

Chapitre 2 :

Langage d'interrogation des données en SQL





Introduction

- › La commande SELECT est utilisée pour extraire des données à partir d'une base de données.
- › La syntaxe de la commande SELECT :

SELECT [*DISTINCT*] liste d'attributs

FROM liste de tables

[*WHERE* qualification]

[*GROUP BY* attributs de partitionnement]

[*HAVING* qualification de groupe]

[*ORDER BY* liste de colonnes [*ASC* | *DESC*]]



Introduction

- › Au minimum, la commande SELECT doit avoir l'entête de sélection.
- › Toutes les autres lignes sont optionnelles.
- › Chaque clause représente une étape dans le processus de sélection des données.
- › Exemple : Donnez les dates de naissances et adresses des employés.

```
SELECT Date_Nais, Adresse  
FROM Employe
```

	Date_Nais	Adresse
1	01-SEP-65	731 Fondren Huston, TX
2	08-DEC-55	638 Voss, Huston , TX
3	31-JUL-72	5631 Rice, Humble, TX
4	15-SEP-62	975 FireOak, Humble, TX
5	10-OCT-37	450 Stone, Huston, TX
6	20-JUN-41	291 Berry, Bellaire, TX
7	29-MAR-69	980, Dallas, Huston, TX
8	19-JUL-68	3321 Castle, Spring, TX



Structure des tables pour les exemples

Produit

IdPro	Nom	Marque	Prix
1	PS1	IBM	1200
2	PS2	APPLE	1300
3	PS3	IBM	1400
4	PS4	MICORSOFT	1500

Client

IdCli	Nom	Ville
1	John	Marseille
2	Stéphane	Paris
3	Christophe	Marseille
4	Jean	Lyon

Vente

IdPro ^F	IdCli ^F	Qte
1	1	500
2	1	1000
2	2	2000
4	2	500



PROJECTION EN SQL

- › La projection permet de retenir que certaines colonnes d'une table → Elle correspond à un découpage vertical
- › Quels sont les références et les prix des produits ?

PRODUIT (IdPro, Nom, Marque, Prix)



IdPro	Nom	Marque	Prix
P	PS1	IBM	1000
Q	Mac	Apple	2000
R	PS2	IBM	3000
S	Word	Microsoft	4000



IdPro	Prix
P	1000
Q	2000
R	3000
S	4000



PROJECTION EN SQL

- › La projection permet de retenir que certaines colonnes d'une table → Elle correspond à un découpage vertical
- › Quels sont les références et les prix des produits ?
 - **SELECT** IdPro, prix
FROM produit;
 - **SELECT** produit.IdPro, produit.prix
FROM produit;



PROJECTION EN SQL – LES ALIAS

- › On peut introduire un *alias* dans la clause FROM en le plaçant immédiatement après le nom de la table
- › Quels sont les références et les prix des produits ?

```
SELECT P.IdPro, P.prix
```

```
FROM produit P;
```

- › Les alias sont utilisés pour préfixer les noms de colonnes
- › Les préfixes sont obligatoires dans des cas particuliers (ex. pour une auto-jointure). Leur emploi est cependant conseillé.



PROJECTION EN SQL - DISTINCT

› Quelles sont les marques des produits ?

PRODUIT (IdPro, Nom, Marque, Prix)



IdPro	Nom	Marque	Prix
P	PS1	IBM	1000
Q	Mac	Apple	2000
R	PS2	IBM	3000
S	Word	Microsoft	4000



Marque
IBM
Apple
Microsoft



PROJECTION EN SQL - DISTINCT

- › Pour éliminer les doublons il faut spécifier **DISTINCT**
- › `Quelles sont les marques des produits ?

```
SELECT DISTINCT P.marque
```

```
FROM produit P;
```



PROJECTION EN SQL – Projection sur toutes les colonnes

- › Une étoile (*) permet de lister tous les attributs
- › Donner tous les renseignements sur les clients

```
SELECT * FROM client;
```



PROJECTION EN SQL - RENOMMAGE (AS)

› But:

- Résoudre des problèmes de compatibilité entre noms des colonnes des tables.
- Afficher un nom plus significatif pour les colonnes de la requête.

Pays

Nom	Capitale	Surface
Autriche	Vienne	83 000
UK	Londres	244 000
Suisse	Berne	41 000

- ## › Lister les informations des pays. La réponse doit avoir comme titre pour la colonne de Surface « Surface (Km2) ».



PROJECTION EN SQL - RENOMMAGE (AS)

- › La clause « AS » permet de choisir un autre nom autre que le nom de la colonne pour l'afficher avec la requête.
- › On peut mettre un espace à la place du **AS**.
 - **SELECT** Nom, Capitale, Surface **AS** "Surface(Km2)"
FROM Pays;
 - **SELECT** Nom, Capitale, Surface "Surface(Km2)"
FROM Pays;



PROJECTION – EXPRESSIONS ARITHMÉTIQUES

Opération	Description
+	Addition
-	Soustraction
*	Multiplication
/	Division

- › Afficher les noms des employés ainsi que leurs salaires mensuels et annuels.

```
SELECT nom, salaire AS "Salaire Mensuel",  
salaire * 12 AS "Salaire Annuel"  
FROM employé;
```



PROJECTION – CONCATENATION

- › La concaténation permet de lier des colonnes ou des chaînes de caractères à d'autres colonnes
- › L'opérateur de concaténation est double barre verticale (||)

```
SELECT nom || ' est un ' || job_id AS "Details de l'employe"  
FROM employé;
```



RESTRICTION EN SQL

- › La restriction permet d'extraire les lignes qui satisfont une condition → Elle correspond à un découpage horizontal
- › Quelles sont les produits de marque 'IBM' ?

PRODUIT (IdPro, Nom, Marque, Prix)

IdPro	Nom	Marque	Prix
P	PS1	IBM	1000
Q	Mac	Apple	2000
R	PS2	IBM	3000
S	Word	Microsoft	4000



IdPro	Nom	Marque	Prix
P	PS1	IBM	1000
R	PS2	IBM	3000



RESTRICTION EN SQL

› Quelles sont les produits de marque 'IBM' ?

SELECT *

FROM produit P

WHERE P.marque = 'IBM';



RESTRICTION EN SQL – Condition de restriction

- › La condition de restriction est spécifiée après la clause **WHERE**
- › La condition de restriction peut être **simple** en utilisant :
 - un prédicat d'égalité ou d'inégalité (=, <>, <, >, <=, >=)
 - un des prédicats **LIKE, BETWEEN, IN, IS NULL, EXISTS, ALL** ou **ANY**
- › La condition de restriction peut être **composée** à l'aide des connecteurs **AND, OR** et **NOT**



RESTRICTION EN SQL – Exemples (LIKE)

- › Le prédicat **LIKE** compare une chaîne avec un modèle
 - (`_`) remplace n'importe quel caractère
 - (`%`) remplace n'importe quelle suite de caractères

SQL SERVER

- (`[charlist]`) spécifie une liste de caractères
- (`[!charlist]`) ne contient pas une liste de caractères

ORACLE

- **REGEXP_LIKE** : https://www.techonthenet.com/oracle/regexp_like.php



RESTRICTION EN SQL – Exemples (LIKE)

- › Lister les clients dont le nom comporte la lettre A en 2^{ième} position

```
SELECT * FROM client C WHERE C.nom LIKE '_A%';
```

- › Lister les clients dont le nom commence par A, B ou C et ne se termine par S ou E

```
SELECT * FROM client C WHERE C.nom LIKE '[A-C]%[!S,E]';
```



RESTRICTION EN SQL – Exemples (BETWEEN)

- › Le prédicat **BETWEEN** teste l'appartenance à un intervalle
- › Exemple : Lister les détails des produits dont le prix est compris entre 5000 et 12000

SELECT *

FROM produit P

WHERE P.prix **BETWEEN** 5000 **AND** 12000;



RESTRICTION EN SQL – Exemples (IN)

- › Le prédicat **IN** teste l'appartenance à une liste de valeurs
- › Exemple : Lister les produits de marque IBM, Apple ou Dec

SELECT *

FROM produit P

WHERE P.marque **IN** ('IBM', 'Apple', 'Dec');



RESTRICTION EN SQL – Exemples (IS NULL)

- › Le prédicat IS NULL teste si une colonne contient une valeur NULL
- › Exemple : Lister les produits dont le prix est inconnu

SELECT *

FROM produit P

WHERE P.prix **IS NULL;**



RESTRICTION EN SQL – Exemples (AND)

- › Le connecteur **AND** relie les 2 prédicats de comparaison
- › Exemple : Lister les produits de marque IBM dont le prix est inférieur à 12000

SELECT *

FROM produit P

WHERE P.marque = 'IBM' **AND** P.prix < 12000;



RESTRICTION EN SQL – Exemples (OR)

- › Le connecteur **OR** relie les 2 prédicats de comparaison
- › Exemple : Lister les produits de marque IBM ou ceux de marque MICROSOFT

SELECT *

FROM produit P

WHERE P.marque = 'IBM' **OR** P.marque = 'MICROSOFT';



RESTRICTION EN SQL – Nombre de lignes

- › On peut limiter le nombre de lignes retournées par la clause WHERE de restriction d'un SQL
- › MySQL / PostgreSQL : Avec le mot clé **LIMIT**

SELECT P.nom

FROM produit P

WHERE P.marque = 'IBM'

LIMIT 10;



RESTRICTION EN SQL – Nombre de lignes

- › On peut limiter le nombre de lignes retournées par la clause WHERE de restriction d'un SQL
- › Oracle : Avec la clause **FETCH FIRST x ROWS ONLY**

SELECT P.nom

FROM produit P

WHERE P.marque = 'IBM'

FETCH FIRST 10 ROWS ONLY;



RESTRICTION EN SQL - Les sous requêtes

- › SQL permet l'imbrication de sous-requêtes au niveau de la clause **WHERE**
- › Les sous-requêtes sont utilisées :
 - Dans des prédicats de comparaison (=, <>, <, <=, >, >=)
 - Dans des prédicats *IN*, *EXISTS*, *ALL* ou *ANY*



RESTRICTION EN SQL - Les sous requêtes :

Le prédicat IN

- › Donner les noms des clients qui ont acheté le produit 'p1'

```
SELECT C.nom FROM client C
WHERE C.IdCli IN
(
    SELECT V.IdCli
    FROM vente V
    WHERE V.IdPro = 'p1' );
```



RESTRICTION EN SQL - Les sous requêtes :

Le prédicat EXISTS

- › L'opérateur EXISTS renvoie TRUE si la sous-requête renvoie un ou plusieurs enregistrements.
- › Donner les noms des produits qui n'ont pas été acheté

SELECT P.nom

FROM produit P

WHERE NOT EXISTS

(**SELECT** * **FROM** vente V

WHERE V.IdPro = P.IdPro);



RESTRICTION EN SQL - Les sous requêtes :

Le prédicat ALL

- › Permet de tester si un prédicat de comparaison est vrai pour tous (ALL) les résultats d'une sous-requête
- › Donner les IdCli des clients ayant acheté un produit avec une quantité supérieure à chacune des quantités de produits achetées par le client 'c1'

```
SELECT V.IdCli
FROM vente V
WHERE V.qte > ALL
(
    SELECT W.qte
    FROM vente W
    WHERE W.IdCli = 'c1'
);
```



RESTRICTION EN SQL - Les sous requêtes :

Le prédicat ANY

- › Permet de tester si un prédicat de comparaison est vrai pour au moins un (ANY) des résultats d'une sous-requête
- › Donner les identifiants des clients ayant acheté un produit en quantité supérieure à au moins l'une des quantités de produits achetées par le client 'c1'

```
SELECT V.IdCli
FROM vente V
WHERE V.qte >= ANY
(
    SELECT W.qte
    FROM vente W
    WHERE W.IdCli = 'c1'
);
```



Exercice

Journal(Code-j, Titre, Prix, Type, Periodicite)

Depot(No-dep, Nom-Dep, Adresse)

Livraison(No-dep#, Code-j#, Date-liv, Qte-liv)

- › Quels sont les titres et les prix des journaux?

```
SELECT Titre, Prix  
FROM Journal;
```

- › Donnez tous les renseignements connus sur les Dépôts

```
SELECT *  
FROM Depot;
```

- › Donner les différents types de journaux

```
SELECT DISTINCT type FROM Journal;
```



Exercice

Journal(Code-j, Titre, Prix, Type, Periodicite)

Depot(No-dep, Nom-Dep, Adresse)

Livraison(No-dep#, Code-j#, Date-liv, Qte-liv)

- › Donnez tous les renseignements connus sur les journaux hebdomadaires

```
SELECT Code-j, Titre, Prix, Type
FROM Journal
WHERE Periodicite = 'hebdomadaire';
```

- › On veut afficher les informations des journaux dans une seule colonne "Informations journaux" sous la forme :

Le journal <type> <titre> apparait de façon <périodicité>

```
SELECT 'Le journal ' || type || ' ' || titre || 'apparait de façon ' || periodicite
AS "Informations journaux"
FROM journal;
```



Exercice

Journal(Code-j, Titre, Prix, Type, Periodicite)

Depot(No-dep, Nom-Dep, Adresse)

Livraison(No-dep#, Code-j#, Date-liv, Qte-liv)

- › Donnez les numéros des dépôts qui livre des journaux de périodicité "hebdomadaire"

```
SELECT No-dep
FROM Livraison
WHERE Code-j IN
      (SELECT code-j FROM Journal
       WHERE Periodicite = 'hebdomadaire') ;
```



JOINTURE DE TABLES EN SQL

- › La jointure des tables en SQL permet de rechercher des informations dans différentes tables.
- › La jointure produit une table en mémoire dont les lignes sont obtenues en concaténant les lignes des tables utilisées dans la jointure et en ne gardant que ceux qui vérifient une condition de liaison.



JOINTURE EN SQL : PRODUIT CARTÉSIEN AVEC CONDITION

R1(A, B, C)

A	B	C
A1	B1	10
A2	B2	10
A3	B3	20
A4	B4	30



R2(U, V)

U	V
10	V1
20	V2
30	V3

R1 × R2 (R1.C = R2.U)



A	B	C	U	V
A1	B1	10	10	V1
A2	B2	10	10	V1
A3	B3	20	20	V2
A4	B4	30	30	V3



JOINTURE CLASSIQUE EN SQL : EXEMPLE

Soit les deux tables :

- Pays (Nom, Capitale, Population, Surface, **Continent**)

- LangueParlee (**Langue**, NomPays#, %Population)

› Pour chaque langue, dans quels continents est-elle parlée?



JOINTURE CLASSIQUE EN SQL : EXEMPLE

Pays

Nom	Capitale	Population	Surface	Continent
Tunisie	Tunis	11 000 000	163 610	Afrique
France	Paris	67 795 000	672 051	Europe
Algérie	Alger	44 200 000	2 381 741	Afrique

LangueParlee

Langue	NomPays#	%Population
Arabe	Tunisie	99
Français	Tunisie	63
Arabe	Algérie	90
Français	Algérie	60
Français	France	99



Nom	Capitale	Population	Surface	Continent	Langue	NomPays	%Population
Tunisie	Tunis	11 000 000	163 610	Afrique	Arabe	Tunisie	99
Tunisie	Tunis	11 000 000	163 610	Afrique	Français	Tunisie	63
France	Paris	67 795 000	672 051	Europe	Français	France	99
Algérie	Alger	44 200 000	2 381 741	Afrique	Arabe	Algérie	90
Algérie	Alger	44 200 000	2 381 741	Afrique	Français	Algérie	60



JOINTURE CLASSIQUE EN SQL : EXEMPLE

Soit les deux tables :

- Pays (Nom, Capitale, Population, Surface, **Continent**)

- LangueParlee (**Langue**, NomPays#, %Population)

› Pour chaque langue, dans quels continents est-elle parlée?

SELECT DISTINCT Langue, Contient

FROM Pays, LangueParlee

WHERE Pays.Nom = LangueParlee.NomPays;



JOINTURE CLASSIQUE EN SQL

- › La jointure s'exprime simplement en incluant plusieurs tables après la clause **FROM**
- › La condition de jointure est exprimée après **WHERE**
- › Exemple : Donner les références et les noms des produits vendus



JOINTURE CLASSIQUE EN SQL

Produit

<u>IdPro</u>	Nom	Prix
1	PS1	1200
2	PS2	1300
3	PS3	1400
4	PS4	1500

Vente

<u>IdPro#</u>	<u>IdCli#</u>	Qte
1	1	500
2	1	1000
2	2	2000
4	2	500

Client

<u>IdCli</u>	Nom	Ville
1	John	Marseille
2	Stéphane	Paris
3	Christophe	Marseille
4	Jean	Lyon



<u>IdPro</u>	Nom	Prix	<u>IdPro</u>	<u>IdCli</u>	Qte
1	PS1	1200	1	1	500
2	PS2	1300	2	1	1000
2	PS2	1300	2	2	200
4	PS4	1500	4	2	500



Nom	IdPro
PS1	1
PS2	2
PS4	4



JOINTURE CLASSIQUE EN SQL

- › La jointure s'exprime simplement en incluant plusieurs tables après la clause **FROM**
- › La condition de jointure est exprimée après **WHERE**
- › Exemple : Donner les références et les noms des produits vendus

```
SELECT DISTINCT V.IdPro, P.nom
```

```
FROM produit P , vente V
```

```
WHERE P.IdPro = V.IdPro;
```



JOINTURE CLASSIQUE EN SQL

- › Donner les noms des clients qui ont acheté le produit de nom 'PS1'



JOINTURE CLASSIQUE EN SQL

Produit

IdPro	Nom	Prix
1	PS1	1200
2	PS2	1300
3	PS3	1400
4	PS4	1500

Vente

IdPro#	IdCli#	Qte
1	1	500
2	1	1000
2	2	2000
4	2	500

Client

IdCli	Nom	Ville
1	John	Marseille
2	Stéphane	Paris
3	Christophe	Marseille
4	Jean	Lyon



IdPro	Nom	Prix	IdPro	IdCli	Qte	IdCli	Nom	Ville
1	PS1	1200	1	1	500	1	John	Marseille
2	PS2	1300	2	1	1000	1	John	Marseille
2	PS2	1300	2	2	200	2	Stéphane	Paris
4	PS4	1500	4	2	500	2	Stéphane	Paris

Nom

John



JOINTURE CLASSIQUE EN SQL

- › Donner les noms des clients qui ont acheté le produit de nom 'PS1'

SELECT C.nom

FROM client C , produit P, vente V

WHERE C.IdCli = V.IdCli

AND V.IdPro = P.IdPro

AND P.nom = 'PS1';



Exercice

Journal(Code-j, Titre, Prix, Type, Periodicite)

Depot(No-dep, Nom-Dep, Adresse)

Livraison(No-dep#, Code-j#, Date-liv, Qte-liv)

- › Donnez les numéros des dépôts qui livre des journaux de périodicité "hebdomadaire".

```
SELECT L.No-dep
```

```
FROM Livraison L, Journal J
```

```
WHERE L.Code-j = J.Code-j
```

```
AND J.Periodicite = 'hebdomadaire' ;
```



Exercice

- › Donnez les codes des journaux livrés aux dépôts de « Tunis ».

Depot(No-dep, Nom-Dep, Adresse)

Livraison(No-dep#, Code-j#, Date-liv, Qte-liv)

?

SELECT L.Code-j

FROM Depot D, Livraison L

WHERE L.No-dep = D.No-dep **AND** D.Adresse = 'Tunis';



JOINTURE EN SQL – Nouveau Syntaxe

```
SELECT table1.nom_colonne, table2.nom_colonne  
FROM table1  
[[INNER] JOIN table2 USING (nom_colonne)] |  
[[INNER] JOIN table2  
ON (table1.nom_colonne = table2.nom_colonne)] |  
[{LEFT | RIGHT | FULL} [OUTER] JOIN table2  
ON (table1.nom_colonne = table2.nom_colonne)] |  
[CROSS JOIN table2];
```



JOINTURE EN SQL – INNER JOIN

Soit les deux tables :

- Pays (Nom, Capitale, Population, Surface, **Continent**)

- LangueParlee (**Langue**, NomPays#, %Population)

› Pour chaque langue, dans quels continents est-elle parlée?

```
SELECT DISTINCT Langue, Contient
```

```
FROM Pays, LangueParlee
```

```
WHERE Pays.Nom = LangueParlee.NomPays;
```



JOINTURE EN SQL – INNER JOIN

Soit les deux tables :

- Pays (Nom, Capitale, Population, Surface, **Continent**)

- LangueParlee (**Langue**, NomPays#, %Population)

› Pour chaque langue, dans quels continents est-elle parlée?

```
SELECT DISTINCT Langue, Contient
```

```
FROM Pays INNER JOIN LangueParlee
```

```
ON (Pays.Nom = LangueParlee.NomPays);
```



JOINTURE EN SQL – INNER JOIN

Soit les deux tables :

- Pays (Nom, Capitale, Population, Surface, **Continent**)

- LangueParlee (**Langue**, NomPays#, %Population)

› Pour chaque langue, dans quels continents est-elle parlée?

```
SELECT DISTINCT Langue, Contient
```

```
FROM Pays JOIN LangueParlee
```

```
ON (Pays.Nom = LangueParlee.NomPays);
```



JOINTURE EN SQL – INNER JOIN

- › Exemple : Donner les références et les noms des produits vendus

```
SELECT V.IdPro, P.nom
```

```
FROM produit P , vente V
```

```
WHERE P.IdPro = V.IdPro;
```



JOINTURE EN SQL – INNER JOIN

- › Exemple : Donner les références et les noms des produits vendus

```
SELECT IdPro, P.nom
```

```
FROM produit P JOIN vente V
```

```
USING (idPro);
```

- › **ATTENTION :** Avec USING, la colonne de jointure n'admet pas d'alias dans toute la commande SELECT



JOINTURE EN SQL - INNER JOIN

- › Donner les noms des clients qui ont acheté le produit de nom 'PS1'

SELECT C.nom

FROM client C , produit P, vente V

WHERE C.IdCli = V.IdCli

AND V.IdPro = P.IdPro

AND P.nom = 'PS1';



JOINTURE EN SQL - INNER JOIN

- › Donner les noms des clients qui ont acheté le produit de nom 'PS1'

SELECT C.nom

FROM client C **JOIN** vente V **USING** IdCli

JOIN produit P **USING** idPro

WHERE P.nom = 'PS1';



Exercice

Journal(Code-j, Titre, Prix, Type, Periodicite)

Depot(No-dep, Nom-Dep, Adresse)

Livraison(No-dep#, Code-j#, Date-liv, Qte-liv)

› Donnez les codes des journaux livrés aux dépôts de « Tunis ».



Exercice

Journal(Code-j, Titre, Prix, Type, Periodicite)

Depot(No-dep, Nom-Dep, Adresse)

Livraison(No-dep#, Code-j#, Date-liv, Qte-liv)

› Donnez les codes des journaux livrés aux dépôts de « Tunis ».

SELECT L.Code-j

FROM Depot D **JOIN** Livraison L

ON (L.No-dep = D.No-dep)

WHERE D.Adresse = 'Tunis';



Exercice

Journal(Code-j, Titre, Prix, Type, Periodicite)

Depot(No-dep, Nom-Dep, Adresse)

Livraison(No-dep#, Code-j#, Date-liv, Qte-liv)

› Donnez les codes des journaux livrés aux dépôts de « Tunis ».

SELECT L.Code-j

FROM Depot D **JOIN** Livraison L

USING (No-dep)

WHERE D.Adresse = 'Tunis';



AUTOJOINTURE EN SQL

› Donner les noms des clients de la même ville que John

Client C1

IdCli	Nom	Ville
1	John	Marseille
2	Stéphane	Paris
3	Christophe	Marseille
4	Jean	Lyon

Client C2

IdCli	Nom	Ville
1	John	Marseille
2	Stéphane	Paris
3	Christophe	Marseille
4	Jean	Lyon



IdCli	Nom	Ville	IdCli	Nom	Ville
1	John	Marseille	1	John	Marseille
1	John	Marseille	3	Christophe	Marseille
2	Stéphane	Paris	2	Stéphane	Paris
3	Christophe	Marseille	1	John	Marseille
3	Christophe	Marseille	3	Christophe	Marseille
4	Jean	Lyon	4	Jean	Lyon



Nom
Christophe



AUTOJOINTURE EN SQL

- › Donner les noms des clients de la même ville que John

```
SELECT C2.nom  
FROM client C1 , client C2  
WHERE C2.ville = C1.ville  
AND C1.nom = 'John'  
AND C2.nom <> 'John';
```

- › Pour pouvoir distinguer les références ville dans les 2 copies,
il faut introduire 2 alias différents C1 et C2 de la table client



AUTOJOINTURE EN SQL - INNER JOIN

- › Donner les noms des clients de la même ville que John

```
SELECT C2.nom  
FROM client C1 JOIN client C2  
ON (C2.ville = C1.ville)  
WHERE C1.nom = 'John' AND C2.nom <> 'John';
```

```
SELECT C2.nom  
FROM client C1 JOIN client C2  
USING (ville)  
WHERE C1.nom = 'John' AND C2.nom <> 'John';
```



Exercice

Journal(Code-j, Titre, Prix, Type, Periodicite)

Depot(No-dep, Nom-Dep, Adresse)

Livraison(No-dep#, Code-j#, Date-liv, Qte-liv)

- › Donnez les codes des journaux de même périodicité que le journal de code "j1".



Exercice

Journal(Code-j, Titre, Prix, Type, Periodicite)

Depot(No-dep, Nom-Dep, Adresse)

Livraison(No-dep#, Code-j#, Date-liv, Qte-liv)

- › Donnez les codes des journaux de même périodicité que le journal de code "j1".

```
SELECT j2.Code-j
```

```
FROM Journal j1, Journal j2
```

```
WHERE j1.Peridocite = j2.Periodicite
```

```
AND j1.Code-j = 'j1'
```

```
AND j2.Code-j <> 'j1';
```



Exercice

Journal(Code-j, Titre, Prix, Type, Periodicite)

Depot(No-dep, Nom-Dep, Adresse)

Livraison(No-dep#, Code-j#, Date-liv, Qte-liv)

- › Donnez les codes des journaux de même périodicité que le journal de code "j1".

```
SELECT j2.Code-j
FROM Journal j1 JOIN Journal j2
ON (j1.Peridocite = j2.Periodicite)
WHERE j1.Code-j = 'j1'
AND j2.Code-j <> 'j1';
```



Exercice

Journal(Code-j, Titre, Prix, Type, Periodicite)

Depot(No-dep, Nom-Dep, Adresse)

Livraison(No-dep#, Code-j#, Date-liv, Qte-liv)

- › Donnez les codes des journaux de même périodicité que le journal de code "j1".

```
SELECT j2.Code-j
FROM Journal j1 JOIN Journal j2
USING (Peridocite)
WHERE j1.Code-j = 'j1'
AND j2.Code-j <> 'j1';
```



JOINTURE INTERNE ET JOINTURER EXTERNE

Département

DEPARTEMENT_NON	DEPARTEMENT_ID
Administration	10
Marketing	20
Shipping	50
Sales	80
Executive	90
Contracting	190

Le département 190 ne compte pas d'employés

Employé

DEPARTEMENT_ID	NOM
90	King
90	Kochhar
10	Whalen
80	Taylor
NULL	Grant
50	Mourgos
20	Fay

L'employé nommé Grant n'est pas attaché à un département

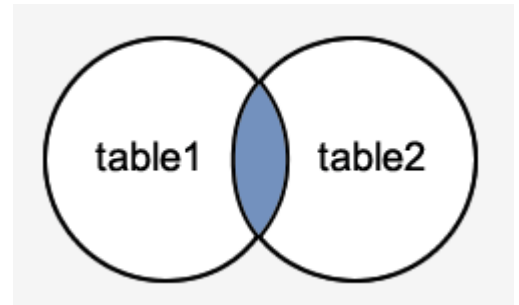


JOINTURE INTERNE ET JOINTURER EXTERNE

- › Une jointure interne entre deux tables (INNER JOIN) ne renvoie que les lignes concordantes entre les deux tables
- › Une jointure externe gauche (LEFT OUTER JOIN) (ou droite RIGHT OUTER JOIN) entre deux tables renvoie les résultats d'une jointure interne (INNER JOIN), ainsi que les lignes sans correspondance de la table de gauche (ou de droite).
- › une jointure externe complète (FULL OUTER JOIN) entre deux tables renvoie les résultats d'une jointure interne (INNER), ainsi que les résultats d'une jointure à gauche et à droite.



JOINTURE EN SQL : INNER JOIN



SELECT *

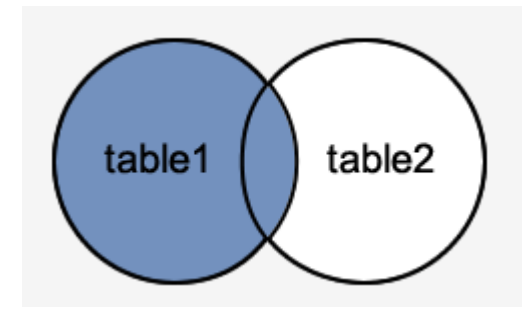
FROM DEPARTEMENT D **INNER JOIN** EMPLOYE E

USING (DEPARTEMENT_ID);

DEPARTEMENT_NON	DEPARTEMENT_ID	DEPATEMENT_ID	NOM
Administration	10	10	Whalen
Marketing	20	20	Fay
Shipping	50	50	Mourgos
Sales	80	80	Taylor
Executive	90	90	King
Executive	90	90	Kochhar



JOINTURE EN SQL : LEFT OUTER JOIN



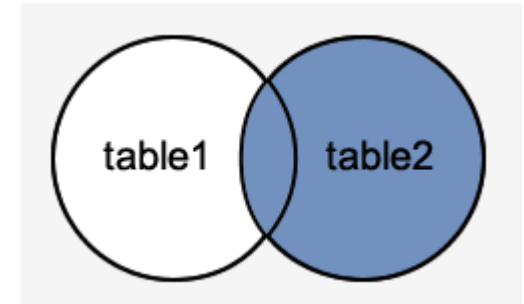
SELECT *

FROM DEPARTEMENT D **LEFT OUTER JOIN** EMPLOYE E
USING (DEPARTEMENT_ID);

DEPARTEMENT_NON	DEPARTEMENT_ID	DEPATEMENT_ID	NOM
Administration	10	10	Whalen
Marketing	20	20	Fay
Shipping	50	50	Mourgos
Sales	80	80	Taylor
Executive	90	90	King
Executive	90	90	Kochhar
Contracting	190	NULL	NULL



JOINTURE EN SQL : RIGHT OUTER JOIN



SELECT *

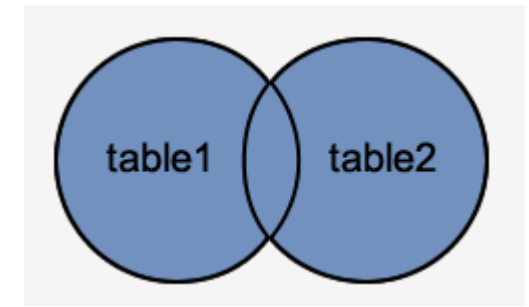
FROM DEPARTEMENT D **RIGHT OUTER JOIN** EMPLOYE
E

USING (DEPARTEMENT_ID);

DEPARTEMENT_NON	DEPARTEMENT_ID	DEPATEMENT_ID	NOM
Administration	10	10	Whalen
Marketing	20	20	Fay
Shipping	50	50	Mourgos
Sales	80	80	Taylor
Executive	90	90	King
Executive	90	90	Kochhar
NULL	NULL	NULL	Grant



JOINTURE EN SQL : FULL OUTER JOIN



SELECT *

FROM DEPARTEMENT D **FULL OUTER JOIN** EMPLOYE E
USING (DEPARTEMENT_ID);

DEPARTEMENT_NON	DEPARTEMENT_ID	DEPATEMENT_ID	NOM
Administration	10	10	Whalen
Marketing	20	20	Fay
Shipping	50	50	Mourgos
Sales	80	80	Taylor
Executive	90	90	King
Executive	90	90	Kochhar
Contracting	190	NULL	NULL
NULL	NULL	NULL	Grant



JOINTURE EN SQL : CROSS JOIN

- › La clause CROSS JOIN génère le produit "croisé" de deux tables.
- › On parle également de produit cartésien entre les deux tables.

SELECT *

FROM DEPARTEMENT D **CROSS JOIN** EMPLOYE E;



Exercice

Journal(Code-j, Titre, Prix, Type, Periodicite)

Depot(No-dep, Nom-Dep, Adresse)

Livraison(No-dep#, Code-j#, Date-liv, Qte-liv)

› Donnez les titres des journaux qui n'ont jamais été livrés.



Exercice

Journal(Code-j, Titre, Prix, Type, Periodicite)

Depot(No-dep, Nom-Dep, Adresse)

Livraison(No-dep#, Code-j#, Date-liv, Qte-liv)

› Donnez les titres des journaux qui n'ont jamais été livrés.

```
SELECT J.titre
```

```
FROM Journal J LEFT OUTER JOIN Livraison L
```

```
ON (J.Code-j = L.Code-j)
```

```
WHERE L.Code-j IS NULL;
```



Exercice

Journal(Code-j, Titre, Prix, Type, Periodicite)

Depot(No-dep, Nom-Dep, Adresse)

Livraison(No-dep#, Code-j#, Date-liv, Qte-liv)

› Donnez les titres des journaux qui n'ont jamais été livrés.

```
SELECT J.titre
```

```
FROM Journal J LEFT JOIN Livraison L
```

```
ON (J.Code-j = L.Code-j)
```

```
WHERE L.Code-j IS NULL;
```



Exercice

Journal(Code-j, Titre, Prix, Type, Periodicite)

Depot(No-dep, Nom-Dep, Adresse)

Livraison(No-dep#, Code-j#, Date-liv, Qte-liv)

- › Donnez les Numéros des depot qui n'ont jamais reçu de livraisons.

SELECT D.No-dep

FROM Livraison L **RIGHT OUTER JOIN** Depot D

ON(L.No-dep = D.No-dep)

WHERE L.No-dep **IS NULL**;



Exercice

Journal(Code-j, Titre, Prix, Type, Periodicite)

Depot(No-dep, Nom-Dep, Adresse)

Livraison(No-dep#, Code-j#, Date-liv, Qte-liv)

- › Donnez les Numéros des depot qui n'ont jamais reçu de livraisons.

SELECT D.No-dep

FROM Livraison L **RIGHT JOIN** Depot D

ON(L.No-dep = D.No-dep)

WHERE L.No-dep **IS NULL**;



TRI du résultat d'un SELECT (ORDER BY)

- › La clause **ORDER BY** permet de spécifier les colonnes définissant les critères de tri
- › Le tri se fera d'abord selon la première colonne spécifiée, puis selon la deuxième colonne etc...
- › L'ordre de tri est précisé par ASC (croissant) ou DESC (décroissant) ; par défaut ASC



TRI du résultat d'un SELECT (ORDER BY)

- › Lister les produits en les triant par marques et à l'intérieur d'une marque par prix décroissants

SELECT *

FROM produit P

ORDER BY P.marque, P.prix **DESC;**



TRI du résultat d'un SELECT (ORDER BY)

IdPro	Nom	Marque	Prix
1	Produit1	ibm	1500
2	Produit2	microsoft	1200
3	Produit3	ibm	2000
4	Produit4	microsoft	1000

IdPro	Nom	Marque	Prix
3	Produit3	ibm	2000
1	Produit1	ibm	1500
2	Produit2	microsoft	1200
4	Produit4	microsoft	1000



ORDER BY – TRI par Alia de colonne

```
SELECT nom, salaire AS "Salaire Mensuel",  
salaire * 12 AS "Salaire Annuel"  
FROM employé;  
ORDER BY "Salaire Annuel"
```



ORDER BY – TRI par position numérique de colonne

```
SELECT Id_Pro, Nom, Marque, Prix  
  
FROM produit  
  
ORDER BY 3;
```



Les fonctions de calcul

- › SQL fournit des fonctions de calcul opérant sur l'ensemble des valeurs d'une colonne de table
 - *COUNT* : nombre de valeurs
 - *SUM* : somme des valeurs
 - *AVG* : moyenne des valeurs
 - *MAX* : plus grande valeur
 - *MIN* : plus petite valeur
- › La fonction spéciale COUNT (*) compte toutes les lignes dans une table



Les fonctions de calcul

› SUM, AVG, MAX, MIN

```
SELECT SUM (Salaire), MAX (Salaire), MIN (Salaire), AVG (Salaire)  
FROM Employe;
```

	SUM (Salaire)	MAX (Salaire)	MIN (Salaire)	AVG (Salaire)
1	281000.0	55000.0	25000.0	35125.0

› COUNT

– Quel est le nombre d'employés?

```
SELECT COUNT (*) AS 'Nombre d'employés'  
FROM EMPLOYE;
```

	Nombre d'employés
1	8



Les fonctions de calcul

- › Donner le nombre total de clients

```
SELECT COUNT ( IdCli ) FROM client
```

- › Donner le nombre total de 'PS1' vendus

```
SELECT SUM ( V.qte ) FROM vente V , produit P  
WHERE P.IdPro = V.IdPro AND P.nom = 'PS1'
```

- › Donner les noms des produits moins chers que la moyenne des prix de tous les produits

```
SELECT P1.nom FROM produit P1  
WHERE P1.prix <  
(  
    SELECT AVG ( P2.prix )  
    FROM produit P2  
)
```



Grouperement du résultat d'un SELECT (GROUP BY)

- › La clause **GROUP BY** permet de partitionner une table en plusieurs groupes
- › Chaque groupe contient les lignes qui ont la même valeur pour les attributs de partitionnement spécifiés après GROUP BY
- › Les fonctions de calcul opèrent sur chaque groupe de valeurs



Grouperement du résultat d'un SELECT (GROUP BY)

- › Donner pour chaque référence de produit la quantité totale vendue

```
SELECT V.IdPro, SUM ( V.qte )  
FROM vente V  
GROUP BY V.IdPro
```

- › Donner la quantité totale achetée par chaque client

```
SELECT V.IdCli, SUM ( V.qte )  
FROM vente V  
GROUP BY V.IdCli
```



Restriction des groupes en SQL

- › La clause **HAVING** permet de spécifier une condition de restriction des groupes
- › Elle sert à éliminer certains groupes, comme WHERE sert à éliminer des lignes
- › Donner les noms des marques dont le prix moyen des produits est < 5000

```
SELECT P.marque, AVG ( P.prix ) FROM produit P  
GROUP BY P.marque  
HAVING AVG ( P.prix )  $< 5000$ 
```

- › Donner les références des produits achetés en qte > 10 par plus de 50 clients

```
SELECT V.IdPro FROM vente V  
WHERE V.qte  $> 10$   
GROUP BY V.IdPro  
HAVING COUNT (*)  $> 50$ 
```