Database - Bug #8249

Introduce a fast fill method to optimize Dataset.fill

02/06/2024 10:00 AM - Alexandru Lungu

Status:	New	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Alexandru Donica	**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**case_num:**	
vendor_id:	GCD		

Description

History

#1 - 02/06/2024 10:34 AM - Alexandru Lungu

This task is aimed to optimize the Dataset.fill implementation as it represents a bottle-neck in a large customer application.

I am attempting to follow the general optimization effort of introducing "fast-" methods for usual cases. I am constantly investigating the flow to find any potential place to optimize. I will list in this task incrementally what I could find:

1. For BufferImpl.fillRow, there is a field-by-field assignment from some source buffers to a target buffer. Often, the field mapping is quite trivial, matching names **and** offsets. I recall an optimization on BUFFER-COPY, where OrmUtils.setAllFields was implemented. This was simply moving the data from the data array source to destination. I agree that the fillRow is using direct-access to fasten up the things, but maybe we can avoid the whole iteration of the mapping if we can detect before hand that a "fastFillRow" can happen (using DMOSignatures). I am attempting to measure how much time is spent exclusively in fillRow across the whole execution.

#2 - 02/08/2024 04:28 AM - Alexandru Lungu

- Assignee set to Alexandru Donica

#3 - 02/08/2024 09:20 AM - Alexandru Donica

A little note:
DataSet.fill calls BufferImpl.fill which calls FillWorker.fill (class inside BufferImpl), which calls FillWorker.processDataSource, which calls FillWorker.fillRow.
In 10 runs of performance tests, that lasted ~319 seconds, the method call update.run() (which is called in FillWorker.processDataSource and calls FillWorker.fillRow) occupied ~2.6 seconds, almost 1% of total time. (not much compared to the rest of the method)
In the method FillWorker.processDataSource, just the first part of the code (shown below, comes before update.run()), takes ~16 seconds, so 5% of the total time.

```
            fillQuery.queryOpen();
            fillQuery.first();
            Long[] rowids = dataSource.getRestartRowids();
            if (rowids[0] != null)
            {
               fillQuery.repositionByID(rowids);
               fillQuery.getNext();
            }
            else
            {
               int value = dataSource.getRestartRow().intValue();
               if (value > 0)
               {
                  fillQuery.reposition(value);
```

```
                    fillQuery.getNext();
                }
            }
```

The rest of the method AFTER the update.run() and BEFORE the finally block, takes ~11 seconds, which is almost 3.5% of total time (the finally block takes 0.2 seconds).
Out of these 11 seconds, this bit of code spends ~7 seconds of total time.

```
    selfBuf.release();
    f (!fillOk.getValue())
    {
        BlockManager.leave();
    }
    hasMore.assign(fillQuery.getNext());
```

These are the hotter areas in this method.

#4 - 02/08/2024 02:14 PM - Alexandru Lungu

I discussed with Alexandru [ad] and there is a reasonable observation that the fill queries are usual AdaptiveQuery that use ProgressiveResults. In this case, there is no need to fetch the records progressively, as they will be iterated now anyway (with a very very low chance to be invalidated). I think that the only way to invalidate the query is to have another user delete some records in the moment of this fill.

Anyway, it seems reasonable to "hint" the query to avoid progressive approach. This way, the AdaptiveQuery will forcefully use the ScrollingResults and avoid subsequent database calls.

I think that this approach is a very safe alternative to #7991 for the moment. I will check what is the gain of such approach.

#5 - 03/05/2024 06:01 AM - Alexandru Lungu

- File hint.patch added

Constantin, this is the patch I used in an attempt to optimize ProgressiveResults into ScrollingResults when using DATASET:FILL. I don't recall which branch I used to generate this, so it may be a bit outdated (the lines may not match).

#6 - 03/07/2024 02:32 AM - Constantin Asofiei

Alexandru, using the patch helps #8363. But, I wonder if your patch is just something to do the same thing as injecting a PRESELECT option for the

query - maybe 4GL assumes 'preselect' for FILL?

#7 - 03/07/2024 02:45 AM - Alexandru Lungu

Hmm, I am not 100% familiar of how the FILL works. I was supposing it can take any arbitrary query to do the fill. So, maybe it is developers choice to use FOR EACH or PRESELECT.

Anyway, there are subtle nuances between FOR EACH using ScrollingResults and PRESELECT. With my patch still, the query can be invalidated in the time of fill. AdaptiveQuery is still listening for changes, so it can invalidate. PreselectQuery doesn't do that. So it is about what happens if another session inserts or updates a records just in the time of the fill. Will it be visible or not to the other user that does the fill? As long as there is no EXCLUSIVE-LOCK on fill operation, I suppose this problem is not relevant, as we can't guarantee consistency between sessions anyway.

#8 - 03/07/2024 02:49 AM - Constantin Asofiei

I've committed the patch to 8363a rev 15038.

#10 - 03/12/2024 03:57 AM - Constantin Asofiei

The changes in the patch were committed to trunk via 8363a, which was merged to trunk as revision 15049.

#11 - 03/27/2024 11:39 AM - Constantin Asofiei

With 8363f where DATASET:FILL is optimized to use hintFullResults, this test fails if AdaptiveQuery.hintFullResults is not limited only for if (isScrolling()) cases:

```
do transaction:
   for each order:
     delete order.
   end.
   for each orderline:
     delete orderline.
   end.
   for each item:
     delete item.
   end.

   create order.
   order.ordernum = 1.
   order.customernum = 2.
   order.orderdate = today.
   order.shipdate = today + 1.
   order.address = "abc".
   order.comment = "comment".
   release order.

   create orderline.
   orderline.ordernum = 1.
   orderline.itemnum = 1234.
   orderline.price = 456.89.
   orderline.quantity = 890.
   release orderline.

   create orderline.
   orderline.ordernum = 1.
   orderline.itemnum = 934.
   orderline.price = 96.89.
   orderline.quantity = 90.
   release orderline.

   create item.
   item.itemnum = 1234.
   item.itemname = "item1".
   item.price = 890.
   item.weight = 111.
   release item.
```

```
   create item.
   item.itemnum = 934.
   item.itemname = "item2".
   item.price = 990.
   item.weight = 911.
   release item.
end.

/* 1. Define the Temp-tables you wish to use in the ProDataSet. */
DEFINE TEMP-TABLE ttOrder BEFORE-TABLE ttOrderBefore
   FIELD OrderNum LIKE Order.OrderNum
   FIELD OrderDate LIKE Order.OrderDate
   FIELD CustName LIKE Customer.CustomerName
   FIELD Discount LIKE Customer.CustomerEmail
   FIELD RepName LIKE Customer.CustomerAddress
   INDEX OrderNum IS UNIQUE OrderNum.

DEFINE TEMP-TABLE ttOLine LIKE Orderline BEFORE-TABLE ttOLineBefore.

DEFINE TEMP-TABLE ttItem
   FIELD ItemNum LIKE Item.ItemNum
   FIELD ItemName LIKE Item.ItemName
   FIELD Price LIKE Item.Price
   INDEX ItemNum IS UNIQUE ItemNum.

/* 2. Define the DataSet by specifying the temp-tables (actually temp-table default buffers) and their relatio
nships.
      As soon as a buffer to a temp-table is used in a dataset, that *table* is considered part of the dataset
 */

DEFINE DATASET dsOrder FOR ttOrder, ttOLine, ttItem
   DATA-RELATION drOrderLine FOR ttOrder, ttOLine RELATION-FIELDS (OrderNum,OrderNum)
   DATA-RELATION drLineItem  FOR ttOLine, ttItem  RELATION-FIELDS (ItemNum,ItemNum).

/* 3. Identify, define and attach the datasources to prepare FILLing dataset from the database */
DEFINE QUERY qOrder FOR Order.
DEFINE DATA-SOURCE srcOrder FOR QUERY qOrder
   Order KEYS (OrderNum) .
DEFINE DATA-SOURCE srcOLine FOR OrderLine.
DEFINE DATA-SOURCE srcItem FOR Item.

/* 4. Attach the datasources and do the FILL() */
BUFFER ttOrder:ATTACH-DATA-SOURCE(DATA-SOURCE srcOrder:HANDLE,?,?).
BUFFER ttOLine:ATTACH-DATA-SOURCE(DATA-SOURCE srcOLine:HANDLE,?,?).
BUFFER ttItem:ATTACH-DATA-SOURCE(DATA-SOURCE srcItem:HANDLE,?,?).

/* note that only the top-level query gets prepared.
   The queries for the child tables will be prepared automatically based on the data-relations */
QUERY qOrder:QUERY-PREPARE("FOR EACH ORDER").
DATASET dsOrder:FILL().

/* 5. Detaching the Data-Sources. */

/* Note that after the FILL operation, the data in the dataset has become independent of the data source (data
base).
   The Data-source object no longer needs to be attached at this point, until another FILL or a save operation
 takes place. */
BUFFER ttOrder:DETACH-DATA-SOURCE().
BUFFER ttOLine:DETACH-DATA-SOURCE().
BUFFER ttItem:DETACH-DATA-SOURCE().

/* 6. Display the data */
def var k as int.
FOR EACH ttOrder:
   DISPLAY ttOrder.OrderNum
           ttOrder.OrderDate
   WITH FRAME ttOrderFrame 1 COL TITLE "ORDER".

   FOR EACH ttOLine of ttOrder:
      DISPLAY ttOLine.OrderNum
              ttOLine.ItemNum
              ttOLine.Price
              ttOLine.Quantity
      WITH FRAME ttOrderLineFrame 1 COL TITLE "ORDERLINE".
```

```
      FOR EACH ttitem of ttOline:
         DISPLAY ttitem.itemname
                 ttitem.ItemNum
                 ttitem.Price
         WITH FRAME ttItemFrame 1 COL TITLE "ITEM".
         k = k + 1.
         message "item" k.
         pause.
      END.

      k = k + 1.
      message "ttoline" k.
      pause.
   END.
   k = k + 1.
      message "ttorder" k.
   pause.
END.
```

The tables are from tstcasesdb.df from xfer testcases project.

The exception is this:

```
org.postgresql.util.PSQLException: Operation requires a scrollable ResultSet, but this ResultSet is FORWARD_ON
LY.
    at org.postgresql.jdbc.PgResultSet.checkScrollable(PgResultSet.java:280)
    at org.postgresql.jdbc.PgResultSet.first(PgResultSet.java:355)
    at com.mchange.v2.c3p0.impl.NewProxyResultSet.first(NewProxyResultSet.java:815)
    at com.goldencode.p2j.persist.orm.ScrollableResults.execute(ScrollableResults.java:496)
    at com.goldencode.p2j.persist.orm.ScrollableResults.first(ScrollableResults.java:166)
    at com.goldencode.p2j.persist.ScrollingResults.first(ScrollingResults.java:137)
    at com.goldencode.p2j.persist.ResultsAdapter.first(ResultsAdapter.java:130)
    at com.goldencode.p2j.persist.PreselectQuery.first(PreselectQuery.java:2638)
    at com.goldencode.p2j.persist.AdaptiveQuery.first(AdaptiveQuery.java:1309)
    at com.goldencode.p2j.persist.PreselectQuery.first(PreselectQuery.java:2601)
    at com.goldencode.p2j.persist.QueryWrapper.first(QueryWrapper.java:1980)
    at com.goldencode.p2j.persist.BufferImpl$FillWorker.processDataSource(BufferImpl.java:11610)
    at com.goldencode.p2j.persist.BufferImpl$FillWorker.fill(BufferImpl.java:11528)
```

Files

hint.patch	2.61 KB	03/05/2024	Alexandru Lungu