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

SQL: le bloc select-from-where

Philippe Rigaux

le cnam

Récapitulatif SQL : le bloc `select-from-where`

Dans cette session : présentation de quelques extensions pratiques au SQL fondamental.

- Les valeurs nulles
- La jointure externe
- Le tri

Avant d'écouter : consulter le schéma et le contenu de la base des immeubles.

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

Valeurs nulles

Une **valeur nulle**, ou plus précisément **valeur à `null`** est une **valeur manquante**. Ne pas confondre avec la valeur "null" ou "".

Dans notre table des occupants, le prénom de Prof est à null.

id	prénom	nom	profession	idAppart
1		Prof	Enseignant	202
2	Alice	Grincheux	Cadre	103
3	Léonie	Atchoum	Stagiaire	100
4	Barnabé	Simplet	Acteur	102
5	Alphonsine	Joyeux	Rentier	201
6	Brandon	Timide	Rentier	104
7	Don-Jean	Dormeur	Musicien	200

La présence de valeurs à `null` **fausse** le résultat attendu des requêtes.

Comparaisons avec valeurs nulles

On ne sait pas comparer un valeur manquante, ou lui appliquer une fonction.

```
select * from Personne where prénom like '%'
```

Prof (pas de prénom) n'est pas trouvé.

```
select  * from Personne where prénom not like  '%'
```

Prof n'est pas trouvé non plus !

Une comparaison avec `null` ne donne ni vrai, ni faux, mais
une troisième valeur de vérité, `unknown`.

Calculs avec valeurs à null

Tout calcul appliqué avec une valeur à `null` renvoie `null` !

```
select concat(prénom, ' ', nom) as 'nomComplet'
from Personne
```

nomComplet
null
Alice Grincheux
Léonie Atchoum
Barnabé Simplet
Alphonsine Joyeux
Brandon Timide
Don-Jean Dormeur

Le test `is null`

Seule approche correcte : il faut tester explicitement l'absence de valeur avec `is null`.

En SQL :

```
select * from Personne
where prénom like '%'
or prénom is null
```

Attention le test `prénom = null` **ne marche pas**.

Conclusion : éviter autant que possible les valeurs à `null` en les interdisant (dans le schéma).

La jointure externe

On veut la liste des appartements avec leurs occupants.

```
select idImmeuble, no, niveau, surface, nom, prénom
from  Appart as a, Personne as p
where  p.idAppart=a.id
```

idI	no	niveau	surface	nom	prénom
2	2	2	250	Prof	null
1	52	5	50	Grincheux	Alice
1	1	14	150	Atchoum	Léonie
1	51	2	200	Simplet	Barnabé
2	1	1	250	Joyeux	Alphonsine
1	43	3	75	Timide	Brandon
2	10	0	150	Dormeur	Don-Jean

Il manque l'appartement 34 qui n'a pas d'occupant.

L'opérateur `outer join`

L'opérateur algébrique `outer join`

- Renvoie tous les nuplets de la table directrice (celle de gauche)
- Associe à chaque nuplet un nuplet de la table de droite **si un tel nuplet existe**
- Sinon, les attributs provenant de la table de droite sont affichés à `null`

```
select idImmeuble, no niveau, surface, nom, prénom
from  Appart as a left outer join Personne as p on (p.idAppart=a.id)
```

On obtient, en plus de la jointure standard :

1	34	50	null	null

Le tri, `order by`

On peut demander explicitement le tri du résultat sur une ou plusieurs expressions avec la clause `order by`

```
select *
from Appart
order by surface, niveau
```

En ajoutant des clauses sur l'ordre du tri (`ascending` ou `descending`)

```
select *
from Appart
order by surface desc, niveau desc
```

À retenir

Les fondements du langage SQL (logique ou algèbre) sont une base sur laquelle beaucoup d'extensions pratiques sont possibles.

- Valeurs à `null`, jointures externes, tris
- Mais aussi des fonctions, spécifiques à chaque système
- Ne change en rien **l'interprétation** du langage, que vous devez maintenant maîtriser.

SQL est un standard, mais chaque système propose des spécificités au-delà du standard.