



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

EMSI-Rabat 3IIR

2023-2024

Equipe pédagogique

Prof. Mahmoud NASSAR

Prof. Mohammed SALIHOUN

Prof. Maria EL HAIBA

- Partie 1 -

Partie 1 : Le langage SQL

Introduction

Le langage de définition des données (LDD)

CREATE
ALTER
DROP

Le langage de manipulation des données (LMD)

SELECT
INSERT
UPDATE
DELETE

Le langage de contrôle des données (LCD)

GRANT
REVOKE
COMMIT
ROLLBACK

Le langage de manipulation des données (LMD)

SELECT

INSERT

UPDATE

DELETE

Syntaxe simplifiée

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

Condition

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 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 ;
```

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

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 ;
```

Exemples

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

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

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

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**

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.

Examples

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

EMP	Deptno	sal
	10	2600
	20	2175
	30	1566.7

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

```
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

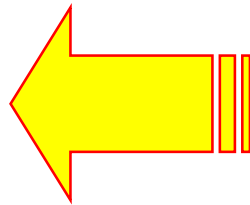
GROUP BY avec HAVING

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

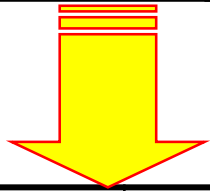

Exemple

**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

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

Requêtes sur plusieurs tables

Les opérateurs de jointures

L'imbrication de requêtes

Les opérateurs ensemblistes

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*)


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
```



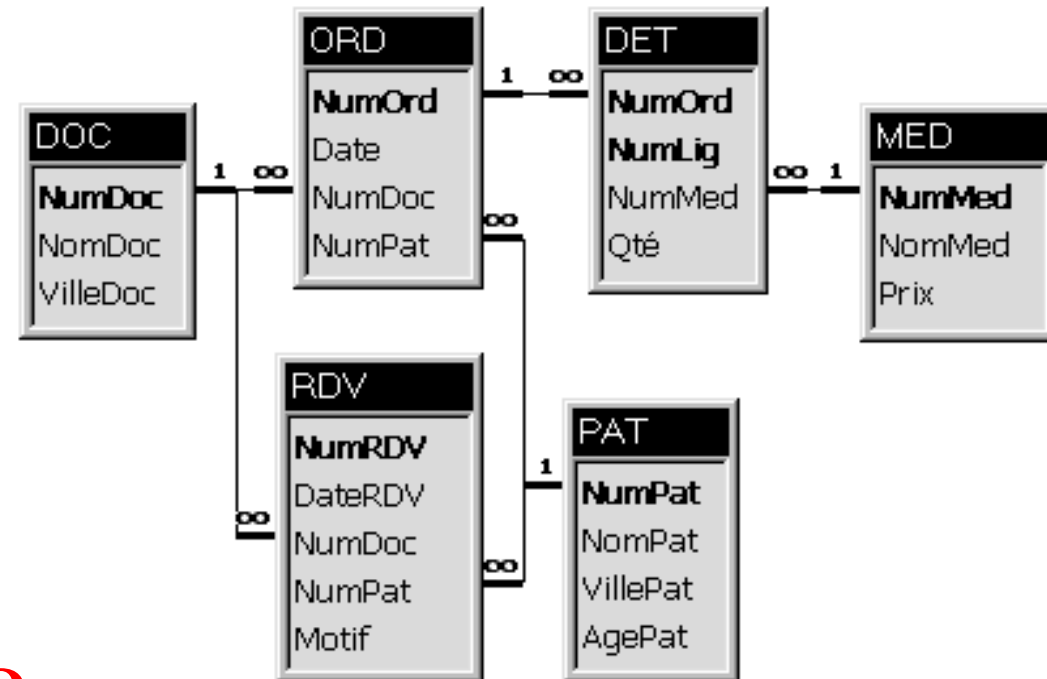
tab1			col1

tab2			col2

Equijointure (*jointure naturelle*)

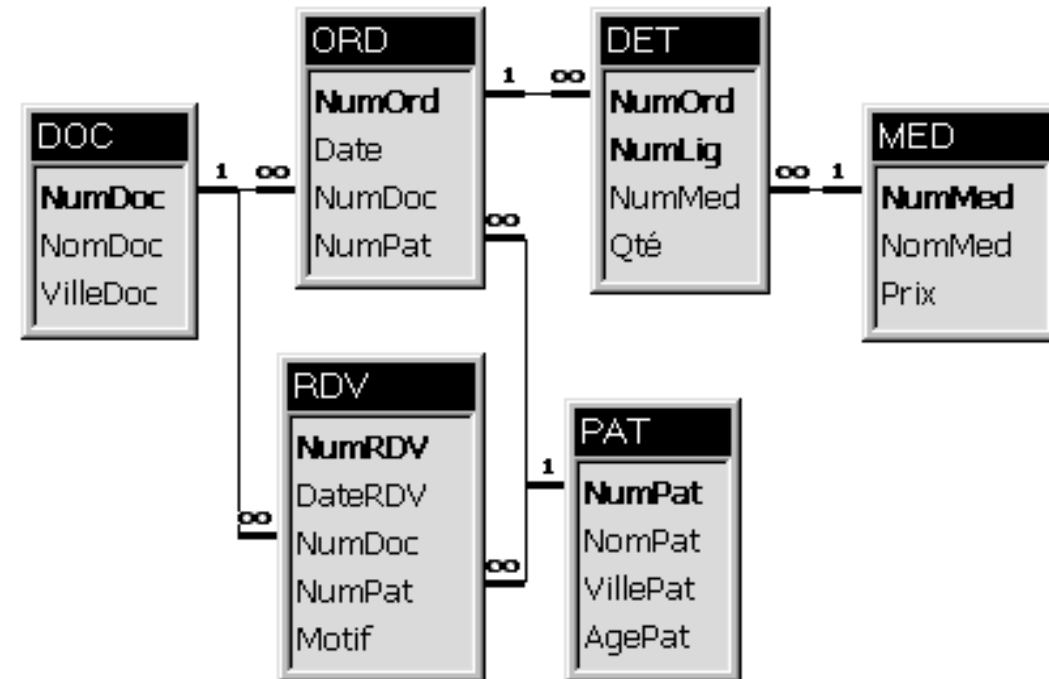
Liste des RDV
avec le docteur 'Alaoui'

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



Equijointure (*jointure naturelle*)

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



SELECT
FROM
WHERE

PAT.NomPat

PAT , RDV , DOC

PAT.NumPat = RDV.NumPat

and RDV.NumDoc = DOC.NumDoc

and DOC.NomDoc = 'Alaoui';

Autojointure

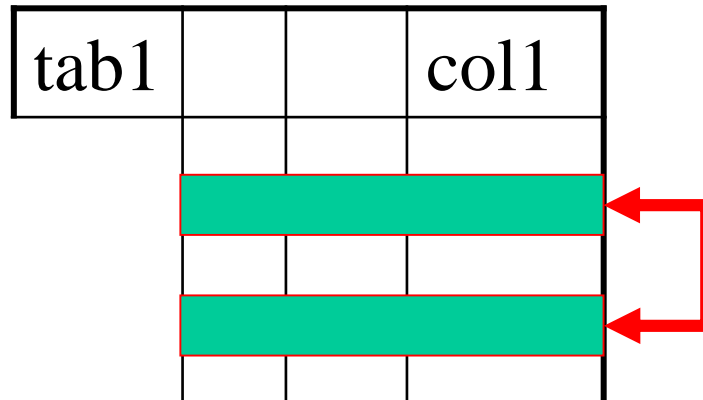
Liste des employés ayant un salaire égale à celui de « Azhari »

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

Autojointure

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



Autojointure

Liste des employés ayant un salaire égal 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

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';
```

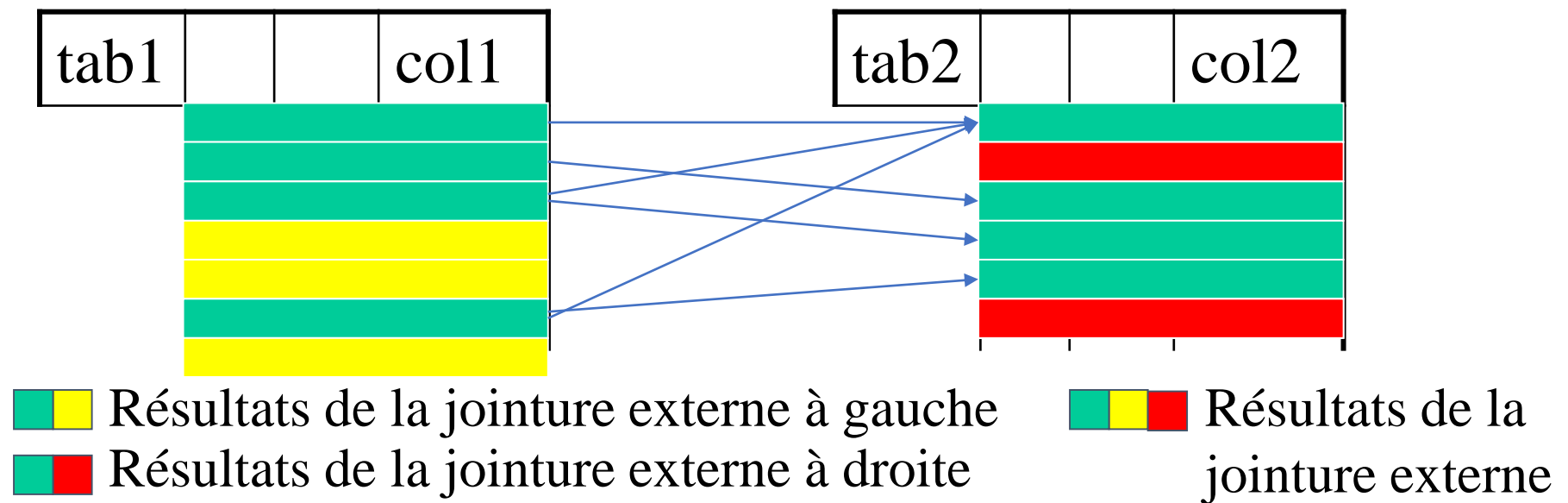
EMP	Deptno	NOM	sal
	10	Alaoui	5000
	10	Filali	1500
	10	Rachidi	1250
	20	Tahiri	2975
	20	Rochdi	3000
	20	Ouazzani	1100
	30	Zohri	2850
	30	Azhari	1250
	30	Taouil	1600
	30	Rbati	1500
	30	Andaloussi	950
	30	Soussi	1250

Requêtes sur plusieurs tables: la jointure externe

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

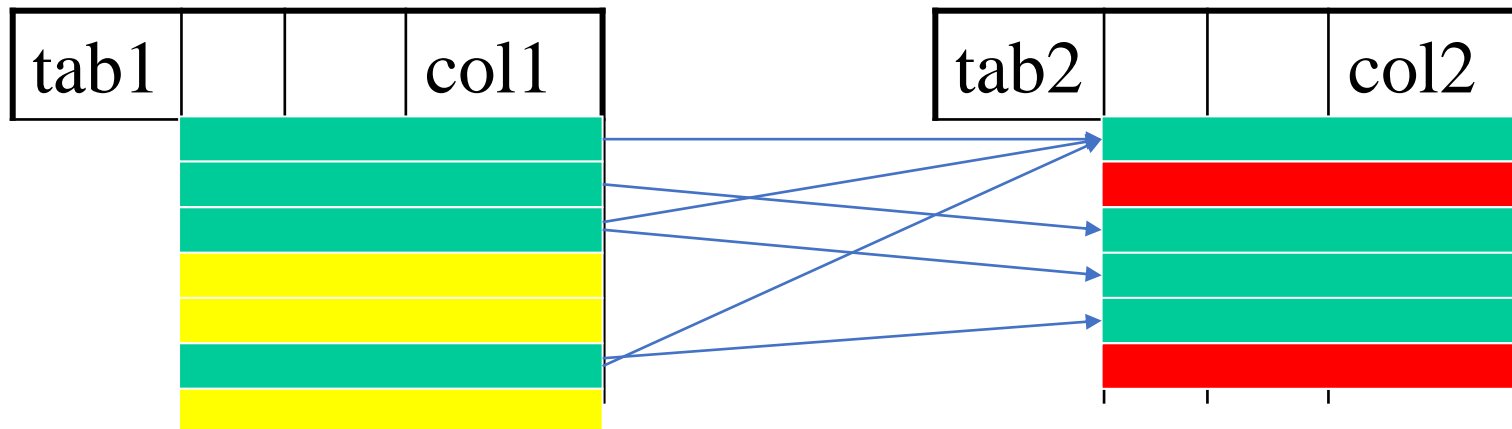


Requêtes sur plusieurs tables: la jointure externe

Jointure Externe

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

SQL standard



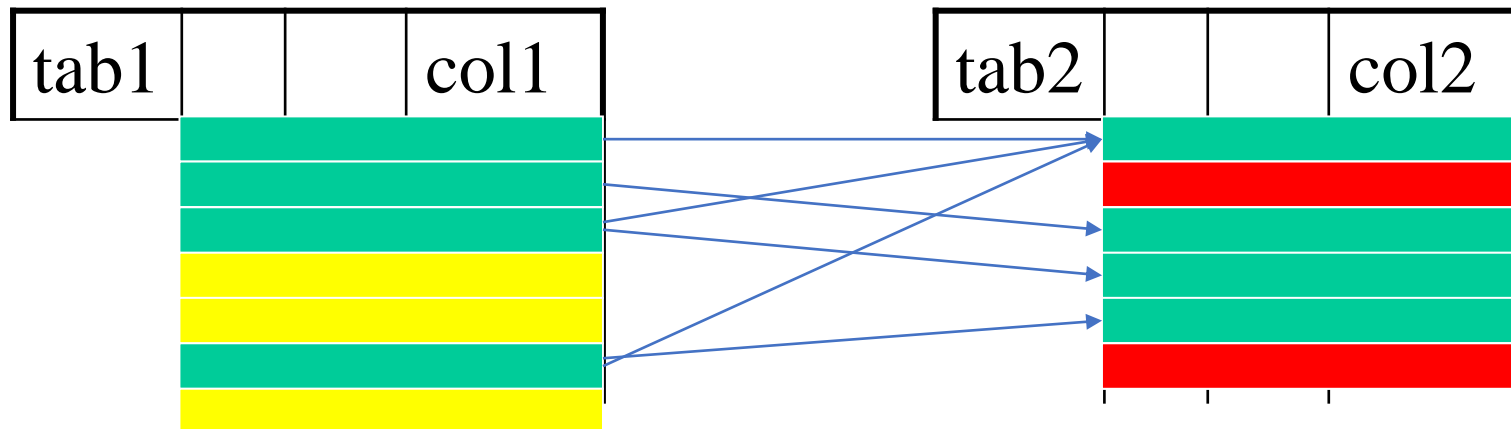
Résultats de la jointure externe

Requêtes sur plusieurs tables: la jointure externe

Jointure Externe à gauche

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

SQL standard



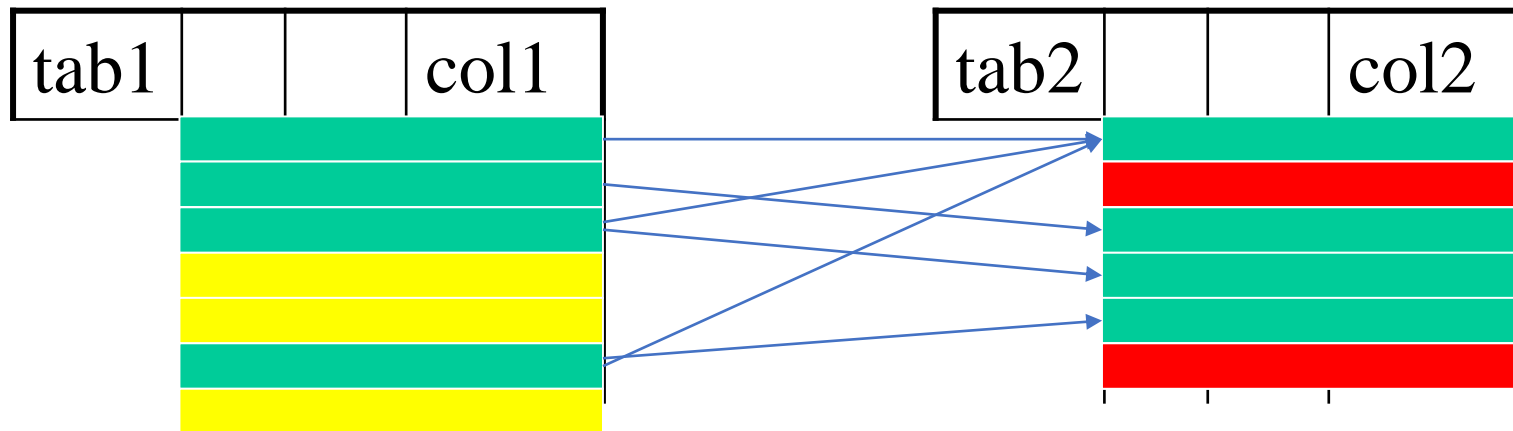
 Résultats de la jointure externe à gauche

Requêtes sur plusieurs tables: la jointure externe

Jointure Externe à gauche

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

Oracle



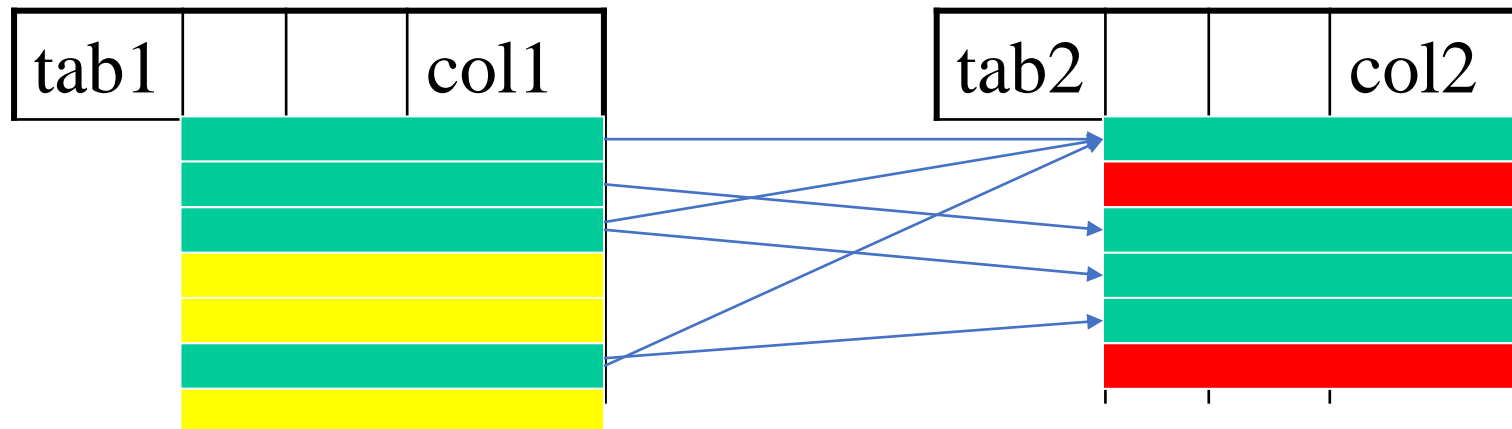
 Résultats de la jointure externe à gauche

Requêtes sur plusieurs tables: la jointure externe

Jointure Externe à droite

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

SQL standard



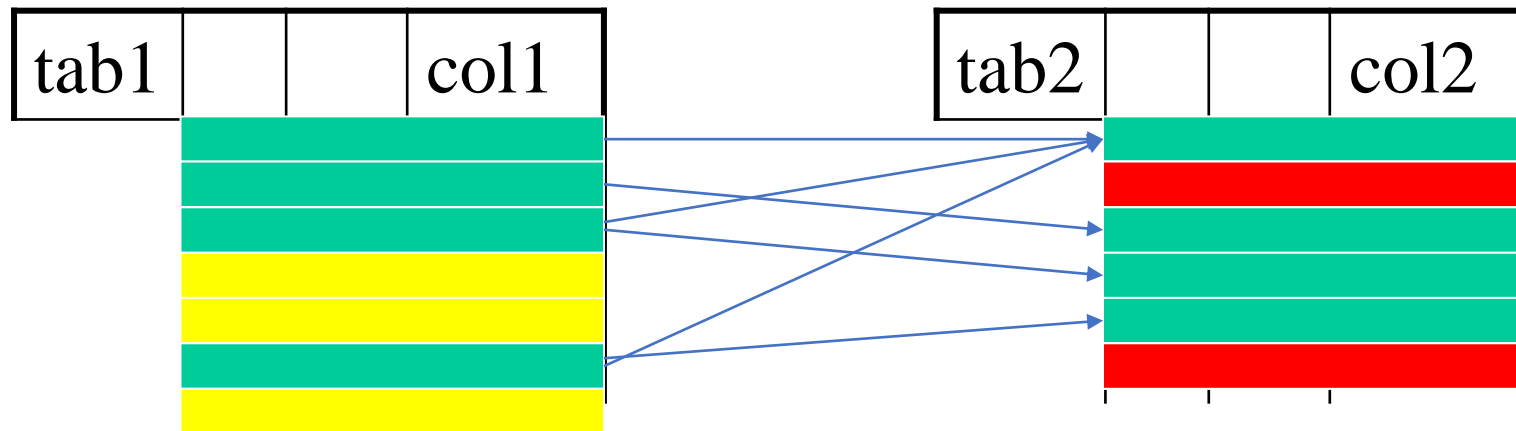
 Résultats de la jointure externe à droite

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
	Alaoui	11
	Filali	22
	Rachidi	11
	Azhari	NULL

DEP	depno	depnom
	11	RD
	30	Compta
	22	Etudes

 Résultats de la jointure externe à droite

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

EMP	nom	depno
	Alaoui	11
	Filali	22
	Rachidi	11
	Azhari	NULL

DEP	depno	depnom
	11	RD
	30	Compta
	22	Etudes

 Résultats de la jointure externe à gauche

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

Requêtes imbriquées

Syntaxe générale

sens d'exécution ↑

```
SELECT colonnes_de_projection  
FROM table  
WHERE expr operator (  
    SELECT colonnes_de_projection  
    FROM table  
    WHERE .....  
);
```

Remarque:

Pas de ORDER BY ou UNION dans la sous requête

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

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 );
```

Le nom de l'employé le mieux payé?

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

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

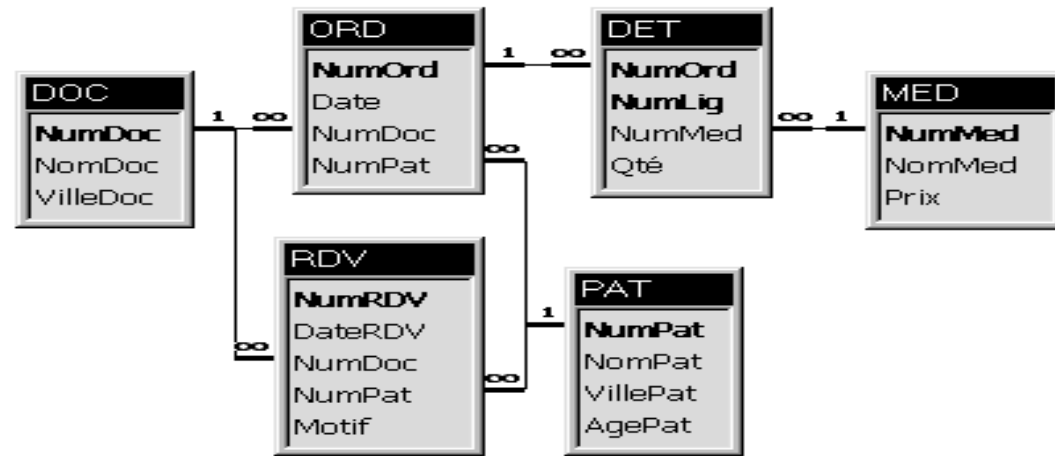
Select * from DOC

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

Avec EXISTS :

Select * from DOC **D**

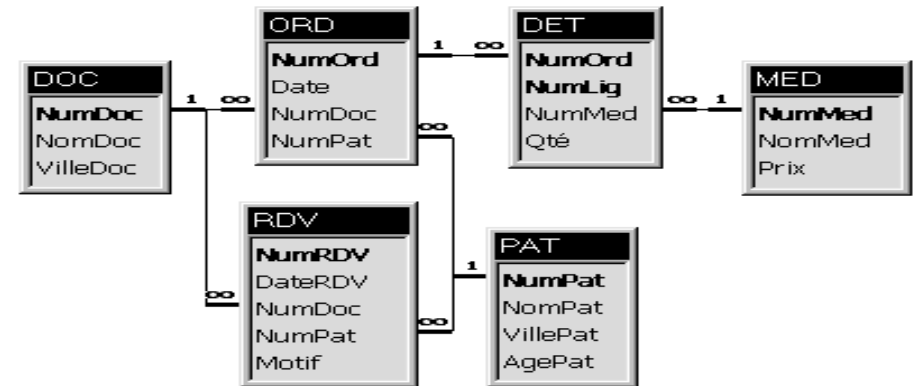
Where NOT EXISTS (select *
from RDV R
where dateRDV Between
'01/01/2009' and '31/12/2009'
AND R.NumDOC=**D**.numDoc
);



Exercice

Q1 : Les numéros d'ordonnances et leur montant total

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

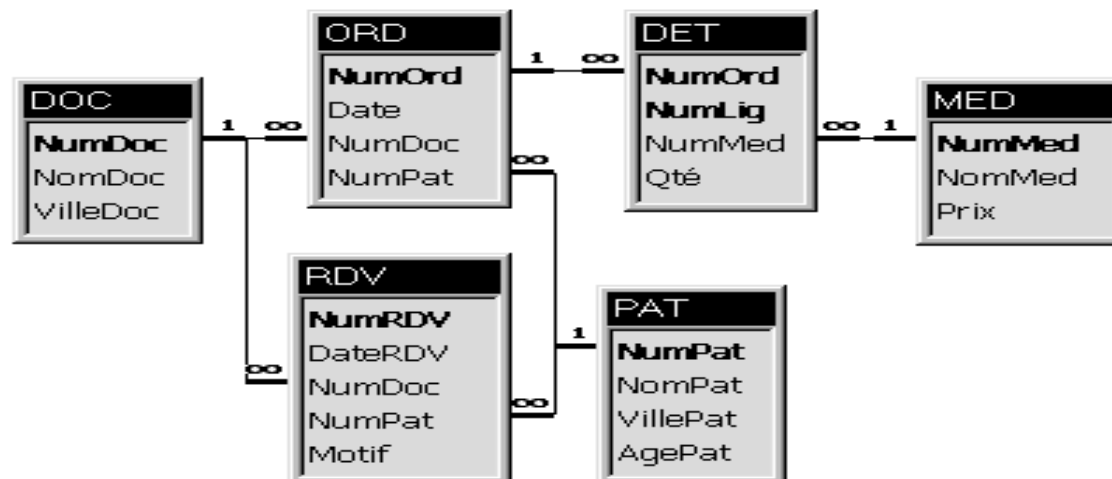
Q4 : Patients sans RDV en 2019

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 médicaments le plus prescrit en 2019.



Opérateurs ensemblistes

INTERSECT
UNION
UNION ALL
MINUS

Requête SELECT
<Opérateur ensembliste>
Requête SELECT

Le langage de définition des données (LDD)

CREATE
ALTER
DROP

Le langage de manipulation des données (LMD)

SELECT
INSERT
UPDATE
DELETE

Le langage de contrôle des données (LCD)

GRANT
REVOKE
COMMIT
ROLLBACK

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;
```

Syntaxe UPDATE

```
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 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  
                    );
```

Le langage de définition des données (LDD)

CREATE
ALTER
DROP

Le langage de manipulation des données (LMD)

SELECT
INSERT
UPDATE
DELETE

Le langage de contrôle des données (LCD)

GRANT
REVOKE
COMMIT
ROLLBACK

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
```

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] ...) ;
```

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

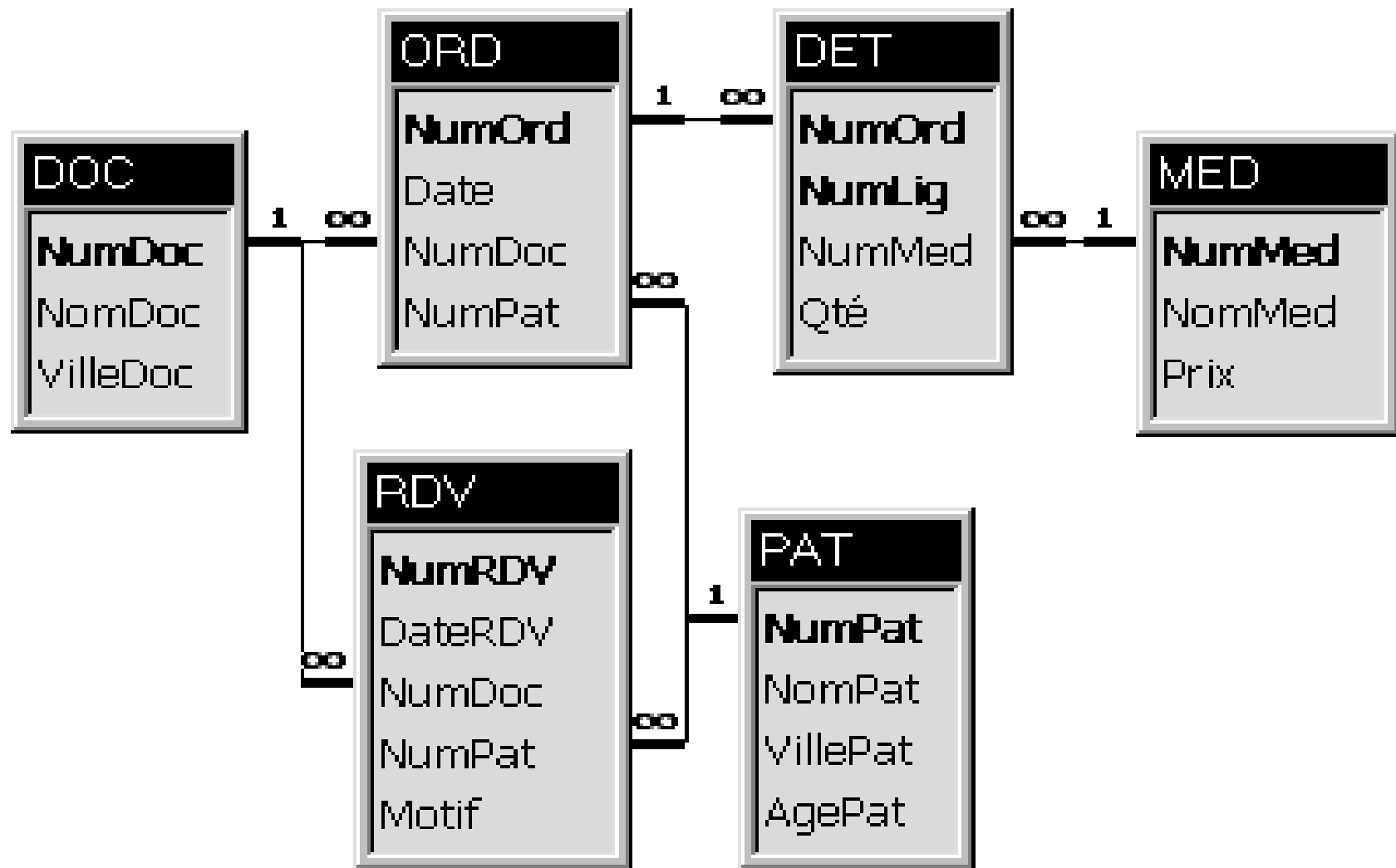
```
[ CONSTRAINT <nom de la contrainte>  
    [  
      UNIQUE (liste de colonnes) |  
  
      PRIMARY KEY (liste de colonnes) |  
  
      CHECK (condition) |  
  
      FOREIGN KEY (liste de colonnes)  
      REFERENCES <nom de la table> (liste colonnes) [<mode>]  
    ]  
]
```

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

Remarque:

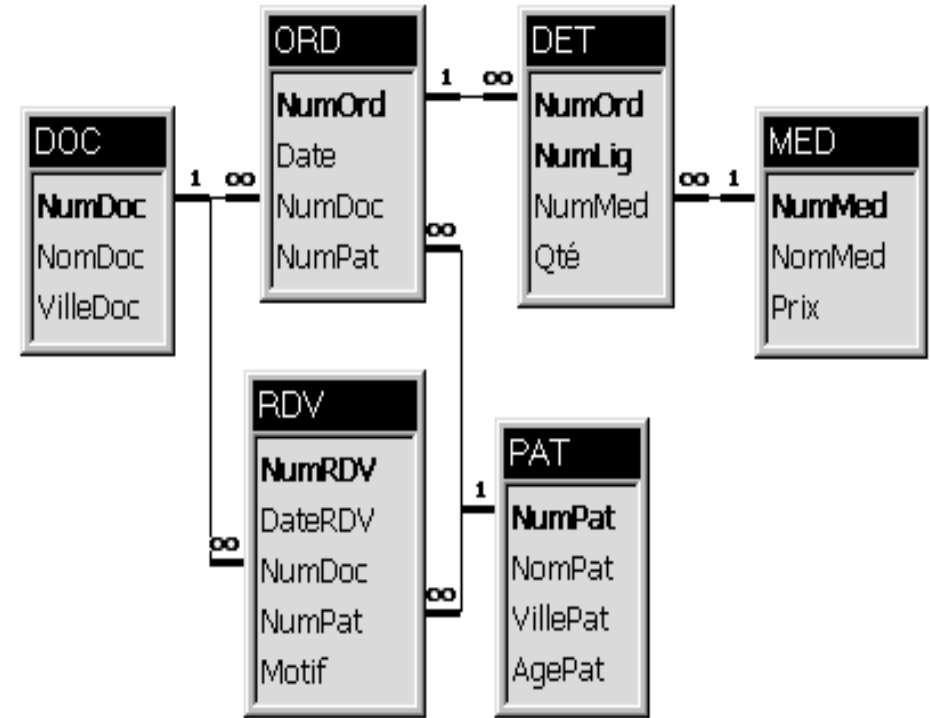
Sur la colonne ou la table: si la contrainte ne fait intervenir qu'un
SEUL ATTRIBUT.

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

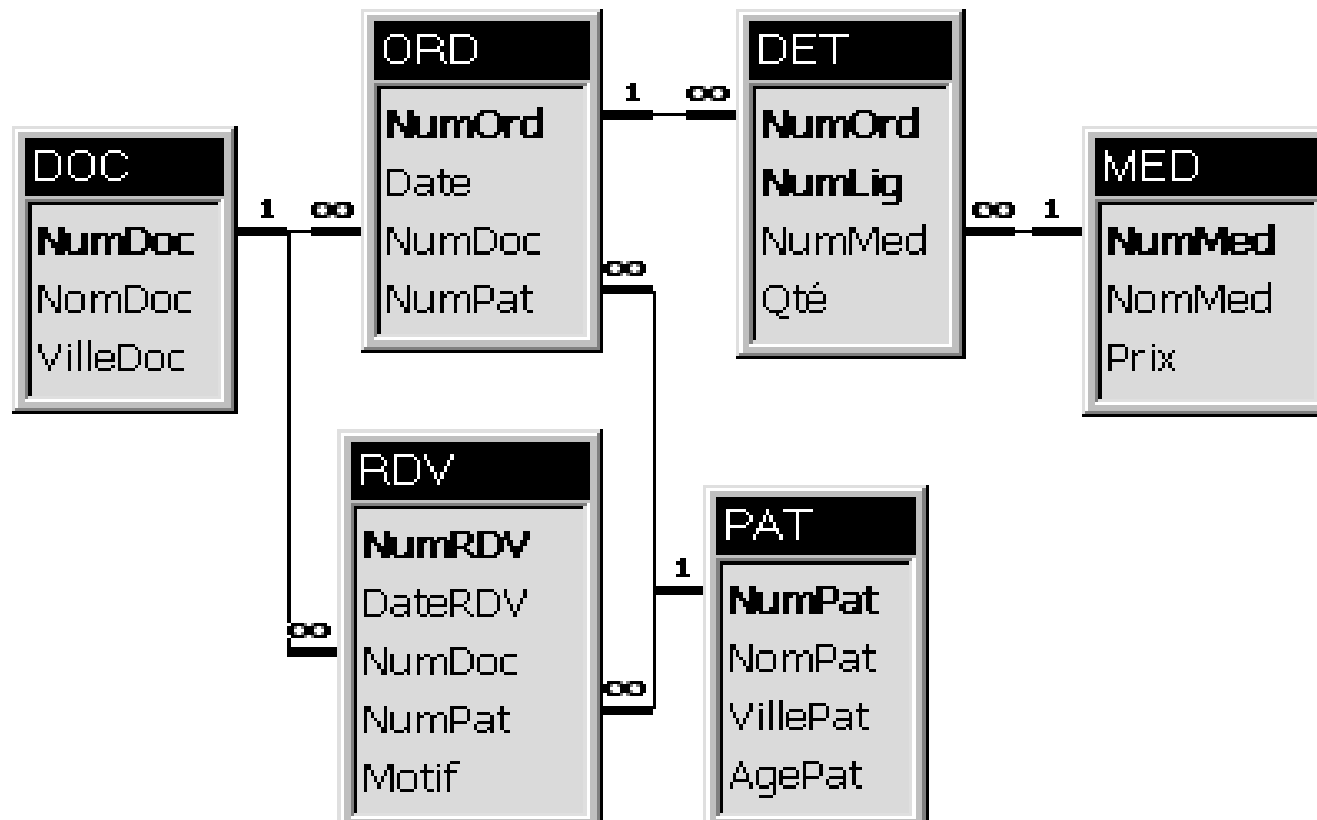


Create Table DOC(
 NumDOC integer **PRIMARY KEY**,
 NomDOC VARCHAR2(20),
 VilleDOC VARCHAR2(20)
);

Create Table DOC(
 NumDOC Number(4),
 NomDOC VARCHAR2(20),
 VilleDOC VARCHAR2(20),
Constraint PK_DOC Primary Key (NumDOC)
);



Create Table DOC(NumDOC Number(4),NomDOC VARCHAR2(20), VilleDOC VARCHAR2(20), **Constraint PK_DOC Primary Key (NumDOC)**);



Create Table DOC(NumDOC Number(4),NomDOC VARCHAR2(20), VilleDOC VARCHAR2(20), **Constraint PK_DOC Primary Key (NumDOC)**);

Create Table PAT(NumPat Number(4), NomPat VARCHAR2(20),VillePat VARCHAR2(20), AgePat Number(3), **Constraint PK_Pat Primary Key (NumPat)**);

Create Table RDV(NumRDV Number(4), DateRDV Date, NumDoc Number(4), NumPat Number(4), Motif Varchar2(20), **Constraint PK_RDV Primary Key (NumRDV)**, **Constraint Ref_RDV_Doc Foreign Key(NumDoc) References DOC(NumDoc)**, **Constraint Ref_RDV_PAT Foreign Key(NumPat) References PAT(NumPat));**

Create Table ORD(NumOrd Number(4), DateOrd Date, NumDoc Number(4), NumPat Number(4), **Constraint PK_ORD Primary Key (NumORD)**, **Constraint Ref_Ord_Doc Foreign Key(NumDoc) References DOC(NumDoc)**, **Constraint Ref_Ord_PAT Foreign Key(NumPat) References PAT(NumPat));**

Create Table MED(NumMed Number(4), NomMed VARCHAR2(20), Prix Number(8,2), **Constraint PK_Med Primary Key (NumMed)**);

Create Table DET(NumORD Number(4), NumLigne Number(4), NumMed Number(4), Qte Number(4), **Constraint PK_DET Primary Key (NumORD,NumLigne)**, **Constraint Ref_Det_Ord Foreign Key(NumOrd) References ORD(NumOrd)**, **Constraint Ref_Det_Med Foreign Key(NumMed) References MED(NumMed)**);

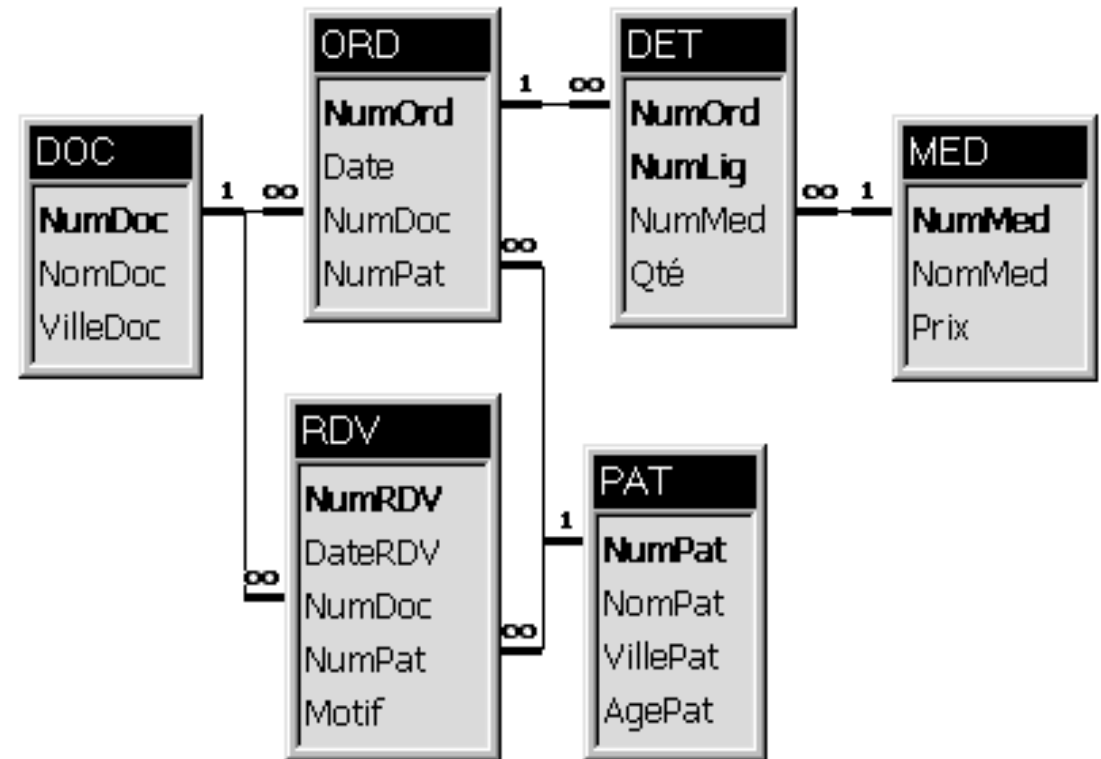
Create Table DOC(

NumDOC integer **Constraint PK_DOC PRIMARY KEY,**

NomDOC VARCHAR2(20),

VilleDOC VARCHAR2(20)

);



Create Table DET(

NumORD NUMBER(5),

NumLigne NUMBER(2),

NumMED NUMBER(4),

QTE Number(3) **Not Null**,

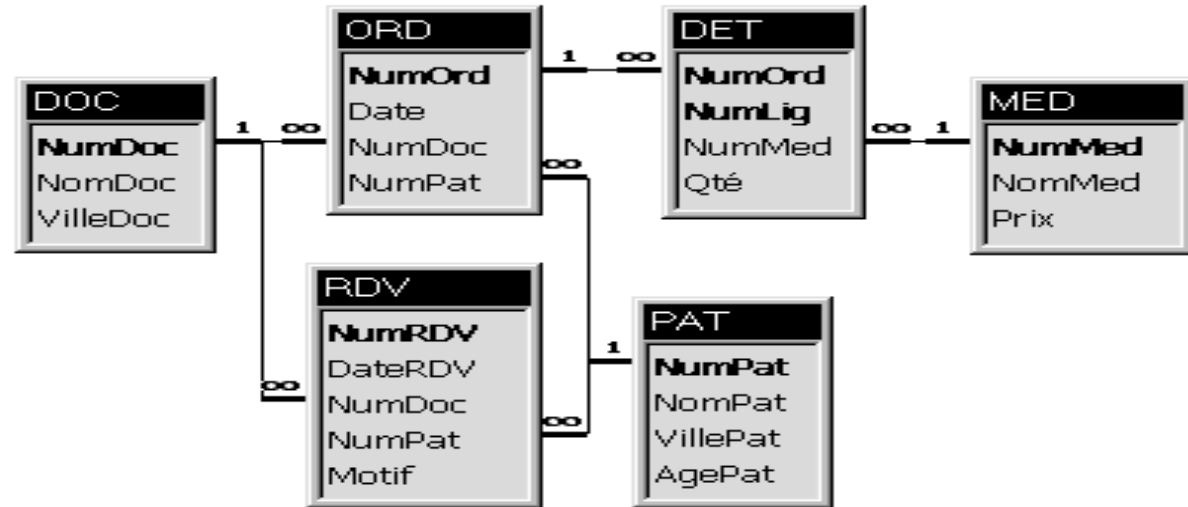
Constraint PK_DET Primary Key (**NumORD**, **NumLigne**),

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

Constraint Ref_DET_ORD Foreign Key (**NumORD**) References **ORD**(**NumORD**),

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

);



```
SELECT CONSTRAINT_NAME, CONSTRAINT_TYPE, TABLE_NAME
FROM USER_CONSTRAINTS;
```

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

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;

Syntaxe DROP TABLE

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

Exemple:

```
Drop table DOC;
```

LES VUES

LES VUES:

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

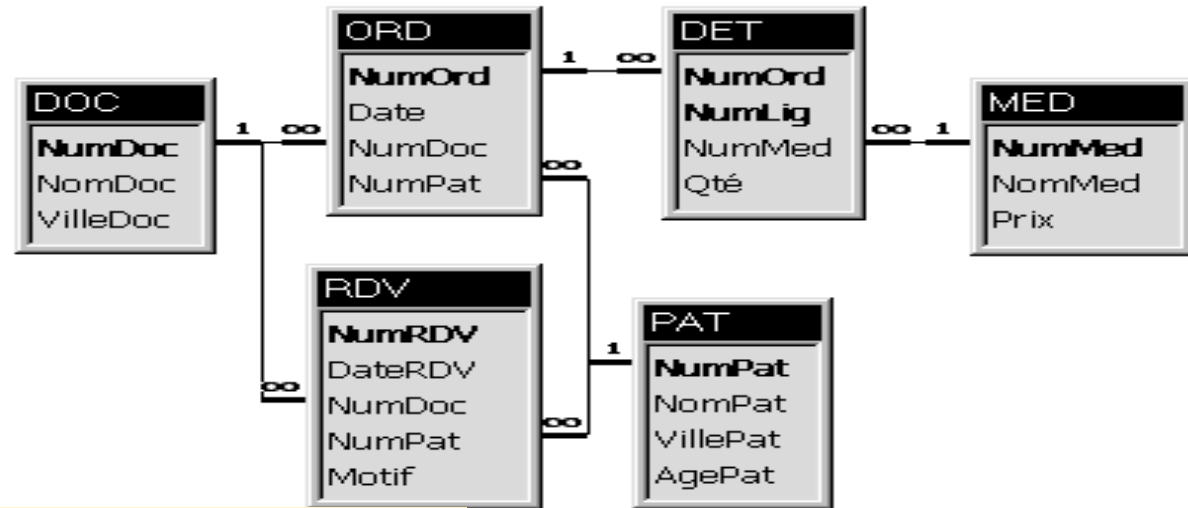
Syntaxe de CREATE VIEW

```
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.

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;
```

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, INTERSETC, MINUS	OUI	NON	NON	NON

Fin