4 Le langage commercial de requête SQL

- Une relation n'est pas implémentée comme un ensemble, mais comme un *multi-ensemble* (m.e.).
- Multi-ensemble : les répétitions sont permises, mais l'ordre ne compte pas.
  Par ex., dans le cas des m.e. :
  $\{1, 4, 2, 5\} = \{1, 2, 4, 5\} \neq \{1, 2, 4, 5, 5\} \neq \{1, 2, 4, 5, 5, 5\}$
  $\{1, 4, 2, 5\} \cup \{1, 7\} = \{1, 1, 4, 2, 5, 7\}$
- Pourquoi ce choix de structure de données ? Rapidité des calculs ! Penser au calcul de l'union ensembliste...

4.1 Format d'une Requête Simple

A1,...,AN : attributs, R: nom de relation

SELECT A1,...,AN
FROM R
WHERE C;

Ici, C est une condition de sélection sur les n-uplets de R, comme dans le σ_C de l'algèbre.

Le SELECT fait une projection (op. algèbre) du résultat de la sélection sur les attributs A1,...,AN. Le σ_C est simulé par le WHERE !

Exemple de schéma de base pour illustrer SQL

$\mathcal{S} = \{Films(Titre, Date, Longueur, Couleur, NomStudio, IdProducteur),$
$Joue(NomFilm, FilmDate, NomActeur, Paye),$
$Acteur(Nom, Adresse, Sex, Date\_Naissance),$
$Studio(Nom, Adresse),$
$Producteurs(Nom, Adresse, Id)\}$

SELECT Titre, Longueur
FROM Films
WHERE NomStudio='Disney' AND Date=1990;

Réponse : une table

Titre	Longueur
Pretty Woman	119
⋮	⋮

SELECT *
FROM Films
WHERE NomStudio='Disney' AND Date=1990;

* est une syntaxe pour indiquer tous les attributs de R

Réponse : une table

Titre	Date	Longueur	Couleur	NomStudio	IdProducteur
Pretty Woman	1990	119	Vrai	Disney	1
⋮	⋮	⋮	⋮	⋮	⋮

Conditions de sélection plus complexes

SELECT Titre
FROM Films
WHERE (Date $>$ 1970 OR Longueur $<$ 90) AND NOT Couleur;

NB :
1) Partie WHERE C d'une requête SQL = opérateur σ_C de l'algèbre relationnelle !
2) Ici, Couleur est de type bool;

Valeurs Nulles

- A différence de l'algèbre, une colonne peut avoir la valeur NULL.
- "NULL *op valeur*" s'évalue NULL si *op* est un opération arithmétique ($+, \times$ etc.).
- "NULL *rel valeur*" s'évalue UNKNOWN, si *rel* est $=$ ou $>$ ou $<$".

SELECT *
FROM Films
WHERE Date>= 1970 OR Date < 1970;

renvoie la table Films privée des n-uplets où la valeur de Date est NULL.
Pouquoi ?

4.2 Requêtes à Plusieurs Arguments

Soient R1,...,Rk des relations.

Format :

SELECT A1,...,AN
FROM R1,...,Rk
WHERE C;

Ici, la relation dans laquelle on va chercher les n-uplets est le produit cartésien des relations nommées R1,...,Rk.

Exemple. “Quel est le nom du producteur de “Star Wars” ?

SELECT Nom
FROM Films, Producteurs
WHERE Titre =’Star Wars’ AND IdProducteur=Id;

NB. : Ici, on a utilisé un AND, pour simuler $\bowtie$, comme en Calcul relationnel !

En algèbre, le produit cartésien requiert que les schémas des arguments soient disjoints. Et si des attributs sont communs ?

Exemple(Acteur et Producteur partagent les attributs Nom et Adresse)

"Quels acteurs et quels producteurs ont la même adresse ?

SELECT Acteur.Nom, Producteurs.nom
FROM Acteur, Producteurs
WHERE Acteur.Adresse=Producteurs.Adresse;

Réponse :

Acteur.Nom	Producteur.Nom
Jane Fonda	Ted Turner
⋮	⋮

Renommage des attributs partagés, implementation de l'operateur ρ de l'algébre.

Ordonnancement des n-uplets ($\notin$ AR !)

Format :
Si { B1,...,Bk} $\subseteq$ A1,...,An

SELECT A1,...,An
FROM R
WHERE C
ORDER BY B1,...,Bk;

Exemple

SELECT *
FROM Films
WHERE NomStudio='Disney' AND Date=1999
ORDER BY Longueur, Titre;

Dans la réponse, si n(Longueur)$<$ n'(Longeur) alors n avant n' et si n(Longueur)=n'(Longueur), n avant n' ssi n(Titre) $<$ n'(Titre).

Quelques autres Opérations de l'Algèbre et les Sous-Requêtes Imbriquées

Exemple

(SELECT Nom, Adresse FROM Acteur)
MINUS
(SELECT Nom, Adresse FROM Producteurs);

En algèbre relationnelle :

$$\pi_{Nom,Adresse}(Acteur) \setminus \pi_{Nom,Adresse}(Producteurs)$$

Opérateur IN

“Quels sont les producteurs de films dans lesquels Harrison Ford joue ?”

```
SELECT Nom
FROM Producteurs
WHERE Id IN
        (SELECT IdProducteur
         FROM Films
         WHERE Titre IN
                (SELECT NomFilm
                 FROM JOUE
                 WHERE NomActeur = 'Harrison Ford'
                 )
        );
```

4.3 Utilisation des variables n-uplets dans SQL

Utilité : raisonner sur plusieurs n-uplets d'un même relation, les comparer etc.

Exemple

"Quel acteurs ont la même adresse ?"

SELECT Star1.Nom, Star2.Nom
FROM Acteur Star1, Acteur Star2
WHERE Star1.Adresse = Star2.Adresse AND Star1.Nom < Star2.Nom;

N.B : ici, Star1 et Star2 sont des variables pour des n-uplets. Chacune d'elles varie sur les n-uplets de la table Acteur. Ecriture équivalente :

SELECT t1.Nom, t2.Nom
FROM Acteur t1, Acteur t2
WHERE t1.Adresse = t2.Adresse AND t1.Nom < t2.Nom;

Suite de l'Exemple. Format de la réponse :

Acteur1.Nom	Acteur2.Nom
Balwin	Basinger
Cruise	Fonda
⋮	⋮

Même requête SQL, mais **seulement** :

WHERE t1.Adresse = t2.Adresse

Risque de :

Acteur1.Nom	Acteur2.Nom
Balwin	Balwin
Balwin	Basinger
Basinger	Balwin
⋮	⋮

Pourquoi ?

Suite de l'Exemple

Si :

WHERE t1.Adresse = t2.Adresse AND t1.Nom <> t2.Nom

encore risque de redondances :

Acteur1.Nom	Acteur2.Nom
Balwin	Basinger
Basinger	Balwin
⋮	⋮

Quantification Sur Les Variables n-uplets, Opérateur EXISTS

Exemple

"Quels acteurs n'ont joué dans aucun films paru après 2000?"'

```
SELECT t1.NomActeur
FROM Joue t1
WHERE NOT EXISTS
        (SELECT t2.FilmDate
         FROM Joue t2
         WHERE t2.FilmDate > 2000 AND t1.NomActeur = t2.NomActeur)
```

Comment on écrirait en CR ?

Quantification Sur Les Variables n-uplets, Opérateur ANY

Exemple

"Quels titres ont été utilisés pour plusieurs films" ? (remakes)

```
SELECT t1.Titre
FROM Films t1
WHERE t1.Date < ANY
                (SELECT Date
                  FROM FILMS
                  WHERE Titre =t1.Titre);
```

"*any* date" en anglais (dans ce contexte) : au moins une date, peu importe la quelle.

Comment on écrirait en C.R. ?

4.4 Opérations de SQL qui ne sont pas dans l'algèbre

- Elimination des répétitions dans un multi-ensemble
- Aggregation
- Formation de groupes

DISTINCT : après le SELECT, produit une seule copie chaque n-uplet du résultat.

Exemple

"Quel est l'ensemble des producteurs de films dans lesquels Harrison Ford joue" ?

```
SELECT DISTINCT Nom
FROM Producteur, Films, Joue
WHERE ProducteurId=Id AND
                Titre = NomFilm AND
                Date = FilmDate AND
                NomActeur = 'Harrison Ford';
```

Les opérateur d'Aggregation SUM et AVG

Après le SELECT, appliqués à un attribut approprié : SUM somme les valeurs, AVG calcule la moyenne.

Exemples

Schema de Contrats :$\{NomActeur, NomFilm, Paye\}$.

SELECT SUM(Paye)
FROM Contrats
WHERE NomActeur = 'Harrison Ford';

SELECT AVG(Paye)
FROM Contrats
WHERE NomActeur = 'Harrison Ford';

Comportement semblable : MIN et MAX.

L'opérateur d'Aggregation COUNT

Après le SELECT, appliqués à un attribut, compte le nombre des valeurs.

Exemples

```
SELECT COUNT(NomActeur)
FROM Joue;
```

```
SELECT COUNT(DISTINCT NomActeur)
FROM Joue;
```

Cas particulier où on compte toutes les n-uplet de la table :

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

Regroupements : l'opérateur GROUP BY

Après la partie WHERE C, appliqué à un attribut A, partitionne les n-uplets selon la valeur de A.

Exemple.

On veut, film par film, la liste des acteurs qui jouent dans ce film.

SELECT Titre, NomActeur
FROM Films, Joue
WHERE Titre = NomFilm
GROUP BY Titre ;;

Exemple, suite

Format de la réponse :

Titre	NomActeur
Pretty Woman	Richard Gere
	a2
	a3
Eyes Wide Shut	Nicole Kidman
	TomCruise
	⋮