# Base de données

Séance 10

## Requêtes avec SELECT (suite)

# Plan

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

# Agrégats

- Partitionnement horizontal d'une table selon les valeurs d'un groupe d'attributs appelés facteurs de groupage
- Suivi d'un regroupement par une fonction de calcul en colonne (SUM, MIN, MAX, AVG, COUNT, ...)
- La table résultat contient une ligne par partition avec :
  - des facteurs de groupage (valeur unique par partition)
  - le résultat du calcul

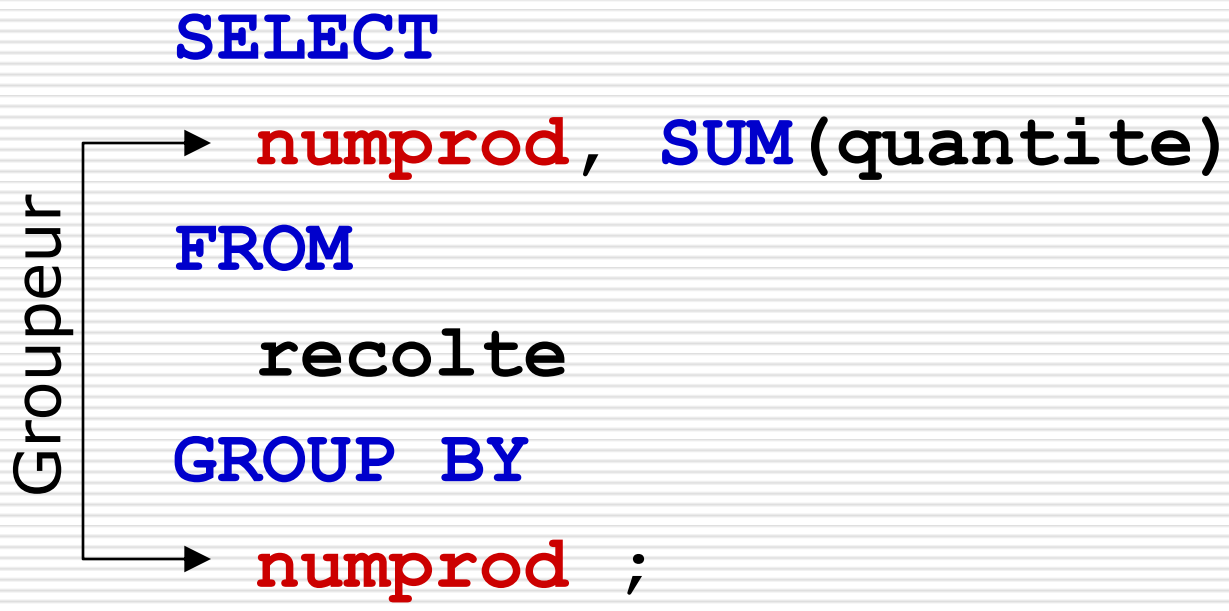
# Syntaxe SQL de groupage

- Pour regrouper des données, il faut utiliser la clause GROUP BY suivi du facteur de groupage.

```
SELECT
    <col1>, <col2>, <fonction_agrégat>(<col3>)
FROM
    ...
WHERE
    ...
GROUP BY <col1>, <col2>;
```

# Groupage : exemple 1

- Calculer les quantités récoltés pour chaque producteur.




```
SELECT
   numprod, SUM(quantite)
FROM
   recolte
GROUP BY
   numprod ;
```

# Groupage : exemple 2

- Calculer le nombre de films dramatiques réalisés par chaque réalisateur.

Film(num_film, *num_ind*, titre, genre, annee)

```
SELECT num_ind, COUNT(*)
FROM Film
WHERE genre = 'Drame'
GROUP BY num_ind;
```

# Groupage : la clause HAVING

- Elle permet d'opposer un prédicat à un groupe.

```
SELECT num_client, SUM(montant)
FROM commandes
GROUP BY num_client
HAVING SUM(montant) > 1000;
```

- HAVING s'applique sur le résultat de la requête.
- *La clause **HAVING** élimine des groupes comme la clause **WHERE** élimine des lignes !*

# WHERE ou HAVING ?

- Question 1 : Calculer pour chaque cinéma le nombre de projections dont la date est antérieure à '2000-01-01'.

- Question 2 : Quels sont les cinémas dont le nombre de projections est supérieur ou égal à 4 ?

  Projection(*num_cine, num_film*, pdate)

# WHERE ou HAVING ?

- Réponse 1 :

```
SELECT num_cine, COUNT(*)
FROM Projection
WHERE pdate < DATE '2000-01-01'
GROUP BY num_cine;
```

- Réponse 2 :

```
SELECT num_cine, COUNT(*)
FROM Projection
GROUP BY num_cine
HAVING COUNT(*) >= 4;
```

# 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.

# L’echec 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

# 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

# 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 : LEFT avec pivot

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

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

| | | | |
|---|---|---|---|
| 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 : RIGHT avec pivot

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

```
SELECT *
FROM tab1,tab2
WHERE 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 |

# Les jointures externes : exemple 1

**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 : 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 jouer 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, jouer j
WHERE i.num_ind = j.num_ind (+)
GROUP BY i.num_ind, i.nom, i.prenom;
```

# 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**

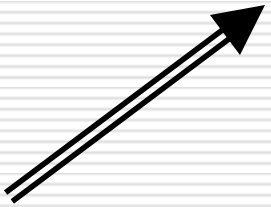
```
SELECT * FROM tab1
LEFT OUTER JOIN tab2
ON tab1.col11 = tab2.col21
WHERE tab1.col11 <> 'a';
```

# 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 $\bowtie_{col11=col21 \text{ and } col11 \neq 'a'}$ 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' ;
```

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

# Requêtes complexes

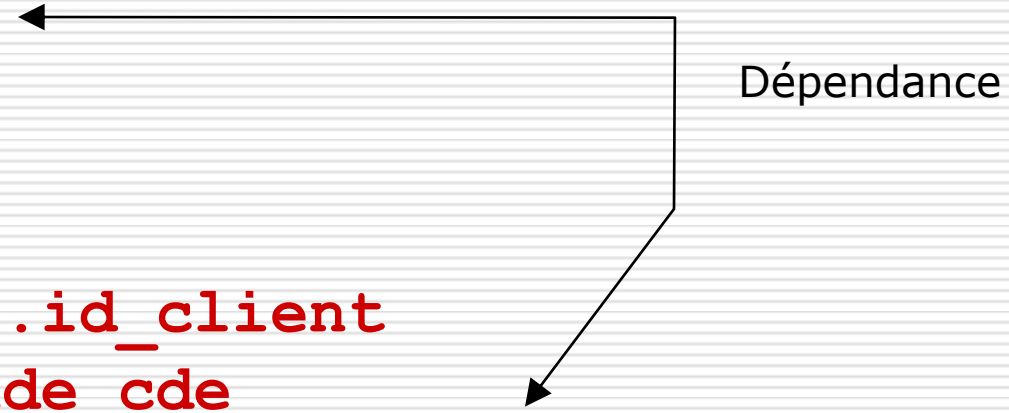
- Décomposer le problème en sous-problèmes simples
- Relier les sous-résultats par :
  - opérateur ensembliste
  - jointure
  - sous-requête
- Projeter les informations utiles

# 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)
```




- 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 ÷ (films de Lars von Trier)

# 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, jouer 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');
```