Bases de Données

Amélie Gheerbrant

Université Paris Diderot
UFR Informatique
Institut de Recherche en Informatique Fondamentale

amelie@irif.fr

18 mars 2021

L'agrégation

- Dans l'algèbre relationnelle, les conditions sont évaluées pour un tuple à la fois
- Or, parfois on s'intéresse à des propriétés dépendant de groupes de tuples
- Exemple : trouver le nombre de films qui passent en ce moment
- Le nombre de films ou le nombre de séances par films ?
- Dans ce contexte, la question des doublons est importante.

cinéma	titre
Le Louxor	Les Créatures
MK2 Quai de Seine	Virgin Suicides
MK2 Quai de Seine	Lost in Translation
Le Louxor	Cléo de 5 à 7
Le Louxor	Lost in Translation
Le Louxor	Virgin Suicides

Doublons

```
SELECT * FROM T1

A1   A2
---- ----
1    2
2    1
1    1
2    2

SELECT A1 FROM T1

A1
--
1
2
1
2
```

Doublons

- SELECT ne correspond pas exactement à l'opérateur de projection de l'algèbre relationnelle.
- La projection retourne l'ensemble $\{1, 2\}$
- SELECT conserve les doublons
- Comment omettre les doublons ? Utiliser SELECT DISTINCT

```
SELECT DISTINCT A1 FROM T1

A1
--
1
2
```

Gérer les doublons

- Jusqu'à présent dans l'algèbre relationnelle, on a opéré sur des ensembles. SQL, opère en fait sur des multi-ensembles, i.e., des ensembles pouvant contenir des doublons.
- Requière de petits ajustements
- La projection π ne retire plus les doublons :

$$\pi_A \left(\begin{array}{c|c} \text{A} & \text{B} \\ \hline a_1 & b_1 \\ \hline a_2 & b_2 \\ \hline a_1 & b_2 \end{array} \right) = \{a_1, a_2, a_1\}$$

  Ici a_1 apparaît deux fois.
- Il y a une opération spéciale d'élimination des doublons δ : $\delta(\{a1, a2, a1\}) = \{a1, a2\}$

Gérer les doublons : l'union

- L'opération d'union groupe deux multi-ensembles :

$$S = \{1, 1, 2, 2, 3, 3\}$$

$$T = \{1, 2, 2, 2, 3\}$$

$$S \cup T = \{1, 1, 1, 2, 2, 2, 2, 2, 3, 3, 3\}$$

  i.e., si *a* occure k fois dans S, et m fois dans T, alors *a* occure k + m fois dans $S \cup T$.

- Ceci ne correspond pas à l'opération `UNION` de SQL, qui élimine les doublons.
- Pour conserver les doublons, utiliser `UNION ALL` :

```
SELECT * FROM S
   UNION ALL
SELECT * FROM T;
```

Gérer les doublons : l'intersection

- L'opération d'intersection conserve le nombre d'occurrences minimal d'un élément :

$$S = \{1, 1, 2, 2, 3, 3\}$$

$$T = \{1, 2, 2, 2, 3\}$$

$$S \cap T = \{1, 2, 2, 3\}$$

  i.e., si *a* occure k fois dans S, et m fois dans T, alors *a* occure min(k, m) fois dans $S \cap T$.
- Ceci ne correspond pas à l'opération `INTERSECT` de SQL, qui élimine les doublons.
- Pour conserver les doublons, utiliser `INTERSECT ALL` :

```
SELECT * FROM S
   INTERSECT ALL
SELECT * FROM T;
```

Gérer les doublons : la différence

- L'opération de différence fonctionne comme suit :

$$S = \{1, 1, 2, 2, 3, 3\}$$

$$T = \{1, 2, 2, 2, 3\}$$

$$S - T = \{1, 3\}$$

  i.e., si a occure k fois dans S, et m fois dans T, alors a occure $k - m$ fois dans $S - T$.

- Ceci ne correspond pas à l'opération `EXCEPT` de SQL, qui élimine les doublons.
- Pour conserver les doublons, utiliser `EXCEPT ALL` :

```
SELECT * FROM S
   EXCEPT ALL
SELECT * FROM T;
```

Lois de l'algèbre relationnelle sur les multi-ensembles

- Certaines lois qui valent sur les ensembles valent aussi sur les multi-ensembles, e.g., :
  - commutativité de $\cup$ et $\cap$
  - associativité de $\cup$ et $\cap$
- Mais beaucoup d'autres lois ne passent pas aux multi-ensembles...
- Exemple : $R \cap (S \cup T) \equiv (R \cap S) \cup (R \cap T)$
  Soit $R = S = T = \{1\}$:
  - $S \cup T = \{1, 1\}$ et donc $R \cap (S \cup T) = \{1\}$
  - $(R \cap S) \cup (R \cap T) = \{1\} \cup \{1\} = \{1, 1\}$

  Or $\{1, 1\} \neq \{1\}$...

Post Scriptum : les pièges de l'ensemble vide

- Soit trois relations, S, T, R, sur le même attribut A.
- Requête : calculer $Q = R \cap (S \cup T)$
- La requête suivante a l'air d'exprimer ça correctement :

```
SELECT R.A
FROM R, S, T
WHERE R.A=S.A OR R.A=T.A;
```

- Soit $R = S = \{1\}$, $T = \emptyset$. Alors $Q = \{1\}$, mais la requête SQL produit la table vide...
- Pourquoi ?

Post Scriptum : les pièges de l'ensemble vide

- Soit trois relations, S, T, R, sur le même attribut A.
- Requête : calculer $Q = R \cap (S \cup T)$
- La requête suivante a l'air d'exprimer ça correctement :

```
SELECT R.A
FROM R, S, T
WHERE R.A=S.A OR R.A=T.A;
```

- Soit $R = S = \{1\}$, $T = \emptyset$. Alors $Q = \{1\}$, mais la requête SQL produit la table vide...
- Pourquoi ?
- Si T est vide, alors $R \times S \times T$ est vide aussi !

Post Scriptum : les pièges de l'ensemble vide

- Soit trois relations, S, T, R, sur le même attribut A.
- Requête : calculer $Q = R \cap (S \cup T)$
- S'il y a des chances que S ou T soit vide, préférer par exemple :

```
(SELECT R.A
FROM R, S
WHERE R.A=S.A)
UNION
(SELECT R.A
FROM R, T
WHERE R.A=T.A);
```

Les expressions dans les requêtes

- Avec les types (numériques, chaînes de caractères, etc), nous avons des opérations spécifiques aux types et donc des conditions de sélection spécifiques à ces types.

```
CREATE TABLE Finance (titre char(20), budget int,
recette int);
INSERT INTO Finance VALUES ('Shining', 19, 100);
INSERT INTO Finance VALUES ('Star wars', 11, 513);
INSERT INTO Finance VALUES ('Wild wild west', 170, 80);
```

Les expressions dans les requêtes

- Trouver les films qui ont perdu de l'argent

```
SELECT titre
FROM Finance
WHERE recette < budget;
```

- Trouver les films qui ont généré au moins 10 fois plus de recette que ce qu'ils ont coûté

```
SELECT titre
FROM Finance
WHERE recette > 10 * budget;
```

- Trouver le bénéfice généré par chaque film :

```
SELECT titre, recette - budget as bénéfice
FROM Finance
WHERE recette - budget > 0;
```

Les expressions dans les requêtes

On peut utiliser des expressions dans les requêtes, à la place des attributs simples, par exemple dans les clauses `WHERE` et `SELECT` :

```
create table Article(
   id integer primary key,
   nom varchar(25) not null,
   prix decimal(7,2),
   prix_solde decimal(7,2),
   check(prix_solde < prix)
);
```

Article

id	nom	prix	prix_solde
8	pantalon	99.99	59.95
5	pull	40	26
3	veste	120	79.95

```
select nom, prix - prix_solde
from Article
where prix - prix_solde > 30 ;
```

→

nom	prix - prix_solde
pantalon	40.04
veste	40.05

Fonctions et opérateurs prédéfinis

On peut utiliser des fonctions et opérateurs prédéfinis dans les expressions (spécifiques au SGBD $\neq$ standard SQL). Exemples avec PostgreSQL :

- ABS(num) : valeur absolue
- str1 || str2 : concaténation de chaînes de caractères
- NOW() : la date et heure courante
- CEILING(num) : l'entier immédiatement supérieur ou égal à num
- FLOOR(num) : l'entier immédiatement inférieur ou égal à num
- current_date : la date courante
- current_time : l'heure courante
- ...

Fonctions et opérateurs prédéfinis

```
SELECT 'réalisateur : ' || nom AS real
FROM Artistes ;
```

```
+----------------------------+
|           real             |
+----------------------------+
|réalisateur : Scott         |
|réalisateur : Hitchcock     |
|réalisateur : Kurosawa      |
+----------------------------+
```

```
SELECT EXTRACT(YEAR FROM NOW()) - naissance AS
age
FROM Artistes ;
```

```
+------+
| age  |
+------+
|73    |
|117   |
|106   |
+------+
```

Artistes

id	*nom*	*prenom*	*naissan*
1	Scott	Ridley	1943
2	Hitchcock	Alfred	1899
3	Kurosawa	Akira	1910

Fonctions et opérateurs prédéfinis

- POSITION (subtring IN string) Position de substring dans string
- LENGTH (str) Renvoie la longueur de str (nombre de caractères)
- SUBSTRING(chaîne [from int] [for int])
- `CASE WHEN cond THEN val1`
  `ELSE val2`
  `END`
  Renvoie val1 si cond est vrai, val2 sinon (plusieurs WHEN possibles)

C.f. https ://docs.postgresql.fr/9.5/functions.html
Exemples avec CASE : https ://sql.sh/cours/case

Les comparaisons avec LIKE

- Motifs d'attributs avec LIKE
- Les motifs sont construits à partir de :
  _, qui représente n'importe quelle lettre
  %, qui représente n'importe quelle sous-chaîne, dont l'ensemble vide
- Exemples :
  addresse LIKE '%Paris%'
  le motif '_a_b_' représente cacbc, aabba, etc
  le motif %a%b_' représente ccaccbc, aaaabcbcbbd, aba, etc

Les comparaisons avec LIKE

```
SELECT titre, realisateur
FROM film
WHERE realisateur LIKE 'Kieslowsk_';
```

retourne l'ensemble vide.

- Parce que parfois x=y est vrai, alors que x `LIKE` y est faux !
- Raison : les espaces
- `'Kieslowski '` = `'Kieslowski '` est vrai, mais `'Kieslowski '` `LIKE` `'Kieslowski '` est faux.
- Si realisateur défini comme char(10), alors `'Kieslowski'` est vraiment `'Kieslowski '` et ne correspond donc pas à `'Kieslowski_'`.

LIKE et les espaces

- Solution 1 : utiliser des déclaration de type varchar (ou char varying)
- Solution 2 : utiliser 'Kieslowsk%' comme motif
- Solution 3 : utiliser la fonction TRIM :

```
SELECT titre, realisateur
FROM Film
WHERE TRIM(TRAILING FROM realisateur) LIKE 'Polansk_';
```

- TRIM TRAILING élimine les espaces de fin (LEADING élimine les espaces de début, BOTH élimine les deux)
- Attention : tous les systèmes n'aiment pas ça...
- Remarque : type char utile par exemple pour les codes postaux (toujours 5 chiffres en France)

Ajouter des attributs... vers les requêtes avec agrégation

```
ALTER TABLE Film ADD COLUMN Duree int DEFAULT 0;

UPDATE Film
SET Duree = 131
WHERE titre='Chinatown';

UPDATE Film
SET Duree = 146
WHERE titre='Shining';
```

ajoute l'attribut durée, et insère des valeurs pour cet attribut.

Ajouter des attributs... vers les requêtes avec agrégation

```
ALTER TABLE Séance ADD COLUMN heure int DEFAULT 0;

UPDATE Séance
SET heure = 18
WHERE cinema='Le Champo' AND titre='Chinatown';

INSERT INTO Film VALUES ('Le Champo', 'Chinatown', 21);
```

ajoute l'attribut heure, et insère des valeurs pour cet attribut.

Plus d'une séance par film : utiliser d'abord `UPDATE`, puis `INSERT`.

Plus d'exemples avec de l'arithmétique

Requête : je veux voir un film de Varda. Je ne peux pas y aller avant 19h, et je veux être sortie avant 23h.

Je veux retourner : les cinémas et l'heure exacte à laquelle je sortirai, si mes conditions sont satisfaites.

```
SELECT S.cinema, S.heure + (F.duree/60.0) AS heurefin
FROM Séance S, Film F
WHERE F.titre=S.titre
    AND F.realisateur='Varda'
    AND S.heure >= 19
    AND S.heure + (F.duree/60.0) < 23;
```

Attention à l'ordre de préséance des opérateurs

Sachant que j'ai un temps de trajet maximum de 45mn, je veux retourner : les cinémas et l'heure maximum à laquelle je serai rentrée chez moi, si mes conditions sont satisfaites.

```
SELECT S.cinema, S.heure + ((F.duree+45)/60.0) AS h_retour
FROM Séance S, Film F
WHERE F.titre=S.titre
    AND F.realisateur='Varda'
    AND S.heure >= 19
    AND S.heure + (F.duree/60.0) < 23;
```

Sans les parenthèses en rouge autour de `F.duree+45`, les valeurs retournées pour h_retour ne correspondront pas à des heures de retour, mais à `F.duree+(45/60.0)`, c'est à dire `F.duree+0,75` (/ et * sont prioritaires sur + et -).

SQL n'est pas un langage de programmation

- On peut donc calculer beaucoup de choses en SQL
- Comment calculer 2+2 ?
- D'abord, il nous faut une table sur laquelle opérer :

```
CREATE TABLE Foo (a int);
```

- 2+2 doit aller dans une clause SELECT. Il faut aussi lui donner un nom d'attribut.
- Essai :

```
SELECT 2+2 as X
FROM Foo;
```

```
X
-----------
  0 record(s) selected.
```

SQL n'est pas un langage de programmation

- Problème : il n'y avait pas de tuple dans `Foo`
- Remplissons notre table :

```
INSERT INTO Foo VALUES 1;
INSERT INTO Foo VALUES 5;
SELECT 2+2 as X
FROM Foo;
```

```
 X
 -----------
           4
           4

  2 record(s) selected.
```

SQL n'est pas un langage de programmation

- Il faut aussi éliminer les doublons...
- Et finalement :

```
SELECT DISTINCT 2+2 as X
FROM Foo;
```

```
 X
-----------
           4

  1 record(s) selected.
```

Requêtes d'agrégat simple

- Compter le nombre de tuples dans Film

```
SELECT COUNT(*)
FROM Film;
```

- Additionner la durée de tous les films

```
SELECT SUM(durée)
FROM Film;
```

Les doublons et l'agrégation

- Trouver le nombre de réalisateurs, approche naïve :

```
SELECT COUNT(realisateur)
FROM Film;
```

retourne le nombre de tuples dans Film.
Raison : `SELECT` ne supprime pas les doublons.

- Requête correcte :

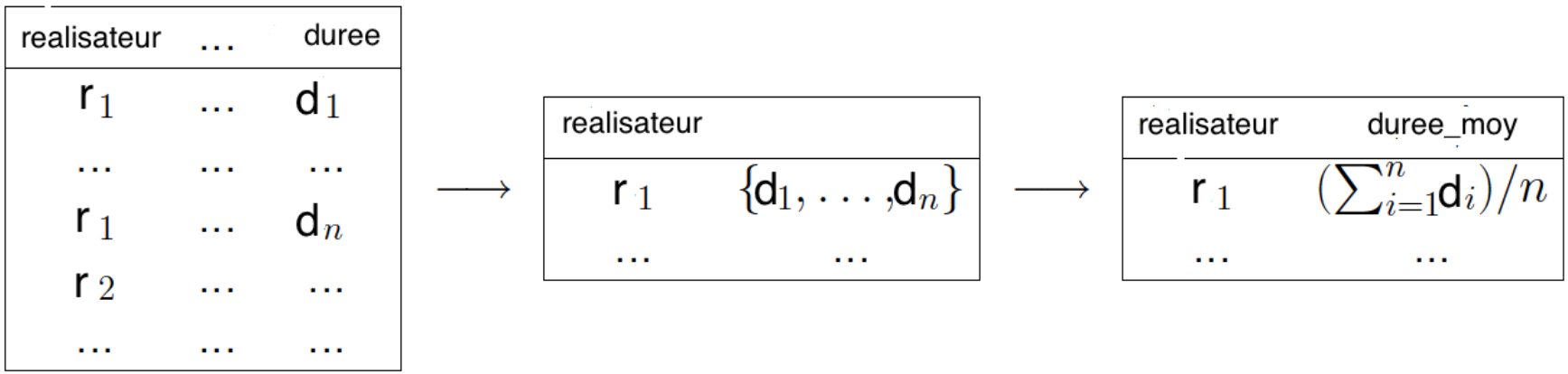
```
SELECT COUNT(DISTINCT realisateur)
FROM Film;
```

Agrégation et GROUP BY

Pour chaque réalisateur, retourner la duréee moyenne de ses films.

```
SELECT realisateur, AVG(duree) AS duree_moy
FROM Film
GROUP BY realisateur;
```

Comment GROUP BY fonctionne-t-il ?



(Attention : tous les attributs qui ne sont pas des agrégats et qui figurent dans la clause SELECT doivent figurer aussi dans la clause GROUP BY.)

Agrégation et GROUP BY

Pour chaque réalisateur, retourner le temps moyen de ses films.

```
SELECT realisateur, AVG(duree) AS duree_moy
FROM Film
GROUP BY realisateur;
```

Comment GROUP BY fonctionne-t-il ?

réalisateur	titre	durée
Varda	Les Créatures	92
Coppola	Virgin Suicides	96
Chytilova	Sedmikrasky	74
Varda	Le bonheur	79
Coppola	Lost inTranslation	102
Varda	Cléo de 5 à 7	90

Agrégation et GROUP BY

Pour chaque réalisateur, retourner le temps moyen de ses films.

```
SELECT realisateur, AVG(duree) AS duree_moy
FROM Film
GROUP BY realisateur;
```

Les tuples de la table `Film` sont d'abord répartis en groupes

réalisateur	titre	durée
Coppola	Lost inTranslation	102
Coppola	Virgin Suicides	96
Chytilova	Sedmikrasky	74
Varda	Le bonheur	79
Varda	Les Créatures	92
Varda	Cléo de 5 à 7	90

Agrégation et GROUP BY

Pour chaque réalisateur, retourner le temps moyen de ses films.

```
SELECT realisateur, AVG(duree) AS duree_moy
FROM Film
GROUP BY realisateur;
```

Puis, pour chaque groupe, la moyenne des durées est calculée

réalisateur	titre	durée
Coppola	Lost inTranslation	102
Coppola	Virgin Suicides	96
Chytilova	Sedmikrasky	74
Varda	Le bonheur	79
Varda	Les Créatures	92
Varda	Cléo de 5 à 7	90

calcul de l'agrégat

réalisateur	durée_moy
Coppola	99
Chytilova	74
Varda	87

Agrégation et GROUP BY

Tous les attributs qui ne sont pas des agrégats et qui figurent dans le SELECT doivent figurer aussi dans le GROUP BY : en présence d'agrégats, SELECT fait référence aux groupes et non aux tuples

```
SELECT realisateur, titre, AVG(duree) AS duree_moy
FROM Film
GROUP BY realisateur;
```

Comment choisir le titre associé à un groupe ? !

réalisateur	titre	durée
Coppola	Lost inTranslation	102
Coppola	Virgin Suicides	96
Chytilova	Sedmikrasky	74
Varda	Le bonheur	79
Varda	Les Créatures	92
Varda	Cléo de 5 à 7	90

calcul de l'agrégat

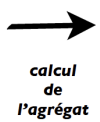
réalisateur	titre	durée moy
Coppola	?!	99
Chytilova	?!	74
Varda	?!	87

Agrégation et GROUP BY

```
SELECT realisateur, titre, AVG(duree) AS duree_moy
FROM Film
GROUP BY realisateur, titre ;
```

Pas très intéressant :

réalisateur	titre	durée
Varda	Les Créatures	92
Coppola	Virgin Suicides	96
Chytilova	Sedmikrasky	74
Varda	Le bonheur	79
Coppola	Lost inTranslation	102
Varda	Cléo de 5 à 7	90



réalisateur	titre	durée_moy
Varda	Les Créatures	92
Coppola	Virgin Suicides	96
Chytilova	Sedmikrasky	74
Varda	Le bonheur	79
Coppola	Lost inTranslation	102
Varda	Cléo de 5 à 7	90

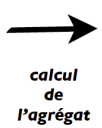
(L'ordre des attributs listés dans le GROUP BY est sans importance)

Agrégation et GROUP BY

Même sémantique que :

```
SELECT realisateur, titre, duree AS duree_moy
FROM Film ;
```

réalisateur	titre	durée
Varda	Les Créatures	92
Coppola	Virgin Suicides	96
Chytilova	Sedmikrasky	74
Varda	Le bonheur	79
Coppola	Lost in Translation	102
Varda	Cléo de 5 à 7	90



réalisateur	titre	durée_moy
Varda	Les Créatures	92
Coppola	Virgin Suicides	96
Chytilova	Sedmikrasky	74
Varda	Le bonheur	79
Coppola	Lost in Translation	102
Varda	Cléo de 5 à 7	90

Agrégation et doublons

Table	A1	A2	A3
	a	1	5
	a	1	2
	a	2	2
	a	2	3

```
SELECT A1, AVG(A3) as A4
FROM Table
GROUP BY A1;
```

A1	A4
a	?

Agrégation et doublons

Une approche : prendre toutes les valeurs de A3 et calculer leur moyenne

$$\frac{5+2+2+3}{4} = 3$$

Une autre approche : se restreindre aux attributs A1 et A3

$$\pi_{\mathsf{A1,A3}} \left(\begin{array}{ccc} \mathsf{A1} & \mathsf{A2} & \mathsf{A3} \\ \hline \mathsf{a} & 1 & 5 \\ \mathsf{a} & 1 & 2 \\ \mathsf{a} & 2 & 2 \\ \mathsf{a} & 2 & 3 \end{array} \right) = \left(\begin{array}{cc} \mathsf{A1} & \mathsf{A3} \\ \hline \mathsf{a} & 5 \\ \mathsf{a} & 2 \\ \mathsf{a} & 3 \end{array} \right)$$

$$\frac{5+2+3}{3} = \frac{10}{3}$$

Agrégation et doublons

- Approche SQL : toujours garder les doublons.
- La bonne réponse est donc 3.
- Attention, cependant :

  `SELECT AVG(A2) FROM Table;`

  retourne 1
- Raison : l'arrondi
- Solution : convertir comme nombre réel :

  `SELECT AVG(CAST (A2 AS REAL)) FROM Table;`

  retourne 1.5
- Syntaxe de `CAST`

  CAST (<attribut> AS <type>)

Plus sur les doublons

- Et si on veut éliminer les doublons avant de calculer l'agrégat ?

- Utiliser `DISTINCT`

  ```
  SELECT AVG(DISTINCT A3) FROM Table;
  ```

  donne 3, à cause de l'arrondi, mais

  ```
  SELECT AVG(DISTINCT CAST (A3 AS REAL)) FROM Table;
  ```

  produit 3.3333..., comme prévu

- Un truc pour convertir les entiers en réels :

  ```
  SELECT AVG(A3 + 0.0) FROM Table;
  ```

  ou encore

  ```
  SELECT AVG(A3)::real FROM Table;
  ```

Autres fonctions d'agrégation

- `MIN` calcule la valeur minimum d'une colonne
- `MAX` calcule la valeur maximum d'une colonne
- `MIN` et `MAX` produisent le même résultat peu importe les doublons
- `MIN` et `MAX` s'appliquent aussi aux chaînes de caractères : ordre lexicographique
- `SUM` additionne tous les éléments d'une colonne
- `SUM DISTINCT` additionne tous les éléments distincts d'une colonne
- `COUNT` compte les éléments d'une colonne ;
- `COUNT DISTINCT` compte les éléments distincts d'une colonne

SUM, COUNT et doublons

- `SELECT COUNT(A3) FROM Table;` donne 4
- `SELECT COUNT(DISTINCT A3) FROM Table;` donne 3
- `SELECT SUM(A3) FROM Table;` donne 12
- `SELECT SUM(DISTINCT A3) FROM Table;` donne 10
- `SELECT MIN(A3) FROM Table;` et `SELECT MIN(DISTINCT A3) FROM Table;` donnent le même résultat.
- Idem pour `MAX(A3)` et `MAX(DISTINCT A3)`.

Table	A1	A2	A3
	a	1	5
	a	1	2
	a	2	2
	a	2	3

Sélections basées sur des résultats d'agrégation : HAVING

- Pour chaque réalisateur retourner le temps moyen de ses films, à condition qu'il ait réalisé au moins un film de plus de 1 heure et demi.
- Idée : calculer deux agrégats : AVG(duree) et MAX(duree) et choisir seulement les réalisateurs pour lesquels MAX(duree) > 90.
- Syntaxe de SQL : HAVING

```
SELECT realisateur, AVG(duree+0.0)
FROM Film
GROUP BY realisateur
HAVING MAX(duree) > 90;
```

(Attention : seuls des attributs contenant des opérateurs d'agrégation, ou bien des attributs figurant dans le group by, peuvent apparaître dans la clause HAVING)

Sélections basées sur des résultats d'agrégation : HAVING

Pour chaque réalisateur retourner le temps moyen de ses films, à condition qu'il ait réalisé au moins un film de plus de 1 heure et demi.

```
SELECT realisateur, AVG(duree+0.0)
FROM Film
GROUP BY realisateur
HAVING MAX(duree) > 90;
```

réalisateur	titre	durée
Varda	Les Créatures	92
Coppola	Virgin Suicides	96
Chytilova	Sedmikrasky	74
Varda	Le bonheur	79
Coppola	Lost inTranslation	102
Varda	Cléo de 5 à 7	90

Sélections basées sur des résultats d'agrégation : HAVING

Pour chaque réalisateur retourner le temps moyen de ses films, à condition qu'il ait réalisé au moins un film de plus de 1 heure et demi.

```
SELECT realisateur, AVG(duree+0.0)
FROM Film
GROUP BY realisateur
HAVING MAX(duree) > 90;
```

Les tuples de la table `Film` sont d'abord répartis en groupes

réalisateur	titre	durée
Coppola	Lost inTranslation	102
Coppola	Virgin Suicides	96
Chytilova	Sedmikrasky	74
Varda	Le bonheur	79
Varda	Les Créatures	92
Varda	Cléo de 5 à 7	90

Sélections basées sur des résultats d'agrégation : HAVING

Pour chaque réalisateur retourner le temps moyen de ses films, à condition qu'il ait réalisé au moins un film de plus de 1 heure et demi.

```
SELECT realisateur, AVG(duree+0.0)
FROM Film
GROUP BY realisateur
HAVING MAX(duree) > 90;
```

Seuls les groupes satisfaisant la condition du HAVING sont conservés :

réalisateur	titre	durée
Coppola	Lost inTranslation	102
Coppola	Virgin Suicides	96
Varda	Le bonheur	79
Varda	Les Créatures	92
Varda	Cléo de 5 à 7	90

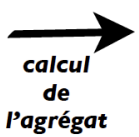
Sélections basées sur des résultats d'agrégation : HAVING

Pour chaque réalisateur retourner le temps moyen de ses films, à condition qu'il ait réalisé au moins un film de plus de 1 heure et demi.

```
SELECT realisateur, AVG(duree+0.0)
FROM Film
GROUP BY realisateur
HAVING MAX(duree) > 90;
```

Enfin, l'agrégat du SELECT est calculé :

réalisateur	titre	durée
Coppola	Lost inTranslation	102
Coppola	Virgin Suicides	96
Varda	Le bonheur	79
Varda	Les Créatures	92
Varda	Cléo de 5 à 7	90



réalisateur	durée_moy
Coppola	99
Varda	87

Sélections basées sur des résultats d'agrégation : HAVING

Seuls des attributs contenant des opérateurs d'agrégation, ou bien des attributs figurant dans le group by, peuvent apparaître dans le `HAVING`

```
SELECT realisateur, AVG(duree+0.0)
FROM Film
GROUP BY realisateur
HAVING duree > 90 ;
```

Quelle durée ? !

réalisateur	titre	durée
Coppola	Lost in Translation	102
Coppola	Virgin Suicides	96
Chytilova	Sedmikrasky	74
Varda	Le bonheur	79
Varda	Les Créatures	92
Varda	Cléo de 5 à 7	90

Sélections basées sur des résultats d'agrégation : HAVING

Seuls des attributs contenant des opérateurs d'agrégation, ou bien des attributs figurant dans le group by, peuvent apparaître dans le `HAVING`

```
SELECT realisateur, AVG(duree+0.0)
FROM Film
GROUP BY realisateur
HAVING duree > 90 ;
```

En présence d'agrégats SELECT et HAVING font référence aux groupes et non aux tuples

réalisateur	titre	durée
Coppola	Lost in Translation	102
Coppola	Virgin Suicides	96
Chytilova	Sedmikrasky	74
Varda	Le bonheur	79
Varda	Les Créatures	92
Varda	Cléo de 5 à 7	90

Agrégation et jointures

- Les requêtes d'agrégation peuvent utiliser plus d'une relation.
- Pour tout cinéma montrant au moins un film de plus de 1 heure 30, trouver la durée moyenne des films qu'on y passe.

```
SELECT F.cinema, AVG(CAST(F.duree AS REAL))
FROM Séance S NATURAL JOIN Film F
GROUP BY F.cinema
HAVING MAX(F.duree) > 90;
```

- Interprétation : générer la jointure Film $\bowtie$ Séance, et sur cette jointure exécuter la requête d'agrégation qui calcule la moyenne.

Agrégation et jointures

Pour tout cinéma montrant au moins un film de plus de 1 heure 30, trouver la durée moyenne des films qui y sont projetés.

```
SELECT F.cinema, AVG(CAST(F.duree AS REAL))
FROM Séance S NATURAL JOIN Film F
GROUP BY F.cinema
HAVING MAX(F.duree) > 90;
```

réalisateur	titre	durée
Varda	Les Créatures	92
Coppola	Virgin Suicides	96
Chytilova	Sedmikrasky	74
Varda	Le bonheur	79
Coppola	Lost in Translation	102
Varda	Cléo de 5 à 7	90

cinéma	titre
Le Louxor	Les Créatures
MK2 Quai de Seine	Virgin Suicides
MK2 Quai de Seine	Lost in Translation
Le Louxor	Cléo de 5 à 7
Le Louxor	Lost in Translation

Agrégation et jointures

Pour tout cinéma montrant au moins un film de plus de 1 heure 30, trouver la durée moyenne des films qui y sont projetés.

```
SELECT F.cinema, AVG(CAST(F.duree AS REAL))
FROM Séance S NATURAL JOIN Film F
GROUP BY F.cinema
HAVING MAX(F.duree) > 90;
```

On évalue d'abord la partie `FROM` (i.e., on fait la jointure) :

réalisateur	titre	durée	cinéma
Varda	Les Créatures	92	Le Louxor
Coppola	Virgin Suicides	96	MK2 Quai de Seine
Coppola	Lost inTranslation	79	Le Louxor
Coppola	Lost inTranslation	102	MK2 Quai de Seine
Varda	Cléo de 5 à 7	90	Le Louxor

Agrégation et jointures

Pour tout cinéma montrant au moins un film de plus de 1 heure 30, trouver la durée moyenne des films qui y sont projetés.

```
SELECT F.cinema, AVG(CAST(F.duree AS REAL))
FROM Séance S NATURAL JOIN Film F
GROUP BY F.cinema
HAVING MAX(F.duree) > 90;
```

Les tuples de la jointures sont répartis par groupes :

réalisateur	titre	durée	cinéma
Varda	Les Créatures	92	Le Louxor
Coppola	Lost inTranslation	79	Le Louxor
Varda	Cléo de 5 à 7	90	Le Louxor
Coppola	Virgin Suicides	96	MK2 Quai de Seine
Coppola	Lost inTranslation	102	MK2 Quai de Seine

Agrégation et jointures

Pour tout cinéma montrant au moins un film de plus de 1 heure 30, trouver la durée moyenne des films qui y sont projetés.

```
SELECT F.cinema, AVG(CAST(F.duree AS REAL))
FROM Séance S NATURAL JOIN Film F
GROUP BY F.cinema
HAVING MAX(F.duree) > 90;
```

Seuls les groupes satisfaisant la condition du `HAVING` sont conservés (ici, tous) :

réalisateur	titre	durée	cinéma
Varda	Les Créatures	92	Le Louxor
Coppola	Lost inTranslation	79	Le Louxor
Varda	Cléo de 5 à 7	90	Le Louxor
Coppola	Virgin Suicides	96	MK2 Quai de Seine
Coppola	Lost inTranslation	102	MK2 Quai de Seine

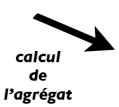
Agrégation et jointures

Pour tout cinéma montrant au moins un film de plus de 1 heure 30, trouver la durée moyenne des films qui y sont projetés.

```
SELECT F.cinema, AVG(CAST(F.duree AS REAL))
FROM Séance S NATURAL JOIN Film F
GROUP BY F.cinema
HAVING MAX(F.duree) > 90;
```

Enfin, l'agrégat du SELECT est calculé :

réalisateur	titre	durée	cinéma
Varda	Les Créatures	92	Le Louxor
Coppola	Lost inTranslation	79	Le Louxor
Varda	Cléo de 5 à 7	90	Le Louxor
Coppola	Virgin Suicides	96	MK2 Quai de Seine
Coppola	Lost inTranslation	102	MK2 Quai de Seine



cinéma	durée_moy
Le Louxor	87
MK2 Quai de Seine	99

Agrégation, jointures et doublons

- Les doublons peuvent donner lieu à des résultats inattendus.
- Deux tables :

R	A1	A2
	'a'	1
	'b'	2

S	A1	A3
	'a'	5
	'a'	7
	'b'	3

- Requête :

```
SELECT R.A1, SUM(R.A2)
FROM R, S
WHERE R.A1=S.A1 AND R.A1='a'
GROUP BY R.A1
HAVING MIN(S.A3) > 0;
```

- Résultat ?

Agrégation, jointures et doublons

- La table S n'est pas pertinente, et le résultat devrait être le même que celui de :

```
SELECT A1, SUM(A2)
FROM R
WHERE A1='a'
GROUP BY A1;
```

- Retourne ('a', 1)
- alors que la première requête retourne ('a', 2).

Agrégation, jointures et doublons

- Que se passe-t-il ?
- La requête construit d'abord la jointure $R \bowtie S$

R ⋈ S	A1	A2	A3
	'a'	1	5
	'a'	1	7
	'b'	2	3

- et exécute ensuite la partie agrégation sur la jointure :

```
SELECT A1, SUM(A2)
FROM R NATURAL JOIN S
WHERE A1='a'
GROUP BY A1
HAVING MIN(A3) > 0
```

- la réponse est donc ('a',2)

Agrégation, jointures et doublons

- Morale : attention aux doublons, même quand il n'y en a apparemment pas.

- Pour retourner ('a',1), utiliser DISTINCT :

```
SELECT R.A1, SUM(DISTINCT R.A2)
FROM R, S
WHERE R.A1=S.A1 AND R.A1='a'
GROUP BY R.A1
HAVING MIN(S.A3) > 0;
```

Agrégats dans le WHERE

- Les résultats d'un agrégat peuvent aussi être utilisés dans des comparaisons hors de la clause HAVING.
- Trouver les films qui sont plus longs que tous ceux qui passent actuellement au cinéma :

```
SELECT F.titre
FROM Film F
WHERE F.duree > (SELECT MAX(F1.duree)
                 FROM Film F1, Séance S
                 WHERE F1.titre=S.titre);
```

Attention : on peut utiliser $>$ ici uniquement parce que le tableau résultat de la sous-requête ne comporte qu'une colonne et un seul tuple (requête "scalaire")

Agrégats dans le WHERE

- Attention à ne pas écrire :

```
SELECT F.titre
FROM Film F
WHERE F.duree > MAX(SELECT F1.duree
                    FROM Film F1, Séance S
                    WHERE F1.titre=S.titre);
```

qui est incorrect

- A la place on peut écrire :

```
SELECT F.titre
FROM Film F
WHERE F.duree > ALL(SELECT F1.duree
                    FROM Film F1, Séance S
                    WHERE F1.titre=S.titre);
```

Agrégats dans le WHERE

- De même :
- Trouver les films qui sont plus courts qu'au moins un film qui passe actuellement au cinéma :

```
SELECT F.titre
FROM Film F
WHERE F.duree < (SELECT MAX(F1.duree)
                 FROM Film F1, Séance S
                 WHERE F1.titre=S.titre);
```

ou

```
SELECT F.titre
FROM Film F
WHERE F.duree < ANY(SELECT F1.duree
                    FROM Film F1, Séance S
                    WHERE F1.titre=S.titre);
```

- Ici on utilise `ANY` et non `ALL`

ALL versus ANY

- <valeur> <condition> ALL (<requête>) est vrai si :
  - le résultat de <requête> est l'ensemble vide, ou
  - pour toute <valeur1> dans le résultat de <requête>, <valeur> <condition> <valeur1> est vrai.
- Par exemple,
  $5 > ALL(\emptyset)$ est vrai ;
  $5 > ALL(\{1, 2, 3\})$ est vrai ;
  $5 > ALL(\{1, 2, 3, 4, 5, 6\})$ est faux.
- Remarque : `NOT IN` signifie la même chose que <>`ALL`.
- Attention : le tableau résultat de la sous requête <requête> ne doit contenir qu'une seule colonne.

ALL versus ANY

- <valeur> <condition> ANY (<requête>) est vrai s'il existe une <valeur1> dans le résultat de <requête>, telle que <valeur> <condition> <valeur1> est vrai.
- Par exemple,
  $5 < ANY(\emptyset)$ est faux ;
  $5 < ANY(\{1, 2, 3\})$ est faux ;
  $5 < ANY(\{1, 2, 3, 4, 5, 6\})$ est vrai.
- Remarque : `IN` signifie la même chose que `=ANY`
- Attention : le tableau résultat de la sous requête <requête> ne doit contenir qu'une seule colonne.

Agrégats dans le WHERE

- Toutes les comparaisons avec des résultats d'agrégat ne peuvent pas être remplacées par des comparaisons avec ANY et ALL.

- Est-ce qu'il y a un film dont la durée correspond au moins à 10% de la longueur totale de tous les autres films combinés ?

```
SELECT F.titre
FROM Film F
WHERE F.duree >= 0.1 * (SELECT SUM(F1.duree)
                        FROM Film F1
                        WHERE F1.titre <> F.titre);
```

GROUP BY et HAVING : principe

- `WHERE` filtre les lignes individuellement, alors que `HAVING` filtre les groupes (donc après regroupement)
- Conséquence :
  - dans la partie `HAVING` on ne met que des conditions à base d'agrégations ou à base d'attributs situés dans le `GROUP BY`
  - dans le `WHERE` les agrégations sont interdites

Exemple un peu plus compliqué

Trouver le réalisateur qui a tourné le plus de films :

```
SELECT realisateur, count(distinct titre)
FROM Film
GROUP BY realisateur
HAVING count(distinct titre) >=
                          ALL (SELECT count(distinct titre)
                          FROM Film
                          GROUP BY realisateur);
```

La requête principale retournera la valeur supérieure ou égale à toutes les valeurs de la sous requête.

Jointures et agrégation dans les requêtes

- Tous les systèmes n'acceptent pas le `NATURAL JOIN`
- Il y a une syntaxe plus générale :

```
SELECT A1, SUM(A2)
FROM R JOIN S ON R.A1=S.A1
WHERE A1='a'
GROUP BY A1
HAVING MIN(A3) > 0;
```

- `R JOIN S ON c` calcule

$$\sigma_c(R \times S)$$

- c peut être une condition plus compliquée qu'une simple égalité entre attributs, e.g., `R.A2 > S.A3 - 4`

Jointures et agrégation dans les requêtes

- Exemple : tous les couples de films différents ayant le même réalisateur

```
SELECT F1.réalisateur, F1.titre, F2.titre
FROM Film F1 JOIN Film F2 ON
(F1.réalisateur=F2.réalisateur
    AND
F1.titre <> F2.titre);
```

- Formulation alternative avec `USING` (suivi d'une liste d'attributs) :

```
SELECT réalisateur, F1.titre, F2.titre
FROM Film F1 JOIN Film F2
USING(réalisateur)
WHERE F1.titre <> F2.titre;
```

Jointures et agrégation dans les requêtes

- Attention :
  - dans la version avec `JOIN ON` les attributs de jointures ne sont pas fusionnés
  - dans la version avec `USING` les attributs de jointures sont fusionnés (équivalent à la jointure naturelle)
- $\Rightarrow$: une seule versus deux colonnes `realisateur` dans le tableau résultat

Thêta-Jointures

- Les expressions de type `R join S on c` sont souvent appelées Thêta-jointures et sont souvent inclues dans l'algèbre relationnelle :

$$R \bowtie_{\theta} S$$

- Il ne s'agit pas d'une nouvelle opération de l'algèbre relationnelle mais simplement d'une abréviation pour $\sigma_{\theta}(R \times S)$
- Raison pour le nom : les conditions étaient traditionnellement dénotées par θ

Jointures et agrégation dans les requêtes

- Attention : la relation dont provient un attribut n'est plus claire :

```
SELECT A1, SUM(A2)
FROM R JOIN S ON R.A1=S.A1
GROUP BY R.A1;
```

- Si l'on utilise `GROUP BY A1`, et non `GROUP BY R.A1`, SQL proteste : la référence à la colonne "A1" est ambiguë
- `SELECT * FROM R JOIN S ON R.A1=S.A1;`

```
A1 A2           A1 A3
-- ---------- -- ----------
a           1 a           5
a           1 a           7
b           2 b           3
```

Jointures et agrégation dans les requêtes

- Pour utiliser l'agrégation, il faut spécifier d'où les attributs viennent :

```
SELECT F.cinema, MAX(F.duree)
FROM Film F JOIN Séance S ON F.titre=S.titre
GROUP BY S.cinema;
```

  trouve les cinémas et la durée du film le plus long qu'on y joue
- Notez l'utilisation d'alias à l'intérieur du JOIN
- On peut aussi donner des noms spécifiques aux jointures :

```
SELECT JC.cinema, MAX(JC.duree)
FROM (Film NATURAL JOIN Séance) AS JC;
GROUP BY JC.cinema;
```

Jointures et agrégation dans les requêtes

- Les jointures peuvent vite devenir compliquées :

```
( ( R JOIN S ON <cond1> )  AS  Table1
     JOIN
     ( U JOIN V ON <cond2> )  AS  Table2
    ON <cond3> )
```

- Il faut faire attention lorsqu'on référence les tables dans les conditions, e.g., :
  - `<cond1>` peut faire référence à `R, S`, mais pas à `U, V, Table1, Table2`
  - `<cond2>` peut faire référence à `U, V`, mais pas à `R, S, Table1, Table2`
  - `<cond3>` peut faire référence à `Table1, Table2`, mais pas à `R, S, U, V`

Jointures et agrégation dans les requêtes

- Ambiguïté, un même nom d'attribut peut figurer deux fois dans une même table !
- Procédons à quelques tests avec Postgres :

```
ameliegheerbrant=# select * from traduction ;
 travail | nom_livre |    nom_traducteur    | premierepage | dernierepage | fini
---------+-----------+----------------------+--------------+--------------+------
      14 | La pluie  | Brigitte Durand      |           10 |           33 | f
      15 | La pluie  | Renaud Blanc         |           70 |           93 | t
      16 | La pluie  | Brigitte Durand      |           94 |          150 | t
      17 | La pluie  | Yukiko Tanaka        |            1 |            9 | f
      21 | La neige  | Salim BenNour        |          100 |          180 | f
      22 | La neige  | Brigitte Durand      |           30 |          130 | f
      31 | Le soleil | Yukiko Tanaka        |            1 |           40 | f
      32 | Le soleil | Renaud Blanc         |          100 |          150 | f
      33 | Le soleil | Douglas Brown        |            1 |           10 | f
      41 | Le vent   | Salim BenNour        |           30 |           51 | t
      42 | Le vent   | Brigitte Durand      |            1 |           12 | t
      43 | Le vent   | Dolores De Las Casas |           13 |           29 | t
(12 rows)

ameliegheerbrant=# select * from traducteur ;
    nom_traducteur    | tarifpage
----------------------+-----------
 Brigitte Durand      |    14.500
 Renaud Blanc         |    10.140
 Douglas Brown        |    12.133
 Yukiko Tanaka        |    25.450
 Salim BenNour        |    15.142
 Dolores De Las Casas |    10.020
(6 rows)
```

Jointures et agrégation dans les requêtes

Pas de problème avec JOIN ON :

```
ameliegheerbrant=# select * from traducteur join traduction on traduction.nom_traducteur=traducteur.nom_traducteur ;
    nom_traducteur    | tarifpage | travail | nom_livre |    nom_traducteur    | premierepage | dernierepage | fini
----------------------+-----------+---------+-----------+----------------------+--------------+--------------+------
 Brigitte Durand      |    14.500 |      14 | La pluie  | Brigitte Durand      |           10 |           33 | f
 Renaud Blanc         |    10.140 |      15 | La pluie  | Renaud Blanc         |           70 |           93 | t
 Brigitte Durand      |    14.500 |      16 | La pluie  | Brigitte Durand      |           94 |          150 | t
 Yukiko Tanaka        |    25.450 |      17 | La pluie  | Yukiko Tanaka        |            1 |            9 | f
 Salim BenNour        |    15.142 |      21 | La neige  | Salim BenNour        |          100 |          180 | f
 Brigitte Durand      |    14.500 |      22 | La neige  | Brigitte Durand      |           30 |          130 | f
 Yukiko Tanaka        |    25.450 |      31 | Le soleil | Yukiko Tanaka        |            1 |           40 | f
 Renaud Blanc         |    10.140 |      32 | Le soleil | Renaud Blanc         |          100 |          150 | f
 Douglas Brown        |    12.133 |      33 | Le soleil | Douglas Brown        |            1 |           10 | f
 Salim BenNour        |    15.142 |      41 | Le vent   | Salim BenNour        |           30 |           51 | t
 Brigitte Durand      |    14.500 |      42 | Le vent   | Brigitte Durand      |            1 |           12 | t
 Dolores De Las Casas |    10.020 |      43 | Le vent   | Dolores De Las Casas |           13 |           29 | t
(12 rows)
```

Jointures et agrégation dans les requêtes

Idem en utilisant un alias :

```
ameliegheerbrant=# select * from (traducteur join traduction on traduction.nom_traducteur=traducteur.nom_traducteur) as T
    nom_traducteur    | tarifpage | travail | nom_livre |    nom_traducteur    | premierepage | dernierepage | fini
----------------------+-----------+---------+-----------+----------------------+--------------+--------------+------
 Brigitte Durand      |    14.500 |      14 | La pluie  | Brigitte Durand      |           10 |           33 | f
 Renaud Blanc         |    10.140 |      15 | La pluie  | Renaud Blanc         |           70 |           93 | t
 Brigitte Durand      |    14.500 |      16 | La pluie  | Brigitte Durand      |           94 |          150 | t
 Yukiko Tanaka        |    25.450 |      17 | La pluie  | Yukiko Tanaka        |            1 |            9 | f
 Salim BenNour        |    15.142 |      21 | La neige  | Salim BenNour        |          100 |          180 | f
 Brigitte Durand      |    14.500 |      22 | La neige  | Brigitte Durand      |           30 |          130 | f
 Yukiko Tanaka        |    25.450 |      31 | Le soleil | Yukiko Tanaka        |            1 |           40 | f
 Renaud Blanc         |    10.140 |      32 | Le soleil | Renaud Blanc         |          100 |          150 | f
 Douglas Brown        |    12.133 |      33 | Le soleil | Douglas Brown        |            1 |           10 | f
 Salim BenNour        |    15.142 |      41 | Le vent   | Salim BenNour        |           30 |           51 | t
 Brigitte Durand      |    14.500 |      42 | Le vent   | Brigitte Durand      |            1 |           12 | t
 Dolores De Las Casas |    10.020 |      43 | Le vent   | Dolores De Las Casas |           13 |           29 | t
(12 rows)
```

Jointures et agrégation dans les requêtes

En revanche, dès que l'on essaie de retourner uniquement "la" colonne dédoublée, rien ne va plus :

```
ameliegheerbrant=# select nom_traducteur from (traducteur join traduction on traduction.nom_traducteur=traducteur.nom_traducteur) as T;
ERROR:  column reference "nom_traducteur" is ambiguous
LINE 1: select nom_traducteur from (traducteur join traduction on tr...
```

- Ne dérange pas trop Postgres... jusqu'à un certain point.

Jointures et agrégation dans les requêtes

N'améliore pas les choses :

```
ameliegheerbrant=# select traducteur.nom_traducteur from (traducteur join traduction on traduction.nom_traducteur=traducteur.nom_traducteur) as T;
ERROR:  invalid reference to FROM-clause entry for table "traducteur"
LINE 1: select traducteur.nom_traducteur from (traducteur join tradu...
               ^
HINT:  There is an entry for table "traducteur", but it cannot be referenced from this part of the query.
ameliegheerbrant=#
```

Jointures et agrégation dans les requêtes

Pas mieux :

```
ameliegheerbrant=# select traduction.nom_traducteur from (traducteur join traduction on traduction.nom_traducteur=traducteur.nom_traducteur) as T;
ERROR:  invalid reference to FROM-clause entry for table "traduction"
LINE 1: select traduction.nom_traducteur from (traducteur join tradu...
               ^
HINT:  There is an entry for table "traduction", but it cannot be referenced from this part of the query.
```

Jointures et agrégation dans les requêtes

Enfin :

```
ameliegheerbrant=# select traduction.nom_traducteur from (traducteur join traduction on traduction.nom_traducteur=traducteur.nom_traducteur);
     nom_traducteur
----------------------
 Brigitte Durand
 Renaud Blanc
 Brigitte Durand
 Yukiko Tanaka
 Salim BenNour
 Brigitte Durand
 Yukiko Tanaka
 Renaud Blanc
 Douglas Brown
 Salim BenNour
 Brigitte Durand
 Dolores De Las Casas
(12 rows)
```

utilisation du nom d'une des deux relations participant à la jointure

pas d'alias

Morale : pour comprendre l'implémentation de SQL propre à un système, il n'y a pas de secret, il faut connaître le standard et consulter la documentation propre à l'implémentation (qui n'est JAMAIS exhaustive), mais surtout faire des tests... La sémantique exacte de SQL est encore inconnue à ce jour et fait toujours l'objet de recherches.

Sous requête dans le SELECT

- Jusqu'à présent nous avons vu les sous requêtes dans le `WHERE`, et de façon limitée, dans le `FROM`.
- Mais elles peuvent apparaître partout !
- Exemple : une alternative au `GROUP BY`

```
SELECT DISTINCT S.cinema,
          (SELECT MAX(F.duree)
          FROM Film F
          WHERE F.titre=S.titre) as maxi
FROM Séance S;
```

- Pas conseillé, peu intuitif, préférer cet équivalent plus lisible :

```
SELECT cinema, MAX(duree)
FROM Film
GROUP BY cinema ;
```

Sous requête dans le WHERE

- Alternative (tordue !) au `HAVING` : sous requête dans le `WHERE`

```
SELECT DISTINCT S.cinema,
          (SELECT MAX(F.duree)
          FROM Film F
          WHERE F.titre=S.titre) as maxi
FROM Séance S
WHERE (SELECT COUNT(DISTINCT titre)
       FROM Film F1
       WHERE F1.titre IN (SELECT S1.titre
                          FROM Séance S1
                          WHERE S1.cinema=S.cinema))>5;
```

  restreint la requête précédente aux cinémas montrant 6 films ou plus.
- En général le nouveau standard est très libéral quant à l'usage des sous requêtes, bien qu'il y ait des variations d'un système à l'autre.

Sous requête dans le WHERE

- Préférer cet équivalent plus lisible :

```
SELECT cinema, MAX(duree)
FROM Film
GROUP BY cinema
HAVING COUNT(DISTINCT titre) > 5 ;
```

Pour chaque cinéma montrant au moins 6 films, retourne la durée du film le long plus passant dans ce cinéma.

Sous requête dans le FROM

Pour trouver le nombre de films maximum projeté par cinéma, on utilise une sous requête qui calcule le nombre de films projetés par chaque cinéma :

```
SELECT COUNT(DISTINCT titre)
FROM Film NATURAL JOIN Séance
GROUP BY cinema;
```

Requête complète :

```
SELECT MAX(nbre)
FROM (SELECT COUNT(DISTINCT titre) as nbre
  FROM Film NATURAL JOIN Séance
  GROUP BY cinema) AS S;
```

Remarque : il est obligatoire de donner un nom à la sous requête lorsqu'elle est dans le FROM, même si on ne s'en sert pas.

Sous requête dans le FROM

A partir de la durée maximale des films réalisés par chaque réalisateur, calculer la moyenne de ces valeurs maximales :

```
SELECT MAX(duree)
FROM Film
GROUP BY realisateur ;
```

Requête complète :

```
SELECT AVG(duree)
FROM
(SELECT MAX(duree) as duree
FROM Film
GROUP BY realisateur)) AS foo ;
```

Remarque : sans `as duree` la requête ne serait pas correcte (pas de colonne `duree` dans le tableau résultat de la sous-requête...)

Une fonctionnalité utile : ordonner la réponse

- `SELECT * FROM S;`

```
A1 A3
-- -----------
a             5
a             7
b             3
```

- `SELECT * FROM S ORDER BY A3;`

```
A1 A3
-- -----------
b             3
a             5
a             7
```

Une fonctionnalité utile : ordonner la réponse

- Ordre décroissant :

```
SELECT * FROM S ORDER BY A3 DESC;
```

```
A1 A3
-- -----------
a            7
a            5
b            3
```

- Ordre sur plusieurs attributs :

```
SELECT * FROM S ORDER BY A1, A3;
```

```
A1 A3
-- -----------
a            5
a            7
b            3
```

Une fonctionnalité utile : ordonner la réponse

On peut également trier au moyen d'opérations

R	A1	A2	A3
	5	1	b
	1	4	g
	2	1	e

```
SELECT * FROM R ORDER BY A1+A2;
```

A1	A2	A3
2	1	e
1	4	g
5	1	b

Une fonctionnalité utile : ordonner la réponse

On peut également trier au moyen d'opérations

R	A1	A2	A3
	5	1	b
	1	4	g
	2	1	e

Même si les attributs sur lesquels sont effectués les opérations ne font pas partie de la réponse

```
SELECT A3 FROM R ORDER BY A1+A2;
```

A3
e
g
b

Une fonctionnalité utile : ordonner la réponse

On peut également tronquer la réponse

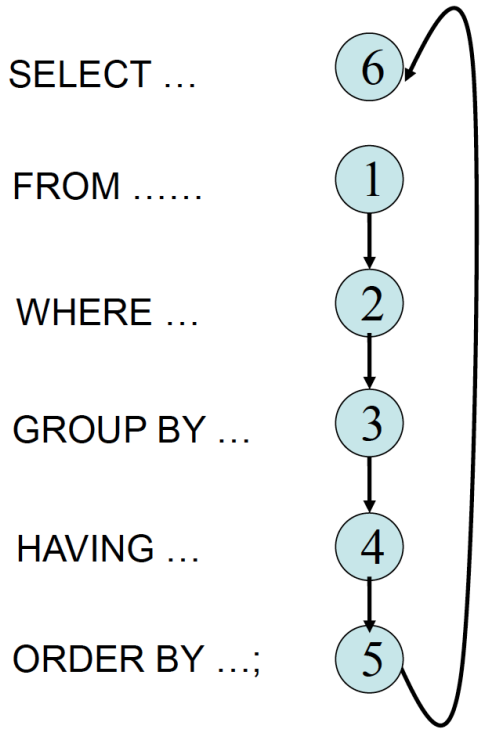
R	A1	A2	A3
	5	1	b
	1	4	g
	2	1	e

```
SELECT A3 FROM R ORDER BY A1+A2 LIMIT 2;
```

A3
e
g

(Attention aux cas où le résultat n'est pas ordonné de manière unique par `ORDER BY`.)

Ordre d'exécution des clauses



Attention : l'ordre correct d'écriture des clauses dans la requête correspond à cet ordre d'exécution !
(E.g., on ne met jamais le `order by` avant le `group by` ou le `from`.)

Retour sur la sémantique des requêtes imbriquées

Sous requête corrélée

- Les cinémas qui passent tous les films.

```
SELECT S.cinema
FROM Séance S
WHERE NOT EXISTS
     (SELECT F1.titre FROM FILM F1
      WHERE F1.titre NOT IN
              (SELECT F2.titre
               FROM Film F2 NATURAL JOIN Séance S2
               WHERE S2.cinéma=S.cinéma));
```

Sémantique : les sous requête la plus imbriquée est corrélée.

```
SELECT S.cinema
FROM Séance S
WHERE NOT EXISTS
    (SELECT F1.titre FROM FILM F1
     WHERE F1.titre NOT IN
           (SELECT F2.titre
                 FROM Film F2 NATURAL JOIN Séance S2
                 WHERE S2.cinéma=S.cinéma));
```

On ne peut donc pas commencer par calculer le résultat de la réponse à cette sous requête sans regarder le reste.

```
SELECT S.cinema
FROM Séance S
WHERE NOT EXISTS
    (SELECT F1.titre FROM FILM F1
     WHERE F1.titre NOT IN
            (SELECT F2.titre
             FROM Film F2 NATURAL JOIN Séance S2
             WHERE S2.cinéma=S.cinéma));
```

On parcourt d'abord tous les tuples de la table Séance :

```
for S.cinéma in Séance S do :


    SELECT F1.titre FROM FILM F1
    WHERE F1.titre NOT IN
            (SELECT F2.titre
             FROM Film F2 NATURAL JOIN Séance S2
             WHERE S2.cinéma=S.cinéma)  =  ∅ ?
```

Pour chaque S.cinéma in Séance S, calculer le résultat de :

```
SELECT F1.titre FROM FILM F1
WHERE F1.titre NOT IN
        (SELECT F2.titre
          FROM Film F2 NATURAL JOIN Séance S2
         WHERE S2.cinéma=S.cinéma)
```

Il y a donc une jointure à faire entre Film F2 et Séance S2 :

réalisateur	titre	durée	cinéma
Varda	Les Créatures	92	Le Louxor
Coppola	Virgin Suicides	96	MK2 Quai de Seine
Coppola	Lost inTranslation	79	Le Louxor
Coppola	Lost inTranslation	102	MK2 Quai de Seine
Varda	Cléo de 5 à 7	90	Le Louxor
Coppola	Virgin Suicides	96	Le Louxor

Pour chaque S.cinéma in Séance S, calculer le résultat de :

```
SELECT F1.titre FROM FILM F1
WHERE F1.titre NOT IN
        (SELECT F2.titre
         FROM Film F2 NATURAL JOIN Séance S2
         WHERE S2.cinéma=S.cinéma)
```

Mais que faire de la condition `S2.cinéma=S.cinéma` ?

réalisateur	titre	durée	cinéma
Varda	Les Créatures	92	Le Louxor
Coppola	Virgin Suicides	96	MK2 Quai de Seine
Coppola	Lost inTranslation	79	Le Louxor
Coppola	Lost inTranslation	102	MK2 Quai de Seine
Varda	Cléo de 5 à 7	90	Le Louxor
Coppola	Virgin Suicides	96	Le Louxor

```
SELECT F1.titre FROM FILM F1
WHERE F1.titre NOT IN
        (SELECT F2.titre
         FROM Film F2 NATURAL JOIN Séance S2
         WHERE S2.cinéma=S.cinéma)
```

Il faut examiner chaque S.cinéma dans Séance S, par exemple :

réalisateur	titre	durée
Varda	Les Créatures	92
Coppola	Virgin Suicides	96
Coppola	Lost in Translation	102
Varda	Cléo de 5 à 7	90

cinéma	titre
Le Louxor	Les Créatures
MK2 Quai de Seine	Virgin Suicides
MK2 Quai de Seine	Lost in Translation
Le Louxor	Cléo de 5 à 7
Le Louxor	Lost in Translation
Le Louxor	Virgin Suicides

```
SELECT F1.titre FROM FILM F1
WHERE F1.titre NOT IN
        (SELECT F2.titre
         FROM Film F2 NATURAL JOIN Séance S2
         WHERE S2.cinéma=S.cinéma)
```

Pour S.cinéma = Le Louxor on obtient :

réalisateur	titre	durée	cinéma
Varda	Les Créatures	92	Le Louxor
~~Coppola~~	~~Virgin Suicides~~	~~96~~	~~MK2 Quai de Seine~~
Coppola	Lost inTranslation	79	Le Louxor
~~Coppola~~	~~Lost inTranslation~~	~~102~~	~~MK2 Quai de Seine~~
Varda	Cléo de 5 à 7	90	Le Louxor
Coppola	Virgin Suicides	96	Le Louxor

```
SELECT F1.titre FROM FILM F1
WHERE F1.titre NOT IN
        (SELECT F2.titre
         FROM Film F2 NATURAL JOIN Séance S2
         WHERE S2.cinéma=Le Louxor)
```

Pour S.cinéma = Le Louxor on obtient :

réalisateur	titre	durée	cinéma
Varda	Les Créatures	92	Le Louxor
~~Coppola~~	~~Virgin Suicides~~	~~96~~	~~MK2 Quai de Seine~~
Coppola	Lost inTranslation	79	Le Louxor
~~Coppola~~	~~Lost inTranslation~~	~~102~~	~~MK2 Quai de Seine~~
Varda	Cléo de 5 à 7	90	Le Louxor
Coppola	Virgin Suicides	96	Le Louxor

```
SELECT F1.titre FROM FILM F1
WHERE F1.titre NOT IN
        (SELECT F2.titre
         FROM Film F2 NATURAL JOIN Séance S2
         WHERE S2.cinéma=Le Louxor)
```

réalisateur	titre	durée	cinéma
Varda	Les Créatures	92	Le Louxor
Coppola	Lost inTranslation	79	Le Louxor
Varda	Cléo de 5 à 7	90	Le Louxor
Coppola	Virgin Suicides	96	Le Louxor

$\Rightarrow$ tableau résultat de la sous requête pour S.cinéma = Le Louxor

Pour vérifier si Le Louxor appartient à la réponse, il ne nous reste plus qu'à évaluer la requête principale :

```
SELECT F1.titre FROM FILM F1
WHERE F1.titre NOT IN
        (SELECT F2.titre
         FROM Film F2 NATURAL JOIN Séance S2
         WHERE S2.cinéma=Le Louxor)
```

réalisateur	titre	durée	cinéma
Varda	Les Créatures	92	Le Louxor
Coppola	Lost inTranslation	79	Le Louxor
Varda	Cléo de 5 à 7	90	Le Louxor
Coppola	Virgin Suicides	96	Le Louxor

réalisateur	titre	durée
Varda	Les Créatures	92
Coppola	Virgin Suicides	96
Coppola	Lost inTranslation	102
Varda	Cléo de 5 à 7	90

Est-ce qu'il y a un élément qui appartient au premier ensemble mais pas au second ?

réalisateur	titre	durée	cinéma
Varda	Les Créatures	92	Le Louxor
Coppola	Lost inTranslation	79	Le Louxor
Varda	Cléo de 5 à 7	90	Le Louxor
Coppola	Virgin Suicides	96	Le Louxor

```
SELECT F1.titre FROM FILM F1
WHERE F1.titre NOT IN
        (SELECT F2.titre
         FROM Film F2 NATURAL JOIN Séance S2
         WHERE S2.cinéma=Le Louxor)
```

réalisateur	titre	durée
Varda	Les Créatures	92
Coppola	Virgin Suicides	96
Coppola	Lost inTranslation	102
Varda	Cléo de 5 à 7	90

Est-ce qu'il y a un élément qui appartient au premier ensemble mais pas au second ?

réalisateur	titre	durée	cinéma
Varda	Les Créatures	92	Le Louxor
Coppola	Lost inTranslation	79	Le Louxor
Varda	Cléo de 5 à 7	90	Le Louxor
Coppola	Virgin Suicides	96	Le Louxor

```
SELECT F1.titre FROM FILM F1
WHERE F1.titre NOT IN
        (SELECT F2.titre
         FROM Film F2 NATURAL JOIN Séance S2
         WHERE S2.cinéma=Le Louxor)
```

réalisateur	titre	durée
Varda	Les Créatures	92
Coppola	Virgin Suicides	96
Coppola	Lost in Translation	102
Varda	Cléo de 5 à 7	90

Est-ce qu'il y a un élément qui appartient au premier ensemble mais pas au second ?

réalisateur	titre	durée	cinéma
Varda	Les Créatures	92	Le Louxor
Coppola	Lost in Translation	79	Le Louxor
Varda	Cléo de 5 à 7	90	Le Louxor
Coppola	Virgin Suicides	96	Le Louxor

```
SELECT F1.titre FROM FILM F1
WHERE F1.titre NOT IN
        (SELECT F2.titre
         FROM Film F2 NATURAL JOIN Séance S2
         WHERE S2.cinéma=Le Louxor)
```

réalisateur	titre	durée
Varda	Les Créatures	92
Coppola	Virgin Suicides	96
Coppola	Lost in Translation	102
Varda	Cléo de 5 à 7	90

Est-ce qu'il y a un élément qui appartient au premier ensemble mais pas au second ?

réalisateur	titre	durée	cinéma
Varda	Les Créatures	92	Le Louxor
Coppola	Lost in Translation	79	Le Louxor
Varda	Cléo de 5 à 7	90	Le Louxor
Coppola	Virgin Suicides	96	Le Louxor

```
SELECT F1.titre FROM FILM F1
WHERE F1.titre NOT IN
        (SELECT F2.titre
         FROM Film F2 NATURAL JOIN Séance S2
         WHERE S2.cinéma=Le Louxor)
```

réalisateur	titre	durée
Varda	Les Créatures	92
Coppola	Virgin Suicides	96
Coppola	Lost in Translation	102
Varda	Cléo de 5 à 7	90

Est-ce qu'il y a un élément qui appartient au premier ensemble mais pas au second ?

réalisateur	titre	durée	cinéma
Varda	Les Créatures	92	Le Louxor
Coppola	Lost in Translation	79	Le Louxor
Varda	Cléo de 5 à 7	90	Le Louxor
Coppola	Virgin Suicides	96	Le Louxor

```
SELECT F1.titre FROM FILM F1
WHERE F1.titre NOT IN
        (SELECT F2.titre
         FROM Film F2 NATURAL JOIN Séance S2
         WHERE S2.cinéma=Le Louxor)
```

réalisateur	titre	durée
Varda	Les Créatures	92
Coppola	Virgin Suicides	96
Coppola	Lost in Translation	102
Varda	Cléo de 5 à 7	90

Est-ce qu'il y a un élément qui appartient au premier ensemble mais pas au second ?

Non ! Le Louxor appartient donc à la réponse.

réalisateur	titre	durée	cinéma
Varda	Les Créatures	92	Le Louxor
Coppola	Lost in Translation	79	Le Louxor
Varda	Cléo de 5 à 7	90	Le Louxor
Coppola	Virgin Suicides	96	Le Louxor

```
SELECT F1.titre FROM FILM F1
WHERE F1.titre NOT IN
        (SELECT F2.titre
         FROM Film F2 NATURAL JOIN Séance S2
         WHERE S2.cinéma=Le Louxor)
```

Pour chaque S.cinéma in Séance S, calculer le résultat de :

```
SELECT F1.titre FROM FILM F1
WHERE F1.titre NOT IN
        (SELECT F2.titre
         FROM Film F2 NATURAL JOIN Séance S2
         WHERE S2.cinéma=S.cinéma)
```

Ce n'est pas fini, il reste encore MK2 Quai de Seine :

réalisateur	titre	durée	cinéma
Varda	Les Créatures	92	Le Louxor
Coppola	Virgin Suicides	96	MK2 Quai de Seine
Coppola	Lost inTranslation	79	Le Louxor
Coppola	Lost inTranslation	102	MK2 Quai de Seine
Varda	Cléo de 5 à 7	90	Le Louxor
Coppola	Virgin Suicides	96	Le Louxor

Pour chaque S.cinéma in Séance S, calculer le résultat de :

```
SELECT F1.titre FROM FILM F1
WHERE F1.titre NOT IN
        (SELECT F2.titre
         FROM Film F2 NATURAL JOIN Séance S2
         WHERE S2.cinéma=S.cinéma)
```

réalisateur	titre	durée	cinéma
Varda	Les Créatures	92	Le Louxor
Coppola	Virgin Suicides	96	MK2 Quai de Seine
Coppola	Lost inTranslation	79	Le Louxor
Coppola	Lost inTranslation	102	MK2 Quai de Seine
Varda	Cléo de 5 à 7	90	Le Louxor
Coppola	Virgin Suicides	96	Le Louxor

```
SELECT F1.titre FROM FILM F1
WHERE F1.titre NOT IN
        (SELECT F2.titre
         FROM Film F2 NATURAL JOIN Séance S2
         WHERE S2.cinéma=MK2 Quai de Seine)
```

Pour S.cinéma = MK2 Quai de Seine on obtient :

réalisateur	titre	durée	cinéma
Varda	Les Créatures	92	Le Louxor
Coppola	Virgin Suicides	96	MK2 Quai de Seine
Coppola	Lost in Translation	79	Le Louxor
Coppola	Lost in Translation	102	MK2 Quai de Seine
Varda	Cléo de 5 à 7	90	Le Louxor
Coppola	Virgin Suicides	96	Le Louxor

```
SELECT F1.titre FROM FILM F1
WHERE F1.titre NOT IN
        (SELECT F2.titre
         FROM Film F2 NATURAL JOIN Séance S2
         WHERE S2.cinéma=MK2 Quai de Seine)
```

réalisateur	titre	durée	cinéma
Coppola	Virgin Suicides	96	MK2 Quai de Seine
Coppola	Lost inTranslation	102	MK2 Quai de Seine

⇒ tableau résultat de la sous requête pour S.cinéma = MK2 Quai de Seine

Pour vérifier si MK2 Quai de Seine appartient à la réponse, il ne nous reste plus qu'à évaluer la requête principale :

```
SELECT F1.titre FROM FILM F1
WHERE F1.titre NOT IN
        (SELECT F2.titre
         FROM Film F2 NATURAL JOIN Séance S2
         WHERE S2.cinéma=MK2 Quai de Seine)
```

réalisateur	titre	durée	cinéma
Coppola	Virgin Suicides	96	MK2 Quai de Seine
Coppola	Lost in Translation	102	MK2 Quai de Seine

réalisateur	titre	durée
Varda	Les Créatures	92
Coppola	Virgin Suicides	96
Coppola	Lost in Translation	102
Varda	Cléo de 5 à 7	90

Est-ce qu'il y a un élément qui appartient au premier ensemble mais pas au second ?

réalisateur	titre	durée	cinéma
Coppola	Virgin Suicides	96	MK2 Quai de Seine
Coppola	Lost in Translation	102	MK2 Quai de Seine

```
SELECT F1.titre FROM FILM F1
WHERE F1.titre NOT IN
        (SELECT F2.titre
         FROM Film F2 NATURAL JOIN Séance S2
         WHERE S2.cinéma=MK2 Quai de Seine)
```

réalisateur	titre	durée
Varda	Les Créatures	92
Coppola	Virgin Suicides	96
Coppola	Lost in Translation	102
Varda	Cléo de 5 à 7	90

Est-ce qu'il y a un élément qui appartient au premier ensemble mais pas au second ?

réalisateur	titre	durée	cinéma
Coppola	Virgin Suicides	96	MK2 Quai de Seine
Coppola	Lost in Translation	102	MK2 Quai de Seine

```
SELECT F1.titre FROM FILM F1
WHERE F1.titre NOT IN
        (SELECT F2.titre
         FROM Film F2 NATURAL JOIN Séance S2
         AWHERE S2.cinéma=MK2 Quai de Seine)
```

réalisateur	titre	durée
Varda	Les Créatures	92
Coppola	Virgin Suicides	96
Coppola	Lost inTranslation	102
Varda	Cléo de 5 à 7	90

Est-ce qu'il y a un élément qui appartient au premier ensemble mais pas au second ?

réalisateur	titre	durée	cinéma
Coppola	Virgin Suicides	96	MK2 Quai de Seine
Coppola	Lost inTranslation	102	MK2 Quai de Seine

```
SELECT F1.titre FROM FILM F1
WHERE F1.titre NOT IN
        (SELECT F2.titre
         FROM Film F2 NATURAL JOIN Séance S2
         WHERE S2.cinéma=MK2 Quai de Seine)
```

réalisateur	titre	durée
Varda	Les Créatures	92
Coppola	Virgin Suicides	96
Coppola	Lost in Translation	102
Varda	Cléo de 5 à 7	90

Est-ce qu'il y a un élément qui appartient au premier ensemble mais pas au second ?

Oui !

réalisateur	titre	durée	cinéma
Coppola	Virgin Suicides	96	MK2 Quai de Seine
Coppola	Lost in Translation	102	MK2 Quai de Seine

```
SELECT F1.titre FROM FILM F1
WHERE F1.titre NOT IN
        (SELECT F2.titre
         FROM Film F2 NATURAL JOIN Séance S2
         WHERE S2.cinéma=MK2 Quai de Seine)
```

réalisateur	titre	durée
Varda	Les Créatures	92
Coppola	Virgin Suicides	96
Coppola	Lost in Translation	102
Varda	Cléo de 5 à 7	90

Est-ce qu'il y a un élément qui appartient au premier ensemble mais pas au second ?

Il y en a même deux ...

réalisateur	titre	durée	cinéma
Coppola	Virgin Suicides	96	MK2 Quai de Seine
Coppola	Lost in Translation	102	MK2 Quai de Seine

```
SELECT F1.titre FROM FILM F1
WHERE F1.titre NOT IN
        (SELECT F2.titre
         FROM Film F2 NATURAL JOIN Séance S2
         WHERE S2.cinéma=MK2 Quai de Seine)
```

réalisateur	titre	durée
Varda	Les Créatures	92
Coppola	Virgin Suicides	96
Coppola	Lost in Translation	102
Varda	Cléo de 5 à 7	90

Est-ce qu'il y a un élément qui appartient au premier ensemble mais pas au second ?

Oui : MK2 Quai de Seine n'appartient pas à la réponse

réalisateur	titre	durée	cinéma
Coppola	Virgin Suicides	96	MK2 Quai de Seine
Coppola	Lost in Translation	102	MK2 Quai de Seine

```
SELECT F1.titre FROM FILM F1
WHERE F1.titre NOT IN
        (SELECT F2.titre
         FROM Film F2 NATURAL JOIN Séance S2
         WHERE S2.cinéma=MK2 Quai de Seine)
```

C'est fini ? On a calculé notre résultat ?
La réponse est donc Le Louxor ?

```
SELECT S.cinema
FROM Séance S
WHERE NOT EXISTS
    (SELECT F1.titre FROM FILM F1
     WHERE F1.titre NOT IN
           (SELECT F2.titre
                 FROM Film F2 NATURAL JOIN Séance S2
                 WHERE S2.cinéma=S.cinéma));
```

Non. Pas exactement.
En fait SQL examine un à un tous les tuples de la table Séance :

cinéma	titre
Le Louxor	Les Créatures
MK2 Quai de Seine	Virgin Suicides
MK2 Quai de Seine	Lost in Translation
Le Louxor	Cléo de 5 à 7
Le Louxor	Lost in Translation
Le Louxor	Virgin Suicides

Le Louxor sera donc bien DANS la réponse, mais en 4 exemplaires.

Pour obtenir comme réponse Le Louxor sans doublon il faut écrire :

```
SELECT DISTINCT S.cinema
FROM Séance S
WHERE NOT EXISTS
     (SELECT F1.titre FROM FILM F1
      WHERE F1.titre NOT IN
            (SELECT F2.titre
                  FROM Film F2 NATURAL JOIN Séance S2
                  WHERE S2.cinéma=S.cinéma)));
```

Conditions de totalité avec les agrégats

Pour qu'un cinéma appartienne à la requête il faut :

- qu'il n'existe pas de film qui ne passe pas dans ce cinéma

Ou, maintenant qu'on sait grouper les tuples et compter :

- que le nombre de films différents soit égal au nombre de films différents qui passent dans ce cinéma

Conditions de totalité avec les agrégats

A ⊆ B et A = B peuvent être exprimés sans negation si on utilise les agrégats

A ⊆ B

équivalent à la condition

|A ∩ B| = |A|

(nombre d'elements à la fois dans A et dans B = nombre d'elements dans A)

$$|A| = |B| \text{ ssi } |A| \subseteq |B| \text{ et } |B| \subseteq |A|$$

Conditions de totalité avec les agrégats

Formulation alternative plus simple

- Les cinémas qui passent tous les films.

```
SELECT cinema
FROM Séance
GROUP BY cinéma
HAVING count(DISTINCT titre)
            =
  (SELECT count(DISTINCT titre) FROM FILM);
```

- Technique générale : représentation d'une clause universelle ("pour tout") en utilisant l'agrégation

Conditions de totalité avec les agrégats

Sémantique opérationnelle

```
SELECT cinema
FROM Séance
GROUP BY cinéma
HAVING count(DISTINCT titre)
           =
  (SELECT count(DISTINCT titre) FROM FILM);
```

On calcule d'abord le résultat de la sous requête (non corrélée) :

	réalisateur	titre	durée
1.	Varda	Les Créatures	92
2.	Coppola	Virgin Suicides	96
3.	Coppola	Lost inTranslation	102
4.	Varda	Cléo de 5 à 7	90

= 4

Conditions de totalité avec les agrégats

Puis on calcule le résultat de la requête principale :

```
SELECT cinema
FROM Séance
GROUP BY cinéma
HAVING count(DISTINCT titre)
           =
  (SELECT count(DISTINCT titre) FROM FILM);
```

On commence par regrouper par cinémas :

cinéma	titre
MK2 Quai de Seine	Lost in Translation
MK2 Quai de Seine	Virgin Suicides
Le Louxor	Les Créatures
Le Louxor	Cléo de 5 à 7
Le Louxor	Lost in Translation
Le Louxor	Virgin Suicides

Conditions de totalité avec les agrégats

```
SELECT cinema
FROM Séance
GROUP BY cinéma
HAVING count(DISTINCT titre)
           =
  (SELECT count(DISTINCT titre) FROM FILM);
```

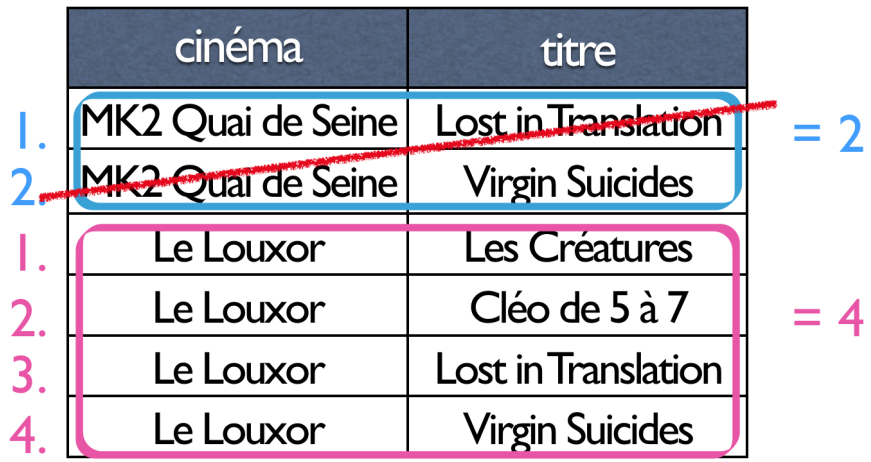
On calcule les agrégats :

	cinéma	titre	
1.	MK2 Quai de Seine	Lost in Translation	= 2
2.	MK2 Quai de Seine	Virgin Suicides	
1.	Le Louxor	Les Créatures	= 4
2.	Le Louxor	Cléo de 5 à 7	
3.	Le Louxor	Lost in Translation	
4.	Le Louxor	Virgin Suicides	

Conditions de totalité avec les agrégats

```
SELECT cinema
FROM Séance
GROUP BY cinéma
HAVING count(DISTINCT titre)
           =
  (SELECT count(DISTINCT titre) FROM FILM);
```

On opère la condition de sélection du `HAVING` :



Conditions de totalité avec les agrégats

```
SELECT cinema
FROM Séance
GROUP BY cinéma
HAVING count(DISTINCT titre)
           =
  (SELECT count(DISTINCT titre) FROM FILM);
```

Enfin, on ne conserve que les cinémas ayant passé le test :

	cinéma	titre	
1.	MK2 Quai de Seine	Lost in Translation	= 2
2.	MK2 Quai de Seine	Virgin Suicides	
1.	Le Louxor	Les Créatures	
2.	Le Louxor	Cléo de 5 à 7	= 4
3.	Le Louxor	Lost in Translation	
4.	Le Louxor	Virgin Suicides	

Remarque : un seul représentant par groupe (pas de doublons)

Conditions de totalité avec les agrégats

Attention : plus vraiment de sens sans DISTINCT

- Les cinémas qui ont autant de séances qu'il existe de films ? !

```
SELECT cinema
FROM Séance
GROUP BY cinéma
HAVING count(titre)
           =
  (SELECT count(titre) FROM FILM);
```

- Pour peu que la table Seance contienne des attributs supplémentaires tels que `jour` et `heure`, un cinéma qui passerait un unique film lors de nombreuses séances pourrait très bien être dans la réponse...

Remarque : DISTINCT peut être omis si l'attribut est clef primaire d'une table mentionnée toute seule dans le FROM (attention aux jointures générant des doublons !).

Conditions de totalité avec les agrégats

- Les cinémas qui passent tous les films de Varda.

```
SELECT S.cinema
FROM Séance NATURAL JOIN Film
WHERE realisateur='Varda'
GROUP BY cinema
HAVING count(DISTINCT titre)
            =
  (SELECT count(DISTINCT titre) FROM FILM
  WHERE realisateur='Varda' );
```

- Les cinémas pour lesquels : nombre de films de Varda qui passent dans le cinéma = nombre de films de Varda

Conditions de totalité avec les agrégats

Les cinémas qui passent tous les films de Varda et aucun autre.

```
SELECT cinema
FROM (Séance NATURAL JOIN Film) as J
WHERE realisateur='Varda'
GROUP BY cinema
HAVING count(DISTINCT titre)
           =
  (SELECT count(DISTINCT titre) FROM FILM
  WHERE realisateur='Varda' )
AND count(DISTINCT titre)
           =
 (SELECT count(DISTINCT titre) FROM Seance S
  WHERE J.cinema=S.cinema ) ;
```

Les cinémas pour lesquels : nombre de films de Varda qui passent dans le cinéma = nombre de films de Varda = nombre de films qui passent dans le cinéma