

Base de données

Séance 8

Algèbre relationnelle et SQL :

- Jointures externes
- Requêtes complexes

La jointure externe

- ❖ La jointure externe permet de récupérer les lignes des tables correspondant au critère de jointure, mais **aussi** celles pour lesquelles il n'existe pas de correspondances.
- ❖ Plusieurs types de jointure externe :
 - ❖ Jointure (externe) à gauche
 - ❖ Jointure (externe) à droite
 - ❖ Jointure (externe) complète

Jointure à gauche : introduction

- 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.

L'échec de la jointure classique

SELECT

`id_commercial, SUM(a.ca)`

FROM

`commercial AS c, affaire AS a`

WHERE

`id_commercial = c.id`

GROUP BY `id_commercial;`

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

Solution : la jointure gauche

```
SELECT
    id_commercial, SUM(a.ca)
FROM
    commercial AS c
LEFT JOIN
    affaire AS 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
3	Chuck

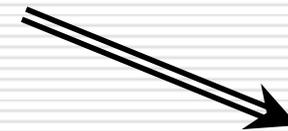


affaire		
id	ca	id_commercial
1	100	2
2	350	1
3	50	1
4	200	2



$R = \rho_c(\text{commercial}) \bowtie_{\text{commercial.id} = \text{id_commercial}} \text{affaire}$

c.id	nom	id	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



$F_{c.id, \text{SUM}(ca)}(R)$	
c.id	SUM(ca)
1	400
2	300
3	0

La jointure externe : LEFT, RIGHT et FULL

□ **RIGHT OUTER JOIN** :

- la table à droite de l'expression clef "RIGHT OUTER" renvoie des lignes sans correspondance avec la table à gauche.

□ **LEFT OUTER JOIN** :

- la table à gauche de l'expression clef "LEFT OUTER" renvoie des lignes sans correspondance avec la table à droite.

□ **FULL OUTER JOIN** :

- les deux tables renvoient des lignes sans correspondance entre elles.

La jointure externe : syntaxe SQL

SELECT ... FROM

<table gauche>

LEFT | RIGHT | FULL [OUTER] JOIN

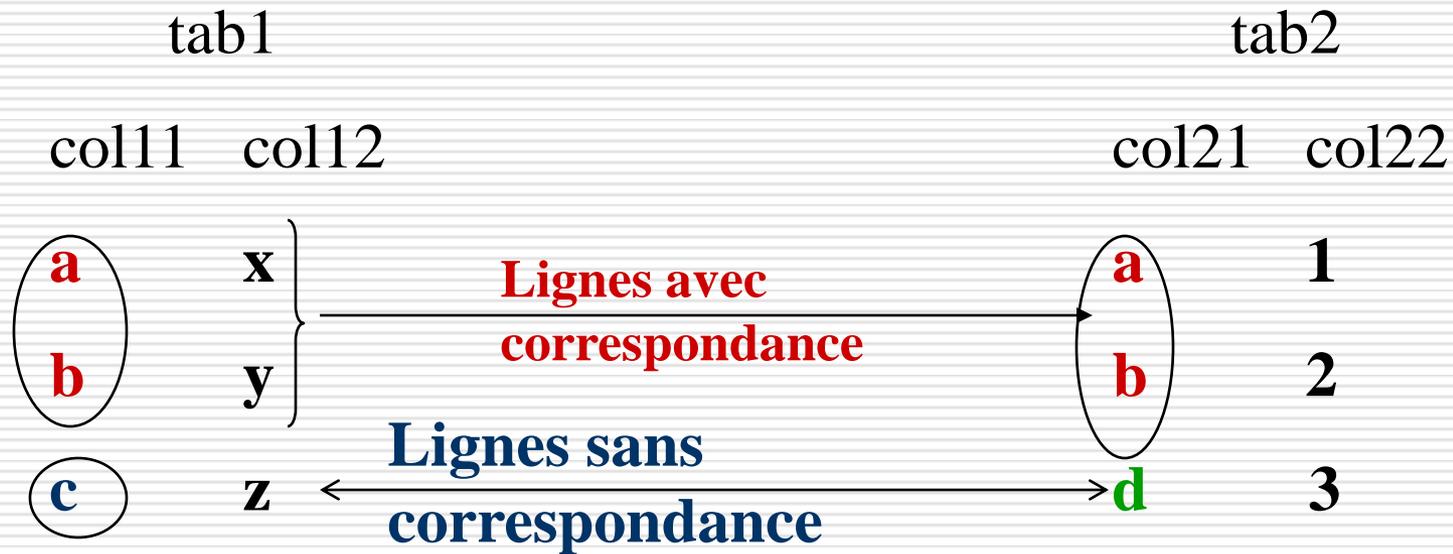
<table droite>

ON

<condition de jointure>

WHERE ... ;

Les jointures externes



Les jointures externes : LEFT

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

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

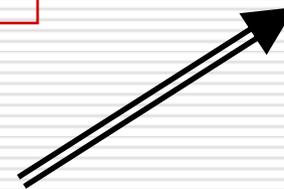
```
SELECT *  
FROM tab1  
LEFT OUTER JOIN tab2  
ON tab1.col11 = tab2.col21;
```

tab1 $\bowtie_{\text{col11}=\text{col21}}$ tab2

Les jointures externes : LEFT 2

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

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

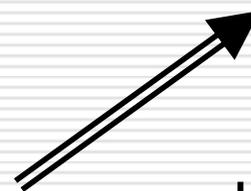


```
SELECT *  
FROM tab1,tab2  
WHERE tab1.col1 = tab2.col1 (+) ;
```

Les jointures externes : RIGHT

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

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



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

tab1 $\bowtie_{\text{col11}=\text{col21}}$ tab2

Les jointures externes : RIGHT 2

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

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

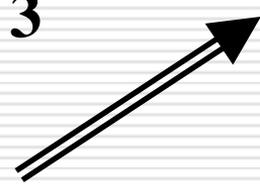


```
SELECT *  
FROM tab1,tab2  
WHERE tab1.col11(+) = tab2.col121;
```

Les jointures externes : FULL

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

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



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

tab1 $\bowtie_{\text{col11}=\text{col21}}$ tab2

Les jointures externes : exemple 1

```
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 ;
```

$$\rho_p(\text{producteur}) \bowtie_{p.\text{id}=\text{id_producteur}} \text{recolte}$$

Les jointures externes : exemple 1

```
SELECT
    p.id, COUNT (r.id_vin)
FROM
    producteur p, recolte r
WHERE
    p.id = r.id_producteur (+)
GROUP BY
    p.id ;
```

Les jointures externes : exemple 2

Calculer le nombre de films joués par chaque individu (y compris ceux qui ne sont pas acteurs)

```

SELECT i.nom, i.prenom, COUNT(j.num_film)
FROM individu i LEFT OUTER JOIN joueur j
ON i.num_ind = j.num_ind
GROUP BY i.num_ind, i.nom, i.prenom;

```

```

SELECT i.nom, i.prenom, COUNT(j.num_film)
FROM individu i, joueur j
WHERE i.num_ind = j.num_ind (+)
GROUP BY i.num_ind, i.nom, i.prenom;

```

$$i.\text{num_ind}, i.\text{nom}, i.\text{prenom} \overset{F}{\rho} (i.\text{nom}, i.\text{prenom}, \text{COUNT}(j.\text{num_film})) ($$

$$\rho_i(\text{individu}) \bowtie_{i.\text{num_ind}=j.\text{num_ind}} \rho_j(\text{joueur}))$$

Les jointures externes : clauses WHERE

```
SELECT * FROM tab1  
LEFT OUTER JOIN tab2  
ON tab1.col11 = tab2.col121  
AND tab1.col11 <> 'a';
```

N'est pas identique à

```
SELECT * FROM tab1  
LEFT OUTER JOIN tab2  
ON tab1.col11 = tab2.col121  
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';
```

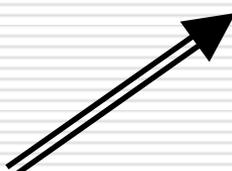
tab1 ⋈_{col11=col21 and col11≠'a'} tab2

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' ;
```



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

Les jointures croisées

```
SELECT  
    colonnes  
FROM  
    table1 t1  
CROSS JOIN  
    table2 t2  
[WHERE prédicat] ...
```

Jointures : résumé

- ❑ Jointure interne = jointure avec pivot
 - ❑ Jointure naturelle
 - ❑ Jointure externe : gauche, droit, complète
 - ❑ Jointure croisée : produit cartésien
-

Requêtes complexes

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

Décomposition 1

- Donner les **degrés** des vins de **cru Morgon** et de **millésime 1978**

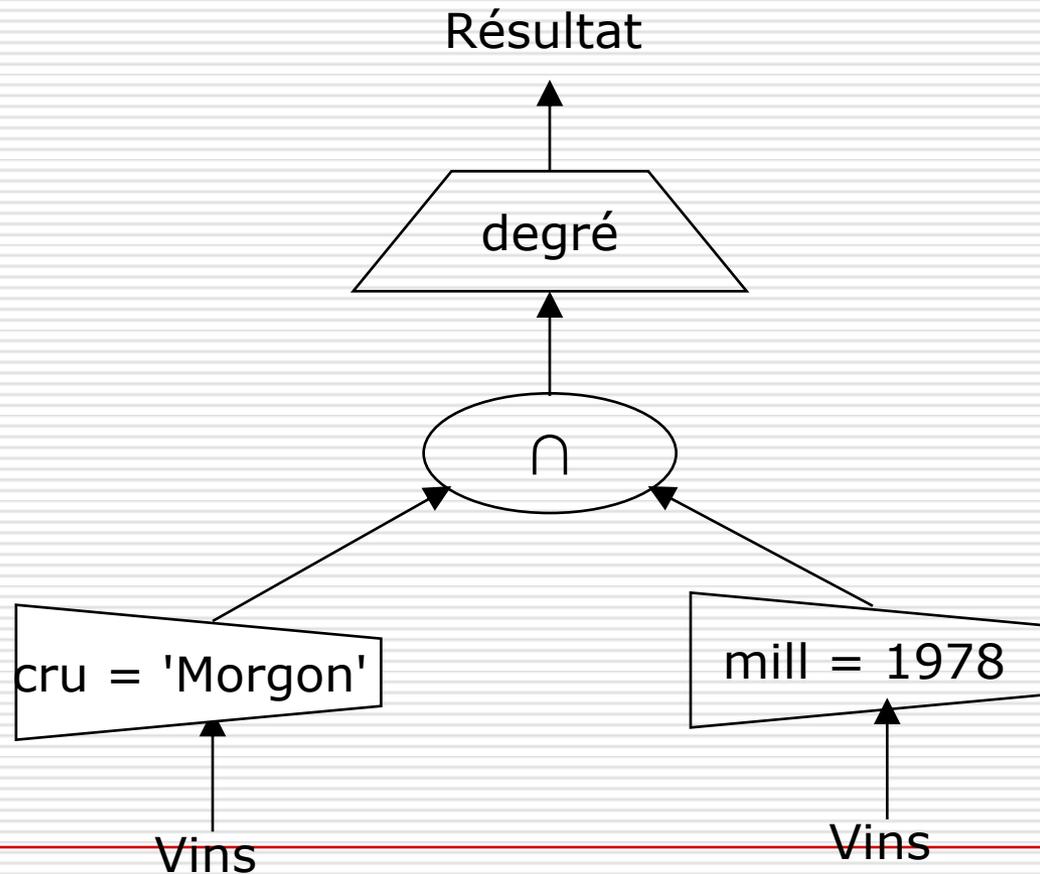
$$R1 = \sigma_{\text{cru} = \text{'Morgon'}}(\text{VINS})$$

$$R2 = \sigma_{\text{mill} = 1978}(\text{VINS})$$

$$R3 = R1 \cap R2$$

$$\text{Résultat} = \Pi_{\text{degré}}(R3)$$

Arbre algébrique : exemple 1



Décomposition 2

- Donner les noms et prénoms des buveurs habitant à Paris ayant bu du Chablis depuis le 1^{er} janvier 1992

$R1 = \sigma_{\text{adresse} = \text{'Paris'}}(\text{BUVEURS})$

$R2 = \sigma_{\text{cru} = \text{'Chablis'}}(\text{VINS})$

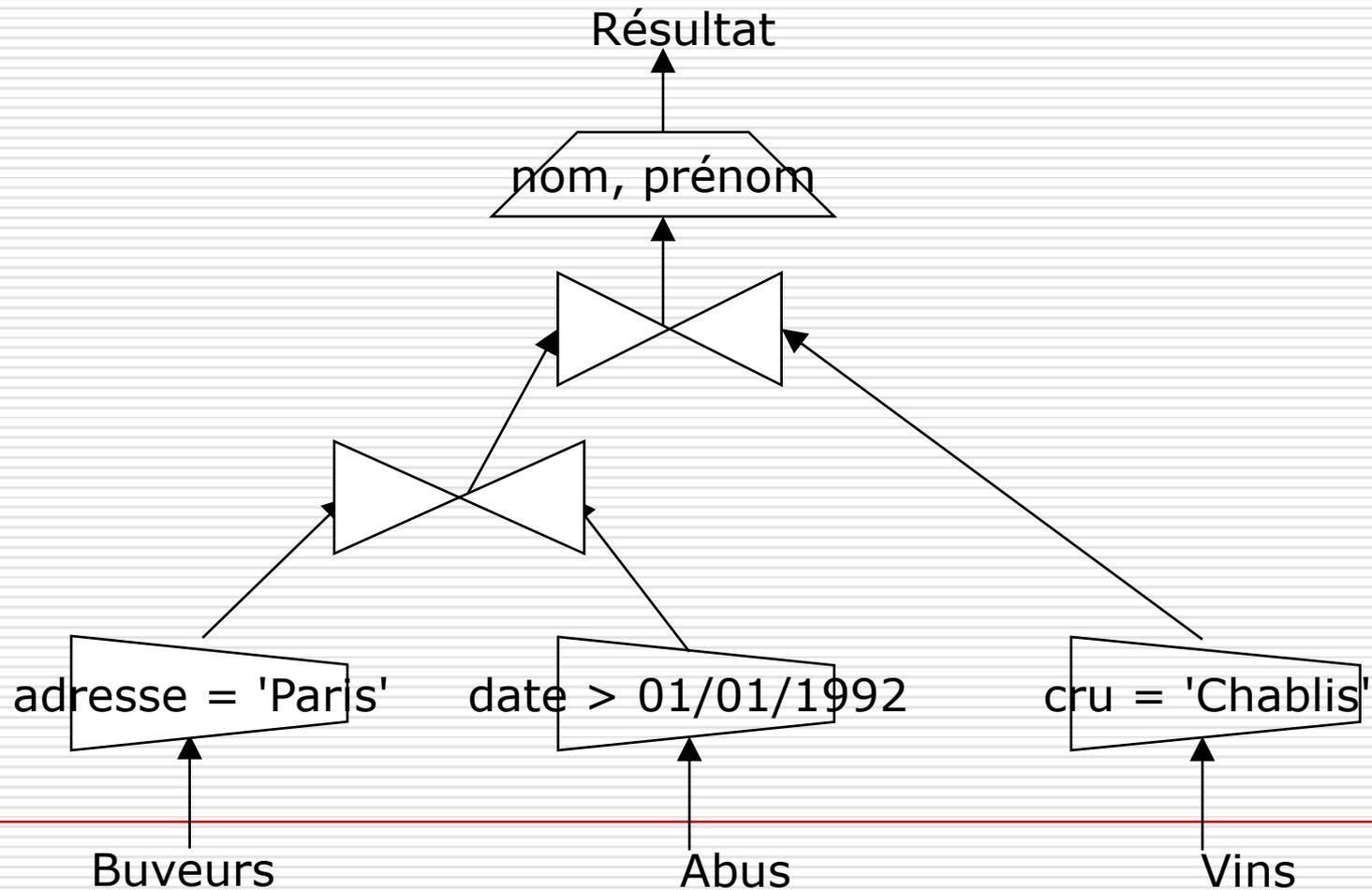
$R3 = \sigma_{\text{date} \geq \text{01/01/1992}}(\text{ABUS})$

$R4 = \text{JOIN}(R1, R3)$

$R5 = \text{JOIN}(R2, R4)$

Résultat = $\Pi_{\text{nom, prénom}}(R5)$

Arbre algébrique : exemple 2

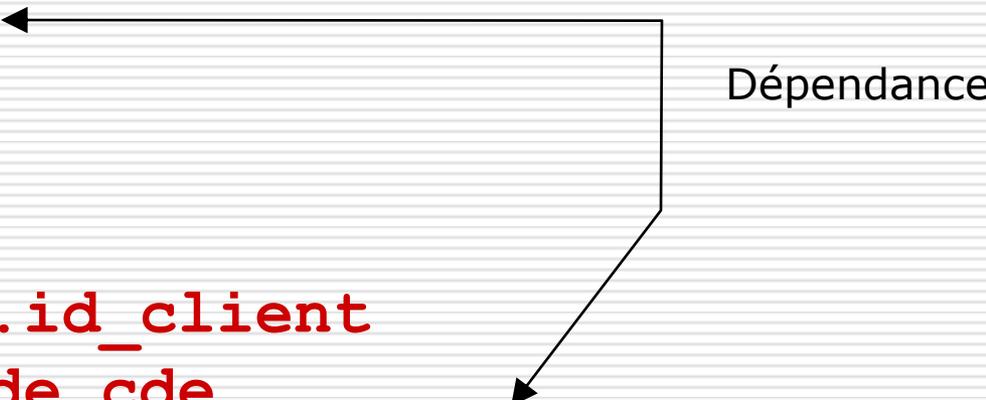


Sous-requêtes dépendantes

- 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).

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 :

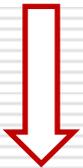
Résultat = Jouer \div (films de Lars von Trier)

Division : AR vers SQL

□ 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

÷



R2

num_film
05
04

num_ind

01
02
04

∩

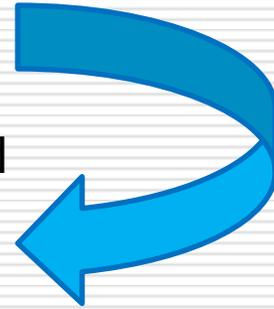


R3

num_ind
04

num_ind

03
04



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

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

÷

R2
num_film
05
04

num_ind

01
02
04



num_ind

03
04

R3
num_ind
04



Individu

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

R4

prenom nom
Stellan Skarsgard

Division en SQL avec EXISTS

- ❑ Quels sont les acteurs qui ont joué dans tous les films de Lars von Trier ?

 - ❑ Reformulation :
Quels sont les acteurs qui vérifient : il est faux qu'il existe un film de Lars von Trier dans lequel l'acteur n'a pas joué.
-

Traduction SQL

- SELECT DISTINCT nom, prenom FROM individu **acteur tous lars**
WHERE NOT EXISTS (

```
SELECT * FROM film film_lars JOIN individu i  
ON film_lars.num_ind = i.num_ind  
AND nom = 'von Trier' AND prenom = 'Lars'  
WHERE NOT EXISTS (
```

```
SELECT * FROM individu i JOIN jouer j  
ON i.num_ind = j.num_ind  
WHERE i.num_ind = acteur tous lars.num_ind  
AND num_film = film_lars.num_film));
```

Autre formulation 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');
```
