

# Chapitre 10 : Les jointures modernes et la gestion des accès aux objets

## INF3080 BASES DE DONNÉES (SGBD)

Guy Francoeur

Aucune reproduction sans autorisation

3 septembre 2019

**UQÀM** | **Département d'informatique**

- ▶ Les droits de lecture sont accordés aux étudiants inscrits au cours INF3080-030 A2019 uniquement;
- ▶ Aucun droit pédagogique ou reproduction n'est accordé sans autorisation;

# Table des matières

1. Au dernier cours
2. jointure moderne
3. gestion des accès

# Table des matières

1. Au dernier cours
2. jointure moderne
3. gestion des accès

- ▶ Questions, précisions, sur les curseurs et les boucles

# Table des matières

1. Au dernier cours
2. jointure moderne
  - définition
  - inner join
  - left join
  - right join
  - full join
  - jointure exemple
3. gestion des accès

## jointure - définition

Les jointures permettent de lier des tuples d'ensemble différents qui ont une clé (un lien) en commun. Mais ceci n'est pas toujours vrai. Les jointures peuvent aussi retourner des lignes (tuples, rows) qui ne partagent pas de valeurs communes.

- ▶ Il existe 4 types de jointures.
- ▶ INNER, LEFT, RIGHT, FULL.
- ▶ EQUIJOIN est un INNER JOIN qui utilise le signe égal.

Nous avons déjà vu une façon de faire des jointures. Comment ?

```
SQL > SELECT * FROM ...
```

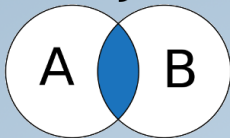
- ▶ INNER
- ▶ OUTER

L'instruction `OUTER` en opposition avec `INNER` retourne en plus des lignes qui sont communes toutes les autres lignes qui ne sont liées par une clé commune aux ensembles.



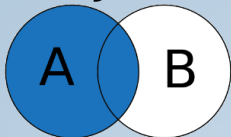
# inner join

## INNER JOIN



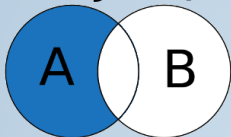
```
SELECT *  
FROM A  
INNER JOIN B ON A.key = B.key
```

## LEFT JOIN



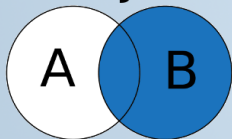
```
SELECT *  
FROM A  
LEFT JOIN B ON A.key = B.key
```

## LEFT JOIN (sans l'intersection de B)



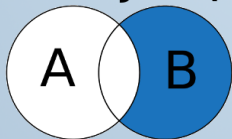
```
SELECT *  
FROM A  
LEFT JOIN B ON A.key = B.key  
WHERE B.key IS NULL
```

## RIGHT JOIN



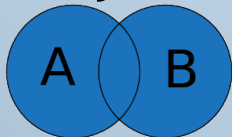
```
SELECT *  
FROM A  
RIGHT JOIN B ON A.key = B.key
```

## RIGHT JOIN (sans l'intersection de A)



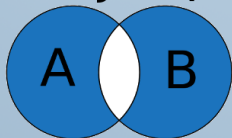
```
SELECT *  
FROM A  
RIGHT JOIN B ON A.key = B.key  
WHERE B.key IS NULL
```

## FULL JOIN



```
SELECT *  
FROM A  
FULL JOIN B ON A.key = B.key
```

## FULL JOIN (sans intersection)



```
SELECT *  
FROM A  
FULL JOIN B ON A.key = B.key  
WHERE A.key IS NULL  
OR B.key IS NULL
```

- ▶ LEFT JOIN vs LEFT OUTER JOIN
- ▶ RIGHT JOIN vs RIGHT OUTER JOIN

Maintenant que nous avons vu toute la théorie, des ensembles et la syntaxe moderne des jointures. Pouvez-vous dire quelle est la différence entre les deux instructions ci-haut. Prenez le temps d'y penser.

**Pause, voir page suivante**

## jointure - exemple question

```
SELECT A.*, B.qte  
FROM A LEFT OUTER JOIN B  
ON A.cProduit = B.cProduit
```

A : prix		B : Quantité	
patate	3\$	patate	45
avocat	4\$	avocat	63
kiwi	2\$	kiwi	19
oignon	1\$	oignon	20
melon	5\$	melon	66
orange	5\$	brocoli	27
tomate	6\$	courge	92

## jointure - exemple

```
SELECT A.*, B.qte  
FROM A LEFT OUTER JOIN B  
ON A.cProduit = B.cProduit
```

A : prix		B : Quantité		C : résultat		
patate	3\$	patate	45	patate	3\$	45
avocat	4\$	avocat	63	avocat	4\$	63
kiwi	2\$	kiwi	19	kiwi	2\$	19
oignon	1\$	oignon	20	oignon	1\$	20
melon	5\$	melon	66	melon	5\$	66
orange	5\$	brocoli	27	orange	5\$	NULL
tomate	6\$	courge	92	tomate	6\$	NULL

## jointure - exemple

```
SELECT B.*, A.prix  
FROM A RIGHT OUTER JOIN B  
ON A.cProduit = B.cProduit
```

A : prix		B : Quantité		C : résultat		
patate	3\$	patate	45	patate	45	3\$
avocat	4\$	avocat	63	avocat	63	4\$
kiwi	2\$	kiwi	19	kiwi	19	2\$
oignon	1\$	oignon	20	oignon	20	1\$
melon	5\$	melon	66	melon	66	5\$
orange	5\$	brocoli	27	brocoli	27	NULL
tomate	6\$	courge	92	courge	92	NULL



## jointure - exemple

```
SELECT A.*, B.*  
FROM A FULL OUTER JOIN B  
ON A.cProduit = B.cProduit
```

C : résultat

A : prix		B : Quantité					
patate	3\$	patate	45	patate	3\$	patate	45
avocat	4\$	avocat	63	avocat	4\$	avocat	63
kiwi	2\$	kiwi	19	kiwi	2\$	kiwi	19
oignon	1\$	oignon	20	oignon	1\$	oignon	20
melon	5\$	melon	66	melon	5\$	melon	66
orange	5\$	brocoli	27	orange	5\$	NULL	NULL
tomate	6\$	courge	92	tomate	6\$	NULL	NULL
				NULL	NULL	brocoli	27
				NULL	NULL	courge	92

# Table des matières

1. Au dernier cours
2. jointure moderne
3. gestion des accès
  - type de droit
  - grant
  - revoke
  - synonyme

Un SGBDR doit garantir que les informations gérées sont sécurisées. Il existe deux niveaux de sécurité.

- ▶ Premier niveau de sécurité : username/password;
- ▶ Deuxième niveau de sécurité : droit d'usage ou création;

Un SGBDR doit garantir que les informations gérées sont sécurisées. La sécurité des accès aux objets passe par les mots clés GRANT et REVOKE. Par la suite, nous devons spécifier le niveau de droit offert.

- ▶ SELECT, DELETE, UPDATE, INSERT;
- ▶ EXECUTE;
- ▶ ALTER, ... ;
- ▶ ALL PRIVILEGES;
- ▶ TO public;

Pour donner des droits, nous utilisons GRANT. Chacun des droits devra être donné sur un objet que vous n'êtes pas propriétaire, ou le créateur. La liste est assez longue nous ne les verrons pas tous, à ce point, il est important de comprendre que ça existe et la syntaxe générale à utiliser.

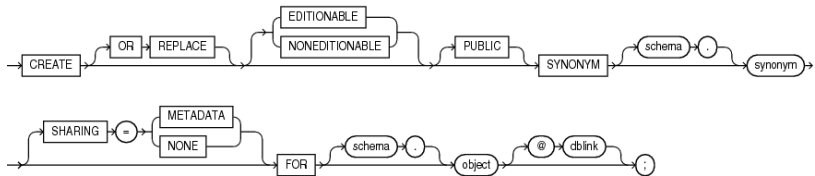
```
GRANT SELECT ON view_name TO codems;  
  
GRANT EXECUTE ON procedure_name TO public;  
  
GRANT DELETE, UPDATE ON table_name TO schema_name;
```

Nous utilisons REVOKE pour enlever, révoquer, des droits sur des objets pour des usagers, aussi nommer schéma.

```
REVOKE SELECT ON view_name TO codems;  
REVOKE EXECUTE ON p_procedure_name TO public;  
REVOKE DELETE, UPDATE ON table_name TO schema_name;
```

# synonyme

Les synonymes sont des objets qui font référence à des objets d'autres schémas.



```
-- public
CREATE or REPLACE public SYNONYM XSession for codems.XSession;
-- privé
CREATE SYNONYM Client FOR codems.Client;
```

- ▶ Questions ?
- ▶ Votre conclusion du cours d'aujourd'hui est ?
- ▶ Nous avons terminé.