Langage d'Interrogation des Données LID

Introduction
Sélection ou restriction
- ✓ L'opérateur LIKE
- ✓ L'opérateur IS NULL

Projection
- ✓ L'opérateur DISTINCT

Alias (Renommage)
Tri
Jointure
- ✓ Jointure naturelle
- ✓ θ-jointure
- ✓ Produit cartésien
- ✓ Equijointure
- ✓ Jointure externe droite
- ✓ Jointure externe gauche
- ✓ Jointure externe
- ✓ Auto-jointure

INTRODUCTION

- Le Langage d'Interrogation des Données **LID** permet d'exprimer **toutes les questions que l'on peut poser à une base de données;**
- La forme générale d'une requête d'interrogation est la suivante:

```
SELECT Clause_projection
FROM Tables_Sources
WHERE Clause_restriction/Clause_jointure
GROUP BY Clause_regroupement_resultat_agregat
HAVING Clause_restriction_resultats_agregat
ORDER BY Clause_tri;
```

- **L'ordre des clauses est important;**
- **SELECT et FROM sont obligatoires.**

SÉLECTION OU RESTRICTION

Rappel:

La sélection (ou restriction) d'une table R par une qualification Q est une table R' de même structure dont les lignes sont ceux de R qui satisfont à la qualification.

Syntaxe:

```
SELECT *
FROM nom_table
WHERE Exp1  op  Exp2 ...;
```

Expression sous forme : < colonne> <opérateur> <valeur>

Connecteurs d'expressions: AND (et logique), OR (ou logique)

Opérateurs de comparaison: >, >=, <, <=, =, !=

Opérateurs intégrés: BETWEEN, LIKE, IN, NOT IN, IS NULL, NOT NULL

EXEMPLE 1

Soit l'extension de la table Etudiant:

NumInscrEt	NomEt	PreEt	VilleEt	AnneeNaisEt
10001	Bakkali	Ahmed	Safi	2000
10002	Benali	Rachid	Rabat	1999
10003	Chadid	Fadoua	Safi	2000
10004	Hajjaji	Hind	El Jadida	1998
10005	Himich	Ahmed	Marrakech	1998

Requête pour afficher l'étudiant qui porte le **NumInscrEt**10004:

```
SELECT *
FROM Etudiant
WHERE NumInscrEt=10004;
```

Donne le résultat suivant:

NumInscrEt	NomEt	PreEt	VilleEt	AnneeNaisEt
10004	Hajjaji	Hind	El Jadida	1998

EXEMPLE 2

Soit l'extension de la table Etudiant:

NumInscrEt	NomEt	PreEt	VilleEt	AnneeNaisEt
10001	Bakkali	Ahmed	Safi	2000
10002	Benali	Rachid	Rabat	1999
10003	Chadid	Fadoua	Safi	2000
10004	Hajjaji	Hind	El Jadida	1998
10005	Himich	Ahmed	Marrakech	1998

Requête pour afficher les étudiants qui sont nés entre 1999 et 2002

```
SELECT *
FROM Etudiant
WHERE AnneeNaisEt >= 1999
AND AnneeNaisEt <=2002;
```

Donne le résultat suivant:

NumInscrEt	NomEt	PreEt	VilleEt	AnneeNaisEt
10001	Bakkali	Ahmed	Safi	2000
10002	Benali	Rachid	Rabat	1999
10003	Chadid	Fadoua	Safi	2000

EXEMPLE 3

Soit l'extension de la table Etudiant:

NumInscrEt	NomEt	PreEt	VilleEt	AnneeNaisEt
10001	Bakkali	Ahmed	Safi	2000
10002	Benali	Rachid	Rabat	1999
10003	Chadid	Fadoua	Safi	2000
10004	Hajjaji	Hind	El Jadida	1998
10005	Himich	Ahmed	Marrakech	1998

Requête pour afficher les étudiants qui sont nés entre 1999 et 2002

```
SELECT *
FROM Etudiant
WHERE AnneeNaisEt
BETWEEN 1999 AND 2002;
```

Donne le résultat suivant:

NumInscrEt	NomEt	PreEt	VilleEt	AnneeNaisEt
10001	Bakkali	Ahmed	Safi	2000
10002	Benali	Rachid	Rabat	1999
10003	Chadid	Fadoua	Safi	2000

REMARQUE

```
SELECT *
FROM Etudiant
WHERE AnneeNaisEt >= 1999
AND AnneeNaisEt <= 2002;
```

Est équivalente à:

```
SELECT *
FROM Etudiant
WHERE AnneeNaisEt
BETWEEN 1999 AND 2002;
```


L'OPÉRATEUR LIKE

- Le langage SQL fournit des options pour les recherches par motif (pattern matching) à l'aide de l'opérateur **LIKE**;
- L'opérateur **LIKE** compare de manière générique des chaînes de caractères;
- On utilise l'opérateur LIKE avec le symbole '_' pour désigner n'importe quel caractère (chaque '_' sera désigné par un seul caractère), et le sympbole '%' pour désigner n'importe quelle chaîne de caractères.

EXEMPLE 4

Soit l'extension de la table Etudiant:

NumInscrEt	NomEt	PreEt	VilleEt	AnneeNaisEt
10001	Bakkali	Ahmed	Safi	2000
10002	Benali	Rachid	Rabat	1999
10003	Chadid	Fadoua	Safi	2000
10004	Hajjaji	Hind	El Jadida	1998
10005	Himich	Ahmed	Marrakech	1998

Requête pour afficher les étudiants dont leur nom se termine par « li »:

```
SELECT *
FROM Etudiant
WHERE NomEt LIKE '%li';
```

Donne le résultat suivant:

NumInscrEt	NomEt	PreEt	VilleEt	AnneeNaisEt
10001	Bakkali	Ahmed	Safi	2000
10002	Benali	Rachid	Rabat	1999

EXEMPLE 5

Soit l'extension de la table Etudiant:

NumInscrEt	NomEt	PreEt	VilleEt	AnneeNaisEt
10001	Bakkali	Ahmed	Safi	2000
10002	Benali	Rachid	Rabat	1999
10003	Chadid	Fadoua	Safi	2000
10004	Hajjaji	Hind	El Jadida	1998
10005	Himich	Ahmed	Marrakech	1998

Requête pour afficher les étudiants dont leur prénom contient la lettre « c »:

```
SELECT *
FROM Etudiant
WHERE PreEt LIKE '%c%';
```

Donne le résultat suivant:

NumInscrEt	NomEt	PreEt	VilleEt	AnneeNaisEt
10002	Benali	Rachid	Rabat	1999

EXEMPLE 6

Soit l'extension de la table Etudiant:

NumInscrEt	NomEt	PreEt	VilleEt	AnneeNaisEt
10001	Bakkali	Ahmed	Safi	2000
10002	Benali	Rachid	Rabat	1999
10003	Chadid	Fadoua	Safi	2000
10004	Hajjaji	Hind	El Jadida	1998
10005	Himich	Ahmed	Marrakech	1998

Requête pour afficher les étudiants dont leur prénom commence par la lettre 'H' et comprend exactement 4 caractères:

```
SELECT *
FROM Etudiant
WHERE PreEt LIKE 'H___';
```

Donne le résultat suivant:

NumInscrEt	NomEt	PreEt	VilleEt	AnneeNaisEt
10004	Hajjaji	Hind	El Jadida	1998

EXEMPLE 7

Soit l'extension de la table Etudiant:

NumInscrEt	NomEt	PreEt	VilleEt	AnneeNaisEt
10001	Bakkali	Ahmed	Safi	2000
10002	Benali	Rachid	Rabat	1999
10003	Chadid	Fadoua	Safi	2000
10004	Hajjaji	Hind	El Jadida	1998
10005	Himich	Ahmed	Marrakech	1998

Requête pour afficher les étudiants habitant à Safi, Rabat ou à Marrakech

```
SELECT *
FROM Etudiant
WHERE VilleEt='Safi' OR
VilleEt='Rabat' OR VilleEt='Marrakech';
```

Donne le résultat suivant:

NumInscrEt	NomEt	PreEt	VilleEt	AnneeNaisEt
10001	Bakkali	Ahmed	Safi	2000
10002	Benali	Rachid	Rabat	1999
10003	Chadid	Fadoua	Safi	2000
10005	Himich	Ahmed	Marrakech	1998

EXEMPLE 8

Soit l'extension de la table Etudiant:

NumInscrEt	NomEt	PreEt	VilleEt	AnneeNaisEt
10001	Bakkali	Ahmed	Safi	2000
10002	Benali	Rachid	Rabat	1999
10003	Chadid	Fadoua	Safi	2000
10004	Hajjaji	Hind	El Jadida	1998
10005	Himich	Ahmed	Marrakech	1998

Requête pour afficher les étudiants habitant à Safi, Rabat ou à Marrakech

```
SELECT *
FROM Etudiant
WHERE
VilleEt IN ('Safi', 'Rabat','Marrakech');
```

Donne le résultat suivant:

NumInscrEt	NomEt	PreEt	VilleEt	AnneeNaisEt
10001	Bakkali	Ahmed	Safi	2000
10002	Benali	Rachid	Rabat	1999
10003	Chadid	Fadoua	Safi	2000
10005	Himich	Ahmed	Marrakech	1998

REMARQUE

```
SELECT *
FROM Etudiant
WHERE VilleEt='Safi' OR
VilleEt='Rabat' OR
VilleEt='Marrakech';
```

Est équivalente à:

```
SELECT *
FROM Etudiant
WHERE
VilleEt IN ('Safi','Rabat','Marrakech');
```


L'OPÉRATEUR IS NULL

- **NULL représente une valeur inconnue;**
- **NULL** ne peut pas être **égal à (ou différent de) n'importe quelle expression;**
- Les **opérateurs de comparaison (=, !=, <, et >) ne peuvent pas être appliqués** pour comparer **une expression à NULL;**
- Pour comparer des expressions à NULL, **vous devez utilisez les opérateurs**:
  - ✓ **IS NULL;**
  - ✓ **IS NOT NULL.**

EXEMPLE 1

Soit l'extension de la table Note:

NumInscrEt	CodeMat	Date_obtention	Note
NULL	4	2019-03-01	12.00
2	4	2019-03-01	03.00
3	4	2019-03-01	05.00
4	4	2019-03-01	13.00
5	4	2019-03-01	06.00

La requête suivante:

```
SELECT *
FROM Etudiant
WHERE
NumInscrEt IS NOT NULL;
```

Donne le résultat suivant:

NumInscrEt	CodeMat	Date_obtention	Note
2	4	2019-03-01	03.00
3	4	2019-03-01	05.00
4	4	2019-03-01	13.00
5	4	2019-03-01	06.00

EXEMPLE 2

Soit l'extension de la table Note:

NumInscrEt	CodeMat	Date_obtention	Note
NULL	4	2019-03-01	12.00
2	4	2019-03-01	03.00
3	4	2019-03-01	05.00
4	4	2019-03-01	13.00
5	4	2019-03-01	06.00

La requête suivante:

```
SELECT *
FROM Etudiant
WHERE
NumInscrEt IS NULL;
```

Donne le résultat suivant:

NumInscrEt	CodeMat	Date_obtention	Note
NULL	4	2019-03-01	12.00

PROJECTION

Rappel:

La projection d'une table R de structure R(C_1, C_2, ..., C_n) sur un ensemble de colonnes C_{i1}, C_{i2}, ..., C_{ip} donne une table R' ayant pour structure cet ensemble colonnes (C_{i1}, C_{i2}, ..., C_{ip}) et toutes les lignes **(SQL admet les doublons)**.

- **SQL permet d'avoir des lignes doublons dans les résultats des requêtes,** donc il ne s'agit pas **d'ensemble au sens strict du terme;**
- La spécification de **clés primaire permet d'éviter les doublons dans les tables,** mais les **doublons peuvent apparaître dans le résultat d'une requête.**

Syntaxe:

```
SELECT Ci1, Ci2, ..., Cip
FROM nom_table ;
```


EXEMPLE 1

Soit l'extension de la table Etudiant:

NumInscrEt	NomEt	PreEt	VilleEt	AnneeNaisEt
10001	Bakkali	Ahmed	Safi	2000
10002	Benali	Rachid	Rabat	1999
10003	Chadid	Fadoua	Safi	2000
10004	Hajjaji	Hind	El Jadida	1998
10005	Himich	Ahmed	Marrakech	1998

Requête pour afficher les prénoms des étudiants.

```
SELECT PreEt
FROM Etudiant ;
```

Donne le résultat suivant:

PreEt
Ahmed
Rachid
Fadoua
Hind
Ahmed

L'OPÉRATEUR DISTINCT

- Le langage SQL fournit l'opérateur « **DISTINCT** » qui **permet d'éviter d'avoir des lignes identiques (doublons) dans les résultats** d'une requête;

Exemple:

Requête pour afficher les prénoms des étudiants.

```
SELECT DISTINCT PreEt
FROM Etudiant ;
```

Donne le résultat suivant:

PreEt
Ahmed
Rachid
Fadoua
Hind

LES ALIAS(RENOMMAGE)

Rappel:

Le renommage permet de renommer une table sans changé sa structure ou de renommer des colonnes d'une table sans changé leurs types de données.

- Les noms des colonnes sont **par défaut ceux indiqués dans la clause SELECT** (noms choisis pendant la création de la table), on utilise le mot-clé **AS pour créer des alias** des colonnes(renommer ces colonnes **temporairement**);
- Les noms des tables sont **par défaut ceux indiqués dans la clause FROM** (noms choisis pendant la création des tables), on utilise le mot-clé **AS pour créer des alias des tables** (renommer ces tables **temporairement**);
- Les **alias permettent de simplifié l'écriture et la lisibilité des requêtes** (voir jointures, fonctions...).

Syntaxe:

SELECT C_1 **AS** C'_1, C_2 **AS** C'_2, ..., C_{ip} **AS** C'_{ip}
FROM nom_table **AS** nouveau_nom_table**;**

EXEMPLE 1

Soit l'extension de la table Etudiant:

NumInscrEt	NomEt	PreEt	VilleEt	AnneeNaisEt
10001	Bakkali	Ahmed	Safi	2000
10002	Benali	Rachid	Rabat	1999
10003	Chadid	Fadoua	Safi	2000
10004	Hajjaji	Hind	El Jadida	1998
10005	Himich	Ahmed	Marrakech	1998

Requête pour renommer les colonnes de la table

SELECT NumInscrEt **AS** Nu,NomEt **AS** No,PreEt **AS** Pr,VilleEt **AS** Vi,AnneeNaisEt **AS** AN
FROM Etudiant;

Donne le résultat suivant:

Nu	No	Pr	Vi	AN
10001	Bakkali	Ahmed	Safi	2000
10002	Benali	Rachid	Rabat	1999
10003	Chadid	Fadoua	Safi	2000
10004	Hajjaji	Hind	El Jadida	1998
10005	Himich	Ahmed	Marrakech	1998

EXEMPLE 2

Soit l'extension de la table Etudiant:

NumInscrEt	NomEt	PreEt	VilleEt	AnneeNaisEt
10001	Bakkali	Ahmed	Safi	2000
10002	Benali	Rachid	Rabat	1999
10003	Chadid	Fadoua	Safi	2000
10004	Hajjaji	Hind	El Jadida	1998
10005	Himich	Ahmed	Marrakech	1998

Requête pour renommer le nom de la table:

SELECT *
FROM Etudiant **AS** ET
WHERE ET. VilleEt='Safi';

Donne le résultat suivant:

NumInscrEt	NomEt	PreEt	VilleEt	AnneeNaisEt
10001	Bakkali	Ahmed	Safi	2000
10003	Chadid	Fadoua	Safi	2000

LE TRI

- Le langage SQL donne la possibilité de **trier les lignes d'une table sur une ou plusieurs colonnes, par ordre ascendant ou descendant**;
- Pour trier les lignes d'une table utiliser la commande **ORDER BY** suivie de la liste des colonnes à triées suivi de l'ordre de tri(ascendant ou descendant).

Syntaxe:

SELECT C_{i1}, C_{i2}, ..., C_{ip}
FROM nom_table
ORDER BY C_{j1} **ordre_de_tri**, C_{j2} **ordre_de_tri**, ..., C_{jn} **ordre_de_tri;**

- **Ordre_de_tri** prend la valeur:
  - ✓ **ASC**: ordre ascendant (plus petit au plus grand);
  - ✓ **DESC**: ordre descendant (plus grand au plus petit);
- **Par défaut les lignes sont triées par ordre ascendant**.

EXEMPLE 1

Soit l'extension de la table Etudiant:

NumInscrEt	NomEt	PreEt	VilleEt	AnneeNaisEt
10001	Bakkali	Ahmed	Safi	2000
10002	Benali	Rachid	Rabat	1999
10003	Chadid	Fadoua	Safi	2000
10004	Hajjaji	Hind	El Jadida	1998
10005	Himich	Ahmed	Marrakech	1998

Donne le résultat suivant:

Requête de tri par défaut des prénoms:

```
SELECT *
FROM Etudiant
ORDER BY PreEt;
```

NumInscrEt	NomEt	PreEt	VilleEt	AnneeNaisEt
10001	Bakkali	Ahmed	Safi	2000
10005	Himich	Ahmed	Marrakech	1998
10003	Chadid	Fadoua	Safi	2000
10004	Hajjaji	Hind	El Jadida	1998
10002	Benali	Rachid	Rabat	1999

EXEMPLE 2

Soit l'extension de la table Etudiant:

NumInscrEt	NomEt	PreEt	VilleEt	AnneeNaisEt
10001	Bakkali	Ahmed	Safi	2000
10002	Benali	Rachid	Rabat	1999
10003	Chadid	Fadoua	Safi	2000
10004	Hajjaji	Hind	El Jadida	1998
10005	Himich	Ahmed	Marrakech	1998

Donne le résultat suivant:

Requête de tri par ordre descendant des prénoms:

```
SELECT *
FROM Etudiant
ORDER BY PreEt DESC;
```

NumInscrEt	NomEt	PreEt	VilleEt	AnneeNaisEt
10002	Benali	Rachid	Rabat	1999
10004	Hajjaji	Hind	El Jadida	1998
10003	Chadid	Fadoua	Safi	2000
10001	Bakkali	Ahmed	Safi	2000
10005	Himich	Ahmed	Marrakech	1998

EXEMPLE 3

Soit l'extension de la table Etudiant:

NumInscrEt	NomEt	PreEt	VilleEt	AnneeNaisEt
10001	Bakkali	Ahmed	Safi	2000
10002	Benali	Rachid	Rabat	1999
10003	Chadid	Fadoua	Safi	2000
10004	Hajjaji	Hind	El Jadida	1998
10005	Himich	Ahmed	Marrakech	1998

Donne le résultat suivant:

Requête de tri par ordre descendant des prénoms puis ascendant des années_naiss:

```
SELECT *
FROM Etudiant
ORDER BY PreEt DESC, AnneeNaisEt;
```

NumInscrEt	NomEt	PreEt	VilleEt	AnneeNaisEt
10002	Benali	Rachid	Rabat	1999
10004	Hajjaji	Hind	El Jadida	1998
10003	Chadid	Fadoua	Safi	2000
10005	Himich	Ahmed	Marrakech	1998
10001	Bakkali	Ahmed	Safi	2000

EXEMPLE 4

Soit l'extension de la table Etudiant:

NumInscrEt	NomEt	PreEt	VilleEt	AnneeNaisEt
10001	Bakkali	Ahmed	Safi	2000
10002	Benali	Rachid	Rabat	1999
10003	Chadid	Fadoua	Safi	2000
10004	Hajjaji	Hind	El Jadida	1998
10005	Himich	Ahmed	Marrakech	1998

Donne le résultat suivant:

Requête de tri par ordre ascendant des années_naiss puis descendant des prénoms :

```
SELECT *
FROM Etudiant
ORDER BY AnneeNaisEt, PreEt DESC;
```

NumInscrEt	NomEt	PreEt	VilleEt	AnneeNaisEt
10004	Hajjaji	Hind	El Jadida	1998
10005	Himich	Ahmed	Marrakech	1998
10002	Benali	Rachid	Rabat	1999
10003	Chadid	Fadoua	Safi	2000
10001	Bakkali	Ahmed	Safi	2000

JOINTURE

- La **jointure est l'opération primordiale (la plus courante) dans le langage SQL car elle permet d'exprimer des requêtes qui établissent des liens entre plusieurs tables ;**
- Les **principaux types de jointures** sont:
  - ✓ **Jointure Naturelle;**
  - ✓ **θ-Jointure;**
  - ✓ **Equijointure;**
  - ✓ **Jointure externe;**
  - ✓ **Auto-Jointure.**

JOINTURE NATURELLE

Rappel:

La jointure naturelle de deux tables R et S qui ont des colonnes communes, donne une table R' ayant pour structure les colonnes appartenant à R et S (les colonnes communes n'apparaissent qu'une seule fois) formée de toutes les combinaisons des lignes de R et de S ayant les mêmes valeurs pour les colonnes communes.

- Pour exprimer la jointure naturelle utiliser le mot clé **NATURAL JOIN** en respectant la syntaxe suivante:

Syntaxe:

```
SELECT *
FROM nom_table_1 NATURAL JOIN nom_table_2;
```


EXEMPLE

Soit les extensions des tables Service, Grade et Médecin:

IdSrv	NomSrv
1	Pédiatrie
2	Gynécologie
3	Chirurgie

Grade	SalMin	SalMax
A	8 000	9 000
B	9 001	12 000
C	12 001	14 000
D	14 001	20 000

IdMed	NomMed	PreMed	Specialite	Salaire	IdSrv	aChef
10001	Bakkali	Ahmed	Chirurgie digestive	8 900	3	10007
10002	Benali	Rachid	Chirurgie cardiaque	19 000	NULL	NULL
10003	Chadid	Fadoua	Neuropédiatrie	8 600	1	10005
10004	Hajjaji	Hind	Chirurgie cardiaque	10 000	3	10007
10005	Himich	Ahmed	Orthopédie	14 800	1	10002
10006	Mazouz	Sanae	Orthopédie	11 000	1	10005
10007	Mouradi	Yahaya	Chirurgie vasculaire	15 000	3	10002

EXEMPLE

La requête suivante:

```
SELECT *
FROM Service NATURAL JOIN Medecin ;
```

Donne le résultat suivant:

IdSrv	NomSrv	IdMed	NomMed	PreMed	Specialite	Salaire	aChef
3	Chirurgie	10001	Bakkali	Ahmed	Chirurgie digestive	8900	10007
1	Pédiatrie	10003	Chadid	Fadoua	Neuropédiatrie	8600	10005
3	Chirurgie	10004	Hajjaji	Hind	Chirurgie cardiaque	10000	10007
1	Pédiatrie	10005	Himich	Ahmed	Orthopédie	14800	10002
1	Pédiatrie	10006	Mazouz	Sanae	Orthopédie	11000	10005
3	Chirurgie	10007	Mouradi	Yahaya	Chirurgie vasculaire	15000	10002

Θ-JOINTURE

Rappel:

La thêta jointure de deux tables R et S selon une qualification Q donne une table R' ayant pour structure les colonnes appartenant à R et S formée de toutes les combinaisons des lignes de R et de S qui satisfont à la qualification Q.

- Pour exprimer la thêta jointure utiliser le mot clé **JOIN** en respectant la syntaxe suivante:

Syntaxe:

```
SELECT *
FROM nom_table_1 JOIN nom_table_2
ON nom_table_1.colonne_i op nom_table_2.colonne_j ;
```

Opérateurs de comparaison:

>, >=, <, <=, =, !=

EXEMPLE

La requête suivante:

```
SELECT *
FROM Grade AS g JOIN Medecin AS m
ON m.Salaire BETWEEN g.SalMin AND g.SalMax ;
```

Donne le résultat suivant:

Grade	SalMin	SalMax	IdMed	NomMed	PreMed	Specialite	Salaire	IdSrv	aChef
A	8000	9000	10001	Bakkali	Ahmed	Chirurgie digestive	8900	3	10007
D	14001	20000	10002	Benali	Rachid	Chirurgie cardiaque	19000	NULL	NULL
A	8000	9000	10003	Chadid	Fadoua	Neuropédiatrie	8600	1	10005
B	9001	12000	10004	Hajjaji	Hind	Chirurgie cardiaque	10000	3	10007
C	12001	14000	10005	Himich	Ahmed	Orthopédie	13800	1	10002
B	9001	12000	10006	Mazouz	Sanae	Orthopédie	11000	1	10005
C	12001	14000	10007	Mouradi	Yahaya	Chirurgie vasculaire	14000	3	10002

PRODUIT CARTÉSIEN

Rappel:

Le produit cartésien de deux tables R et S donne une table R' ayant pour structure la concaténation des colonnes de R et de S, formée de toutes les combinaisons des lignes de R avec toutes les lignes de S.

- On obtient un produit cartésien lorsque :
  - ✓ **Une condition de jointure est omise;**
  - ✓ Le mot clé **CROSS JOIN** est utilisé;
- Pour éviter un produit cartésien, il faut **toujours insérer une condition de jointure** après le mot clé **ON**.

EXEMPLE

La requête suivante:

```
SELECT *
FROM Service JOIN
Medecin ;
```

⇔

```
SELECT *
FROM Service CROSS JOIN
Medecin;
```

⇔

```
SELECT *
FROM Service ,
Medecin;
```

Donne le résultat suivant:

IdSrv	NomSrv	IdMed	NomMed	PreMed	Specialite	Salaire	IdSrv	aChef
1	Pédiatrie	10001	Bakkali	Ahmed	Chirurgie digestive	8900	3	10007
2	Gynécologie	10001	Bakkali	Ahmed	Chirurgie digestive	8900	3	10007
3	Chirurgie	10001	Bakkali	Ahmed	Chirurgie digestive	8900	3	10007
1	Pédiatrie	10002	Benali	Rachid	Chirurgie cardiaque	19000	NULL	NULL
2	Gynécologie	10002	Benali	Rachid	Chirurgie cardiaque	19000	NULL	NULL
3	Chirurgie	10002	Benali	Rachid	Chirurgie cardiaque	19000	NULL	NULL
1	Pédiatrie	10003	Chadid	Fadoua	Neuropédiatrie	8600	1	10005
2	Gynécologie	10003	Chadid	Fadoua	Neuropédiatrie	8600	1	10005
					...			

EQUIJOINTURE

Rappel:

L'équijointure est un cas particulier de la θ-jointure, lorsque la qualification Q est une égalité.

- Pour exprimer l'équijointure utiliser la syntaxe suivante:

Syntaxe:

```
SELECT *
FROM nom_table_1 JOIN nom_table_2
ON nom_table_1.colonne_i = nom_table_2.colonne_j ;
```


JOINTURE EXTERNE DROITE

Rappel:

La jointure externe droite: est une θ-jointure sur deux tables R et S en ajoutant à la table résultante (de la θ-jointure) toutes les lignes de la deuxième table (S) qui ne vérifient pas les critères de jointure combinées avec des valeurs NULL.

- Pour exprimer la jointure externe droite utiliser le mot clé **RIGHT OUTER JOIN** en respectant la syntaxe suivante:

Syntaxe:

```
SELECT *
FROM nom_table_1 RIGHT OUTER JOIN nom_table_2
ON nom_table_1.colonne_i op nom_table_2.colonne_j ;
```


EXEMPLE

La requête suivante:

```
SELECT *
FROM Service AS s RIGHT OUTER JOIN Medecin AS m
ON s.IdSrv=m.IdSrv;
```

Donne le résultat suivant:

IdSrv	NomSrv	IdMed	NomMed	PreMed	Specialite	Salaire	IdSrv	aChef
3	Chirurgie	10001	Bakkali	Ahmed	Chirurgie digestive	8900	3	10007
NULL	NULL	10002	Benali	Rachid	Chirurgie cardiaque	19000	NULL	NULL
1	Pédiatrie	10003	Chadid	Fadoua	Neuropédiatrie	8600	1	10005
3	Chirurgie	10004	Hajjaji	Hind	Chirurgie cardiaque	10000	3	10007
1	Pédiatrie	10005	Himich	Ahmed	Orthopédie	13800	1	10002
1	Pédiatrie	10006	Mazouz	Sanae	Orthopédie	11000	1	10005
3	Chirurgie	10007	Mouradi	Yahaya	Chirurgie vasculaire	14000	3	10002

JOINTURE EXTERNE GAUCHE

Rappel:

La jointure externe gauche: est une θ-jointure sur deux tables R et S en ajoutant à la table résultante (de la θ-jointure) toutes les lignes de la première table(R) qui ne vérifient pas les critères de jointure combinées avec des valeurs NULL.

- Pour exprimer la jointure externe gauche utiliser le mot clé **LEFT OUTER JOIN** en respectant la syntaxe suivante:

Syntaxe:

```
SELECT *
FROM nom_table_1 LEFT OUTER JOIN nom_table_2
ON nom_table_1.colonne_i op nom_table_2.colonne_j ;
```

Exemple

La requête suivante:

```
SELECT *
FROM Service AS s LEFT OUTER JOIN Medecin AS m
ON s.IdSrv=m.IdSrv;
```

Donne le résultat suivant:

IdSrv	NomSrv	IdMed	NomMed	PreMed	Specialite	Salaire	IdSrv	aChef
1	Pédiatrie	10003	Chadid	Fadoua	Neuropédiatrie	8600	1	10005
1	Pédiatrie	10005	Himich	Ahmed	Orthopédie	13800	1	10002
1	Pédiatrie	10006	Mazouz	Sanae	Orthopédie	11000	1	10005
2	Gynécologie	NULL	NULL	NULL	NULL	NULL	NULL	NULL
3	Chirurgie	10001	Bakkali	Ahmed	Chirurgie digestive	8900	3	10007
3	Chirurgie	10004	Hajjaji	Hind	Chirurgie cardiaque	10000	3	10007
3	Chirurgie	10007	Mouradi	Yahaya	Chirurgie vasculaire	14000	3	10002

JOINTURE EXTERNE

Rappel:

La jointure externe de deux tables R et S est une **θ-jointure de ces deux tables** en ajoutant les lignes de R et de S qui ne satisfont pas à la qualification Q en leurs combinant avec des valeurs nulles.

- **MySQL ne supporte pas ce type de jointure** (sous les autres SGBD comme **Oracle** ou **SQL Server** il existe la commande **FULL OUTER JOIN** qui permet de réaliser cette jointure);
- Je vous propose la solution (il existe d'autres) suivante pour réaliser la jointure externe:

Syntaxe:

```
SELECT *
FROM nom_table_1 LEFT OUTER JOIN nom_table_2
ON nom_table_1.colonne_i op nom_table_2.colonne_j
UNION # opérateur de l'union
SELECT *
FROM nom_table_1 RIGHT OUTER JOIN nom_table_2
ON nom_table_1.colonne_i op nom_table_2.colonne_j ;
```


Exemple

La requête suivante:

```
SELECT *
FROM Service AS s LEFT OUTER JOIN Medecin AS m ON s.IdSrv=m.IdSrv
UNION
SELECT *
FROM Service AS s RIGHT OUTER JOIN Medecin AS m ON s.IdSrv=m.IdSrv;
```

Donne le résultat suivant:

IdSrv	NomSrv	IdMed	NomMed	PreMed	Specialite	Salaire	IdSrv	aChef
1	Pédiatrie	10003	Chadid	Fadoua	Neuropédiatrie	8600	1	10005
1	Pédiatrie	10005	Himich	Ahmed	Orthopédie	13800	1	10002
1	Pédiatrie	10006	Mazouz	Sanae	Orthopédie	11000	1	10005
2	Gynécologie	NULL	NULL	NULL	NULL	NULL	NULL	NULL
3	Chirurgie	10001	Bakkali	Ahmed	Chirurgie digestive	8900	3	10007
3	Chirurgie	10004	Hajjaji	Hind	Chirurgie cardiaque	10000	3	10007
3	Chirurgie	10007	Mouradi	Yahaya	Chirurgie vasculaire	14000	3	10002
NULL	NULL	10002	Benali	Rachid	Chirurgie cardiaque	19000	NULL	NULL

AUTO-JOINTURE

Rappel:

L'auto-jointure est un cas particulier de la θ-jointure, lorsque qu'il s'agit d'une jointure d'une table avec elle même.

La requête suivante:

```
SELECT EstChefDe.IdMed, EstChefDe.NomMed, EstChefDe.PreMed,
Medecin.IdMed AS IdSubordonne, Medecin.NomMed AS NomSubordonne,
medecin.PreMed AS PrenomSubordonne
FROM Medecin JOIN Medecin AS EstChefDe
ON Medecin.aChef=EstChefDe.IdMed;
```