Jointures avancées

Table des matières

Contexte

Durée : 2h

Environnement de travail : DB Fiddle

Pré-requis : Aucun

[cf. 9ICIQUXL]

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.

Jointure : rappel

[cf. M4BTWG5x]

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.

Jointure — ***Rappel***

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.

```
SELECT *
FROM R1
INNER JOIN R2 O <condition>
```

Exemple


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

```
CREATE TABLE pays (
code CHAR(2) PRIMARY KEY,
nom VARCHAR(255) UNIQUE NOT NULL
);

CREATE TABLE foret (
id INTEGER PRIMARY KEY,
nom VARCHAR(255) NOT NULL,
```

```
surface INTEGER NOT NULL
);

CREATE TABLE contient (
pays CHAR(2) REFERENCES pays(code),
foret INTEGER REFERENCES foret(id),
PRIMARY KEY (foret, pays)
);
```

```
INSERT INTO pays VALUES ('FR', 'France');
INSERT INTO pays VALUES ('ES', 'Espagne');
INSERT INTO foret VALUES (1, 'Broceliande', 12000);
INSERT INTO foret VALUES (2, 'Pyrénéenne', 23000);
INSERT INTO foret VALUES (3, 'Compiégnoise', 14000);
INSERT INTO contient VALUES ('FR',1);
INSERT INTO contient VALUES ('FR',2);
INSERT INTO contient VALUES ('FR',3);
INSERT INTO contient VALUES ('ES',2);
```

```
SELECT p.code, f.nom
FROM pays p JOIN contient c
ON p.code = c.pays
JOIN foret f
ON c.foret = f.id
```

```
 code |     nom
------+-------------
 FR   | Broceliande
 FR   | Pyrénéenne
 FR   | Compiégnoise
 ES   | Pyrénéenne
```

Auto-jointure *Complément*

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`.

Auto-jointure *Exemple*

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

[cf. nR8bJ1pF]

Exercice : Appliquer la notion

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

```sql
CREATE TABLE assurance (
num_siret INTEGER PRIMARY KEY,
denomination VARCHAR(50),
adresse_siege_social VARCHAR(200)
);

CREATE TABLE locataire (
id INTEGER PRIMARY KEY,
prenom VARCHAR(50),
nom VARCHAR(50),
assurance INTEGER REFERENCES assurance(num_siret));

INSERT INTO assurance
VALUES (839204929, 'Pro Gama', '13, rue des Peupliers 42920 Saint-Jean-le-Cèdre');

INSERT INTO assurance
VALUES (839201119, 'Faim', '41, avenue des Tulipes 11001 Maintes-sur-Champs');

INSERT INTO assurance
VALUES (839283114, 'Xitour', '26, chemins de la mine 61920 Sourcieux');

INSERT INTO locataire
VALUES(47289, 'Jean', 'Durand', 839283114);

INSERT INTO locataire
VALUES(47211, 'Anne', 'Durand', 839283114);

INSERT INTO locataire
VALUES(47111, 'Camille', 'Dupont', 839201119);

INSERT INTO locataire
VALUES(47291, 'Marie', 'Martin', 839204929);


```

Question

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

Produit cartésien

[cf. 5e0z92ps]

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 ensemble 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.

Produit cartésien — *Définition*

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

Syntaxe

```
SELECT *
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)

Explosion combinatoire du produit cartésien

Attention

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.

Usage du produit cartésien

Remarque

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.

Sémantique des syntaxes de jointure

Remarque

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.

[cf. gQaFxCti]

Exercice : Appliquer la notion

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

```sql
CREATE TABLE agentSecret(
pseudonyme  VARCHAR(20) PRIMARY KEY,
nom VARCHAR(20),
prenom VARCHAR(20)
);

CREATE TABLE mission(
nom_de_code VARCHAR(20) PRIMARY KEY,
lieu_suppose VARCHAR(200)
);

INSERT INTO agentSecret
VALUES ('OSS 117', 'Bonisseur de la Bath', 'Hubert');

INSERT INTO agentSecret
VALUES ('SCEP 2421', 'El Akmar Betouche', 'Larmina');

INSERT INTO agentSecret
VALUES ('MOSS_1411', 'Koulechov', 'Dolores');

INSERT INTO mission
VALUES ('Silence Radio', 'Rio de Janeiro');

INSERT INTO mission
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.

Jointures multi-tables

[cf. 4vWUrFcS]

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.

Jointure par la clause JOIN *Syntaxe*

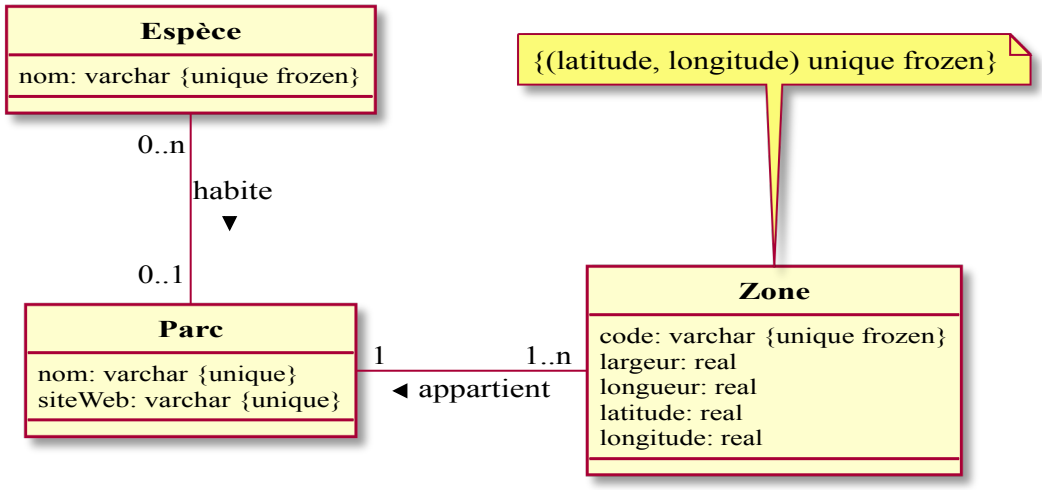
```sql
SELECT *
FROM a
JOIN b ON a.att_1 = b.att_1
JOIN c ON b.att_2 = c.att2;
```

Jointure par la clause WHERE *Syntaxe*

```sql
SELECT *
FROM a, b, c
WHERE a.att_1 = b.att_1
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.




```sql
CREATE TABLE parc(
  nom VARCHAR(50) PRIMARY KEY,
  site_web VARCHAR(50) UNIQUE NOT NULL
);

CREATE TABLE zone_parc(
  code VARCHAR(50) PRIMARY KEY,
  parc VARCHAR(50) REFERENCES parc(nom),
  largeur REAL,
  longueur REAL,
  latitude REAL,
  longitude REAL
);

CREATE TABLE espece(
  nom VARCHAR(50) PRIMARY KEY,
  parc VARCHAR(50) REFERENCES parc(nom)
);

INSERT INTO parc VALUES ('Parc Floral de Paris',
'https://www.parcfloraldeparis.com/fr');
INSERT INTO parc VALUES ('Jardin des Plantes de Paris',
'https://www.jardindesplantesdeparis.fr/fr');

INSERT INTO zone_parc VALUES ('NFI1', 'Jardin des Plantes de Paris', 20, 50, 48.50,
2.21);
INSERT INTO zone_parc VALUES ('AB35', 'Jardin des Plantes de Paris', 20, 50, 48.84,
2.35);

INSERT INTO zone_parc VALUES ('VEF1', 'Parc Floral de Paris', 100, 20, 48.834594,
2.442491);
INSERT INTO zone_parc VALUES ('FJ29', 'Parc Floral de Paris', 20, 80, 48.8010341,
2.502201);

INSERT INTO espece VALUES ('Ausmas', 'Parc Floral de Paris');
INSERT INTO espece VALUES ('Orchidée Cymbidium', 'Parc Floral de Paris');

INSERT INTO espece VALUES ('Orchidée Asconcenda', 'Jardin des Plantes de Paris');
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 :

```sql
SELECT p.nom, z.code
FROM espece e, parc p, zone_parc z
WHERE e.nom = 'Orchidée Phalaenopsis'
AND e.parc = p.nom
AND z.parc = p.nom;
```

```
| nom                         | code |
| --------------------------- | ---- |
| Jardin des Plantes de Paris | NFI1 |
| Jardin des Plantes de Paris | AB35 |
```

Ordre des tables *Remarque*

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

[cf. cHKIgq5g]

Exercice : Appliquer la notion

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

```sql
CREATE TABLE intervention (
id INTEGER PRIMARY KEY,
categorie VARCHAR(200),
longitude FLOAT,
latitude FLOAT
);

CREATE TABLE vehicule(
immatriculation VARCHAR(14) PRIMARY KEY,
genre VARCHAR(50)
);

CREATE TABLE selection(
heure TIME,
intervention INTEGER REFERENCES Intervention(id),
vehicule VARCHAR(14) REFERENCES Vehicule(immatriculation),
PRIMARY KEY (heure, intervention, vehicule)
);

INSERT INTO intervention
VALUES (1042, 'Feux', 48.8662, 2.3078);

INSERT INTO intervention
VALUES (1183, 'Secours à Victimes', 48.8641, 2.3011);

INSERT INTO intervention
VALUES (1190, 'Feux', 48.8641, 2.3078);

INSERT INTO vehicule
VALUES ('48VD249103', 'FPT');

INSERT INTO vehicule
VALUES ('81VD239107', 'FPT');

INSERT INTO vehicule
VALUES ('18VD239117', 'VSAV');

INSERT INTO vehicule
VALUES ('82VD291899', 'VSAV');

INSERT INTO selection
VALUES ('13:34:01', 1183, '18VD239117');

INSERT INTO selection
VALUES ('13:44:11', 1183, '82VD291899');

INSERT INTO selection
VALUES ('07:22:19', 1190, '81VD239107');

```

```sql
INSERT INTO selection
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.

Jointure externe

[cf. O3dkOReR]

Objectifs

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

Mise en situation

Imaginez que vous gériez 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.

Type de jointures — *Fondamental*

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`.

Jointure externe — *Définition*

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.

Jointure externe gauche — *Définition*

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

Jointure externe droite

Définition

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

Jointure externe, gauche ou droite

Syntaxe

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`.

Jointure externe gauche

Exemple

```
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 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)

Trouver les enregistrements non joints

Méthode

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 *référençante* 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)

Non-support des jointures externes par SQLite

Complément

SQLite ne supporte pas, actuellement, les jointures externes.

[cf. ZenbJPyW]

Exercice : Appliquer la notion

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

```sql
CREATE TABLE intervention (
id INTEGER PRIMARY KEY,
categorie VARCHAR(200),
longitude FLOAT,
latitude FLOAT
);

CREATE TABLE vehicule(
immatriculation VARCHAR(14) PRIMARY KEY,
genre VARCHAR(50)
);

CREATE TABLE selection(
heure TIME,
intervention INTEGER REFERENCES Intervention(id),
vehicule VARCHAR(14) REFERENCES Vehicule(immatriculation),
PRIMARY KEY (heure, intervention, vehicule)
);

INSERT INTO intervention
VALUES (1042, 'Feux', 48.8662, 2.3078);

INSERT INTO intervention
VALUES (1183, 'Secours à Victimes', 48.8641, 2.3011);

INSERT INTO intervention
VALUES (1190, 'Feux', 48.8641, 2.3078);

INSERT INTO vehicule
VALUES ('48VD249103', 'FPT');

INSERT INTO vehicule
VALUES ('81VD239107', 'FPT');

INSERT INTO vehicule
VALUES ('18VD239117', 'VSAV');

INSERT INTO vehicule
VALUES ('82VD291899', 'VSAV');

INSERT INTO selection
VALUES ('13:34:01', 1183, '18VD239117');

INSERT INTO selection
VALUES ('13:44:11', 1183, '82VD291899');

INSERT INTO selection
VALUES ('07:22:19', 1190, '81VD239107');

```

```sql
INSERT INTO selection
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.

Essentiel

[cf. umIzIQTQ]

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`.

Quiz

Exercice 1 : 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 6 : 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 9 : Quiz - Code

Exercice

Que réalise la requête suivante ?

```sql
SELECT a.nom, v.immatriculation, p.prenom, p.nom
FROM assurance a, vehicule v, proprietaire p
WHERE a.id = p.assurance
AND v.proprietaire = p.id;
```

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

Exercice

Que réalise la requête suivante ?

```sql
SELECT b1.etiquette
FROM boite b1, boite b2
WHERE b1.composite = b2.etiquette;
```

- ○ Une jointure simple
- ○ Une jointure multi-tables
- ○ Une auto-jointure

- ○ Une jointure externe

Exercice

Que réalise la requête suivante ?

```sql
SELECT b.nom, p.latitude, p.longitude
FROM bateau n JOIN port p
ON b.port = p.id;
```

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

Exercice

Que réalise la requête suivante ?

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

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

Index

Crédits des ressources

9ICIQUXL p. 3
http://creativecommons.org/licenses/by-sa/4.0/fr/, Quentin Duchemin (réalisation vidéo https://www.studi.frStudi)

M4BTWG5x p. 4
http://creativecommons.org/licenses/by-sa/4.0/fr/, Quentin Duchemin (réalisation vidéo https://www.studi.frStudi)

5e0z92ps p. 7
http://creativecommons.org/licenses/by-sa/4.0/fr/, Quentin Duchemin (réalisation vidéo https://www.studi.frStudi)

4vWUrFcS p. 10
http://creativecommons.org/licenses/by-sa/4.0/fr/, Quentin Duchemin (réalisation vidéo https://www.studi.frStudi)

p. 11
http://creativecommons.org/licenses/by-sa/3.0/fr/, Julien Jerphanion

O3dkOReR p. 15
http://creativecommons.org/licenses/by-sa/4.0/fr/, Quentin Duchemin (réalisation vidéo https://www.studi.frStudi)

umIzIQTQ p. 21
http://creativecommons.org/licenses/by-sa/4.0/fr/, Quentin Duchemin (réalisation vidéo https://www.studi.frStudi)