Algèbre relationnelle

Skander Zannad et Judicaël Courant

2014-04-12

1 Résumé des épisodes précédents

1. MCD / MLD ;
2. Implantation en SQL (MPD) et requêtes ;
3. Calcul relationnel pour modéliser les requêtes.

2 Problème

Expressivité de SQL ?

Calcul relationnel modélise les requêtes qu'on veut faire ;

Algèbre relationnelle modélise les requêtes que SQL peut faire.

3 Algèbre relationnelle

Algèbre : on étudie des opérations sur les données d'une base (similaire aux LCI).
Neuf opérations :

1. sélection) ;
2. projection ;
3. renommage ;
4. produit cartésien ;
5. division cartésienne ;
6. jointure (naturelle) ;
7. union ;
8. intersection ;
9. différence.

3.1 Sélection

Quelles sont les personnes dont le prénom est «Clint» ?

Pour répondre à la question, on *sélectionne*, dans la table `PERSONNE`, les nuplets dont le champ `nom` est «Clint».

Définition 3.1.1 (Sélection)**.**
Pour $A, B \in \mathbf{att}$ et $a \in \mathbf{dom}$, on appelle opérations de sélection $\sigma_{A=a}$ (resp. $\sigma_{A=B}$), les fonctions définies par

$$\sigma_{A=a}(I) = \{\, t \in I \mid t(A) = a \,\}$$
$$(\text{resp. } \sigma_{A=B}(I) = \{\, t \in I \mid t(A) = t(B) \,\})$$

pour toute relation I ayant au moins A (resp. et B) comme attribut(s).

3.2 Projection

Quels sont les numéros des personnes qui sont des réalisateurs ?

Pour répondre à la question, il suffit de prendre la colonne `idrealisateur` de la table `FILM`. On dit qu'on *projette* la table film sur l'attribut `idrealisateur`.

Définition 3.2.1 (Projection)**.**
Soit $n \in \mathbb{N}^*$ et $A_1, \ldots, A_n \in \mathbf{att}$. On appelle opération de *projection sur les attributs* $A_1, \ldots, A_n$ et on note $\pi_{A_1,\ldots,A_n}$ l'opération définie par

$$\pi_{A_1,\ldots,A_n}(I) = \{\, t[A_1 \ \ldots \ A_n] \mid t \in I \,\}$$

pour toute relation I ayant au moins les attributs A_1, …, A_n.

3.3 Renommage

Définition 3.3.1 (Renommage)**.**
Soit U un ensemble fini d'attributs.

On appelle *renommage d'attributs* toute $f : U \to \mathbf{att}$ injective.

On appelle alors *renommage associé à f d'un nuplet u sur U* le nuplet $\rho_f(u)$ sur $f(U)$ défini par

$$\forall a \in \mathbf{att} \quad \rho_f(u)(f(a)) = u(a)$$

Autrement dit, si $U = \{\, \alpha_1, \ldots, \alpha_p \,\}$, pour tous $v_1, \ldots, v_p \in \mathbf{dom}$, on a

$$\rho_f(\langle \alpha_1 : v_1, \ldots, \alpha_p : v_p \rangle) = \langle f(\alpha_1) : v_1, \ldots, f(\alpha_p) : v_p \rangle$$

On appelle *opération de renommage ρ_f associée à f* l'opération définie par

$$\rho_f(I) = \{\, \rho_f(u) \mid u \in I \,\}$$

pour toute relation I sur U.

Souvent :

— U est clair d'après le contexte ;

— et f laisse invariant tous les éléments de U sauf n éléments $A_1, \ldots, A_n$ dont les images respectives sont $B_1, \ldots, B_n$.

l'opération de renommage ρ_f est alors notée $\rho_{A_1 \to B_1, \ldots, A_n \to B_n}$

3.4 Produit cartésien

En mathématiques, $A \times B$ désigne l'ensemble des couples (x, y) pour $x \in A$ et $y \in B$. Ici, ce sera l'ensemble des $x \oplus y$ où $x \oplus y$ désigne la concaténation des deux nuplets x et y, supposés n'avoir aucun attribut commun :

Définition 3.4.1 (Produit cartésien)**.**
Soit I et J deux relations de *sorts* U et V avec $U \cap V = \emptyset$. On note $I \times J$ la relation de sort $U \cup V$ définie par

$$\begin{aligned} I \times J &= \{\, u \oplus v \mid u \in I \text{ et } v \in J \,\} \\ &= \{\, u : U \cup V \to \mathbf{att} \mid u[U] \in I \text{ et } u[V] \in J \,\} \end{aligned}$$

3.5 Division cartésienne

Notons :

— $J[\texttt{nomtitre}]$ l'ensemble des $\langle \texttt{nom} : n, \texttt{titre} : t \rangle$ pour n nom d'une personne et t film tel que n a joué dans t et Clint Eastwood a réalisé t ;

— $P[\texttt{nom}]$ l'ensemble des $\langle \texttt{nom} : n \rangle$ pour n nom d'une personne ($P = \pi_{\texttt{nom}}(\texttt{PERSONNE})$) ;

— $E[\texttt{titre}]$ l'ensemble des $\langle \texttt{titre} : t \rangle$ pour t film de Clint Eastwood.

— $K[\texttt{nom}]$ l'ensemble $\langle \texttt{nom} : n \rangle$ pour n ayant joué dans **tous** les films de Clint Eastwood.

K est la plus grande relation sur $\{\, \texttt{nom} \,\}$ vérifiant $K \times E \subset J$.
Cette relation est appelée *division cartésienne* de J par E.

Définition 3.5.1 (Division cartésienne)**.**
Soit I et J deux relations, respectivement sur U et V avec $U \subset V$. Alors la *division cartésienne* de J par I est la relation sur $V \setminus U$ notée $J \div I$ et définie par :

$$\begin{aligned} J \div I &= \{\, t : V \setminus U \to \mathbf{att} \mid \forall u \in I \quad u \oplus t \in J \,\} \\ &= \{\, t : V \setminus U \to \mathbf{att} \mid \forall u \in I \exists v \in J \quad v[U] = u \text{ et } v[V \setminus U] = t \,\} \end{aligned}$$

3.6 Jointure simple

Quels sont les titres des films réalisés par des personnes dont le prénom est «Clint» ?
Pour répondre :

1. on calcule $I = \sigma_{\texttt{prenom}="Clint"}(\texttt{PERSONNE})$;
2. on calcule $J = \pi_{\texttt{titre,idrealisateur}}(\texttt{FILM})$;
3. on calcule le produit $I \times J$;
4. on calcule la sélection $S = \sigma_{\texttt{id=idrealisateur}}(I \times J)$;
5. le résultat est $\pi_{\texttt{titre}}(S)$.

Les étapes 3 et 4 constituent un calcul de *jointure*.

Définition 3.6.1 (Jointure)**.**
Soit I et J deux relations de *sorts* U et V avec $U \cap V = \emptyset$, $A \in U$ et $B \in V$. Alors la *jointure symétrique de I et J selon* (A, B) est la relation $I[A = B]J$ de *sort* $U \cup V$ définie par

$$I[A = B]J = \sigma_{A=B}(I \times J)$$

La jointure :

— N'apporte **aucune expressivité** par rapport au produit suivi d'une sélection ;

— En général, **se calcule plus facilement** (si on s'y prend bien).

Exemple : prenez un annuaire téléphonique de Lyon et la liste des enseignants de MPSI, calculez la jointure sur le couple nom de l'enseignant/nom de l'abonné :

— par produit puis sélection ;

— directement.

3.7 Union

Même définition qu'en mathématiques.
Quels sont les personnes dont le prénom est «Clint» ou «Martin» ?

$$\sigma_{\texttt{prenom}="Clint"}(\texttt{PERSONNE}) \cup \sigma_{\texttt{prenom}="Martin"}(\texttt{PERSONNE})$$

3.8 Intersection

Définition mathématique habituelle.
Quelles sont les personnes dont le prénom est «Clint» et le nom «Eastwood» ?

$$\sigma_{\texttt{prenom}="Clint"}(\texttt{PERSONNE}) \cap \sigma_{\texttt{nom}="Eastwood"}(\texttt{PERSONNE})$$

NB : il est souvent plus simple de combiner deux opérations de sélection. Par exemple ici :

$$\sigma_{\texttt{prenom}="Clint"}(\sigma_{\texttt{nom}="Eastwood"}(\texttt{PERSONNE}))$$

3.9 Différence

Définition mathématique habituelle.
Les personnes qui ont réalisé un film :

$$I = \pi_{\texttt{id,nom,prenom,datenaissance}}(\texttt{PERSONNE}[\texttt{id} = \texttt{idrealisateur}]\texttt{FILM})$$

Quelles sont les personnes qui n'ont réalisé aucun film ?

$$\texttt{PERSONNE} \setminus I$$

4 Théorème de Codd

Les deux questions qui nous intéressent :

1. Étant donné une requête exprimée en calcul relationnel, l'ensemble répondant à cet requête peut-il être exprimé comme une expression de l'algèbre relationnelle ?
2. Étant donné une expression de l'algèbre relationnelle, l'ensemble qu'elle représente peut-il être exprimé comme l'ensemble répondant à une requête en calcul relationnel ?

Théorème 4.0.1 (de Codd)**.**
Une requête est exprimable en calcul relationnel si et seulement si elle peut être exprimée par une expression de l'algèbre relationnelle.

Mieux : la démonstration de ce théorème repose sur un algorithme de traduction qui donne un moyen effectif de passer de l'un à l'autre.

Idée de l'algorithme de traduction calcul vers algèbre :
ramener la traduction d'une formule à la traduction de sous-formules.

Exemple : pour une requête $\{ t_1, \ldots, t_n \mid \phi \wedge \psi \}$, on traduit les requêtes $\{ t_1, \ldots, t_n \mid \phi \}$ et $\{ t_1, \ldots, t_n \mid \psi \}$ en des expressions calculant des relations I_1 et I_2, puis on calcule $I_1 \cap I_2$.

Opérations associées au différentes formules :

1. A atomique : c'est déjà une relation
2. $\phi \wedge \psi$: $\cap$
3. $\phi \vee \psi$: $\cup$
4. $\phi \Rightarrow \psi$: traduire $\neg\phi \vee \psi$
5. $\neg\phi$: différence
6. $\exists x\ \phi$: projection (on oublie le champ x)
7. $\forall x\ \phi$: division (par une table contenant tous les x) ou traduire $\neg\exists x \neg\phi$.

5 SQL et l'algèbre relationnelle

La question initiale :

Quelle est l'expressivité de SQL ?

Nous n'y avons pas vraiment répondu : SQL n'est ni le calcul relationnel, ni l'algèbre relationnelle.

Mais :

— la traduction SQL vers calcul relationnel très facile ;

— la traduction algèbre relationnelle vers SQL facile ;

— la traduction SQL vers algèbre relationnelle facile.

5.1 Traduction de l'algèbre relationnelle vers SQL

Exercices : sur des exemples, comment traduire :

5.1.1 Sélection

$\sigma_{\texttt{id}=3}(\texttt{PERSONNE})$

5.1.2 Renommage

$\sigma_{\texttt{id}\to\texttt{idrealisateur}}(\texttt{PERSONNE})$

5.1.3 Projection

$\pi_{\texttt{nom}}(\texttt{PERSONNE})$

5.1.4 Produit cartésien

$\texttt{PERSONNE} \times \texttt{FILM}$

5.1.5 Jointure

PERSONNE[id = idrealisateur]FILM

Jointure implicite :

```
SELECT * FROM PERSONNE, FILM
WHERE PERSONNE.id = idrealisateur;
```

Jointure explicite :

```
SELECT * FROM PERSONNE
INNER JOIN FILM ON id = idrealisateur;

SELECT * FROM PERSONNE
JOIN FILM ON id = idrealisateur;
```

5.1.6 Union, intersection, différence

$\pi_{\texttt{nom}}(\texttt{PERSONNE}) \cup \pi_{\texttt{nom}}(\texttt{PERSONNAGE})$

```
SELECT nom FROM PERSONNE
UNION
SELECT nom FROM PERSONNAGE;
```

Intersection : INTERSECT
Différence : EXCEPT
Requête sans intérêt mais à la syntaxe incorrecte :

```
PERSONNE UNION PERSONNE;
```

On doit toujours avoir un «SELECT» :

```
SELECT * FROM PERSONNE
UNION
SELECT * FROM PERSONNE;
```

5.1.7 Division cartésienne

On veut l'ensemble des $\langle \texttt{nom} : n \rangle$ pour n ayant joué dans **tous** les films de Clint Eastwood.

Malheureusement : pas d'opérateur de division dans SQL !

Possibilité : calculer d'abord l'ensemble des $\langle \texttt{nom} : n \rangle$ pour n n'ayant pas joué dans **tous** les films de Clint Eastwood (réalisateur no 3), c'est-à-dire pour n tel qu'il existe au moins un film de Clint Eastwood dans lequel n n'a pas joué, puis calculer le complémentaire par différence.

C'est (horriblement) compliqué :

```
SELECT nom FROM PERSONNE
EXCEPT
SELECT DISTINCT nom FROM (
  SELECT nom, titre FROM
   (SELECT nom FROM PERSONNE),
   (SELECT titre FROM FILM WHERE idrealisateur=3)
  EXCEPT
  SELECT nom, titre FROM PERSONNE, FILM, JOUE
  WHERE idfilm=FILM.id AND idacteur = PERSONNE.id
  AND idrealisateur=3
);
```

et encore, notre requête :

1. utilise le `id` de Clint Eastwood ;
2. est boguée en cas d'homonymes (on aurait dû faire des recherches sur les `id` et non les `nom`).

Deux façons de faire plus simple :

1. Faire autrement ;
2. Décomposer la requête grâce à des vues.

Faire autrement :

```
SELECT nom FROM PERSONNE, FILM, JOUE
WHERE idfilm=FILM.id AND idacteur = PERSONNE.id
AND idrealisateur = 3
GROUP BY PERSONNE.id
HAVING COUNT(*) =
  (SELECT COUNT(*) FROM FILM WHERE idrealisateur=3)
```

— Résout le bogue précédent ;

— En introduit un autre si le diviseur est vide (i.e. si le réalisateur 3 n'a réalisé aucun film).

Décomposer la requête grâce à des vues.

```
-- table des id des films de Clint Eastwood
CREATE VIEW FILMS_EASTWOOD AS
SELECT FILM.id as idfilm
FROM FILM, PERSONNE
WHERE idrealisateur = PERSONNE.id
AND nom = 'Eastwood' AND prenom = 'Clint';
```

(crée une table virtuelle qui contiendra le résultat de cette requête ; si le contenu des tables `FILM` et `PERSONNE` est modifié, la vue est modifiée)

```
-- table des couples acteurs/films
-- pour les films de Clint Eastwood
CREATE VIEW ACTEURS_EASTWOOD AS
SELECT id, nom, JOUE.idfilm
FROM FILMS_EASTWOOD, JOUE, PERSONNE
WHERE FILMS_EASTWOOD.idfilm = JOUE.idfilm
AND idacteur=id;

-- produit cartésien PERSONNE * FILMS_EASTWOOD
CREATE VIEW PROD_PERSONNE_FILMS_EASTWOOD AS
SELECT id, nom, idfilm
FROM PERSONNE, FILMS_EASTWOOD;
```

Et finalement, la requête s'écrit

```
SELECT id, nom FROM personne
EXCEPT
SELECT id, nom FROM (
  SELECT * FROM PROD_PERSONNE_FILMS_EASTWOOD
  EXCEPT
  SELECT * FROM ACTEURS_EASTWOOD
);
```

(Ouf !)

6 Mis sous le tapis

En fait, SQL a quelques autres différences avec l'algèbre relationnelle :

— requêtes agrégats en SQL ;

— résultats en SQL : listes et non ensembles.

7 Agrégats

On peut ajouter un opérateur d'agrégation à l'algèbre relationnelle.

Définition 7.0.1 (fonction d'agrégation)**.**
Soit f une fonction prenant en argument une liste $\mathcal{L}$ d'éléments de $\mathbf{dom}$. On dit que f est une *fonction d'agrégation* si la valeur de $f(\mathcal{L})$ ne dépend pas de l'ordre des éléments de $\mathcal{L}$.

En pratique, on prendra pour fonctions d'agrégation :

— la fonction de comptage (de la longueur de la liste) ;

— $\max$;

— $\min$;

— la fonction moyenne arithmétique des éléments de la liste ;

— la fonction somme des éléments de la liste.

Définition 7.0.2 (opération d'agrégation)**.**
Soit A_1, ..., A_n et $B_1, \ldots, B_p$ des attributs, R une relation dont le **sort** contient au moins tous ces attributs et f_1, ..., f_p des fonctions d'agrégation. Alors on note ${}_{A_1,\ldots,A_n}\gamma_{f(B_1),\ldots,f(B_p)}(R)$ la relation obtenue :

— en regroupant les valeurs de R identiques sur les attributs A_1, ..., A_n ;

— et en définissant de nouveaux attributs, notés $f_i(B_i)$, pour ces valeurs regroupées, pour tout $i \in [\![1, m[\![$, par application de la fonction d'agrégation f_i sur chacun de ces agrégats sur l'attribut B_i.

Cette fonction permet de modéliser les calculs d'agrégats de SQL.

Exemple : Nombre de films joués par les différents acteurs :

$$_{\texttt{idacteur}}\gamma_{f(\texttt{idfilm})}(\pi_{\texttt{idacteur},\texttt{idfilm}}(\texttt{JOUE}))[\texttt{idacteur} = \texttt{id}]\texttt{PERSONNE}$$

où f est la fonction de comptage.

En SQL :

```
SELECT id, nbfilms, nom, prenom, datenaissance
FROM
  (SELECT idacteur, count(idfilm) as nbfilms
   FROM (SELECT DISTINCT idacteur, idfilm FROM JOUE)
   GROUP BY idacteur),
  PERSONNE
WHERE id=idacteur;
```

8 Conclusion

On a vu :

— calcul relationnel ;

— algèbre relationnelle ;

— (une partie de) SQL ;

— le lien entre les trois.

Reste à voir :

— comment on utilise SQL depuis un langage de programmation (par exemple Python) ;
— quelle architecture ont les systèmes utilisant des bases de données SQL ;
— quelles questions ça soulève.