Bases de Données

Amélie Gheerbrant

Université de Paris
UFR Informatique
Institut de Recherche en Informatique Fondamentale

amelie@irif.fr

4 février 2021

Le Modèle Relationnel

- Les données sont organisées dans des relations (tables)
- Schéma de bases de données relationnelles
  - ensemble de noms de tables
  - liste d'attributs pour chaque table
- Les tables sont spécifiées sous la forme :

  `<nom de la table> : <liste d'attributs>`
- Exemples :

  `Compte: numero, agence, clientId`

  `Film: titre, réalisateur, acteur`
- Dans une même table les attributs ont des noms différents
- Les tables ont des noms différents

Exemple de base de données relationnelle

Film	Titre	Réalisateur	Acteur
	Shining	Kubrick	Nicholson
	The Player	Altman	Robbins
	Chinatown	Polanski	Nicholson
	Chinatown	Polanski	Polanski
	Repulsion	Polanski	Deneuve

Séance	Cinéma	Titre
	Le Champo	Shining
	Le Champo	Chinatown
	Le Champo	The Player
	Odéon	Chinatown

Exemples de requêtes

- Trouver le nom des films qui passent en ce moment :

réponse	titre
	Shining
	The Player
	Chinatown

- Trouver les cinémas qui passent des films de Polanski :

réponse	cinéma
	Le Champo
	Odéon

- Trouver les cinémas qui passent des films avec Nicholson :

réponse	cinéma
	Le Champo
	Odéon

- Trouver tous les réalisateurs qui se sont dirigés eux-mêmes :

réponse	réalisateur
	Polanski

- Trouver tous les réalisateurs dont les films sont joués dans tous les cinémas :

réponse	cinéma
	Polanski

- Trouver tous les cinémas qui ne passent que des films avec Nicholson :

réponse	cinéma

mais si le Champo cesse de passer 'The Player', la réponse devient :

réponse	cinéma
	Le Champo

Les Résultats des Requêtes

- Ce sont des tables construites à partir d'autres tables de la base de données

Comment formuler une requête ?

- Deux types de langages de requêtes :
  - Commercial : SQL
  - Théorique : le Calcul Relationnel, l'Algèbre Relationnelle, Datalog, etc

Déclaratif versus Procédural

Déclaratif :

$$\{\text{titre} \mid \quad (\text{titre, réalisateur, acteur}) \in \text{Film}\}$$

```
SELECT Titre
FROM FILM ;
```

Procédural :

```
for each tuple (t,r,a) in relation Film do
        output t
    end
```

Déclaratif versus Procédural

- **Langages théoriques :**
  - Déclaratif : le Calcul Relationnel, les requêtes basées sur des règles
  - Procédural : l'Algèbre Relationnelle
- **Langages utilisés en pratique :** mélange des deux, mais surtout déclaratif

Exemples de requêtes

- Trouver le nom des films présents dans la base de données :

  `réponse(ti) :- film(ti, real, act)`

  se lit,

  le tuple `(ti)` est dans la table réponse,
  SI le tuple `(ti, real, act)` est dans la table film,

- Formulation des requêtes comme règles indiquant quand certains éléments appartiennent à la réponse.

Exemples de requêtes

- Trouver le nom des films présents dans la base de données :

  `réponse(ti) :- film(ti, real, act)`

  i.e.,

  while tuple `(ti, real, act)` dans la relation film,
  output `ti` (valeur de l'attribut titre)

- Formulation des requêtes comme règles indiquant quand certains éléments appartiennent à la réponse.

- On appelle ça des requêtes conjonctives

Autre exemple

- Trouver les cinémas qui passent des films de Polanski :

`réponse(ci) :- Film(ti, 'Polanski', act), Séance(ci, ti)`

i.e.,

while `(ti, real, act)` dans la relation Film,
tester : `real='Polanski'` ? ;
si oui, considérer tous les tuples `(ci,ti)` dans Séance,
et pour chacun, *output* `ci` (valeur de l'attribut cinema,)
sinon, considérer le tuple suivant

= type de requête le plus répandu.

- Trouver les réalisateurs qui se sont dirigés eux-mêmes :

```
réponse(real) :- film(ti, real, act), real=act
```

i.e.,

while `(ti, real, act)` dans la relation film,
tester : `real=act?` ;
si oui, `output real`
sinon, considérer le tuple suivant.

Un exemple plus compliqué

Trouver les réalisateurs dont les films passent dans tous les cinémas...

- "Tous" : souvent problématique.
- Besoin du quantificateur universel $\forall$

$$\{\text{real} \mid \forall\ (\text{ci}, \text{ti}) \in \text{Séance}, \exists\ (\text{ti'}, \text{act}) : (\text{ti'}, \text{real}, \text{act}) \in \text{Film} \wedge (\text{ci}, \text{ti'}) \in \text{Séance}\ \}$$

pour tester si `real` $\in$ réponse, pour tout nom de cinéma `ci`, tester s'il existe un tuple `(ti', real, act)` dans Film et un tuple (ci, ti') dans Séance.

Notation de la logique mathématique :

- $\forall$ signifie "pour tout", $\exists$ signifie "il existe"
- $\wedge$ est une conjonction (ET logique)

SQL, les raisons du succès

- SQL = *Structured Query Language* (IBM fin 1970')
- Standards : SQL-86, SQL-92, SQL-99 / SQLL3 (+1000 pages)
- Requêtes basées sur le modèle relationnel : langage logique, simple et compréhensible.
- Une requête du calcul peut être facilement traduite en une expression de l'algèbre qui s'évalue simplement (Théorème de Codd)
- Algèbre relationnelle = modèle limité de calcul (n'autorise pas les fonctions arbitraires). Autorise l'optimisation de l'évaluation des expressions algébriques.
- Parallélisme facilité pour les très grandes bases de données.

Exemples de requêtes SQL

- Trouver le nom des films projetés en ce moment :

```
SELECT Titre
FROM FILM ;
```

- `SELECT` liste les attributs retournés par la requête
- `FROM` liste les relations prises en entrée

Plus d'exemples

- Trouver les cinémas qui passent des films de Polanski :

```
SELECT Séance.cinema
FROM Séance, Film
WHERE Film.titre = Séance.titre
AND Film.réalisateur = 'Polanski' ;
```

- SELECT spécifie maintenant de quelles relations viennent les attributs - parce qu'on en utilise plus d'une
- FROM liste deux relations
- WHERE spécifie les conditions de sélection des tuples

Jointures de relations

- `WHERE` autorise à faire la jointure de plusieurs relations.

Requête : lister les réalisateurs avec les cinémas dans lesquels on peut voir leurs films

- Requête conjonctive :

```
réponse(real, ci) :- Séance(ci, ti), Film(ti, real, act)
```

- Requête SQL :

```
SELECT Film.réalisateur, Séance.cinema
FROM Séance, Film
WHERE Film.titre = Séance.titre ;
```

Jointures de relations

- ```
  SELECT Film.réalisateur, Séance.cinema
  FROM Séance, Film
  WHERE Film.titre = Séance.titre ;
  ```

- Sémantique : boucles imbriquées sur les relations listées dans le `FROM`

```
for each tuple (titre1, réalisateur, acteur) in Film do
     for each tuple (cinema, titre2) in Séance do
           if titre1=titre2 then output (réalisateur, cinema)
     end
end
```

- Cette opération s'appelle une jointure : une des opérations les plus fondamentales en BD.
```

Un langage procédural : l'algèbre relationnelle

- Commençons par un sous-ensemble de l'algèbre relationnelle qui suffit à capturer les requêtes basées sur les règles simples et les énoncés SQL simples de la forme `SELECT-FROM-WHERE`.

- Ce sous-ensemble a trois opérations :

  Projection π
  Sélection σ
  Produit Cartésien $\times$

- Parfois on utilise aussi le renommage ρ, mais on peut l'éviter sous certaines conditions.

La Projection

- Choisit des attributs dans une relation.
- $\pi_{A_1,\ldots,A_n}(R)$: conserve uniquement les attributs $A_1, \ldots, A_n$ dans la relation R
- Exemple

$\pi_{titre,realisateur}$

titre	réalisateur	acteur
Shining	Kubrick	Nicholson
The Player	Altman	Robins
Chinatown	Polanski	Nicholson
Chinatown	Polanski	Polanski
Repulsion	Polanski	Deneuve

=

titre	réalisateur
Shining	Kubrick
The Player	Altman
Chinatown	Polanski
Repulsion	Polanski

- Fournit à l'utilisateur une vue des données en omettant certains attributs

La sélection

- Choisit des tuples satisfaisant certaines conditions.
- $\sigma_{cond}(R)$: conserve uniquement les tuples t pour lesquels la condition $cond(t)$ est vraie.
- Conditions : conjonctions de
  - `R.A = S.B` deux attributs ont la même valeur
  - `R.A = c` la valeur de l'attribut est la constante c
  - Idem mais avec $\neq, <, >, <=, >=$ à la place de $=$
- Exemples :
  - `Film.acteur = Film.réalisateur`
  - `Film.acteur` $\neq$ `'Nicolson'`
  - `Film.acteur = Film.réalisateur` $\wedge$ `Film.acteur = 'Nicolson'`
- Fournit à l'utilisateur une vue des données en omettant les tuples qui ne satisfont pas certaines conditions voulues par l'utilisateur.

La sélection : exemple

$$\sigma_{acteur=realisateur \wedge realisateur='Polanski'}$$

titre	réalisateur	acteur
Shining	Kubrick	Nicholson
The Player	Altman	Robins
Chinatown	Polanski	Nicholson
Chinatown	Polanski	Polanski
Repulsion	Polanski	Deneuve

=

titre	réalisateur	acteur
Chinatown	Polanski	Polanski

Combiner sélection et projection

- Trouver les réalisateurs qui ont joué dans leurs propres films :
- `réponse(real) :- film(ti,real,act), act=real`
- ```
  SELECT réalisateur
  FROM Film
  WHERE réalisateur=acteur ;
  ```
- Requête de l'algèbre relationnelle :

$$Q = \pi_{realisateur}(\sigma_{realisateur=acteur}(\text{Film}))$$

- $\sigma_{realisateur=acteur}(\text{Film})$ donne

titre	réalisateur	acteur
Chinatown	Polanski	Polanski

- D'où $\pi_{realisateur}(\sigma_{realisateur=acteur}(\text{Film}))$ donne

réalisateur
Polanski
  ```

Combiner sélection et projection

- Il peut y avoir plusieurs façons d'écrire la même chose
- Exemple : trouver les couples (titre, réalisateur) en excluant les films de Polanski
- `réponse(ti,real) :- film(ti,real,act), real` $\neq$ `'Polanski'`
- Requête de l'algèbre relationnelle :

$$Q_1 = \sigma_{realisateur \neq \text{`}Polanski\text{'}}(\pi_{titre,realisateur}(Film))$$

- Une autre requête, équivalente, de l'algèbre relationnelle :

$$Q_2 = \pi_{titre,realisateur}(\sigma_{realisateur \neq \text{`}Polanski\text{'}}(Film))$$

- La même requête déclarative peut avoir plusieurs traductions procédurales.

Combiner sélection et projection

- Q_1 et Q_2, est-ce que c'est *la même* requête ?
- Sémantiquement, oui : elles produisent le même résultat
- Mais elles différent en termes d'efficacité.
- Q_1 parcourt d'abord `Film`, projette deux attributs, et parcourt à nouveau le résultat.
- Q_2 parcourt `Film`, sélectionne certains tuples, et parcourt ensuite les tuples sélectionnés.
- Q_2 semble donc plus efficace dans le contexte présent.
- Les langages procéduraux peuvent être optimisés : il y a des manières sémantiquement équivalentes de calculer la même requête, et certaines sont plus efficaces que d'autres.

Le Produit Cartésien

- Met deux relations ensemble.
- $R_1 \times R_2$ associe chaque tuple t_1 de R_1 avec chaque tuple t_2 de R_2
- Exemple :

R_1	A	B
	a_1	b_1
	a_2	b_2

$\times$

R_2	C	D
	c_1	d_1
	c_2	d_2
	c_3	d_3

$=$

A	B	C	D
a_1	b_1	c_1	d_1
a_1	b_1	c_2	d_2
a_1	b_1	c_3	d_3
a_2	b_2	c_1	d_1
a_2	b_2	c_2	d_2
a_2	b_2	c_3	d_3

- Condition : R_1 et R_2 ne doivent avoir aucun attribut en commun.

Le Produit Cartésien

- Convention : pour distinguer les attributs en commun de deux relations R_1 et R_2, on peut préfixer leur nom par le nom de la relation dont ils proviennent.
- Exemple :

R_1	A	B
	a_1	b_1
	a_2	b_2

$\times$

R_2	A	C
	a_1	c_1
	a_2	c_2
	a_3	c_3

$=$

$R_1.A$	$R_1.B$	$R_2.A$	$R_2.C$
a_1	b_1	a_1	c_1
a_1	b_1	a_2	c_2
a_1	b_1	a_3	c_3
a_2	b_2	a_1	c_1
a_2	b_2	a_2	c_2
a_2	b_2	a_3	c_3

- Nous avons renommé les attributs pour inclure le nom des relations : dans la table résultante, tous les attributs doivent avoir des noms différents.

Le Produit Cartésien

- Si R_1 a n tuples et R_2 a m tuples, alors $R_1 \times R_2$ a $n \times m$ tuples
- Opération coûteuse : si R et S ont chacun 1000 tuples (petites relations), $R \times S$ a 1000×1000=1 000 000 tuples ($\neq$ petit).
- Les algorithmes d'optimisation de requêtes essaient d'éviter la construction des produits - à la place ils tentent d'en construire seulement des sous-ensembles ne contenant que les informations pertinentes.

Le Produit Cartésien : exemple

Trouver les cinémas qui jouent des films de Polanski :

```
réponse(ci) :- Film(ti,real,act), Seance(ci,ti), real='Polanski'
```

- Étape 1 : Soit $R_1 = Film \times Seance$
- On ne veut que les tuples dans lesquels les titres sont identiques, d'où :
- Étape 2 : Soit $R_2 = \sigma_{Film.titre=Seance.titre}(R_1)$
- Étape 3 : On ne veut que les films de Polanski, d'où :

$$R_3 = \sigma_{realisateur='Polanski'}(R_2)$$

- Étape 4 : Dans la réponse, on ne veut que des cinémas, donc :

$$\text{Réponse} = \pi_{cinema}(R_3)$$

- Réponse :

$$\pi_{cinema}(\sigma_{realisateur='Polanski'}(\sigma_{Film.titre=Seance.titre}(Film \times Seance)))$$

Le Produit Cartésien : exemple

- Réponse :

$$\pi_{cinema}(\sigma_{realisateur='Polanski'}(\sigma_{Film.titre=Seance.titre}(Film \times Seance)))$$

- On peut aussi combiner plusieurs sélections en une seule :

$$\sigma_{cond_1}(\sigma_{cond_2}(R)) = \sigma_{cond_1 \wedge cond_2}(R)$$

(avec $cond_1$, $cond_2$ des conditions Booléennes sur les tuples)

- Réponse :

$$\pi_{cinema}(\sigma_{realisateur='Polanski' \wedge Film.titre=Seance.titre}(Film \times Seance))$$

Un autre exemple

Film	Titre	Réalisateur	Acteur
	Solaris	Tarkovski	Bondarchuk
	Shining	Kubrick	Nicholson
	Chinatown	Polanski	Nicholson
	Chinatown	Polanski	Polanski
	Repulsion	Polanski	Deneuve

Séance	Cinéma	Titre
	Le Champo	Shining
	Le Champo	Chinatown
	Le Champo	Solaris
	Odéon	Chinatown
	Odéon	Répulsion

Trouver les cinémas qui passent un film dans lequel joue Polanski :

- Calcul relationnel : $\{Cinema \mid \exists Titre, Realisateur$
  $(Titre, Realisateur, Polanski) \in Film,$
  $(Titre, Cinema) \in Seance\}$
- SQL :

```
Select Cinema
From Film, Seance
Where Acteur='Polanski'
AND Film.Titre=Seance.Titre ;
```

Un autre exemple

Film	Titre	Réalisateur	Acteur
	Solaris	Tarkovski	Bondarchuk
	Shining	Kubrick	Nicholson
	Chinatown	Polanski	Nicholson
	Chinatown	Polanski	Polanski
	Repulsion	Polanski	Deneuve

Séance	Cinéma	Titre
	Le Champo	Shining
	Le Champo	Chinatown
	Le Champo	Solaris
	Odéon	Chinatown
	Odéon	Répulsion

Trouver les cinémas qui passent un film dans lequel joue Polanski :

$$\pi_{Cinema}(\sigma_{Film.Titre=Seance.Titre}(\sigma_{Acteur='Polanski'}(Film) \times Seance))$$

On réalise d'abord une sélection σ :

Film	Titre	Réalisateur	Acteur
	Solaris	Tarkovski	Bondarchuk
	Shining	Kubrick	Nicholson
	Chinatown	Polanski	Nicholson
	Chinatown	Polanski	Polanski
	Repulsion	Polanski	Deneuve

6

Séance	Cinéma	Titre
	Le Champo	Shining
	Le Champo	Chinatown
	Le Champo	Solaris
	Odéon	Chinatown
	Odéon	Répulsion

Trouver les cinémas qui passent un film dans lequel joue Polanski :

$$\pi_{Cinema}(\sigma_{Film.Titre=Seance.Titre}(\sigma_{Acteur='Polanski'}(Film) \times Seance))$$

On fait ensuite un produit cartésien $\times$:

$\times$	F.Titre	F.Realisateur	F.Acteur	S.Cinema	S.Titre
	Chinatown	Polanski	Polanski	Le Champo	Shining
	Chinatown	Polanski	Polanski	Le Champo	Chinatown
	Chinatown	Polanski	Polanski	Le Champo	Solaris
	Chinatown	Polanski	Polanski	Odéon	Chinatown
	Chinatown	Polanski	Polanski	Odéon	Repulsion

Trouver les cinémas qui passent un film dans lequel joue Polanski :

$$\pi_{Cinema}(\sigma_{Film.Titre=Seance.Titre}(\sigma_{Acteur='Polanski'}(Film) \times Seance))$$

Une autre sélection σ :

×	F.Titre	F.Réalisateur	F.Acteur	S.Cinéma	S.Titre
	Chinatown	Polanski	Polanski	Le Champo	Shining
	Chinatown	Polanski	Polanski	Le Champo	Chinatown
σ	Chinatown	Polanski	Polanski	Le Champo	Solaris
	Chinatown	Polanski	Polanski	Odéon	Chinatown
	Chinatown	Polanski	Polanski	Odéon	Repulsion

Trouver les cinémas qui passent un film dans lequel joue Polanski :

$$\pi_{Cinema}(\sigma_{Film.Titre=Seance.Titre}(\sigma_{Acteur='Polanski'}(Film) \times Seance))$$

Et finalement une projection π :

×	F.Titre	F.Réalisateur	F.Acteur	S.Cinéma	S.Titre
	Chinatown	Polanski	Polanski	Le Champo	Shining
	Chinatown	Polanski	Polanski	Le Champo	Chinatown
σ	Chinatown	Polanski	Polanski	Le Champo π	Solaris
	Chinatown	Polanski	Polanski	Odéon	Chinatown
	Chinatown	Polanski	Polanski	Odéon	Repulsion

Trouver les cinémas qui passent un film dans lequel joue Polanski :

$$\pi_{Cinema}(\sigma_{Film.Titre=Seance.Titre}(\sigma_{Acteur='Polanski'}(Film) \times Seance))$$

Film	Titre	Réalisateur	Acteur
	Solaris	Tarkovski	Bondarchuk
	Shining	Kubrick	Nicholson
	Chinatown	Polanski	Nicholson
	Chinatown	Polanski	Polanski
	Repulsion	Polanski	Deneuve

Séance	Cinéma	Titre
	Le Champo	Shining
	Le Champo	Chinatown
	Le Champo	Solaris
	Odéon	Chinatown
	Odéon	Répulsion

Trouver les cinémas qui passent un film dans lequel joue Polanski :

Reponse	S.Cinéma
	Le Champo
	Odéon

$$\pi_{Cinema}(\sigma_{Film.Titre=Seance.Titre}(\sigma_{Acteur='Polanski'}(Film) \times Seance))$$

Film	Titre	Réalisateur	Acteur
	Solaris	Tarkovski	Bondarchuk
	Shining	Kubrick	Nicholson
	Chinatown	Polanski	Nicholson
	Chinatown	Polanski	Polanski
	Repulsion	Polanski	Deneuve

Séance	Cinéma	Titre
	Le Champo	Shining
	Le Champo	Chinatown
	Le Champo	Solaris
	Odéon	Chinatown
	Odéon	Répulsion

Il y a plusieurs façons différentes pour le SGBD de calculer la même requête.

Considérons maintenant une façon un peu moins efficace de trouver les cinémas qui passent un film dans lequel joue Polanski :

$$\pi_{Cinema}(\sigma_{Film.Titre=Seance.Titre \wedge Acteur='Polanski'}(Film \times Seance))$$

Une technique un peu moins efficace :

$$\pi_{Cinema}(\sigma_{Film.Titre=Seance.Titre \wedge Acteur='Polanski'}(Film \times Seance))$$

On fait d'abord un produit cartésien :

×	F.Titre	F.Réalisateur	F.Acteur	S.Cinéma	S.Titre
	Solaris	Tarkovski	Bondarchuk	Le Champo	Shining
	Solaris	Tarkovski	Bondarchuk	Le Champo	Chinatown
	Solaris	Tarkovski	Bondarchuk	Le Champo	Solaris
	Solaris	Tarkovski	Bondarchuk	Odéon	Chinatown
	Solaris	Tarkovski	Bondarchuk	Odéon	Repulsion
	Shining	Kubrik	Nicholson	Le Champo	Shining
	Shining	Kubrik	Nicholson	Le Champo	Chinatown
	Shining	Kubrik	Nicholson	Le Champo	Solaris
	Shining	Kubrik	Nicholson	Odéon	Chinatown
	Shining	Kubrik	Nicholson	Odéon	Repulsion
	Chinatown	Polanski	Nicholson	Le Champo	Shining
	Chinatown	Polanski	Nicholson	Le Champo	Chinatown
	Chinatown	Polanski	Nicholson	Le Champo	Solaris
	Chinatown	Polanski	Nicholson	Odéon	Chinatown
	Chinatown	Polanski	Nicholson	Odéon	Repulsion
	Chinatown	Polanski	Polanski	Le Champo	Shining
	Chinatown	Polanski	Polanski	Le Champo	Chinatown
	Chinatown	Polanski	Polanski	Le Champo	Solaris
	Chinatown	Polanski	Polanski	Odéon	Chinatown
	Chinatown	Polanski	Polanski	Odéon	Repulsion
	Repulsion	Polanski	Deneuve	Le Champo	Shining
	Repulsion	Polanski	Deneuve	Le Champo	Chinatown
	Repulsion	Polanski	Deneuve	Le Champo	Solaris
	Repulsion	Polanski	Deneuve	Odéon	Chinatown
	Repulsion	Polanski	Deneuve	Odéon	Repulsion

Une technique un peu moins efficace :

$$\pi_{Cinema}(\sigma_{Film.Titre=Seance.Titre \wedge Acteur='Polanski'}(Film \times Seance))$$

On fait ensuite une sélection :

×	F.Titre	F.Réalisateur	F.Acteur	S.Cinéma	S.Titre
	Solaris	Tarkovski	Bondarchuk	Le Champo	Shining
	Solaris	Tarkovski	Bondarchuk	Le Champo	Chinatown
	Solaris	Tarkovski	Bondarchuk	Le Champo	Solaris
	Solaris	Tarkovski	Bondarchuk	Odéon	Chinatown
	Solaris	Tarkovski	Bondarchuk	Odéon	Repulsion
	Shining	Kubrik	Nicholson	Le Champo	Shining
	Shining	Kubrik	Nicholson	Le Champo	Chinatown
	Shining	Kubrik	Nicholson	Le Champo	Solaris
	Shining	Kubrik	Nicholson	Odéon	Chinatown
	Shining	Kubrik	Nicholson	Odéon	Repulsion
	Chinatown	Polanski	Nicholson	Le Champo	Shining
	Chinatown	Polanski	Nicholson	Le Champo	Chinatown
	Chinatown	Polanski	Nicholson	Le Champo	Solaris
	Chinatown	Polanski	Nicholson	Odéon	Chinatown
	Chinatown	Polanski	Nicholson	Odéon	Repulsion
	Chinatown	Polanski	Polanski	Le Champo	Shining
6	Chinatown	Polanski	Polanski	Le Champo	Chinatown
	Chinatown	Polanski	Polanski	Le Champo	Solaris
	Chinatown	Polanski	Polanski	Odéon	Chinatown
	Chinatown	Polanski	Polanski	Odéon	Repulsion
	Repulsion	Polanski	Deneuve	Le Champo	Shining
	Repulsion	Polanski	Deneuve	Le Champo	Chinatown
	Repulsion	Polanski	Deneuve	Le Champo	Solaris
	Repulsion	Polanski	Deneuve	Odéon	Chinatown
	Repulsion	Polanski	Deneuve	Odéon	Repulsion

- Une technique un peu moins efficace :

$$\pi_{Cinema}(\sigma_{Film.Titre=Seance.Titre \wedge Acteur='Polanski'}(Film \times Seance))$$

Et on termine par une projection :

×	F.Titre	F.Réalisateur	F.Acteur	S.Cinéma	S.Titre
	Solaris	Tarkovski	Bondarchuk	Le Champo	Shining
	Solaris	Tarkovski	Bondarchuk	Le Champo	Chinatown
	Solaris	Tarkovski	Bondarchuk	Le Champo	Solaris
	Solaris	Tarkovski	Bondarchuk	Odéon	Chinatown
	Solaris	Tarkovski	Bondarchuk	Odéon	Repulsion
	Shining	Kubrik	Nicholson	Le Champo	Shining
	Shining	Kubrik	Nicholson	Le Champo	Chinatown
	Shining	Kubrik	Nicholson	Le Champo	Solaris
	Shining	Kubrik	Nicholson	Odéon	Chinatown
	Shining	Kubrik	Nicholson	Odéon	Repulsion
	Chinatown	Polanski	Nicholson	Le Champo	Shining
	Chinatown	Polanski	Nicholson	Le Champo	Chinatown
	Chinatown	Polanski	Nicholson	Le Champo	Solaris
	Chinatown	Polanski	Nicholson	Odéon	Chinatown
	Chinatown	Polanski	Nicholson	Odéon	Repulsion
	Chinatown	Polanski	Polanski	Le Champo	Shining
σ	Chinatown	Polanski	Polanski	Le Champo	Chinatown
	Chinatown	Polanski	Polanski	Le Champo	Solaris
	Chinatown	Polanski	Polanski	Odéon	Chinatown
	Chinatown	Polanski	Polanski	Odéon	Repulsion
	Repulsion	Polanski	Deneuve	Le Champo	Shining
	Repulsion	Polanski	Deneuve	Le Champo	Chinatown
	Repulsion	Polanski	Deneuve	Le Champo	Solaris
	Repulsion	Polanski	Deneuve	Odéon	Chinatown
	Repulsion	Polanski	Deneuve	Odéon	Repulsion

π

$$\pi_{Cinema}(\sigma_{Film.Titre=Seance.Titre}(\sigma_{Acteur='Polanski'}(Film) \times Seance))$$

vs

$$\pi_{Cinema}(\sigma_{Movie.Titre=Seance.Titre \wedge Acteur='Polanski'}(Film \times Seance))$$

- Dans le premier cas, on a d'abord opéré une sélection. Le produit cartésien réalisé ensuite contenait 1×5=5 tuples. Mais si on fait le produit cartésien d'abord, il contient 4×5=20 tuples.
- Pas optimal : nous voulons de petites tables intermédiaires.
- Moral : pousser les sélections aussi loin que possible à l'intérieur des requêtes.
- Une des règles derrière l'optimisation des requêtes.

Rappels : SQL et l'algèbre relationnelle

Pour sélectionner toutes les colonnes de la table Film on peut utiliser :

- SQL : `SELECT * FROM Film ;`
- Algèbre relationnelle : *Film*

Résultat :

Film	Titre	Réalisateur	Acteur
	Shining	Kubrick	Nicholson
	The Player	Altman	Robbins
	Chinatown	Polanski	Nicholson
	Chinatown	Polanski	Polanski
	Repulsion	Polanski	Deneuve

Rappels : SQL et l'algèbre relationnelle

On peut sélectionner certaines colonnes en particulier :

- SQL : `SELECT Titre, Realisateur FROM Film ;`
- Algèbre relationnelle : $\pi_{Titre,Realisateur}(Film)$

Résultat :

Film	Titre	Réalisateur
	Shining	Kubrick
	The Player	Altman
	Chinatown	Polanski
	Chinatown	Polanski
	Repulsion	Polanski

Rappels : SQL et l'algèbre relationnelle

On peut sélectionner certaines colonnes en particulier :

- SQL : `SELECT Titre FROM Film ;`

Résultat :

Film	Titre
	Shining
	The Player
	Chinatown
	Chinatown
	Repulsion

Rappels : SQL et l'algèbre relationnelle

On peut sélectionner certaines colonnes en particulier :

- Algèbre relationnelle : $\pi_{Titre}(Film)$

Résultat :

Film	Titre
	Shining
	The Player
	Chinatown
	Repulsion

Rappels : SQL et l'algèbre relationnelle

La clause WHERE et l'opérateur de sélection σ permettent de définir un ensemble de conditions de sélection qui doivent être vérifiées par les lignes retenues :

- SQL :

```
SELECT Titre
FROM Film
WHERE Realisateur = 'Altman';
```

- Algèbre relationnelle : $\pi_{Titre}(\sigma_{Realisateur='Altman'}(Film))$;

Résultat :

Film	Titre
	The Player

Rappels : SQL et l'algèbre relationnelle

- La clause FROM permet d'accéder à plusieurs tables
- But : recomposer de l'information distribuée dans plusieurs tables
- Exemple : déterminer dans quel cinéma on peut voir un film de quel réalisateur

Film	Titre	Réalisateur	Acteur
	Solaris	Tarkovski	Bondarchuk
	Shining	Kubrick	Nicholson
	Chinatown	Polanski	Nicholson
	Chinatown	Polanski	Polanski
	Repulsion	Polanski	Deneuve

Séance	Cinéma	Titre
	Le Champo	Shining
	Le Champo	Chinatown
	Le Champo	Solaris
	Odéon	Chinatown
	Odéon	Répulsion

Rappels : SQL et l'algèbre relationnelle

La clause FROM avec deux tables renvoie leur produit cartésien :

- chaque ligne de la 1ère table concatenée avec chaque ligne de la 2nde table

×	F.Titre	F.Réalisateur	F.Acteur	S.Cinéma	S.Titre
	Solaris	Tarkovski	Bondarchuk	Le Champo	Shining
	Solaris	Tarkovski	Bondarchuk	Le Champo	Chinatown
	Solaris	Tarkovski	Bondarchuk	Le Champo	Solaris
	Solaris	Tarkovski	Bondarchuk	Odéon	Chinatown
	Solaris	Tarkovski	Bondarchuk	Odéon	Repulsion
	Shining	Kubrik	Nicholson	Le Champo	Shining
	Shining	Kubrik	Nicholson	Le Champo	Chinatown
	Shining	Kubrik	Nicholson	Le Champo	Solaris
	Shining	Kubrik	Nicholson	Odéon	Chinatown
	Shining	Kubrik	Nicholson	Odéon	Repulsion
	Chinatown	Polanski	Nicholson	Le Champo	Shining
	Chinatown	Polanski	Nicholson	Le Champo	Chinatown
	Chinatown	Polanski	Nicholson	Le Champo	Solaris
	Chinatown	Polanski	Nicholson	Odéon	Chinatown
	Chinatown	Polanski	Nicholson	Odéon	Repulsion
	Chinatown	Polanski	Polanski	Le Champo	Shining
	Chinatown	Polanski	Polanski	Le Champo	Chinatown
	Chinatown	Polanski	Polanski	Le Champo	Solaris
	Chinatown	Polanski	Polanski	Odéon	Chinatown
	Chinatown	Polanski	Polanski	Odéon	Repulsion
	Repulsion	Polanski	Deneuve	Le Champo	Shining
	Repulsion	Polanski	Deneuve	Le Champo	Chinatown
	Repulsion	Polanski	Deneuve	Le Champo	Solaris
	Repulsion	Polanski	Deneuve	Odéon	Chinatown
	Repulsion	Polanski	Deneuve	Odéon	Repulsion

Rappels : SQL et l'algèbre relationnelle

La clause FROM avec deux tables renvoie leur produit cartésien

- SQL : `SELECT * FROM Film, Seance ;`
- Algèbre relationnelle : $Film \times Seance$

×	F.Titre	F.Réalisateur	F.Acteur	S.Cinéma	S.Titre
	Solaris	Tarkovski	Bondarchuk	Le Champo	Shining
	Solaris	Tarkovski	Bondarchuk	Le Champo	Chinatown
	Solaris	Tarkovski	Bondarchuk	Le Champo	Solaris
	Solaris	Tarkovski	Bondarchuk	Odéon	Chinatown
	Solaris	Tarkovski	Bondarchuk	Odéon	Repulsion
	Shining	Kubrik	Nicholson	Le Champo	Shining
	Shining	Kubrik	Nicholson	Le Champo	Chinatown
	Shining	Kubrik	Nicholson	Le Champo	Solaris
	Shining	Kubrik	Nicholson	Odéon	Chinatown
	Shining	Kubrik	Nicholson	Odéon	Repulsion
	Chinatown	Polanski	Nicholson	Le Champo	Shining
	Chinatown	Polanski	Nicholson	Le Champo	Chinatown
	Chinatown	Polanski	Nicholson	Le Champo	Solaris
	Chinatown	Polanski	Nicholson	Odéon	Chinatown
	Chinatown	Polanski	Nicholson	Odéon	Repulsion
	Chinatown	Polanski	Polanski	Le Champo	Shining
	Chinatown	Polanski	Polanski	Le Champo	Chinatown
	Chinatown	Polanski	Polanski	Le Champo	Solaris
	Chinatown	Polanski	Polanski	Odéon	Chinatown
	Chinatown	Polanski	Polanski	Odéon	Repulsion
	Repulsion	Polanski	Deneuve	Le Champo	Shining
	Repulsion	Polanski	Deneuve	Le Champo	Chinatown
	Repulsion	Polanski	Deneuve	Le Champo	Solaris
	Repulsion	Polanski	Deneuve	Odéon	Chinatown
	Repulsion	Polanski	Deneuve	Odéon	Repulsion

Rappels : SQL et l'algèbre relationnelle

La plupart du temps le produit cartésien contient des lignes "inutiles" :

- quel intérêt de concaténer "Tarkovski" avec "Odéon" si aucun des films de Tarkovski n'est joué à l'Odéon ?

Pour sélectionner uniquement les lignes du produit qui sont reliées on peut utiliser :

- SQL :

```
SELECT *
FROM Film, Seance
WHERE Film.Titre = Seance.Titre;
```

- Algèbre relationnelle : $\sigma_{Film.Titre=Seance.Titre}(Film \times Seance)$

Rappels : SQL et l'algèbre relationnelle

- ```
  SELECT *
  FROM Film, Seance
  WHERE Film.Titre = Seance.Titre;
  ```
- $\sigma_{Film.Titre=Seance.Titre}(Film \times Seance)$

×	F.Titre	F.Réalisateur	F.Acteur	S.Cinéma	S.Titre
	Solaris	Tarkovski	Bondarchuk	Le Champo	Shining
	Solaris	Tarkovski	Bondarchuk	Le Champo	Chinatown
	Solaris	Tarkovski	Bondarchuk	Le Champo	Solaris
	Solaris	Tarkovski	Bondarchuk	Odéon	Chinatown
	Solaris	Tarkovski	Bondarchuk	Odéon	Repulsion
	Shining	Kubrik	Nicholson	Le Champo	Shining
	Shining	Kubrik	Nicholson	Le Champo	Chinatown
	Shining	Kubrik	Nicholson	Le Champo	Solaris
	Shining	Kubrik	Nicholson	Odéon	Chinatown
	Shining	Kubrik	Nicholson	Odéon	Repulsion
	Chinatown	Polanski	Nicholson	Le Champo	Shining
	Chinatown	Polanski	Nicholson	Le Champo	Chinatown
	Chinatown	Polanski	Nicholson	Le Champo	Solaris
	Chinatown	Polanski	Nicholson	Odéon	Chinatown
	Chinatown	Polanski	Nicholson	Odéon	Repulsion
	Chinatown	Polanski	Polanski	Le Champo	Shining
	Chinatown	Polanski	Polanski	Le Champo	Chinatown
	Chinatown	Polanski	Polanski	Le Champo	Solaris
	Chinatown	Polanski	Polanski	Odéon	Chinatown
	Chinatown	Polanski	Polanski	Odéon	Repulsion
	Repulsion	Polanski	Deneuve	Le Champo	Shining
	Repulsion	Polanski	Deneuve	Le Champo	Chinatown
	Repulsion	Polanski	Deneuve	Le Champo	Solaris
	Repulsion	Polanski	Deneuve	Odéon	Chinatown
	Repulsion	Polanski	Deneuve	Odéon	Repulsion
  ```

Rappels : SQL et l'algèbre relationnelle

Revient à concaténer uniquement les lignes des deux tuples qui satisfont la condition du WHERE / de la sélection

×	F.Titre	F.Réalisateur	F.Acteur	S.Cinéma	S.Titre
	Solaris	Tarkovski	Bondarchuk	Le Champo	Solaris
	Shining	Kubrik	Nicholson	Le Champo	Shining
	Chinatown	Polanski	Nicholson	Le Champo	Chinatown
	Chinatown	Polanski	Nicholson	Odéon	Chinatown
	Chinatown	Polanski	Polanski	Le Champo	Chinatown
	Chinatown	Polanski	Polanski	Odéon	Chinatown
	Repulsion	Polanski	Deneuve	Odéon	Repulsion

Rappels : SQL et l'algèbre relationnelle

On retient finalement certaines colonnes via `SELECT` ou via π :

- ```
  SELECT Realisateur, Cinema
  FROM Film, Seance
  WHERE Film.Titre = Seance.Titre;
  ```
- $\pi_{Realisateur, Cinema}(\sigma_{Film.Titre=Seance.Titre}(Film \times Seance))$

Film	Réalisateur	Cinéma
	Tarkovski	Le Champo
	Kubrik	Le Champo
	Polanski	Le Champo
	Polanski	Odéon
	Polanski	Le Champo
	Polanski	Odéon
	Polanski	Odéon
  ```

Rappels : SQL et l'algèbre relationnelle

On retient finalement certaines colonnes via `SELECT` ou via π :

- ```
  SELECT DISTINCT Realisateur, Cinema
  FROM Film, Seance
  WHERE Film.Titre = Seance.Titre;
  ```
- $\pi_{Realisateur,Cinema}(\sigma_{Film.Titre=Seance.Titre}(Film \times Seance))$

Film	Réalisateur	Cinéma
	Tarkovski	Le Champo
	Kubrik	Le Champo
	Polanski	Le Champo
	Polanski	Odéon

On supposera dans ce cours que l'algèbre relationnelle travaille sur des tables sans doublons. (En fait, il existe une opération d'"élimination des doublons" qui est implémentée en pratique.)
  ```

La jointure

- Le produit cartésien (`FROM Table1, Table2` ou $Table1 \times Table2$) suivi d'une condition de sélection (`WHERE <condition>` ou $\sigma_{<condition>}$) est appelé jointure (join)
- La `<condition>` est appelée condition de jointure
- L'opérateur correspondant en algèbre relationnelle est l'opérateur de théta jointure $\underset{\theta}{\bowtie}$ (la tradition veut que le petit nom de la condition soit θ...)

Syntaxes alternatives :

- ```
  SELECT Realisateur, Cinema
  FROM Film JOIN Seance ON (Film.titre=Seance.Titre) ;
  ```
- $\pi_{Realisateur,Cinema}(Film \underset{Film.Titre=Seance.Titre}{\bowtie} Seance)$
  ```

La jointure

$$\pi_{A,X.B,C}(X \underset{X.B=Y.B}{\bowtie} Y)$$

Un exemple plus général (chaque ligne est en jointure avec plusieurs lignes)

X

A	B
1	2
4	7
4	2
8	3
5	4

Y

B	C
1	6
2	2
4	2
4	3

```
SELECT  A, X.B, C
FROM X, Y
WHERE X.B = Y.B ;
```

X,Y

X.A	X.B	Y.B	Y.C
1	2	1	6
4	7	1	6
4	2	1	6
8	3	1	6
5	4	1	6
1	2	2	2
4	7	2	2
4	2	2	2
8	3	2	2
5	4	2	2
1	2	4	2
4	7	4	2
4	2	4	2
8	3	4	2
5	4	4	2
1	2	4	3
4	7	4	3
4	2	4	3
8	3	4	3
5	4	4	3

La jointure

$$\pi_{A,X.B,C}(X \underset{X.B=Y.B}{\bowtie} Y)$$

Un exemple plus général (chaque ligne est en jointure avec plusieurs lignes)

X

A	B
1	2
4	7
4	2
8	3
5	4

Y

B	C
1	6
2	2
4	2
4	3

```
SELECT  A, X.B, C
FROM X, Y
WHERE X.B = Y.B ;
```

X,Y

X.A	X.B	Y.B	Y.C
1	2	1	6
4	7	1	6
4	2	1	6
8	3	1	6
5	4	1	6
1	2	2	2
4	7	2	2
4	2	2	2
8	3	2	2
5	4	2	2
1	2	4	2
4	7	4	2
4	2	4	2
8	3	4	2
5	4	4	2
1	2	4	3
4	7	4	3
4	2	4	3
8	3	4	3
5	4	4	3

La jointure

$$\pi_{A,X.B,C}(X \underset{X.B=Y.B}{\bowtie} Y)$$

X

A	B
1	2
4	7
4	2
8	3
5	4

Y

B	C
1	6
2	2
4	2
4	3

```
SELECT  A, X.B, C
FROM X, Y
WHERE X.B = Y.B ;
```

Résultat final :

A	X.B	C
1	2	2
4	2	2
5	4	2
5	4	3

La jointure naturelle

- Étape importante tout à l'heure :
  1. $Table_1 = Séance \times Film$
  2. S'assurer que l'on parle bien du même film :

$$Table_2 = \sigma_{Séance.titre=Film.titre}(Table_1)$$

- Attributs de $Table_2$: Séance.cinema, Séance.titre, Film.titre, Film.réalisateur, Film.acteur
- Mais l'un des attributs est redondant :
  Film.titre et Séance.titre sont toujours identiques dans $Table_2$.
- On réduit donc $Table_2$ à une relation plus simple avec les attributs :
  Séance.cinema, titre, Film.réalisateur, Film.acteur
- Il s'agit de la jointure naturelle : Séance $\bowtie$ Film

La jointure naturelle : un exemple

Titre	Réalisateur	Acteur
Shining	Kubrick	Nicholson
The Player	Altman	Robbins
Chinatown	Polanski	Nicholson
Chinatown	Polanski	Polanski
Repulsion	Polanski	Deneuve

$\bowtie$

Cinéma	Titre
Le Champo	Shining
Le Champo	Chinatown
Le Champo	The Player
Odéon	Chinatown

=

Titre	Réalisateur	Acteur	Cinema
Shining	Kubrick	Nicholson	Le Champo
The Player	Altman	Robbins	Le Champo
Chinatown	Polanski	Nicholson	Le Champo
Chinatown	Polanski	Nicholson	Odéon
Chinatown	Polanski	Polanski	Le Champo
Chinatown	Polanski	Polanski	Odéon

La jointure naturelle : un exemple

Simplifie l'écriture de certaines requêtes, quand le schéma utilise les mêmes noms pour les clefs étrangères et les clefs primaires référencées.

- Algèbre relationnelle : $Film \bowtie Seance$
- SQL :

```
SELECT *
FROM Film NATURAL JOIN Seance ;
```

Titre	Réalisateur	Acteur	Cinema
Shining	Kubrick	Nicholson	Le Champo
The Player	Altman	Robbins	Le Champo
Chinatown	Polanski	Nicholson	Le Champo
Chinatown	Polanski	Nicholson	Odéon
Chinatown	Polanski	Polanski	Le Champo
Chinatown	Polanski	Polanski	Odéon

La jointure naturelle : un exemple

- Algèbre relationnelle : $Film \bowtie Seance$
- SQL :

```
SELECT Film.Titre, Réalisateur, Acteur, Cinéma
FROM Film, Seance
WHERE Film.Titre=Seance.Titre ;
```

Titre	Réalisateur	Acteur	Cinema
Shining	Kubrick	Nicholson	Le Champo
The Player	Altman	Robbins	Le Champo
Chinatown	Polanski	Nicholson	Le Champo
Chinatown	Polanski	Nicholson	Odéon
Chinatown	Polanski	Polanski	Le Champo
Chinatown	Polanski	Polanski	Odéon

La jointure naturelle

- La jointure n'est pas une nouvelle opération de l'algèbre relationnelle.
- Elle est définissable à partir de $\pi, \sigma, \times$
- Soit R une relation sur des attributs $A_1, \ldots, A_n, B_1, \ldots, B_k$
- S une relation sur des attributs $A_1, \ldots, A_n, C_1, \ldots, C_m$
- $R \bowtie S$ a pour attributs $A_1, \ldots, A_n, B_1, \ldots, B_k, C_1, \ldots, C_m$

$$R \bowtie S$$
$$=$$
$$\pi_{R.A_1,\ldots,R.A_n,B_1,\ldots,B_k,C_1,\ldots,C_m}(\sigma_{R.A_1=S.A_1 \wedge \ldots \wedge R.A_n=S.A_n}(R \times S))$$

Exercice : montrer que la théta jointure est également définissable

Propriétés de la jointure

Quelques équivalences algébriques :

- Commutativité : $R \bowtie S = S \bowtie R$
- Associativité : $R \bowtie (S \bowtie T) = (R \bowtie S) \bowtie T$
- On peut donc écrire $R_1 \bowtie R_2 \bowtie \ldots \bowtie R_n$

Du point de vue de l'optimisation des requêtes, l'ordre dans lequel sont opérées les jointures est crucial. (Le SGBD travaille pour vous et détermine la solution la plus efficace.)

Commutativité et associativité de la jointure

R	employee	department
	Jones	D1
	Brown	D2
	Smith	D3

S	department	office
	D1	USA
	D2	UK

T	office	head
	USA	Andrews
	UK	Morrison

$R \bowtie S \bowtie T$	employee	department	office	head
	Jones	D1	USA	Andrews
	Brown	D2	UK	Morrison

L'ordre des jointures

Exemple d'un schéma avec trois relations :

- `Enseignant(ID*, Nom, UFR, Salaire)`
- `Enseigne(#ID*, #ID_cours*, Semestre, Année)`
- `Cours(ID_cours*, intitulé, UFR, coef)`

La liste des cours donnés par des enseignants de l'UFR d'Informatique, avec pour chaque cours, l'enseignant qui le donne :

```
SELECT Nom, Intitulé
FROM Enseignant, Enseigne, Cours
WHERE Enseignant.ID=Enseigne.ID
AND Enseigne.ID_cours=Cours.ID_Cours
AND Enseignant.UFR='Informatique' ;
```

L'ordre des jointures

Exemple d'un schéma avec trois relations :

- `Enseignant(ID*, Nom, UFR, Salaire)`
- `Enseigne(#ID*, #IDcours*, Semestre, Année)`
- `Cours(IDcours*, intitulé, UFR, coef)`

La liste des cours donnés par des enseignants de l'UFR d'Informatique, avec pour chaque cours, l'enseignant qui le donne :

$$\pi_{Nom,Intitule}(\sigma_{UFR='Informatique'}(Enseignant) \bowtie Enseigne \bowtie \pi_{IDcours,Intitule}(Cours))$$

L'ordre des jointures

Dans quel ordre traiter les jointures ?

$$\pi_{Nom,Intitule}(\sigma_{UFR='Informatique'}(Enseignant) \bowtie Enseigne \bowtie \pi_{IDcours,Intitule}(Cours))$$

- Option 1 : construire d'abord $Enseigne \bowtie \pi_{IDcours,Intitule}(Cours)$, puis joindre le résultat avec $\sigma_{UFR='Informatique'}(Enseignant)$
- Option 2 : construire d'abord $\sigma_{UFR='Informatique'}(Enseignant) \bowtie Enseigne$, puis joindre le résultat avec $\pi_{IDcours,Intitule}(Cours)$

L'ordre des jointures

Dans quel ordre traiter les jointures ?

$$\pi_{Nom,Intitule}(\sigma_{UFR='Informatique'}(Enseignant) \bowtie Enseigne \bowtie \pi_{IDcours,Intitule}(Cours))$$

- Option 1 : $Enseigne \bowtie \pi_{IDcours,Intitule}(Cours)$ risque d'être très grande, puisqu'elle contient un tuple pour chaque cours de chaque UFR
- Option 2 : $\sigma_{UFR='Informatique'}(Enseignant) \bowtie Enseigne$ contient certainement moins de tuples : seule une petite proportion des enseignants de Paris Diderot appartiennent à l'UFR d'Informatique

L'ordre des jointures

Dans quel ordre traiter les jointures ?

$$\pi_{Nom,Intitule}(\sigma_{UFR='Informatique'}(Enseignant) \bowtie Enseigne \bowtie \pi_{IDcours,Intitule}(Cours))$$

- Option 2 : la meilleure option, car elle minimise la taille des tables intermédiaires.

Requêtes Select-Project-Join (SPJ)

- Il s'agit des requêtes les plus courantes.
- Requêtes SELECT-FROM-WHERE simples.
- Trouver les cinémas qui passent des films de Polanski :

```
SELECT Cinéma
FROM Film NATURAL JOIN Seance
WHERE Film.Realisateur=Polanski ;
```

- Comme requête SPJ :

$$\pi_{Seance.cinema}(\sigma_{realisateur='Polanski'}(Film \bowtie Seance))$$

- En quoi est-ce une simplification de la version précédente ?

$$\pi_{Seance.cinema}(\sigma_{Film.realisateur='Polanski'}(\sigma_{Film.titre=Seance.titre}(Film \times Seance)))$$

- $\sigma_{Film.titre=Seance.titre}$ est éliminé ; car impliqué par la jointure.

Propriétés des opérateurs de l'algèbre relationnelle

Taille des résultats

- Taille des résultats = nombre de tuples dans la table résultat
- Projection : $taille(\pi(R)) \leq taille(R)$
  (taille = nombre de tuples)
- Parfois, $taille(\pi(R)) < taille(R)$
- Se produit lorsque deux attributs ont les mêmes valeurs

$$\pi_{\mathsf{A}} \left(\begin{array}{cc} \mathsf{A} & \mathsf{B} \\ \hline \mathsf{a} & \mathsf{b1} \\ \mathsf{a} & \mathsf{b2} \end{array} \right) = \begin{array}{c} \mathsf{A} \\ \hline \mathsf{a} \end{array}$$

- Sélection : $0 \leq taille(\sigma(R)) \leq taille(R)$
- Dépend du nombre de tuples satisfaisant les conditions.

Taille des jointures et des produits Cartésiens

- $taille(R \times S) = taille(R) \times taille(S)$, mais :

$$0 \leq taille(R \bowtie S) \leq taille(R) \times taille(S)$$

- Certains tuples peuvent ne pas participer à la jointure :

R	employee	department
	Jones	D1
	Brown	D2
	Smith	D3

$\bowtie$

S	department	office
	D1	USA
	D2	UK

$=$

$R \bowtie S$	employee	department	office
	Jones	D1	USA
	Brown	D2	UK

- $(Smith, D3)$ n'est joint avec aucun tuple de S : S ne contient pas d'information sur le département D3.

Jointure vides

Les jointures peuvent être vides :

R	employee	department
	Jones	D1
	Brown	D2
	Smith	D3

$\bowtie$

S'	department	office
	D4	France
	D5	Italy

$=$

$R \bowtie S'$	employee	department	office

Vu jusqu'à présent

- Requêtes SQL simples `SELECT-FROM-WHERE`
- Correspondent aux requêtes définissables dans l'algèbre relationnelle avec $\pi, \sigma, \times$

Sauver de l'espace

- Répéter les noms de relations plusieurs fois est un peu lourd
- SQL nous laisse donc utiliser des noms de relations temporaires pour les relations
- ```
  SELECT S.cinema
  FROM Séance S, Film F
  WHERE S.titre=F.titre AND F.réalisateur='Polanski'
  ```
- Utiliser une variable après le nom d'une relation indique que la relation est temporairement renommée
  ```

Les requêtes imbriquées : un exemple simple

- Jusqu'à présent dans la clause `WHERE` nous avons utilisé des comparaisons d'attributs
- En général, une clause `WHERE` peut contenir une autre requête, et tester une relation entre un attribut et le résultat d'une autre requête.
- On parle de requêtes imbriquées, car elles utilisent des sous-requêtes
- Exemple : trouver les cinémas qui passent des films de Polanski :

```
SELECT Séance.cinema
FROM Séance
WHERE Séance.titre IN
                    (SELECT Film.titre
                    FROM Film
                    WHERE Film.réalisateur='Polanski');
```

Requêtes imbriquées : comparaison

```
SELECT Séance.cinema
FROM Séance
WHERE Séance.titre IN
      (SELECT Film.titre
      FROM Film
      WHERE Film.réalisateur='Polanski');
```

```
SELECT S.cinema
   FROM Séance S, Film F
WHERE S.titre=F.titre
    AND F.réalisateur='Polanski';
```

- Sémantique = même requête
- A gauche, chaque sous-requête réfère à une relation
- Avantage de l'imbrication : on peut utiliser des prédicats plus complexes que IN

Remarque : évitez d'imbriquer à outrance ! L'optimiseur n'aime pas trop ça...

Disjonction dans les requêtes

- Trouver des acteurs qui ont joué dans des films de Kubrick OU de Polanski
- ```
  SELECT acteur
  FROM Film
  WHERE réalisateur='Kubrik' OR réalisateur='Polanski';
  ```
- Est-ce qu'on pourrait définir ça avec une seule règle (sans le or) ?
- Non !
  ```

Disjonction dans les requêtes

- Solution : les disjonctions peuvent être représentées par un ensemble de règles :

```
réponse(act) :- Film(ti,real,act), real='Kubrick'
réponse(act) :- Film(ti,real,act), real='Polanski'
```

- Sémantique : calculer la réponse à chacune des règles, puis prendre leur union
- Syntaxe alternative en SQL :

```
        SELECT acteur
        FROM Film
        WHERE réalisateur='Kubrick'
UNION
        SELECT acteur
        FROM Film
        WHERE réalisateur='Polanski';
```

Disjonction dans les requêtes

- Comment traduire une requête avec des disjonctions dans l'algèbre relationnelle ?
- `réponse(act) :- Film(ti,real,act), real='Kubrick'` est traduit par

$$Q_1 = \pi_{acteur}(\sigma_{realisateur='Kubrick'}(Film))$$

- `réponse(act) :- Film(ti,real,act), real='Polanski'` est traduit par

$$Q_2 = \pi_{acteur}(\sigma_{realisateur='Polanski'}(Film))$$

- On traduit la requête entière par $Q_1 \cup Q_2$:

$$\pi_{acteur}(\sigma_{realisateur='Kubrick'}(Film)) \bigcup \pi_{acteur}(\sigma_{realisateur='Polanski'}(Film))$$

L'union dans l'algèbre relationnelle

- Une autre opération de l'algèbre relationnelle : l'union
  $R \cup S$ est l'union des relations R et S

- R et S doivent avoir les mêmes attributs.
  On a maintenant quatre opérations de l'algèbre relationnelle :

  $$\pi, \sigma, \times, \cup$$

  (et bien sûr, $\bowtie$ et $\underset{\theta}{\bowtie}$, qui sont définissables à partir de $\pi, \sigma, \times$)

- Ce fragment, est appelé algèbre relationnelle positive, ou requêtes SPJU (select-project-join-union)

Interaction des opérateurs de l'algèbre relationnelle

- $\pi_{\bar{A}}(R \cup S) = \pi_{\bar{A}}(R) \cup \pi_{\bar{A}}(S)$
- $\sigma_{cond}(R \cup S) = \sigma_{cond}(R) \cup \sigma_{cond}(S)$
- $(R \cup S) \times T = (R \times T) \cup (S \times T)$
- $T \times (R \cup S) = (T \times R) \cup (T \times S)$

où $\bar{A} = A_1, \ldots, A_n$ est une séquence d'attributs et *cond* est une condition

Requêtes SPJU

Toute requête SPJU est équivalente à une union de requêtes SPJ.

- Il suffit de propager l'opérateur d'union
- Exemple :

$$\pi_A(\sigma_{cond}((R \times (S \cup T)) \cup W)$$

$$=$$

$$\pi_A(\sigma_{cond}((R \times S) \cup (R \times T) \cup W))$$

$$=$$

$$\pi_A(\sigma_{cond}(R \times S) \cup \sigma_{cond}(R \times T) \cup \sigma_{cond} W))$$

$$=$$

$$\pi_A(\sigma_{cond}(R \times S)) \cup \pi_A(\sigma_{cond}(R \times T)) \cup \pi_A((\sigma_{cond} W))$$

Équivalences

Algèbre positive relationnelle

=

Union de requêtes SPJ

=

requêtes définies par un ensemble de règles

=

requêtes SQL `SELECT-FROM-WHERE-UNION`

=

unions de requêtes conjonctives

=

requêtes définies avec $\exists, \wedge, \vee$

- Question : est-ce que l'intersection est une requête SPJU ?
- i.e., étant donné R, S, avec les mêmes ensembles d'attributs, peut-on définir $R \cap S$?

Équivalences

Algèbre positive relationnelle

=

Union de requêtes SPJ

=

requêtes définies par un ensemble de règles

=

requêtes SQL `SELECT-FROM-WHERE-UNION`

=

unions de requêtes conjonctives

=

requêtes définies avec $\exists, \wedge, \vee$

- Question : est-ce que l'intersection est une requête SPJU ?
- i.e., étant donné R, S, avec les mêmes ensembles d'attributs, peut-on définir $R \cap S$? Oui, il suffit de prendre $R \bowtie S$.

Plus sur l'union

- Relation R_1 : *pere*, *enfant*

R_1	pere	enfant
	George	Elizabeth
	Philip	Charles
	Charles	William

- Relation R_2 : *mere*, *enfant*

R_2	mere	enfant
	Elizabeth	Charles
	Elizabeth	Andrews

- Nous voulons leur union, qui devrait être la relation "parent-enfant"
- Mais nous ne pouvons pas utiliser $R_1 \cup R_2$, parce que R_1 et R_2 ont des attributs différents !
- Nous devons donc renommer les attributs

Le renommage

- Soit R une relation qui a pour attribut A, mais pas B
- $\rho_{B \leftarrow A}(R)$ est la relation qui est comme R à ceci près que A est renommé B

$$\rho_{\text{parent} \leftarrow \text{pere}} \left(\begin{array}{cc} \text{pere} & \text{enfant} \\ \hline \text{George} & \text{Elizabeth} \\ \text{Philip} & \text{Charles} \\ \text{Charles} & \text{William} \end{array} \right) = \left(\begin{array}{cc} \text{parent} & \text{enfant} \\ \hline \text{George} & \text{Elizabeth} \\ \text{Philip} & \text{Charles} \\ \text{Charles} & \text{William} \end{array} \right)$$

$$\rho_{\text{parent} \leftarrow \text{mere}} \left(\begin{array}{cc} \text{mere} & \text{enfant} \\ \hline \text{Elizabeth} & \text{Charles} \\ \text{Elizabeth} & \text{Andrew} \end{array} \right) = \left(\begin{array}{cc} \text{parent} & \text{enfant} \\ \hline \text{Elizabeth} & \text{Charles} \\ \text{Elizabeth} & \text{Andrew} \end{array} \right)$$

Le renommage

L'union désirée est :

$$\rho_{parent \leftarrow pere}(R_1) \bigcup \rho_{parent \leftarrow mere}(R_2)$$

ce qui donne

parent	enfant
George	Elizabeth
Philip	Charles
Elizabeth	Charles
Elizabeth	Andrew
Charles	William

SQL et le renommage

- De nouveaux attributs peuvent être introduits dans les clauses SELECT en utilisant le mot AS

```
SELECT pere AS parent, enfant
FROM R1;

SELECT mere AS parent, enfant
FROM R2;
```

- On peut prendre l'union des deux requêtes, car elles ont le même ensemble d'attributs

```
      SELECT pere AS parent, enfant
      FROM R1
UNION
      SELECT mere AS parent, enfant
      FROM R2;
```

Les requêtes avec "pour tout"

$$\{real \mid \forall (ci, ti') \in Séance\ , \exists ti\ \exists act(Séance(ci, ti) \wedge Film(ti, real, act))\}$$

- Nouvel élément ici : la quantification universelle $\forall$ "pour tout"
- $\forall x\ F(x) \ = \ \neg \exists x\ \neg F(x)$
- Le nouvel élément est donc en fait la négation $\neg$

$$\{real \mid \neg \exists (ci, ti') \in Séance\ , \neg \exists ti\ \exists act(Séance(ci, ti) \wedge Film(ti, real, act))\}$$

La différence

L'expression de la négation en algèbre relationnelle :

- Si R et S sont deux relations avec le même ensemble d'attributs, alors $R - S$ est leur différence : l'ensemble des tuples qui occurrent dans R mais pas dans S

- Exemple :

A	B
a1	b1
a2	b2
a3	b3

$-$

A	B
a2	b2
a3	b3
a4	b4

$=$

A	B
a1	b1

L'algèbre relationnelle

- Inclut les opérateurs $\pi, \sigma, \times, \cup, -, \rho$

  Théorème fondamental de la théorie des bases de données relationnelles :
  Le calcul relationnel sûr = l'algèbre relationnelle

- Calcul relationnel "sûr" = on ne peut pas exprimer des choses comme "l'ensemble des réalisateurs qui ne figurent pas dans la base de donnée" (comment le définir ? ensemble infini ?)

Les requêtes avec "pour tout" dans l'algèbre relationnelle

- Trouver les réalisateurs dont les films sont joués dans tous les cinémas.

  $\{real \mid \forall (ci, ti') \in Séance\ \exists ti, act(Séance(ci, ti) \wedge Film(ti, real, act))\}$

- On définit :
  - $C_1 = \pi_{cinéma}(S)$
  - $C_2 = \pi_{cinéma,réalisateur}(F \bowtie S)$

  (pour sauver de l'espace on va utiliser S pour Séance et F pour Film)
- C_1 contient tous les cinémas, C_2 contient tous les réalisateurs avec les cinémas où leurs films passent.
- Notre requête est :

$$\{real \mid \forall ci \in C_1,\ (ci, real) \in C_2)\}$$

Requêtes avec "pour tout"

$$\{real \mid \forall ci \in C_1 \ (ci, real) \in C_2)\}$$

se réécrit

$$\{real \mid \neg(\exists ci \in C_1 \ (ci, real) \notin C_2)\}$$

La réponse à la requête est donc

$$\pi_{réalisateur}(F) - V$$

où $V = \{real \mid (\exists ci \in C_1 \ (ci, real) \notin C_2)\}$

Les paires $(ci, real)$ qui ne sont pas dans C_2 sont

$$(C_1 \times \pi_{réalisateur}(F)) - C_2$$

D'où :

$$V = \pi_{réalisateur}((C_1 \times \pi_{réalisateur}(F)) - C_2)$$

Requêtes avec "pour tout"

- Requête : trouver les réalisateurs dont les films passent dans tous les cinémas.
- On obtient donc :

$$\pi_{réalisateur}(F) - \pi_{réalisateur}((\pi_{cinéma}(S) \times \pi_{réalisateur}(F)) - \pi_{cinéma,réalisateur}(F \bowtie S))$$

- Beaucoup moins intuitif que la description logique de la requête.
- Les langages procéduraux sont loin d'être aussi compréhensibles que les langages déclaratifs...

Pour tout et la négation dans SQL

- Trouver les réalisateurs dont les films passent dans tous les cinémas.
- La façon SQL de dire ça : trouver tous les réalisateurs tels qu'il n'existe pas de cinéma dans lesquels leurs films ne passent pas.

```
SELECT F1.realisateur
FROM Film F1
WHERE NOT EXISTS (SELECT S.cinema
                  FROM Séance S
                  WHERE NOT EXISTS (SELECT F2.realisateur
                                    FROM Film F2
                                    WHERE F2.titre=S.titre
                                        AND
                                    F1.realisateur=F2.realisateur))
```

Pour tout et la négation dans SQL

- Sous requête corrélée assez complexe.
- Avant de l'analyser, revenons aux requêtes imbriquées.

```
SELECT F1.realisateur
FROM Film F1
WHERE NOT EXISTS (SELECT S.cinema
                  FROM Séance S
                  WHERE NOT EXISTS (SELECT F2.realisateur
                                    FROM Film F2
                                    WHERE F2.titre=S.titre
                                        AND
                                    F1.realisateur=F2.realisateur))
```

Les requêtes imbriquées

- Retour sur l'exemple du cours précédent
- En général, une clause `WHERE` peut contenir une autre requête, et tester une relation entre un attribut et le résultat d'une autre requête.
- On parle de requêtes imbriquées, car elles utilisent des sous-requêtes
- Exemple : trouver les cinémas qui passent des films de Polanski :

```
SELECT Séance.cinema
FROM Séance
WHERE Séance.titre IN
                    (SELECT Film.titre
                    FROM Film
                    WHERE Film.réalisateur='Polanski');
```

Les requêtes imbriquées

- Sémantique : `Attribut IN (SELECT ...)` est vrai si la valeur de `attribut` appartient au résultat de la requête `(SELECT ...)`
- Exemple : trouver les cinémas qui passent des films de Polanski :

```
SELECT Séance.cinema
FROM Séance
WHERE Séance.titre IN
                    (SELECT Film.titre
                    FROM Film
                    WHERE Film.réalisateur='Polanski');
```

$$\pi_{cinema}(\sigma_{realisateur='Polanski'}(Seance \bowtie Film))$$

Les requêtes imbriquées

- Exemple : trouver les cinémas qui passent des films de Polanski avec Deneuve :

```
SELECT Séance.cinema
FROM Séance
WHERE Séance.titre IN
              (SELECT Film.titre
               FROM Film
               WHERE Film.réalisateur='Polanski'
                AND
               Film.acteur IN
                   (SELECT Film.acteur
                   FROM Film
                   WHERE Film.acteur='Deneuve'));
```

$$\pi_{cinema}(\sigma_{realisateur='Polanski' \wedge acteur='Deneuve'}(Seance \bowtie Film))$$

Les requêtes imbriquées

```
SELECT Séance.cinema
FROM Séance
WHERE Séance.titre IN
            (SELECT Film.titre
             FROM Film
             WHERE Film.réalisateur='Polanski'
              AND
             Film.acteur IN
                 (SELECT Film.acteur
                 FROM Film
                 WHERE Film.acteur='Deneuve'));
```

Attention à ne pas trop imbriquer les requêtes.
(Souvent inutile. En plus l'optimiseur n'aime pas ça...)

Les requêtes imbriquées

- Cette structure emboitée correspond à l'association des tables Séance et Film sur la base des valeurs identiques de titre.

```
SELECT Séance.cinema
FROM Séance
WHERE Séance.titre IN
                (SELECT Film.titre
                FROM Film
                WHERE Film.réalisateur='Polanski');
```

- Comme les jointures, les sous-requêtes exploitent les liens entre tables formés par la couplage d'une clef primaire et d'une clef étrangère

Généralisation

- Les films qui partagent avec Chinatown le même réalisateur et au moins un acteur :

```
SELECT Film.titre
FROM Film
WHERE (Film.réalisateur, Film.acteur) IN
                    (SELECT Film.réalisateur, Film.acteur
                    FROM Film
                    WHERE Film.titre='Chinatown');
```

(Attention : ce type de requête SQL n'est pas implementé par tous les systèmes.)

$$\pi_{Film2.titre}(\sigma_{Film1.realisateur=Film2.realisateur \wedge Film1.acteur=Film2.acteur}$$
$$(\sigma_{Film1.titre=Chinatown}(Film1) \times Film2))$$

Généralisation

- Les films qui partagent avec Chinatown le même réalisateur et au moins un acteur (sans imbrication) :

```
SELECT F1.titre
FROM Film F1, Film F2
WHERE F2.titre='Chinatown'
AND F1.réalisateur=F2.réalisateur
AND F1.acteur=F2.acteur ;
```

$$\pi_{Film2.titre}(\sigma_{Film1.realisateur=Film2.realisateur \wedge Film1.acteur=Film2.acteur}$$

$$(\sigma_{Film1.titre=Chinatown}(Film1) \times Film2))$$

Forme générale des requêtes imbriquées

- Une condition du `WHERE` plus complexe : la valeur de l'ensemble spécifié peut être le résultat d'une sous-requête
- Trois types de conditions introduisant des sous requêtes, en :
  - IN
  - EXISTS
  - ALL, SOME, ANY (plus tard dans le semestre)
- Toutes ces conditions peuvent être combinées entre elles et avec d'autres plus simples à l'aide des opérateurs logiques `AND`, `OR` et `NOT`

Forme générale des requêtes imbriquées

Trouver les acteurs qui ont joué dans un film de Polanski mais pas dans un film de Kubrick :

```
SELECT F1.acteur
FROM Film F1
WHERE F1.acteur IN
      (SELECT F2.acteur
      FROM Film F2
      WHERE F2.realisateur='Polanski')
            AND
      F1.acteur NOT IN
      (SELECT F3.acteur
      FROM Film F2
      WHERE F3.realisateur='Kubrick');
```

Attention aux conditions d'association négatives

- Les films dans lesquels Deneuve ne joue pas :

```
SELECT Film.titre
FROM Film
WHERE Film.titre NOT IN
                  (SELECT Film.titre
                  FROM Film
                  WHERE Film.acteur='Deneuve');
```

$$\pi_{titre}(Film) - \pi_{titre}(\sigma_{acteur='Deneuve'}(Film))$$

Sémantique des sous requêtes en IN et NOT IN

Les films dans lesquels Deneuve ne joue pas :
$\pi_{titre}(Film) - \pi_{titre}(\sigma_{acteur='Deneuve'}(Film))$

```
SELECT Film.titre
FROM Film
WHERE Film.titre NOT IN
                  (SELECT Film.titre
                  FROM Film
                  WHERE Film.acteur = 'Deneuve');
```

Pour calculer le résultat de la sous requête, on commence par calculer $\sigma_{acteur='Deneuve'}(Film)$:

$Film$

réalisateur	titre	acteur
Varda	Les Créatures	Deneuve
Varda	Les Créatures	Piccoli
Chytilova	Sedmikrasky	Cerhova
Bunuel	Belle de jour	Deneuve
Varda	Cléo de 5 à 7	Marchand
Polanski	Repulsion	Deneuve

$\sigma_{acteur='Deneuve'}(Film)$

réalisateur	titre	acteur
Varda	Les Créatures	Deneuve
~~Varda~~	~~Les Créatures~~	~~Piccoli~~
~~Chytilova~~	~~Sedmikrasky~~	~~Cerhova~~
Bunuel	Belle de jour	Deneuve
~~Varda~~	~~Cléo de 5 à 7~~	~~Marchand~~
Polanski	Repulsion	Deneuve

Sémantique des sous requêtes en IN et NOT IN

Les films dans lesquels Deneuve ne joue pas :
$\pi_{titre}(Film) - \pi_{titre}(\sigma_{acteur='Deneuve'}(Film))$

```
SELECT Film.titre
FROM Film
WHERE Film.titre NOT IN
                (SELECT Film.titre
                FROM Film
                WHERE Film.acteur = 'Deneuve');
```

Pour calculer le résultat de la sous requête, on commence par calculer $\sigma_{acteur='Deneuve'}(Film)$:

Film

réalisateur	titre	acteur
Varda	Les Créatures	Deneuve
Varda	Les Créatures	Piccoli
Chytilova	Sedmikrasky	Cerhova
Bunuel	Belle de jour	Deneuve
Varda	Cléo de 5 à 7	Marchand
Polanski	Repulsion	Deneuve

$\sigma_{acteur='Deneuve'}(Film)$

réalisateur	titre	acteur
Varda	Les Créatures	Deneuve
Bunuel	Belle de jour	Deneuve
Polanski	Repulsion	Deneuve

Sémantique des sous requêtes en IN et NOT IN

Les films dans lesquels Deneuve ne joue pas :
$\pi_{titre}(Film) - \pi_{titre}(\sigma_{acteur='Deneuve'}(Film))$

```
SELECT Film.titre
FROM Film
WHERE Film.titre NOT IN
                (SELECT Film.titre
                FROM Film
                WHERE Film.acteur = 'Deneuve');
```

On calcule maintenant le résultat de la sous-requête :

$$\Pi_{titre}(\sigma_{acteur='Deneuve'}(Film))$$

réalisateur	titre	acteur
Varda	Les Créatures	Deneuve
Bunuel	Belle de jour	Deneuve
Polanski	Repulsion	Deneuve

Sémantique des sous requêtes en IN et NOT IN

Les films dans lesquels Deneuve ne joue pas :
$\pi_{titre}(Film) - \pi_{titre}(\sigma_{acteur='Deneuve'}(Film))$

```
SELECT Film.titre
FROM Film
WHERE Film.titre NOT IN
                (SELECT Film.titre
                FROM Film
                WHERE Film.acteur = 'Deneuve');
```

On projette ensuite également l'attribut titre sur la table Film :

$\Pi_{titre}(Film)$

réalisateur	titre	acteur
Varda	Les Créatures	Deneuve
Varda	Les Créatures	Piccoli
Chytilova	Sedmikrasky	Cerhova
Bunuel	Belle de jour	Deneuve
Varda	Cléo de 5 à 7	Marchand
Polanski	Repulsion	Deneuve

$\Pi_{titre}(\sigma_{acteur='Deneuve'}(Film))$

réalisateur	titre	acteur
Varda	Les Créatures	Deneuve
Bunuel	Belle de jour	Deneuve
Polanski	Repulsion	Deneuve

Sémantique des sous requêtes en IN et NOT IN

Les films dans lesquels Deneuve ne joue pas :
$\pi_{titre}(Film) - \pi_{titre}(\sigma_{acteur='Deneuve'}(Film))$

```
SELECT Film.titre
FROM Film
WHERE Film.titre NOT IN
                (SELECT Film.titre
                FROM Film
                WHERE Film.acteur = 'Deneuve');
```

On projette ensuite également l'attribut titre sur la table Film :

$$\Pi_{titre}(Film)$$

titre
Les Créatures
Sedmikrasky
Belle de jour
Cléo de 5 à 7
Repulsion

$$\Pi_{titre}(\sigma_{acteur='Deneuve'}(Film))$$

titre
Les Créatures
Belle de jour
Repulsion

Sémantique des sous requêtes en IN et NOT IN

Les films dans lesquels Deneuve ne joue pas :
$\pi_{titre}(Film) - \pi_{titre}(\sigma_{acteur='Deneuve'}(Film))$

```
SELECT Film.titre
FROM Film
WHERE Film.titre NOT IN
                (SELECT Film.titre
                FROM Film
                WHERE Film.acteur = 'Deneuve');
```

On calcule la différence ensembliste de ces deux tables :

$\Pi_{titre}(Film)$

titre
Les Créatures
Sedmikrasky
Belle de jour
Cléo de 5 à 7
Repulsion

différence ensembliste : −

$\Pi_{titre}(\sigma_{acteur='Deneuve'}(Film))$

titre
Les Créatures
Belle de jour
Repulsion

Sémantique des sous requêtes en IN et NOT IN

Les films dans lesquels Deneuve ne joue pas :
$\pi_{titre}(Film) - \pi_{titre}(\sigma_{acteur='Deneuve'}(Film))$

```
SELECT Film.titre
FROM Film
WHERE Film.titre NOT IN
                (SELECT Film.titre
                FROM Film
                WHERE Film.acteur = 'Deneuve');
```

Le tableau résultat de $\pi_{titre}(Film) - \pi_{titre}(\sigma_{acteur='Deneuve'}(Film))$ est :

$\Pi_{titre}(Film)$

titre
Les Créatures
Sedmikrasky
Belle de jour
Cléo de 5 à 7
Repulsion

différence ensembliste :

-

$\Pi_{titre}(\sigma_{acteur='Deneuve'}(Film))$

titre
Les Créatures
Belle de jour
Repulsion

=

tableau résultat :

titre
~~Les Créatures~~
Sedmikrasky
~~Belle de jour~~
Cléo de 5 à 7
~~Repulsion~~

Sémantique des sous requêtes en IN et NOT IN

Les films dans lesquels Deneuve ne joue pas :
$\pi_{titre}(Film) - \pi_{titre}(\sigma_{acteur='Deneuve'}(Film))$

```
SELECT Film.titre
FROM Film
WHERE Film.titre NOT IN
                (SELECT Film.titre
                FROM Film
                WHERE Film.acteur = 'Deneuve');
```

Le tableau résultat de $\pi_{titre}(Film) - \pi_{titre}(\sigma_{acteur='Deneuve'}(Film))$ est :

$\Pi_{titre}(Film)$

titre
Les Créatures
Sedmikrasky
Belle de jour
Cléo de 5 à 7
Repulsion

différence ensembliste :

-

$\Pi_{titre}(\sigma_{acteur='Deneuve'}(Film))$

titre
Les Créatures
Belle de jour
Repulsion

=

tableau résultat :

titre
Sedmikrasky
Cléo de 5 à 7

Attention !

La projection n'est pas distributive sur la différence :

$$\pi_{titre}(Film) - \pi_{titre}(\sigma_{acteur='Deneuve'}(Film))$$

(les films dans lesquels Deneuve ne joue pas)

$$\not\Leftrightarrow$$

$$\pi_{titre}(Film - \sigma_{acteur='Deneuve'}(Film))$$

(les films dans lesquels Deneuve n'est pas la seule personne à jouer)

$\pi_{titre}(Film - \sigma_{acteur='Deneuve'}(Film))$ correspond à :

```
SELECT Film.titre
FROM Film
WHERE Film.acteur NOT IN
                (SELECT Film.acteur
                FROM Film
                WHERE Film.acteur = 'Deneuve');
```

Attention !

La projection n'est pas distributive sur la différence :

$$\pi_{titre}(Film) - \pi_{titre}(\sigma_{acteur='Deneuve'}(Film))$$

(les films dans lesquels Deneuve ne joue pas)

$$\not\Leftrightarrow$$

$$\pi_{titre}(Film - \sigma_{acteur='Deneuve'}(Film))$$

(les films dans lesquels Deneuve n'est pas la seule personne à jouer)

ou, plus simplement :

```
SELECT Film.titre
FROM Film
WHERE Film.acteur <> 'Deneuve';
```

c'est-à-dire : $\pi_{titre}(\sigma_{acteur\neq'Deneuve'}(Film))$

Attention !

La projection n'est pas distributive sur la différence :

$$\pi_{titre}(Film) - \pi_{titre}(\sigma_{acteur='Deneuve'}(Film))$$

(les films dans lesquels Deneuve ne joue pas)

$$\not\Leftrightarrow$$

$$\pi_{titre}(Film - \sigma_{acteur='Deneuve'}(Film))$$

(les films dans lesquels Deneuve n'est pas la seule personne à jouer)

qu'on peut toujours réécrire de façon plus compliquée (et un peu tordue) comme :

```
SELECT Film.titre
FROM Film
WHERE Film.titre IN
                (SELECT Film.titre
                FROM Film
                WHERE Film.acteur <> 'Deneuve');
```

Sémantique des sous requêtes en IN et NOT IN

Les films dans lesquels Deneuve n'est pas la seule personne à jouer :
$\pi_{titre}(Film) \bowtie \sigma_{acteur \neq' Deneuve'}(Film)$

```
SELECT Film.titre
FROM Film
WHERE Film.titre IN
                  (SELECT Film.titre
                  FROM Film
                  WHERE Film.acteur <> 'Deneuve');
```

Pour calculer le résultat de la sous requête, on commence par calculer $\sigma_{acteur \neq' Deneuve'}(Film)$:

$Film$

réalisateur	titre	acteur
Varda	Les Créatures	Deneuve
Varda	Les Créatures	Piccoli
Chytilova	Sedmikrasky	Cerhova
Bunuel	Belle de jour	Deneuve
Varda	Cléo de 5 à 7	Marchand
Polanski	Repulsion	Deneuve

$\sigma_{acteur \neq' Deneuve'}(Film)$

réalisateur	titre	acteur
~~Varda~~	~~Les Créatures~~	~~Deneuve~~
Varda	Les Créatures	Piccoli
Chytilova	Sedmikrasky	Cerhova
~~Bunuel~~	~~Belle de jour~~	~~Deneuve~~
Varda	Cléo de 5 à 7	Marchand
~~Polanski~~	~~Repulsion~~	~~Deneuve~~

Sémantique des sous requêtes en IN et NOT IN

Les films dans lesquels Deneuve n'est pas la seule personne à jouer :
$\pi_{titre}(Film) \bowtie \sigma_{acteur \neq' Deneuve'}(Film)$

```
SELECT Film.titre
FROM Film
WHERE Film.titre IN
                  (SELECT Film.titre
                  FROM Film
                  WHERE Film.acteur <> 'Deneuve');
```

Pour calculer le résultat de la sous requête, on commence par calculer $\sigma_{acteur \neq' Deneuve'}(Film)$:

Film

réalisateur	titre	acteur
Varda	Les Créatures	Deneuve
Varda	Les Créatures	Piccoli
Chytilova	Sedmikrasky	Cerhova
Bunuel	Belle de jour	Deneuve
Varda	Cléo de 5 à 7	Marchand
Polanski	Repulsion	Deneuve

$\sigma_{acteur \neq' Deneuve'}(Film)$

réalisateur	titre	acteur
Varda	Les Créatures	Piccoli
Chytilova	Sedmikrasky	Cerhova
Varda	Cléo de 5 à 7	Marchan

Sémantique des sous requêtes en IN et NOT IN

Les films dans lesquels Deneuve n'est pas la seule personne à jouer :
$\pi_{titre}(Film) \bowtie \sigma_{acteur \neq' Deneuve'}(Film)$

```
SELECT Film.titre
FROM Film
WHERE Film.titre IN
                  (SELECT Film.titre
                  FROM Film
                  WHERE Film.acteur <> 'Deneuve');
```

On calcule maintenant le résultat de la sous-requête :

$$\Pi_{titre}(\sigma_{acteur \neq' Deneuve'}(Film))$$

réalisateur	titre	acteur
Varda	Les Créatures	Piccoli
Chytilova	Sedmikrasky	Cerhova
Varda	Cléo de 5 à 7	Marchand

Sémantique des sous requêtes en IN et NOT IN

Les films dans lesquels Deneuve n'est pas la seule personne à jouer :
$\pi_{titre}(Film) \bowtie \pi_{titre}(\sigma_{acteur \neq' Deneuve'}(Film))$

```
SELECT Film.titre
FROM Film
WHERE Film.titre IN
                  (SELECT Film.titre
                  FROM Film
                  WHERE Film.acteur <> 'Deneuve');
```

On projette ensuite l'attribut titre sur la table Film :

$\Pi_{titre}(Film)$

réalisateur	titre	acteur
Varda	Les Créatures	Piccoli
Chytilova	Sedmikrasky	Cerhova
Bunuel	Belle de jour	Deneuve
Varda	Cléo de 5 à 7	Marchand
Polanski	Repulsion	Deneuve

$\Pi_{titre}(\sigma_{acteur \neq' Deneuve'}(Film))$

réalisateur	titre	acteur
Varda	Les Créatures	Piccoli
Chytilova	Sedmikrasky	Cerhova
Varda	Cléo de 5 à 7	Marchand

Sémantique des sous requêtes en IN et NOT IN

Les films dans lesquels Deneuve n'est pas la seule personne à jouer :
$\pi_{titre}(Film) \bowtie \pi_{titre}(\sigma_{acteur \neq' Deneuve'}(Film))$

```
SELECT Film.titre
FROM Film
WHERE Film.titre IN
                (SELECT Film.titre
                FROM Film
                WHERE Film.acteur <> 'Deneuve');
```

On finit par prendre la jointure naturelle de ces deux tables :

$\Pi_{titre}(Film)$

titre
Les Créatures
Sedmikrasky
Belle de jour
Cléo de 5 à 7
Repulsion

jointure naturelle : $\bowtie$

$\Pi_{titre}(\sigma_{acteur \neq' Deneuve'}(Film))$

titre
Les Créatures
Sedmikrasky
Cléo de 5 à 7

Sémantique des sous requêtes en IN et NOT IN

Les films dans lesquels Deneuve n'est pas la seule personne à jouer :
$\pi_{titre}(Film) \bowtie \pi_{titre}(\sigma_{acteur \neq' Deneuve'}(Film))$

```
SELECT Film.titre
FROM Film
WHERE Film.titre IN
                (SELECT Film.titre
                FROM Film
                WHERE Film.acteur <> 'Deneuve');
```

Le tableau résultat de $\pi_{titre}(Film) \bowtie \pi_{titre}(\sigma_{acteur \neq' Deneuve'}(Film))$ est :

$\Pi_{titre}(Film)$

titre
Les Créatures
Sedmikrasky
Belle de jour
Cléo de 5 à 7
Repulsion

jointure naturelle :

$\bowtie$

$\Pi_{titre}(\sigma_{acteur \neq' Deneuve'}(Film))$

titre
Les Créatures
Sedmikrasky
Cléo de 5 à 7

=

tableau résultat :

titre
Les Créatures
Sedmikrasky
Cléo de 5 à 7

Sémantique des sous requêtes en IN et NOT IN

Les films dans lesquels Deneuve n'est pas la seule personne à jouer :
$\pi_{titre}(Film) \bowtie \pi_{titre}(\sigma_{acteur \neq' Deneuve'}(Film))$

```
SELECT Film.titre
FROM Film
WHERE Film.titre IN
                (SELECT Film.titre
                FROM Film
                WHERE Film.acteur <> 'Deneuve');
```

Conclusion : ça aurait été quand même plus simple de ne calculer que $\pi_{titre}(\sigma_{acteur \neq' Deneuve'}(Film))$...

$\Pi_{titre}(Film)$

titre
Les Créatures
Sedmikrasky
Belle de jour
Cléo de 5 à 7
Repulsion

jointure naturelle :

$\bowtie$

$\Pi_{titre}(\sigma_{acteur \neq' Deneuve'}(Film))$

titre
Les Créatures
Sedmikrasky
Cléo de 5 à 7

=

tableau résultat :

titre
Les Créatures
Sedmikrasky
Cléo de 5 à 7

Attention aux conditions d'association négatives

- Les films dans lesquels ne joue que Deneuve :

```
SELECT Film.titre
FROM Film
WHERE Film.titre NOT IN
                  (SELECT Film.titre
                  FROM Film
                  WHERE Film.acteur <> 'Deneuve');
```

$$\pi_{titre}(Film) - \pi_{titre}(\sigma_{acteur \neq' Deneuve'}(Film))$$

Troisième requête complètement différente.

Sémantique des sous requêtes en IN et NOT IN

Les films dans lesquels ne joue que Deneuve :
$\pi_{titre}(Film) - \pi_{titre}(\sigma_{acteur \neq' Deneuve'}(Film))$

```
SELECT Film.titre
FROM Film
WHERE Film.titre NOT IN
                  (SELECT Film.titre
                  FROM Film
                  WHERE Film.acteur <> 'Deneuve');
```

Pour calculer le résultat de la sous requête, on commence par calculer $\sigma_{acteur \neq' Deneuve'}(Film)$:

Film

réalisateur	titre	acteur
Varda	Les Créatures	Deneuve
Varda	Les Créatures	Piccoli
Chytilova	Sedmikrasky	Cerhova
Bunuel	Belle de jour	Deneuve
Varda	Cléo de 5 à 7	Marchand
Polanski	Repulsion	Deneuve

$\sigma_{acteur \neq' Deneuve'}(Film)$

réalisateur	titre	acteur
~~Varda~~	~~Les Créatures~~	~~Deneuve~~
Varda	Les Créatures	Piccoli
Chytilova	Sedmikrasky	Cerhova
~~Bunuel~~	~~Belle de jour~~	~~Deneuve~~
Varda	Cléo de 5 à 7	Marchand
~~Polanski~~	~~Repulsion~~	~~Deneuve~~

Sémantique des sous requêtes en IN et NOT IN

Les films dans lesquels ne joue que Deneuve :
$\pi_{titre}(Film) - \pi_{titre}(\sigma_{acteur \neq' Deneuve'}(Film))$

```
SELECT Film.titre
FROM Film
WHERE Film.titre NOT IN
                  (SELECT Film.titre
                  FROM Film
                  WHERE Film.acteur <> 'Deneuve');
```

Pour calculer le résultat de la sous requête, on commence par calculer $\sigma_{acteur \neq' Deneuve'}(Film)$:

Film

réalisateur	titre	acteur
Varda	Les Créatures	Deneuve
Varda	Les Créatures	Piccoli
Chytilova	Sedmikrasky	Cerhova
Bunuel	Belle de jour	Deneuve
Varda	Cléo de 5 à 7	Marchand
Polanski	Repulsion	Deneuve

$\sigma_{acteur \neq' Deneuve'}(Film)$

réalisateur	titre	acteur
Varda	Les Créatures	Piccoli
Chytilova	Sedmikrasky	Cerhova
Varda	Cléo de 5 à 7	Marchan d

Sémantique des sous requêtes en IN et NOT IN

Les films dans lesquels ne joue que Deneuve :
$\pi_{titre}(Film) - \pi_{titre}(\sigma_{acteur \neq' Deneuve'}(Film))$

```
SELECT Film.titre
FROM Film
WHERE Film.titre NOT IN
                  (SELECT Film.titre
                  FROM Film
                  WHERE Film.acteur <> 'Deneuve');
```

On calcule maintenant le résultat de la sous-requête :

$$\Pi_{titre}(\sigma_{acteur \neq' Deneuve'}(Film))$$

réalisateur	titre	acteur
Varda	Les Créatures	Piccoli
Chytilova	Sedmikrasky	Cerhova
Varda	Cléo de 5 à 7	Marchand

Sémantique des sous requêtes en IN et NOT IN

Les films dans lesquels ne joue que Deneuve :
$\pi_{titre}(Film) - \pi_{titre}(\sigma_{acteur \neq' Deneuve'}(Film))$

```
SELECT Film.titre
FROM Film
WHERE Film.titre NOT IN
                  (SELECT Film.titre
                  FROM Film
                  WHERE Film.acteur <> 'Deneuve');
```

On projette ensuite également l'attribut titre sur la table Film :

$\Pi_{titre}(Film)$

réalisateur	titre	acteur
Varda	Les Créatures	Piccoli
Chytilova	Sedmikrasky	Cerhova
Bunuel	Belle de jour	Deneuve
Varda	Cléo de 5 à 7	Marchand
Polanski	Repulsion	Deneuve

$\Pi_{titre}(\sigma_{acteur \neq' Deneuve'}(Film))$

réalisateur	titre	acteur
Varda	Les Créatures	Piccoli
Chytilova	Sedmikrasky	Cerhova
Varda	Cléo de 5 à 7	Marchand

Sémantique des sous requêtes en IN et NOT IN

Les films dans lesquels ne joue que Deneuve :
$\pi_{titre}(Film) - \pi_{titre}(\sigma_{acteur \neq' Deneuve'}(Film))$

```
SELECT Film.titre
FROM Film
WHERE Film.titre NOT IN
                  (SELECT Film.titre
                  FROM Film
                  WHERE Film.acteur <> 'Deneuve');
```

On calcule la différence ensembliste de ces deux tables :

$\Pi_{titre}(Film)$

titre
Les Créatures
Sedmikrasky
Belle de jour
Cléo de 5 à 7
Repulsion

différence ensembliste : —

$\Pi_{titre}(\sigma_{acteur \neq' Deneuve'}(Film))$

titre
Les Créatures
Sedmikrasky
Cléo de 5 à 7

=

tableau résultat :

titre
~~Les Créatures~~
~~Sedmikrasky~~
Belle de jour
~~Cléo de 5 à 7~~
Repulsion

Sémantique des sous requêtes en IN et NOT IN

Les films dans lesquels ne joue que Deneuve :
$\pi_{titre}(Film) - \pi_{titre}(\sigma_{acteur \neq' Deneuve'}(Film))$

```
SELECT Film.titre
FROM Film
WHERE Film.titre NOT IN
                (SELECT Film.titre
                FROM Film
                WHERE Film.acteur <> 'Deneuve');
```

Le tableau résultat de $\pi_{titre}(Film) - \pi_{titre}(\sigma_{acteur \neq' Deneuve'}(Film))$ est :

$\Pi_{titre}(Film)$

titre
Les Créatures
Sedmikrasky
Belle de jour
Cléo de 5 à 7
Repulsion

différence ensembliste : —

$\Pi_{titre}(\sigma_{acteur \neq' Deneuve'}(Film))$

titre
Les Créatures
Sedmikrasky
Cléo de 5 à 7

=

tableau résultat :

titre
Belle de jour
Repulsion

Les quantificateurs ensemblistes

- Une condition peut porter sur l'existence (`EXISTS`) d'au moins une ligne dans le résultat d'une sous requête
- Les films qui passent dans un cinéma :

```
SELECT  Film.titre FROM Film
WHERE EXISTS (SELECT *
                    FROM Séance
                    WHERE  Film.titre=Séance.titre) ;
```

$$\pi_{titre}(Film \bowtie Seance)$$

- Sémantique : `EXISTS (SELECT ...)` est vrai si `(SELECT ...)` retourne une table non vide

Les quantificateurs ensemblistes

- Une condition peut porter sur l'existence (EXISTS) d'au moins une ligne dans le résultat d'une sous requête
- Les films qui passent dans un cinéma :

```
SELECT  Film.titre FROM Film
WHERE EXISTS (SELECT *
                    FROM Séance
                    WHERE  Film.titre=Séance.titre) ;
```

$$\pi_{titre}(Film \bowtie Seance)$$

- La sous requête désigne, pour l'ensemble des tuples de Film, l'ensemble des tuples de Séance qui y font référence (i.e., Film.titre=Séance.titre)
- La requête principale ne retient que les films pour lesquels cet ensemble est non vide

Sémantique des requêtes en EXISTS

Les films qui passent dans un cinéma : $\pi_{titre}(Film \bowtie Seance)$

```
SELECT  Film.titre FROM Film
WHERE EXISTS (SELECT *
                FROM Séance
                WHERE  Film.titre=Séance.titre) ;
```

On calcule d'abord la jointure de `Film` et `Seance` sur l'attribut `titre` :

Film

réalisateur	titre	acteur
Varda	Les Créatures	Piccoli
Chytilova	Sedmikrasky	Cerhova
Bunuel	Belle de jour	Deneuve
Varda	Cléo de 5 à 7	Marchand
Polanski	Repulsion	Deneuve

$\bowtie$
Film.titre=Séance.titre

Séance

titre	cinéma
Les Créatures	Le Champo
Belle de jour	Le Champo
Cléo de 5 à 7	Odéon
Repulsion	Odéon
Belle de jour	Odéon

Sémantique des requêtes en EXISTS

Les films qui passent dans un cinéma : $\pi_{titre}(Film \bowtie Seance)$

```
SELECT  Film.titre FROM Film
WHERE EXISTS (SELECT *
                FROM Séance
                WHERE  Film.titre=Séance.titre) ;
```

On calcule d'abord la jointure de `Film` et `Seance` sur l'attribut `titre` :

(Film ⋈ Séance)

Film.titre=Séance.titre

réalisateur	Film.titre	Séance.titre	acteur	cinéma
Varda	Les Créatures	Les Créatures	Piccoli	Le Champo
Bunuel	Belle de jour	Belle de jour	Deneuve	Le Champo
Varda	Cléo de 5 à 7	Cléo de 5 à 7	Marchand	Odéon
Polanski	Répulsion	Répulsion	Deneuve	Odéon
Bunuel	Belle de jour	Belle de jour	Deneuve	Odéon

Sémantique des requêtes en EXISTS

Les films qui passent dans un cinéma : $\pi_{titre}(Film \bowtie Seance)$

```
SELECT  Film.titre FROM Film
WHERE EXISTS (SELECT *
                FROM Séance
                WHERE  Film.titre=Séance.titre) ;
```

On projette ensuite l'attribut `Film.titre` :

$$\Pi_{Film.titre}(\text{Film} \underset{\text{Film.titre=Séance.titre}}{\bowtie} \text{Séance})$$

réalisateur	Film.titre	Séance.titre	acteur	cinéma
Varda	Les Créatures	Les Créatures	Piccoli	Le Champo
Bunuel	Belle de jour	Belle de jour	Deneuve	Le Champo
Varda	Cléo de 5 à 7	Cléo de 5 à 7	Marchand	Odéon
Polanski	Répulsion	Répulsion	Deneuve	Odéon
Bunuel	Belle de jour	Belle de jour	Deneuve	Odéon

Sémantique des requêtes en EXISTS

Les films qui passent dans un cinéma : $\pi_{titre}(Film \bowtie Seance)$

```
SELECT  Film.titre FROM Film
WHERE EXISTS (SELECT *
                FROM Séance
                WHERE  Film.titre=Séance.titre) ;
```

On projette ensuite l'attribut `Film.titre` :

$$\Pi_{Film.titre}(\text{Film} \underset{\text{Film.titre=Séance.titre}}{\bowtie} \text{Séance})$$

Film.titre
Les Créatures
Belle de jour
Cléo de 5 à 7
Répulsion
Belle de jour

Tous les titres, sauf Sedmikrasky, sont retournés.

Sémantique des requêtes en EXISTS

Une requête SQL équivalente plus naturelle : $\pi_{titre}(Film \bowtie Seance)$

```
SELECT  Film.titre FROM Film, Séance
WHERE Film.titre=Séance.titre ;
```

Les films qui passent dans un cinéma :

$$\Pi_{Film.titre}(\text{Film} \underset{\text{Film.titre=Séance.titre}}{\bowtie} \text{Séance})$$

Film.titre
Les Créatures
Belle de jour
Cléo de 5 à 7
Répulsion
Belle de jour

- En fait `EXISTS` est surtout utilisé avec la négation
- En effet, `EXISTS` sans `NOT` se simule facilement avec la jointure (plus efficace)

Les quantificateurs ensemblistes

- Une condition peut porter sur l'inexistence (`NOT EXISTS`) d'au moins une ligne dans le résultat d'une sous requête
- Les films qui ne passent dans aucun cinéma :

```
SELECT  Film.titre FROM Film
WHERE NOT EXISTS (SELECT *
                  FROM Séance
                  WHERE  Film.titre=Séance.titre) ;
```

$$\pi_{titre}(Film) \; - \; \pi_{titre}(Seance \bowtie Film)$$

- Sémantique : `NOT EXISTS (SELECT ...)` est vrai si `(SELECT ...)` retourne une table vide

Les quantificateurs ensemblistes

- Les films qui ne passent dans aucun cinéma :

```
SELECT  Film.titre FROM Film
WHERE NOT EXISTS (SELECT *
                  FROM Séance
                  WHERE  Film.titre=Seance.titre) ;
```

$$\pi_{titre}(Film) \; - \; \pi_{titre}(Seance \bowtie Film)$$

- La sous requête désigne, pour l'ensemble des tuples de Film, l'ensemble des tuples de Séance qui y font référence (i.e., Film.titre=Séance.titre)
- La requête principale ne retient que les films pour lesquels cet ensemble est vide

Sémantique des requêtes en NOT EXISTS

Les films qui ne passent dans aucun cinéma : $\pi_{titre}(Film) - \pi_{titre}(Seance \bowtie Film)$

```
SELECT  Film.titre FROM Film
WHERE NOT EXISTS (SELECT *
                  FROM Séance
                  WHERE  Film.titre=Seance.titre) ;
```

On calcule d'abord la jointure de `Film` et `Seance` sur l'attribut `titre` :

Film

réalisateur	titre	acteur
Varda	Les Créatures	Piccoli
Chytilova	Sedmikrasky	Cerhova
Bunuel	Belle de jour	Deneuve
Varda	Cléo de 5 à 7	Marchand
Polanski	Repulsion	Deneuve

$\bowtie$
Film.titre=Séance.titre

Séance

titre	cinéma
Les Créatures	Le Champo
Belle de jour	Le Champo
Cléo de 5 à 7	Odéon
Repulsion	Odéon
Belle de jour	Odéon

Sémantique des requêtes en NOT EXISTS

Les films qui ne passent dans aucun cinéma : $\pi_{titre}(Film) - \pi_{titre}(Seance \bowtie Film)$

```
SELECT  Film.titre FROM Film
WHERE NOT EXISTS (SELECT *
                  FROM Séance
                  WHERE  Film.titre=Seance.titre) ;
```

On calcule d'abord la jointure de `Film` et `Seance` sur l'attribut `titre` :

(Film ⋈ Séance)

Film.titre=Séance.titre

réalisateur	Film.titre	Séance.titre	acteur	cinéma
Varda	Les Créatures	Les Créatures	Piccoli	Le Champo
Bunuel	Belle de jour	Belle de jour	Deneuve	Le Champo
Varda	Cléo de 5 à 7	Cléo de 5 à 7	Marchand	Odéon
Polanski	Répulsion	Répulsion	Deneuve	Odéon
Bunuel	Belle de jour	Belle de jour	Deneuve	Odéon

Sémantique des requêtes en NOT EXISTS

Les films qui ne passent dans aucun cinéma : $\pi_{titre}(Film) - \pi_{titre}(Seance \bowtie Film)$

```
SELECT  Film.titre FROM Film
WHERE NOT EXISTS (SELECT *
                  FROM Séance
                  WHERE  Film.titre=Seance.titre) ;
```

On projette ensuite l'attribut `Film.titre` :

$$\Pi_{Film.titre}(\text{Film} \underset{\text{Film.titre=Séance.titre}}{\bowtie} \text{Séance})$$

réalisateur	Film.titre	Séance.titre	acteur	cinéma
Varda	Les Créatures	Les Créatures	Piccoli	Le Champo
Bunuel	Belle de jour	Belle de jour	Deneuve	Le Champo
Varda	Cléo de 5 à 7	Cléo de 5 à 7	Marchand	Odéon
Polanski	Répulsion	Répulsion	Deneuve	Odéon
Bunuel	Belle de jour	Belle de jour	Deneuve	Odéon

Sémantique des requêtes en NOT EXISTS

Les films qui ne passent dans aucun cinéma : $\pi_{titre}(Film) - \pi_{titre}(Seance \bowtie Film)$

```
SELECT  Film.titre FROM Film
WHERE NOT EXISTS (SELECT *
                  FROM Séance
                  WHERE  Film.titre=Seance.titre) ;
```

On projette également l'attribut `titre` sur la table Film :

$\Pi_{titre}(Film)$

réalisateur	titre	acteur
Varda	Les Créatures	Piccoli
Chytilova	Sedmikrasky	Cerhova
Bunuel	Belle de jour	Deneuve
Varda	Cléo de 5 à 7	Marchand
Polanski	Repulsion	Deneuve

Sémantique des requêtes en NOT EXISTS

Les films qui ne passent dans aucun cinéma : $\pi_{titre}(Film) \ - \ \pi_{titre}(Seance \bowtie Film)$

```
SELECT  Film.titre FROM Film
WHERE NOT EXISTS (SELECT *
                  FROM Séance
                  WHERE  Film.titre=Seance.titre) ;
```

On calcule maintenant la différence ensembliste de ces deux tables :

$\Pi_{titre}(Film)$

réalisateur	titre	acteur
Varda	Les Créatures	Piccoli
Chytilova	Sedmikrasky	Cerhova
Bunuel	Belle de jour	Deneuve
Varda	Cléo de 5 à 7	Marchand
Polanski	Repulsion	Deneuve

différence ensembliste :

—

$\Pi_{Film.titre}(\text{Film} \bowtie_{\text{Film.titre=Séance.titre}} \text{Séance})$

Film.titre
Les Créatures
Belle de jour
Cléo de 5 à 7
Répulsion
Belle de jour

Sémantique des requêtes en NOT EXISTS

Les films qui ne passent dans aucun cinéma : $\pi_{titre}(Film) \ - \ \pi_{titre}(Seance \bowtie Film)$

```
SELECT  Film.titre FROM Film
WHERE NOT EXISTS (SELECT *
                  FROM Séance
                  WHERE  Film.titre=Seance.titre) ;
```

On calcule maintenant la différence ensembliste de ces deux tables :

$\Pi_{titre}(Film)$

titre
Les Créatures
Sedmikrasky
Belle de jour
Cléo de 5 à 7
Repulsion

différence ensembliste :

−

$\Pi_{Film.titre}$ (Film $\bowtie_{\text{Film.titre=Séance.titre}}$ Séance)

Film.titre
Les Créatures
Belle de jour
Cléo de 5 à 7
Répulsion
Belle de jour

=

tableau résultat :

titre
Sedmikrasky

Remarque : en toute rigueur, il faudrait ici opérer un renommage de Film.titre en titre.

Sémantique des requêtes en NOT EXISTS

Une requête d'algèbre relationnelle équivalente plus simple :

$$\pi_{titre}(Film) \ - \ \pi_{titre}(Seance)$$

Une alternative SQL plus simple :

```
SELECT titre FROM Film
WHERE titre NOT IN
        (SELECT titre FROM Séance) ;
```

Cette requête SQL est plus simple car elle est non corrélée.

Exists et sous requêtes corrélées

Une sous requête est corrélée lorsque l'évaluation de la sous-requête dépend de celle de la requête externe.

- Les cinémas qui ne passent aucun film de Polanski :

```
SELECT  Séance.cinéma FROM Séance
WHERE NOT EXISTS (SELECT *
                  FROM Film
                  WHERE  Séance.titre=Film.titre
                  AND Film.réalisateur=Polanski) ;
```

$$\pi_{cinema}(Seance) - \pi_{cinema}(\sigma_{realisateur='Polanski'}(Film) \bowtie Seance)$$

La corrélation est particulièrement utile avec la condition `NOT EXISTS`, mais possible avec toutes les autres.

La négation dans SQL

- Deux mécanismes principaux : les sous requêtes et les expressions ensemblistes
- Sous-requêtes souvent plus naturelles
- Syntaxe de SQL pour $R - S$:

  `R EXCEPT S`
- Syntaxe de SQL pour $R \cap S$ (pas de négation ici, mais rappel) :

  `R INTERSECT S`
- Trouver tous les acteurs qui

- ne sont pas réalisateurs :

```
SELECT acteur AS personne
FROM Film
 EXCEPT
SELECT realisateur AS personne
FROM Film;
```

- sont aussi réalisateurs :

```
SELECT acteur AS personne
FROM Film
 INTERSECT
SELECT realisateur AS personne
FROM Film;
```

La négation dans SQL

- Deux mécanismes principaux : les sous requêtes et les expressions ensemblistes avec `EXCEPT`
- Sous-requêtes souvent plus naturelles
- Syntaxe de SQL pour $R \cap S$:

  `R INTERSECT S`
- Syntaxe de SQL pour $R - S$:

  `R EXCEPT S`
- Trouver tous les acteurs qui

- ne sont pas réalisateurs :

$$\pi_{personne}(\rho_{personne \leftarrow acteur}(Film) - \rho_{personne \leftarrow realisateur}(Film))$$

- sont aussi réalisateurs :

$$\pi_{personne}(\rho_{personne \leftarrow acteur}(Film) \cap \rho_{personne \leftarrow realisateur}(Film))$$

La négation dans SQL

- Deux mécanismes principaux : les sous requêtes et les expressions ensemblistes
- Sous-requêtes souvent plus naturelles
- Syntaxe de SQL pour $R - S$:

  `R EXCEPT S`
- Syntaxe de SQL pour $R \cap S$ (rappel) :

  `R INTERSECT S`
- Trouver tous les acteurs qui

- ne sont pas réalisateurs :

```
SELECT acteur
FROM Film
WHERE acteur NOT IN
 (SELECT realisateur
 FROM Film);
```

- sont aussi réalisateurs :

```
SELECT acteur
FROM Film
WHERE acteur IN
(SELECT realisateur
FROM Film);
```

NOT IN, NOT EXISTS et EXCEPT

- Les requêtes qui nécessitent une forme de négation peuvent être exprimées aussi bien avec `EXCEPT`, `NOT IN` et `NOT EXISTS`
- En revanche, dans certains cas il est difficile de faire l'impasse sur la corrélation dans les sous-requêtes
- Trouver les acteurs qui n'ont joué que dans un seul film :

```
SELECT acteur
FROM Film
EXCEPT
(SELECT F1.acteur
FROM Film F1, Film F2
WHERE F1.acteur=F2.acteur and
F1.titre <> F2.titre);
```

NOT IN, NOT EXISTS et EXCEPT

- Les requêtes qui nécessitent une forme de négation peuvent être exprimées aussi bien avec `EXCEPT`, `NOT IN` et `NOT EXISTS`
- En revanche, dans certains cas il est difficile de faire l'impasse sur la corrélation dans les sous-requêtes
- Trouver les acteurs qui n'ont joué que dans un seul film :

```
SELECT F1.acteur
FROM Film F1
WHERE F1.acteur NOT IN
                (SELECT F2.acteur
                FROM Film F2
                WHERE F1.acteur=F2.acteur and
                F1.titre <> F2.titre);
```

NOT IN, NOT EXISTS et EXCEPT

- Les requêtes qui nécessitent une forme de négation peuvent être exprimées aussi bien avec `EXCEPT`, `NOT IN` et `NOT EXISTS`
- En revanche, dans certains cas il est difficile de faire l'impasse sur la corrélation dans les sous-requêtes
- Trouver les acteurs qui n'ont joué que dans un seul film :

```
SELECT F1.acteur
FROM Film F1
WHERE NOT EXISTS
      (SELECT *
      FROM Film F2
      WHERE F1.acteur=F2.acteur and
      F1.titre <> F2.titre);
```

EXISTS et NOT EXISTS sans corrélation

Une requête en EXISTS ou NOT EXISTS perd tout son sens sans corrélation :

```
SELECT acteur
FROM Film
WHERE NOT EXISTS
      (SELECT *
      FROM Film);
```

Ne retourne jamais rien (pas très intéressant).

EXISTS et NOT EXISTS sans corrélation

Une requête en EXIST ou NON EXISTS perd tout son sens sans corrélation :

```
SELECT acteur
FROM Film
WHERE NOT EXISTS
      (SELECT *
      FROM Seance);
```

Retourne l'intégralité des acteurs de la table `Film` si la table `Seance` est vide (pas très intéressant non plus...).

Requêtes typiques nécessitant de la négation

- Les cinémas qui ne passent aucun film de Polanski :

```
SELECT  Séance.cinéma FROM Séance
WHERE NOT EXISTS (SELECT *
                  FROM Film
                  WHERE  Séance.titre=Film.titre
                  AND Film.réalisateur=Polanski) ;
```

Requêtes typiques nécessitant de la négation

- Souvent la négation est un peu cachée...
- Les cinémas qui ne passent que des films de Polanski :

```
SELECT  Séance.cinéma FROM Séance
WHERE NOT EXISTS (SELECT *
                  FROM Film
                  WHERE  Séance.titre=Film.titre
                  AND Film.réalisateur<>Polanski) ;
```

$\pi_{cinema}(Seance) - \pi_{cinema}(\sigma_{realisateur<>Polanski}(Film) \bowtie Seance)$

- = les cinémas qui ne jouent aucun film d'un réalisateur autre que Polanski

Requêtes dans lesquelles la négation est "cachée"

- SQL n'a pas d'opérateur explicite pour exprimer qu'une propriété est vérifiée par tous les éléments d'un ensemble (ex : les cinémas dans lesquels tous les films projetés sont de Polanski)
- SQL fournit un moyen explicite uniquement pour exprimer qu'il existe un élément d'un ensemble qui satisfait une propriété (ex : il existe un film dont le réalisateur n'est pas Polanski)
- Mais la combinaison de la négation et de "il existe" peut exprimer "pour tout"

Tous les films sont de Polanski

$\leftrightarrow$

Il n'existe pas de film dont le réalisateur n'est pas Polanski

Requêtes dans lesquelles la négation est "cachée"

- Parfois plusieurs négations imbriquées sont "cachées"
- Cas des requêtes qui demandent de vérifier l'inclusion ou l'égalité de deux ensembles

$$A \subseteq B \text{ ou } A = B$$

Exemples :

- Les cinémas qui passent tous les films de Polanski :
  - vérifier : films de Polanski $\subseteq$ films qui passent dans le cinéma
- Les cinémas qui passent exactement les films de Polanski (i.e. tous ses films et aucun autre)
  - vérifier : films de Polanski $=$ films qui passent dans le cinéma

Requêtes dans lesquelles la négation est "cachée"

Reformulation d'inclusion et égalité d'ensembles en termes de négations imbriquées

chacune des deux négations peut être exprimée par `NOT EXISTS` ou `NOT IN` ou `EXCEPT`

$A \subseteq B$

équivalent à la condition

```
NOT EXISTS (A EXCEPT B)
```

$A = B$

équivalent à la condition

$(A \subseteq B)$ AND $(B \subseteq A)$

Requêtes dans lesquelles la négation est "cachée"

- Les cinémas qui ne passent que des films de Polanski :
- vérifier : films qui passent dans le cinéma $\subseteq$ films de Polanski

```
SELECT Séance.cinéma FROM Séance
WHERE NOT EXISTS (SELECT *
                  FROM Film
                  WHERE  Séance.titre NOT IN
                       (SELECT titre
                       FROM Film
                       WHERE réalisateur=Polanski)) ;
```

Requêtes dans lesquelles la négation est "cachée"

- En fait cette requête se simplifie en poussant comme suit la négation dans la condition de sélection du `WHERE`
- Les cinémas qui ne passent que des films de Polanski :

```
SELECT  Séance.cinéma FROM Séance
WHERE NOT EXISTS (SELECT *
                  FROM Film
                  WHERE  Séance.titre=Film.titre
                  AND Film.réalisateur<>Polanski) ;
```

Conditions de totalité (pour tout)

- Celle-ci en revanche ne peut se simplifier de la même manière...
- Les réalisateurs dont les films passent dans tous les cinémas (au moins un film dans chaque cinéma) :

```
SELECT F1.realisateur
FROM Film F1
WHERE NOT EXISTS (SELECT S.cinema
                  FROM Séance S
                  WHERE NOT EXISTS (SELECT F2.realisateur
                                    FROM Film F2
                                    WHERE F2.titre=S.titre
                                        AND
                                    F1.realisateur=F2.realisateur))
```

- Le réalisateur F1 est retenu
- s'il n'existe pas de cinéma S
- tel qu'il n'existe pas de film F2 du réalisateur F1
- tel que le film F2 est joué dans le cinéma S

Pour tout et la négation dans SQL

Même requête avec EXCEPT

```
SELECT F.realisateur
FROM Film F
WHERE NOT EXISTS (SELECT S.cinema
                  FROM Séance S
                    EXCEPT
                  SELECT S1.cinema
                  FROM Séance S1, Film F1
                  WHERE S1.titre=F1.titre
                  AND F1.realisateur=F.realisateur)
```

$$\pi_{réalisateur}(F) - \pi_{réalisateur}((\pi_{cinéma}(S) \times \pi_{réalisateur}(F)) - \pi_{cinéma,réalisateur}(F \bowtie S))$$

Sémantique des requêtes avec conditions universelles

$$\pi_{réalisateur}(F) - \pi_{réalisateur}((\pi_{cinéma}(S) \times \pi_{réalisateur}(F)) - \pi_{cinéma,réalisateur}(F \bowtie S))$$

```
SELECT F.realisateur
FROM Film F
WHERE NOT EXISTS (SELECT S.cinema
                  FROM Séance S
                    EXCEPT
                  SELECT S1.cinema
                  FROM Séance S1, Film F1
                  WHERE S1.titre=F1.titre
                  AND F1.realisateur=F.realisateur)
```

On commence par calculer la jointure de Film et Séance :

Film

réalisateur	titre	acteur
Varda	Les Créatures	Piccoli
Chytilova	Sedmikrasky	Cerhova
Bunuel	Belle de jour	Deneuve
Varda	Cléo de 5 à 7	Marchand
Polanski	Repulsion	Deneuve

$\bowtie$

Séance

titre	cinéma
Les Créatures	Le Champo
Belle de jour	Le Champo
Cléo de 5 à 7	Odéon
Repulsion	Odéon
Belle de jour	Odéon

Sémantique des requêtes avec conditions universelles

$$\pi_{réalisateur}(F) - \pi_{réalisateur}((\pi_{cinéma}(S) \times \pi_{réalisateur}(F)) - \pi_{cinéma,réalisateur}(F \bowtie S))$$

```
SELECT F.realisateur
FROM Film F
WHERE NOT EXISTS (SELECT S.cinema
                  FROM Séance S
                    EXCEPT
                  SELECT S1.cinema
                  FROM Séance S1, Film F1
                  WHERE S1.titre=F1.titre
                  AND F1.realisateur=F.realisateur)
```

On commence par calculer la jointure de Film et Séance :

Film ⋈ Séance

réalisateur	titre	acteur	cinéma
Varda	Les Créatures	Piccoli	Le Champo
Bunuel	Belle de jour	Deneuve	Le Champo
Varda	Cléo de 5 à 7	Marchand	Odéon
Polanski	Répulsion	Deneuve	Odéon
Bunuel	Belle de jour	Deneuve	Odéon

Sémantique des requêtes avec conditions universelles

$$\pi_{réalisateur}(F) - \pi_{réalisateur}((\pi_{cinéma}(S) \times \pi_{réalisateur}(F)) - \pi_{cinéma,réalisateur}(F \bowtie S))$$

```
SELECT F.realisateur
FROM Film F
WHERE NOT EXISTS (SELECT S.cinema
                  FROM Séance S
                    EXCEPT
                  SELECT S1.cinema
                  FROM Séance S1, Film F1
                  WHERE S1.titre=F1.titre
                  AND F1.realisateur=F.realisateur)
```

On projette les attributs `realisateur` et `cinema` :

$$\Pi_{realisateur,cinema}(\text{Film} \bowtie \text{Séance})$$

réalisateur	titre	acteur	cinéma
Varda	Les Créatures	Piccoli	Le Champo
Bunuel	Belle de jour	Deneuve	Le Champo
Varda	Cléo de 5 à 7	Marchand	Odéon
Polanski	Répulsion	Deneuve	Odéon
Bunuel	Belle de jour	Deneuve	Odéon

Sémantique des requêtes avec conditions universelles

$$\pi_{réalisateur}(F) - \pi_{réalisateur}((\pi_{cinéma}(S) \times \pi_{réalisateur}(F)) - \pi_{cinéma,réalisateur}(F \bowtie S))$$

```
SELECT F.realisateur
FROM Film F
WHERE NOT EXISTS (SELECT S.cinema
                  FROM Séance S
                    EXCEPT
                  SELECT S1.cinema
                  FROM Séance S1, Film F1
                  WHERE S1.titre=F1.titre
                  AND F1.realisateur=F.realisateur)
```

On projette les attributs `realisateur` et `cinema` :

$$\Pi_{realisateur,cinema}(\text{Film} \bowtie \text{Séance})$$

réalisateur	cinéma
Varda	Le Champo
Bunuel	Le Champo
Varda	Odéon
Polanski	Odéon
Bunuel	Odéon

Sémantique des requêtes avec conditions universelles

$\pi_{réalisateur}(F) - \pi_{réalisateur}((\pi_{cinéma}(S) \times \pi_{réalisateur}(F)) - \pi_{cinéma,réalisateur}(F \bowtie S))$

```
SELECT F.realisateur
FROM Film F
WHERE NOT EXISTS (SELECT S.cinema
                  FROM Séance S
                    EXCEPT
                  SELECT S1.cinema
                  FROM Séance S1, Film F1
                  WHERE S1.titre=F1.titre
                  AND F1.realisateur=F.realisateur)
```

On s'attaque maintenant à $\pi_{cinéma}(Seance) \times \pi_{réalisateur}(Film)$:

$\Pi_{realisateur}$(Film)

réalisateur	titre	acteur
Varda	Les Créatures	Piccoli
Chytilova	Sedmikrasky	Cerhova
Bunuel	Belle de jour	Deneuve
Varda	Cléo de 5 à 7	Marchand
Polanski	Repulsion	Deneuve

Produit cartésien : ×

Π_{cinema}(Séance)

titre	cinéma
Les Créatures	Le Champo
Belle de jour	Le Champo
Cléo de 5 à 7	Odéon
Repulsion	Odéon
Belle de jour	Odéon

Sémantique des requêtes avec conditions universelles

$\pi_{réalisateur}(F) - \pi_{réalisateur}((\pi_{cinéma}(S) \times \pi_{réalisateur}(F)) - \pi_{cinéma,réalisateur}(F \bowtie S))$

```
SELECT F.realisateur
FROM Film F
WHERE NOT EXISTS (SELECT S.cinema
                  FROM Séance S
                    EXCEPT
                  SELECT S1.cinema
                  FROM Séance S1, Film F1
                  WHERE S1.titre=F1.titre
                  AND F1.realisateur=F.realisateur)
```

On s'attaque maintenant à $\pi_{cinéma}(Seance) \times \pi_{réalisateur}(Film)$:

$\Pi_{realisateur}$(Film) Π_{cinema}(Séance)

réalisateur
Varda
Chytilova
Bunuel
Varda
Polanski

Produit cartésien : ×

cinéma
Le Champo
Le Champo
Odéon
Odéon
Odéon

Sémantique des requêtes avec conditions universelles

$\pi_{réalisateur}(F) - \pi_{réalisateur}((\pi_{cinéma}(S) \times \pi_{réalisateur}(F)) - \pi_{cinéma,réalisateur}(F \bowtie S))$

```
SELECT F.realisateur
FROM Film F
WHERE NOT EXISTS (SELECT S.cinema
                  FROM Séance S
                    EXCEPT
                  SELECT S1.cinema
                  FROM Séance S1, Film F1
                  WHERE S1.titre=F1.titre
                  AND F1.realisateur=F.realisateur)
```

On s'attaque maintenant à $\pi_{cinéma}(Seance) \times \pi_{réalisateur}(Film)$:

Produit cartésien :

$$\Pi_{realisateur}(\text{Film}) \times \Pi_{cinema}(\text{Séance})$$

réalisateur	cinéma
Varda	Le Champo
Varda	Odéon
Chytilova	Le Champo
Chytilova	Odéon
Bunuel	Le Champo
Bunuel	Odéon
Polanski	Le Champo
Polansk	Odéon

Sémantique des requêtes avec conditions universelles

$\pi_{réalisateur}(F) - \pi_{réalisateur}((\pi_{cinéma}(S) \times \pi_{réalisateur}(F)) - \pi_{cinéma,réalisateur}(F \bowtie S))$

```
SELECT F.realisateur
FROM Film F
WHERE NOT EXISTS (SELECT S.cinema
                  FROM Séance S
                    EXCEPT
                  SELECT S1.cinema
                  FROM Séance S1, Film F1
                  WHERE S1.titre=F1.titre
                  AND F1.realisateur=F.realisateur)
```

On calcule maintenant $(\pi_{cinéma}(S) \times \pi_{réalisateur}(F)) - \pi_{cinéma,réalisateur}(F \bowtie S)$:

Produit cartésien :

$\Pi_{realisateur}(\text{Film}) \times \Pi_{cinema}(\text{Séance})$

réalisateur	cinéma
Varda	Le Champo
Varda	Odéon
Chytilova	Le Champo
Chytilova	Odéon
Bunuel	Le Champo
Bunuel	Odéon
Polanski	Le Champo
Polanski	Odéon

différence ensembliste :

−

$\Pi_{realisateur,cinema}(\text{Film} \bowtie \text{Séance})$

réalisateur	cinéma
Varda	Le Champo
Bunuel	Le Champo
Varda	Odéon
Polanski	Odéon
Bunuel	Odéon

Sémantique des requêtes avec conditions universelles

$$\pi_{réalisateur}(F) - \pi_{réalisateur}((\pi_{cinéma}(S) \times \pi_{réalisateur}(F)) - \pi_{cinéma,réalisateur}(F \bowtie S))$$

```
SELECT F.realisateur
FROM Film F
WHERE NOT EXISTS (SELECT S.cinema
                  FROM Séance S
                    EXCEPT
                  SELECT S1.cinema
                  FROM Séance S1, Film F1
                  WHERE S1.titre=F1.titre
                  AND F1.realisateur=F.realisateur)
```

On calcule maintenant $(\pi_{cinéma}(S) \times \pi_{réalisateur}(F)) - \pi_{cinéma,réalisateur}(F \bowtie S)$:

Produit cartésien :

$$\Pi_{realisateur}(\text{Film}) \times \Pi_{cinema}(\text{Séance})$$

réalisateur	cinéma
~~Varda~~	~~Le Champo~~
~~Varda~~	~~Odéon~~
Chytilova	Le Champo
Chytilova	Odéon
~~Bunuel~~	~~Le Champo~~
~~Bunuel~~	~~Odéon~~
Polanski	Le Champo
~~Polanski~~	~~Odéon~~

différence ensembliste :

−

$$\Pi_{realisateur,cinema}(\text{Film} \bowtie \text{Séance})$$

réalisateur	cinéma
Varda	Le Champo
Bunuel	Le Champo
Varda	Odéon
Polanski	Odéon
Bunuel	Odéon

Sémantique des requêtes avec conditions universelles

$\pi_{réalisateur}(F) - \pi_{réalisateur}((\pi_{cinéma}(S) \times \pi_{réalisateur}(F)) - \pi_{cinéma,réalisateur}(F \bowtie S))$

```
SELECT F.realisateur
FROM Film F
WHERE NOT EXISTS (SELECT S.cinema
                  FROM Séance S
                    EXCEPT
                  SELECT S1.cinema
                  FROM Séance S1, Film F1
                  WHERE S1.titre=F1.titre
                  AND F1.realisateur=F.realisateur)
```

On calcule maintenant $(\pi_{cinéma}(S) \times \pi_{réalisateur}(F)) - \pi_{cinéma,réalisateur}(F \bowtie S)$:

$$(\Pi_{realisateur}(Film) \times \Pi_{cinema}(Seance)) - (\Pi_{realisateur,cinema}(Film \bowtie Seance))$$

réalisateur	cinéma
Chytilova	Le Champo
Chytilova	Odéon
Polanski	Le Champo

Sémantique des requêtes avec conditions universelles

$$\pi_{réalisateur}(F) - \pi_{réalisateur}((\pi_{cinéma}(S) \times \pi_{réalisateur}(F)) - \pi_{cinéma,réalisateur}(F \bowtie S))$$

```
SELECT F.realisateur
FROM Film F
WHERE NOT EXISTS (SELECT S.cinema
                  FROM Séance S
                    EXCEPT
                  SELECT S1.cinema
                  FROM Séance S1, Film F1
                  WHERE S1.titre=F1.titre
                  AND F1.realisateur=F.realisateur)
```

On projette l'attribut `realisateur` sur la table obtenue :

$$\Pi_{realisateur}(\Pi_{realisateur}(Film) \times \Pi_{cinema}(Seance)) - (\Pi_{realisateur,cinema}(Film \bowtie Seance))$$

réalisateur	cinéma
Chytilova	Le Champo
Chytilova	Odéon
Polanski	Le Champo

Sémantique des requêtes avec conditions universelles

$\pi_{réalisateur}(F) - \pi_{réalisateur}((\pi_{cinéma}(S) \times \pi_{réalisateur}(F)) - \pi_{cinéma,réalisateur}(F \bowtie S))$

```
SELECT F.realisateur
FROM Film F
WHERE NOT EXISTS (SELECT S.cinema
                  FROM Séance S
                    EXCEPT
                  SELECT S1.cinema
                  FROM Séance S1, Film F1
                  WHERE S1.titre=F1.titre
                  AND F1.realisateur=F.realisateur)
```

On projette l'attribut `realisateur` sur la table obtenue :

$$\Pi_{realisateur}(\Pi_{realisateur}(Film) \times \Pi_{cinema}(Seance)) - (\Pi_{realisateur,cinema}(Film \bowtie Seance))$$

réalisateur
Chytilova
Polanski

Sémantique des requêtes avec conditions universelles

$$\pi_{réalisateur}(F) - \pi_{réalisateur}((\pi_{cinéma}(S) \times \pi_{réalisateur}(F)) - \pi_{cinéma,réalisateur}(F \bowtie S))$$

```
SELECT F.realisateur
FROM Film F
WHERE NOT EXISTS (SELECT S.cinema
                  FROM Séance S
                    EXCEPT
                  SELECT S1.cinema
                  FROM Séance S1, Film F1
                  WHERE S1.titre=F1.titre
                  AND F1.realisateur=F.realisateur)
```

On projette aussi l'attribut `realisateur` sur la table `Film` :

$$\Pi_{realisateur}(Film)$$

réalisateur	titre	acteur
Varda	Les Créatures	Deneuve
Varda	Les Créatures	Piccoli
Chytilova	Sedmikrasky	Cerhova
Bunuel	Belle de jour	Deneuve
Varda	Cléo de 5 à 7	Marchand
Polanski	Repulsion	Deneuve

Sémantique des requêtes avec conditions universelles

$$\pi_{réalisateur}(F) - \pi_{réalisateur}((\pi_{cinéma}(S) \times \pi_{réalisateur}(F)) - \pi_{cinéma,réalisateur}(F \bowtie S))$$

```
SELECT F.realisateur
FROM Film F
WHERE NOT EXISTS (SELECT S.cinema
                  FROM Séance S
                    EXCEPT
                  SELECT S1.cinema
                  FROM Séance S1, Film F1
                  WHERE S1.titre=F1.titre
                  AND F1.realisateur=F.realisateur)
```

On projette aussi l'attribut `realisateur` sur la table `Film` :

$$\Pi_{realisateur}(Film)$$

réalisateur
Varda
Chytilova
Bunuel
Polanski

Sémantique des requêtes avec conditions universelles

$$\pi_{réalisateur}(F) - \pi_{réalisateur}((\pi_{cinéma}(S) \times \pi_{réalisateur}(F)) - \pi_{cinéma,réalisateur}(F \bowtie S))$$

```
SELECT F.realisateur
FROM Film F
WHERE NOT EXISTS (SELECT S.cinema
                  FROM Séance S
                    EXCEPT
                  SELECT S1.cinema
                  FROM Séance S1, Film F1
                  WHERE S1.titre=F1.titre
                  AND F1.realisateur=F.realisateur)
```

On calcule finalement la différence ensembliste de ces deux tables :

$$\Pi_{realisateur}(Film)$$

réalisateur
Varda
Chytilova
Bunuel
Polanski

$$\Pi_{realisateur}(\Pi_{realisateur}(Film) \times \Pi_{cinema}(Seance)) - (\Pi_{realisateur,cinema}(Film \bowtie Seance))$$

différence ensembliste :

—

réalisateur
Chytilova
Polanski

Sémantique des requêtes avec conditions universelles

$$\pi_{réalisateur}(F) - \pi_{réalisateur}((\pi_{cinéma}(S) \times \pi_{réalisateur}(F)) - \pi_{cinéma,réalisateur}(F \bowtie S))$$

```
SELECT F.realisateur
FROM Film F
WHERE NOT EXISTS (SELECT S.cinema
                  FROM Séance S
                    EXCEPT
                  SELECT S1.cinema
                  FROM Séance S1, Film F1
                  WHERE S1.titre=F1.titre
                  AND F1.realisateur=F.realisateur)
```

On calcule finalement la différence ensembliste de ces deux tables :

$$\Pi_{realisateur}(Film)$$

réalisateur
Varda
~~Chytilova~~
Bunuel
~~Polanski~~

$$\Pi_{realisateur}(\Pi_{realisateur}(Film) \times \Pi_{cinema}(Seance)) - (\Pi_{realisateur,cinema}(Film \bowtie Seance))$$

différence ensembliste :

réalisateur
Chytilova
Polanski

Sémantique des requêtes avec conditions universelles

$$\pi_{réalisateur}(F) - \pi_{réalisateur}((\pi_{cinéma}(S) \times \pi_{réalisateur}(F)) - \pi_{cinéma,réalisateur}(F \bowtie S))$$

```
SELECT F.realisateur
FROM Film F
WHERE NOT EXISTS (SELECT S.cinema
                  FROM Séance S
                    EXCEPT
                  SELECT S1.cinema
                  FROM Séance S1, Film F1
                  WHERE S1.titre=F1.titre
                  AND F1.realisateur=F.realisateur)
```

On calcule finalement la différence ensembliste de ces deux tables :

$$\Pi_{realisateur}(Film)$$

réalisateur
Varda
~~Chytilova~~
Bunuel
~~Polanski~~

$$\Pi_{realisateur}(\Pi_{realisateur}(Film) \times \Pi_{cinema}(Seance)) - (\Pi_{realisateur,cinema}(Film \bowtie Seance))$$

différence ensembliste : —

réalisateur
Chytilova
Polanski

= tableau résultat :

réalisateur
Varda
Bunuel

Parenthèse

Attention :

$$\pi_{réalisateur}(F) - \pi_{réalisateur}((\pi_{cinéma}(S) \times \pi_{réalisateur}(F)) - \pi_{cinéma,réalisateur}(F \bowtie S))$$

(Les réalisateurs dont au moins un film passe dans chaque cinéma.)

correspond en toute rigueur à cette requête SQL :

```
SELECT DISTINCT F.realisateur
FROM Film F
WHERE NOT EXISTS (SELECT S.cinema
                  FROM Séance S
                    EXCEPT
                  SELECT S1.cinema
                  FROM Séance S1, Film F1
                  WHERE S1.titre=F1.titre
                  AND F1.realisateur=F.realisateur)
```

DISTINCT sert à éliminer les doublons (cf seconde partie du cours sur SQL).

L'opérateur de division

- La requête algébrique que nous venons d'analyser n'est pas très facile à lire.
- En fait, il existe en algèbre relationnelle un opérateur de division qui permet une réécriture plus compacte :

$$\pi_{cinema,realisateur}(Seance \bowtie Film) \div \pi_{cinema}(Seance)$$

- Le tableau résultat de cette requête est défini comme la plus grande relation R sur l'unique attribut `realisateur` telle que :

$$R \times \pi_{cinema}(Seance) \subseteq \pi_{cinema,realisateur}(Seance \bowtie Film)$$

L'opérateur de division

$$\pi_{cinema,realisateur}(Seance \bowtie Film) \div \pi_{cinema}(Seance)$$

- Le tableau résultat de cette requête est défini comme la plus grande relation R sur l'unique attribut `realisateur` telle que :

$$R \times \pi_{cinema}(Seance) \subseteq \pi_{cinema,realisateur}(Seance \bowtie Film)$$

Et en effet, on a bien :

tableau résultat :

réalisateur
Varda
Bunuel

$\times$

$\Pi_{cinema}(Seance)$

cinéma
Le Champo
Odéon

$\subseteq$

$\Pi_{realisateur,cinema}(Seance \bowtie Film)$

réalisateur	cinéma
Varda	Le Champo
Bunuel	Le Champo
Varda	Odéon
Polanski	Odéon
Bunuel	Odéon

L'opérateur de division

$$\pi_{cinema,realisateur}(Seance \bowtie Film) \div \pi_{cinema}(Seance)$$

- Le tableau résultat de cette requête est défini comme la plus grande relation R sur l'unique attribut `realisateur` telle que :

$$R \times \pi_{cinema}(Seance) \subseteq \pi_{cinema,realisateur}(Seance \bowtie Film)$$

Alors que :

$\Pi_{realisateur}$

réalisateur
Varda
Bunuel
Polanski

$\times$

$\Pi_{cinema}(Seance)$

cinéma
Le Champo
Odéon

$\not\subseteq$

$\Pi_{realisateur,cinema}(Seance \bowtie Film)$

réalisateur	cinéma
Varda	Le Champo
Bunuel	Le Champo
Varda	Odéon
Polanski	Odéon
Bunuel	Odéon

L'opérateur de division

$$\pi_{cinema,realisateur}(Seance \bowtie Film) \div \pi_{cinema}(Seance)$$

- Le tableau résultat de cette requête est défini comme la plus grande relation R sur l'unique attribut `realisateur` telle que :

$$R \times \pi_{cinema}(Seance) \subseteq \pi_{cinema,realisateur}(Seance \bowtie Film)$$

Et :

$\Pi_{realisateur}$

réalisateur
Varda
Bunuel
Chytilova

$\times$ $\Pi_{cinema}(Seance)$

cinéma
Le Champo
Odéon

$\not\subseteq$ $\Pi_{realisateur,cinema}(Seance \bowtie Film)$

réalisateur	cinéma
Varda	Le Champo
Bunuel	Le Champo
Varda	Odéon
Polanski	Odéon
Bunuel	Odéon

L'opérateur de division : un exemple plus simple

Les titres des films qui passent dans tous les cinéma :

$$Seance \div \pi_{cinema}(Seance)$$

- Le tableau résultat de cette requête est défini comme la plus grande relation R sur l'unique attribut `titre` telle que :

$$R \times \pi_{cinema}(Seance) \subseteq Seance$$

Et en effet on a bien :

tableau résultat :

titre
Belle de jour

$\times$ $\Pi_{cinema}(Seance)$

cinéma
Le Champo
Odéon

$\subseteq$ $Seance$

titre	cinéma
Les Créatures	Le Champo
Belle de jour	Le Champo
Cléo de 5 à 7	Odéon
Repulsion	Odéon
Belle de jour	Odéon

L'opérateur de division : un exemple plus simple

Les titres des films qui passent dans tous les cinéma :

$$Seance \div \pi_{cinema}(Seance)$$

- Le tableau résultat de cette requête est défini comme la plus grande relation R sur l'unique attribut `titre` telle que :

$$R \times \pi_{cinema}(Seance) \subseteq Seance$$

Alors que :

$\Pi_{cinema}(Seance)$

titre
Belle de jour
Répulsion

$\times$

cinéma
Le Champo
Odéon

$\not\subseteq$

$Seance$

titre	cinéma
Les Créatures	Le Champo
Belle de jour	Le Champo
Cléo de 5 à 7	Odéon
Repulsion	Odéon
Belle de jour	Odéon

L'opérateur de division : un exemple plus simple

Les titres des films qui passent dans tous les cinéma :

$$Seance \div \pi_{cinema}(Seance)$$

- Le tableau résultat de cette requête est défini comme la plus grande relation R sur l'unique attribut `titre` telle que :

$$R \times \pi_{cinema}(Seance) \subseteq Seance$$

Et :

$\Pi_{cinema}(Seance)$

titre
Belle de jour
Les Créatures

$\times$

cinéma
Le Champo
Odéon

$\not\subseteq$

$Seance$

titre	cinéma
Les Créatures	Le Champo
Belle de jour	Le Champo
Cléo de 5 à 7	Odéon
Repulsion	Odéon
Belle de jour	Odéon

L'opérateur de division : un exemple plus simple

Les titres des films qui passent dans tous les cinéma :

$$Seance \div \pi_{cinema}(Seance)$$

- Le tableau résultat de cette requête est défini comme la plus grande relation R sur l'unique attribut `titrer` telle que :

$$R \times \pi_{cinema}(Seance) \subseteq Seance$$

Et :

$\Pi_{cinema}(Seance)$

titre
Belle de jour
Sedmikrasky

$\times$

cinéma
Le Champo
Odéon

$\not\subseteq$

$Seance$

titre	cinéma
Les Créatures	Le Champo
Belle de jour	Le Champo
Cléo de 5 à 7	Odéon
Repulsion	Odéon
Belle de jour	Odéon

Définition formelle de la division $R \div S$

Si $R(A_1, \ldots, A_k, A_{k+1}, \ldots, A_{k+n})$ et $S(A_1, \ldots, A_k)$ sont deux relations telles que $schéma(R) \subseteq schéma(S)$, alors $T(A_{k+1}, \ldots, A_{k+n}) = R \div S$ ssi :

- $T(A_{k+1}, \ldots, A_{k+n})$ est la plus grande relation telle que $T \times S \subseteq R$

L'opérateur de division est définissable dans le fragment $\pi, \sigma, \times$ de l'algèbre relationnelle :

$$R \div S = \pi_{A_{k+1},\ldots,A_{k+n}}(R) - \pi_{A_{k+1},\ldots,A_{k+n}}((\pi_{A_{k+1},\ldots,A_{k+n}}(R) \times S) - R)$$

Conclusion provisoire

- L'écriture des requêtes du type de celles que nous venons de voir devient vite franchement compliqué...
- Mais on verra bientôt des moyens plus simples d'écrire certaines de ces requêtes.
- Grâce à l'agrégation.