Database - Bug #7647

Leverage internal H2 _ROWID_ with FWD recid

07/31/2023 11:15 AM - Alexandru Lungu

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

Description

Related issues:

Related to Database - Bug #7535: Temporary tables require trailing recid comp...	**Test**

History

#1 - 07/31/2023 11:30 AM - Alexandru Lungu

H2 is keeping an internal *rowid* information for each row. For in-memory tables, this is basically the index at which the row resides in the scan-index (implemented using an ArrayList). There is a similarity between how the legacy recid works, how we reimplement it in FWD and how H2 stores it. As we already have a custom fork of FWD-H2, we can do some fine-tuning on *rowid* to exactly match the behavior of recid.

Implementation

In 4GL, FWD **and** H2, the id can be reclaimed. The reclaim process is very similar: the row id is assigned from an embedded sequence. When a row is deleted, the recid can be reclaimed. The recid is reclaimed in fact by the next inserted row.

Challenges

Just recently (#7535) Danut investigated the recid reclaiming for _temp tables and how we didn't do it right in FWD. 4GL does the whole recid reclaiming at the very end of the current transaction (or sub-transaction? - Danut, please advice). H2 however is eagerly re-assigning the recid. Therefore, this will require a bit of rework. Also, the whole "recid" field should be removed when dealing with FWD-H2. Its occurrences should be replaced with _rowid_ in FWD-H2.

Advantages

- H2 is already "appending" he *rowid* at the end of each row inside indexes, just to make them completely unique. This is exactly what we do in FWD, but not natively. In fact, we always add a trailing recid to make this happen (this means extra comparisons). This task will make all indexes one component smaller.
- Row look-up will be a lot faster. We already use direct-access to retrieve a row by its recid, but using the (recid) index. However, if we are to do the look-up on *rowid*, then the search will be almost instant.
- Fast temp-table copy is not doing a great job in regard to the recid. The INSERT INTO SELECT FROM technique will copy new records with ids generated by a H2 sequence. Therefore, there is no reclaiming when using fast-copy. This is basically a regression after implementing fast temp-table copy, but, IMHO, one that is very hard to exploit. If we move to a native *rowid*, this regression will be fixed.
- The whole FWD-side pending keys state can be removed for an extra bit of boost.

I am not sure of the degree at which we can achieve a performance boost, so I am looking forward for a custom indexed recid vs *rowid* comparison in some sample JDBC action.

#2 - 07/31/2023 02:01 PM - Ovidiu Maxiniuc

While working on first implementation of datasets I encountered an issue related to records from different temp-tables. In concrete, I needed some confirmation that the records were created and the relations between before and after images are correctly set. However, since all tables in FWD start at 0x01, it was very difficult to check whether the recid relation is correct. So I did a small change in TemporaryBuffer which allowed me to have a *decent* emulation between 4GL and FWD.

```
diff --git a/src/com/goldencode/p2j/persist/TemporaryBuffer.java b/src/com/goldencode/p2j/persist/TemporaryBuf
fer.java
--- a/src/com/goldencode/p2j/persist/TemporaryBuffer.java
+++ b/src/com/goldencode/p2j/persist/TemporaryBuffer.java    (date 1689618629121)
@@ -8457,6 +8457,8 @@
          return id;
       }

+      static Long bigIdSeed = 0x0000000000019900L - 0x2801L;
+
       /**
        * Create a temp table in the database for the given DMO interface, if it does not already
        * exist. This will also create all indexes defined for the temp table.
@@ -8476,7 +8478,7 @@
          boolean notPendingDrop = !pendingDrop.remove(dmoIface);

          // add to open tables, if not already present
-         boolean notOpen = (openTables.putIfAbsent(dmoIface, new AtomicLong()) == null);
+         boolean notOpen = (openTables.putIfAbsent(dmoIface, new AtomicLong(bigIdSeed += 0x800)) == null);

          // create table only if it doesn't already exist (don't want overhead of using IF NOT
          // EXISTS at the database)
```

The bigIdSeed needs - 0x2801 adjustments to get the a match with the first created record from 4GL (0x19900 for an individual run) and the 0x800 increment is the difference to recid intervals of the next table.

While this is not working perfectly (maybe the tables were not created in same order, I do not know what happens when the 0x800 gap is exausted), this may be a small help for this task.

#3 - 08/04/2023 06:44 AM - Alexandru Lungu

- Related to Bug #7535: Temporary tables require trailing recid component even for unique indexes added

#4 - 08/17/2023 03:20 AM - Alexandru Lungu

- Assignee set to Dănuț Filimon