Techniques de pagination DB2 SQL avec PHP

- Les articles -

Date de mise en ligne : jeudi 5 février 2015

Description :

Dans un article publié en 2010 sur XDocs400.com, j'avais présenté la manière d'effectuer une pagination au sein d'un jeu de données, au moyen de SQL DB2. Cette technique reste bien évidemment tout à fait pertinente aujourd'hui, mais si vous développez en PHP, vous serez peut être intéressé par la technique alternative que je présente dans cet article et qui consiste à implémenter un curseur scrollable.

Environnement iSeries

Dans un article publié en 2010 sur XDocs400.com, j'avais présenté la manière d'effectuer une pagination au sein d'un jeu de données, au moyen de SQL DB2. Cette technique reste bien évidemment tout à fait pertinente aujourd'hui, mais si vous développez en PHP, vous serez peut être intéressé par la technique alternative que je présente dans cet article et qui consiste à implémenter un curseur scrollable.

En introduction de cet article, il me semble utile de revenir sur la technique de pagination DB2 SQL que j'avais présenté dans un article publié sur XDocs en septembre 2010 (fichtre ! comme le temps passe...).

Pour effectuer une pagination, il nous faut une requête de base, que nous allons "triturer" un peu dans quelques instants. J'ai choisi de baser mon exemple sur la table système bien connue SYSTABLES de la bibliothèque QSYS2.

Voici donc une requête basique, qui sélectionne toutes le colonnes de la table SYSTABLES, et sélectionne également les lignes qui appartiennent à un schéma en particulier, et ont un nom de table DB2 (nom long) commençant par une valeur variable, ou un nom de table système (nom court) commençant par une autre valeur variable (qui est très certainement la même que la précédente, mais peu importe) :

```
SELECT A.*
 FROM QSYS2/SYSTABLES A
 WHERE A.TABLE_SCHEMA = ? AND (A.TABLE_NAME LIKE ? OR A.SYSTEM_TABLE_NAME LIKE ?)
```

Cette requête en l'état ne peut pas gérer une pagination. Si l'on souhaite implémenter le mécanisme de pagination SQL de DB2, par exemple avec une pagination par plages de 10 lignes, il nous faut modifier la requête précédente pour aboutir au résultat ci-dessous :

```
SELECT foo.* FROM (
 SELECT row_number() over (ORDER BY TABLE_NAME) as rn,
 A.*
 FROM QSYS2/SYSTABLES A
 WHERE A.TABLE_SCHEMA = ? AND (A.TABLE_NAME LIKE ? OR A.SYSTEM_TABLE_NAME LIKE ?)
 ) AS foo
 WHERE foo.rn BETWEEN ? AND ?
```

La technique « full SQL » présentée ci-dessus donne de bons résultats sur des tables de taille raisonnable (difficile de donner un chiffre précis car cela dépend beaucoup de la puissance du (des) processeurs(s) de votre serveur IBM i).

Mais elle présente quelques défauts :

- La technique "full SQL" est « intrusive » dans le sens où elle nécessite de modifier la requête SQL pour y insérer un certain nombre d'éléments (modification du début du SELECT, inclusion du tri dans la clause OVER...).

- De plus, j'ai constaté qu'avec cette technique, DB2 donnait des signes d'essouflement sur des tables de très grande taille, donc si vous rencontrez des difficultés avec cette technique, je vous recommande de recourir à la technique du curseur scrollable (que nous allons voir dans un instant) qui donne de bons résultats dans la plupart des cas.

La technique du curseur scrollable peut être implémentée en PHP soit en utilisant le jeu de fonctions liées à l'extension ibm_db2, soit en utilisant la classe PDO et son jeu de méthodes. Voici deux exemples d'implémentation :

- Technique du curseur scrollable pour la librairie « ibm_db2 »

```
// on présuppose que la variable $db a été initialisée en amont au moyen de la fonction PHP db2_connect()
 function exemple ($db, $sql, $args) {
 $rows = array();
 try {
 $st = db2_prepare($db, $sql, array('cursor' => DB2_SCROLLABLE));
 if (!$st) {
 // message d'erreur à implémenter ici selon la méthode de votre choix
 } else {
 if (!db2_execute($st, $args)) {
 // message d'erreur à implémenter ici selon la méthode de votre choix
 } else {
 for (
 $tofetch = $nbl_by_page,
 $row = db2_fetch_assoc($st, $offset);
 $row !== false && $tofetch-- > 0;
 $row = db2_fetch_assoc($st)
 ) {
 $rows [] = $row;
 }
 }
 db2_free_stmt($st);
 }
 unset($st);
 } catch (Exception $e) {
 // message d'erreur à implémenter ici selon la méthode de votre choix
 }
 return $rows;
 }
```

- Technique du curseur scrollable pour PDO :

```
// on présuppose que la variable $db a été initialisée en amont au moyen de classe PDO
 function exemple_PDO ($db, $sql, $args) {
 $rows = array() ;
 try {
 $st = $db->prepare($sql, array(PDO::ATTR_CURSOR => PDO::CURSOR_SCROLL));
 $st->execute($args);

 if ($offset > 0) {
 /*
 * Un bug d'origine inconnu oblige à effectuer un premier
 * positionnement
 * sur la ligne n° 0, quand on affiche les offsets > 0
 * Dans le cas où on affiche l'offset 0, il ne faut surtout pas faire
 * ce premier positionnement, car il interfère avec celui qui est
 * effecuté par la boucle d'affichage (for).
 */
 $lost = $st->fetch(PDO::FETCH_ASSOC, PDO::FETCH_ORI_REL, 0);
```

```
}

for (
$tofetch = $nbl_by_page,
$row = $st->fetch(PDO::FETCH_ASSOC, PDO::FETCH_ORI_REL, $offset);
$row !== false && $tofetch-- > 0;
$row = $st->fetch(PDO::FETCH_ASSOC)
) {
$rows [] = $row;
if ($profiler_status) {
$profiler_nb_rows++;
}
}
unset($st);
} catch (Exception $e) {
// message d'erreur à implémenter ici selon la méthode de votre choix
}
return $rows;
}
```

La technique du curseur scrollable présente de très bonnes performances, et ce quelle que soit la taille du jeu de données renvoyé par la requête. De plus, cette technique présente l'avantage de ne pas être intrusive par rapport au code de la requête SQL à exécuter (votre requête SQL ne subit aucune modification, ce qui est appréciable surtout si elle est complexe).

A noter : la technique du curseur scrollable fonctionne très bien sous PDO avec les bases DB2 pour IBM i et DB2 Express C. Mais j'ai détecté un bug il y a quelques temps déjà, dont je n'ai pu identifier l'origine. Après pas mal de tests, j'ai trouvé une solution de contournement qui consiste à effectuer un premier positionnement sur l'offset zéro, quand on est sur l'affichage d'offset de valeur supérieure à zéro. C'est la raison des quelques lignes de commentaire que j'ai insérées dans l'exemple ci-dessus (associées au test suivant : if $offset > 0).

On notera également que la technique du curseur scrollable peut être utile dans le cadre de traitements batchs, quand on souhaite par exemple copier de gros volumes de données d'une base DB2 vers une base MySQL. Cela permet d'éviter certains phénomènes de saturation mémoire du côté du ZendServer.

Post-scriptum :

Pour des digressions autour du développement en général, et du développement web en particulier, je vous invite à me retrouver sur mon blog http://gregphplab.com