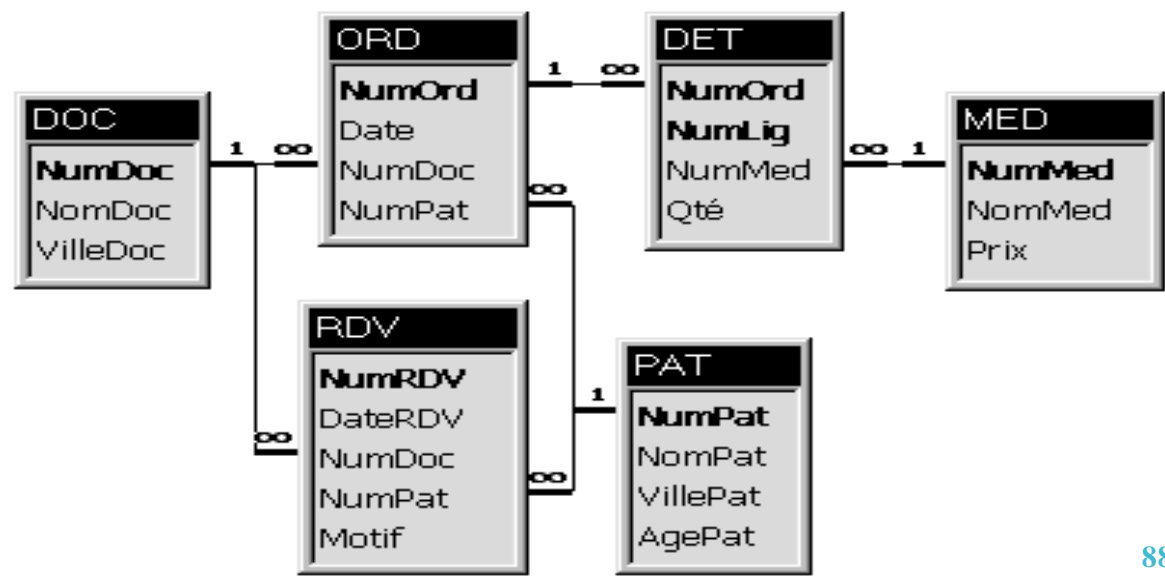


**Solution fausse :**

```
Select NomPAT From PAT P, ORD O, DET D, MED M
Where P.NumPAT=O.NumPAT and O.NumORD=D.NumORD and
D.NumMed=M.NumMED
And Prix<150 ;
```

## Exercice

- Q3 :** Le nombre de RDV par docteur en 2019.
- Q4 :** Patients sans RDV en 2019.
- Q5 :** Les patients ayant eu des RDV avec tous les docteurs.
- Q6 :** Les docteurs ayant eu des RDV avec tous les patients.
- Q7 :** Les patients ayant eu des RDV avec les mêmes docteurs que le patient N° 10.
- Q8 :** Le médicament le plus prescrit en 2019.



Voir Série TP N° 1 (SQL)



### 3. Opérateurs Ensemblistes

**INTERSECT** : Récupère les enregistrements communs à 2 requêtes.

**UNION** : Concatène les résultats de 2 requêtes ou plus (sans doublons).

**UNION ALL** : Inclut tous les enregistrements, même les doublons.

**MINUS** : Récupère les enregistrements de la première instruction sans inclure les résultats de la seconde requête.

#### Syntaxe

Requête SELECT  
<Opérateur ensembliste>  
Requête SELECT

#### Exemple

```
SELECT * FROM Table1  
UNION  
SELECT * FROM Table2;
```

## Quelques exemples

Soit les exemples de tables suivantes :

Graduates

Number	Surname	Age
7274	Robinson	37
7432	O'Malley	39
9824	Darkes	38

Managers

Number	Surname	Age
9297	O'Malley	56
7432	O'Malley	39
9824	Darkes	38

▪ Union :

Graduates  $\cup$  Managers

Number	Surname	Age
7274	Robinson	37
7432	O'Malley	39
9824	Darkes	38
9297	O'Malley	56

▪ Intersection :

Graduates  $\cap$  Managers

Number	Surname	Age
7432	O'Malley	39
9824	Darkes	38

▪ Différence :

Graduates - Managers

Number	Surname	Age
7274	Robinson	37

## Syntaxe d'insertion d'une ligne

**INSERT INTO** <nom table>

[( colonne1 [, colonne2] ... )]

{ **VALUES** (<valeur1> [, <valeur2>] ... )

| <requête SELECT > };

### Exemples :

Table: **DOC** (NumDoc, NomDoc, VilleDoc)

**Insérer une ligne en spécifiant toutes les colonnes :**

```
INSERT INTO DOC (NumDoc, NomDoc, VilleDoc) values (123, 'Filali','Fès');
```

```
INSERT INTO DOC values (123, ',', 'Fès');
```

**Insérer une ligne en spécifiant seulement les colonnes souhaitées :**

```
INSERT INTO DOC (NumDoc, NomDoc) values (444, 'Alaoui');
```

**Insérer à partir d'une autre table**

```
INSERT INTO DOC select * from Tabledesmedecins;
```

## Syntaxe de Mise à jour d'une ligne

**UPDATE** <nom table>

**SET** <colonne> = valeur

[, <colonne> = valeur ] ...

[**WHERE** <condition de modification> ];

### Exemples:

Table: **DOC** (NumDoc, NomDoc, VilleDoc)

**UPDATE** DOC

**SET** NomDoc='Tati', NomVille='VilleàTati'

**Where** NumDoc = 444;

**UPDATE** DOC

**SET** VilleDoc = Null;

## Syntaxe de Suppression d'une ligne

**DELETE FROM** <nom table>

[**WHERE** <condition>]

### Exemples:

Table: **DOC** (NumDoc, NomDoc, VilleDoc)

**DELETE FROM DOC WHERE** NumDoc = 444;

**DELETE FROM DOC**

**WHERE** NomDoc IN (   Select NomDoc  
                          from DOC  
                          **WHERE** VilleDoc = NULL  
                          );

# Fin

## Partie 1



**Voir Série TP N° 1 (SQL)**