Jointures

Une jointure sert à combiner les enregistrements de deux tables ou plus, en faisant en sorte que les données ne se répètent pas pour chaque ligne. Il est souvent possible de construire cela avec une requête de type SELECT * FROM commande, produit WHERE ..., mais les jointures proposent des alternatives permettant de contrôler les données renvoyées.

Exemple utilisé pour la suite:

Table Employee

LastName	DepartmentID
Rafferty	31
Jones	33
Steinberg	33
Robinson	34
Smith	34
John	NULL

Table Department

DepartmentID	DepartmentName
31	Sales
33	Engineering
34	Clerical
35	Marketing

Les jointures internes

Comme il s'agit de la plus commune des jointures c'est celle qui s'exerce par défaut si on ne précise pas le type de jointure. Après le mot clef **ON** on doit préciser le critère de jointure.

Jointure implicite

```
Rem: le département marketing n'a pas d'employés.De même l'employé "John n'est
repris dans aucun département.
SELECT *
FROM   employee, department
WHERE  employee.DepartmentID = department.DepartmentID
```

jointure explicite :Inner join

```
SELECT *
FROM   employee
       INNER JOIN department
          ON employee.DepartmentID = department.DepartmentID;
```

Ces deux requêtes donnent:

Employee.LastName	Employee.DepartmentID	Department.DepartmentName	Department.DepartmentID
Robinson	34	Clerical	34
Jones	33	Engineering	33
Smith	34	Clerical	34
Steinberg	33	Engineering	33
Rafferty	31	Sales	31

Les jointures externes

Que faut-il modifier dans la requête pour obtenir une ligne "JOHN" avec aucune référence de departement associé dans la réponse ?

Left outer join

Tous les enregistrements de la table de gauche et les éléments de la table de droite s'ils existent sinon NULL

```
SELECT *
FROM   employee  LEFT OUTER JOIN department
          ON employee.DepartmentID = department.DepartmentIDSELECT *
```

Employee.LastName	Employee.DepartmentID	Department.DepartmentName	Department.DepartmentID
Jones	33	Engineering	33
Rafferty	31	Sales	31
Robinson	34	Clerical	34
Smith	34	Clerical	34
John	NULL	NULL	NULL
Steinberg	33	Engineering	33

Right join

Tous les enregistrements de la table de droite et les éléments de la table de gauche s'ils existent sinon NULL

```
FROM   employee RIGHT OUTER JOIN department
          ON employee.DepartmentID = department.DepartmentID
```

Employee.LastName	Employee.DepartmentID	Department.DepartmentName	Department.DepartmentID
Smith	34	Clerical	34
Jones	33	Engineering	33
Robinson	34	Clerical	34
Steinberg	33	Engineering	33
Rafferty	31	Sales	31
NULL	NULL	*Marketing*	*35*

La syntaxe de la jointure externe est la suivante :

```
SELECT ...
FROM   <table gauche>
       LEFT | RIGHT | FULL OUTER JOIN <table droite 1>
            ON <condition de jointure>
       [LEFT | RIGHT | FULL OUTER JOIN <table droite 2>
            ON <condition de jointure 2>]
...
```

*Les mots clefs **LEFT**, **RIGHT** et **FULL** indiquent la manière dont le moteur de requête doit effectuer la jointure externe. Il font référence à la table située à gauche (**LEFT**) du mot clef **JOIN***

*ou à la table située à droite (**RIGHT**) de ce même mot clef. Le mot **FULL** indique que la jointure externe est bilatérale.*

```
SELECT colonnes
FROM TGauche LEFT OUTER JOIN TDroite ON condition de jointure
```

On recherche toutes les valeurs satisfaisant la condition de jointure précisée dans prédicat, puis on rajoute toutes les lignes de la table *TGauche* qui n'ont pas été prises en compte au titre de la satisfaction du critère.

```
SELECT colonnes
FROM TGauche RIGHT OUTER JOIN TDroite ON condition de jointure
```

On recherche toutes les valeurs satisfaisant la condition de jointure précisée dans prédicat, puis on rajoute toutes les lignes de la table *TDroite* qui n'ont pas été prises en compte au titre de la satisfaction du critère.

```
SELECT colonnes
FROM TGauche FULL OUTER JOIN TDroite ON condition de jointure
```

On recherche toutes les valeurs satisfaisant la condition de jointure précisée dans prédicat, puis on rajoute toutes les lignes de la table *TGauche* et *TDroite* qui n'ont pas été prises en compte au titre de la satisfaction du critère.

*Lorsque nous étudions le modèle relationnel de notre base de données nous voyons que le modèle physique des données, répercute les clefs des tables maîtres en tant que clefs étrangères des tables pour lesquelles une jointure est nécessaire. En utilisant la jointure entre clefs primaires et clefs secondaires basée sur l'égalité des valeurs des colonnes nous exécutons ce que les professionnels du SQL appelle une **jointure naturelle**.*

Exercice

Créons une base

```
 create database formations;
```

Elle contient 3 tables

Deux tables pour les participants

```
+----------------------+
| Tables_in_formations |
+----------------------+
| etudiants            |
| formateurs           |
+----------------------+
```

Structures:

describe etudiants;

```
+---------------+-------------+------+-----+---------+----------------+
| Field         | Type        | Null | Key | Default | Extra          |
+---------------+-------------+------+-----+---------+----------------+
| idEtudiant    | int(11)     | NO   | PRI | NULL    | auto_increment |
| nomEtudiant   | varchar(35) | NO   | MUL | NULL    |                |
| prenomEtudiant | varchar(35) | YES  |     | NULL    |                |
+---------------+-------------+------+-----+---------+----------------+
```

describe formateurs;

```
+-----------------+-------------+------+-----+---------+----------------+
| Field           | Type        | Null | Key | Default | Extra          |
+-----------------+-------------+------+-----+---------+----------------+
| idFormateur     | tinyint(4)  | NO   | PRI | NULL    | auto_increment |
| nomFormateur    | varchar(35) | NO   | MUL | NULL    |                |
| prenomFormateur | varchar(35) | YES  |     | NULL    |                |
+-----------------+-------------+------+-----+---------+----------------+
```

etudiants:

```
+------------+-------------+----------------+
| idEtudiant | nomEtudiant | prenomEtudiant |
+------------+-------------+----------------+
|          1 | Porte       | Sarha          |
|          2 | Provist     | Alain          |
|          3 | Hatan       | Charles        |
|          4 | Fair        | Lucie          |
+------------+-------------+----------------+
```

formateurs

```
+-------------+--------------+-----------------+
| idFormateur | nomFormateur | prenomFormateur |
+-------------+--------------+-----------------+
|           1 | Magne        | Charles         |
|           2 | Kier         | Jacques         |
|           3 | Deci         | Alain           |(pas d'étudiant!)
+-------------+--------------+-----------------+
```

une table groupe qui organise les formations

```
+--------------+-------------+------+-----+---------+----------------+
| Field        | Type        | Null | Key | Default | Extra          |
+--------------+-------------+------+-----+---------+----------------+
| idGroupe     | smallint(6) | NO   | PRI | NULL    | auto_increment |
| numEtudiant  | int(11)     | NO   |     | NULL    |                |
| NumFormateur | tinyint(4)  | NO   |     | NULL    |                |
+--------------+-------------+------+-----+---------+----------------+
```

contenu de groupes

```
+----------+-------------+--------------+
| idGroupe | numEtudiant | NumFormateur |
+----------+-------------+--------------+
|        1 |           4 |            1 |
|        2 |           1 |            1 |
|        3 |           2 |            1 |
|        4 |           3 |            2 | pas de NumFormateur=3
+----------+-------------+--------------+
```

Questions

Combien de participants pour chaque formateur ?

```
+-----------+--------------------+
| Formateur | count(numetudiant) |
+-----------+--------------------+
| Deci      |                  0 |
| Magne     |                  3 |
| Kier      |                  1 |
+-----------+--------------------+
```

Liste des formateurs avec ou sans étudiants

```
+-----------+---------+---------+
| Formateur | prenom  | nom     |
+-----------+---------+---------+
| Magne     | Lucie   | Fair    |
| Magne     | Sarha   | Porte   |
| Magne     | Alain   | Provist |
| Kier      | Charles | Hatan   |
| Deci      | NULL    | NULL    |
+-----------+---------+---------+
```