

Base de données

Séance 7

Requêtes avec SELECT (suite)

Plan

- Jointure externe
 - à gauche
 - à droite
 - complète
 - Requêtes complexes
-

Jointure à gauche

- Soit une table décrivant des commerciaux, et une autre décrivant des affaires.
- On veut établir le comptage du chiffre d'affaire de tous les commerciaux, **y compris ceux qui n'ont rien fait.**

Commercial	
id	nom
1	John
2	Henri

Affaire		
id_affaire	ca	id_commercial
1	100	2
2	350	1
3	50	1
4	200	2

3

Chuck

L'échec de la jointure classique

```
SELECT c.id, SUM(ca)
FROM commercial c, affaire a
WHERE id_commercial = c.id
GROUP BY c.id;
```

- ❑ Ne donne pas le résultat escompté
- ❑ Il manque les commerciaux feignants

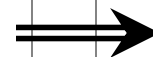
Commercial	
id	nom
1	John
2	Henri

Affaire		
id_affaire	ca	id_commercial
1	100	2
2	350	1
3	50	1
4	200	2

3



R = Commercial		$\bowtie_{id = id_commercial}$ Affaire		
id	nom	id_affaire	ca	id_commercial
1	John	2	350	1
1	John	3	50	1
2	Henri	1	100	2
2	Henri	4	200	2



$F_{id, SUM(ca)}(R)$	
id	SUM(ca)
1	400
2	300

Solution : la jointure gauche

```
SELECT c.id, SUM(ca)
FROM commercial c LEFT JOIN affaire a
ON id_commercial = c.id
GROUP BY c.id;
```

- Ajoute à la jointure interne les commerciaux qui n'ont pas d'affaire

Commercial	
id	nom
1	John
2	Henri



Affaire		
id_affaire	ca	id_commercial
1	100	2
2	350	1
3	50	1
4	200	2

3



Chuck

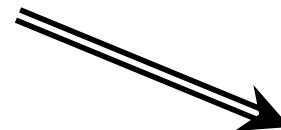
R = Commercial



id = id_commercial

Affaire

id	nom	id_affaire	ca	id_commercial
1	John	2	350	1
1	John	3	50	1
2	Henri	1	100	2
2	Henri	4	200	2
3	Chuck	NULL	NULL	NULL



id _{F_{id,SUM(ca)}(R)}	
id	SUM(ca)
1	400
2	300
3	0

La jointure externe : syntaxe SQL

```
SELECT ... FROM  
    <table gauche>  
        LEFT | RIGHT | FULL [OUTER] JOIN  
    <table droite>  
ON  
    <condition de jointure>  
WHERE ... ;
```

Les jointures externes : LEFT

tab1		tab2	
col11	col12	col21	col22
a	x	a	1
b	y	b	2
c	z	d	3

```
SELECT *  
FROM tab1 LEFT OUTER JOIN tab2  
ON tab1.col11 = tab2.col21;
```



a	x	a	1
b	y	b	2
c	z	NULL	NULL

Les jointures externes : RIGHT

tab1		tab2	
col11	col12	col21	col22
a	x	a	1
b	y	b	2
c	z	d	3

```
SELECT *  
FROM tab1 RIGHT OUTER JOIN tab2  
ON tab1.col11 = tab2.col21;
```



a	x	a	1
b	y	b	2
NULL	NULL	d	3

Les jointures externes : FULL

tab1		tab2	
col11	col12	col21	col22
a	x	a	1
b	y	b	2
c	z	d	3

```
SELECT *  
FROM tab1  
FULL OUTER JOIN tab2  
ON tab1.col11 = tab2.col21;
```



a	x	a	1
b	y	b	2
c	z	NULL	NULL
NULL	NULL	d	3

Attention, MySQL ne supporte pas FULL OUTER JOIN (LEFT UNION RIGHT)

Les jointures externes : exemple

Comptez pour chaque producteur le nombre de vins qu'il a récolté, y compris ceux qui n'ont rien produit :

```
SELECT p.id, COUNT(r.id_vin)
FROM producteur p LEFT OUTER JOIN recolte r
ON r.id_producteur = p.id
GROUP BY p.id ;
```

Les jointures externes : clauses WHERE

```
SELECT * FROM tab1  
LEFT OUTER JOIN tab2  
ON tab1.col11 = tab2.col21  
AND tab1.col11 <> 'a';
```

N'est pas identique à

```
SELECT * FROM tab1  
LEFT OUTER JOIN tab2  
ON tab1.col11 = tab2.col21  
WHERE tab1.col11 <> 'a';
```

Les jointures externes : clauses WHERE

tab1		tab2	
col11	col12	col21	col22
a	x	a	1
b	y	b	2
c	z	d	3

a	x	NULL	NULL
b	y	b	2
c	z	NULL	NULL

```
SELECT * FROM tab1
LEFT OUTER JOIN tab2
ON col11 = col21
AND col11 <> 'a';
```

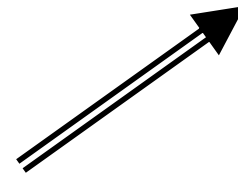
tab1 ⋈_{col11=col21 and col11≠'a'} tab2

Les jointures externes : clauses WHERE

tab1		tab2	
col11	col12	col21	col22
a	x	a	1
b	y	b	2
c	z	d	3

b	y	b	2
c	z	NULL	NULL

```
SELECT * FROM tab1
LEFT OUTER JOIN tab2
ON tab1.col11 = tab2.col21
WHERE tab1.col11 <> 'a' ;
```



$\sigma_{\text{col11} \neq \text{'a'}}(\text{tab1} \bowtie_{\text{col11}=\text{col21}} \text{tab2})$

Requêtes complexes

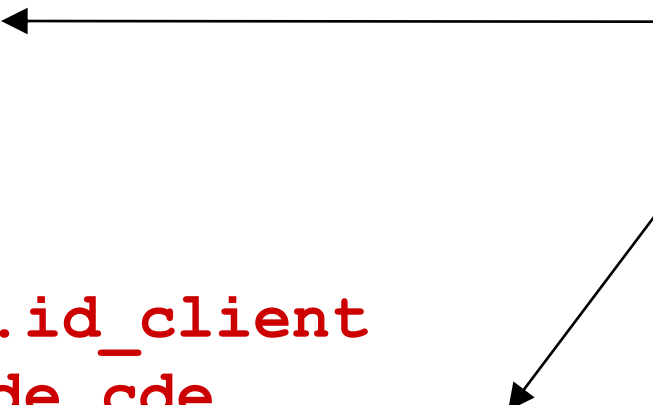
- Décomposer le problème en sous-problèmes simples
- Relier les sous-résultats par :
 - opérateur ensembliste
 - jointure
 - sous-requête
- Projeter les informations utiles

Sous-requêtes dépendantes

- Une sous-requête dépendante utilise une donnée de la requête principale.

WHERE EXISTS et WHERE NOT EXISTS

```
SELECT
    c.id_client
FROM
    client c
WHERE EXISTS
    (SELECT cde.id_client
     FROM commande cde
     WHERE cde.id_client = c.id_client) ;
```



Dépendance

- Cette requête utilise une **sous-requête dépendante**.
- Le prédicat est vrai quand le résultat de la sous-requête est non vide (contient au moins une ligne).

Individu

num_ind	nom	prenom
01	Kidman	Nicole
02	Bettany	Paul
03	Watson	Emily
04	Skarsgard	Stellan
05	Travolta	John
06	L. Jackson	Samuel
07	Willis	Bruce
08	Irons	Jeremy
09	Spader	James
10	Hunter	Holly
11	Arquette	Rosanna
12		John
13	von Trier	Lars
14	Tarantino	Quentin
15	Cronenberg	David
16	Mazursky	Paul
17	Jones	Grace
18	Glen	John

Film

num_film	num_ind	titre	genre	année
05	13	Dogville	Drame	2002
04	13	Breaking the waves	Drame	1996
03	14	Pulp Fiction	Policier	1994
02	15	Faux-Semblants	Epouvante	1988
01	15	Crash	Drame	1996
06	12	Alamo	Western	1960
07	18	Dangereusement vôtre	Espionnage	1985

Jouer

num_ind	num_film	role
01	05	Grace
02	05	Tom Edison
03	04	Bess
04	04	Jan
05	03	Vincent Vega
06	03	Jules Winnfield
07	03	Butch Coolidge
08	02	Beverly & Elliot Mantle
09	01	James Ballard
10	01	Helen Remington
11	01	Gabrielle
04	05	Chuck
16	07	May Day

Division : AR vers SQL

- ❑ Quels sont les acteurs qui ont joué dans tous les films de Lars von Trier ?
- ❑ En algèbre relationnelle :
Individu(num ind, nom, prenom)
Jouer(#num ind, #num film, role)
Film(num film, #num_ind, titre, genre, annee)

Idée :

Résultat = Jouer \div (films de Lars von Trier)

□ Les films de Lars von Trier :

$R1 = \sigma_{\text{nom}='von\ Trier' \text{ and } \text{prenom}='Lars'}(\text{Film} \bowtie \text{Individu})$

num_film	num_ind	titre	genre	année	nom	prenom
05	13	Dogville	Drame	2002	von Trier	Lars
04	13	Breaking ...	Drame	1996	von Trier	Lars


 $R2 = \Pi_{\text{num_film}}(R1)$

$\Pi_{\text{num_ind}, \text{num_film}}(\text{Jouer})$

num_ind num_film

01	05
02	05
03	04
04	04
05	03
06	03
07	03
08	02
09	01
10	01
11	01
04	05
16	07

\div

R2
num_film

05
04

num_ind

01
02
04

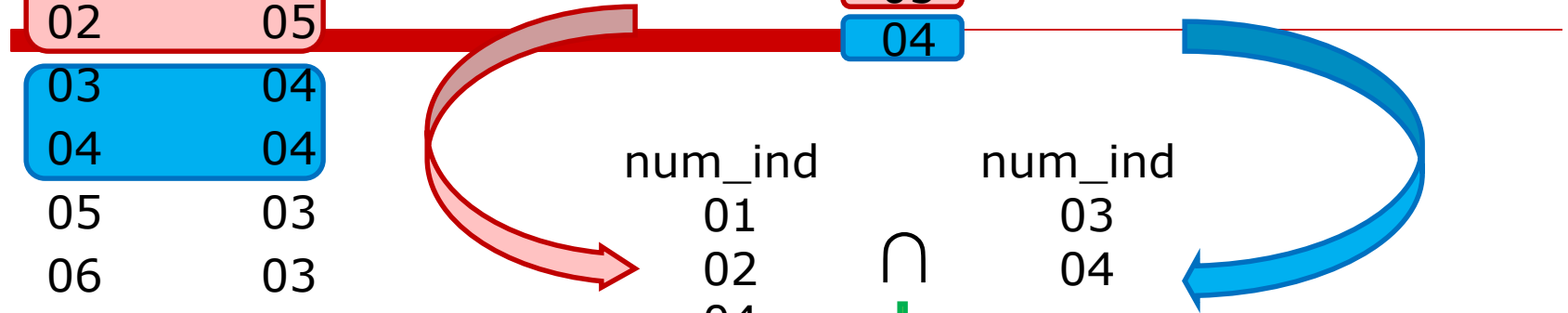
\cap



R3
num_ind
04

num_ind

03
04



$\Pi_{\text{num_ind, num_film}}(\text{Jouer})$

num_ind	num_film
---------	----------

01	05
----	----

02	05
----	----

03	04
----	----

04	04
----	----

05	03
----	----

06	03
----	----

07	03
----	----

08	02
----	----

09	01
----	----

10	01
----	----

11	01
----	----

04	05
----	----

16	07
----	----

÷

R2
num_film

05

04

num_ind

01

02

04

num_ind

03

04

∩

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

↓

R3
num_ind
04

⋈

Individu

$\Pi_{\text{prenom, nom}}$

R4

prenom

nom

Stellan

Skarsgard

Reformulation en SQL

- Quels sont les acteurs qui vérifient :
le nombre de films réalisés par Lars
von Trier dans lesquels l'acteur a joué
est égal au nombre de films réalisés
par Lars von Trier.
-

Traduction SQL

```
SELECT i.nom, i.prenom
FROM  individu i, joueur j, film f
WHERE i.num_ind = j.num_ind
AND   j.num_film = f.num_film
AND   f.num_film IN ( SELECT num_film
                      FROM  film f, individu i
                      WHERE f.num_ind = i.num_ind
                      AND   i.nom = 'von Trier'
                      AND   i.prenom = 'Lars')
GROUP BY i.num_ind,i.nom, i.prenom
HAVING COUNT (DISTINCT f.num_film) = (SELECT COUNT(*)
                                       FROM film NATURAL JOIN individu
                                       WHERE nom = 'von Trier' AND prenom = 'Lars');
```
