

# Jointures avancées

# Table des matières

<b>I. Contexte</b>	<b>3</b>
<b>II. Jointure : rappel</b>	<b>3</b>
<b>III. Exercice : Appliquer la notion</b>	<b>5</b>
<b>IV. Produit cartésien</b>	<b>5</b>
<b>V. Exercice : Appliquer la notion</b>	<b>7</b>
<b>VI. Jointures multi-tables</b>	<b>7</b>
<b>VII. Exercice : Appliquer la notion</b>	<b>9</b>
<b>VIII. Jointure externe</b>	<b>10</b>
<b>IX. Exercice : Appliquer la notion</b>	<b>14</b>
<b>X. Essentiel</b>	<b>15</b>
<b>XI. Auto-évaluation</b>	<b>15</b>
A. Exercice final .....	15
B. Exercice : Défi .....	18
<b>Solutions des exercices</b>	<b>19</b>

## I. Contexte

**Durée** : 2h

**Environnement de travail** : DB Fiddle

**Pré-requis** : Aucun

### Contexte

Lorsque des informations sont présentes dans deux tables différentes, on peut les combiner à l'aide de jointures. Un exemple typique serait de récupérer l'ensemble des commandes passées par un client en particulier.

Mais les jointures sont plus puissantes : elles permettent par exemple de combiner les informations issues d'un nombre quelconque de tables. Elles permettent aussi de répondre à des questions complexes, comme pour savoir quels sont les clients qui n'ont jamais passé de commandes.

Tout au long de ces modules, vous découvrirez la puissance des jointures, ainsi que des cas d'utilisation plus spécifiques.

## II. Jointure : rappel

### Objectif

- Savoir réaliser une jointure d'une table avec elle-même.

### Mise en situation

Une base de données relationnelle est en général composée de plusieurs tables liées par des clés étrangères. L'opération de jointure permet de combiner les données de plusieurs tables grâce aux clés étrangères.

Mais l'opération de jointure peut aussi être utilisée pour joindre une table avec elle-même, par exemple pour gérer les relations hiérarchiques au sein d'un système de commentaire. Chaque commentaire qui est une réponse à un autre commentaire référence son commentaire « *parent* ». Une jointure permettra de retrouver le commentaire parent de chaque commentaire enfant.

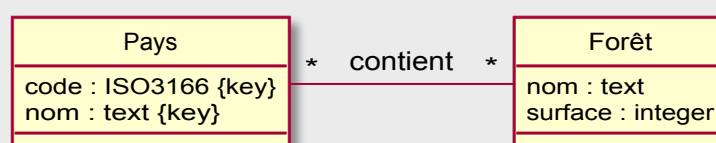
On appelle auto-jointure une telle jointure, et vous allez découvrir comment les réaliser dans ce module.

### Rappel Jointure

Une jointure est une opération permettant de consulter les données de plusieurs tables en se basant sur les valeurs identiques de certains des attributs de ces tables, en général une clé étrangère et une clé primaire.

```
1 SELECT *
2 FROM R1
3 INNER JOIN R2 ON <condition>
```

### Exemple



```
1 Pays(#code:char(2), nom:text) avec nom clé
2 Forêt (#id:integer, nom:text, surface:integer)
3 Contient (#pays=>Pays, #foret=>Forêt)
```

```
1 CREATE TABLE pays (
2 code CHAR(2) PRIMARY KEY,
3 nom VARCHAR(255) UNIQUE NOT NULL
4 );
5
6 CREATE TABLE foret (
7 id INTEGER PRIMARY KEY,
8 nom VARCHAR(255) NOT NULL,
9 surface INTEGER NOT NULL
10 );
11
12 CREATE TABLE contient (
13 pays CHAR(2) REFERENCES pays(code),
14 foret INTEGER REFERENCES foret(id),
15 PRIMARY KEY (foret, pays)
16 );
17
18 INSERT INTO pays VALUES ('FR', 'France');
19 INSERT INTO pays VALUES ('ES', 'Espagne');
20 INSERT INTO foret VALUES (1, 'Broceliande', 12000);
21 INSERT INTO foret VALUES (2, 'Pyrénéenne', 23000);
22 INSERT INTO foret VALUES (3, 'Compiégnoise', 14000);
23 INSERT INTO contient VALUES ('FR',1);
24 INSERT INTO contient VALUES ('FR',2);
25 INSERT INTO contient VALUES ('FR',3);
26 INSERT INTO contient VALUES ('ES',2);
27
28 SELECT p.code, f.nom
29 FROM pays p JOIN contient c
30 ON p.code = c.pays
31 JOIN foret f
32 ON c.foret = f.id
```

code	nom
FR	Broceliande
FR	Pyrénéenne
FR	Compiégnoise
ES	Pyrénéenne

### Complément Auto-jointure

Une auto-jointure est la jointure d'une table avec elle-même.

Pour réaliser une auto-jointure, on doit utiliser les alias des tables. Pour donner un alias à une table, on note dans la clause FROM l'alias après le nom de la relation : FROM nom\_table alias.

### Exemple Auto-jointure

```
1 SELECT e1.nom
2 FROM employe e1 INNER JOIN employe e2
3 ON e1.nom=e2.nom
```

### III. Exercice : Appliquer la notion

On implémente ce modèle et peuple les tables avec les requêtes suivantes :

```
1 CREATE TABLE assurance (  
2 num_siret INTEGER PRIMARY KEY,  
3 denomination VARCHAR(50),  
4 adresse_siege_social VARCHAR(200)  
5 );  
6  
7 CREATE TABLE locataire (  
8 id INTEGER PRIMARY KEY,  
9 prenom VARCHAR(50),  
10 nom VARCHAR(50),  
11 assurance INTEGER REFERENCES assurance(num_siret));  
12  
13 INSERT INTO assurance  
14 VALUES (839204929, 'Pro Gama', '13, rue des Peupliers 42920 Saint-Jean-le-Cèdre');  
15  
16 INSERT INTO assurance  
17 VALUES (839201119, 'Faim', '41, avenue des Tulipes 11001 Maintes-sur-Champs');  
18  
19 INSERT INTO assurance  
20 VALUES (839283114, 'Xitour', '26, chemins de la mine 61920 Sourcieux');  
21  
22 INSERT INTO locataire  
23 VALUES(47289, 'Jean', 'Durand', 839283114);  
24  
25 INSERT INTO locataire  
26 VALUES(47211, 'Anne', 'Durand', 839283114);  
27  
28 INSERT INTO locataire  
29 VALUES(47111, 'Camille', 'Dupont', 839201119);  
30  
31 INSERT INTO locataire  
32 VALUES(47291, 'Marie', 'Martin', 839204929);  
33  
34
```

#### Question

Réaliser une jointure qui permet d'afficher les prénoms et noms des locataires et la dénomination de leur assurance.

### IV. Produit cartésien

#### Objectifs

- Comprendre le concept de produit cartésien.
- Savoir réaliser un produit cartésien en SQL.

#### Mise en situation

Un produit cartésien désigne en général l'ensemble des combinaisons possibles entre deux ensembles.

Par exemple, lors d'un tournoi sportif, si toutes les équipes d'un pays doivent rencontrer toutes les équipes d'un autre pays, on réalisera un produit cartésien sur ces deux ensembles de villes. La grille des matchs contiendra ainsi, pour chaque ville du premier pays, une entrée avec chaque ville du second pays.

Ce concept est aussi exploité par SQL pour combiner les informations de plusieurs tables, en général dans le but de réaliser des jointures, et vous découvrirez comment dans ce module.

**Définition**    **Produit cartésien**

Un produit cartésien de plusieurs tables est l'ensemble de toutes les concaténations ordonnées possibles des enregistrements de ces tables.

**Syntaxe**

```
1 SELECT *
2 FROM R1, R2, Ri
```

**Exemple**

R1	#A	B	C=>R2
	1	Alpha	10
	2	Bravo	10
	3	Charlie	20
	4	Delta	

R2	#X	Y
	10	Echo
	20	Fox
	30	Golf

A	B	C	X	Y
1	Alpha	10	10	Echo
1	Alpha	10	20	Fox
1	Alpha	10	30	Golf
2	Bravo	10	10	Echo
2	Bravo	10	20	Fox
2	Bravo	10	30	Golf
3	Charlie	20	10	Echo
3	Charlie	20	20	Fox
3	Charlie	20	30	Golf
4	Delta		10	Echo
4	Delta		20	Fox
4	Delta		30	Golf

**Produit (R1, R2)**

```
SELECT *
FROM R1, R2
```

*Exemple de produit (SQL et Algèbre)*

**Attention**    **Explosion combinatoire du produit cartésien**

On tiendra compte du fait que réaliser des produits cartésiens amène à une explosion combinatoire. En effet, si on dispose de  $m$  tables chacune contenant de l'ordre de  $n$  éléments, alors le résultat contiendra de l'ordre de  $n^m$  éléments.

**Remarque**    **Usage du produit cartésien**

Le produit cartésien est rarement utilisé tel quel ; généralement on le restreint à un sous-ensemble via des restrictions comme dans le cas des jointures.

**Remarque**    **Sémantique des syntaxes de jointure**

Nous avons vu qu'il y avait deux types de syntaxe pour les jointures : la syntaxe avec JOIN et la syntaxe avec les clauses FROM et WHERE.

Les deux syntaxes donnent des résultats équivalents, mais la sémantique est différente : la première syntaxe avec JOIN est entièrement dédiée aux jointures et n'est utilisée que pour cela.

La seconde syntaxe correspond à une restriction d'un produit cartésien sur des valeurs d'égalité sur certains attributs.

## V. Exercice : Appliquer la notion

On se donne les tables suivantes stockant des informations sur des missions et des agents secrets :

```

1 CREATE TABLE agentSecret(
2 pseudonyme VARCHAR(20) PRIMARY KEY,
3 nom VARCHAR(20),
4 prenom VARCHAR(20)
5 );
6
7 CREATE TABLE mission(
8 nom_de_code VARCHAR(20) PRIMARY KEY,
9 lieu_suppose VARCHAR(200)
10 );
11
12 INSERT INTO agentSecret
13 VALUES ('OSS 117', 'Bonisseur de la Bath', 'Hubert');
14
15 INSERT INTO agentSecret
16 VALUES ('SCEP 2421', 'El Akmar Betouche', 'Larmina');
17
18 INSERT INTO agentSecret
19 VALUES ('MOSS_1411', 'Koulechov', 'Dolores');
20
21 INSERT INTO mission
22 VALUES ('Silence Radio', 'Rio de Janeiro');
23
24 INSERT INTO mission
25 VALUES ('Oisillons', 'Le Caire');

```

On souhaite savoir qui pourrait être envoyé en mission.

### Question

Réaliser un produit cartésien qui permet d'afficher les pseudonyme des agents secrets et le nom de code des missions.

## VI. Jointures multi-tables

### Objectif

- Savoir combiner des informations présentes dans plus de deux tables.

**Mise en situation**

Si les jointures sont en général utilisées pour récupérer une information présente dans une seconde table, il y a des situations où une troisième, voire une quatrième table entre en jeu.

Imaginez par exemple une base de données musicale : les morceaux sont stockés dans une table, avec leur titre et leur durée, les albums sont stockés dans une seconde table, et les artistes sont stockés dans une troisième table.

Pour récupérer le nom d'un artiste à partir d'un morceau, il faut d'abord récupérer l'album, et enfin l'artiste. En d'autres termes, il faut combiner les données de trois tables différentes.

C'est ce qu'on appelle une jointure multi-table.

**Syntaxe** Jointure par la clause JOIN

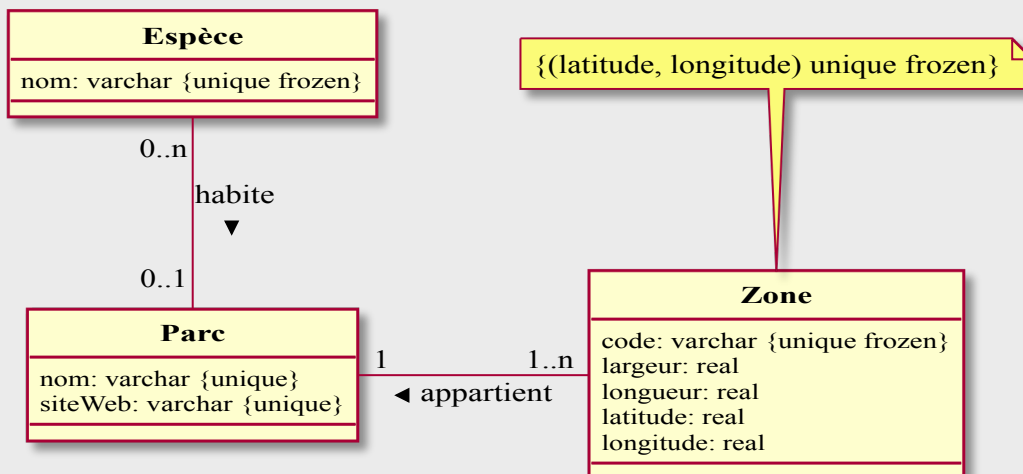
```
1 SELECT *
2 FROM a
3 JOIN b ON a.att_1 = b.att_1
4 JOIN c ON b.att_2 = c.att2;
```

**Syntaxe** Jointure par la clause WHERE

```
1 SELECT *
2 FROM a, b, c
3 WHERE a.att_1 = b.att_1
4 AND b.att_2 = c.att2;
```

**Exemple**

On se donne la situation suivante de la gestion de parcs se décomposant en zones peuplées d'espèces d'êtres vivants.



```
1 CREATE TABLE parc(
2   nom VARCHAR(50) PRIMARY KEY,
3   site_web VARCHAR(50) UNIQUE NOT NULL
4 );
5
6 CREATE TABLE zone_parc(
7   code VARCHAR(50) PRIMARY KEY,
8   parc VARCHAR(50) REFERENCES parc(nom),
9   largeur REAL,
10  longueur REAL,
```

```

11 latitude REAL,
12 longitude REAL
13 );
14
15 CREATE TABLE espece(
16 nom VARCHAR(50) PRIMARY KEY,
17 parc VARCHAR(50) REFERENCES parc(nom)
18 );
19
20 INSERT INTO parc VALUES ('Parc Floral de Paris', 'https://www.parcfloraldeparis.com/fr');
21 INSERT INTO parc VALUES ('Jardin des Plantes de Paris',
22 'https://www.jardindesplantesdeparis.fr/fr');
23
24 INSERT INTO zone_parc VALUES ('NFI1', 'Jardin des Plantes de Paris', 20, 50, 48.50, 2.21);
25 INSERT INTO zone_parc VALUES ('AB35', 'Jardin des Plantes de Paris', 20, 50, 48.84, 2.35);
26
27 INSERT INTO zone_parc VALUES ('VEF1', 'Parc Floral de Paris', 100, 20, 48.834594, 2.442491);
28 INSERT INTO zone_parc VALUES ('FJ29', 'Parc Floral de Paris', 20, 80, 48.8010341, 2.502201);
29
30 INSERT INTO espece VALUES ('Ausmas', 'Parc Floral de Paris');
31 INSERT INTO espece VALUES ('Orchidée Cymbidium', 'Parc Floral de Paris');
32
33 INSERT INTO espece VALUES ('Orchidée Asconceda', 'Jardin des Plantes de Paris');
34 INSERT INTO espece VALUES ('Orchidée Phalaenopsis', 'Jardin des Plantes de Paris');

```

On peut connaître le nom du parc et le code des zones du parc contenant des *Phalænopsis* ainsi :

```

1 SELECT p.nom, z.code
2 FROM espece e, parc p, zone_parc z
3 WHERE e.nom = 'Orchidée Phalaenopsis'
4 AND e.parc = p.nom
5 AND z.parc = p.nom;

```

1	nom	code
2	-----	----
3	Jardin des Plantes de Paris	NFI1
4	Jardin des Plantes de Paris	AB35

#### Remarque **Ordre des tables**

L'ordre des tables dans la jointure n'influence pas les résultats obtenus.

## VII. Exercice : Appliquer la notion

On se donne les tables suivantes d'un système de suivi d'intervention d'urgence :

```

1 CREATE TABLE intervention (
2 id INTEGER PRIMARY KEY,
3 categorie VARCHAR(200),
4 longitude FLOAT,
5 latitude FLOAT
6 );
7
8 CREATE TABLE vehicule(
9 immatriculation VARCHAR(14) PRIMARY KEY,
10 genre VARCHAR(50)
11 );

```

```

12
13 CREATE TABLE selection(
14 heure TIME,
15 intervention INTEGER REFERENCES Intervention(id),
16 vehicule VARCHAR(14) REFERENCES Vehicule(immatriculation),
17 PRIMARY KEY (heure, intervention, vehicule)
18 );
19
20 INSERT INTO intervention
21 VALUES (1042, 'Feux', 48.8662, 2.3078);
22
23 INSERT INTO intervention
24 VALUES (1183, 'Secours à Victimes', 48.8641, 2.3011);
25
26 INSERT INTO intervention
27 VALUES (1190, 'Feux', 48.8641, 2.3078);
28
29 INSERT INTO vehicule
30 VALUES ('48VD249103', 'FPT');
31
32 INSERT INTO vehicule
33 VALUES ('81VD239107', 'FPT');
34
35 INSERT INTO vehicule
36 VALUES ('18VD239117', 'VSAV');
37
38 INSERT INTO vehicule
39 VALUES ('82VD291899', 'VSAV');
40
41 INSERT INTO selection
42 VALUES ('13:34:01', 1183, '18VD239117');
43
44 INSERT INTO selection
45 VALUES ('13:44:11', 1183, '82VD291899');
46
47 INSERT INTO selection
48 VALUES ('07:22:19', 1190, '81VD239107');
49
50 INSERT INTO selection
51 VALUES ('07:23:11', 1190, '81VD239107');

```

## Question

On veut connaître les genres des véhicules sélectionnés pour chaque intervention.

Écrire une requête qui réalise une jointure multiple permettant d'obtenir pour chaque intervention son identifiant, sa catégorie et les immatriculations et genres de véhicules utilisés.

## VIII. Jointure externe

### Objectifs

- Comprendre la différence entre jointure interne et jointure externe.
- Connaître les cas d'utilisation des jointures externes.

## Mise en situation

Imaginez que vous gérez une bases de données pour le compte d'un refuge d'animaux. Les animaux recueillis sont enregistrés dans une table, et peuvent être adoptés par des personnes, enregistrées dans une autre table.

Une jointure classique permettra de retrouver le nom des animaux adoptés par une personne en particulier, mais comment faire pour trouver toutes les personnes enregistrées qui n'ont adopté aucun animal ?

Le problème paraît épineux, car les jointures permettent de combiner les informations de plusieurs tables. Or, une personne qui n'a adopté aucun animal n'a justement aucune information associée dans la table des animaux !

Les jointures externes répondent spécifiquement à ce problème.

### **Fondamental** Type de jointures

Il existe différents types de jointure :

- INNER JOIN
- LEFT JOIN
- RIGHT JOIN
- OUTER JOIN

Les jointures simples JOIN sont des jointures du type INNER JOIN.

### **Définition** Jointure externe

La jointure externe entre R1 et R2 est une jointure qui produit une relation R3 à laquelle on ajoute les enregistrement de R1 et de R2 exclus par la jointure, en complétant avec des valeurs nulles pour les attributs de l'autre relation.

### **Définition** Jointure externe gauche

La jointure externe gauche entre R1 et R2 est une jointure externe pour laquelle on ajoute seulement les tuples de R1 (c'est-à-dire la relation de gauche) ayant été exclus.

Synonymes : Jointure gauche

### **Définition** Jointure externe droite

La jointure externe droite entre R1 et R2 est une jointure externe pour laquelle on ajoute seulement les tuples de R2 (c'est-à-dire la relation de droite) ayant été exclus.

Bien entendu, une jointure externe droite peut être réécrite par une jointure externe gauche (et réciproquement) en substituant les relations opérandes R1 et R2.

Synonymes : Jointure droite

### **Syntaxe** Jointure externe, gauche ou droite

Pour exprimer une jointure externe on se base sur la syntaxe INNER JOIN en utilisant à la place OUTER JOIN, LEFT OUTER JOIN ou RIGHT OUTER JOIN.

**Exemple** Jointure externe gauche

```
1 SELECT Num
2 FROM Avion
3 LEFT OUTER JOIN Vol
4 ON Avion.Num=Vol.Num
```

Cette requête permet de sélectionner tous les avions, y compris ceux non-affectés à un vol.

**Remarque**

Remarquons que « Avion LEFT OUTER JOIN Vol » est équivalent à « Vol RIGHT OUTER JOIN Avion » en terme de résultat.

Intuitivement, on préfère utiliser la jointure gauche pour sélectionner tous les tuple du côté N d'une relation 1 : N, même si il ne sont pas référencés ; et la jointure droite pour sélectionner tous les tuples d'une relation 0 : N, y compris ceux qui ne font pas de référence. Cette approche revient à toujours garder à gauche de l'expression JOIN la relation « principale », i.e. celle dont on veut tous les tuples, même s'ils ne référencent pas (ou ne sont pas référencés par) la relation « secondaire ».

**Exemple**

R1	#A	B	C=>R2
	1	Alpha	10
	2	Bravo	10
	3	Charlie	20
	4	Delta	

R2	#X	Y
	10	Echo
	20	Fox
	30	Golf

R	A	B	C	X	Y
	1	Alpha	10	10	Echo
	2	Bravo	10	10	Echo
	3	Charlie	20	20	Fox

Jointure (R1 ,R2 ,R1 .C=R2 . X)

```
SELECT *
FROM R1 INNER JOIN R2
ON R1 .C=R2 . X
```

Exemple de jointure (SQL et Algèbre)

**Exemple**

R1	#A	B	C=>R2
	1	Alpha	10
	2	Bravo	10
	3	Charlie	20
	4	Delta	

R2	#X	Y
	10	Echo
	20	Fox
	30	Golf

R	A	B	C	X	Y
	1	Alpha	10	10	Echo
	2	Bravo	10	10	Echo
	3	Charlie	20	20	Fox
	4	Delta			

```
JointureExterneGauche (R1 , R2 , R1 . C=R2 . X)

SELECT *
FROM R1 LEFT OUTER JOIN R2
ON R1 . C=R2 . X
```

Exemple de jointure externe gauche (SQL et Algèbre)

**Exemple**

R1	#A	B	C=>R2
	1	Alpha	10
	2	Bravo	10
	3	Charlie	20
	4	Delta	

R2	#X	Y
	10	Echo
	20	Fox
	30	Golf

R	A	B	C	X	Y
	1	Alpha	10	10	Echo
	2	Bravo	10	10	Echo
	3	Charlie	20	20	Fox
				30	Golf

```
JointureExterneDroite (R1 , R2 , R1 . C=R2 . X)

SELECT *
FROM R1 RIGHT OUTER JOIN R2
ON R1 . C=R2 . X
```

Exemple de jointure externe droite (SQL et Algèbre)

**Méthode** Trouver les enregistrements non joints

La jointure externe sert en particulier pour trouver les enregistrements d'une table qui ne sont pas référencés par une clé étrangère. Il suffit de sélectionner, après la jointure externe, tous les enregistrements pour lesquels la clé de la relation *référencante* est nulle, on obtient alors ceux de la relation référencée qui ne sont **pas** référencés.

R1	#A	B	C=>R2
	1	Alpha	10
	2	Bravo	10
	3	Charlie	20
	4	Delta	

R2	#X	Y
	10	Echo
	20	Fox
	30	Golf

R	A	B	C	X	Y
				30	Golf

```

Restriction(
    JointureExterneDroite (R1 , R2 , R1 . C=R2 . X ) ,
    R1 . A IS NULL)

SELECT *
FROM R1 RIGHT OUTER JOIN R2
ON R1 . C=R2 . X
WHERE R1 . A IS NULL
    
```

*Exemple de sélection d'enregistrements non référencés (SQL et Algèbre)*

**Complément**    **Non-support des jointures externes par SQLite**

SQLite ne supporte pas, actuellement, les jointures externes.

## IX. Exercice : Appliquer la notion

On se donne les tables suivantes d'un système de suivis d'intervention d'urgence :

```

1 CREATE TABLE intervention (
2 id INTEGER PRIMARY KEY,
3 categorie VARCHAR(200),
4 longitude FLOAT,
5 latitude FLOAT
6 );
7
8 CREATE TABLE vehicule(
9 immatriculation VARCHAR(14) PRIMARY KEY,
10 genre VARCHAR(50)
11 );
12
13 CREATE TABLE selection(
14 heure TIME,
15 intervention INTEGER REFERENCES Intervention(id),
16 vehicule VARCHAR(14) REFERENCES Vehicule(immatriculation),
17 PRIMARY KEY (heure, intervention, vehicule)
18 );
19
20 INSERT INTO intervention
21 VALUES (1042, 'Feux', 48.8662, 2.3078);
22
23 INSERT INTO intervention
24 VALUES (1183, 'Secours à Victimes', 48.8641, 2.3011);
    
```

```
25
26 INSERT INTO intervention
27 VALUES (1190, 'Feux', 48.8641, 2.3078);
28
29 INSERT INTO vehicule
30 VALUES ('48VD249103', 'FPT');
31
32 INSERT INTO vehicule
33 VALUES ('81VD239107', 'FPT');
34
35 INSERT INTO vehicule
36 VALUES ('18VD239117', 'VSAV');
37
38 INSERT INTO vehicule
39 VALUES ('82VD291899', 'VSAV');
40
41 INSERT INTO selection
42 VALUES ('13:34:01', 1183, '18VD239117');
43
44 INSERT INTO selection
45 VALUES ('13:44:11', 1183, '82VD291899');
46
47 INSERT INTO selection
48 VALUES ('07:22:19', 1190, '81VD239107');
49
50 INSERT INTO selection
51 VALUES ('07:23:11', 1190, '81VD239107');
```

## Question

On veut connaître les véhicules qui n'ont jamais été sélectionnés pour une intervention.

Écrire une requête qui réalise une jointure externe permettant d'obtenir les informations d'un tel véhicule.

## X. Essentiel

Les jointures sont un des outils les plus puissants du langage SQL. Si leur utilisation basique permet de combiner les données de deux tables selon un critère particulier, il existe des cas d'utilisations plus avancés.

Par exemple, pour combiner des informations issues de plus deux tables, on pourra chaîner les clauses `JOIN ON`. Plus il y a d'entités gérées dans une base de données, plus cette situation a des chances d'arriver.

Aussi, vous aurez parfois besoin de connaître les enregistrements d'une table qui ne sont référencés dans aucune autre table. Une jointure classique ne suffit pas et ignore ces enregistrements. Dans ce cas, on pourra utiliser une jointure externe grâce à la syntaxe `OUTER JOIN`.

## XI. Auto-évaluation

### A. Exercice final

#### Exercice : Quiz - Culture

Exercice

Une jointure peut être formalisée comme un produit cartésien sur lequel on pose une restriction.

- Vrai
- Faux

Exercice

L'ordre des tables dans les requêtes influence (à une permutation près des attributs) les résultats :

- Des jointures simples
- Des jointures multiples
- Des auto-jointures
- Des jointures externes

Exercice

On ne peut réaliser que des jointures entre des tables différentes.

- Vrai
- Faux

Exercice

Parmi les propositions suivantes, quels sont les mots-clés utilisés pour réaliser une jointure externe ?

- EXCEPT
- OUTER
- JOIN
- GROUP BY
- EXTERN

### **Exercice : Quiz - Méthode**

Exercice

Pour réaliser une auto-jointure, il faut nécessairement :

- Définir des alias sur la table.
- Définir des alias sur les attributs.
- Une restriction avec `WHERE`.
- Utiliser les alias des attributs pour la jointure.
- Utiliser les alias de table pour la jointure.

Exercice

Pour réaliser une jointure multi-tables, il faut nécessairement :

- Définir des alias sur la table.
- Définir des alias sur les attributs.
- Une restriction avec `WHERE`.
- Utiliser les alias des attributs pour la jointure.
- Utiliser les alias de table pour la jointure.

### **Exercice : Quiz - Code**

Exercice

Que réalise la requête suivante ?

```
1 SELECT a.nom, v.immatriculation, p.prenom, p.nom
2 FROM assurance a, vehicule v, proprietaire p
3 WHERE a.id = p.assurance
4 AND v.proprietaire = p.id;
```

- Une jointure simple
- Une jointure multi-tables
- Une auto-jointure
- Une jointure externe

Exercice

Que réalise la requête suivante ?

```
1 SELECT b1.etiquette
2 FROM boite b1, boite b2
3 WHERE b1.composite = b2.etiquette;
```

- Une jointure simple
- Une jointure multi-tables
- Une auto-jointure
- Une jointure externe

Exercice

Que réalise la requête suivante ?

```
1 SELECT b.nom, p.latitude, p.longitude
2 FROM bateau b JOIN port p
3 ON b.port = p.id;
```

- Une jointure simple
- Une jointure multi-tables
- Une auto-jointure
- Une jointure externe

Exercice

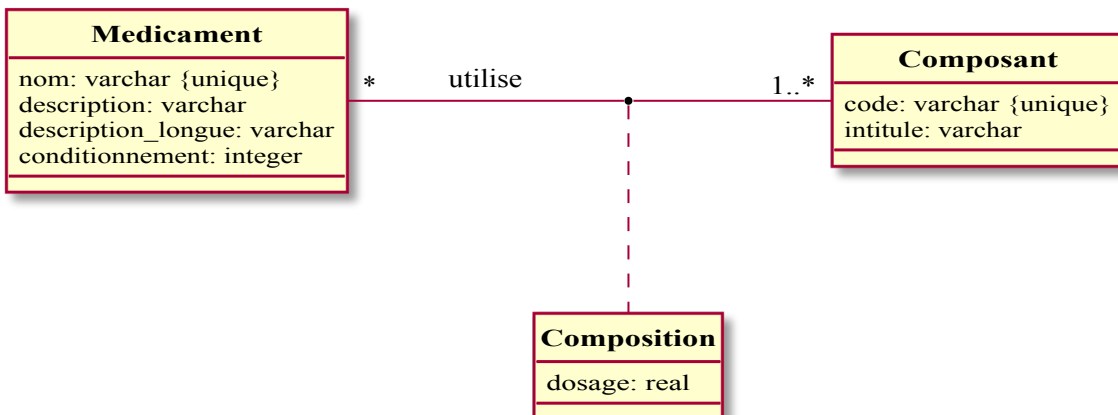
Que réalise la requête suivante ?

```
1 SELECT a.id, a.nom
2 FROM ventes v LEFT OUTER JOIN article a
3 ON v.article=a.id
4 WHERE v.id is NULL;
```

- Une jointure simple
- Une jointure multi-tables
- Une auto-jointure
- Une jointure externe

## B. Exercice : Défi

On se donne la modélisation suivante de la composition d'un médicament :



On implémente ce modèle et peuple les tables avec les requêtes suivantes :

```

1 CREATE TABLE medicament (
2   nom VARCHAR(50) PRIMARY KEY,
3   description VARCHAR(50),
4   description_longue VARCHAR(200),
5   conditionnement INTEGER
6 );
7
8 CREATE TABLE composant (
9   code VARCHAR(50) PRIMARY KEY,
10  intitule VARCHAR(50)
11 );
12
13 CREATE TABLE composition (
14   medicament VARCHAR(50) REFERENCES medicament(nom),
15   composant VARCHAR(50) REFERENCES composant(code),
16   dosage REAL,
17   PRIMARY KEY (medicament, composant)
18 );
19
20
21 INSERT INTO medicament VALUES ('Paracétamoule', 'Analgésique et fébrifugeur commun', 'Le
22   paracétamoule est utilisé pour traiter nombre de maux', 12);
23 INSERT INTO medicament VALUES ('Smectoub', 'Pensement digestif', 'Traite la diarrhée aïgue.',
24   5);
25 INSERT INTO medicament VALUES ('Ladvi', 'Pensement digestif', 'Soulage votre estomac.', 10);
26 INSERT INTO medicament VALUES ('Plakenille', 'Prétendu remède', 'Traitement de la malaria
27   causé par des morcure de moustiques mais non de pangolins.', 42);
28 INSERT INTO medicament VALUES ('Isparine', 'Prétendu remède', 'Traitement de maux de tête.',
29   12);
30
31 INSERT INTO composant VALUES ('J729FND01', 'Povidone');
32 INSERT INTO composant VALUES ('919R020N1', 'Amidon prégélatinisé');
33 INSERT INTO composant VALUES ('NJAI29R1C', 'Carboxyméthylamidon sodique');
34 INSERT INTO composant VALUES ('CQI2FN111', 'Talc');
35 INSERT INTO composant VALUES ('LUC11NSKY', 'N-diéthyllysergamide');
36 INSERT INTO composant VALUES ('PFOC30CE0', 'Stéarate de magnésium');
37 INSERT INTO composant VALUES ('N91NF01NN', 'Lactose monohydraté');
38 INSERT INTO composant VALUES ('J4M41SC3L4', 'Syanure');
39 INSERT INTO composant VALUES ('2NCKQ0213', 'Magnésium stéarate');
40 INSERT INTO composant VALUES ('WQOCZIIXA', 'Éthanol');
  
```

```

37 INSERT INTO composant VALUES ('Y0LOW1337', 'Hydroxychloroquine');
38
39 INSERT INTO composition VALUES ('Paracétamoule', 'J729FND01', 0.2);
40 INSERT INTO composition VALUES ('Paracétamoule', '919R020N1', 0.1);
41 INSERT INTO composition VALUES ('Paracétamoule', 'NJAI29R1C', 0.2);
42 INSERT INTO composition VALUES ('Paracétamoule', 'CQI2FN111', 0.5);
43 INSERT INTO composition VALUES ('Paracétamoule', 'PFOC30CE0', 1.2);
44 INSERT INTO composition VALUES ('Plakenille', 'N91NF01NN', 1.7);
45 INSERT INTO composition VALUES ('Plakenille', '2NCKQ0213', 1.9);
46 INSERT INTO composition VALUES ('Plakenille', 'WQOCZIIXA', 2.4);
47 INSERT INTO composition VALUES ('Plakenille', 'Y0LOW1337', 0.99);
48 INSERT INTO composition VALUES ('Smectoub', 'N91NF01NN', 1.9);
49 INSERT INTO composition VALUES ('Smectoub', 'J729FND01', 0.3);
50 INSERT INTO composition VALUES ('Smectoub', '919R020N1', 0.4);
51

```

### Question 1

On veut connaître les médicaments composés d'amidon pré-gélatinisé (Amidon pré-gélatinisé dans la base) et le dosage associé.

Écrire une requête de jointure avec JOIN et ON qui permet d'obtenir ces informations.

### Question 2

Écrire une requête de jointure avec WHERE qui permet d'obtenir ces mêmes informations.

### Question 3

Afin de tester des médecins sur leurs connaissances des compositions des médicaments, on souhaite générer la liste de tous les couples (nom de médicament, intitulé du composant) que l'on peut créer avec les données de la base.

Écrire une requête qui réalise le produit cartésien permettant d'obtenir cette liste.

### Question 4

On veut connaître la liste des paires de médicaments et composants pour lesquels le dosage est considéré comme important (plus de 1g). Cette liste doit contenir le nom du médicament, son conditionnement, l'intitulé du composant et le dosage. La table Composition contient l'information de dosage en grammes.

Écrire une requête qui réalise une jointure multiple permettant d'obtenir cette liste.

### Question 5

On veut connaître les paires de noms de médicaments ayant la même description.

Écrire une requête qui réalise une auto-jointure permettant d'obtenir cette liste.

### Indice :

Pour n'avoir que des couples uniques, on pourra ajouter à la restriction une condition supplémentaire de la forme suivante :

```
1 m1.nom != m2.nom;
```

### Question 6

On veut connaître les intitulés des composants qui ne sont contenus dans aucun médicament.

Écrire une requête qui réalise une jointure externe permettant d'obtenir cette liste.

## Solutions des exercices



### Exercice p. Solution n°1

```
1 SELECT l.prenom, l.nom, a.denomination
2 FROM locataire l JOIN assurance a
3 ON l.assurance = a.num_siret ;
```

On obtient :

1	prenom	nom	denomination	
2	-----	-----	-----	
3	Jean	Durand	Xitour	
4	Anne	Durand	Xitour	
5	Camille	Dupont	Faim	
6	Marie	Martin	Pro Gama	

### Exercice p. Solution n°2

```
1 SELECT a.pseudonyme, m.nom_de_code
2 FROM agentSecret a, mission m;
```

On obtient :

1	pseudonyme	nom_de_code	
2	-----	-----	
3	OSS 117	Silence Radio	
4	OSS 117	Oisillons	
5	SCEP 2421	Silence Radio	
6	SCEP 2421	Oisillons	
7	MOSS_1411	Silence Radio	
8	MOSS_1411	Oisillons	

### Exercice p. Solution n°3

```
1 SELECT i.id, i.categorie, v.immatriculation, v.genre
2 FROM intervention i, vehicule v, selection s
3 WHERE s.intervention = i.id
4 AND s.vehicule = v.immatriculation;
```

On obtient :

1	id	categorie	immatriculation	genre
2	----	-----	-----	-----
3	1183	Secours à Victimes	18VD239117	VSAV
4	1183	Secours à Victimes	82VD291899	VSAV
5	1190	Feux	81VD239107	FPT
6	1190	Feux	81VD239107	FPT

### Exercice p. Solution n°4

On utilise la requête suivante :

```
1 SELECT v.immatriculation, v.genre
2 FROM vehicule v LEFT OUTER JOIN selection s
3 ON s.vehicule = v.immatriculation
4 WHERE s.intervention IS NULL;
```

On obtient :

1	immatriculation	genre	
2	-----	-----	
3	48VD249103	FPT	

**Exercice p. 15 Solution n°5**

Exercice

Une jointure peut être formalisée comme un produit cartésien sur lequel on pose une restriction.

- Vrai
- Faux


Exercice

L'ordre des tables dans les requêtes influence (à une permutation près des attributs) les résultats :

- Des jointures simples
- Des jointures multiples
- Des auto-jointures
- Des jointures externes

Exercice

On ne peut réaliser que des jointures entre des tables différentes.

- Vrai
- Faux
-  On peut aussi réaliser des auto-jointures, c'est-à-dire des jointures sur une seule table.

Exercice

Parmi les propositions suivantes, quels sont les mots-clés utilisés pour réaliser une jointure externe ?

- EXCEPT
- OUTER
- JOIN
- GROUP BY
- EXTERN

**Exercice p. 16 Solution n°6**

Exercice


Pour réaliser une auto-jointure, il faut nécessairement :

- Définir des alias sur la table.
- Définir des alias sur les attributs.
- Une restriction avec WHERE.
- Utiliser les alias des attributs pour la jointure.

- Utiliser les alias de table pour la jointure.

Exercice

Pour réaliser une jointure multi-tables, il faut nécessairement :

- Définir des alias sur la table.
- Définir des alias sur les attributs.
- Une restriction avec `WHERE`.
- Utiliser les alias des attributs pour la jointure.
- Utiliser les alias de table pour la jointure.
-  Les éléments proposés facilitent l'écriture des requêtes mais ne sont pas nécessaires.

### Exercice p. 16 Solution n°7

Exercice

Que réalise la requête suivante ?

```
1 SELECT a.nom, v.immatriculation, p.prenom, p.nom
2 FROM assurance a, vehicule v, proprietaire p
3 WHERE a.id = p.assurance
4 AND v.proprietaire = p.id;
```

- Une jointure simple
- Une jointure multi-tables
- Une auto-jointure
- Une jointure externe

Exercice

Que réalise la requête suivante ?

```
1 SELECT b1.etiquette
2 FROM boite b1, boite b2
3 WHERE b1.composite = b2.etiquette;
```

- Une jointure simple
- Une jointure multi-tables
- Une auto-jointure
- Une jointure externe

Exercice

Que réalise la requête suivante ?

```
1 SELECT b.nom, p.latitude, p.longitude
2 FROM bateau b JOIN port p
3 ON b.port = p.id;
```

- Une jointure simple
- Une jointure multi-tables

- Une auto-jointure
- Une jointure externe

Exercice

Que réalise la requête suivante ?

```
1 SELECT a.id, a.nom
2 FROM ventes v LEFT OUTER JOIN article a
3 ON v.article=a.id
4 WHERE v.id is NULL;
```

- Une jointure simple
- Une jointure multi-tables
- Une auto-jointure
- Une jointure externe

**Exercice p. Solution n°8**

```
1 SELECT d.medicament, d.dosage
2 FROM composant c JOIN composition d
3 ON c.code = d.composant
4 WHERE c.intitule = 'Amidon prégélatinisé';
```

On obtient :

1	medicament	dosage
2	-----	-----
3	Paracétamoule	0.1
4	Smectoub	0.4

**Exercice p. Solution n°9**

```
1 SELECT d.medicament, d.dosage
2 FROM composant c, composition d
3 WHERE c.code = d.composant
4 AND c.intitule = 'Amidon prégélatinisé';
```

**Exercice p. Solution n°10**

```
1 SELECT m.nom, c.intitule
2 FROM medicament m, composant c;
```

On obtient :

1	nom	intitule
2	-----	-----
3	Paracétamoule	Povidone
4	Smectoub	Povidone
5	Ladvi	Povidone
6	Plakenille	Povidone
7	Isparine	Povidone
8	Paracétamoule	Amidon prégélatinisé
9	Smectoub	Amidon prégélatinisé
10	Ladvi	Amidon prégélatinisé

11	Plakenille	Amidon prégélatinisé
12	Isparine	Amidon prégélatinisé
13	Paracétamoule	Carboxyméthylamidon sodique
14	Smectoub	Carboxyméthylamidon sodique
15	Ladvi	Carboxyméthylamidon sodique
16	Plakenille	Carboxyméthylamidon sodique
17	Isparine	Carboxyméthylamidon sodique
18	Paracétamoule	Talc
19	Smectoub	Talc
20	Ladvi	Talc
21	Plakenille	Talc
22	Isparine	Talc
23	Paracétamoule	N-diéthyllysergamide
24	Smectoub	N-diéthyllysergamide
25	Ladvi	N-diéthyllysergamide
26	Plakenille	N-diéthyllysergamide
27	Isparine	N-diéthyllysergamide
28	Paracétamoule	Stéarate de magnésium
29	Smectoub	Stéarate de magnésium
30	Ladvi	Stéarate de magnésium
31	Plakenille	Stéarate de magnésium
32	Isparine	Stéarate de magnésium
33	Paracétamoule	Lactose monohydraté
34	Smectoub	Lactose monohydraté
35	Ladvi	Lactose monohydraté
36	Plakenille	Lactose monohydraté
37	Isparine	Lactose monohydraté
38	Paracétamoule	Syanure
39	Smectoub	Syanure
40	Ladvi	Syanure
41	Plakenille	Syanure
42	Isparine	Syanure
43	Paracétamoule	Magnésium stéarate
44	Smectoub	Magnésium stéarate
45	Ladvi	Magnésium stéarate
46	Plakenille	Magnésium stéarate
47	Isparine	Magnésium stéarate
48	Paracétamoule	Éthanol
49	Smectoub	Éthanol
50	Ladvi	Éthanol
51	Plakenille	Éthanol
52	Isparine	Éthanol
53	Paracétamoule	Hydroxychloroquine
54	Smectoub	Hydroxychloroquine
55	Ladvi	Hydroxychloroquine
56	Plakenille	Hydroxychloroquine
57	Isparine	Hydroxychloroquine

### Exercice p. Solution n°11

On utilise la requête suivante :

```

1 SELECT m.nom, m.conditionnement, c.intitule, d.dosage
2 FROM medicament m, composition d, composant c
3 WHERE m.nom = d.medicament
4 AND d.composant = c.code
5 AND d.dosage > 1;

```

On obtient :

1	nom	conditionnement	intitule	dosage
2	-----	-----	-----	-----
3	Paracétamoule	12	Stéarate de magnésium	1.2
4	Plakenille	42	Lactose monohydraté	1.7
5	Plakenille	42	Magnésium stéarate	1.9
6	Plakenille	42	Éthanol	2.4
7	Smectoub	5	Lactose monohydraté	1.9

### Exercice p. Solution n°12

On utilise la requête suivante :

```
1 SELECT m1.nom AS nom1, m2.nom AS nom2
2 FROM medicament m1
3 INNER JOIN medicament m2
4 ON m1.description = m2.description
5 WHERE m1.nom < m2.nom;
```

On obtient :

1	nom1	nom2
2	-----	-----
3	Ladvi	Smectoub
4	Isparine	Plakenille

### Exercice p. Solution n°13

On utilise la requête suivante :

```
1 SELECT c.intitule
2 FROM composition d RIGHT OUTER JOIN composant c
3 ON d.composant = c.code
4 WHERE d.medicament IS NULL;
```

On obtient :

1	intitule
2	-----
3	N-diéthyllysergamide
4	Syanure