

Bases de Données Relationnelles

Les opérations ensemblistes

L2

Rafael Angarita
Maitre de Conférences
rangarit@parisnanterre.fr

Les opérations ensemblistes

Les opérations ensemblistes en SQL, sont celles définies dans l'algèbre relationnelle. Elles sont réalisées grâce aux opérateurs :

- UNION
- INTERSECT
 - ne fait pas partie de la norme SQL et n'est donc pas implémenté dans tous les SGBD
- EXCEPT
 - ne fait pas partie de la norme SQL et n'est donc pas implémenté dans tous les SGBD
- Ces opérateurs s'utilisent entre deux clauses SELECT.

L'opération UNION

Syntaxe :

```
SELECT ...  
UNION [ALL | DISTINCT] SELECT ...  
[UNION [ALL | DISTINCT] SELECT ...]
```

UNION combine le résultat de plusieurs instructions **SELECT** dans un seul ensemble de résultats. Exemple :

```
mysql> SELECT 1, 2;  
+----+----+  
| 1 | 2 |  
+----+----+  
mysql> SELECT 'a', 'b';  
+----+----+  
| a | b |  
+----+----+  
| a | b |  
+----+----+
```

```
mysql> SELECT 1, 2 UNION SELECT 'a', 'b';  
+----+----+  
| 1 | 2 |  
+----+----+  
| 1 | 2 |  
+----+----+  
| a | b |  
+----+----+
```

Les noms de colonne d'un ensemble de résultats **UNION** sont extraits des noms de colonne de la première instruction **SELECT**.

Avec des alias :

```
SELECT 'a' AS c1, 'b' as c2 UNION SELECT 1, 2;
```

UNION DISTINCT et UNION ALL

- Par défaut, les lignes en double **sont supprimées** des résultats `UNION`. Le mot clé facultatif `DISTINCT` a le même effet mais le rend explicite.
- Avec le mot clé facultatif `ALL`, la suppression des lignes en double ne se produit pas et le résultat inclut toutes les lignes correspondantes de toutes les instructions `SELECT`.
- Vous pouvez mélanger `UNION ALL` et `UNION DISTINCT` dans la même requête. Les types `UNION` mixtes sont traités de telle sorte qu'une union `DISTINCT` remplace toute union `ALL` à sa gauche.
- Une union `DISTINCT` peut être produite explicitement en utilisant `UNION DISTINCT` ou implicitement en utilisant `UNION` sans mot-clé `DISTINCT` ou `ALL`.

ORDER BY et LIMIT dans des UNIONS

- Pour appliquer une clause **ORDER BY** ou **LIMIT** à un **SELECT** individuel, mettez le **SELECT** entre parenthèses et placez la clause entre parenthèses :

```
(SELECT a FROM t1 WHERE a=10 AND B=1 ORDER BY a LIMIT 10)
UNION
(SELECT a FROM t2 WHERE a=11 AND B=2 ORDER BY a LIMIT 10);
```

- L'utilisation de **ORDER BY** pour des instructions **SELECT** individuelles n'implique rien sur l'ordre dans lequel les lignes apparaissent dans le résultat final car **UNION** produit par défaut un ensemble de lignes non ordonné.
- **ORDER BY** dans ce contexte est généralement utilisé conjointement avec **LIMIT**, pour déterminer le sous-ensemble des lignes sélectionnées à récupérer pour le **SELECT**, même si cela n'affecte pas nécessairement l'ordre de ces lignes dans le résultat **UNION** final.
- Si **ORDER BY** apparaît sans **LIMIT** dans un **SELECT**, il est supprimé pour optimiser car il n'a aucun effet dans tous les cas.

ORDER BY et LIMIT dans des UNIONS

- Pour utiliser une clause `ORDER BY` ou `LIMIT` pour trier ou limiter l'intégralité du résultat `UNION`, mettez entre parenthèses les instructions `SELECT` individuelles et placez `ORDER BY` ou `LIMIT` après la dernière :

```
(SELECT a FROM t1 WHERE a=10 AND B=1)
UNION
(SELECT a FROM t2 WHERE a=11 AND B=2)
ORDER BY a LIMIT 10;
```

L'opération INTERSECT

Syntaxe :

```
SELECT ...  
INTERSECT [ALL | DISTINCT] SELECT ...  
[INTERSECT [ALL | DISTINCT] SELECT ...]
```

INTERSECT limite le résultat de plusieurs instructions **SELECT** aux lignes qui sont communes à toutes. Exemple:

```
mysql> SELECT * FROM a;
```

m	n
1	2
2	3
3	4

3 rows in set (0.00 sec)

```
mysql> SELECT * FROM b;
```

m	n
1	2
1	3
3	4

3 rows in set (0.00 sec)

```
mysql> SELECT * FROM c;
```

m	n
1	3
1	3
3	4

3 rows in set (0.00 sec)

```
mysql> SELECT * FROM a INTERSECT SELECT * from b;
```

m	n
1	2
3	4

2 rows in set (0.00 sec)

```
mysql> SELECT * FROM a INTERSECT SELECT * from c;
```

m	n
3	4

1 row in set (0.00 sec)

L'opération INTERSECT

Comme pour **UNION** et **EXCEPT**, si ni **DISTINCT** ni **ALL** ne sont spécifiés, la valeur par défaut est **DISTINCT**. **DISTINCT** peut supprimer les doublons de l'intersection, comme illustré ici :

```
mysql> SELECT * FROM c;
```

m	n
1	3
1	3
3	4

3 rows in set (0.00 sec)

```
mysql> SELECT * FROM c INTERSECT DISTINCT SELECT * from c;
```

m	n
1	3
3	4

2 rows in set (0.00 sec)

```
mysql> SELECT * FROM c INTERSECT ALL SELECT * from c;
```

m	n
1	3
1	3
3	4

3 rows in set (0.00 sec)

L'opération INTERSECT

- Comme pour **UNION**, les opérandes doivent avoir le même nombre de colonnes.
- Les types de colonne d'ensemble de résultats sont également déterminés comme pour **UNION**.
- **INTERSECT** a une priorité supérieure à et est évalué avant **UNION** et **EXCEPT**, de sorte que les deux instructions présentées ici sont équivalentes :

```
SELECT * FROM r EXCEPT SELECT * FROM s INTERSECT SELECT * FROM t;
```

```
SELECT * FROM r EXCEPT (SELECT * FROM s INTERSECT SELECT * FROM t);
```

L'opération INTERSECT



INTERSECT a été ajouté dans MySQL 8.0.31 (2022-10-11, General Availability)

Vous pouvez facilement simuler ce type de requête en utilisant la clause **IN**, **INNER JOIN** ou la clause **EXISTS**

L'opération EXCEPT

Syntaxe :

```
SELECT ...  
EXCEPT [ALL | DISTINCT] SELECT ...  
[EXCEPT [ALL | DISTINCT] SELECT ...]
```

EXCEPT limite le résultat de la première instruction **SELECT** aux lignes qui ne sont (également) pas trouvées dans la seconde ; par exemple :

```
mysql> SELECT * FROM a;
```

m	n
1	2
2	3
3	4

3 rows in set (0.00 sec)

```
mysql> SELECT * FROM b;
```

m	n
1	2
1	3
3	4

3 rows in set (0.00 sec)

```
mysql> SELECT * FROM c;
```

m	n
1	3
1	3
3	4

3 rows in set (0.00 sec)

```
SELECT * FROM a EXCEPT SELECT * FROM b;
```

m	n
2	3

```
SELECT * FROM a EXCEPT SELECT * FROM c;
```

m	n
1	2
2	3

```
SELECT * FROM b EXCEPT SELECT * FROM c;
```

m	n
1	2

L'opération EXCEPT



EXCEPT a été ajouté dans MySQL 8.0.31 (2022-10-11, General Availability)

Vous pouvez facilement simuler ce type de requête en utilisant la clause **LEFT JOIN**

Quand utiliser UNION, INTERSECT et EXCEPT ?

La difficulté réside dans l'identification de l'opérateur à utiliser !

- Lisez bien la question
- Identifiez des mots-clés qui donnent des indices comme 'sauf' pour EXCEPT