# Les jointures externes

Pour ramener toutes les lignes de la table maître, il faut le préciser au SGBDR même s'il n'y a pas de correspondance dans les tables filles. Dans ce cas, le système ramène les colonnes sans correspondance à NULL.

Exemple avec JOIN dans lequel il faut ajouter le mot LEFT (ou RIGHT) pour préciser que l'on veut toutes les lignes de la table à gauche du mot JOIN, ici Tarifs.

```
SELECT H.Libelle, TYP.Description, T.Prix,
CH.NumChambre
FROM Tarifs AS T LEFT OUTER JOIN Hotels AS
H ON H.idHotel = T.hotel
LEFT OUTER JOIN TypesChambre AS TYP ON
TYP.idTypeChambre = T.typeChambre
LEFT OUTER JOIN Chambres AS CH ON
CH.Hotel = H.idHotel AND
CH.TypeChambre = TYP.idTypeChambre;
```

| Libelle | Description | Prix | NumChambre |
|---|---|---|---|
| Ski Hotel | 1 lit double avec douche et WC séparé | 69.99 | 5 |
| Ski Hotel | 1 lit double avec bain et WC séparé | 79.99 | 6 |
| Ski Hotel | 1 lit double large avec bain et WC séparé | 89.99 | NULL |
| Art Hotel | 1 lit simple avec douche | 57.49 | 1 |

Le tarif de 89.99 pour Ski Hotel apparaît sans numéro de chambre, ce qui veut dire qu'il n'existe pas de chambre avec un lit double large avec bain et WC séparés dans l'hôtel Ski Hotel.

# La jointure naturelle

Laisse le système associer les colonnes qui ont le même nom entre elles.
Il n’est pas conseillé d’utiliser ce type de jointure. il y a un risque que deux colonnes portent le même nom sans être des colonnes clés, donc le résultat de la requête sera faussé.

```
SELECT H.Libelle, TYP.Description, T.Prix, CH.NumChambre
FROM Tarifs T NATURAL JOIN Hotels H
NATURAL JOIN TypesChambre TYP
NATURAL JOIN Chambres CH;
```

```
SELECT H.Libelle, TYP.Description, T.Prix,
CH.NumChambre
FROM Tarifs T JOIN Hotels H
JOIN Chambres CH
JOIN TypesChambre TYP;
```

# La jointure croisée

C’est la jointure la plus simple qui existe. On ne précise pas de critère de rapprochement, le système ramène donc le produit cartésien des deux tables.

*SELECT Libelle, Description FROM hotels, typeschambre;*

# Les instructions de condition CASE et IF (IIF)

L’instruction CASE peut être utilisée dans la sélection de données afin de formater un résultat selon le contenu d’une colonne.
Il est possible de tester les valeurs d’une colonne et d’agir en fonction du contenu de celle-ci.
Il existe deux façons de rédiger le CASE, soit en testant des constantes par rapport à une colonne soit en ajoutant des conditions (=, <, >, … ) sur cette colonne.

Test d’une constante

SELECT <colonne 1>, <colonne 2>,
CASE <colonne 3>
WHEN <constante 1> THEN <valeur affichée>
WHEN <constante 2> THEN <valeur affichée>
WHEN <constante 3> THEN <valeur affichée>
... ...
ELSE <valeur par défaut>
END AS <entête de colonne>
FROM <table1>, <table2> ... ...
WHERE ... ...

Test avec valeur d’une condition

SELECT <colonne 1>, <colonne 2>
, IF(<colonne 3> <condition> <constante ou valeur>, <valeur si vrai>,
<valeur si faux>) AS <entête de colonne>
FROM <table1>, <table2> … …
WHERE … … ;

```
SELECT TypeChambre, Description, DateDebut
, CASE
  WHEN DateDebut >= '2017-04-01' AND
DateDebut < '2017-10-01'
THEN 'été 2017'
  WHEN DateDebut >= '2017-10-01' AND
DateDebut < '2018-04-01'
THEN 'hiver 2018'
END AS Saison
FROM Tarifs INNER JOIN TypesChambre ON
TypesChambre.idTypeChambre
= Tarifs.typeChambre
ORDER BY typeChambre, DateDebut;
```

```
SELECT TypeChambre, Description, DateDebut
, IF (DateDebut < '2017-10-01', 'été 2017', 'Hiver
2018') as Saison
FROM Tarifs INNER JOIN TypesChambre ON
TypesChambre.idTypeChambre =
Tarifs.typeChambre
ORDER BY typeChambre, DateDebut;
```

| TypeChambre | description | DateDebut | Saison |
|---|---|---|---|
| 1 | 1 lit simple avec douche | 2017-04-01 | été 2017 |
| 1 | 1 lit simple avec douche | 2017-04-16 | été 2017 |
| 1 | 1 lit simple avec douche | 2017-04-16 | été 2017 |
| 1 | 1 lit simple avec douche | 2017-10-01 | hiver 2018 |
| 1 | 1 lit simple avec douche | 2017-10-01 | hiver 2018 |
| 1 | 1 lit simple avec douche | 2017-12-15 | hiver 2018 |

https://dev.mysql.com/doc/refman/5.7/en/control-flow-functions.html

*Table Hotels_FR*

| idHotel | Libelle | Etoile |
|---|---|---|
| 1 | Ski Hotel | * |
| 2 | Art Hotel | ** |
| 3 | Rose Hotel | *** |
| 4 | Lions Hotel | **** |

*Table Hotels_GB*

| idHotel | Libelle | Etoile |
|---|---|---|
| 1 | Lochness Hotel | * |
| 2 | Art Hotel | ** |
| 3 | Rose Hotel | *** |
| 4 | Lions Hotel | ***** |
| 5 | Trafalgar Hotel | ***** |

# Les opérateurs ensemblistes

## L’opérateur UNION

UNION permet de fusionner les données de plusieurs tables.

*SELECT Etoile FROM Hotels_FR*
*UNION*
*SELECT Etoile FROM Hotels_GB;*

Si on veut récupérer tous les types d’étoiles contenus dans ces deux tables.

Automatiquement, le système va supprimer les doublons et afficher les valeurs unitairement.

| Etoile |
| --- |
| * |
| ** |
| *** |
| **** |
| ***** |

il faut retenir sur l’opérateur UNION :

Il faut que les colonnes sélectionnées dans les différentes tables soient du même type et de la même taille.

On peut joindre autant de tables que l’on veut.

Il y a une suppression automatique des *doublons* par le système.

Si on sélectionne deux colonnes avec la requête ci-dessous, le système supprimera les lignes en double sur les deux colonnes (couples Art Hotel/** et Rose Hotel/***).

```
SELECT Libelle, Etoile FROM Hotels_FR
UNION
SELECT Libelle, Etoile FROM Hotels_GB;
```

| Libelle | Etoile |
|---|---|
| Art Hotel | ** |
| Lions Hotel | **** |
| Lions Hotel | ***** |
| Lochness Hotel | * |
| Rose Hotel | *** |
| Ski Hotel | * |
| Trafalgar Hotel | ***** |

L’union peut s’apparenter à un SELECT DISTINCT sur les deux tables.

On ne veut pas de trois étoiles dans la première table Hotels_FR.

On veut tous les hôtels qui ne sont pas trois étoiles dans les deux tables.

```
SELECT Libelle, Etoile FROM Hotels_FR
WHERE Etoile <> '***'
UNION
SELECT Libelle, Etoile FROM Hotels_GB;
```

Rose hotel est sélectionné car
il se trouve aussi dans l'autre table

| Libelle | Etoile |
|---|---|
| Art Hotel | ** |
| Lions Hotel | **** |
| Lions Hotel | ***** |
| Lochness Hotel | * |
| Rose Hotel | *** |
| Ski Hotel | * |
| Trafalgar Hotel | ***** |

```
SELECT Libelle, Etoile FROM Hotels_FR
WHERE Etoile <> '***'
UNION
SELECT Libelle, Etoile FROM Hotels_GB
WHERE Etoile <> '***';
```

| Libelle | Etoile |
|---|---|
| Art Hotel | ** |
| Lions Hotel | **** |
| Lions Hotel | ***** |
| Lochness Hotel | * |
| Ski Hotel | * |
| Trafalgar Hotel | ***** |

# L’opérateur INTERSECT

L’intersection va sélectionner les données qui sont à la fois dans la table 1 **et** dans la table 2 puis supprimer également les doublons.

Il est également possible d’ajouter les clauses WHERE et ORDER BY comme pour l’UNION.

```
SELECT Libelle, Etoile FROM Hotels_FR
INTERSECT
SELECT Libelle, Etoile FROM Hotels_GB;
```

| Libelle | Etoile |
|---|---|
| Art Hotel | ** |
| Rose Hotel | *** |

```
SELECT Libelle, Etoile FROM Hotels_FR
WHERE Etoile <> '***'
INTERSECT
SELECT Libelle, Etoile FROM Hotels_GB
WHERE Etoile <> '***';
```

| Libelle | Etoile |
|---|---|
| Art Hotel | ** |

!!!À noter que l’opérateur INTERSECT n’est pas implémenté dans mysql !!!
il peut être remplacé par ce type d’ordre

```
SELECT Libelle, Etoile FROM Hotels_FR AS H1
WHERE EXISTS (SELECT Libelle, Etoile FROM
Hotels_GB AS H2
WHERE H1.Libelle = H2.Libelle
AND H1.Etoile = H2.Etoile);
```

# L’opérateur EXCEPT

L’EXCEPT est le contraire de l’INTERSECT, seules les lignes de la première table qui ne sont pas dans la deuxième table seront sélectionnées.

Attention, l’ordre des instructions avec EXCEPT est important, contrairement aux ordres UNION et INTERSECT, puisqu’il se base sur la première table.

Il est également possible d’ajouter les clauses WHERE et ORDER BY comme pour l’UNION.

```
SELECT Libelle, Etoile FROM Hotels_GB
EXCEPT
SELECT Libelle, Etoile FROM Hotels_FR;
```

| Libelle | Etoile |
|---|---|
| Lions Hotel | ***** |
| Lochness Hotel | * |
| Trafalgar Hotel | ***** |

```
SELECT Libelle, Etoile FROM Hotels_FR
EXCEPT
SELECT Libelle, Etoile FROM Hotels_GB;
```

| Libelle | Etoile |
|---|---|
| Lions Hotel | **** |
| Ski Hotel | * |

L’opérateur EXCEPT n’est pas implémenté dans Mysql. Dans ce cas, il peut être remplacé par ce type d’ordre :

```
SELECT Libelle, Etoile FROM Hotels_FR AS H1
 WHERE NOT EXISTS (SELECT Libelle, Etoile
 FROM Hotels_GB AS H2
 WHERE H1.Libelle = H2.Libelle
 AND H1.Etoile = H2.Etoile);
```