Cours de bases de données, , http://sql.bdpedia.fr

SQL, la jointure

Philippe Rigaux

le cnam

Algèbre relationnelle : la jointure algébrique

La jointure $R \bowtie_{a=b} S$ est la composition du produit cartésien et de la sélection.

$R \bowtie_{a=b} S$ est équivalent à $\sigma_{a=b}(R \times S)$.

Opération très importante.

- Elle permet de créer une relation associant des nuplets distincts
- C'est aussi une opération potentiellement coûteuse.

Cette session présente la jointure algébrique et sa syntaxe SQL.

Ces diapositives correspondent au support en ligne disponible sur le site `http://sql.bdpedia.fr/`

Reprenons le produit cartésien Logement $\times$ Activité

code	nom	capacité	type	lieu	codeLogement	codeActivité
ca	Causses	45	Auberge	Cévennes	ca	Randonnée
ge	Génépi	134	Hôtel	Alpes	ca	Randonnée
pi	U Pinzutu	10	Gîte	Corse	ca	Randonnée
ta	Tabriz	34	Hôtel	Bretagne	ca	Randonnée
ca	Causses	45	Auberge	Cévennes	ge	Piscine
ge	Génépi	134	Hôtel	Alpes	ge	Piscine
pi	U Pinzutu	10	Gîte	Corse	ge	Piscine
ta	Tabriz	34	Hôtel	Bretagne	ge	Piscine
...	...	...	...	...	...	...

Beaucoup de lignes (probablement) sans intérêt

Jointure : Logement $\bowtie_{code=codeLogement}$ Activité

On conserve les nuplets avec le **même** code logement.

code	nom	capacité	type	lieu	codeLogement	codeActivité
ca	Causses	45	Auberge	Cévennes	ca	Randonnée
ge	Génépi	134	Hôtel	Alpes	ge	Piscine
ge	Génépi	134	Hôtel	Alpes	ge	Ski
pi	U Pinzutu	10	Gîte	Corse	pi	Plongée
pi	U Pinzutu	10	Gîte	Corse	pi	Voile

On a créé une nouvelle table associant des nuplets initialement distincts

En SQL : `join ... on ...`

La **jointure algébrique** s'effectue en SQL dans la clause `from`.

```
select *
from Logement join Activité on (code=codeLogement)
```

Rappel : la syntaxe **déclarative** est la suivante :

```
select *
from Logement as l, Activité as a
where l.code=a.codeLogement
```

Interprétations différentes, résultat identique.

Résolution des ambiguités

CetLate requête suivante renvoie une erreur à cause de l'ambiguité sur `idVoyageur`.

```
select *
from Voyageur join Séjour on (idVoyageur=idVoyageur)
```

Première solution : on énumère les attributs en effectuant des renommages.

```
select V.idVoyageur as idV1, V.nom, S.idVoyageur as idV2, début, fin
    from Voyageur as V join Séjour as S
         on (V.idVoyageur=S.idVoyageur)
```

Renommage avant jointure

Seconde solution : le renommage a lieu **avant** la jointure.

Expression algébrique :

$$\rho_{idVoyageur \to idV1}(\pi_{idVoyageur,nom}\texttt{Voyageur}) \bowtie_{idV1=idV2} \rho_{idVoyageur \to idV2}(\pi_{idVoyageur,debut,fin}\text{Séjour})$$

Requête SQL :

```
select *
from (select idVoyageur as idV1, nom from Voyageur) as V
                join
     (select idVoyageur as idV2, début, fin from Séjour) as S
            on (V.idV1=S.idV2)
```

On met l'expression algébrique dans le `from` : elle définit la relation interrogée.

Composition des jointures

On peut placer des expressions algébriques quelconques dans le `from`. Ici, deux jointures.

```
select nomVoyageur, nomLogement
from ( (select idVoyageur as idV, nom as nomVoyageur from Voyageur) as V
                    join
               Séjour as S on idV=idVoyageur)
                join
          (select code, nom as nomLogement from Logement) as L
              on codeLogement = code
```

Lisibilité aléatoire... À comparer avec la version déclarative.

À retenir

L'algèbre est un langage pour créer des relations à partir d'autres relations. Les **expressions** se placent dans le `from`.

- La jointure est une opération courante, essentielle et sensible (performances)
- Elle peut s'exprimer littéralement en SQL avec `join ... on ...`
- Ce n'est qu'une version alternative à l'expression équivalente de la syntaxe déclarative de SQL

La multiplication des jointures rend la syntaxe très lourde et peu lisible.