

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

SQL conjonctif

SQL, première partie

Cette session présente les requêtes SQL **conjonctive**, celles qui ne nécessitent ni quantificateur, ni négation.

- Forme d'une requête SQL : variables-nuplet, conditions, construction du nuplet-résultat
- Requête mono-variables
- Requêtes multi-variables

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

Variables nuplet

SQL manipule des nuplets libres de la forme $t(a_1, a_2, \dots, a_n)$

Nous les appellerons des **variables nuplet**.

La quantification porte sur la variable nuplet t : $\exists t$ et $\forall t$.

On désigne les attributs en les rattachant à t : $t.a_1$, $t.a_2$, etc.

Requête mono-variables

Les requêtes les plus simples utilisent une seule variable nuplet. Elles sont de la forme

```
select distinct t.a1, t.a2, ..., t.an
from T as t
where <condition>
```

C'est un "bloc" avec trois clauses :

- le `from` définit la variable libre et sa **portée**
- le `where` définit la condition sur la variable libre
- le `select` (avec `distinct` optionnel) construit le nuplet-résultat

La syntaxe SQL correspond à la formule logique.

$$\{t.a_1, t.a_2, \dots, t.a_n \mid T(t) \wedge F_{cond}(t)\}$$

Parlons du distinct

Une relation n'a pas de doublons. Or certaines requêtes peuvent en produire :

```
select type from Logement
```

<u>type</u>
Auberge
Hôtel
Gîte
Hôtel

Le `distinct` garantit que les doublons sont éliminés.

```
select distinct type from Logement
```

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

Premier exemple

Nom et type des logements en Corse.

```
select t.nom, t.type
from Logement as t
where t.lieu = 'Corse'
```

Correspond à la formule

$$\{t.nom, t.type \mid \text{Logement}(t) \wedge t.lieu = \text{'Corse'}\}$$

Forme simplifiée.

```
select nom, type
from Logement
where lieu = 'Corse'
```

Interprétation

La variable peut être affectée à tous les nuplets fermés de la table définie par la portée.

On garde toutes les affectations qui satisfont la condition F_{cond} .

La seule affectation correcte est surlignée ci-dessous.

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

Trivial ? Oui, et tant mieux, car cette interprétation fonctionne pour **toutes** les requêtes.

Requête multi-variables

Regardons pour deux variables : la généralisation est facile.

Forme de la requête :

```
select t1.a1, ..., t1.an, t2.a1, ..., t2.am  
from T1 as t1, T2 as t2  
where <condition>
```

Interprétation

Parmi toutes les affectations possibles des variables, on ne conserve que celles qui satisfont la condition exprimée par la condition.

Un exemple détaillé : logements où on peut pratiquer le ski

Nous avons besoin de deux variables :

- la première s'affecte aux nuplet de Activité;
- la seconde s'affecte aux nuplets de Logement
- l'attribut code de la première est Ski.
- **les deux variables partagent le même code logement**

```
select l.nom
from Logement as l, Activité as a
where l.code = a.codeLogement
and a.codeActivité = 'Ski'
```

Interprétation : affectation des deux variables

Logement (variable l)

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

Activité (variable a)

codeLogement	codeActivité
ca	Randonnée
ge	Piscine
ge	Ski
pi	Plongée
pi	Voile

Deuxième exemple : les paires de logements qui sont du même type

Nous avons besoin de deux variables,

- chacune ayant pour portée la table Logement
- les deux variables partagent le même attribut type

```
select l1.nom as nom1, l2.nom as nom2
from Logement as l1, Logement as l2
where l1.type = l2.type
```

Soit la formule

$$\{l_1.nom, l_2.nom \mid \text{Logement}(l_1) \wedge \text{Logement}(l_2) \wedge l_1.type = l_2.type\}$$

Interprétation : affectation des deux variables

Logement (variable l_1)

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

Logement (variable l_2)

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

Autre affectation possible

Logement (variable l_1)

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

Logement (variable l_2)

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

Encore une autre (et trois autres encore possibles)

Logement (variable l_1)

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

Logement (variable l_2)

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

À retenir

Quelle que soit sa complexité, l'interprétation d'une requête SQL peut **toujours** se faire de la manière suivante.

- Chaque variable du `from` peut être affectée à tous les nuplets de sa portée.
- Le `where` définit une condition sur ces variables : seules les affectations satisfaisant cette condition sont conservées
- Le nuplet résultat est construit à partir de ces affectations