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

SQL, langage algébrique

Philippe Rigaux

le cnam

Algèbre relationnelle : les opérateurs

SQL propose un autre type d'interrogation, fonctionnelle, basée sur l'algèbre relationnelle.

L'algèbre est un ensemble de 6 opérateurs, qui présentent deux propriétés essentielles

- Clôture : un opérateur s'applique à des relations et produit une relation
- Composition : un opérateur peut prendre en entrée le résultat d'un autre pour définir des requêtes algébriques complexes.

Cette session présente les 6 opérateurs nécessaires et suffisants.

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

La projection

$\pi_{A_1,A_2,\ldots,A_k}(R)$ construit une relation contenant tous les nuplets de R, dans lesquels seuls les attributs $A_1, A_2, \ldots A_k$ sont conservés.

Exemple : nom et lieu des logements : $\pi_{nom,lieu}(Logement)$

```
select nom, lieu
from Logement
```

nom	lieu
Causses	Cévennes
Génépi	Alpes
U Pinzutu	Corse
Tabriz	Bretagne

`select *` permet de conserver tous les attributs.

La sélection, σ

La sélection $\sigma_F(R)$ s'applique à une relation, R, et en extrait les nuplets qui satisfont F

$$\sigma_{lieu='Corse'}(Logement)$$

En SQL :

```
select *
from Logement
where lieu = 'Corse'
```

Les comparaisons s'écrivent $A\Theta B$, où Θ appartient à $\{=, <, >, \leq, \geq\}$.

Le produit cartésien, $\times$

$R \times S$ produit une relation où chaque nuplet de R est associé à chaque nuplet de S.

A	B
a	b
x	y

C	D
c	d
u	v
x	y

Et voici le résultat de $R \times S$:

A	B	C	D
a	b	c	d
a	b	u	v
a	b	x	y
x	y	c	d
x	y	u	v
x	y	x	y

En SQL, `cross join`

En SQL, le produit cartésien s'exprime avec `cross join`. On place l'expression algébrique dans la clause `from`.

```
select *
from R cross join S
```

Raisonnement : `R cross join S` définit une nouvelle relation, les autres opérateurs (sélection, projection) s'appliquent à cette relation.

Le `from` définit donc une relation calculée

Problème des noms d'attribut ambigus

Il peut arriver que les deux relations aient des attributs qui ont le même nom.

On doit alors donner un nom distinct à chaque attribut.

Relation R :

A	B
a	b
x	y

Relation T :

A	B
m	n
o	p

$R \times T$ a pour schéma (A, B, A, B) et présente donc des ambiguités,

Renommage ρ

L'expression $\rho_{A \to C, B \to D}(T)$ renomme A en C et B en D dans la relation T.

$\rho_{A \to C, B \to D}(T)$:

C	D
m	n
o	p

Le résultat de $R \times \rho_{A \to C, B \to D}(T)$

A	B	C	D
a	b	m	n
a	b	o	p
x	y	m	n
x	y	o	p

Renommage en SQL : as

La requête

```
select Voyageur.idVoyageur, Séjour.idVoyageur
from Voyageur cross join Séjour
```

engendre une erreur *duplicate field name :*

Bonne version :

```
select Voyageur.idVoyageur as idV1, Séjour.idVoyageur as idV2
from Voyageur cross join Séjour
```

Le as permet aussi de renommer des relations. Cf. support de cours.

L'union, $\cup$

$R \cup S$ produit une relation contenant l'union de R S (qui doivent avoir le même schéma).

A	B
a	b
x	y

A	B
c	d
u	v
x	y

Et voici le résultat de $R \cup S$:

A	B
a	b
c	d
u	v
x	y

Union en SQL : `union`

La requête donnant l'union des noms de lieu et des noms de logement

```
select lieu from Logement
  union
select région as lieu from Voyageur
```

Nb : le schéma du résultat est `(lieu)`

Rarement utilisée, mais indispensable (pas d'autre expression possible) en cas de besoin

La différence, $-$

RS produit une relation contenant les nuplets de R qui ne sont pas dans S (elles doivent avoir le même schéma).

Relation R

A	B
a	b
x	y

Relation S

A	B
c	d
u	v
x	y

Relation $R - S$

A	B
a	b

En SQL

```
select lieu from Logement
  except
select région as lieu from Voyageur
```

Très peu pratique à cause de la contrainte sur les schémas. La version déclarative, `not exists`, est bien plus facile.

À retenir

- Syntaxe et signification des 6 opérateurs
- Notion de clôture : toute opération s'applique à des relations et produit une relation
- Notion de composition : on crée des requêtes complexes en combinant des opérateurs, pas en créant des opérateurs super-complexes.
- En SQL : l'algèbre opère sur des relations complètes, alors que dans la perspective déclarative, SQL exprime des conditions sur des nuplets.

Nouveauté principale pour SQL : la possibilité d'introduire des expressions dans le `from`