Emuler sous MySQL le principe des tables temporaire DB2

- Les articles -

Date de mise en ligne : vendredi 18 octobre 2013

Description :

J'use et j'abuse - sans vergogne - des tables temporaires DB2, car c'est un mécanisme puissant qui rend de grands services au développeur IBM i que je suis. Quoique je leur préfère dans certains cas l'usage des CTE (common table expressions), mais ce n'est pas le sujet du jour. Quoi qu'il en soit, ces deux techniques (tables temporaires et CTE) n'existent pas sous MySQL, aussi je vais vous présenter une technique permettant de pallier ce manque.

Environnement iSeries

J'use et j'abuse - sans vergogne - des tables temporaires DB2, car c'est un mécanisme puissant qui rend de grands services au développeur IBM i que je suis. Quoique je leur préfère dans certains cas l'usage des CTE (common table expressions), mais ce n'est pas le sujet du jour. Quoi qu'il en soit, ces deux techniques (tables temporaires et CTE) n'existent pas sous MySQL, aussi je vais vous présenter une technique permettant de pallier ce manque.

Pour la petite histoire, j'ai écrit récemment, en DB2 SQL, une requête faisant appel à des mécanismes spécifiques à DB2, comme la fonction rrn(), fonction qui permet de récupérer le numéro relatif de chacune des lignes d'une table. La requête en question me servait à alimenter une table DB2 temporaire, le numéro relatif récupéré via la fonction rrn() me servant ensuite pour différents usages (en l'occurrence il s'agissait d'identifier des doublons, mais j'ai aussi utilisé cette technique récemment pour récupérer des données soumises à date d'effet).

Tout en travaillant sur le développement en question, je me suis demandé comment je pourrais faire pour écrire un traitement équivalent sous MySQL. Non pas que j'en aie besoin dans l'immédiat, mais ça pourrait se révéler utile un jour ou l'autre. Je vous livre ici le fruit de mes recherches.

En explorant la documentation de MySQL, j'ai la surprise de constater qu'il n'existe pas de fonction équivalente à la fonction rrn() de DB2. Il me faut donc trouver un mécanisme de substitution. Je découvre que la fonction MySQL UUID() pourrait faire l'affaire, vu qu'elle permet de produire à la volée un identifiant unique pour chacune des lignes d'une table. On peut donc écrire sous MySQL une requête du genre :

```
select UUID() as uniq_id, * from matable ;
```

L'identifiant unique produit par la fonction UUID() est de type VARCHAR(36), c'est un peu volumineux mais peu importe, de toute façon mon objectif est d'alimenter une table temporaire, alors...

OK, j'ai mon identifiant unique, mais comment créer sous MySQL l'équivalent d'une table temporaire DB2.

MySQL est fourni en standard avec plusieurs moteurs de stockage, les plus connus - et aussi les plus utilisés - étant MyISAM et InnoDB. Mais il existe aussi le moteur MEMORY. La documentation officielle indique ceci :

"Le moteur de stockage MEMORY crée des tables dont le contenu est stocké en mémoire. /.../ Chaque table MEMORY est associée à un fichier sur le disque. Le fichier a le nom de la table, et pour extension .frm pour indiquer la définition de la table. "

On le voit, il ne s'agit pas d'une table temporaire associée à un travail particulier comme peut l'être une table temporaire DB2. Mais c'est quand même intéressant car ce type de table est automatiquement vidé lors de l'arrêt du serveur MySQL. Donc si on oublie de vider nous même une table "MEMORY" en fin de traitement, elle sera vidée tôt ou tard (il serait quand même préférable de ne pas oublier de la vider, ceci afin de ne pas surcharger la mémoire du serveur MySQL).

Il nous faut malgré tout gérer manuellement la notion de travail, et ça c'est possible via la fonction MySQL CONNECTION_ID(), qui comme l'indique la documentation officielle nous fournit "l'identifiant de connexion courant (thread_id)". La documentation indique aussi que "chaque connexion a son propre identifiant unique".

Vous pouvez essayer avec MySQL la requête suivante, qui vous renverra un identifiant unique (et à chaque fois différent), à chaque nouvelle exécution :

```
SELECT CONNECTION_ID( ) ;
```

Vous noterez que, sous MySQL, vous n'êtes pas obligé de préciser une table pivot dans la clause FROM, pour récupérer la valeur renvoyée par une fonction comme CONNECTION_ID(), mais bon, c'est un point de détail.

Pour en revenir à nos moutons, notre identifiant est fluctuant à chaque nouvelle connexion, mais tant que les requêtes sont exécutées dans la même unité de temps, ce n'est pas un problème, et c'est en tout cas suffisant pour émuler la notion de travail (au sens IBM i du terme).

Donc, mon objectif étant de créer une table temporaire, copie d'une table de tarif existante, complétée d'un identifiant unique de connexion (la colonne "conn_id", émulant la notion de travail IBM i) et d'un identifiant unique de ligne (la colonne "line_id", équivalent du numéro relatif de ligne de DB2), voici la structure de la table que j'ai créée pour l'occasion (vous noterez la présence du mot clé MEMORY en face du paramètre ENGINE) :

```
CREATE TABLE IF NOT EXISTS `tmp_prxvte` (
 `conn_id` integer not null default 0,
 `line_id` varchar(36) NOT NULL DEFAULT '',
 `codsoc` char(3) NOT NULL DEFAULT '',
 `codart` decimal(8,0) NOT NULL DEFAULT '0',
 `datdeb` date DEFAULT NULL,
 `datfin` date DEFAULT NULL,
 `pxvent` decimal(11,2) NOT NULL DEFAULT '0.00'
 ) ENGINE=MEMORY DEFAULT CHARSET=utf8;
```

Et voici la requête de recopie de la table tarif "prxvte", vers la table pseudo-temporaire "tmp_prxvte" :

```
INSERT INTO tmp_prxvte (conn_id, line_id, codsoc, codart, datdeb, datfin, pxvent)
 SELECT CONNECTION_ID(), UUID(), codsoc, codart, datdeb, datfin, pxvent FROM prxvte ;
```

Dans les requêtes de sélection qui suivront, je ne devrai pas oublier d'utiliser systématiquement la fonction CONNECTION_ID(), comme critère de sélection de la colonne "conn_id", de manière à être certain de récupérer les seules lignes qui me concernent.

Et bien sûr, il ne faudra pas oublier d'effectuer un vidage en fin de traitement, toujours avec la fonction CONNECTION_ID() :

```
DELETE FROM tmp_prxvte WHERE conn_id = CONNECTION_ID()
```

Post-scriptum :

Retrouvez l'auteur de cet article sur son blog : http://gregphplab.com