La pagination en SQL

Date de dernière mise à jour : 13/05/2014 à 15:14

Tout programmeur a eu affaire à ce genre de problématique : comment ne retourner qu'un certain nombre de lignes pour paginer des résultats ? Les réponses varient en fonction du **SGBDR** (Système de gestion de base de données relationnelles) que vous utilisez.

Diverses solutions existent, l'intérêt étant d'opter pour les plus portables, c'est à dire pour des solutions qui fonctionneront sur plusieurs SGBD en même temps, sans modifications (ou avec des modifications mineures).
Pour tous les exemples qui vont suivre, nous opterons pour une table nommée **alertes** disposant d'un champ nommé **id** auto-incrémenté, unique. Ces exemples fonctionneront de toute façon chez vous avec quelques adaptations. Vous pouvez demander de l'aide sur le forum SQL de VIC.

MySQL et PostgreSQL disposent d'une clause appelée **LIMIT** qui va vous permettre de ne retourner qu'un certain nombre de lignes. Avant d'utiliser cette clause, lisez mon article qui vous dira comment optimiser MySQL et pourquoi il est déconseillé d'utiliser la clause LIMIT. Cette page ayant une vertu pédagogique, nous l'emploierons néanmoins, **sachant que sur des tables contenant peu d'enregistrements, c'est une solution tout à fait valable** (ceci est moins vrai sur PostgreSQL qui dispose d'un optimiseur plus puissant que MySQL).

Avec MySQL uniquement
La clause LIMIT peut s'utiliser de différentes façons, qui ne sont pas forcément portables partout. Nous allons voir quelles sont les bonnes manières d'utiliser LIMIT dans vos requêtes SQL.

LIMIT x,n
Cette façon d'écrire LIMIT est la plus couramment employée sur Internet. C'est une erreur de l'employer car elle n'est pas portable et ne fonctionne que sur MySQL.

Lorsque vous développez des scripts, prenez toujours en compte la portabilité. Le fait de pouvoir changer de base de données en modifiant peu ou pas les requêtes SQL associées est très appréciable.
Le nombre x correspond au nombre de lignes que vous souhaitez sauter avant de récupérer les n prochaines lignes. Dans l'exemple qui suit, nous ordonnons les résultats par le champ **id** dans l'ordre ascendant, puis nous sautons 2 lignes et récupérons les 30 lignes suivantes.

```
SELECT
*
FROM
 alertes ORDER
BY
id
ASC
LIMIT
2
,
30
```

Avec MySQL et PostgreSQL
LIMIT n OFFSET x
Cette deuxième façon d'écrire LIMIT pour paginer vos scripts est meilleure que la première puisqu'elle est compatible avec **PostgreSQL** et **MySQL**.

```
SELECT
*
FROM
 alertes ORDER
BY
```

```
id
ASC
LIMIT
30
 OFFSET 2
```

Avec PostgreSQL et SQL Server 2012

SQL Server ne comprend pas la syntaxe LIMIT ... OFFSET. A la place, vous devez utiliser les instructions **ORDER BY [...] OFFSET x ROWS FETCH NEXT n ROWS ONLY**, ce qui donne pour notre table alertes :

En ordonnant les résultats par un champ spécifique

```
SELECT
*
FROM
 alertes ORDER
BY
id
ASC
 OFFSET 2
ROWS
FETCH
NEXT
30
ROWS
ONLY
```

L'utilisation de la clause ORDER BY est obligatoire avec SQL Server, alors que vous pouvez vous en passer sous PostgreSQL.

Sans ordonnancement

L'utilisation de la clause **ORDER BY** étant obligatoire avec SQL Server, il faut quand même simuler un ordonnancement, avec **ORDER BY RAND()** qui classe les lignes aléatoirement :

```
SELECT
*
FROM
 alertes ORDER
BY
RAND
(
)
 OFFSET 2
ROWS
FETCH
NEXT
30
ROWS
ONLY
```

Par contre, PostgreSQL ne comprend pas la syntaxe ORDER BY RAND(), vous pouvez la remplacer par la mention **ORDER BY 1** (que ne comprend pas SQL Server, ça serait trop simple sinon ^^)SELECT

```
*
FROM
 alertes ORDER
```

```
BY
1
 OFFSET 2
ROWS
FETCH
NEXT
30
ROWS
ONLY
```

Cette syntaxe est la plus simple à porter de PostgreSQL à SQL Server. Par contre, elle ne fonctionnera pas avec SQL Server 2008. Vous pourrez alors utiliser la version ci-après :

Avec PostgreSQL, SQL Server 2012, SQL Server 2008 et SQL Server 2005
SQL Server 2008 et 2005 ne comprennent pas la nouvelle syntaxe de la version 2012. A la place, il faut ruser en réutilisant le numéro de ligne renvoyé. Ce numéro de ligne s'obtient avec la fonction **ROW_NUMBER()**. L'avantage est que celle-ci est également comprise par la version 2012 mais aussi par PostgreSQL :

```
SELECT
*
FROM
(
SELECT
 ROW_NUMBER(
)
OVER
(
ORDER
BY
id
ASC
)
AS
 numero_ligne,
*
FROM
 alertes)
 t WHERE
 t.
numero_ligne >
2
AND
 t.
numero_ligne =
32
```

Il vous faut cette fois calculer le nombre d'enregistrements qui seront retournés en ajoutant l'offset au nombre de lignes que vous souhaitez conserver (Ici, mon OFFSET est 2, je souhaite retourner 30 lignes donc 30+2=32).

Les règles d'ordonnancement sont les mêmes que celles vues précédemment.

Cette syntaxe n'est pas la plus performante mais elle a l'avantage d'être portable. Pour des syntaxes plus performantes en SQL Server, orientez-vous vers l'instruction **SET ROWCOUNT**.