

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

SQL, langage algÃbrique

Algèbre relationnelle : les opérateurs

SQL propose un autre type d'interrogation, basé sur **l'algèbre relationnelle**.

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

- **Clôture** : un opérateur s'applique à des relation 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

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

Exemple : nom et lieu des logements.

$\pi_{nom, lieu}(Station)$

nom	lieu
Causses	Cévennes
Génépi	Alpes
U Pinzutu	Corse
Tabriz	Bretagne

`select *` permet de conserver tous les attributs.

Les doublons

En projetant on risque d'introduire des doublons.

```
select type from Logement
```

<u>type</u>
Auberge
Hôtel
Gîte
Hôtel

```
select distinct type from Logement
```

À n'utiliser que quand c'est nécessaire, car le "distinct" coûte cher.

La sélection, σ

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

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

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 tuple de R est associé à chaque tuple 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 T :

A	B
m	n
o	p

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

Renommage ρ

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

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

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

À retenir

- Syntaxe et signification des 6 opérateurs (union et différence : voir support de cours)
- Notion de **clôture** : toute opération s'applique à des relation 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 relation complète, alors que dans la perspective déclarative, SQL exprime des conditions sur des nuplets.

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