



# Bases de données

Cours Première année – Ingénieur

Prof. M. NASSAR

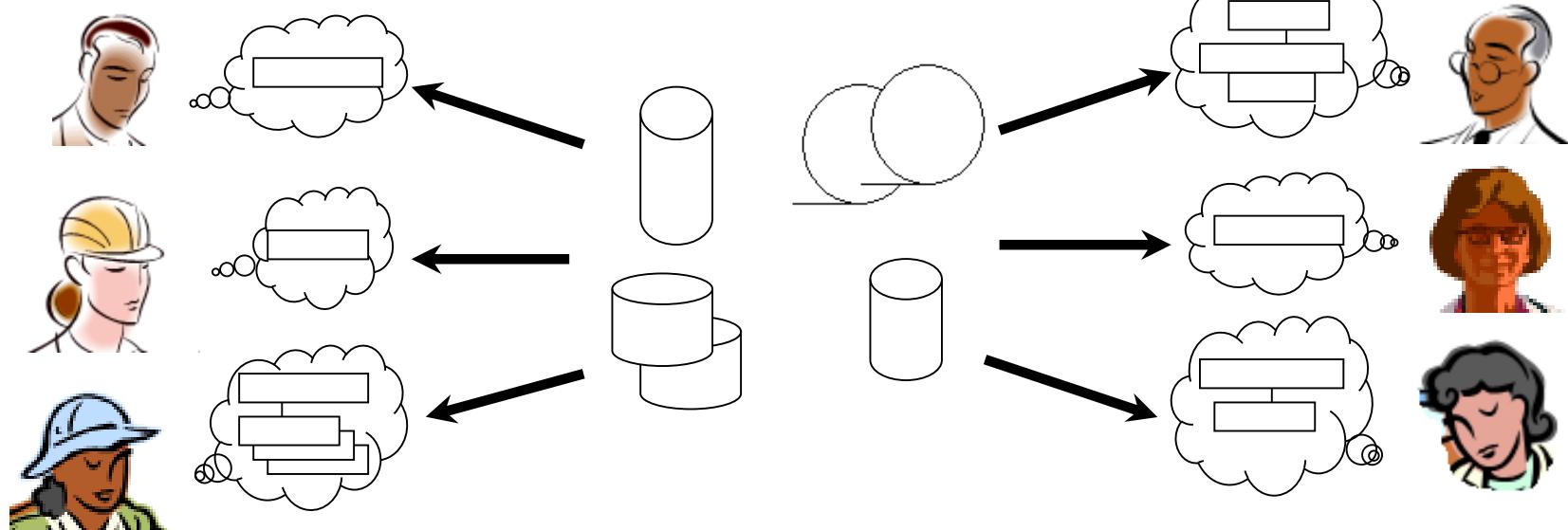
Ecole Nationale Supérieure d'Informatique et d'Analyse  
des Systèmes  
- Rabat -

# **Plan**

- 1.** Introduction aux Bases de données
- 2.** Modèle relationnel et l'Algèbre relationnelle
- 3.** Langage SQL
- 4.** Conception de bases de données

# Introduction aux Bases de données

## SYSTEME D'INFORMATION



Environnement basé sur un système de fichiers

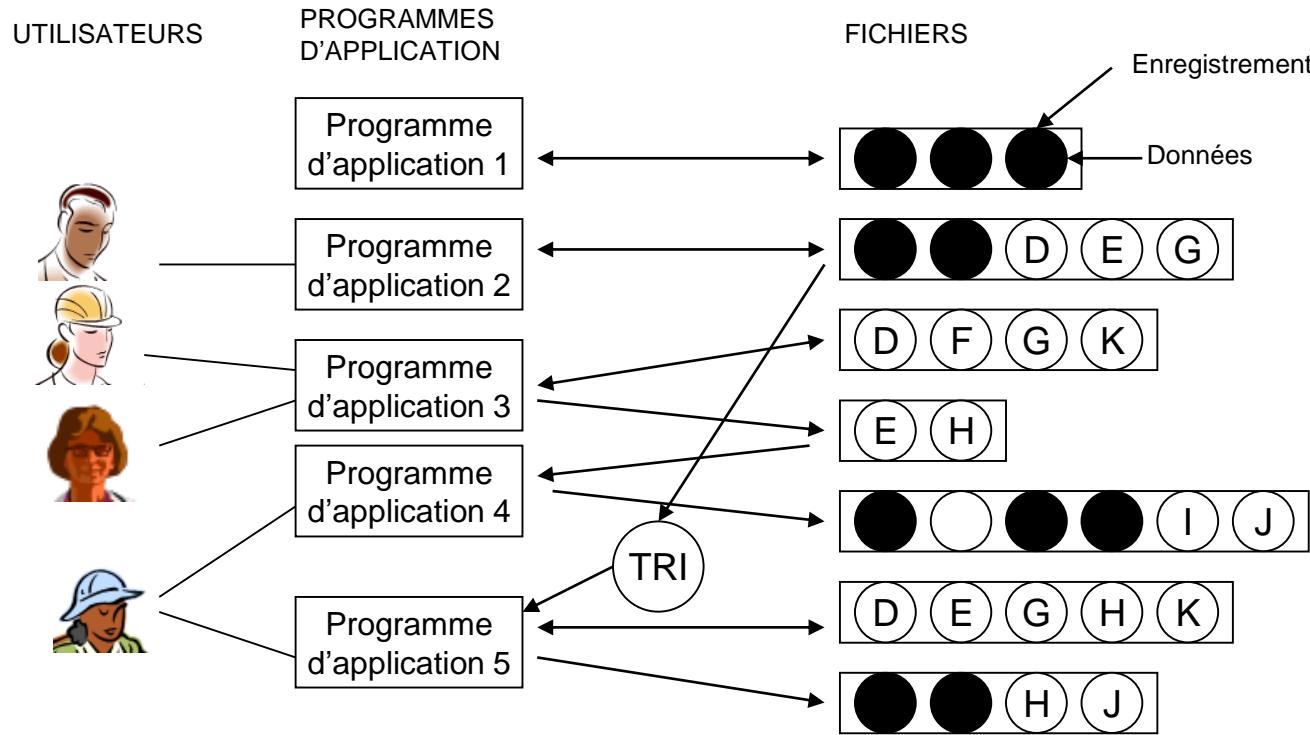
- plusieurs fichiers
- plusieurs programmes

créés en différentes dates, avec éventuellement  
différents langages

# I N C O N V E N I E N T S

## Redondance et inconsistance

même information répliquée dans plusieurs fichiers



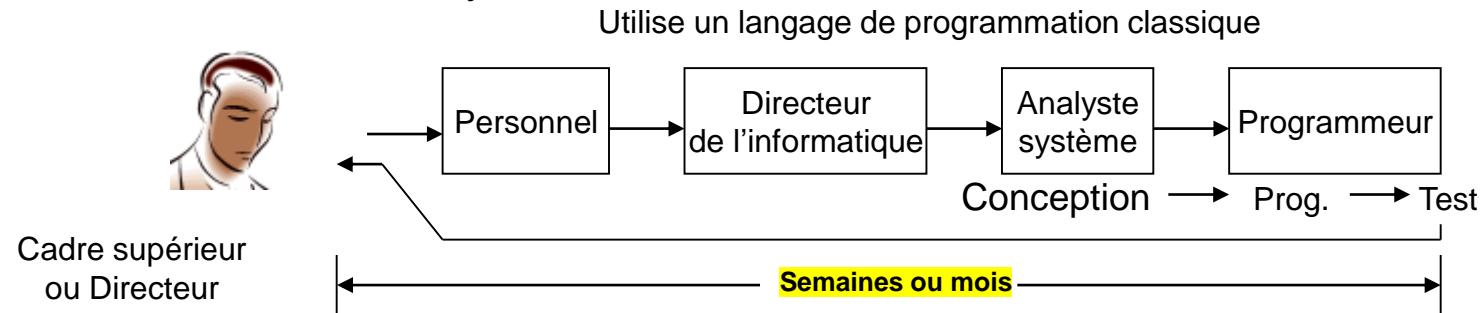
Pour chaque nouvelle application, un programmeur crée un nouveau fichier. Une grande installation à des centaines ou milliers de fichiers avec une grande redondance des données.



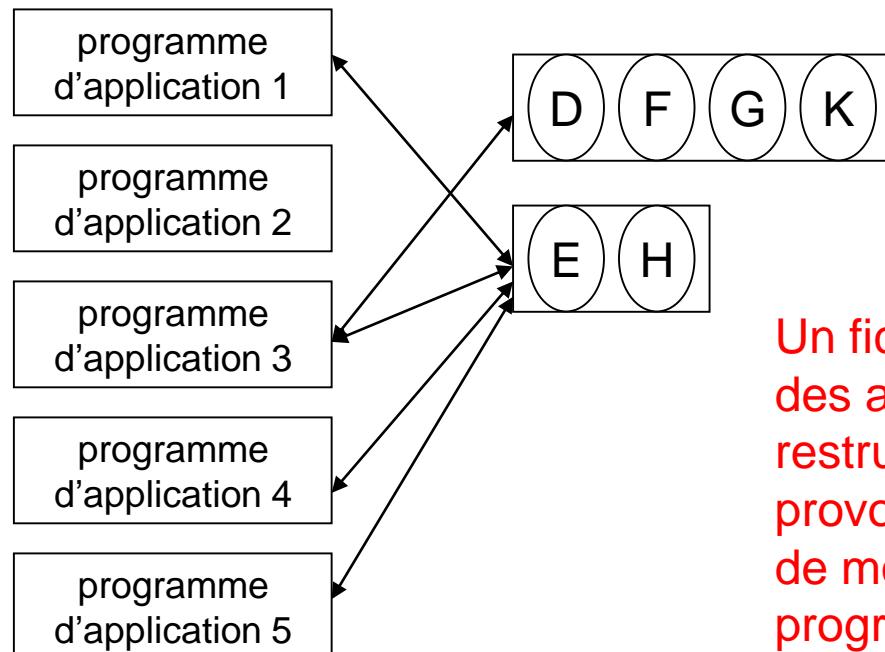
- Accroissement du volume global des données
- Risque d'inconsistance des données : informations sur plusieurs fichiers, non « rafraîchies » au même moment.

## Difficultés d'accès aux données

Les besoins occasionnels ne peuvent être satisfaits de façon efficace, ce qui nuit donc à l'efficacité du système,



## Maintenance



Un fichier finit par être utilisé par des applications multiples. Une restructuration de ce fichier provoquera une réaction en chaîne de modifications dans d'autres programmes d'applications.

## Difficultés de partage de données

Supposons un environnement multi-utilisateurs,

**Utilisateur-1**

compte=500

Lire compte

compte:=compte-50

Écrire compte

—

**Utilisateur-2**

t1

t2

t3

t4

t5

—

Lire compte

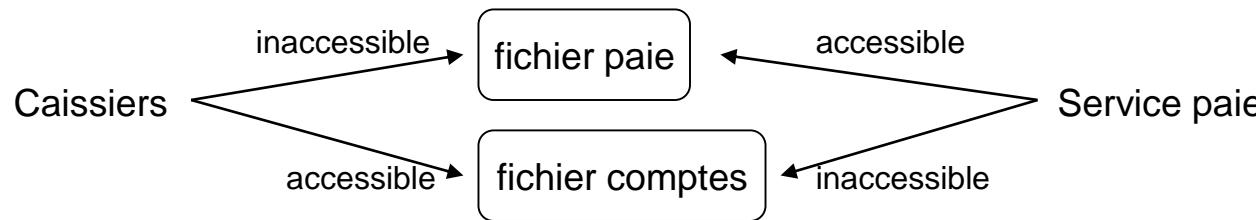
compte:=compte-50

—

Ecrire compte

## Problèmes de sécurité

Les données stockées sur fichiers doivent être protégées contre les accès non autorisés,



## Problèmes d'intégrité

Les données doivent saisir certaines contraintes

Exemples :

- Solde  $\geq 25$

- Un client référencé dans fichier commandes doit figurer d'abord dans fichier clients

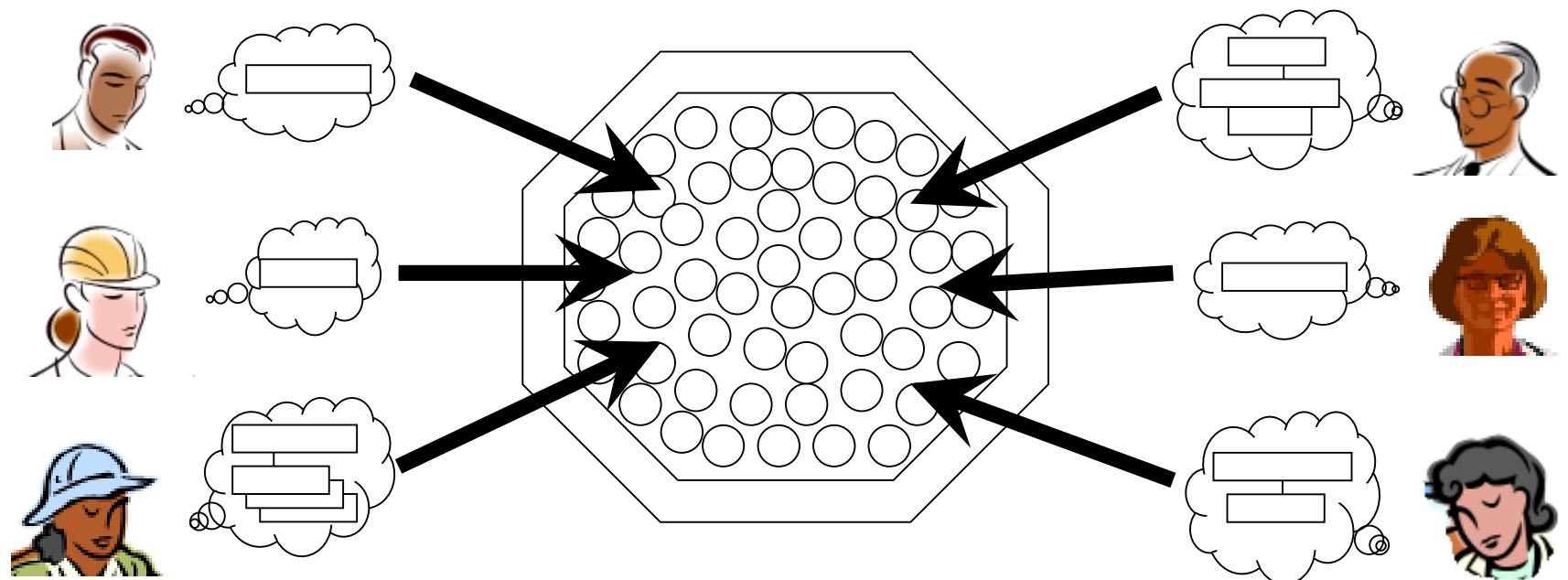
Nécessite donc de modules spécifiques chargés de respecter les contraintes au sein des diverses applications

## BASE DE DONNEES

Une base de données est une collection cohérente de données reliées entre elles, stockées ensemble, aussi peu redondantes que possible pour être utilisées par une ou plusieurs applications d'une façon optimale,

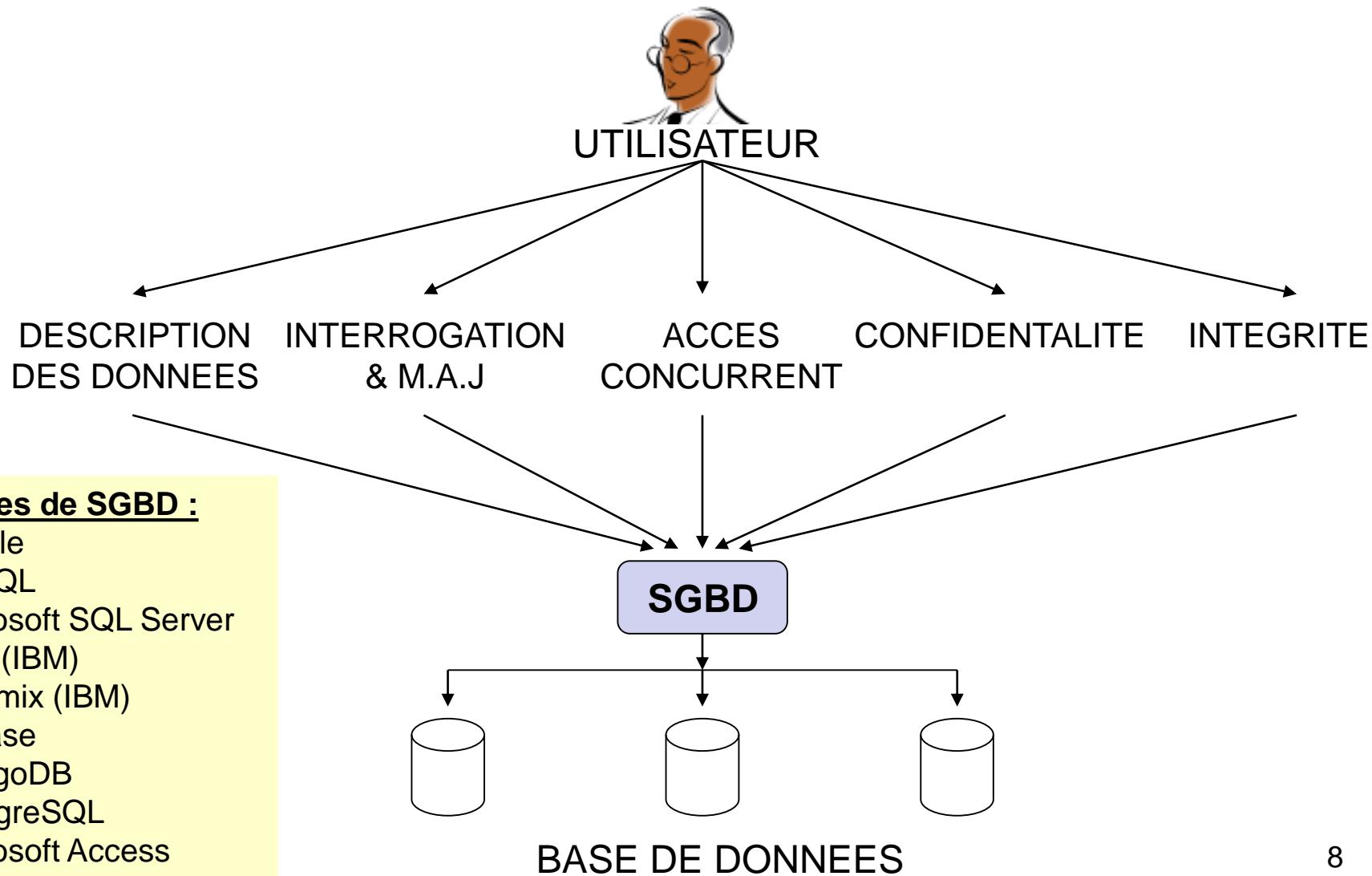
Les données sont stockées de façon à être indépendantes des programmes qui les utilisent,

JAMES MARTIN



# SYSTEME DE GESTION DE BASE DE DONNEES

Logiciel qui gère un ensemble de fichiers qui constituent la BD et qui permet à plusieurs utilisateurs de stocker et d'extraire des données de ces fichiers, dans des conditions d'intégrité et de confidentialité,

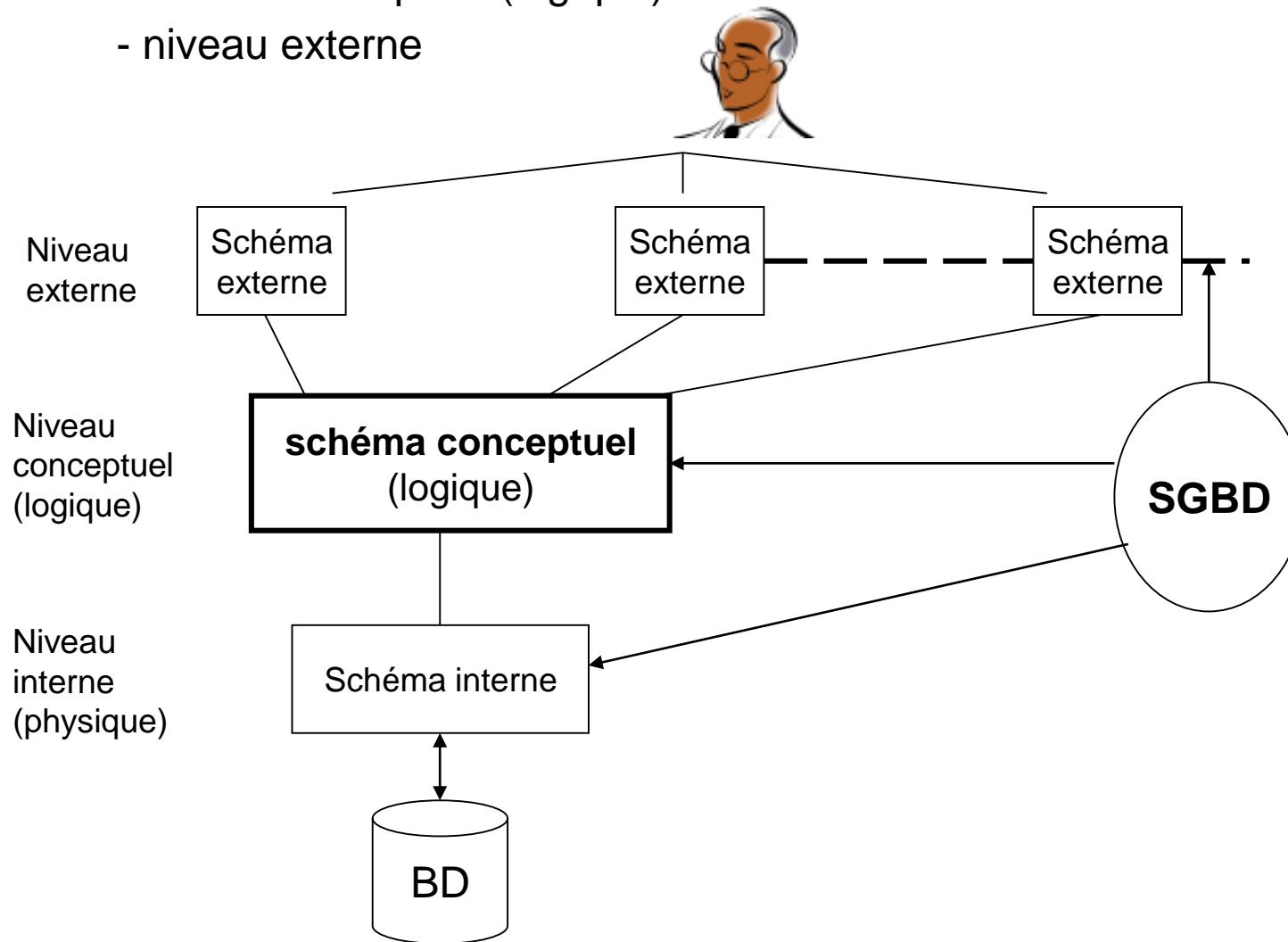


## ARCHITECTURE D'UNE BASE DE DONNEES

Le SGBD présente la BD sous trois niveaux

(niveaux de description de données : architecture ANSI / SPARC) :

- niveau interne (physique)
- niveau conceptuel (logique)
- niveau externe



## Niveau conceptuel

La description des données porte ici essentiellement sur l'aspect sémantique :

- le contenu global de la BD (les entités)
- les liens entre les données (les relations)
- les règles de contrôle, de sécurité et d'intégrité,

## Niveau interne

Il décrit la structure physique des données : mode de stockage de ces données sur support externe et la correspondance avec la structure logique :

- description de l'enregistrement interne
- spécification des index, pointeurs,...
- séquence physique des enregistrements

## Niveau externe

Correspond aux différentes vues que vont avoir les utilisateurs sur la BD.

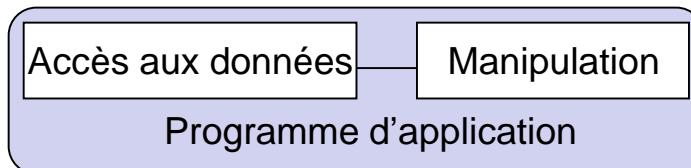
Les utilisateurs n'étant pas concernés par toutes les données de la BD, un SGBD présente les données qu'il gère sous plusieurs vues différentes, simplifiant ainsi l'interaction utilisateur- BD.

## O B J E C T I F   D E S   S G B D

L'objectif essentiel des SGBD est de simplifier et faciliter l'exploitation des données tout en assurant un degré de performances satisfaisant :

### **-Indépendance programmes / données**

La manipulation des données et leur organisation sur support externe sont deux aspects indépendants,



### **-Indépendance physique**

- indépendance par rapport aux mécanismes de stockage de bas niveau (spécification blocs, cylindres, etc...)
- indépendance par rapport à la composition et à la séquence des champs d'un enregistrement : permettre de changer le schéma physique sans affecter le schéma logique et les programmes d'exploitation

### **-Indépendance logique**

- indépendance par rapport à l'architecture logique de la BD, quand elle doit évoluer (ajout de champs ou d'enregistrements)

**Abstraction des données:** masquer les détails de représentation machine pour faciliter le raisonnement au niveau global ,

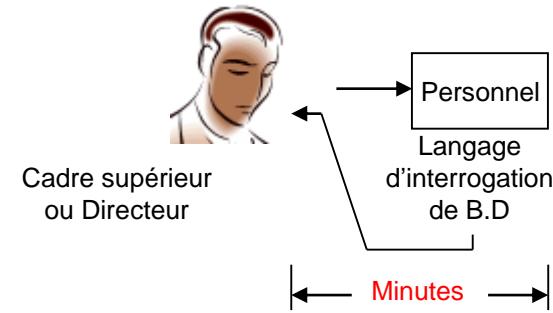
**Autonomie des données** au niveau physique et logique.

## **-Accès par des langages de haut niveau**

(indépendance programme - méthode d'accès)

Un langage de manipulation permettant de :

- sélectionner
- insérer
- modifier
- supprimer



des données définies, sans spécifier le schéma d'accès à ces données.

## **-Intégrité des données**

Les données doivent satisfaire certaines contraintes de cohérence (intégrité).

Ex :  $\text{solde} \geq 25$

$\text{heures\_travail} \leq 208$

Ces contraintes seront spécifiées, le SGBD les contrôlera lors des MAJ des données.

## **-Sécurité des données**

L'accès aux données doit rester sélectif.

Le SGBD fournit les outils assurant cette sélectivité (autorisations d'accès).

## **-Traitements concurrents (simultanés)**

La partageabilité des données peut affecter la cohérence de ces données. Le SGBD doit assurer cette cohérence en étalant ces traitements concurrents dans le temps (verrouillage par exemple).

## **-Récupération en cas d'avarie**

une panne : - destruction du support

- coupure de secteur
- erreur d'exécution du logiciel

peut entraîner la destruction des données en cours de traitement dans l'unité centrale.

Le SGBD dispose d'outils de récupération (transactions, journaux) qui permettent de restaurer les données dans leur état antérieur à l'avarie.

# M O D E L E S D E S D O N N E E S

## Modèle

Ensemble d'outils conceptuels décrivant :

- les données
- les relations entre les données
- les contraintes auxquelles elles sont soumises

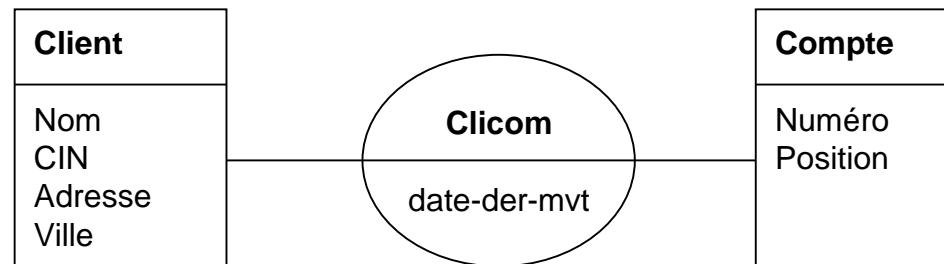
Types des modèles :

- entités-relations
- relationnel
- réseau
- hiérarchique

## Modèle Entités-Relations (Entités-Associations)

Perception du monde réel sous forme de collections d'objets appelés "Entités" unis par des relations,

Exemple :



**Entités** : classes d'objets, définies par un ensemble d'attributs qui les décrivent

**Relations** : associe plusieurs entités

**Contraintes**: cardinalité minimale et maximale d'une relation pour une entité donnée

## Modèle relationnel

Les données et les relations qui les lient sont structurées sous forme de tables comportant des colonnes désignées par des noms uniques

NOM	CIN	ADRESSE	VILLE
Triki	A32144	2,Rue zerhoune	Rabat
Alami	B34325	5,Rue oujda	Rabat
Badi	A43231	70,Rue sebou	Casa
Radi	B56432	2,Av. My Youssef	Rabat

NUMERO	POSITION
900	00550
556	10000
647	12500
801	10533

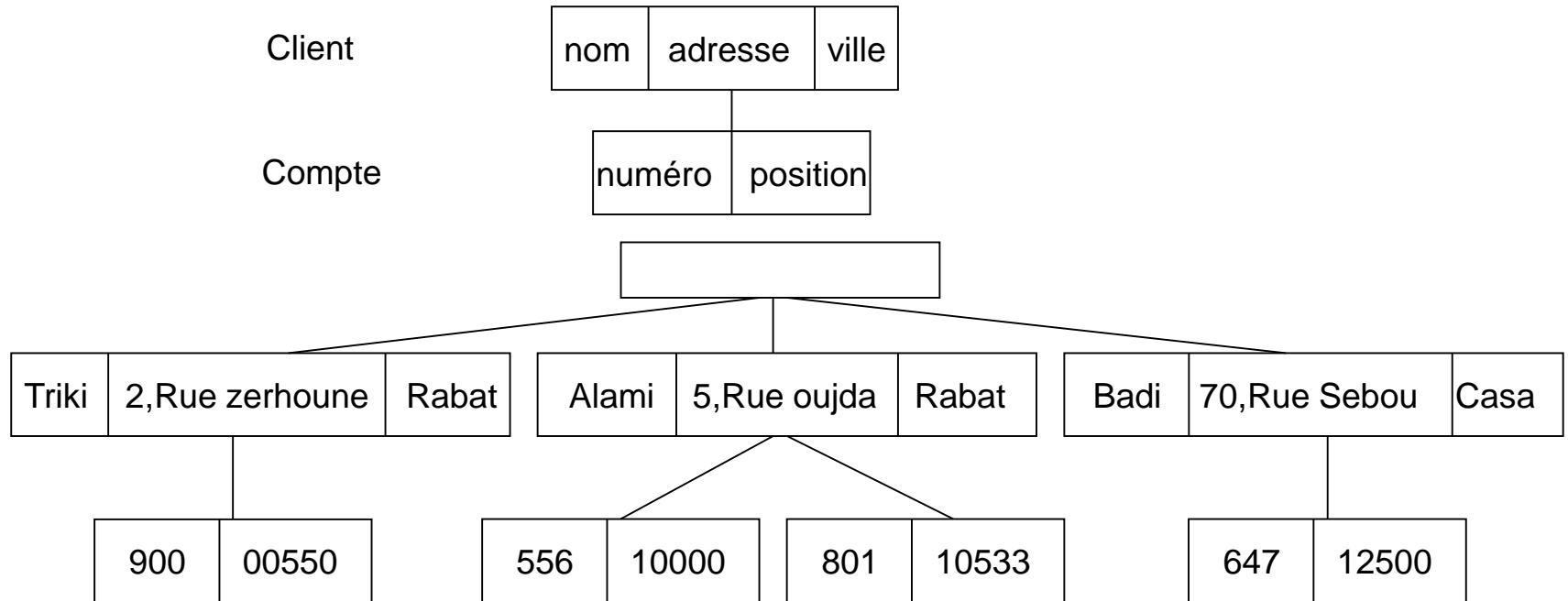
CIN	NUMERO	DATE- DER- MVT
A32144	900	
B34325	556	
B34325	801	
A43231	647	
B56432	647	

- ALAMI détient deux comptes
- BADI et RADI se partagent le même compte

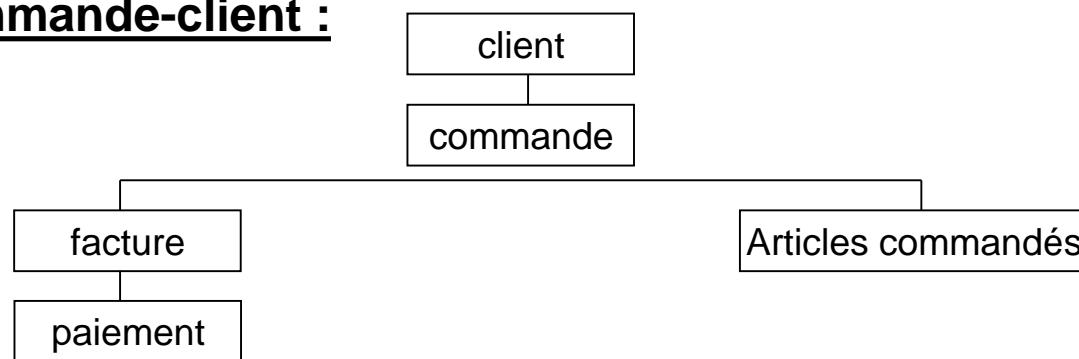
## Modèle hiérarchique

Les données et les relations sont représentées par des ensembles d'enregistrements et des liens (pointeurs) formant une structure arborescente

### Exemple client-compte :



### Ex. commande-client :

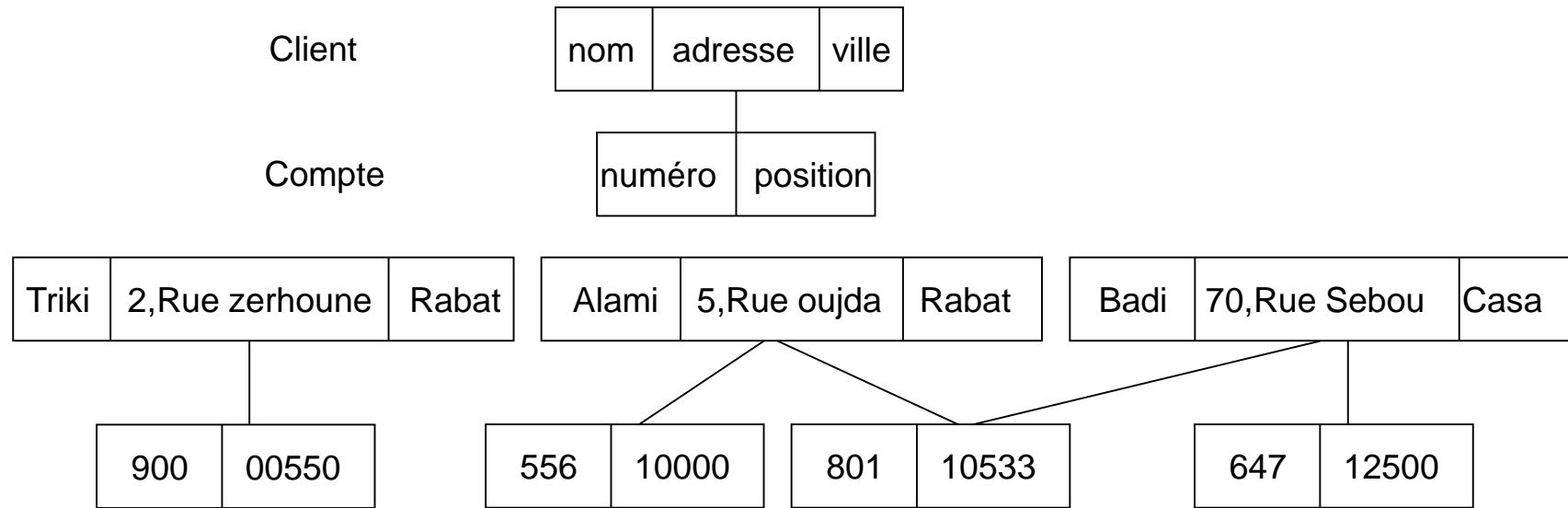


## Modèle réseau

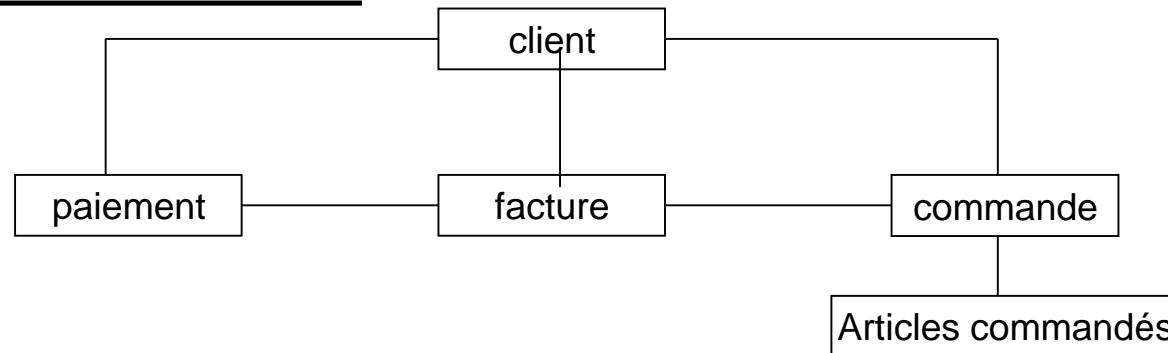
Les données, organisées en réseau, sont représentées par un ensemble d'enregistrements, tandis que les relations sont représentées par des liens (pointeurs)

La BD est stockée sous forme d'une collection de graphes

### Ex. client-compte :



### Ex. commande-client :



## LANGAGE DE DEFINITION DES DONNEES (DDL)

Permet la définition de l'architecture de la BD



## LANGAGE DE MANIPULATION DES DONNEES (DML)

Comprend des opérations :

- d'extraction de données
- d'insertion de nouvelles données
- de remplacement ou d'effacement de données périmées

## LES UTILISATEURS DE LA BASE DE DONNEES

**Des programmeurs d'applications:** utilisant la BD par des instructions du DML intégrées dans un programme écrit en :

- langage hôte : Cobol, Java, C,
- langage de 4<sup>ème</sup> génération : Informix, Oracle, Ingres, ...interprétés ou compilés,

**Des utilisateurs occasionnels:** des personnes informées qui peuvent :

- extraire les données voulues, par sélection, à l'aide d'un langage d'interrogation de la BD
- développer des applications occasionnelles ou des modules en utilisant un L4G adapté

**Des exploitants:** les personnes exploitant la BD en utilisant les programmes d'application : caissier, guichier

## DICTIONNAIRE DE DONNEES

Description centrale de la structure des données stockées dans la BD.

Utilisé à titre informationnel par les modules du SGBD pour savoir où et comment sont stockées les informations.

### **Objet principal**

- masquer les détails de stockage physique
- assurer la sécurité des données
- assurer la consistance au niveau du format de stockage

### **Il décrit :**

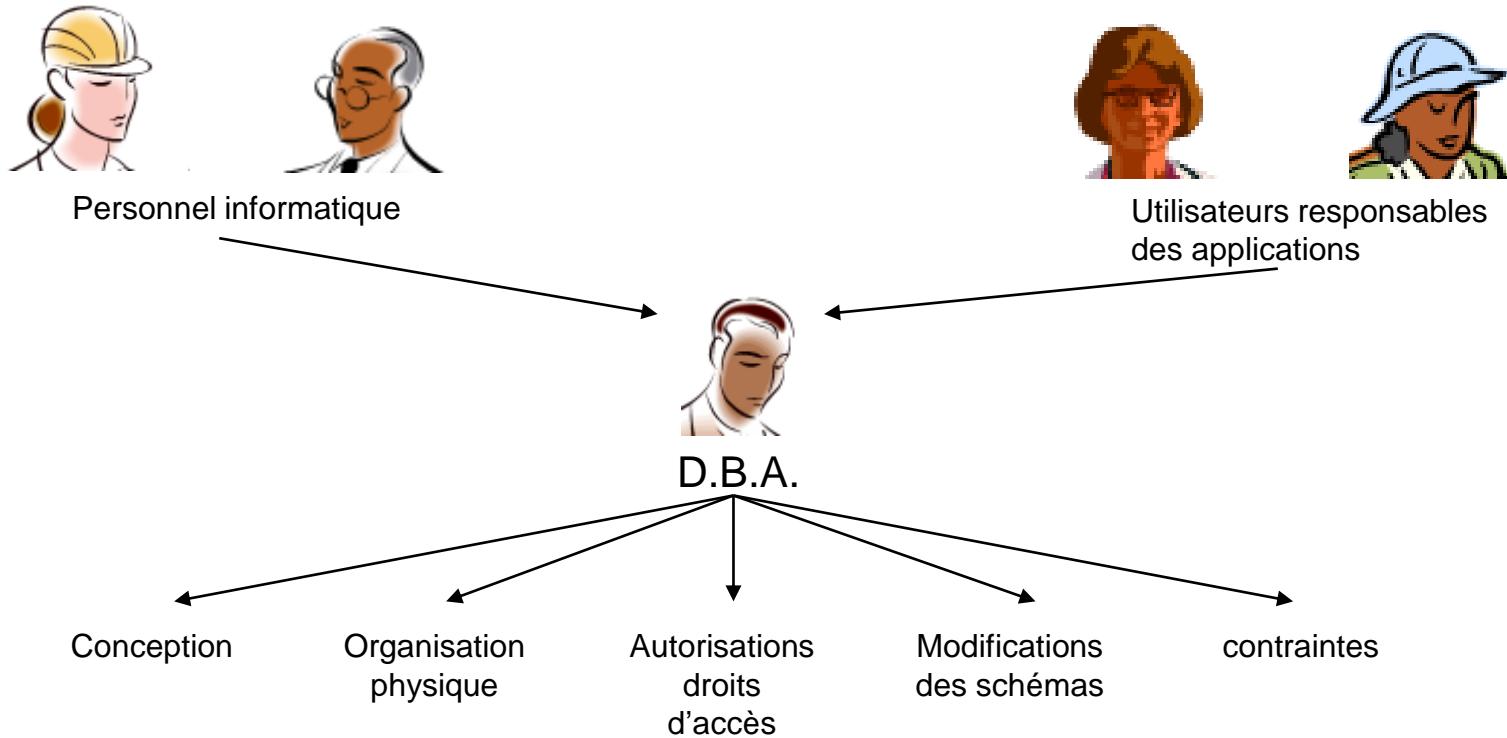
#### **- Par table :**

- l'accessibilité (utilisateurs et opérations autorisées)
- fichier de données, taille enregistrement
- volumes disques utilisés (BD importantes)
- les champs clés - Indexes

#### **- Par champs :**

- le type de données
- l'accessibilité (utilisateurs et opérations autorisées)
- dans certains cas, des restrictions sur la valeur :
  - ne doit pas être nulle
  - doit être unique

# ADMINISTRATEUR DE LA BASE DE DONNEES

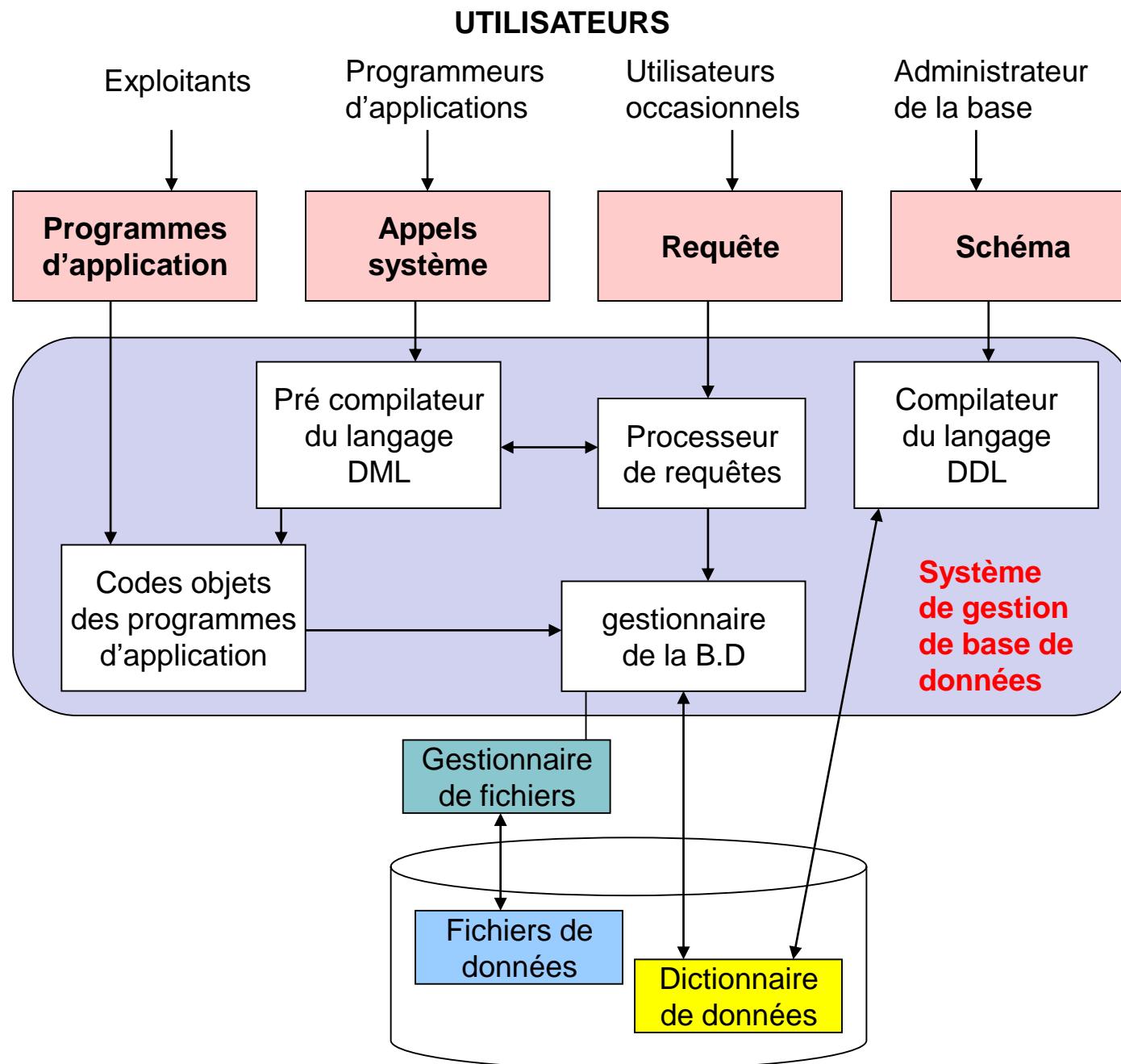


## (DBA : DATA BASE ADMINISTRATOR)

**Responsable de l'ensemble du système, ses fonctions couvrent essentiellement :**

- la définition du schéma conceptuel de la B.D (contenu de la B.D) et sa création
- la définition des structures des stockages et des méthodes d'accès
- la modification (adaptation) du schéma conceptuel et l'amélioration de l'organisation physique de la BD
- définition, en commun avec les utilisateurs, du schéma externe
- gestion des autorisations d'accès
- spécification des contraintes d'intégrité.

# STRUCTURE GLOBALE D'UN SGBD



# MODELE RELATIONNEL

Modèle créé par Edgar Frank CODD au laboratoire d'IBM de SAN- JOSE EN 1970. Il comporte :

- concepts pour la description
- concepts pour la manipulation
- concepts additionnels

## CONCEPTS DESCRIPTIFS

**Notion 1 :** DOMAINE : Ensemble de valeurs

Exemples :

Entier

Réel

Chaîne

Booléen

Salaire={400 . . 15000}

Ville={casa, marrakech, rabat, . . .}

Nom \_agence={Bd Mohammed V, My Youssef, . . .}

## Notion 2 : RELATION

soit le tableau Dépôt :

Agence	compte	client	position
Bd. Mohammed V	101	ALAMI	500
Murs sultan	205	TRIKI	700
My Youssef	102	RADI	400
Bd. Mohammed V	305	BADI	500

Il liste 4 colonnes (attributs) :

Agence  $\in$  nom\_agence (D1)

Compte  $\in$  entier (D2)

Client  $\in$  chaîne (D3)

Position  $\in$  réel (D4)

Dépôt est un sous- ensemble de l'ensemble des quadruplets :

(a1, a2, a3, a4) où a1  $\in$  D1, a2  $\in$  D2, a3  $\in$  D3, a4  $\in$  D4

Résultant du produit cartésien : D1xD2xD3xD4

## Produit cartésien

Le produit cartésien de D1, D2, D3, ..., Dn noté :

$D1 \times D2 \times D3 \times \dots \times Dn$  ou  $\prod_{i=1}^n D_i$

est l'ensemble des tuples :  $\langle a_1, a_2, a_3, \dots, a_n \rangle$  tel que  $a_i \in D_i$

### Exemple :

$D1 = \{i10, i20, i40, ix35\}$      $D1 \times D2 =$

$i10$	Essence
$i10$	Diesel
$i20$	Essence
$i20$	Diesel
$i40$	Essence
$i40$	Diesel
$ix35$	Essence
$ix35$	Diesel

**Relation :** sous ensemble du produit cartésien d'un ensemble de domaines ou simplement :

Table à 2 dimensions où :

- une ligne est un TUPLE

- chaque colonne est repérée par un nom : ATTRIBUT, prenant sa valeur dans un domaine

**Exemple :** - Dépôt : relation à 4 attributs, 4 tuples

**Une relation est définie par son schéma, spécifiant :**

- son nom
- **sa clé primaire** : groupe d'attributs minimum déterminant chaque tuple de façon unique dans une relation
- **sa liste d'attributs et leurs domaines respectifs,**

**Exemples :**

Dépôt= (agence chaîne,  
          compte entier,  
          client chaîne,  
          position réel)

Qu'on notera :

Dépôt = (agence, compte, client, position)

Banque = (agence, avoir, ville)

Clientèle = (client, adresse, ville)

Crédit = (agence, prêt, client, montant)

## Cle étrangère

Commande = (code\_Commande, **code\_Client**, date\_commande) . . . R1

Client = (code\_Client, Nom) . . . R2

code\_Client, dans R1, est clé étrangère :

attribut de R1 apparaissant comme clé primaire dans R2,

- R1, R2 : deux relations,

- A={a1, a2, . . . , an} inclus dans R1, est clé étrangère dans R1 si A n'est pas une clé primaire de R1 et A est clé primaire de R2

## Contraintes d'intégrité référentielles

définies par des clés étrangères :

- lors d'une insertion dans R1, les valeurs de A doivent exister dans R2

- lors d'une suppression dans R2, les tuples correspondants de R1 doivent disparaître

## Dictionnaire de données

La description d'une BD relationnelle est stockée dans une BD relationnelle : une métabase appelée : "Dictionnaire de données",

## CONCEPTS MANIPULATOIRES

Langages formels de consultation, utilisés pour la formulation de demandes d'informations dans une base de données,

2 types de langages :

- **langages procéduraux :**

l'utilisateur définit la méthode de recherche des informations désirées -Algèbre relationnelle-

- **langage non procéduraux :**

l'utilisateur définit l'information désirée, la base se charge de la procédure de recherche  
-calcul relationnel-

La plupart des langages des BD commercialisées combinent ces deux aspects

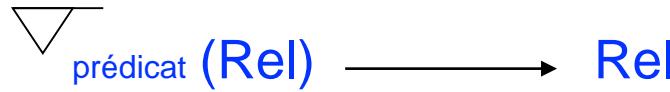
### Algèbre relationnelle

Langage de consultation dont les opérations fondamentales :

- sélection
- projection
- produit cartésien
- union
- différence

et d'autres opérations : **intersection**, **Division** et **jointure**, permettent d'exprimer toutes les requêtes sous forme d'expressions algébriques

## La sélection



Sélectionne les tuples de Rel satisfaisant prédicat

Exemple :

 agence="Bd. Mohamed V"

Agence	compte	client	position
Bd. Mohamed V	101	ALAMI	500
Murs sultan	205	TRIKI	700
My Youssef	102	RADI	400
Bd. Mohamed V	305	BADI	500

=

Agence	compte	client	position
Bd. Mohamed V	101	ALAMI	500
Bd. Mohamed V	305	BADI	500

 position>500 (dépôt)

 agence="Bd. Mohamed V" ^ position>500 (dépôt)

**Prédicat :** condition faisant intervenir attributs et constantes et utilisant des opérateurs de comparaison. Les conditions peuvent être composées à l'aide des opérateurs **^**, **v** (et, ou)

## Projection

$\pi_{a_1, a_2, \dots, a_n} (\text{Rel}) \longrightarrow \text{Rel}$

Elimine certains attributs et supprime les tuples en double

Exemple :

$\pi$

agence, client

Agence	compte	client	position
Bd. Mohamed V	101	ALAMI	500
Murs sultan	205	TRIKI	700
My Youssef	102	RADI	400
Bd. Mohamed V	305	BADI	500
Bd. Mohamed V	405	ALAMI	1.000.000

=

Agence	client
Bd. Mohamed V	ALAMI
Murs sultan	TRIKI
My Youssef	RADI
Bd. Mohamed V	BADI

L'argument peut être une expression algébrique

$\pi_{\text{client}} (\nabla \text{agence} = "Bd. Mohamed V")$

client

agence="Bd. Mohamed V"

## Rel X Rel

## Rel

Une relation composée des attributs des deux relations et dont les tuples s'obtiennent par les combinaisons des tuples des deux relations.

Exemple :

Clientèle	Client	ADRESSE	VILLE
	Triki	2, Rue zerhoune	Rabat
	Alami	5, Rue oujda	Rabat
	Badi	70, Rue sebou	Casa
	Radi	2, Av. My Youssef	Rabat

Affectation	Client	employé	
	Alami	Amine	
	Triki	Kacem	
	Badi	Amine	
	Radi	Kacem	

Les clients de l'employé Amine et les villes où ils résident ?

$r = \text{Affectation} \times \text{clientèle}$

## R =Affectation X clientèle

Affectation .client	Affectation .employé	Clientèle .client	Clientèle .adresse	Clientèle .ville
Alami	Amine	Triki	2, Rue zerhoune	Rabat
Alami	Amine	Alami	5, Rue oujda	Rabat
Alami	Amine	Badi	70, Rue sebou	Casa
Alami	Amine	Radi	2, Av. My Youssef	Rabat
Truki	Kacem	Triki	2, Rue zerhoune	Rabat
Triki	Kacem	Alami	5, Rue oujda	Rabat
Triki	Kacem	Badi	70, Rue sebou	Casa
Triki	Kacem	Radi	2, Av. My Youssef	Rabat
Badi	Amine	Triki	2, Rue zerhoune	Rabat
Badi	Amine	Alami	5, Rue oujda	Rabat
Badi	Amine	Badi	70, Rue sebou	Casa
Badi	Amine	Radi	2, Av. My Youssef	Rabat
Radi	Kacem	Triki	2, Rue zerhoune	Rabat
Radi	Kacem	Alami	5, Rue oujda	Rabat
Radi	Kacem	Badi	70, Rue sebou	Casa
Radi	Kacem	Radi	2, Av. My Youssef	Rabat



Affectation.employé = "Amine" (Affectation X Clientèle)

Affectation .client	Affectation .employé	Clientèle .client	Clientèle .adresse	Clientèle .ville
Alami	Amine	Triki	2, Rue zerhoune	Rabat
Alami	Amine	Alami	5, Rue oujda	Rabat
Alami	Amine	Badi	70, Rue sebou	Casa
Alami	Amine	Radi	2, Av. My Youssef	Rabat
Badi	Amine	Triki	2, Rue zerhoune	Rabat
Badi	Amine	Alami	5, Rue oujda	Rabat
Badi	Amine	Badi	70, Rue sebou	Casa
Badi	Amine	Radi	2, Av. My Youssef	Rabat

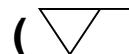


Affectation.client = Clientèle.client

(  $\nabla$  Affectation.employé = "Amine" **(Affectation X Clientèle)**)

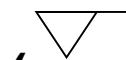
Affectation .client	Affectation .employé	Clientèle .client	Clientèle .adresse	Clientèle .ville
Alami	Amine	Alami	5,Rue oujda	Rabat
Badi	Amine	Badi	70, Rue sebou	Casa

$\pi$



Affectation.client, clientèle.ville

Affectation.client= clientèle.client



Affectation.employé = "Amine" **(Affectation X Clientèle))**

Affection .client	Clientèle .ville
Alami	Rabat
Badi	Casa

## Union

Rel U Rel  $\longrightarrow$  Rel

Union ensembliste entre relations de même schéma

Exemple :

Les clients de l'agence "Bd, Mohamed V" qui y ont un compte ou en ont obtenu un prêt ?

**r1 : ensemble des clients qui ont un compte à l'agence "Bd, Mohamed V" :**

$\pi$  ( (Dépôt))  
 client  $\diagdown$  Agence = "Bd. Mohamed V"

**r2 : ensemble des clients qui ont un crédit à l'agence "Bd, Mohamed V" :**

$\pi$  ( (Crédit))  
 client  $\diagdown$  Agence = "Bd. Mohamed V"

**r1 U r2 :**  $\pi$  ( (Dépôt))  
 client  $\diagdown$  agence = "Bd. Mohamed V"  
 U

$\pi$  ( (Crédit))  
 client  $\diagdown$  Agence = "Bd. Mohamed V"

Conditions de validité de r Us :

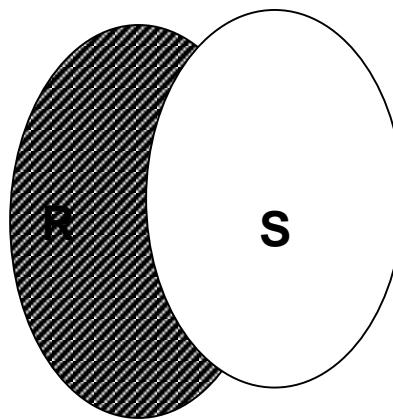
- r et s de même n-arité

- Vi, le domaine du ième attribut de r doit être identique à celui du ième attribut de s.

## Différence

Rel – Rel  $\longrightarrow$  Rel

Différence ensembliste entre des relations de même schéma ( $r - s$  : ensemble de tuples qui satisfont  $r$  sans satisfaire  $s$ )



### Exemple :

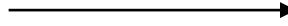
Les clients qui ont un compte à l'agence « Bd, Mohamed V » sans y amortir un prêt ?

$\pi_{\text{client}} (\nabla_{\text{agence} = \text{"Bd. Mohamed V"}} \text{(Dépôt)})$

$-\pi_{\text{client}} (\nabla_{\text{agence} = \text{"Bd. Mohamed V"}} \text{(Crédit)})$

## Autres opérateurs

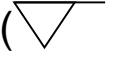
### Intersection

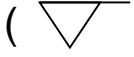
$\text{Rel} \cap \text{Rel}$   Rel

Intersection ensembliste entre relations de même schéma

### Exemple :

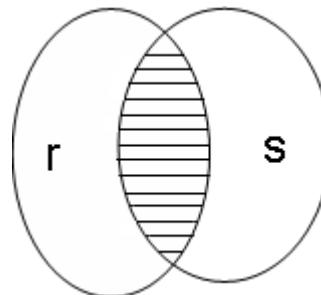
Les clients qui ont un compte et un prêt à l'agence « Bd, Mohamed V » ?

$\pi$   (Dépôt)  
client      agence = "Bd. Mohamed V"

$\cap \pi$   (Crédit)  
client      agence = "Bd. Mohamed V"

Remarque : N'est pas une opération élémentaire :

$$r \cap s = r - (r - s) = s - (s - r)$$



# Division /

- Opération binaire
- syntaxe : R / S
- sémantique : Le quotient de la relation R de schéma  $R = (A_1, A_2, \dots, A_n)$  par la relation S= $(A_{p+1}, \dots, A_n)$  est une relation Q de schéma  $Q = (A_1, A_2, \dots, A_p)$  formée de tous les tuples, qui concaténés à chacun de S donne toujours un tuple de R.
- Exemple :

R	A	B	C
	a	b	a
	y	z	a
	b	b	d

S	A	B
	a	b
	y	z

R/S	C
	a

## Jointure

Rel  Rel  $\longrightarrow$  Rel

Composition de deux relations sur un domaine commun

Exemple :

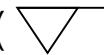
Les clients qui amortissent un prêt et leur lieu de résidence ?

Crédit	agence	prêt	client	montant
	Bd. Mohamed V	15	ALAMI	500
	Murs sultan	20	TRIKI	700
	My Youssef	10	RADI	400

Clientèle	client	adresse	ville
	Triki	2,Rue zerhoune	Rabat
	Alami	5,Rue oujda	Rabat
	Badi	70,Rue sebou	Casa

$\pi$

Crédit.client, Clientèle.adresse, Clientèle.ville

(

(Crédit X Clientèle))

Crédit.client = Clientèle.client'

client	adresse	ville
Triki	2, Rue zerhoune	Rabat
Alami	5, Rue oujda	Rabat

$\pi$

(Crédit  Clientèle)

Crédit.client, Clientèle.adresse, Clientèle.ville

La jonction  $\bowtie$  (appelée également équijonction) est une opération binaire combinant le produit cartésien et la sélection basée sur l'attribut commun

$$\begin{array}{c}
 (\bigtriangleup \\
 \text{Crédit. Client} = \text{Clientèle. client'} \\
 \equiv \\
 (\text{Crédit} \bowtie \text{Client})
 \end{array}
 \quad (\text{Crédit X Client})$$

### Exemple :

Les clients qui amortissent un prêt auprès de l'agence « Bd, Mohamed V » et leur lieu de résidence ?

$\pi$

Crédit.client, Clientèle.adresse, Clientèle.ville

$$\begin{array}{c}
 (\bigtriangleup \\
 \text{agence} = \text{"Bd. Mohamed V"} \\
 \equiv
 \end{array}
 \quad (\text{Crédit} \bowtie \text{Client})$$

### Définition formelle :

- R,S : deux schémas relationnels, avec :

$$R \cap S = \{a_1, a_2, \dots, a_n\}$$

- r(R), s(S) : deux relations de R, S

$$r \bowtie s =$$

$$\begin{array}{ccc}
 \pi & (\bigtriangleup & (r \times s) \\
 R \cup S & r.a_1 = s.a_1 \wedge \dots \wedge r.a_n = s.a_n
 \end{array}$$

$$\text{si } R \cap S = \emptyset \text{ alors } r \bowtie s = r \times s$$

### Exemple :

Les noms et avoirs des agences qui ont des clients de casa ?

## Travaux dirigés (N°1)

### Objectifs :

Maitriser les concepts du modèle relationnel et les opérations de l'algèbre relationnelle

# Travaux dirigés (N°1)

## Exercice 1 :

En considérant les relations R1(A, B, C, D) et R2(A, E, F, G) où :

R1			
A	B	C	D
1	2	4	7
2	7	1	1
4	7	4	8
9	6	4	8
1	2	4	8
4	7	8	4
2	7	4	8
3	54	4	8

R2			
A	E	F	G
2	2	4	20
4	4	5	9
5	2	4	4
1	2	4	9
3	5	7	9

donner les relations résultant des requêtes suivantes :

- 1)  $R = \pi_{[A,B,C]} R_1$
- 2)  $S = \pi_{[A]} R_1 - \pi_{[A]} R_2$
- 3)  $T = R_1 \bowtie R_2$
- 4)  $Q = R_1 / \pi_{[A,B]} R_1$

# Travaux dirigés (N°1) - Suite

## Exercice 2 :

Soit le schéma relationnel de base de données suivant :

**SALLE** (Nom, Horaire, Titre)

**FILM** (Titre, Réalisateur, Acteur)

**PRODUIT** (Producteur, Titre)

**VU** (Spectateur, Titre)

**AIME** (Spectateur, Titre)

### Avec les règles suivantes :

Une salle peut proposer plusieurs films au même horaire

Plusieurs salles peuvent proposer le même film et éventuellement au même horaire

Un réalisateur peut réaliser plusieurs films

Un réalisateur peut également être acteur dans son film ou un autre film

Un producteur peut produire plusieurs films, de même qu'un film peut être produit par plusieurs producteurs

Un spectateur peut aimer plusieurs films, de même qu'il peut voir plusieurs films.

*On suppose que le nom d'une personne est unique.*

### Questions :

1) Donnez le sens de chaque relation dans ce schéma.

2) Donnez la clé primaire de chaque relation et toutes les clés étrangères qui apparaissent dans ce schéma.

# Travaux dirigés (N°1) - Suite

**3)** Ecrire les requêtes suivantes en algèbre relationnelle :

**3.1)** Où et à quelle heure peut-on voir le film " Casa Night " ?

**3.2)** Quels sont les films réalisés par "Farhati" ?

**3.3)** Quels sont les acteurs de "Casa Night" ?

**3.4)** Quels sont les acteurs qui jouent dans des films de "Farhati" ?

**3.5)** Où peut-on voir un film dans lequel joue "Wali" ?

**3.6)** Quels sont les acteurs qui ont produit un film ?

**3.7)** Quels sont les acteurs qui produisent un film dans lequel ils jouent ?

**3.8)** Où peut-on voir un film dans lequel joue " Wali" après 16 h ?

**3.9)** Quels acteurs jouent dans tous les films ?

**3.10)** Quels acteurs jouent dans tous les films de "Farhati" ?

**3.11)** Qui produit tous les films de "Farhati" ?

**3.12)** Quels spectateurs voient tous les films ?

**3.13)** Quels films ne passent dans aucune salle ?

**3.14)** Qui n'aime aucun film ?

**3.15)** Quels spectateurs aiment un film qu'ils n'ont pas vu ?

**3.16)** Quels sont les spectateurs qui aiment tous les films qu'ils voient ?

**3.17)** Quels sont les producteurs qui voient tous les films qu'ils produisent ?

**3.18)** Qui produit un film qui ne passe dans aucune salle ?

**3.19)** Qui ne produit aucun film de "Farhati" ?

**3.20)** Quels producteurs voient tous les films de "Farhati" ?

**3.21)** Quels sont les acteurs qui produisent un film qu'ils n'ont pas réalisé ?

**3.22)** Quels sont les producteurs qui ne voient que les films qu'ils produisent ?

# LANGUAGE SQL (Structured Query Language)

Langage relationnel commercial mettant en jeu, pour la consultation, une combinaison de l'algèbre relationnelle et du calcul relationnel

## La norme SQL

1970 : article de E. F. CODD

“A Relational model for large Data Banks”,

ACM vol 13, No 6, october 1970

présentant la théorie des bases de données relationnelle

IBM → System R, SEQUEL

Barkley → INGRESS, QUEL

1980 : SEQUEL enrichi et amélioré, a donné lieu à SQL : Langage utilisé par les SGBD SQL/DS, DB2

1986 : norme SQL86 préparée par le comité (X3H2) de l' ANSI, adoptée également par ISO et par X/open en 87

1989 : ANSI publie une extension de la norme sous le nom SQL89

# **LANGUAGE SQL**

## **(Structured Query Language)**

Le langage de définition des données (LDD)

**CREATE**  
**ALTER**  
**DROP**

Le langage de manipulation des données (LMD)

**SELECT**  
**INSERT**  
**UPDATE**  
**DELETE**

# Langage de Définition de Données

## Syntaxe CREATE DOMAIN

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

Exemple :

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

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

# Langage de Définition de Données

## Syntaxe de la commande CREATE TABLE

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

# Langage de Définition de Données

## Contrainte sur une colonne

[ **CONSTRAINT** <nom de la contrainte> ]

[  
    **NOT NULL** |  
    **UNIQUE** |  
    **PRIMARY KEY** |  
    **CHECK** (condition) |  
    **REFERENCES** <nom de la table> (colonne)  
]

## 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>**]  
]

# Langage de Définition de Données

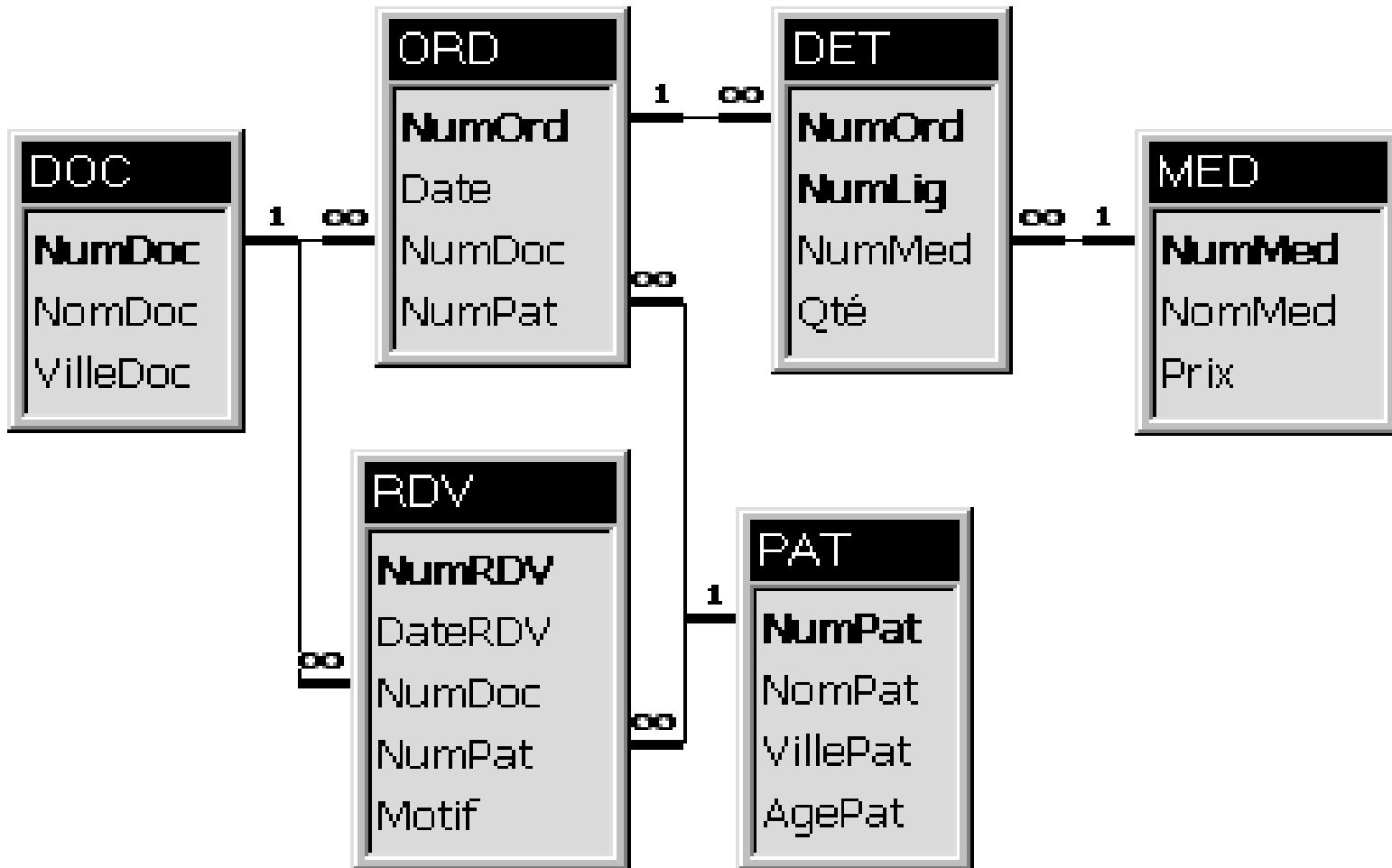
```
[<mode> ::= [ON DELETE {CASCADE|SET DEFAULT|SET NULL}]  
          | [ON UPDATE {CASCADE| SET DEFAULT| SET NULL}]  
]
```

## Remarque :

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

Contrainte sur la table : si la contrainte fait intervenir PLUSIEURS ATTRIBUTS.

# Langage de Définition de Données



# Langage de Définition de Données

Create Table DOC(

    NumDOC integer **PRIMARY KEY**,

    NomDOC VARCHAR2(20),

    VilleDOC VARCHAR2(20)

);

Create Table DOC(

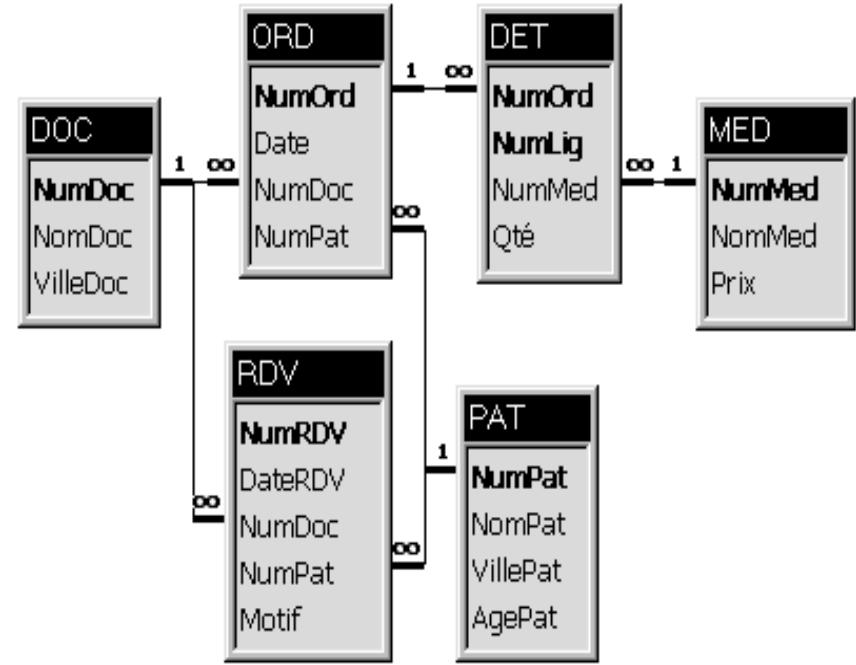
    NumDOC integer,

    NomDOC VARCHAR2(20),

    VilleDOC VARCHAR2(20),

**Constraint PK\_DOC Primary Key (NumDOC)**

);

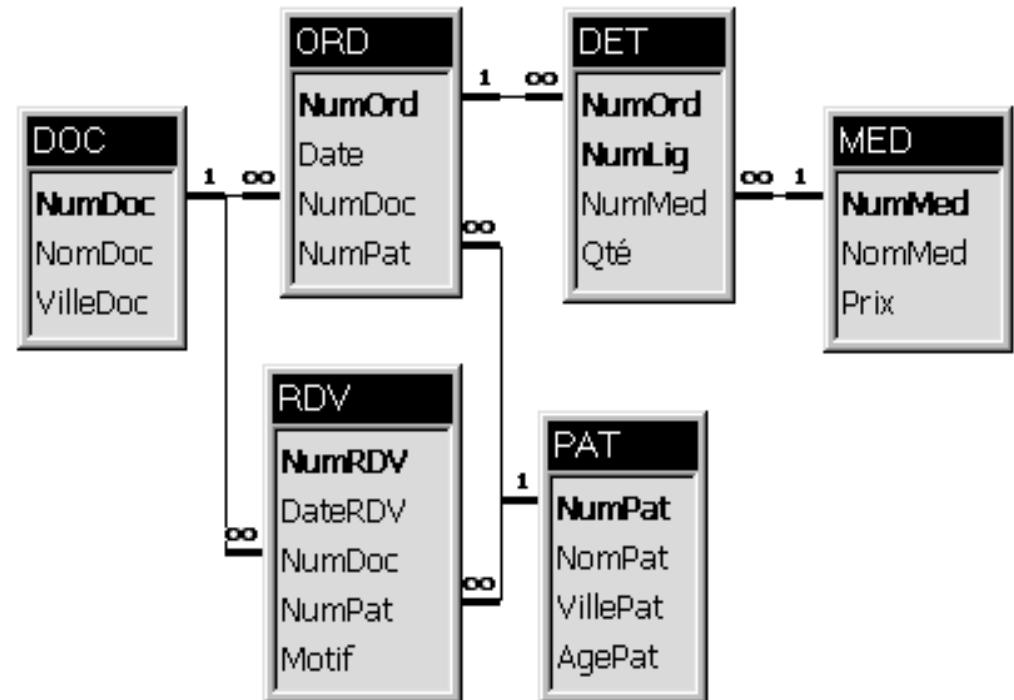


# Langage de Définition de Données

Create Table DOC(

NumDOC integer **Constraint PK\_DOC PRIMARY KEY**,  
NomDOC VARCHAR2(20),  
VilleDOC VARCHAR2(20)

);



# Langage de Définition de Données

Create Table DET(

NumORD integer,

NumLigne integer,

NumMED integer,

QTE integer **NotNull**,

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

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

**Constraint Ref\_ORD**

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

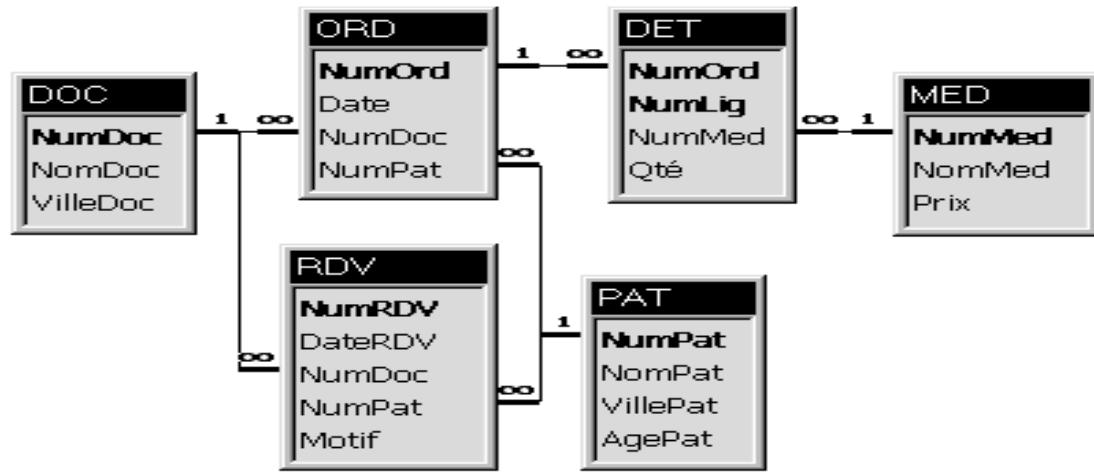
**on delete cascade,**

**Constraint Ref\_MED**

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

**on update cascade**

);



# Langage de Définition de Données

## Définition du concept d'Index

Un index est une structure de données qui permet d'accélérer les recherches dans une table en associant à une clé d'index (la liste des attributs indexés) l'emplacement physique de l'enregistrement sur le disque et ainsi améliorer les performances d'une application utilisant une base de données.

## Syntaxe de création d'un Index en SQL

- **Syntaxe basique :**

```
CREATE INDEX 'Index_Nom' ON 'Table_Nom';
```

- **Index sur une seule colonne :**

```
CREATE INDEX 'Index_Nom' ON 'Table_Nom' ('Colonne1');
```

- **Index sur plusieurs colonnes**

```
CREATE INDEX 'Index_Nom' ON 'Table_Nom' ('Colonne1', 'Colonne 2', ...);
```

- **Index unique (unicité des valeurs de la colonne indexée)**

```
CREATE UNIQUE INDEX 'Index_Nom' ON 'Table_Nom' ('Colonne1');
```

# Langage de Définition de Données

## Syntaxe de la commande ALTER TABLE

**ALTER TABLE <nom de la Table>**

{

**ADD COLUMN <def Colonne>** |

**DROP COLUMN <nom Colonne> [RESTRICT|CASCADE]** |

**ADD CONSTRAINT <def Contrainte>** |

**DROP CONSTRAINT <nom Contrainte> [RESTRICT|CASCADE]** |

}

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

**CASCADE** : propage la destruction

# Langage de Définition de Données

## Exemples :

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

```
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;
```

# Langage de Définition de Données

## Syntaxe DROP TABLE

```
DROP TABLE <Nom de la table>
```

**Exemple :**

Drop table DOC;

# **LANGUAGE SQL** **(Structured Query Language)**

Le langage de définition des données (LDD)

**CREATE**  
**ALTER**  
**DROP**

Le langage de manipulation des données (LMD)

**SELECT**  
**INSERT**  
**UPDATE**  
**DELETE**

# **Langage de Manipulation de Données**

## Syntaxe INSERT INTO

**INSERT INTO <nom table>**

```
[ ( colonne1 [, colonne2] ... ) ]
{ VALUES (<valeur1> [, <valeur2>] ... )
| <requête SELECT > }
```

;

### **Exemples :**

Table: DOC(NumDoc, NomDoc, VilleDoc)

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

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

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

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

# **Langage de Manipulation de Données**

## **Syntaxe UPDATE**

**UPDATE <nom table>**

**SET <colonne> = valeur**

**[, <colonne> = valeur ] ...**

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

## **Exemples :**

Table: DOC(NumDoc, NomDoc, VilleDoc)

**UPDATE DOC**

**SET NomDoc='Andaloussi', NomVille='Safi'**

**Where NumDoc=444;**

**UPDATE DOC**

**SET VilleDoc=NULL;**

# **Langage de Manipulation de Données**

## Syntaxe DELETE

**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  
              );

# **LANGUAGE SQL** **(Structured Query Language)**

Le langage de définition des données (LDD)

**CREATE**  
**ALTER**  
**DROP**

Le langage de manipulation des données (LMD)

**SELECT**  
**INSERT**  
**UPDATE**  
**DELETE**



# Langage de Manipulation de Données

## Syntaxe simplifiée de la requête Select

```
SELECT [ DISTINCT ] * | expr[, expr...]  
FROM table  
[ WHERE condition ]  
[ ORDER BY expr| position [ASC| DESC]] ;
```

# Langage de Manipulation de Données

## Conditions

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**

# **Langage de Manipulation de Données**

**Exemples :**

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

```
SELECT DISTINCT Prenom  
FROM personnes;
```

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

# Langage de Manipulation de Données

## Fonctions de groupe

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

# Langage de Manipulation de Données

## Exemples

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

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

```
SELECT Min(nom), Max(nom)  
FROM personnes ;
```

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

# Langage de Manipulation de Données

## Exemples

```
SELECT Count( telephone)  
FROM personnes
```

```
SELECT Count( DISTINCT prenom)  
FROM personnes;
```

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

# Langage de Manipulation de Données

EMP	Deptno	sal
	10	5000
	10	1500
	10	1300
	20	2975
	20	3000
	20	1100
	30	2850
	30	1250
	30	1600
	30	1500
	30	950
	30	1250

**Table EMP**

## Question :

Salaire moyen pour chaque département de la table EMP

# Langage de Manipulation de Données

## La clause GROUP BY

```
SELECT column, group_fonction
FROM table
[ WHERE condition ]
[ GROUP BY group_by_expression ]
[ ORDER BY column];
```

### Remarque :

Les attributs du select ne peuvent être que

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

# Langage de Manipulation de Données

**Exemple :**

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

EMP	Deptno	sal
	10	2600
	20	2175
	30	1566.7

# Langage de Manipulation de Données

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

## Question

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

# Langage de Manipulation de Données

```
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	6000
20	programmeur	1900
20	Chef projet	2975
30	programmeur	950
30	Chef projet	2850
30	Commercial	4350

# Langage de Manipulation de Données

## GROUP BY avec HAVING

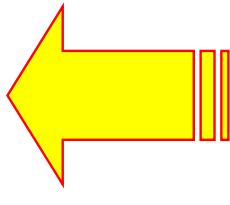
```
SELECT column, group_fonction  
FROM table  
[ WHERE condition ]  
[ GROUP BY group_by_expression  
[ HAVING group_condition ] ]  
[ ORDER BY column];
```

# Langage de Manipulation de Données

## Exemple

```
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

# Langage de Manipulation de Données

## Synthèse

```
SELECT column, group_fonction  
FROM tables  
[ WHERE condition ]  
[ GROUP BY group_by_expression  
[ HAVING group_condition ] ]  
[ ORDER BY column];
```

# Langage de Manipulation de Données

## Requêtes sur plusieurs tables

- Les opérateurs de jointures
- L'imbrication de requêtes
- Les opérateurs ensemblistes

# Langage de Manipulation de Données

## Requêtes sur plusieurs tables : la jointure

Equijointure (*jointure naturelle*)

Autojointure (jointure sur la même table)

Jointure externe

Non-équijointure (*jointure par non égalité, théta jointure*)

# Langage de Manipulation de Données

Requêtes sur plusieurs tables : la jointure

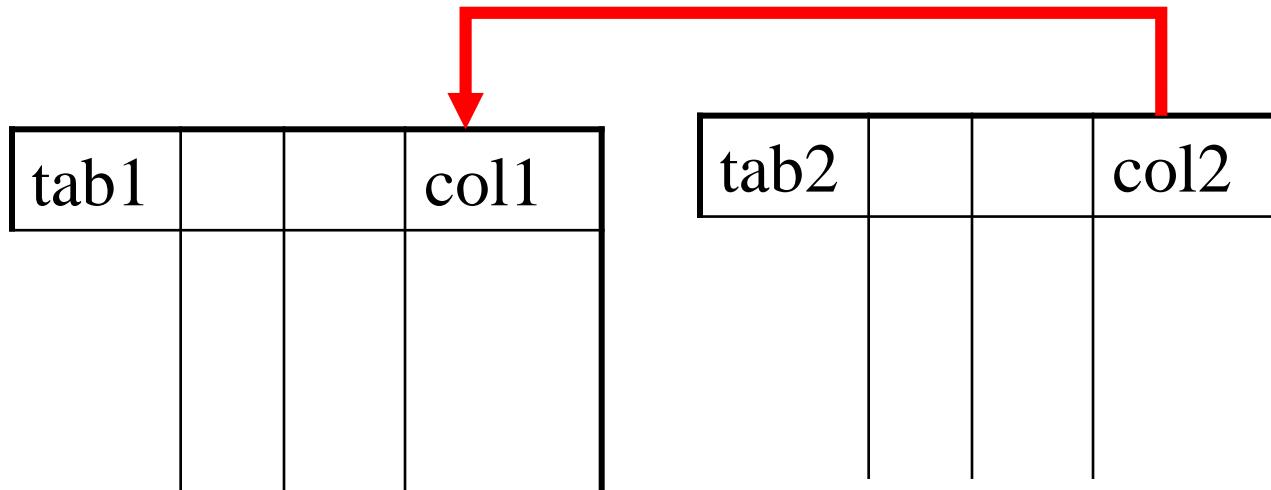
**Equijointure (*jointure naturelle*)**

```
SELECT expr  
FROM table 1  
INNER JOIN table2 ON table1.col1=table2.col2
```

```
SELECT expr  
FROM table 1  
INNER JOIN table2  
WHERE table1.col1=table2.col2
```

ou

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

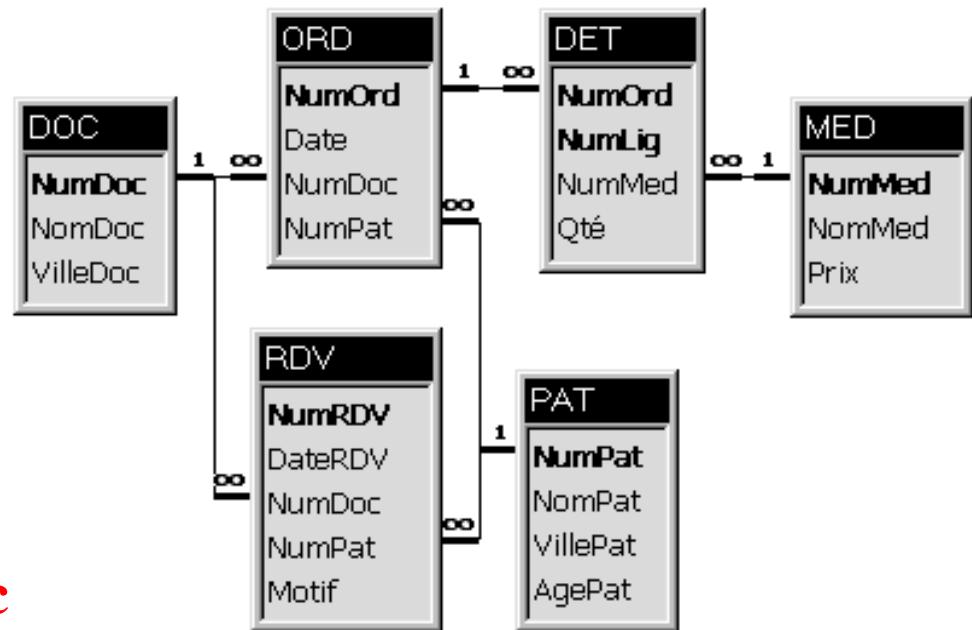


# Langage de Manipulation de Données

Equijointure (*jointure naturelle*)

Liste des RDV  
avec le docteur ‘Alaoui’

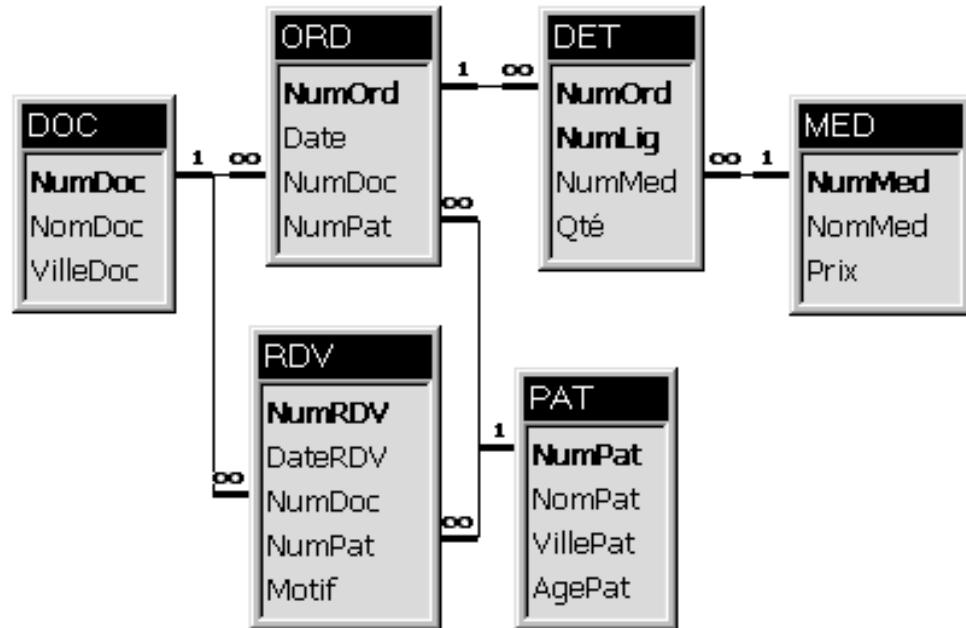
```
SELECT NumRDV
FROM RDV R ,DOC D
WHERE R.NumDoc = D.NumDoc
      and D.NomDoc =‘Alaoui’;
```



# Langage de Manipulation de Données

Equijointure (*jointure naturelle*)

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';
```

# Langage de Manipulation de Données

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

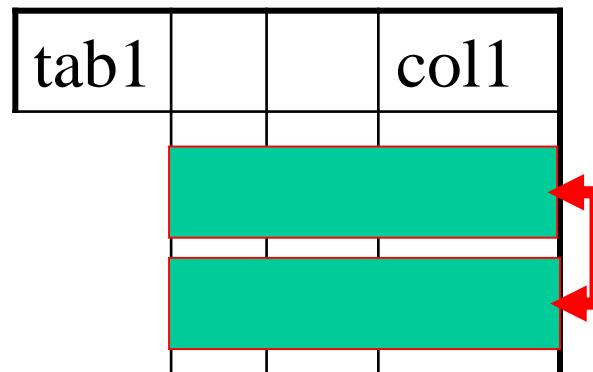
Liste des employés ayant un salaire égal à celui de « Azhari »

# Langage de Manipulation de Données

Requêtes sur plusieurs tables : la jointure

Autojointure

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



# Langage de Manipulation de Données

## Autojointure

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

# Langage de Manipulation de Données

## Autojointure

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

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

```
SELECT E2.Nom
FROM   EMP E1, EMP E2
WHERE  E1.sal<=E2.sal
      and E1.Nom =‘Azhari’;
```

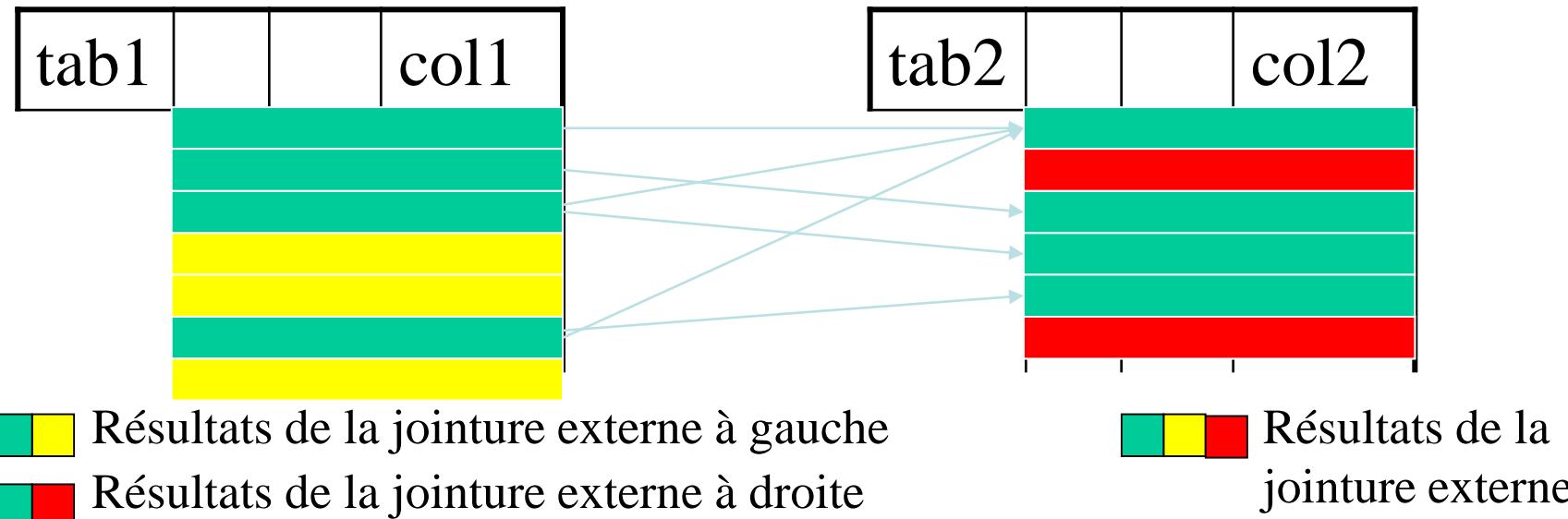
# Langage de Manipulation de Données

## Requêtes sur plusieurs tables: la jointure externe

## 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

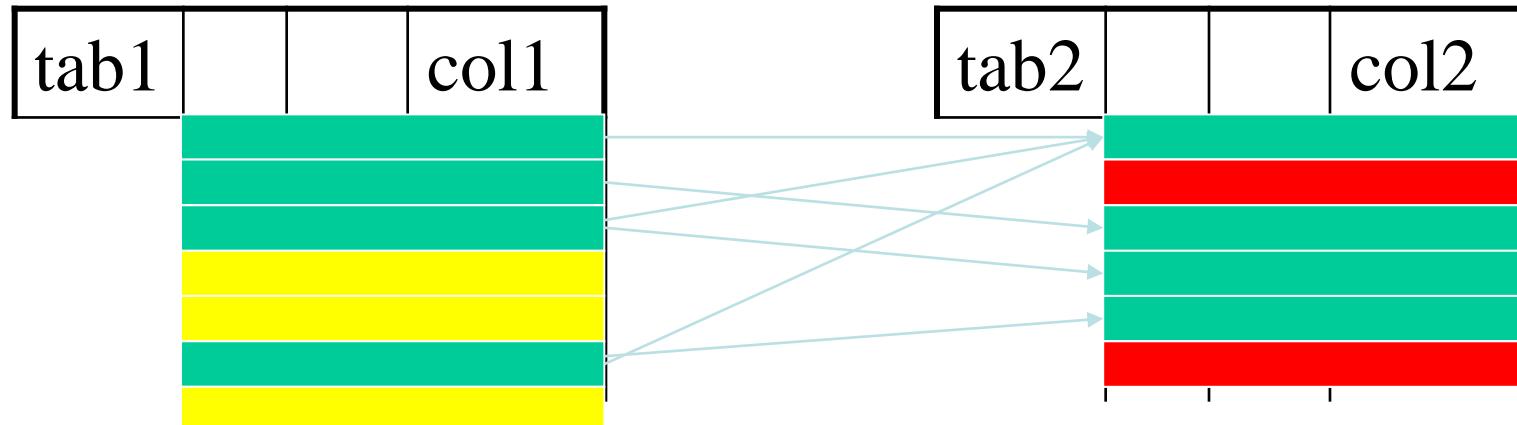


# Langage de Manipulation de Données

Requêtes sur plusieurs tables: la jointure externe

## Jointure Externe

```
SELECT table1.colonne, table2.colonne  
FROM table1 FULL OUTER JOIN table2  
ON table1.col1 = table2.col2 ;
```



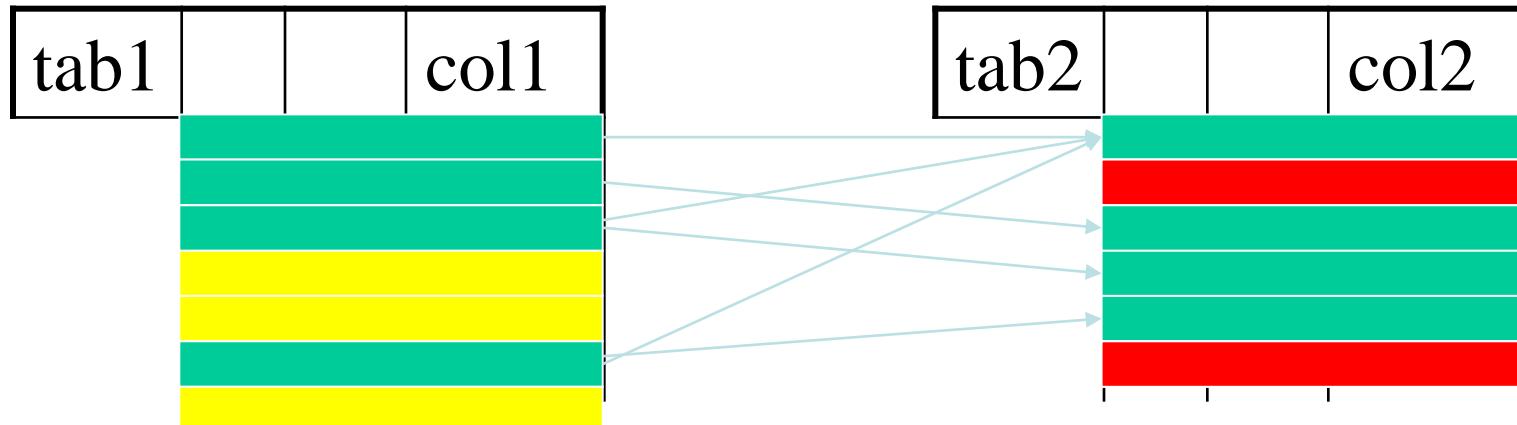
■ ■ ■ Résultats de la jointure externe

# Langage de Manipulation de Données

Requêtes sur plusieurs tables: la jointure externe

## Jointure Externe à gauche

```
SELECT table1.colonne, table2.colonne  
FROM table1 LEFT OUTER JOIN table2  
ON table1.col1 = table2.col2 ;
```



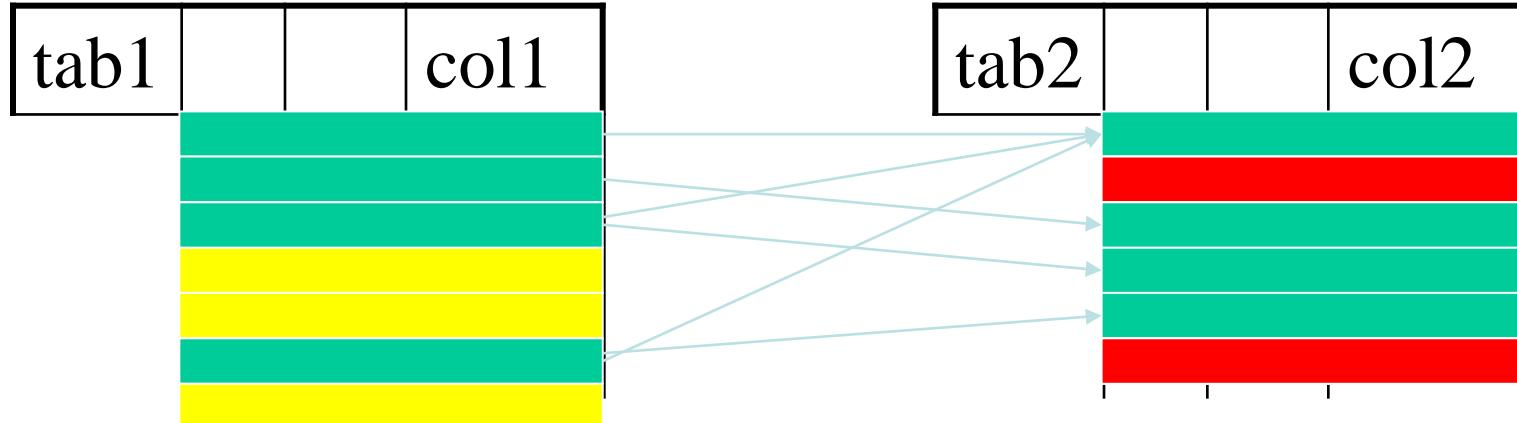
Résultats de la jointure externe à gauche

# Langage de Manipulation de Données

## Requêtes sur plusieurs tables : la jointure externe

### Jointure Externe à droite

```
SELECT table1.colonne, table2.colonne  
FROM table1 RIGHT OUTER JOIN table2  
ON table1.col1 = table2.col2 ;
```



■ Résultats de la jointure externe à droite

# Langage de Manipulation de Données

## Requêtes sur plusieurs tables

### Non Equijointure (thêta jointure)

Exemple : la liste des employés et leurs grades

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



## **Langage de Manipulation de Données**

### **Requêtes imbriquées**

# Langage de Manipulation de Données

## Requêtes imbriquées : Syntaxe générale

```
SELECT colonnes_de_projection  
FROM table  
WHERE expr operator (
```

```
SELECT colonnes_de_projection  
FROM table  
WHERE ....
```

```
);
```

sens d'exécution ↑

# Langage de Manipulation de Données

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

Type de sous requêtes et opérateurs possibles

# Langage de Manipulation de Données

## 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);
```

# Langage de Manipulation de Données

## Exemples

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

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

Le nom de l'employé le mieux payé ?

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

# Langage de Manipulation de Données

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

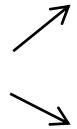
**ANY** : la condition est vraie si la comparaison est vraie pour AU MOINS une des valeurs renvoyées par la sous-requête

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

**EXISTS (sous-requête)**

**FAUX si Resultat(Sous-requête) = &**

**VRAIE si Resultat(Sous-requête) ≠ &**



## Exemples :

Liste des docteurs n'ayant pas eu de RDV en 2009

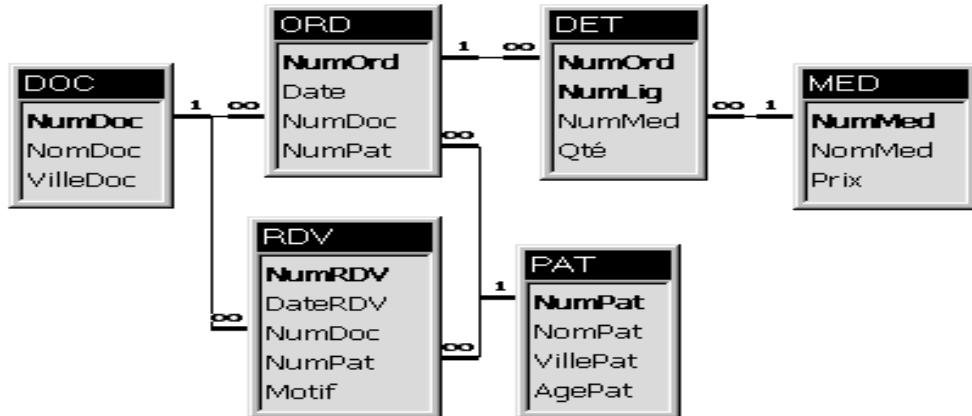
Select \* from DOC

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

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

**Q3 :** Le nombre de RDV par docteur en 2019

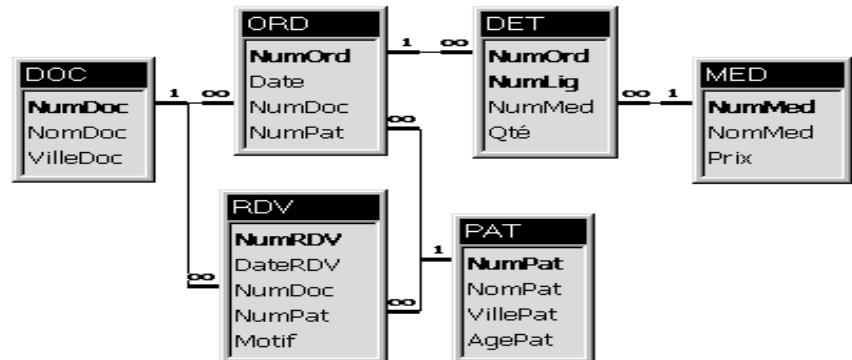
**Q4 :** Patients sans RDV en 2019

**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 médicaments le plus prescrit en 2019.



## **Langage de Manipulation de Données**

### **Opérateurs ensemblistes**

# Langage de Manipulation de Données

## Opérateurs ensemblistes

INTERSECT

UNION

UNION ALL

MINUS

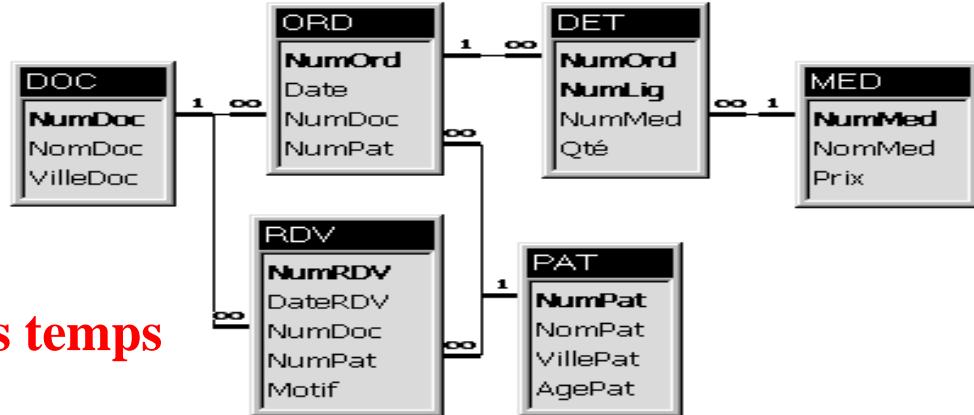
**Requête SELECT**

**<Opérateur ensembliste>**

**Requête SELECT**

**Remarque :** Il est nécessaire que chacune des deux requêtes retourne le même nombre de colonnes, avec les mêmes types de données et dans le même ordre.

## Exemples :



**Les médicaments prescrits en mêmes temps dans les ordonnances 1 et 3**

Select NumMed from DET Where NumOrd=1

**INTERSECT**

Select NumMed from DET Where NumOrd=3;

**Les docteurs n'ayant pas eu de RDV en 2022**

Select NumDoc from DOC

**MINUS**

Select NumDoc from RDV Where dateRDV Between  
'01/01/2022' and '31/12/2022' ;

# LES VUES

# Les vues

## Définition :

Une vue est une Table virtuelle calculée à partir d'autres tables ou vues par une requête  
Pas d'existence physique mais recalculée chaque fois qu'elle est invoquée

Vue mono table

Vue multi-tables

## Intérêts :

Indépendance application/données

Personnalisation des données selon les besoins des utilisateurs

Confidentialité

Rapidité des requêtes

## Utilisation :

Pour les sélections, comme une table ordinaire

Pour les maj. (insert, update, delete), y a des restrictions

## Les vues

### Syntaxe de CREATE VIEW

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

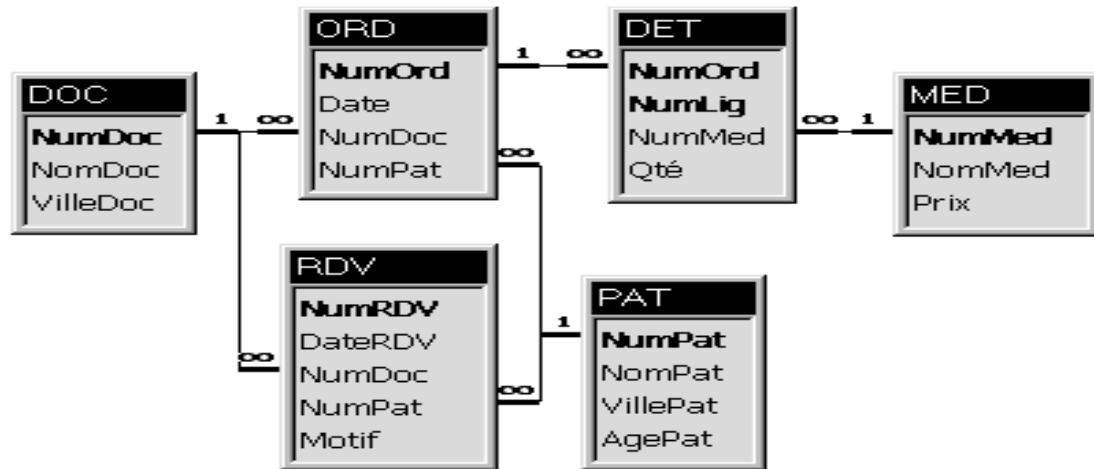
**AS <requête de sélection>**

**[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.

# Les vues



Exemples :

```
CREATE VIEW MedecinsDeRabat AS
Select *
From DOC
Where villeDoc='Rabat';
```

```
CREATE VIEW ORD_Total (NumORD, Total) AS
Select NumORD, SUM(Qte*Pri)
FROM DET D, MED M
WHERE D.NumMed=M.NumMed
GROUP BY NumORD;
```

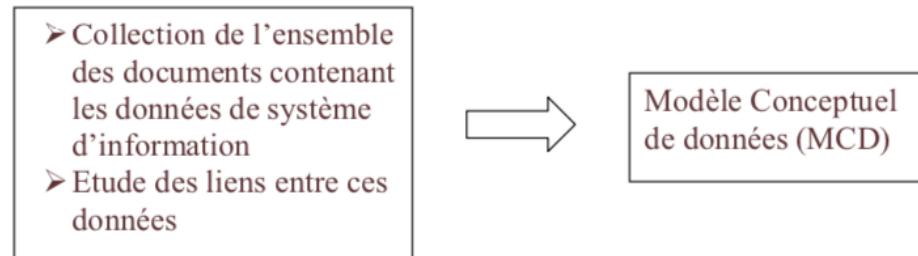
# Les vues

## 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	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

# **Conception de bases de données relationnelles**

## I- Introduction



## II- Définitions

**a) Propriété :** C'est un attribut (caractéristique) que l'on perçoit sur une entité ou une relation.

Exemple : *nom client, adresse client* sont des propriétés de l'entité «*Client*».

Une propriété peut être :

- ◆ **Elémentaire** (simple) : *Age, N°pièce;*
- ◆ **Composée** (concaténée) : *Adresse =(Rue, Ville);*
- ◆ **Calculée** : *Montant TVA = Montant HT \* Taux TVA.*

**b) Entité :** Elle est décrite par une liste de propriétés qui lui sont spécifiques.

Exemples :

*Client (N° Client, Nom Client, Adresse Client)*

*Pièce(N° Pièce, Nom Pièce, Prix unitaire)*

# Modèle Conceptuel de Données

c) **Relation** : C'est un ensemble d'interactions semblables qui existent entre les entités. Elle permet d'associer plusieurs entités.

Exemple :

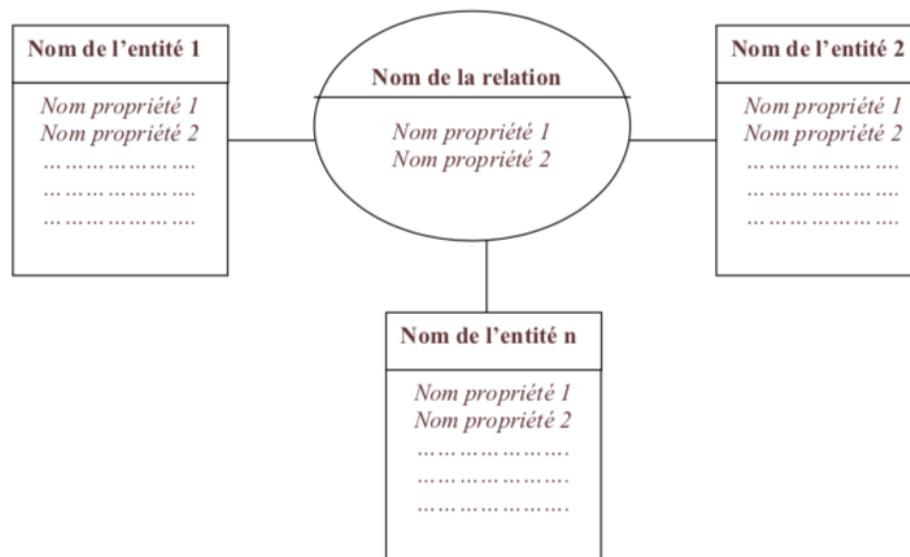
Soit l'entité **Commande** (*N° commande, date commande*) ;  
et **Client** (*N° Client, Nom Client, Adresse Client*)  
La Commande Concerne le Client.  
«Concerne» est une relation entre les deux entités **Commande** et **Client**.

Formalisme de Modèle Entité – Relation :

Entité → Rectangle

Relation → Ellipse

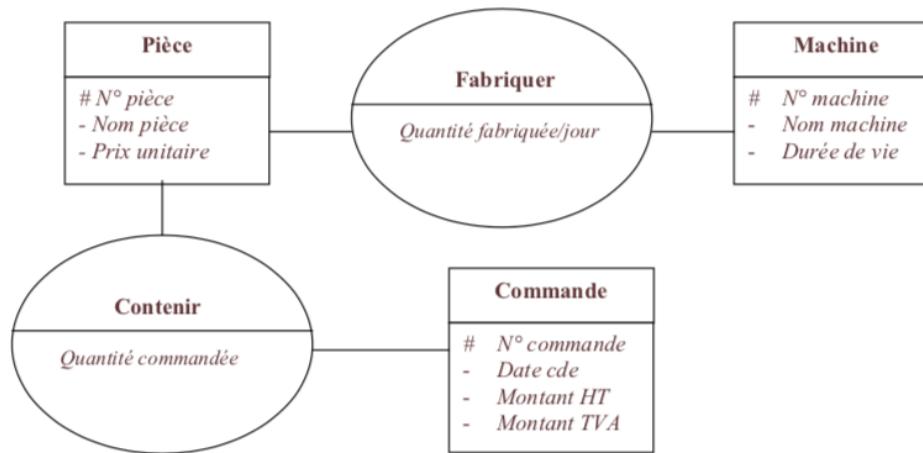
Lien Entité-Relation → Trait



# Modèle Conceptuel de Données

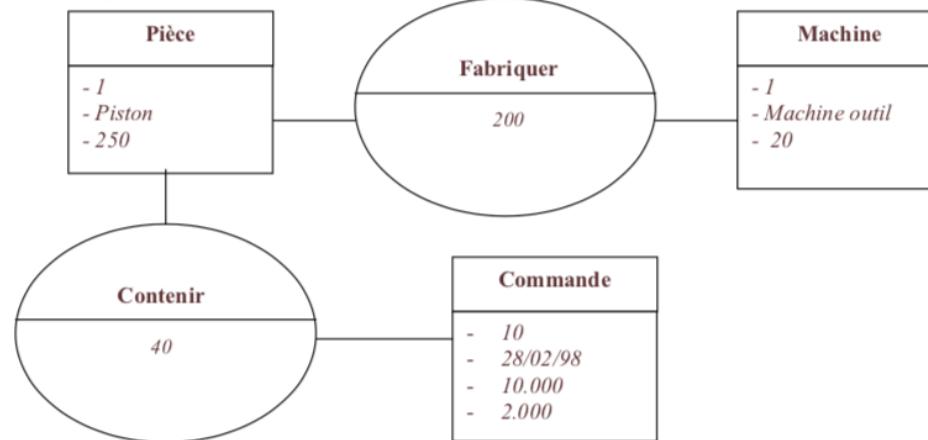
## Exemple :

Un **bon de commande** contient un ensemble de quantités de **pièces** fabriquées par des **machines** spécifiques.



d) **Occurrence** : Une occurrence d'une entité, relation ou propriété est un élément particulier de ce type.

## Exemple :



## **Modèle Conceptuel de Données**

---

- *(1,'Piston',250)* est une occurrence de l'entité « *Pièce* » ;
- *200* est une occurrence de la relation « *Fabriquer* ».

**e) Identifiant (Clé) :** C'est une propriété d'une entité ou d'une relation qui permet d'identifier une et une seule occurrence.

**Exemple :**

*N° pièce* est une clé de l'entité « *Pièce* ».

*N° machine* est une clé de l'entité « **Machine** ».

*N° commande* est une clé de l'entité « **Commande** ».

**f) Clé primaire d'une entité :** C'est le groupe minimum de propriétés qui l'identifient.

**Exemple :**

- *N° pièce* et *Nom pièce* constituent une clé de l'entité « *Pièce* »;
- *N° pièce* est la clé primaire de l'entité « *Pièce* ».

**g) Clé primaire d'une relation :** elle est composée des clés primaires des entités formant cette association.

**h) Cardinalité :** Les cardinalités **maximales** et **minimales** d'une entité via une relation indiquent les nombres **maximums** et **minimums** d'occurrences de la relation pouvant exister pour une occurrence de cette entité.

# Modèle Conceptuel de Données

❶ **Cardinalité minimale** : Représente le nombre minimum d'occurrences de la relation que possède une occurrence de l'entité.

❷ **Cardinalité maximale** : Représente le nombre maximum d'occurrences de la relation que possède une occurrence de l'entité.

Exemple 1 :

◆ Règles de gestion :

- Le bon de commande contient au moins une pièce fabriquée au moins par une machine.
- Chaque machine fabrique au moins une pièce ;
- Une pièce peut ne pas être commandée.

◆ **Cardinalité minimale pour l'entité « Pièce »** : La pièce peut-elle ne pas être commandée ?

- Oui : Cardinalité minimale = 0,  0
- Non : Cardinalité minimale = 1.

◆ **Cardinalité maximale pour l'entité « Pièce »** : La pièce figure-t-elle au maximum dans une commande ?

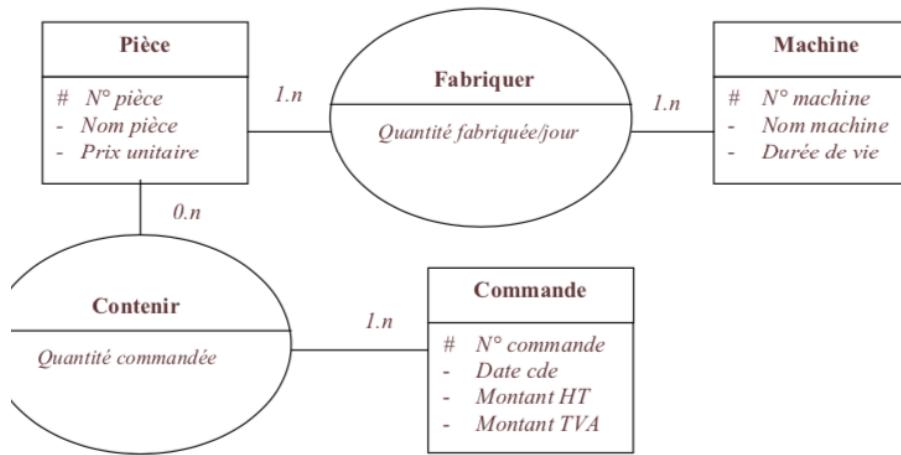
- Oui : Cardinalité maximale = 1,  1
- Non : Cardinalité maximale = n.

Remarque :

$$1 \leq \text{Cardinalité maximale} \leq n.$$

$$0 \leq \text{Cardinalité minimale} \leq 1.$$

# Modèle Conceptuel de Données



## Exemple 2:

### ◆ Règles de gestion :

- Une usine contient des machines qui peuvent fabriquer au moins un type de pièce ;
- Chaque pièce peut être fabriquée par une ou plusieurs machines ;
- Chaque type de machine est construite par un ou plusieurs fournisseurs ;
- Un fournisseur peut construire une ou plusieurs marques de machines.

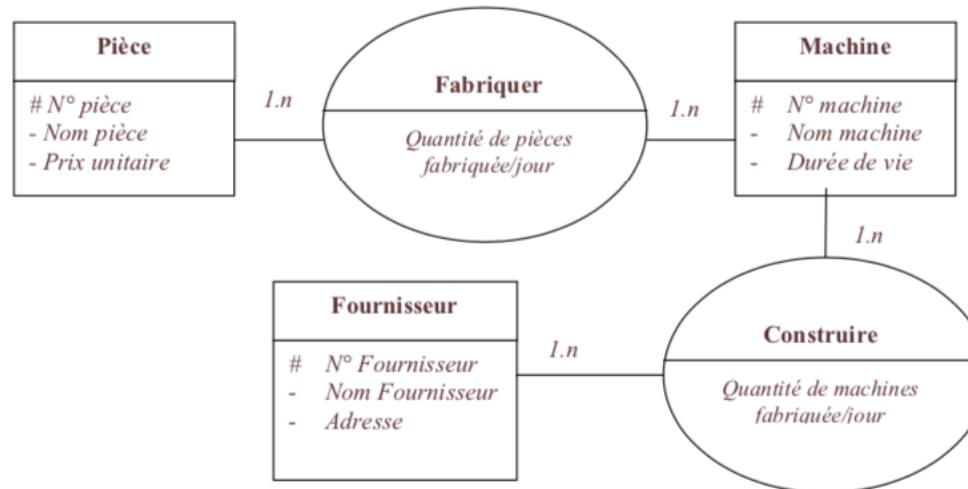
### ◆ Cardinalité minimale pour l'entité « Fournisseur » : Le fournisseur peut-il ne construire aucune machine ?

- Oui : Cardinalité minimale = 0,  $\longrightarrow 1$
- Non : Cardinalité minimale = 1.

### ◆ Cardinalité maximale pour l'entité « Fournisseur » : Le fournisseur construit-il au maximum une machine ?

- Oui : Cardinalité maximale = 1,  $\longrightarrow n$
- Non : Cardinalité maximale = n.

# Modèle Conceptuel de Données



## i) Dépendance Fonctionnelle entre les propriétés

On dit que deux propriétés  $p_1$  et  $p_2$  sont reliées par une dépendance fonctionnelle si la connaissance de  $p_1$  détermine une et une seule valeur de  $p_2$ .

Notation :  $p1 \xrightarrow{DF} p2$

Exemple :

N°pièce  $\xrightarrow{DF}$  Nom pièce

N°pièce  $\xrightarrow{DF}$  Prix unitaire

N°pièce  $\xrightarrow{DF}$  Nom pièce, Prix unitaire

N°pièce, N°machine  $\xrightarrow{DF}$  Quantité de pièces fabriquée/jour

N°fournisseur, N°machine  $\xrightarrow{DF}$  Quantité de machines fabriquée /jour

# Modèle Conceptuel de Données

## j) Dépendance Fonctionnelle élémentaire

On dit que deux propriétés  $p_1$  et  $p_2$  sont reliées par une dépendance fonctionnelle élémentaire si on a les deux conditions suivantes :

- Aucune partie stricte de  $p_1$  ne détermine  $p_2$
- $p_1 \xrightarrow{DF} p_2$

Notation :  $p_1 \longrightarrow p_2$

N°pièce  $\longrightarrow$  Nom pièce, Prix unitaire

N°pièce, N°machine  $\longrightarrow$  Quantité de pièces fabriquée/jour

N°fournisseur, N°machine  $\longrightarrow$  Quantité de machines fabriquée /jour

N°fournisseur, Nom fournisseur  $\xrightarrow{DF} \longrightarrow$  Adresse (pas une d.f.e)

## k) Dépendance Fonctionnelle élémentaire directe

On dit que deux propriétés  $p_1$  et  $p_2$  sont reliées par une dépendance fonctionnelle élémentaire directe s'il n'existe pas de transitivité entre les deux.

Autrement, il n'existe aucune propriété  $p_3$  telle que :

$P_1 \xrightarrow{DF} P_3$  et  $P_3 \xrightarrow{DF} P_2$

Exemple :

N°pièce  $\longrightarrow$  N°machine

N°machine  $\longrightarrow$  N°fournisseur

$\Rightarrow$  N°pièce  $\longrightarrow$  N°fournisseur (n'est directe)

## III- Etapes de construction de Modèle Conceptuel de données

1<sup>ère</sup> étape : Etablissement de liste des propriétés

2<sup>ème</sup> étape : Etablissement de dictionnaire de données

3<sup>ème</sup> étape : Etablissement des Dépendances Fonctionnelles

4<sup>ème</sup> étape : Etablissement de Modèle Conceptuel de Données  
(MCD)

# Modèle Conceptuel de Données

---

## 1<sup>ère</sup> étape : Etablissement de liste des propriétés

- Etablir la liste des propriétés à partir des documents échangés entre les différents acteurs du système d'information.
- Supprimer les *synonymes* : Supprimer les propriétés de *noms différents* provenant d'un ou plusieurs documents ayant la *même signification*.

Exemple de synonymes : N° pièce et Code Pièce

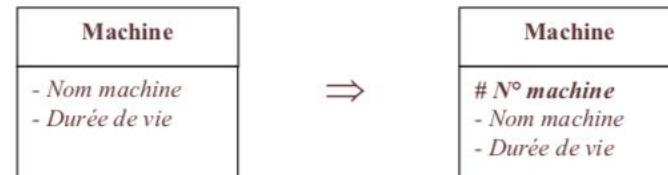
- Régler les *polysèmes* : Un polysème est une propriété qui a *plusieurs signification* dans un ou plusieurs documents.

Exemple de polysèmes : N° dans un document *Pièce* signifie N°*pièce* et N° dans un autre document *Machine* signifie N°*machine*

⇒ régler ce problème par N°*pièce* et N°*machine*

- Ajouter des codes identifiant une entité s'ils n'existeraient pas.

Exemple :



## 2<sup>ème</sup> étape : Etablissement de dictionnaire de données

Un dictionnaire de données est un tableau contenant :

- Le nom abrégé de la propriété,
- Sa signification (description),
- Son type (Numérique, Alphabétique, AlphaNumérique,...),
- Sa longueur,
- Sa nature (Elémentaire, Concaténée, Calculée),
- Observations indiquant les contraintes d'intégrités des propriétés, leurs formats de saisie, ...

Il faut **enlever** du dictionnaire de données les propriétés calculées, concaténées, les synonymes et **régler** les polysèmes.

# Modèle Conceptuel de Données

Exemple : Document bon de commande

N° commande : .....		Date commande : ...../...../.....		
N° CLIENT : .....		NOM CLIENT : .....		
ADRESSE : .....				
N° PIECE	LIBELLE PIECE	QUANTITE COMMANDÉE	PRIX UNITAIRE	MONTANT HT
.....	.....	.....	.....	.....
.....	.....	.....	.....	.....
.....	.....	.....	.....	.....
TOTAL HT : .....				

Le dictionnaire de données associé :

Nom	Signification	Type	Longueur	Nature	Observation
NCDE	N° commande	N	6	E	
DATEC	Date commande	Date	8	E	jj/mm/aa
NCLI	N° client	N	6	E	
NOMCLI	Nom client	A	25	E	
ADRCLI	Adresse	AN	50	CO	Rue + Ville
RCLI	Rue Client	AN	30	E	
VCLI	Ville client	A	20	E	
NPIECE	N° pièce	N	6	E	
LIBPIECE	Libellé pièce	A	30	E	
QTITEC	Quantité commandée	N	4	E	
PUNIT	Prix unitaire de pièce	N	6	E	
MONT	Montant HT	N	10	CA	PUNIT*QTITEC
TOTAL	TOTAL HT de commande	N	12	CA	$\Sigma$ MONT

## 3<sup>ème</sup> étape : Etablissement des Dépendances Fonctionnelles

- Déterminer les liens entre les données afin d'identifier l'ensemble des entités et des relations qui les associent ;
- Eliminer toute transitivité existante.

Exemple : document bon de commande

on déduit les DFs suivantes :

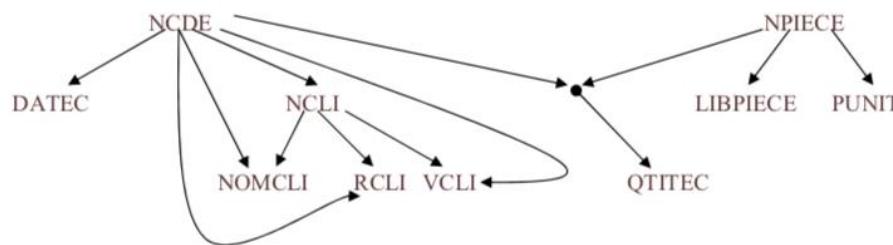
*NCDE → DATEC, NCLI, NOMCLI, RCLI, VCLI*

*NPIECE → LIBPIECE, PUNIT*

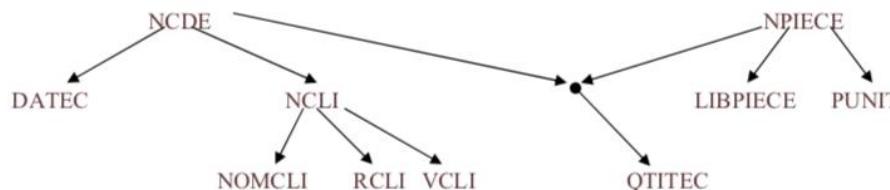
*NCLI → NOMCLI, RCLI, VCLI*

*NCDE, NPIECE → QTITEC*

### Graphe de Dépendances Fonctionnelles



Elimination des transitivités :



# Modèle Conceptuel de Données

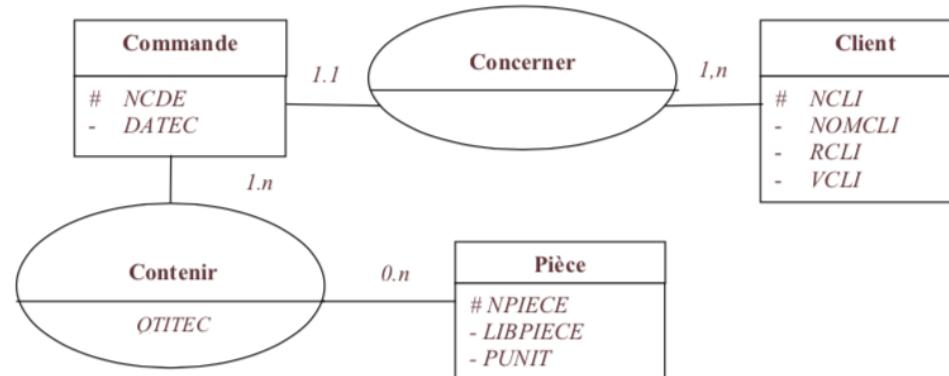
## 4<sup>ème</sup> étape : Etablissement de Modèle Conceptuel de Données (MCD)

- Etablir le modèle Conceptuel des Données associé à chaque document.

### Identification des entités et relations :

- Les propriétés qui dépendent d'une propriété **simple** (non décomposable) forment une entité dont sa clé est cette propriété simple,
- Les propriétés qui dépendent d'une propriété **composée** des clés des entités identifiées **précédemment** forment une relation qui associe ces entités.

Exemple : MCD associé au document bon de commande



### Etablissement de Modèle Conceptuel de Données en cas de plusieurs documents

- Etablit le MCD de chaque document ;
- Fusionner l'ensemble des MCDs pour construire le MCD global ;
- Normaliser le MCD global.

# Modèle Conceptuel de Données

## Exemple :

En plus du document « *bon de commande* » fournit précédemment, on a les deux documents suivants :

- Document **FCLIENT** : Contient les informations sur les clients ;
- Document **MACHINE** : Identifie pour chaque machine l'ensemble des pièces avec les quantités fabriquées par jour.

### Document FCLIENT

<i>N° client : .....</i>	<i>Nom de client : .....</i>
<i>Adresse : ..... Taux de remise client : .....</i>	

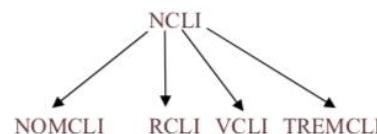
### Document MACHINE

<i>N° machine : ..... Nom machine : .....</i>		
<i>Prix unitaire HT : .....</i>		
<i>N° PIECE</i>	<i>LIBELLE PIECE</i>	<i>QUANTITE FABRIQUEE (par jour)</i>

### ♦ Document **FCLIENT**

#### Dépendances Fonctionnelles :

NCLI → NOMCLI, RCLI, VCLI, TREMCLI



# Modèle Conceptuel de Données

MCD associé au document FCLIENT :

Client
# NCLI
- NOMCLI
- RCLI
- VCLI
- TREMCLI

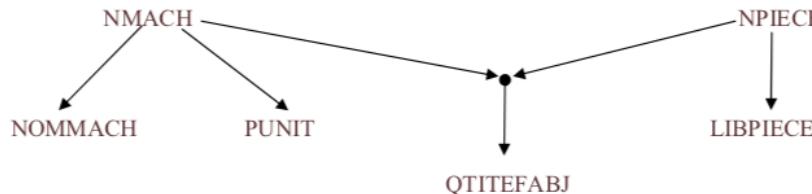
## ◆ Document MACHINE

Dépendances Fonctionnelles :

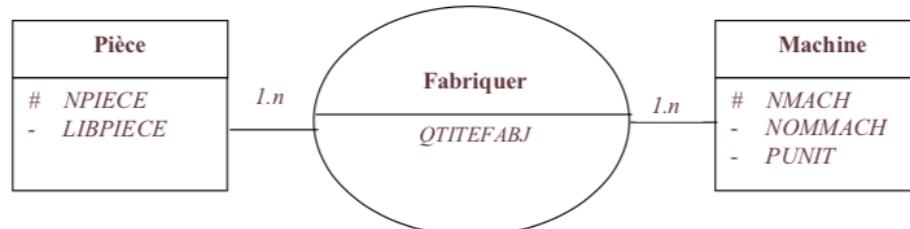
NMACH → NOMMACH, PUNIT

NPIECE → LIBPIECE

NMACH, NPIECE → QTITEFABJ



MCD associé au document MACHINE :



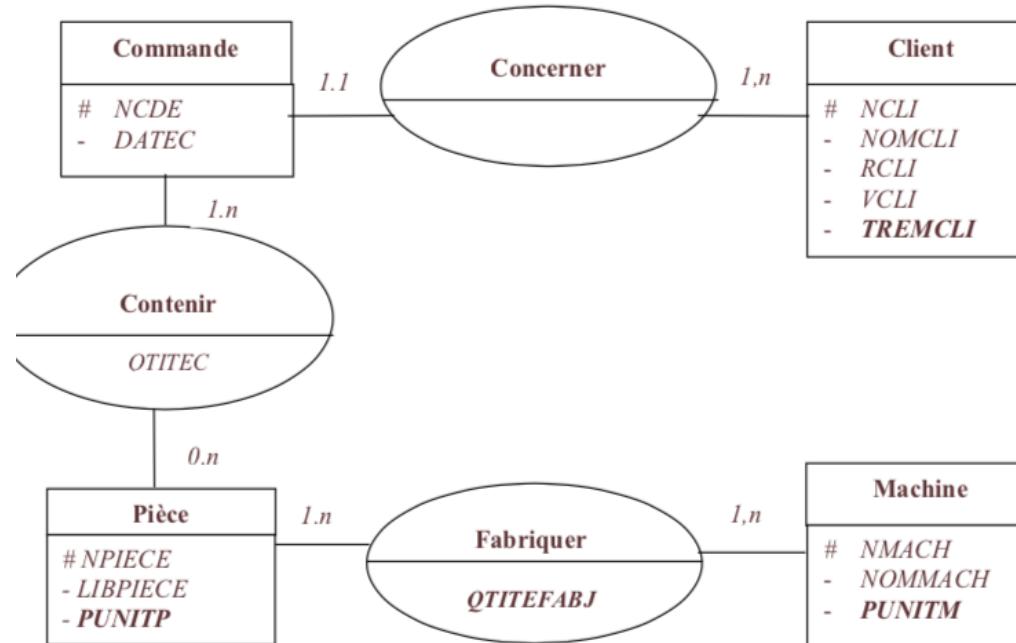
# Modèle Conceptuel de Données

## ◆ Fusion des MCDs associés au différents documents

### Remarque

- *TREMCLI* n'apparaît pas dans le MCD associé au « *Bon de commande* » ;
- *PUNIT* est un *polysème* (elle a deux significations dans les deux MCDs associés à MACHINE et Bon de commande) ; On spécifiera donc *PUNITM* pour le prix unitaire de machine et *PUNITP* pour celui de la pièce.

MCD global :



# Modèle Conceptuel de Données

## ◆ Normalisation

- La normalisation d'un MCD permet d'établir une conception **minimale** de la base de données.
- Un MCD normalisé doit vérifier 3 Formes Normales : 1FN, 2FN et 3FN.

### a) Première Forme Normale (1FN)

**Définition :** Pour toute occurrence d'une entité, respectivement relation, chaque propriété ne peut prendre qu'une seule valeur.

Le **MCD est dit en 1FN** si toutes ses entités et ses relations sont en 1FN.

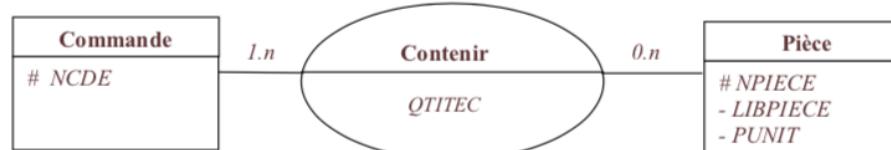
**Exemple 1 :** Passage en 1FN d'une entité non normalisée

Pièce
# NPIECE
- LIBPIECE
- PUNIT
- QTITEC

La commande N°1 contient 20 pièces de type 1  
La commande N°2 contient 30 pièces de type 1 }  $\Rightarrow$  L'entité « Pièce » n'est pas en 1FN

**Solution :**

Pour rendre l'entité « Pièce » en 1FN, il faut supprimer la QTITEC de « Pièce » et **créer** une *relation* entre « Pièce » et une nouvelle *entité* « Commande ».



# Modèle Conceptuel de Données

Exemple 2 : Passage en 1FN d'une relation non normalisée

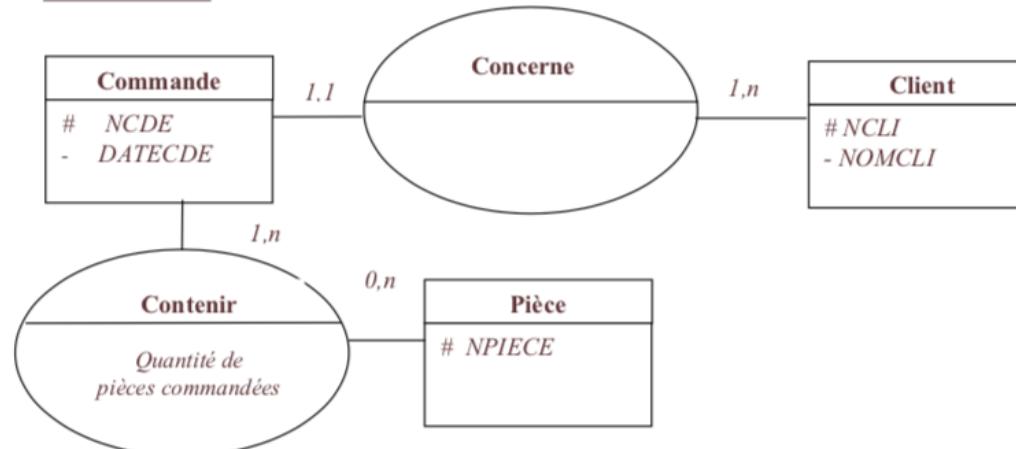


La propriété « *Quantité des pièces commandées* » peut prendre **plusieurs valeurs** pour NCDE et NCLI fixés  $\Rightarrow$  La relation « *Concerne* » n'est pas en 1FN

Solution :

Pour rendre la relation « *Concerne* » en première Forme Normale, il faut supprimer la « *Quantité des pièces commandées* » et **créer** une entité « *Pièce* » associé à « *Commande* » par une relation « *Contenir* ». Cette relation doit contenir la « *Quantité des pièces commandées* ».

MCD en 1FN :



## Modèle Conceptuel de Données

### b) Deuxième Forme Normale (2FN)

Définition : On dit qu'une entité ou relation est en 2FN si chaque propriété de cette entité, respectivement relation, dépend de son *identifiant* par une dépendance fonctionnelle élémentaire (il *n'existe pas* de DF avec une partie de l'identifiant).

Le **MCD** est dit en **2FN** si toutes ses entités et ses relations sont en 2FN.

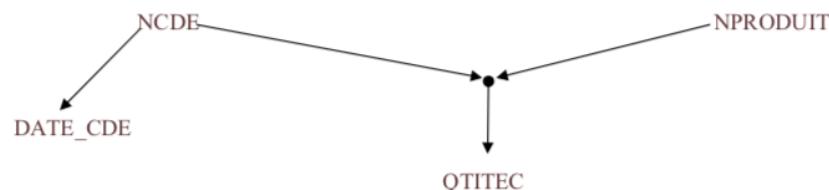
Exemple 1 : Passage en deuxième forme normale d'une entité en 1FN

Commande
# NCDE
# NPRODUIT
- DATE_CDE
- QTITEC

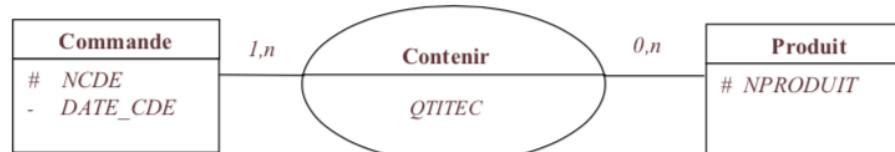
La date de commande *DATE\_CDE* dépend **seulement** de *NCDE* et non pas de *NCDE* et *NPRODUIT* à la fois  $\Rightarrow$  Commande n'est pas en 2FN

Solution :

Graphe de dépendances fonctionnelles :



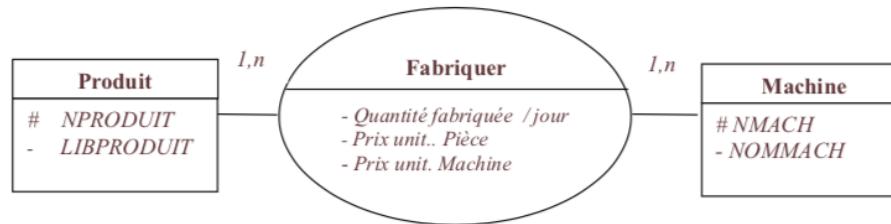
$\Rightarrow$  MCD correspondant (en 2FN) :



# Modèle Conceptuel de Données

Remarque : Toute entité en 1FN possédant une clé **simple** (non composée) est en 2FN.

Exemple 2 : Passage en 2FN d'une relation en 1FN

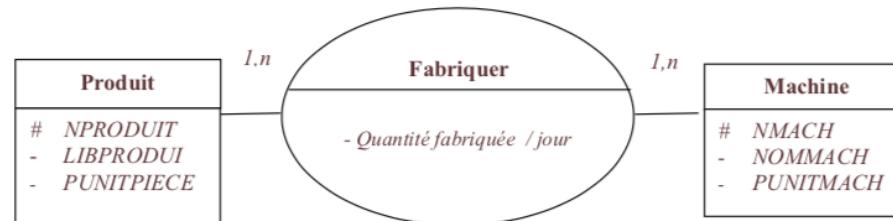


- La relation « *Fabriquer* » est en 1FN ;
- le « *Prix unit. Pièce* », respectivement le « *Prix unit. Machine* » dépend **seulement** de « *NPRODUIT* », respectivement « *NMACH* »  $\Rightarrow$  La relation « *Fabriquer* » n'est pas en 2FN.

Solution :

Pour rendre la relation « *Fabriquer* » en 2FN, il faut déplacer le « *Prix unit. Pièce* » vers l'entité « *Produit* » et le « *Prix unit. Machine* » vers l'entité « *Machine* ».

MCD en 2FN :



## Modèle Conceptuel de Données

### c) Troisième Forme Normale (3FN)

Définition : On dit qu'une entité ou relation est en 3FN si :

- Elle est en 2FN ;
- Toute les propriétés de cette entité, respectivement relation, dépend de son *identifiant* par une dépendance fonctionnelle élémentaire directe.

Exemple : Passage en troisième forme normale d'une entité en 2FN

Règle de gestion : Chaque Produit a un code de TVA ; chaque code identifie le taux de remise de TVA.

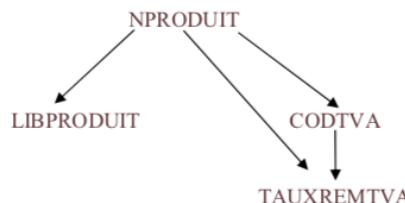
Produit
# NPRODUIT
- LIBPRODUI
- CODTVA
- TAUXREMTVA

- L'entité « Produit » est en 2FN,
- L'entité « Produit » n'est pas en 3FN. En effet :

$$\left. \begin{array}{l} NPRODUIT \rightarrow CODTVA \\ CODTVA \rightarrow TAUXREMTVA \end{array} \right\} \Rightarrow NPRODUIT \rightarrow TAUXREMTVA \text{ n'est pas directe.}$$

Solution :

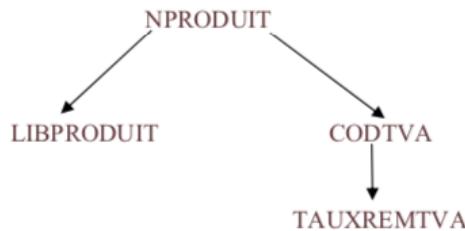
Graphe de DF associé :



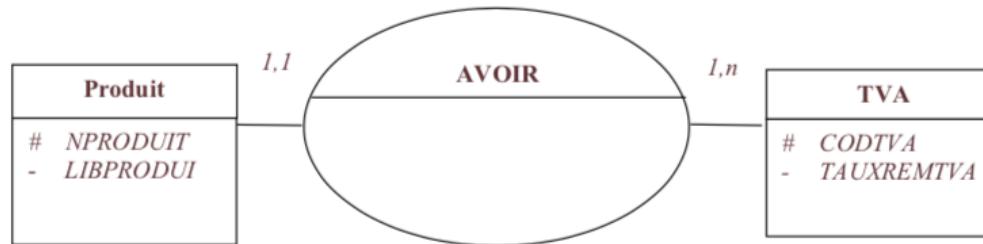
# Modèle Conceptuel de Données

Il faut **enlever** la *transitivité* entre NPRODUIT et TAUXREMTVA

⇒

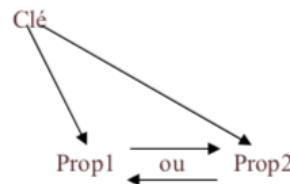


MCD en 3FN :



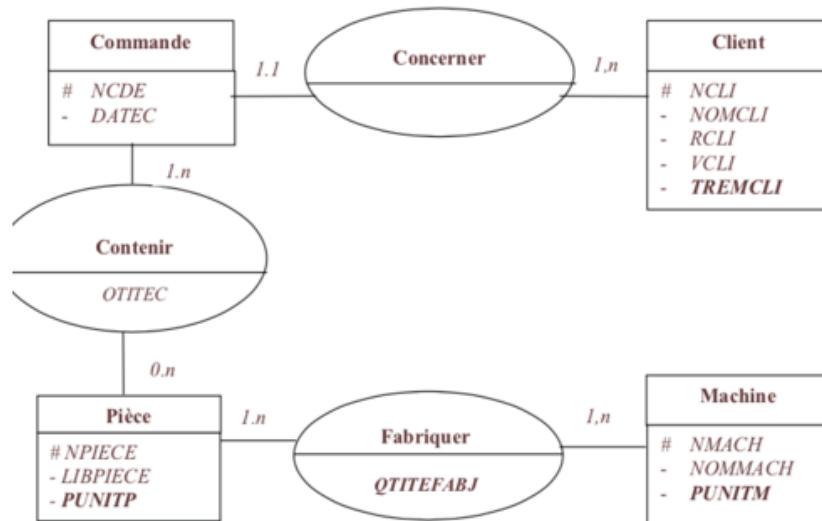
Remarque :

Une entité ou relation en **2FN** qui possède une **seule** propriété (en plus de sa clé) est en **3FN**. (Elle n'existe pas une deuxième propriété qui peut dépendre de la première ou inversement).



# Règles de passage MCD-MLD (1/3)

**Règle 1 :** Toute entité du MCD donne lieu à une table dont la clé primaire est l'identifiant de l'entité



## MLD

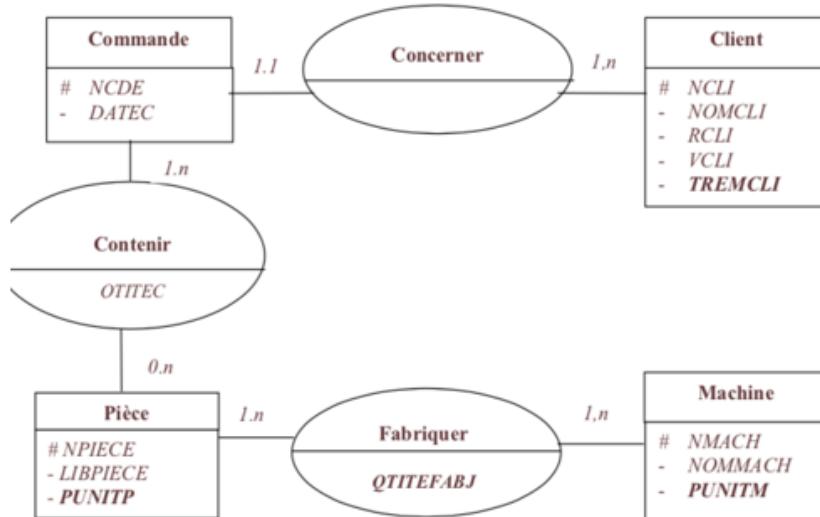
**Client**(NCLI, NOMCLI, RCLI, VCLI, TERMCLI)

**Machine**(NMACH, NOMMACH, PUNITM)

**Pièce**(NPIECE, LIBPIECE, PUNITP)

# Règles de passage MCD-MLD (2/3)

**Règle 2 :** Toute association avec des propriétés donne lieu à une table dont la clé primaire est composée des identifiants de toutes les entités reliées par cette association. Cette table comporte aussi comme attributs les propriétés l'association.



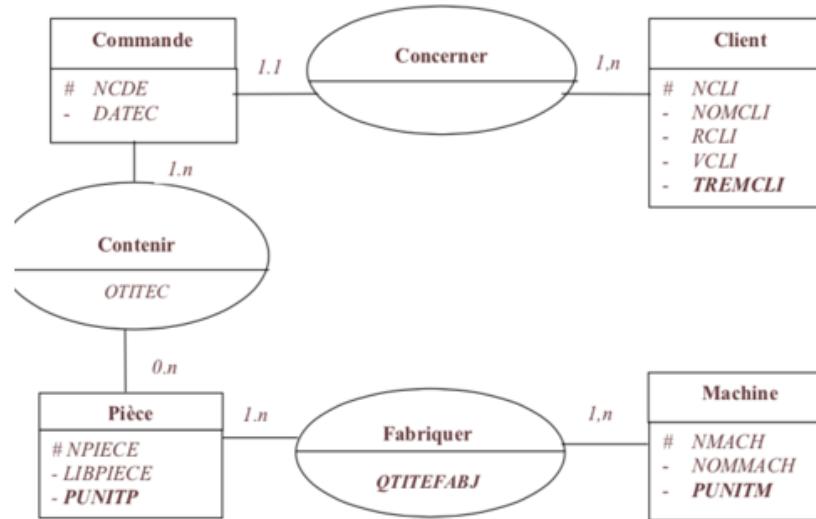
## MLD

**Contenir**(NCDE, NPIECE, QTITEC)

**Fabriquer**(NPIECE, NMACH, QTITEFABJ)

# Règles de passage MCD-MLD (3/3)

**Règle 3 :** Toute association binaire aux cardinalités (x,n) - (x,1) donne lieu à une duplication de la clé primaire de la table basée sur l'entité à cardinalité (x,n) dans la table basée sur l'entité à cardinalité (x,1). Cet attribut est appelé clé étrangère.



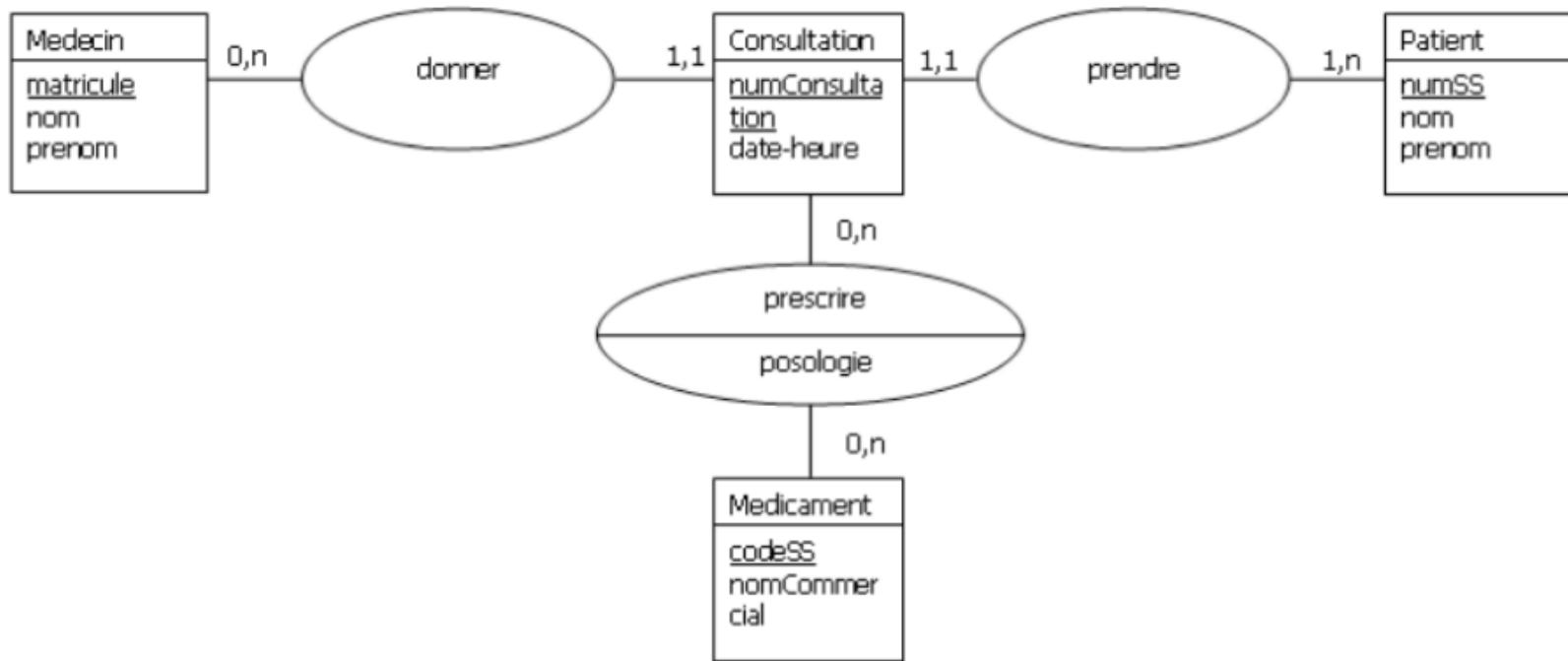
## MLD

**Commande**(NCDE, DATEC, #NCLI)

# Exercices

## Exercice 1 : Consultation médicale

On vous donne le MCD représentant des visites dans un centre médical.



1. Un patient peut-il effectuer plusieurs consultations ?
2. Un médecin peut-il recevoir plusieurs patients dans la même consultation ?
3. Peut-on prescrire plusieurs médicaments dans une même consultation ?
4. Deux médecins différents peuvent-ils prescrire le même médicament ?

# Exercices

## Exercice 2 : Médiathèque

On considère une médiathèque contenant des ouvrages pouvant être empruntés.

Un ouvrage est caractérisé par un numéro unique, un titre, un auteur et un éditeur. En outre, on décrit un ouvrage par un certain nombre de mots-clés qui indiquent les sujets qui y sont traités. La médiathèque dispose d'un ou plusieurs exemplaires de chaque ouvrage, L'exemplaire est identifié par un numéro et caractérisé par sa position dans les rayonnages et sa date d'achat.

Un exemplaire peut être emprunté par un emprunteur. Ces derniers sont identifiés par un numéro d'emprunteur et possèdent un nom et une adresse

Donner le MCD.

# Exercices

## Exercice 3 : Gestion d'un hôpital

On se propose de modéliser la base de données d'un hôpital. L'analyse de l'existant a dégagé les informations suivantes:

- L'hôpital a un ensemble d'employés qui sont des docteurs et des infirmières. Chaque employé possède un numéro d'employé, un nom, un prénom, une adresse et un numéro de téléphone.
- L'hôpital est composé de plusieurs services, pour lesquels on connaît le code, le nom, le bâtiment et le directeur, qui est en fait un docteur.
- Chaque service contient plusieurs salles. Une salle est représentée par un numéro, un surveillant et le nombre de lits qu'elle possède. Le numéro de salle est local à un service (i.e., chaque service possède une salle numéro 1). Un surveillant est un infirmier.
- Un infirmier est affecté à un service et à un seul.
- Les docteurs ne sont pas affectés à un service particulier, mais on connaît sa spécialité.
- On connaît aussi pour chaque infirmier sa rotation et son salaire.
- Les malades de l'hôpital sont représentés par un numéro, un nom, un prénom, une adresse et un numéro de téléphone.
- Un malade est hospitalisé dans une salle avec un numéro de lit et son diagnostic. Il est soigné par un docteur. Au cas où il y a des complications, il peut être transféré dans un autre service avec une autre salle.

## **ANNEXE**

Résumé de quelques commandes MySQL importantes

<http://www.w3schools.com/sql>

Gestion des bases de données	Créer une BD (sur Cogito <i>nomDeLaDB</i> = tixxx)	<b>CREATE DATABASE</b> <i>nomDeLaBD</i> ;
	Afficher les bases de données existantes	<b>SHOW DATABASES;</b>
	Utiliser une base de données	<b>USE</b> <i>nomBD</i> ;
	Effacer une base de données	<b>DROP DATABASE</b> <i>nomDeLaBD</i> ;
Gestion des tables	Créer une table	<b>CREATE TABLE</b> <i>nomDeLaTable</i> ( <i>nomDeLaColonne TypeDeValeurs AttributDesValeurs</i> , <i>nomDeLaColonne TypeDeValeurs AttributDesValeurs</i> , ... );
	Afficher les tables existantes	<b>SHOW TABLES;</b>
	Afficher la structure d'une table	<b>DESCRIBE</b> <i>nomDeLaTable</i> ;
	Modifier une table	<b>ALTER TABLE</b> <i>nomDeLaTable</i> <b>RENAME AS</b> <i>nouveauNomDeLaTable</i> ; <b>ALTER TABLE</b> <i>nomDeLaTable</i> <b>ADD</b> <i>nomDeLaColonne TypeDeValeurs</i> ; <b>ALTER TABLE</b> <i>nomDeLaTable</i> <b>CHANGE</b> <i>nomDeLaColonne nouveauNomDeLaColonne TypeDeValeurs</i> ; <b>ALTER TABLE</b> <i>nomDeLaTable</i> <b>DROP COLUMN</b> <i>nomDeLaColonne</i> ;
	Effacer une table	<b>DROP TABLE</b> <i>nomDeLaTable</i> ;
	Ajouter des données	<b>INSERT INTO</b> <i>nomDeLaTable</i> <b>VALUES</b> ( <i>valeur1, valeur2, ...</i> ); <b>INSERT INTO</b> <i>nomDeLaTable</i> ( <i>nomDeLaColonne1, nomDeLaColonne2, ...</i> ) <b>VALUES</b> ( <i>valeur1, valeur2, ...</i> );
Gestions des données	Afficher des données	<b>SELECT</b> <i>nomDeLaColonne1, nomDeLaColonne2, ...</i> <b>FROM</b> <i>nomDeLaTable</i> <b>WHERE</b> <i>certainnesColonnes=CertainesValeurs</i> <b>GROUP BY</b> <i>nomDeLaColonne</i> <b>ORDER BY</b> <i>nomDeLaColonne AttributD'ordre</i> ;
	Mises à jour de données	<b>UPDATE</b> <i>nomDeLaTable</i> <b>SET</b> <i>nomDeLaColonne1=valeur1, nomDeLaColonne2=valeur2, ...</i> <b>WHERE</b> <i>certainnesColonnes=CertainesValeurs</i> ;
	Modifier l'affichage du nom de colonne	<b>SELECT ... FROM ... AS</b> <i>nomAlias</i> ;
	Recherche d'un pattern. (Retourne ici ce qui commence par 'a')	<b>SELECT ... FROM ... WHERE ... LIKE 'a%';</b>
	Effacer des données	<b>DELETE FROM</b> <i>nomDeLaTable</i> <b>WHERE</b> <i>certainnesColonnes=CertainesValeurs</i> ;

<b>Types de valeurs</b>	Entiers, Flottant, Double	<b>INT, FLOAT, DOUBLE</b>
	Chaîne de caractères à longueur variable	<b>VARCHAR(<i>nombreDeCaractère</i>)</b>
	Liste	<b>ENUM('élément1', 'élément2', ...)</b>
	Date (format aaa-mm-jj)	<b>DATE</b>
	Heure (format hh:mm:ss)	<b>TIME</b>
	Date et Heure (mis à jours lorsque l'enregistrement est modifié)	<b>TIMESTAMP</b>
<b>Attribut des valeurs</b>	Le champ ne peut pas être laissé vide	<b>NOT NULL</b>
	Le champ peut être laissé vide	<b>NULL</b>
	Valeurs positives	<b>UNSIGNED</b>
	Assignation automatique d'une valeur	<b>AUTO_INCREMENT</b>
	Clé primaire	<b>PRIMARY KEY</b>
	Valeurs par défaut	<b>DEFAULT '<i>Texte</i>'</b>
<b>Attribut d'ordre</b>	Ascendant	<b>ASC</b>
	Descendant	<b>DESC</b>
<b>Opérateurs</b>	Opérateurs	<b>AND, OR, NOT, =, !=, &lt;, &gt;, +, -, *, /</b>
<b>Fonctions</b>	Fonctions mathématiques	<b>SUM(<i>nomDeLaColonne</i>), MIN(<i>nomDeLaColonne</i>), MAX(<i>nomDeLaColonne</i>), AVG(<i>nomDeLaColonne</i>), COUNT(<i>nomDeLaColonne</i>), COUNT(*)</b>
	Fonction génératrices de dates	<b>NOW(), CURDATE(), CURRENT_DATE(), CURTIME()</b>
	Fonctions pour extraire de l'information à partir des dates	<b>YEAR(<i>uneDate</i>), MONTH(<i>uneDate</i>), DAYOFMONTH(<i>uneDate</i>), TO_DAYS(<i>uneDate</i>)</b>
	Fonctions de concaténation	<b>CONCAT(<i>nomDeLaColonne1</i>, <i>nomDeLaColonne2</i>, ...)</b>