Question
Eduard Lebedyuk · Jun 23, 2016

How to keep Cache table synchronized with an external table?

I have a MySQL server with "posts" table.

I also have a Caché server with "downloadedposts" table.

They are connected from Caché to MySQL via SQL Gateway

I want to keep Caché table synced with MySQL one (MySQL "posts" table is a master copy), so periodically Caché queries MySQL server and downloads data. So far so good, and if a record appears or changes in MySQL table, Caché downloads the changes.

The problem I'm encountering is that sometimes rows would be deleted from MySQL "posts" table.

How do I synchronize deletions?

I don't want delete data from Caché table and recreate it each time because:

- It's costly
- Caché table is related to other tables
- I don't want to lose data accidentally (for example because MySQL server went under in a middle of a query)

That said, what's the best approach to syncing deletions?

Here's the one I thought up:

- (Once) Create new column "deleted" in "downloadedposts" table
- Before syncing create pool
- Start syncing and add IDs from "posts" to the pool
- After syncing execute sql:

```
UPDATE downloadedposts (Deleted) VALUES (NotInPool('postspool', ID))
```

Some additional considerations:

- IDs in "posts" and "downloadedposts" tables are corresponding and equal
- There is a lot of data, so I can't build a long string of IDs for EXISTS/IN query against MySQL

Here's my pool code:

```
Include (Utils, %occErrors)
Class Utils.Pool [ Abstract ]
{
/// Create new pool
```

```
/// w $System.Status.GetErrorText(##class(Utils.Pool).CreatePool(1))
ClassMethod CreatePool(PoolName As %String(MAXLEN=50) = "") As %Status
{
    Return:((PoolName="") || ($Length(PoolName)>50)) $$$ERROR($$$GeneralError, "Empty
 or long pool name")
    Return:..PoolExists(PoolName) $$$ERROR($$$GeneralError, "Pool already exists")
    Lock +$$$Pool(PoolName):0
     If '$TEST {
         Return $$$ERROR($$$GeneralError, "Cannot lock the pool " _ PoolName)
     }
     Kill $$$Pool(PoolName)
     Set $$$Pool(PoolName) = 0
     Return $$$OK
}
/// Add unique element to existing pool
/// w $System.Status.GetErrorText(##class(Utils.Pool).AddToPool(1, 1))
ClassMethod AddToPool(PoolName As %String, Element As %Integer) As %Status
{
    Set st = $$$OK
    If ..PoolExists(PoolName) {
        If ..NotInPool(PoolName, Element) {
            Set $$$Pool(PoolName, Element) = ""
            Set $$$Pool(PoolName) = $$$Pool(PoolName) + 1
        }
    } Else {
        Set st = $$$ERROR($$$GeneralError, "Pool does not exist")
    }
    Return st
}
/// Check that element is in pool
/// w ##class(Utils.Pool).InPool(1, 1)
ClassMethod InPool(PoolName As %String, Element As %Integer) As %Boolean [ CodeMode =
 expression, SqlName = InPool, SqlProc ]
{
$Data($$$Pool(PoolName, Element))=1
}
/// Check that element is not in pool
/// w ##class(Utils.Pool).NotInPool(1, 1)
ClassMethod NotInPool(PoolName As %String, Element As %Integer) As %Boolean [ CodeMod
e = expression, SqlName = NotInPool, SqlProc ]
{
$Data($$$Pool(PoolName, Element))'=1
}
/// Check that pool exists
ClassMethod PoolExists(PoolName As %String) As %Boolean [ CodeMode = expression ]
{
$Data($$$Pool(PoolName))
}
/// Delete existing pool
/// w $System.Status.GetErrorText(##class(Utils.Pool).DeletePool(1))
ClassMethod DeletePool(PoolName As %String) As %Status
{
    Return:((PoolName="") || ($Length(PoolName)>50)) $$$ERROR($$$GeneralError, "Empty
 or long pool name")
    Kill $$$Pool(PoolName)
    Lock -$$$Pool(PoolName)
    Return $$$OK
}
}
```

And Ultis include file:

```
#Define Pool ^Utils.Pool
```

The code is also available on GitHub.

To sum up, there are the following questions:

- Is there a more effective approach to syncing deleted rows across two tables?
- Any improvements to the pool code?

#Interoperability #JDBC #ODBC #SQL #Caché