

420-C42

Langages d'exploitation des bases de données

Partie 9

DQL III

Interrogation : jointure

DQL III

jointure

- Le concept de jointure nous permet de mettre en relation des informations présentes dans des tables différentes. Par exemple, obtenir le nom d'un employé et le nom du département.
- En fait, la jointure est au cœur de l'algèbre relationnel (le moteur théorique derrière SQL) et par extension au centre du modèle relationnel.
- Pour bien comprendre les jointures, il importe de comprendre le principe de produit cartésien.

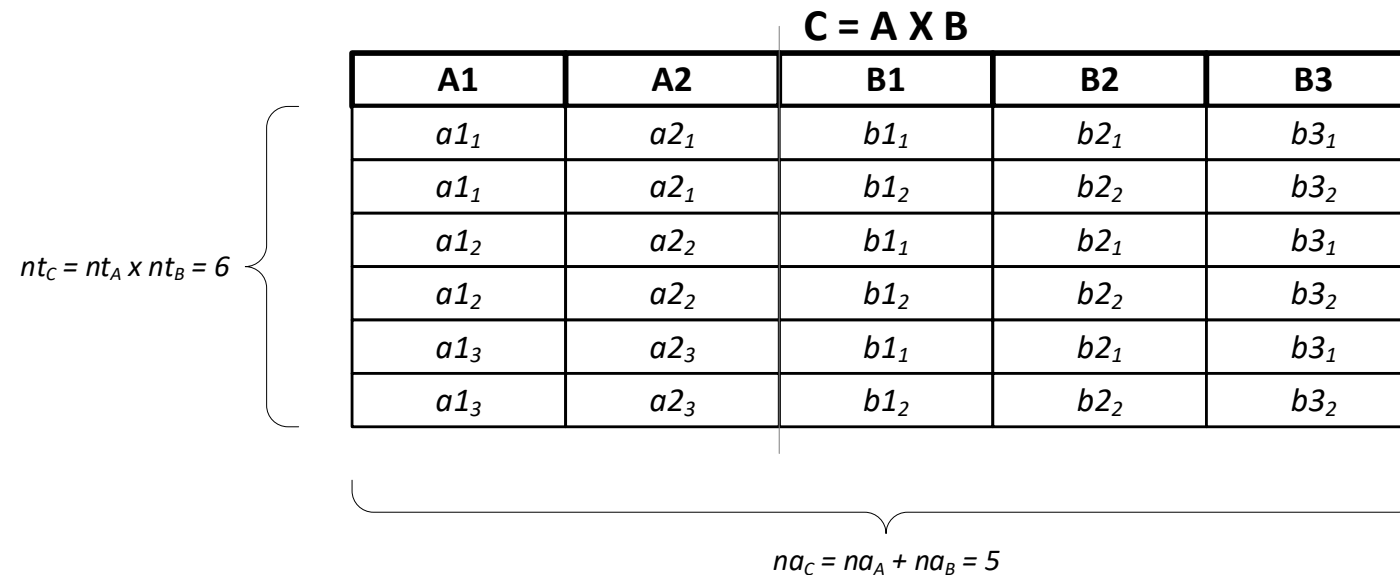
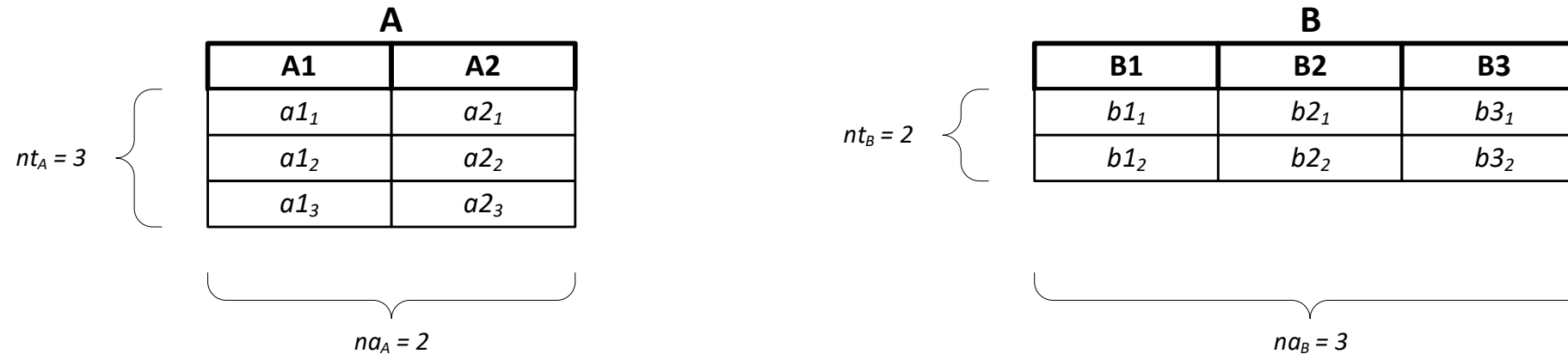
DQL III

produit cartésien

- Le produit cartésien est un opérateur binaire créant toutes les combinaisons des tuples provenant de 2 tables.
- Chaque combinaison ne fait pas nécessairement de sens. Malgré tout, cette combinatoire permet de créer une table pour laquelle il est possible de mettre en relation des données qui sont dans des tables distinctes.
- L'avantage de cette approche est qu'elle est très simple! Le résultat du produit cartésien est une table qui peut être utilisée comme n'importe quelle autre table et avec les mêmes outils.
- Attention, le produit cartésien est rarement utile tel quel. Généralement, seulement quelques tuples sont pertinents et le résultat est inutilement trop gros en mémoire.

DQL III

produit cartésien



Légende

- na_X = nombre d'attributs de la table X (nombre de colonnes)
- nt_X = nombre de tuples de la table X (nombre de lignes)

DQL III

- L'ordre d'appel change l'ordre des colonnes (et des lignes selon les SGBD) :

produit cartésien

$$C = A \times B$$

A1	A2	B1	B2	B3
$a1_1$	$a2_1$	$b1_1$	$b2_1$	$b3_1$
$a1_1$	$a2_1$	$b1_2$	$b2_2$	$b3_2$
$a1_2$	$a2_2$	$b1_1$	$b2_1$	$b3_1$
$a1_2$	$a2_2$	$b1_2$	$b2_2$	$b3_2$
$a1_3$	$a2_3$	$b1_1$	$b2_1$	$b3_1$
$a1_3$	$a2_3$	$b1_2$	$b2_2$	$b3_2$

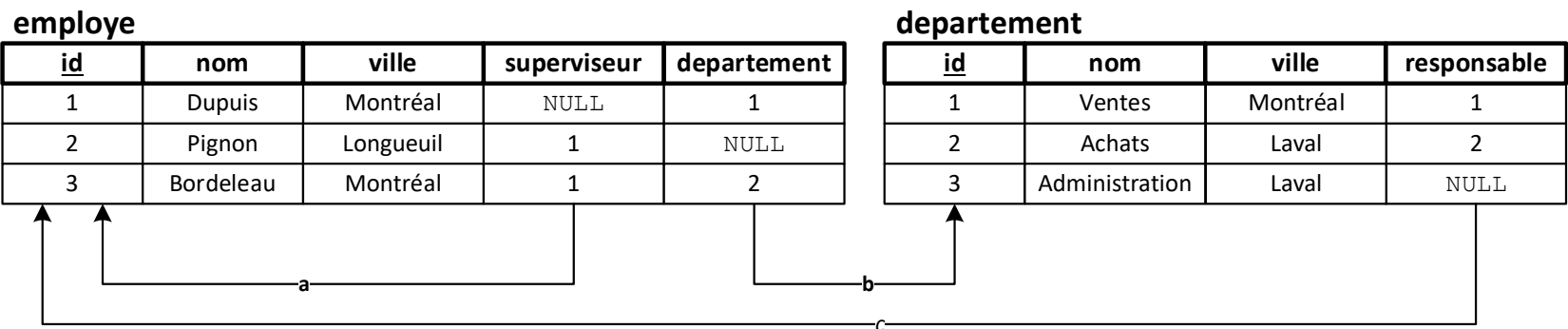
$$D = B \times A$$

B1	B2	B3	A1	A2
$b1_1$	$b2_1$	$b3_1$	$a1_1$	$a2_1$
$b1_2$	$b2_2$	$b3_2$	$a1_1$	$a2_1$
$b1_1$	$b2_1$	$b3_1$	$a1_2$	$a2_2$
$b1_2$	$b2_2$	$b3_2$	$a1_2$	$a2_2$
$b1_1$	$b2_1$	$b3_1$	$a1_3$	$a2_3$
$b1_2$	$b2_2$	$b3_2$	$a1_3$	$a2_3$

DQL III

produit cartésien

- Lors de la mise en relation des tuples des deux tables, ce n'est pas tous les tuples qui font du sens.
- C'est le concepteur de la requête qui doit faire du sens avec le résultat.



employe X departement

employe					departement			
<u>id</u>	nom	ville	superviseur	departement	<u>id</u>	nom	ville	responsable
1	Dupuis	Montréal	NULL	1	1	Ventes	Montréal	1
2	Pignon	Longueuil	1	NULL	1	Ventes	Montréal	1
3	Bordeleau	Montréal	1	2	1	Ventes	Montréal	1
1	Dupuis	Montréal	NULL	1	2	Achats	Laval	2
2	Pignon	Longueuil	1	NULL	2	Achats	Laval	2
3	Bordeleau	Montréal	1	2	2	Achats	Laval	2
1	Dupuis	Montréal	NULL	1	3	Administration	Laval	NULL
2	Pignon	Longueuil	1	NULL	3	Administration	Laval	NULL
3	Bordeleau	Montréal	1	2	3	Administration	Laval	NULL

Validation results (Green checkmark = valid, Red X = invalid):

Row	Valid	Invalid
1	✓	✓
2	✗	✗
3	✗	✗
4	✗	✗
5	✗	✓
6	✗	✗
7	✓	✗
8	✗	✗
9	✗	✗

Arrows labeled 'b' and 'c' indicate the validation process.

DQL III

produit cartésien

- Le langage SQL permet le produit cartésien ainsi :

-- syntaxe de base supporté par tous les SGBD

```
SELECT attribut [, ...]  
  FROM table1, table2 [, ...];
```

-- exemple de la diapositive précédente

```
SELECT *  
  FROM employe, departement;
```

-- syntaxe recommandée et supportée par certains SGBD (dont PostgreSQL)

```
SELECT *  
  FROM employe  
    CROSS JOIN departement;
```


DQL III

produit cartésien

- Considérant un produit cartésien, la requête fait généralement du sens par la sélection de lignes avec l'usage de la clause WHERE :

-- on cherche les employés responsable de département

```
SELECT *  
FROM employe,  
      departement  
WHERE employe.id =  
      departement.responsable;
```

```
SELECT * FROM employe, departement WHERE departement.responsable = employe.id;
```

employe					departement			
id	nom	ville	superviseur	departement	id	nom	ville	responsable
1	Dupuis	Montréal	NULL	1	1	Ventes	Montréal	1
2	Pignon	Longueuil	1	NULL	1	Ventes	Montréal	1
3	Bordeleau	Montréal	1	2	1	Ventes	Montréal	1
1	Dupuis	Montréal	NULL	1	2	Achats	Laval	2
2	Pignon	Longueuil	1	NULL	2	Achats	Laval	2
3	Bordeleau	Montréal	1	2	2	Achats	Laval	2
1	Dupuis	Montréal	NULL	1	3	Administration	Laval	NULL
2	Pignon	Longueuil	1	NULL	3	Administration	Laval	NULL
3	Bordeleau	Montréal	1	2	3	Administration	Laval	NULL

- Problème conceptuel lié au produit cartésien.
- Solution les jointures.

id	nom	ville	superviseur	departement	id	nom	ville	responsable
1	Dupuis	Montréal	NULL	1	1	Ventes	Montréal	1
2	Pignon	Longueuil	1	NULL	2	Achats	Laval	2

DQL III

jointures

- Conceptuellement, une jointure combine dans la même opération le produit cartésien et un ou plusieurs critères de sélection.
- Elle existe sous plusieurs formes :

jointure interne

jointure externe
complète

jointure externe
de gauche

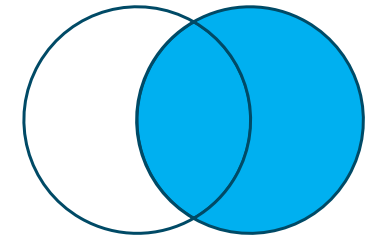
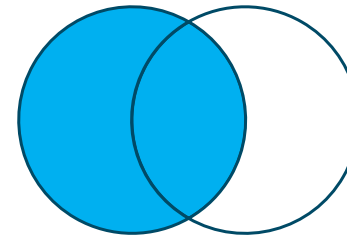
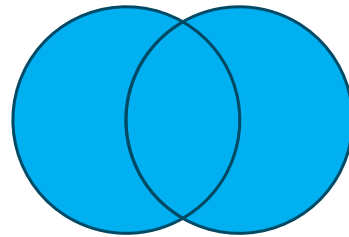
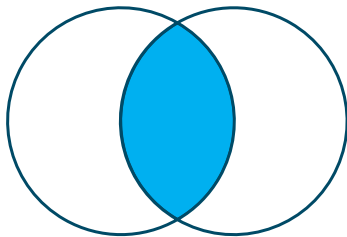
jointure externe
de droite

INNER JOIN

FULL OUTER JOIN

LEFT OUTER JOIN

RIGHT OUTER JOIN



DQL III

INNER JOIN

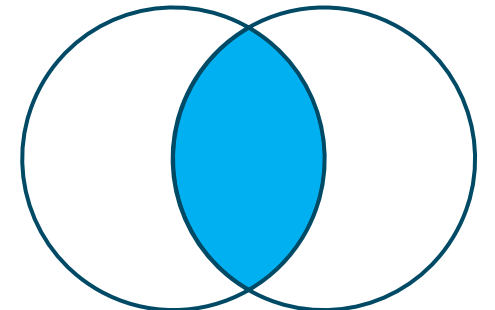
- La jointure interne correspond à :
 - un produit cartésien
 - avec sélection sur un critère de liaison (souvent une clé étrangère avec une clé primaire)

-- synopsis simplifié

```
SELECT attribut [, ...]  
FROM table1  
    [ INNER ] JOIN table2  
        ON table1.attribut = table2.attribut  
    [, ...];
```

-- équivalent avec la syntaxe fondamentale du produit cartésien

```
SELECT attribut [, ...]  
FROM table1, table2  
WHERE table1.attribut = table2.attribut;
```



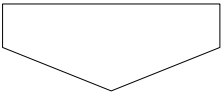
DQL III

INNER JOIN

- on cherche les employés et leur département
- seulement les employés ayant un département

```
SELECT * FROM employe INNER JOIN departement ON employe.departement = departement.id;
```

employe					departement			
id	nom	ville	superviseur	departement	id	nom	ville	responsable
1	Dupuis	Montréal	NULL	1	1	Ventes	Montréal	1
2	Pignon	Longueuil	1	NULL	1	Ventes	Montréal	1
3	Bordeleau	Montréal	1	2	1	Ventes	Montréal	1
1	Dupuis	Montréal	NULL	1	2	Achats	Laval	2
2	Pignon	Longueuil	1	NULL	2	Achats	Laval	2
3	Bordeleau	Montréal	1	2	2	Achats	Laval	2
1	Dupuis	Montréal	NULL	1	3	Administration	Laval	NULL
2	Pignon	Longueuil	1	NULL	3	Administration	Laval	NULL
3	Bordeleau	Montréal	1	2	3	Administration	Laval	NULL



id	nom	ville	superviseur	departement	id	nom	ville	responsable
1	Dupuis	Montréal	NULL	1	1	Ventes	Montréal	1
3	Bordeleau	Montréal	1	2	2	Achats	Laval	2

DQL III

LEFT OUTER JOIN

- La jointure externe de gauche correspond à :
 - un produit cartésien
 - avec sélection sur un critère de liaison
 - en insérant une seule instance de tous les tuples de gauche n'ayant pas de liaison (les données de droite sont mises à nulle).

-- synopsis simplifié

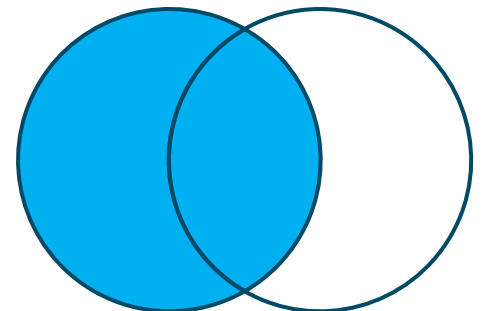
```
SELECT attribut [, ...]
```

```
FROM table1
```

```
LEFT [ OUTER ] JOIN table2
```

```
ON table1.attribut = table2.attribut
```

```
[, ...];
```



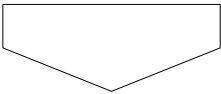
DQL III

LEFT OUTER JOIN

- on cherche les villes où résident les employés **et** où se trouvent les département
- en considérant **tous** les employés

```
SELECT * FROM employe LEFT OUTER JOIN departement WHERE employe.ville = departement.ville;
```

employe					departement			
id	nom	ville	superviseur	departement	id	nom	ville	responsable
1	Dupuis	Montréal	NULL	1	1	Ventes	Montréal	1
2	Pignon	Longueuil	1	NULL	1	Ventes	Montréal	1
3	Bordeleau	Montréal	1	2	1	Ventes	Montréal	1
1	Dupuis	Montréal	NULL	1	2	Achats	Laval	2
2	Pignon	Longueuil	1	NULL	2	Achats	Laval	2
3	Bordeleau	Montréal	1	2	2	Achats	Laval	2
1	Dupuis	Montréal	NULL	1	3	Administration	Laval	NULL
2	Pignon	Longueuil	1	NULL	3	Administration	Laval	NULL
3	Bordeleau	Montréal	1	2	3	Administration	Laval	NULL



id	nom	ville	superviseur	departement	id	nom	ville	responsable
1	Dupuis	Montréal	NULL	1	1	Ventes	Montréal	1
3	Bordeleau	Montréal	1	2	1	Ventes	Montréal	1
2	Pignon	Longueuil	1	NULL	NULL	NULL	NULL	NULL

DQL III

RIGHT OUTER JOIN

- La jointure externe de droite correspond à :
 - un produit cartésien
 - avec sélection sur un critère de liaison
 - en insérant une seule instance de tous les tuples de droite n'ayant pas de liaison (les données de gauche sont mises à nulle).

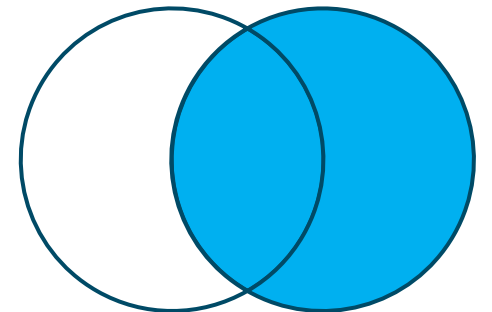
-- synopsis simplifié

```
SELECT attribut [, ...]
```

```
FROM table1
```

```
RIGHT [ OUTER ] JOIN table2
```

```
ON table1.attribut = table2.attribut  
[, ...];
```



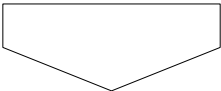
DQL III

RIGHT OUTER JOIN

- on cherche les villes où résident les employés **et** où se trouvent les département
- en considérant **tous** les département

```
SELECT * FROM employe RIGHT OUTER JOIN departement WHERE employe.ville = departement.ville;
```

employe					departement			
id	nom	ville	superviseur	departement	id	nom	ville	responsable
1	Dupuis	Montréal	NULL	1	1	Ventes	Montréal	1
2	Pignon	Longueuil	1	NULL	1	Ventes	Montréal	1
3	Bordeleau	Montréal	1	2	1	Ventes	Montréal	1
1	Dupuis	Montréal	NULL	1	2	Achats	Laval	2
2	Pignon	Longueuil	1	NULL	2	Achats	Laval	2
3	Bordeleau	Montréal	1	2	2	Achats	Laval	2
1	Dupuis	Montréal	NULL	1	3	Administration	Laval	NULL
2	Pignon	Longueuil	1	NULL	3	Administration	Laval	NULL
3	Bordeleau	Montréal	1	2	3	Administration	Laval	NULL



id	nom	ville	superviseur	departement	id	nom	ville	responsable
1	Dupuis	Montréal	NULL	1	1	Ventes	Montréal	1
3	Bordeleau	Montréal	1	2	1	Ventes	Montréal	1
NULL	NULL	NULL	NULL	NULL	2	Achats	Laval	2
NULL	NULL	NULL	NULL	NULL	3	Administration	Laval	NULL

DQL III

FULL OUTER JOIN

- La jointure externe complète correspond à :
 - un produit cartésien
 - avec sélection sur un critère de liaison
 - en insérant une seule instance de tous les tuples de droite et de gauche n'ayant pas de liaison (les données opposées sont mises à nulle).

-- synopsis simplifié

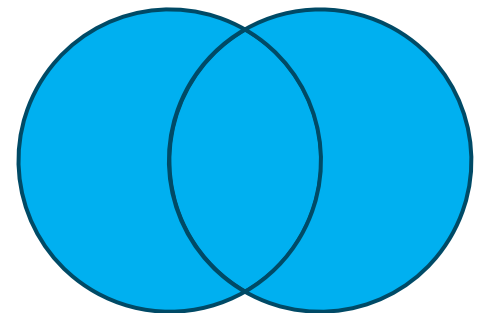
```
SELECT attribut [, ...]
```

```
FROM table1
```

```
    FULL [ OUTER ] JOIN table2
```

```
        ON table1.attribut = table2.attribut
```

```
    [, ...];
```



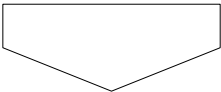
DQL III

FULL OUTER JOIN

- on cherche les villes où résident les employés **et** où se trouvent les département
- en considérant **tous** les employés et **tous** les département

```
SELECT * FROM employe FULL OUTER JOIN departement WHERE employe.ville = departement.ville;
```

employe					departement			
id	nom	ville	superviseur	departement	id	nom	ville	responsable
1	Dupuis	Montréal	NULL	1	1	Ventes	Montréal	1
2	Pignon	Longueuil	1	NULL	1	Ventes	Montréal	1
3	Bordeleau	Montréal	1	2	1	Ventes	Montréal	1
1	Dupuis	Montréal	NULL	1	2	Achats	Laval	2
2	Pignon	Longueuil	1	NULL	2	Achats	Laval	2
3	Bordeleau	Montréal	1	2	2	Achats	Laval	2
1	Dupuis	Montréal	NULL	1	3	Administration	Laval	NULL
2	Pignon	Longueuil	1	NULL	3	Administration	Laval	NULL
3	Bordeleau	Montréal	1	2	3	Administration	Laval	NULL



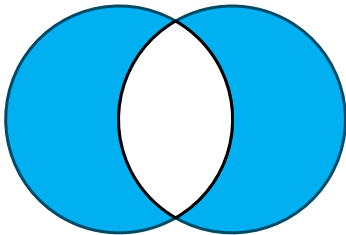
id	nom	ville	superviseur	departement	id	nom	ville	responsable
1	Dupuis	Montréal	NULL	1	1	Ventes	Montréal	1
3	Bordeleau	Montréal	1	2	1	Ventes	Montréal	1
NULL	NULL	NULL	NULL	NULL	2	Achats	Laval	2
NULL	NULL	NULL	NULL	NULL	3	Administration	Laval	NULL
2	Pignon	Longueuil	1	NULL	NULL	NULL	NULL	NULL

DQL III

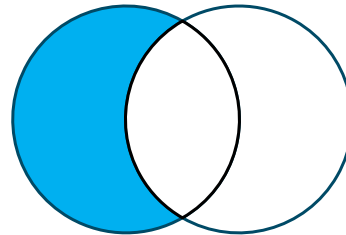
jointures externes exclusives

- Il est parfois nécessaire d'obtenir exclusivement le résultat de la jointure externe où il n'existe pas de partie commune.
- Elle existe sous plusieurs formes :

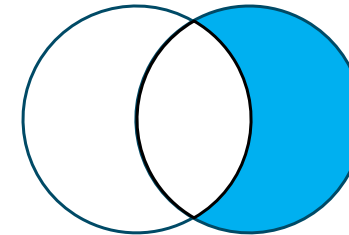
jointure externe
complète
exclusive



jointure externe
de gauche
exclusive



jointure externe
de droite
exclusive

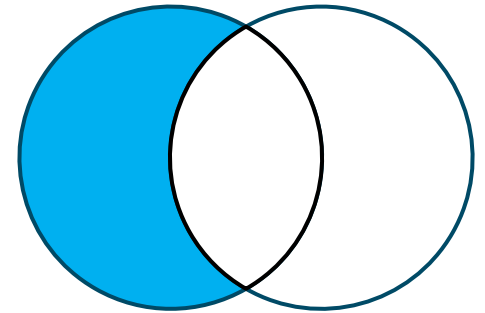


DQL III

jointures externes exclusives

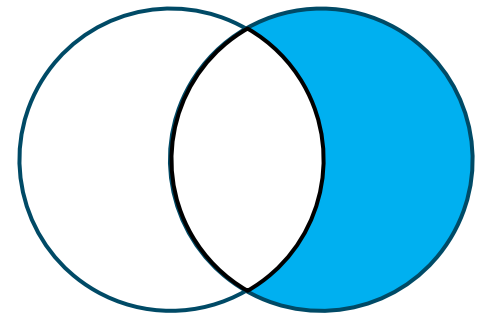
-- synopsis simplifié de la jointure externe exclusive de gauche

```
SELECT attribut [, ...]  
FROM table1  
    LEFT [ OUTER ] JOIN table2  
        ON table1.attribut = table2.attribut  
WHERE table2.attribut IS NULL;
```



-- synopsis simplifié de la jointure externe exclusive de droite

```
SELECT attribut [, ...]  
FROM table1  
    LEFT [ OUTER ] JOIN table2  
        ON table1.attribut = table2.attribut  
WHERE table1.attribut IS NULL;
```



DQL III

jointures externes exclusives

-- synopsis simplifié de la jointure externe exclusive complète

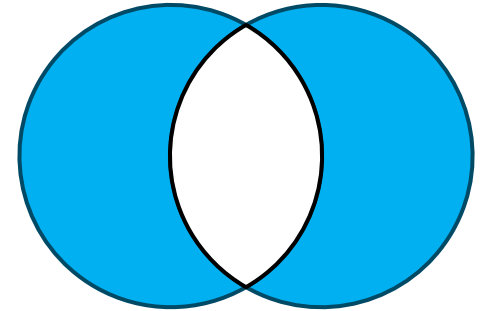
SELECT *attribut* [, ...]

FROM *table1*

FULL [OUTER] JOIN *table2*

ON *table1.attribut* = *table2.attribut*

WHERE *table1.attribut* IS NULL OR *table2.attribut* IS NULL;



DQL III

jointures avec la même table

- Il est parfois nécessaire de faire une jointure avec la même table pour deux contextes différents.
- Dans ce cas, l'alias de table est obligatoire.

-- synopsis simplifié

```
SELECT attribut [, ...]
```

```
FROM table AS contexte1
```

```
JOIN table AS contexte2          -- ou tout autre type de jointure
```

```
ON contexte1.colonne = contexte2.colonne;
```

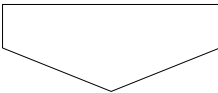
DQL III

jointures avec la même table

-- on cherche les employés et leur superviseur.

```
SELECT emp.nom AS "Employé", sup.nom AS "Superviseur"
FROM employe AS emp INNER JOIN employe AS sup
ON emp.superviseur = sup.id;
```

employe en tant que emp (employé supervisé)					employe en tant que sup (employé superviseur)				
id	nom	ville	superviseur	departement	id	nom	ville	superviseur	departement
1	Dupuis	Montréal	NULL	1	1	Dupuis	Montréal	NULL	1
2	Pignon	Longueuil	1	NULL	1	Dupuis	Montréal	NULL	1
3	Bordeleau	Montréal	1	2	1	Dupuis	Montréal	NULL	1
1	Dupuis	Montréal	NULL	1	2	Pignon	Longueuil	1	NULL
2	Pignon	Longueuil	1	NULL	2	Pignon	Longueuil	1	NULL
3	Bordeleau	Montréal	1	2	2	Pignon	Longueuil	1	NULL
1	Dupuis	Montréal	NULL	1	3	Bordeleau	Montréal	1	2
2	Pignon	Longueuil	1	NULL	3	Bordeleau	Montréal	1	2
3	Bordeleau	Montréal	1	2	3	Bordeleau	Montréal	1	2



Employé	Superviseur
Pignon	Dupuis
Bordeleau	Dupuis