Bases de Données Relationnelles

Les jointures avancées

L2

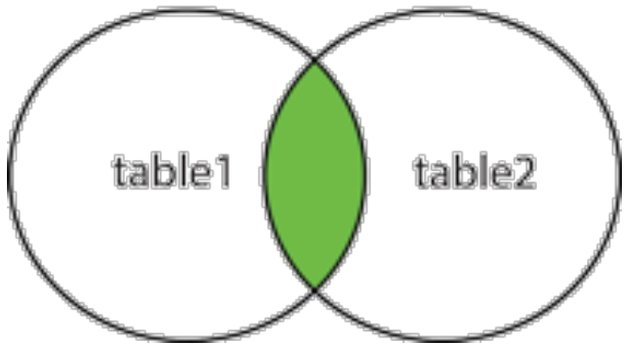
Rafael Angarita
Maitre de Conférences
rangarit@parisnanterre.fr

Université Paris Nanterre

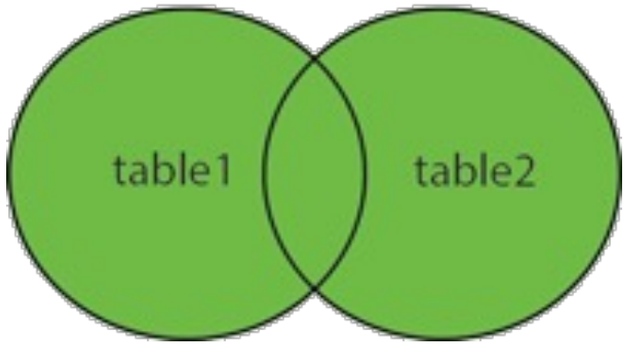
Université Paris Nanterre
UFR SEGMI

FROM : Jointures

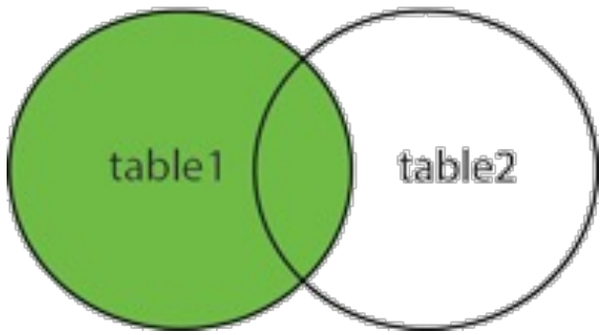
INNER JOIN



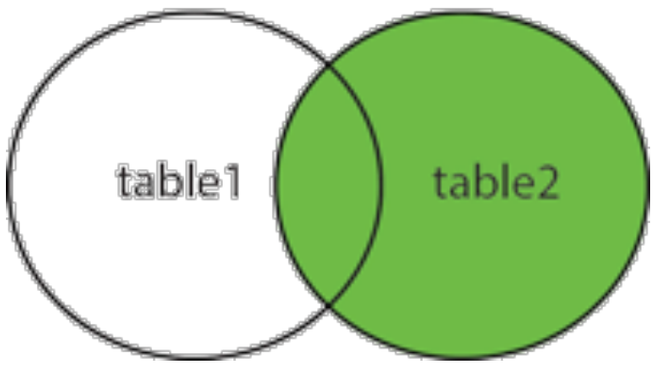
FULL OUTER JOIN



LEFT JOIN



RIGHT JOIN



Source : W3SCHOOLS

Quelles types de jointure avez-vous déjà utilisé ?

Le Self-join

Le Self-join

En SQL, un SELF JOIN correspond à une jointure d'une table avec elle-même.

Ce type de requête n'est pas si commun mais très pratique dans le cas où une table lie des informations avec des enregistrements de la même table :

- Pour interroger des données hiérarchiques
- our comparer une ligne avec d'autres lignes dans la même table.

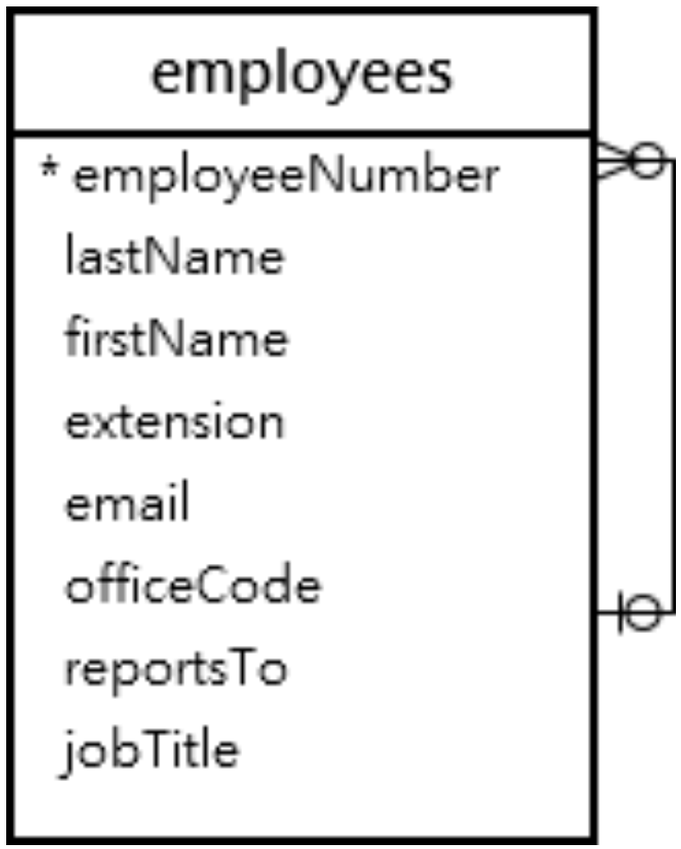
Pouvez-vous penser à un cas concret ?

Le Self-join

- Pour effectuer une auto-jointure, vous devez utiliser des alias de table pour ne pas répéter deux fois le même nom de table dans une même requête.
- Notez que référencer une table deux fois ou plus dans une requête sans utiliser d'alias de table provoquera une erreur.

Le self-join

- L'exemple classique : des employés qui peuvent être des managers d'autres employés.



La table a une relation 1..*
avec elle même :

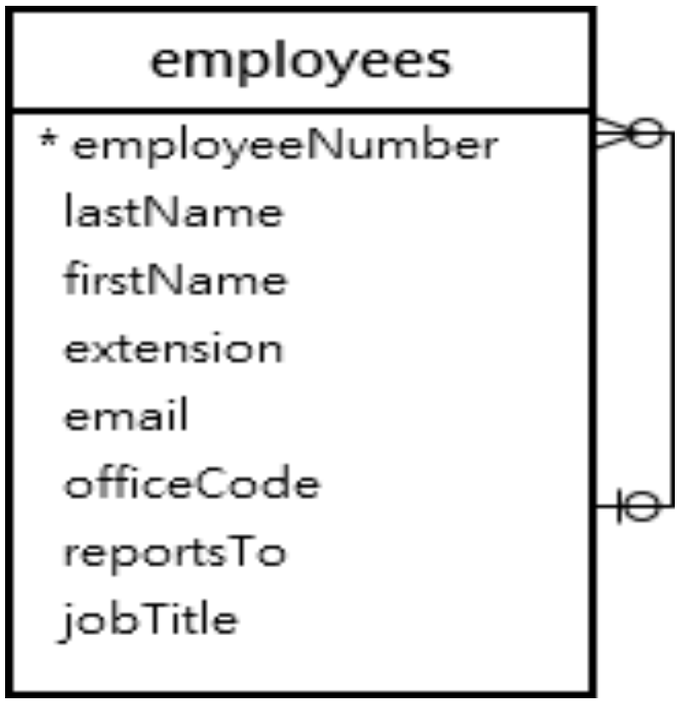
FOREIGN KEY ('reportsTo') **REFERENCES** 'employees' ('employeeNumber')

Donc *reportsTo* contient le numéro du manager

Requete : Obtenir la liste des employés avec leur manager

Le self-join

Requete : Obtenir la liste des employés avec leur manager



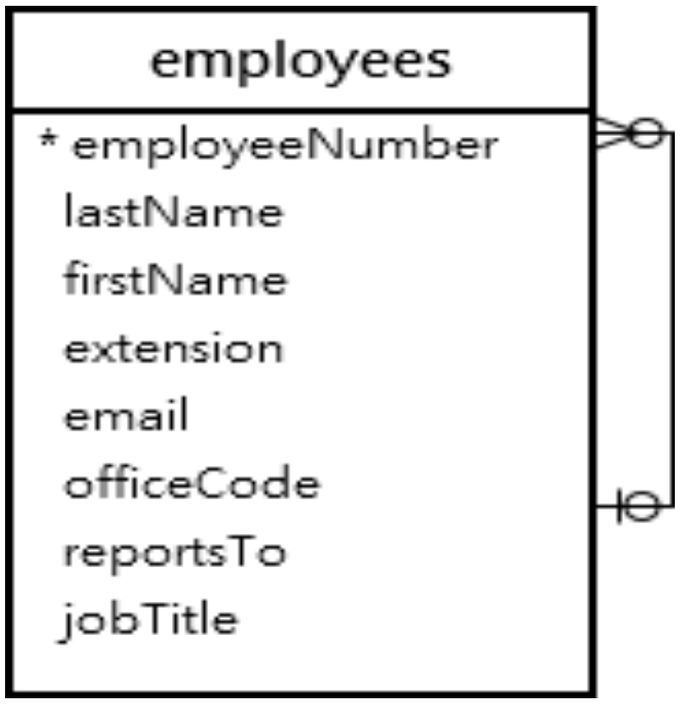
Employee	*Manager*
Jones, Barry	Bondur, Gerard
Bott, Larry	Bondur, Gerard
Castillo, Pamela	Bow, Anthony
Hernandez, Gerard	Bow, Anthony

```
SELECT CONCAT(e.lastname,', ',e.firstname) AS 'Employee',
       CONCAT(m.lastname,', ',m.firstname) AS 'Manager'
FROM employees e
INNER JOIN employees m ON m.employeeNumber = e.reportsto
ORDER BY manager
```

```
SELECT CONCAT(e.lastname,', ',e.firstname) AS 'Employee',
       CONCAT(m.lastname,', ',m.firstname) AS 'Manager'
FROM employees e, employees m WHERE e.reportsto = m.employeeNumber
ORDER BY manager
```

Le self-join

Requete : Obtenir la liste des employés avec leur manager



Employee	*Manager*
Jones, Barry	Bondur, Gerard
Bott, Larry	Bondur, Gerard
Castillo, Pamela	Bow, Anthony
Hernandez, Gerard	Bow, Anthony

```
SELECT CONCAT(lastname,', ',firstname) AS 'Employee',
       CONCAT(lastname,', ', firstname) AS 'Manager'
FROM employees WHERE employeeNumber = reportsto
ORDER BY manager
```

C'est une requete correcte ?

La syntaxe est correcte, mais cette requete retourne que les employés dont leur manager c'est eux mêmes !

Le self-join

Requete : Obtenir la liste des employés avec leur manager

Employee	*Manager*
Jones, Barry	Bondur, Gerard
Bott, Larry	Bondur, Gerard
Castillo, Pamela	Bow, Anthony
Hernandez, Gerard	Bow, Anthony

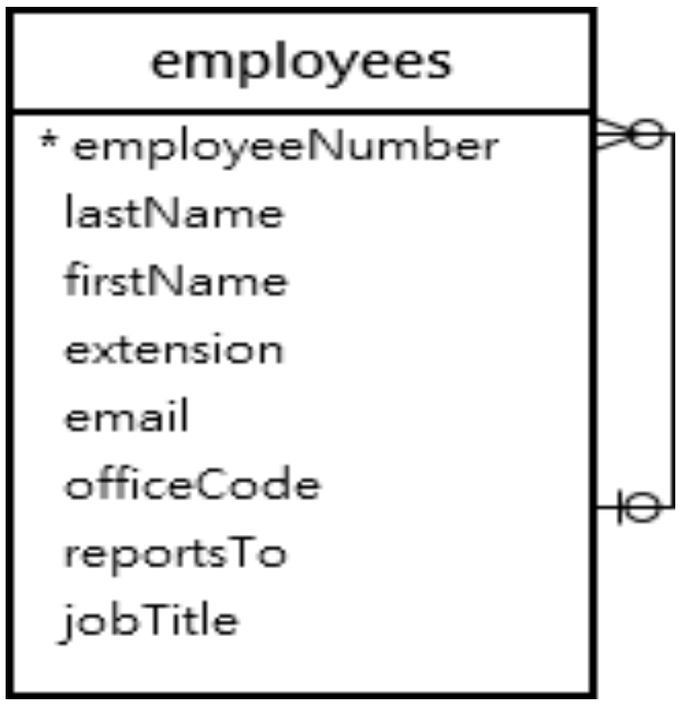
```
SELECT CONCAT(lastname,', ',firstname) AS 'Employee',
       CONCAT(lastname,', ', firstname) AS 'Manager'
FROM employees
INNER JOIN employees ON employeeNumber =.reportsto
ORDER BY manager
```

C'est une requete correcte ?

Error Code: 1066. Not unique table/alias: 'employees'

Le self-join

Requete : Obtenir la liste des employés avec leur manager



Employee	*Manager*
Jones, Barry	Bondur, Gerard
Bott, Larry	Bondur, Gerard
Castillo, Pamela	Bow, Anthony
Hernandez, Gerard	Bow, Anthony
Perez, Maria	NULL

Aussi les employés qui n'ont pas de manager, comme le président

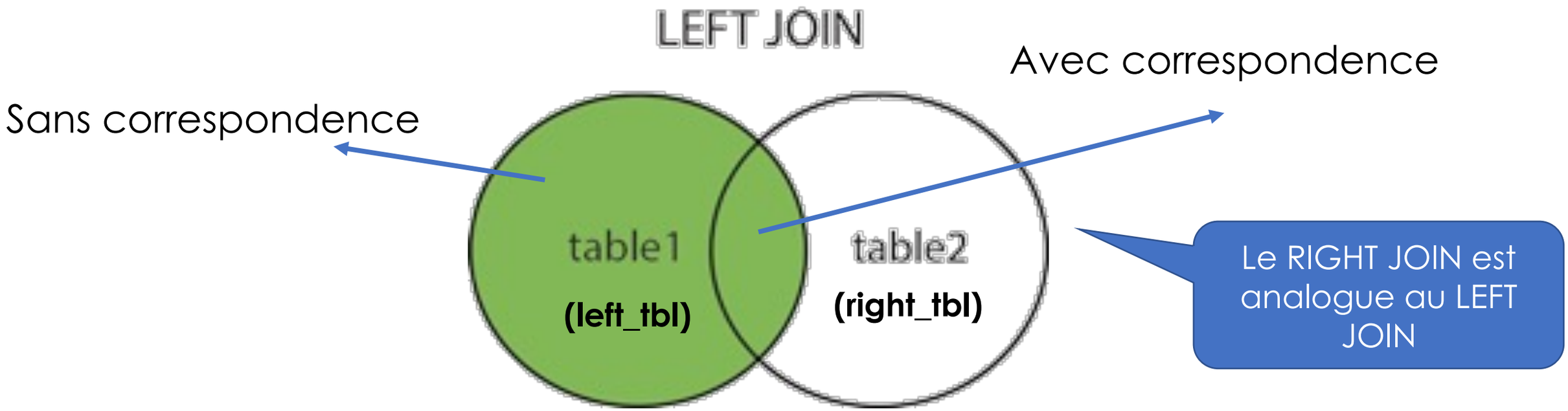
m.employeeNumber = e.reportsto

m.employeeNumber = NULL

Solution : les LEFT ou RIGHT JOIN

LEFT JOIN

Le LEFT JOIN permet de lister tous les résultats de la table de gauche (left = gauche) même s'il n'y a pas de correspondance dans la deuxième tables.

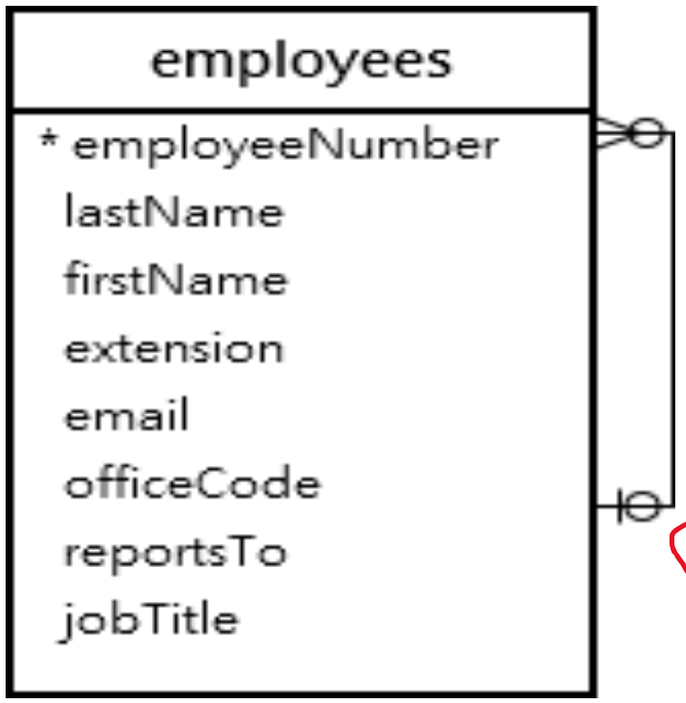



```
SELECT left_tbl.* FROM left_tbl LEFT JOIN
right_tbl ON left_tbl.id = right_tbl.id;
```

```
SELECT left_tbl.* FROM left_tbl LEFT JOIN
right_tbl ON left_tbl.id = right_tbl.id
WHERE right_tbl.id IS NULL;
```

Le self-join + left join

Requete : Obtenir la liste des employés avec leur manager



Employee	*Manager*
Jones, Barry	Bondur, Gerard
Bott, Larry	Bondur, Gerard
Castillo, Pamela	Bow, Anthony
Hernandez, Gerard	Bow, Anthony
Perez, Maria	NULL

Aussi les employés qui n'ont pas de manager, comme le président

m.employeeNumber = e.reportsto

m.employeeNumber = NULL

On a les résultats de la table de gauche même s'il n'y a pas de correspondence dans la table de droite

Solution : les LEFT ou RIGHT JOIN

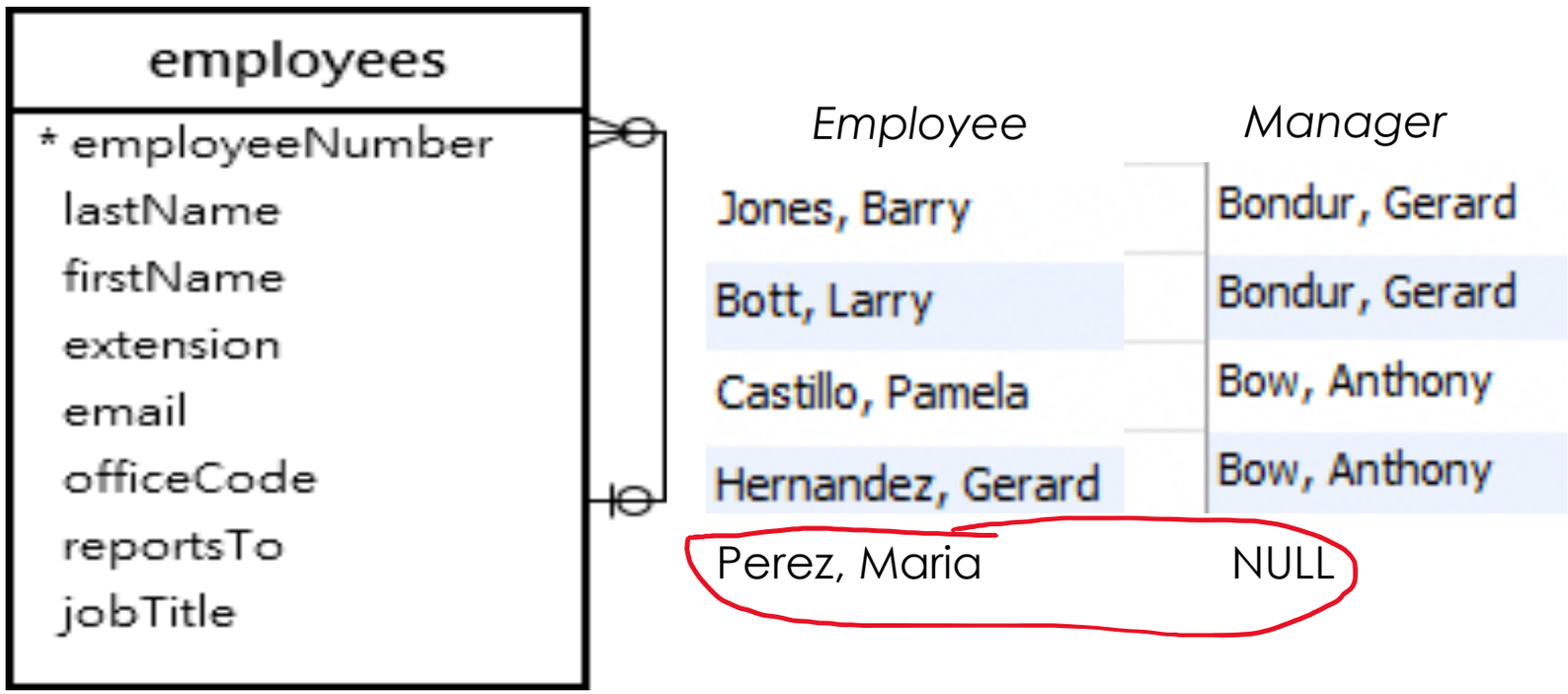
Table de gauche

```
SELECT CONCAT(e.lastname,', ',e.firstname) AS 'Employee',
       CONCAT(m.lastname,', ',m.firstname) AS 'Manager'
FROM employees e LEFT JOIN employees m
ON m.employeeNumber = e.reportsto
ORDER BY manager
```

Table de droite

Le self-join + left join

Requete : Obtenir la liste des employés qui n'ont pas de manager



Employee	*Manager*
Jones, Barry	Bondur, Gerard
Bott, Larry	Bondur, Gerard
Castillo, Pamela	Bow, Anthony
Hernandez, Gerard	Bow, Anthony
Perez, Maria	NULL

Quelle condition manque à la requête suivante ?

```
SELECT CONCAT(e.lastname,', ',e.firstname) AS 'Employee',
       CONCAT(m.lastname,', ',m.firstname) AS 'Manager'
FROM employees e LEFT JOIN employees m
ON m.employeeNumber = e.reportsto
WHERE m.employeeNumber is NULL
ORDER BY manager
```

Le Self-join :
Un autre cas d'usage

Le Self-join :comparisons dans la même table

Liste de clients qui se situent dans la même ville

city	customerName	customerName
Auckland	GiftsForHim.com	Kelly's Gift Shop
Auckland	Kelly's Gift Shop	Down Under Souveniers, Inc
Auckland	Kelly's Gift Shop	GiftsForHim.com
.	.	.
.	.	.

```
SELECT
    c1.city, c1.customerName, c2.customerName
FROM
    customers c1
        INNER JOIN
    customers c2 ON c1.city = c2.city
        AND c1.customername <> c2.customerName
ORDER BY c1.city;
```

Le Cross-Join

Le cross join

Meals
Omlet
Fried Egg
Sausage

CROSS JOIN

Drinks
Orange Juice
Tea
Cofee

MealName	DrinkName
Omlet	Orange Juice
Fried Egg	Orange Juice
Sausage	Orange Juice
Omlet	Tea
Fried Egg	Tea
Sausage	Tea
Omlet	Coffee
Fried Egg	Coffee
Sausage	Coffee

Les cas d'usages dans la vie réelle sont plus rares que pour les autres jointures :

- Certains rapports
- Génération de données de test

```
SELECT MealNale, DrinkName
FROM Meals CROSS JOIN Drinks;
```

```
SELECT MealName, DrinkName
FROM Meals CROSS JOIN Drinks
WHERE MealId = DrinkId;
```

Equivalent au INNER JOIN (même si peut être ça n'a pas trop de sens dans cette exemple)

En résumé

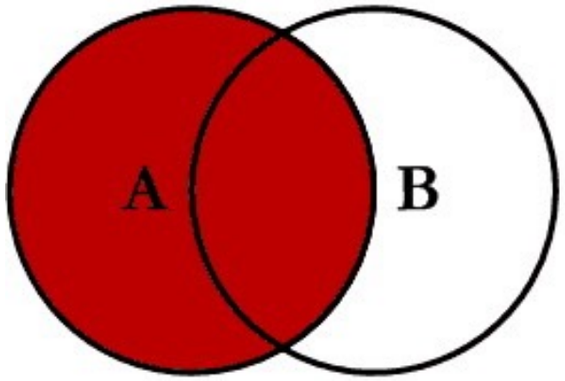
- INNER JOIN (ou avec la notation WHERE) reste le type de jointure le plus utilisé.

- On utilise le self JOIN dans le cas où une table lie des informations avec des enregistrements de la même table :
  - Pour interroger des données hiérarchiques
  - our comparer une ligne avec d'autres lignes dans la même table.

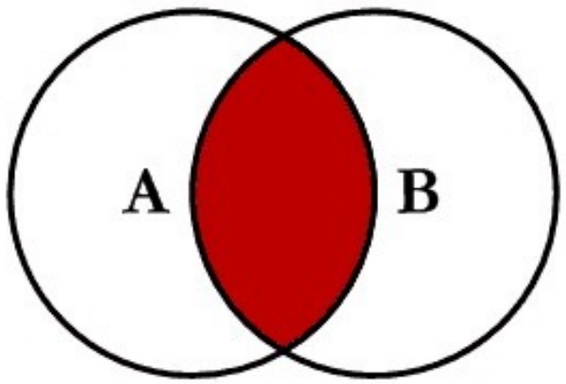
- L'utilisation du CROSS JOIN reste plus rare
- Les LEFT ou RIGHT JOINs peuvent être utilses, aussi avec la condition WHERE pour trouver les lignes sans correspondence.

SQL JOINS

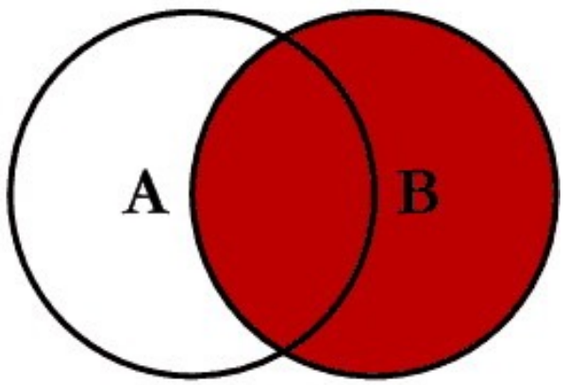
Source : https://www.codeproject.com



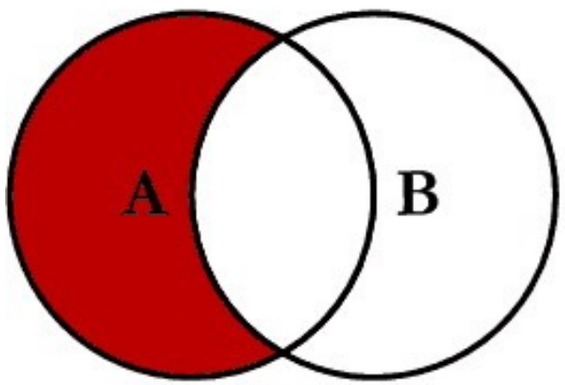

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
```



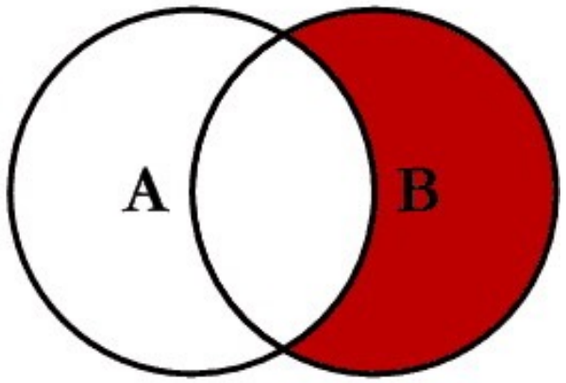

```
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
```



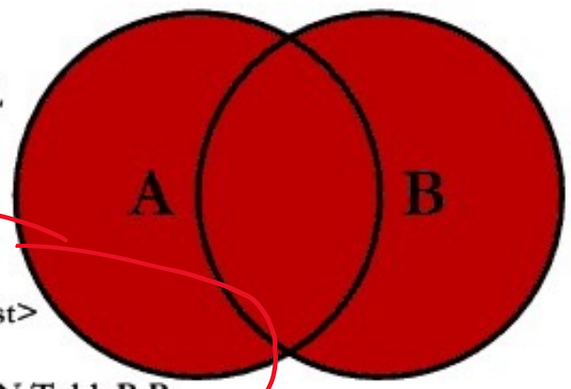

```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
```



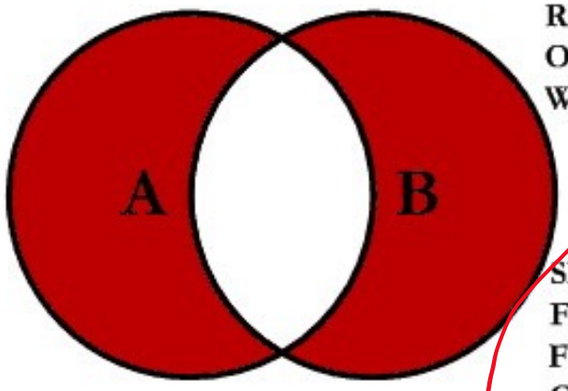

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
```




```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
```




```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
```



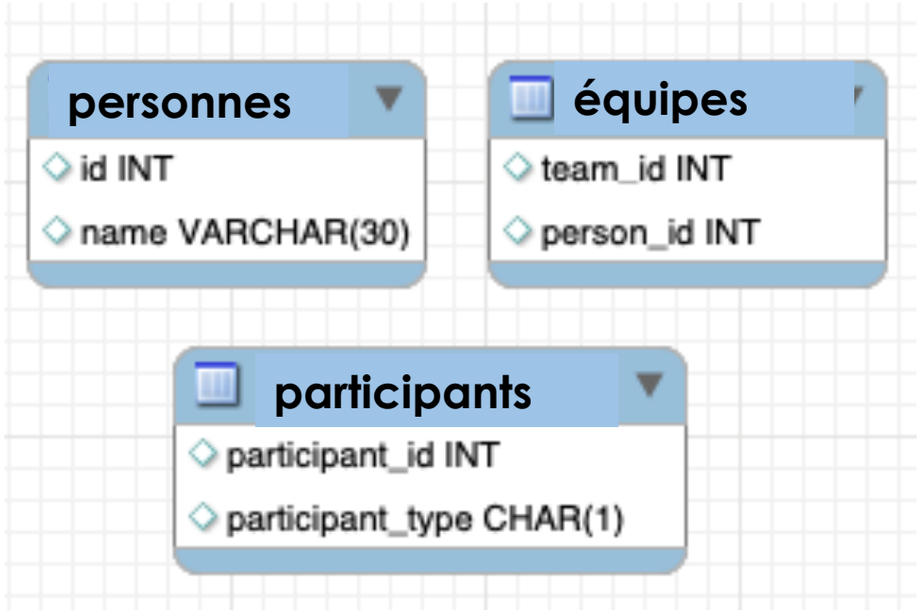

```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
OR B.Key IS NULL
```

© C.L. Moffatt, 2008

Ça n'existe pas en MySQL. On peut utiliser UNION, mais on verra ça plus tard

Les jointures multi-tables

Les jointures multi-tables : exemple 1



- Il y a des personnes et des équipes
- Une personne peut participer à la compétition individuellement (P) ou en équipe (*T*)
- La table *participants* pointe vers *personnes* ou *équipes* selon le type de participant

Afficher toutes les personnes qui participent à une compétition en tant qu'individus ou en tant que membres d'une équipe. Exemple résultat :

personnes *personnes*

participant_id	participant_type	id	name	team_id	person_id	id	name
1	T	NULL	NULL	1	2	2	Tom
1	T	NULL	NULL	1	1	1	Dan
3	P	3	Steve	NULL	NULL	NULL	NULL

participants *équipes*

Les jointures multi-tables : exemple 1

Une approche :

- Construir le résultat pas à pas !

Tout d'abord, on obtient les noms des participants individuels en rejoignant les tables *participants* et *personnes*

participant_id	participant_type	id	name
1	T	NULL	NULL
3	P	3	Steve

Quelle type de jointure on doit utiliser et pourquoi ?

```
SELECT *
   FROM participants c
     LEFT JOIN personnes p ON (c.participant_id = p.id AND
c.participant_type = 'P');
```

Les jointures multi-tables : exemple 1

Une approche :

- Construir le résultat pas à pas !

participant_id	participant_type	id	name	team_id	person_id
1	T	NULL	NULL	1	2
1	T	NULL	NULL	1	1
3	P	3	Steve	NULL	NULL

Ensuite, on obtient l'identifiant des personnes participant en tant que membres d'une équipe

```
SELECT *
   FROM particiants c
     LEFT JOIN personnes p ON (c.participant_id = p.id AND
c.participant_type = 'P')
     LEFT JOIN équipes t ON (c.participant_id = t.team_id AND
c.participant_type = 'T');
```

Les jointures multi-tables : exemple 1

Une approche :

- Construir le résultat pas à pas !

participant_id	participant_type	id	name	team_id	person_id	id	name
1	T	NULL	NULL	1	2	2	Tom
1	T	NULL	NULL	1	1	1	Dan
3	P	3	Steve	NULL	NULL	NULL	NULL

Finalement, on obtient les noms des personnes participant en tant que membres d'une équipe

On peut utiliser IFNULL(p.name, p2.name) pour obtenir que les noms

```
SELECT *
   FROM particiants c
     LEFT JOIN personnes p ON (c.participant_id = p.id AND
c.participant_type = 'P')
     LEFT JOIN équipes t ON (c.participant_id = t.team_id AND
c.participant_type = 'T')
     LEFT JOIN personnes p2 ON tp.person_id = p2.id
```