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

INF3080 BASES DE DONNÉES (SGBD)

Guy Francoeur

3 septembre 2019

UQÀM | Département d'informatique

Matériel et droits

- 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

Table des matières

Retour sur les devoirs

- Questions, précisions, sur les curseurs et les boucles

Table des matières

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 ?

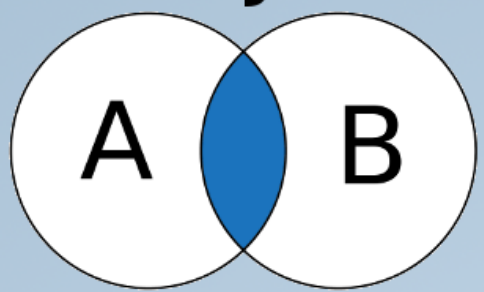
```
SQL > SELECT * FROM ...
```

jointure - inner vs outer

- 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.

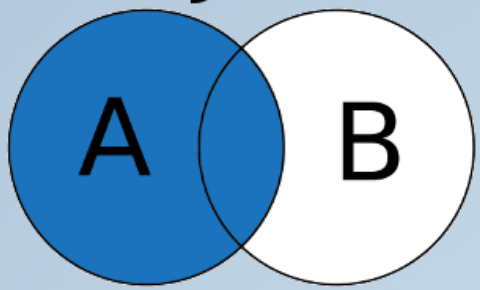
inner join




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

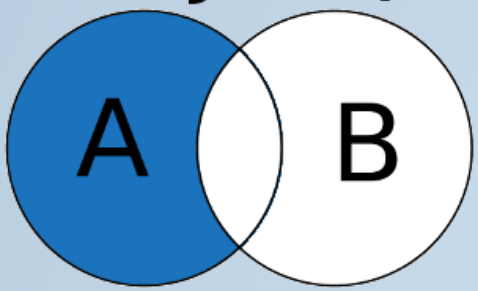
left join

LEFT JOIN




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

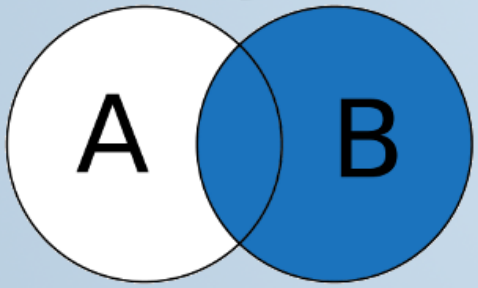
LEFT JOIN (sans l'intersection de B)




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

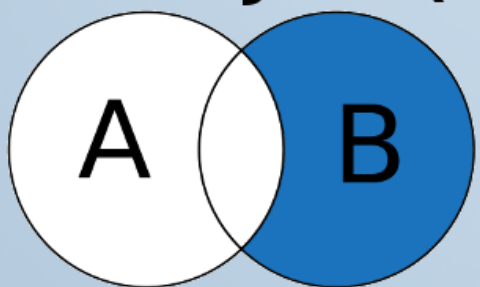
right join

RIGHT JOIN




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

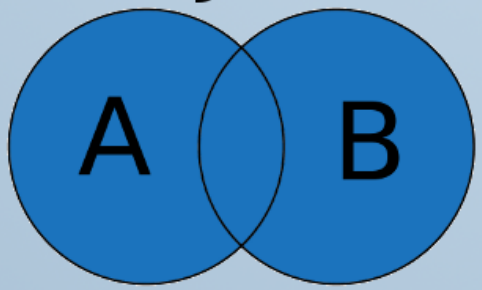
RIGHT JOIN (sans l'intersection de A)




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

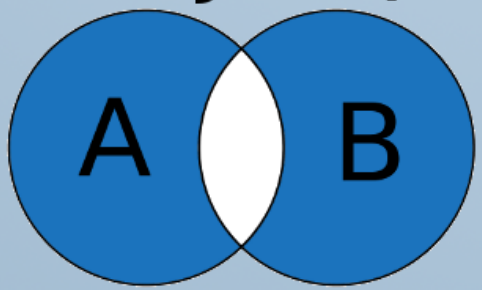
right join

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
```

jointure - différence

- 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	
patate	3$
avocat	4$
kiwi	2$
oignon	1$
melon	5$
orange	5$
tomate	6$

B : Quantité	
patate	45
avocat	63
kiwi	19
oignon	20
melon	66
brocoli	27
courge	92

jointure - exemple

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

A : prix	
patate	3$
avocat	4$
kiwi	2$
oignon	1$
melon	5$
orange	5$
tomate	6$

B : Quantité	
patate	45
avocat	63
kiwi	19
oignon	20
melon	66
brocoli	27
courge	92

C : résultat		
patate	3$	45
avocat	4$	63
kiwi	2$	19
oignon	1$	20
melon	5$	66
orange	5$	NULL
tomate	6$	NULL

jointure - exemple

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

A : prix	
patate	3$
avocat	4$
kiwi	2$
oignon	1$
melon	5$
orange	5$
tomate	6$

B : Quantité	
patate	45
avocat	63
kiwi	19
oignon	20
melon	66
brocoli	27
courge	92

C : résultat		
patate	45	3$
avocat	63	4$
kiwi	19	2$
oignon	20	1$
melon	66	5$
brocoli	27	NULL
courge	92	NULL

jointure - exemple

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

A : prix

patate	3$
avocat	4$
kiwi	2$
oignon	1$
melon	5$
orange	5$
tomate	6$

B : Quantité

patate	45
avocat	63
kiwi	19
oignon	20
melon	66
brocoli	27
courge	92

C : résultat

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

Table des matières

sécurité - définition

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;

type de droit

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

- SELECT, DELETE, UPDATE, INSERT;
- EXECUTE;
- ALTER, $\cdots$;
- ALL PRIVILEGES;
- TO public;

grant

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;
```

revoke

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

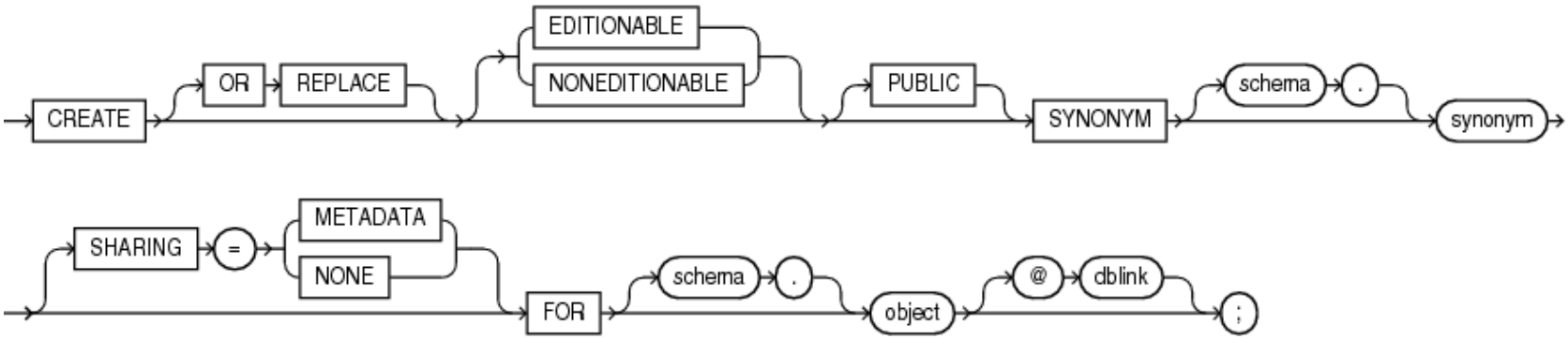
```
REVOKE SELECT ON view_name FROM codems;

REVOKE EXECUTE ON p_procedure_name FROM public;

REVOKE DELETE, UPDATE ON table_name FROM schema_name;
```

synonyme

Les synonymes sont des objets qui font référence à des objets qui sont créés dans d'autres schémas. L'objet référé par le synonyme n'est pas nécessairement un objet de votre création.




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

Résumé

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