Bases de données SQL

La division relationnelle

Guillaume Raschia – Polytech Nantes; Nantes Université
originaux de Lester I. McCann, Univ. Wisconsin

Dernière mise-à-jour : 24 novembre 2025

Plan de la session

Définition

Contexte

Les systèmes de gestion de bases de données relationnelles (SGBD-R) sont fondés sur le modèle de données de E.F. Codd

- enraciné dans la théorie des ensembles

Les langages proposés par Codd sont :

- le calcul relationnel (déclaratif)
  - fondé sur la logique des prédicats (dite du premier ordre)
- l'algèbre relationnelle (langage fonctionnel ou procédural)
  - Six opérateurs fondamentaux : $\sigma, \pi, \times, \rho, -, \cup$
  - Trois opérateurs additionnels : $\cap, \bowtie, \div$
  - Par exemple : $R \cap S = (R \cup S) - (R - S) - (S - R)$

La divion, ÷

- Opérateur le moins facile à appréhender parmi les 9
  - défini à l'aide de 3 opérateurs (π, $-$ et $\times$) et 6 opérations !
  - fondé sur la recherche de valeurs qui ne sont pas des réponses attendues
  - pas facilement transposable en SQL
  - ardu à faire passer auprès des étudiants
- Souvent laissé de côté ou évoqué brièvement…

Et pourtant indispensable pour programmer un certain type de requêtes !

Que fait la division ?

Elle identifie les valeurs d'attributs d'une relation, le dividende, qui forment des nuplets avec toutes les valeurs d'une autre relation, le diviseur.

D'un autre point de vue :

- La division relationnelle (÷) est au produit cartésien ($\times$) ce que la division en arithmétique est à la multiplication.

Du produit cartésien…

Soit les relations unaires R et S et leur produit cartésien $R \times S = T$

$R =$

A
4
8

$S =$

B
3
1
7

$T =$

A	B
4	3
4	1
4	7
8	3
8	1
8	7

…à la division

Opération duale – inverse – du produit cartésien.

$T =$

A	B
4	3
4	1
4	7
8	3
8	1
8	7

$T \div S = R =$

A
4
8

$T \div R = S =$

B
3
1
7

Facile, pourquoi aller plus loin ?!

Avec un exemple plus concret

Soit un extrait de la base de données Commande-Produit-Fournisseur de C. Date

Produit =

pid	pnom	couleur	poids
p_1	stylo	rouge	12.0
...	...	...	...
p_6	équerre	bleue	19.0

CPF =

cid	**pid**	**fid**	quantité
c_1	p_1	f_1	123
...	...	...	...
c_4	p_6	f_5	456

CPF est la table d'association entre les Commandes, les Produits et les Fournisseurs

Suite de l'exemple

La requête

Donner les identifiants des fournisseurs qui ont livrés tous les produits de 17kg.

Rappel

Le schéma de la base de données est

> Produit (**pid**, pnom, couleur, poids)
> CPF (**cid**, **pid**, **fid**, quantité)

Un premier pas vers la réponse consiste à définir les schémas :

- du dividende α = fid, pid
- du diviseur β = pid

Fin de l'exemple

La construction de α et de β ne présente aucune difficulté :

$$\alpha = \pi_{\text{fid,pid}}(\text{CPF}) \qquad \beta = \pi_{\text{pid}}(\sigma_{\text{poids=17}}(\text{P}))$$

$\alpha =$

fid	pid
f_1	p_1
f_2	p_2
f_2	p_3
f_3	p_1
f_3	p_3
f_4	p_1
f_4	p_2
f_4	p_3

$\beta =$

pid
p_2
p_3

$\alpha \div \beta =$

fid
f_2
f_4

Expression algébrique

Construction de l'expression algébrique

Principe

Collecter les nuplets qui n'appartiennent pas au résultat, puis les retirer d'une liste dite de référence.

Dans l'exemple P-CPF, les identifiants de fournisseurs (fid) dans α constituent la liste de toutes les réponses éventuelles :

$$\pi_{\text{fid}}(\alpha) = \begin{array}{|c|}\hline \text{fid} \\ \hline f_1 \\ f_2 \\ f_3 \\ f_4 \\ \hline \end{array}$$

Saturation des nuplets (fid, pid)

Toutes les paires (fid,pid) possibles sont produites à l'aide de l'expression algébrique :

$$\pi_{\text{fid}}(\alpha) = \begin{array}{|c|}\hline \text{fid} \\ \hline f_1 \\ f_2 \\ f_3 \\ f_4 \\ \hline \end{array} \quad \times \quad \beta = \begin{array}{|c|}\hline \text{pid} \\ \hline p_2 \\ p_3 \\ \hline \end{array} \quad = \quad \gamma = \begin{array}{|cc|}\hline \text{fid} & \text{pid} \\ \hline f_1 & p_2 \\ f_1 & p_3 \\ f_2 & p_2 \\ f_2 & p_3 \\ f_3 & p_2 \\ f_3 & p_3 \\ f_4 & p_2 \\ f_4 & p_3 \\ \hline \end{array}$$

Suppression intermédiaire des « bonnes réponses »

Si l'on détruit dans γ – la table saturée – tous les nuplets de α – la table contenant les faits réels –, ne subsistent que les identifiants des fournisseurs qui ne sont pas solution !

$$\gamma = \begin{array}{|cc|}\hline \text{fid} & \text{pid} \\ \hline f_1 & p_2 \\ f_1 & p_3 \\ f_2 & p_2 \\ f_2 & p_3 \\ f_3 & p_2 \\ f_3 & p_3 \\ f_4 & p_2 \\ f_4 & p_3 \\ \hline \end{array} \quad - \quad \alpha = \begin{array}{|cc|}\hline \text{fid} & \text{pid} \\ \hline f_1 & p_1 \\ f_2 & p_2 \\ f_2 & p_3 \\ f_3 & p_1 \\ f_3 & p_3 \\ f_4 & p_1 \\ f_4 & p_2 \\ f_4 & p_3 \\ \hline \end{array} \quad = \quad \delta = \begin{array}{|cc|}\hline \text{fid} & \text{pid} \\ \hline f_1 & p_2 \\ f_1 & p_3 \\ f_3 & p_2 \\ \hline \end{array}$$

Notons que f_2 et f_4, les réponses attendues, ne figurent pas dans δ.

Négation des résultats

Ne reste plus qu'à détruire les nuplets obtenus, à partir de la liste de référence :

$$\pi_{\text{fid}}(\alpha) = \begin{array}{|c|}\hline \text{fid} \\ \hline f_1 \\ f_2 \\ f_3 \\ f_4 \\ \hline \end{array} \quad - \quad \pi_{\text{fid}}(\delta) = \begin{array}{|c|}\hline \text{fid} \\ \hline f_1 \\ f_3 \\ \hline \end{array} \quad = \quad \alpha \div \beta = \begin{array}{|c|}\hline \text{fid} \\ \hline f_2 \\ f_4 \\ \hline \end{array}$$

Version algébrique

L'expression intégrale de la division relationnelle de $\alpha(X, Y)$ par $\beta(Y)$:

$$\alpha \div \beta = \pi_X(\alpha) - \pi_X\underbrace{\left(\underbrace{(\pi_X(\alpha) \times \beta)}_{\gamma} - \alpha\right)}_{\delta}$$

La recette

Sans compter les projections, l'expression se calcule en 3 étapes :

1. construire toutes les associations de valeurs possibles (γ)
2. supprimer les associations existantes (δ)
3. supprimer les résultats intermédiaires de l'ensemble des résultats possibles

Les quatre formes en SQL

Et en SQL ?

- La plupart des présentations de la division relationnelle traitent le point de vue unique de l'algèbre
  - en ignorant la transposition SQL
  - laissant penser que la division n'est pas un opérateur utile en pratique
- Pourquoi un tel manque de considération ?
  - il n'y a pas de construction native (par exemple DIVIDE) en SQL,
  - donc l'expression SQL de la division est un peu complexe

Essayons de démystifier la chose...

Expression de la division en SQL

Il y a – au moins – quatre façons différentes :

1. traduction littérale de l'expression algébrique
2. par quantification tautologique
3. par inclusion d'ensembles
4. par comparaison de cardinalités d'ensembles

La version traditionnellement mise en avant, mais pas nécessairement la plus évidente, est la (2)

N°1 : à partir de l'expression algébrique

Rappel

$$\alpha \div \beta = \pi_X(\alpha) - \pi_X\left((\pi_X(\alpha) \times \beta) - \alpha\right)$$

- Il faut se souvenir également que, en SQL :
  - `EXCEPT` désigne la différence ensembliste ($-$)
  - l'opérateur `CROSS JOIN` exprime le produit cartésien ($\times$) dans la clause `FROM`
  - les requêtes imbriquées requièrent un alias : `(SELECT ...) AS <alias>`

N°1 : à partir de l'expression algébrique (suite)

$$\alpha \div \beta = \pi_{\text{fid}}(\alpha) - \pi_{\text{fid}}\left((\pi_{\text{fid}}(\alpha) \times \beta) - \alpha\right)$$

avec le diviseur $\alpha = \pi_{\text{fid,pid}}(\text{CPF})$, et le dividende $\beta = \pi_{\text{pid}}(\sigma_{\text{poids=17}}(\text{P}))$

```
select fid from CPF                          -- pi_fid(alpha)
    except                                   -- minus (-)
select fid from (                            -- pi_fid(.)
    (
    (select fid from CPF                     -- pi_fid(alpha)
     cross join                              -- x
    (select pid from P where poids = 17)     -- beta
    )                                        -- aka. gamma
        except                               -- minus (-)
    (select fid, pid, from CPF)              -- alpha
    ) as delta                               -- aka. delta
```


N°2 : à partir d'une quantification tautologique

Reprenons la formulation originale de la requête :

Donner les identifiants des fournisseurs qui ont livrés tous les produits de 17kg.

Envisageons une reformulation alignée sur le schéma, qui rende la quantification plus explicite :

Donner les identifiants des fournisseurs tels que pour tous les produits de 17kg, il existe une commande dans laquelle le produit est livré par le fournisseur.

Problème

Nous avons besoin d'exprimer
$\{f.\texttt{fid} \mid \text{F}(f) \wedge \forall p : \text{P}_{17}(p) \rightarrow \exists t : \text{CPF}(t) \wedge t.\texttt{fid} = f.\texttt{id} \wedge t.\texttt{pid} = p.\texttt{id}\}$,
malheureusement SQL ne dispose pas de quantificateur universel…

N°2 : à partir d'une quantification tautologique (suite)

Solution

Convoquons cette tautologie :
$\forall x, p(x) \rightarrow \exists y, q(x,y) \;\Leftrightarrow\; \neg\exists x, \neg\left(p(x) \rightarrow \exists y, q(x,y)\right) \;\Leftrightarrow\; \neg\exists x, p(x) \wedge \neg\exists y, q(x,y)$

Avant

Donner les identifiants des fournisseurs tels que pour tous les produits de 17kg, il existe une commande dans laquelle le produit est livré par le fournisseur.

Après

Donner les identifiants des fournisseurs tels qu'il n'existe pas de produit de 17kg pour lesquels il n'existe pas de commande dans la quelle le produit est livré par le fournisseur.

N°2 : à partir d'une quantification tautologique (fin)

Traduction de la double négation : `not exists... not exists...`

```
select fid from Fournisseur as F   -- alt.: distinct fid from CPF
where  not exists
       ( select 42 from Produit as P
         where  poids = 17
         and    not exists
                ( select 42 from Commande as CPF
                  where  CPF.pid = P.pid
                  and    CPF.fid = F.fid )
       );
```


N°3 : à partir de l'inclusion d'ensembles

Intuition

Si un fournisseur propose un sur-ensemble des produits de 17kg, de facto il les propose tous.

- Si seulement SQL disposait d'un opérateur d'inclusion…

Un peu de logique

- si $A \supseteq B$, alors $B \setminus A$ est vide, autrement dit « $\neg\exists(B \setminus A)$ »
- avec, dans notre cas :
  - A l'ensemble des produits de 17kg proposés par un fournisseur en particulier, et
  - B l'ensemble de tous les produits de 17kg.

N°3 : à partir de l'inclusion d'ensembles (suite et fin)

La requête SQL examine chaque identifiant de fournisseur en calculant l'ensemble A, et l'inclut au résultat si la différence avec B est vide.

```
select fid from Fournisseur as F
where not exists (
    ( select pid from Produit as P where poids = 17 )   -- B
      except
    ( select PP.pid from Produit as PP, CPF
      where  PP.pid = CPF.pid
      and    CPF.fid = F.fid
      and    PP.poids = 17 )                            -- A
    );
```

L'une des deux négations a disparu, ce qui semble plus confortable.

N°4 : à partir d'un dénombrement

Intuition

L'approche n°3 par inclusion d'ensembles invite à compter les éléments de chaque ensemble A et B, puis à comparer leurs cardinalités.

- Grâce à l'opérateur d'agrégation `count` de SQL, il est aisé de réaliser ce dénombrement

La procédure

1. calculer le nombre de produits de 17kg proposés par chaque fournisseur
2. une clause `HAVING` filtre ces quantités par comparaison au nombre total de produits de 17kg.

N°4 : à partir d'un dénombrement (suite et fin)

```
select CPF.fid from CPF, Produit as P
where CPF.pid = P.pid and P.poids = 17
group by CPF.fid
having count(P.pid) =
    ( select count(PP.pid)
        from Produit as PP
        where PP.poids = 17 );
```

- Plus aucune négation !
- Sans surprise, il s'agit bien souvent de la version plébiscitée

Conclusion

À retenir

- La division relationnelle a une portée pratique indéniable
- Son expression algébrique est complexe
- Il n'existe pas d'opérateur SQL de division
- Quatre constructions différentes permettent de traduire la division en SQL
- Des variations SQL avec `JOIN` existent, pour chaque construction

Qu'en est-il si l'on ne souhaite que les valeurs associées exactement au diviseur ?