Database - Bug #11757

Audit browse row numbering for indexed-reposition (random access) negative row indexes

08/18/2026 07:11 AM - Alexandru Lungu

Status:	WIP	**Start date:**	
Priority:	High	**Due date:**	
Assignee:	Stanislav Lomany	**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**reviewer:**	Alexandru Lungu
vendor_id:	GCD	**production:**	No
case_num:		**env_name:**	
version_reported:		**topics:**	
version_resolved:			

Description

History

#1 - 08/18/2026 07:17 AM - Alexandru Lungu

- Priority changed from Normal to High

- Assignee set to Stanislav Lomany

Follow-up on #11596. That ticket fixed FETCH-SELECTED-ROW itself (11596a/11596b); this one collects the surrounding code that was never audited when indexed reposition introduced a second row numbering space. Line numbers are against trunk rev. 16707.

The invariant

mode	query row (currentRowImpl())	browse/UI row
scrolling	1-based absolute	queryRow - 1
random access (after INDEXED-REPOSITION)	signed relative, **0 = the reposition pivot**	queryRow, identical

New rows are numbered from Browse.NEW_ROWS_NUMERATION_START (-2000000000) downwards in both modes. Hence: -1 is a valid row index and cannot be a "none" sentinel, x >= 0 is not a validity test, and the +/- 1 conversion is never unconditional. Correct implementations to copy: BrowseWidget.getCurrentRow() :6121 and setCurrentRow() :6028-6100.

To investigate

- **G1 - targetRepositionRow is off by one.** AbstractQuery.java:1792 computes row.isUnknown() ? -1 : row.intValue() - 1 unconditionally, and that value reaches Browse.queryRepositioned as a UI row. After reposition-to-rowid the pivot (query row 0) is reported as -1, which Browse.refreshScrollRow :7455 then treats as its "no target" sentinel. The predicate must be the cursor state (CursorWrapper.isRandomAccess() :887), not isIndexedReposition(). Most likely cause of the residual #11596-2 failure.
- **G2 - the renumbering offset is dropped.** getRows() nulls indexedRepositionOffset at entry (BrowseWidget:5302) and returns it at :5735, so an offset produced by a reposition elsewhere - fetchSelectedRow :2998, deleteRow :9474, isNewRow :6005 - is discarded and the client never renumbers. The server then scrolls while the client still believes it is in random access mode: permanent desync. Reachable from FETCH-SELECTED-ROW, so #11596 is not fully closed without this.
- **G3 - the selection is not renumbered on revert.** Browse.revertedFromIndexedReposition :6431-6464 shifts dcData, currentRow, editedRow and friends, but not selectedRowIndices or lastSelectedRow, so MULTIPLE browses keep stale relative indexes after the cursor reverts to scrolling. Note the asymmetry: entering random access mode clears the selection (:7881), leaving it does nothing.
- **G4 - sibling sites with the same unaudited conversion.** Wrong offset and/or >= 0 gate in BrowseWidget.deleteRow :9471-9474, deleteResultListEntry :4966, isNewRow(int) :6005, getRows :5434 and :5697. -1 sentinel collisions in Browse.refreshAfterDeletion :7506, getMaximalSelectedRowIndex :3089, refreshScrollRow :7441, scrollRows :9042-9047 and queryRepositioned :7915.
- **G5 - root cause.** BrowseWidget.reposition(int64, boolean) :9494 is documented as "1-based row to set" but is called with three different conventions: UI index plus the mode aware offset (:2998, :5916, :6075), a raw query row (:5539, :6625), and a UI index treated as a query row (:6005, :9474, i.e. the G4 bugs). Adding explicit toQueryRow() / toUiRow() helpers is what stops this class of bug recurring.
- **G6 - FETCH-SELECTED-ROW error handling deviates from its siblings.** It uses displayError instead of recordOrShowError, lacks the checkNoRecords and checkInRowDisplayTrigger guards, and FETCH-SELECTED-ROW(0) returns FALSE silently instead of raising 385. Left unchanged in 11596b because each item needs an OE check first.

- **G7 - test coverage.** TestFetchSelectedRow.cls asserts only the logical return, never which record landed in the buffer, which is precisely why the off-by-one survived. There is no coverage at all for INDEXED-REPOSITION, negative indexes, new rows, MULTIPLE browses, or the revert-to-scrolling path.

Stanislav, please check what side-issues from #11596 were discovered. Let me know which ones are fit for client-side browse fixing and do the required changes in this task. I will do the same for any server-side issues and I will generate some tests before-hand.

#4 - 08/21/2026 09:01 AM - Stanislav Lomany

- Status changed from New to WIP

- reviewer Alexandru Lungu added

#5 - 08/24/2026 02:44 PM - Stanislav Lomany

FYI currently I'm working on DELETE-CURRENT-ROW() issues for INDEXED-REPOSITION case. I met them right away while working on on the first (G1) task.

#6 - 08/25/2026 01:30 PM - Stanislav Lomany

Created task branch 11757a from FWD trunk revision 16721.

#7 - 09/01/2026 10:23 AM - Stanislav Lomany

- File targetRepositionRow-single.p added

Alexandru, I fixed several client-issues related to DELETE-* browse functions, but there are two server-side issues. I investigated them, but I didn't come to a good solution.

1. Abend happens when the number of positive items in ArrayWithNegativeIndexes become zero. ArrayWithNegativeIndexes:getLastIndex() by declaration cannot be below 0 and RandomAccessCursor:getResult fails to properly check the bounds:

```
this.results.getFirstIndex() <= index && index <= this.results.getLastIndex()
```

Please use branch 11757a and the attached testcase. Reproduction:

1. press R;
2. hold D until abend.

Stacktrace:

```
Caused by: java.lang.IndexOutOfBoundsException: Index 0 out of bounds for length 0
    at java.base/jdk.internal.util.Preconditions.outOfBounds(Preconditions.java:64)
    at java.base/jdk.internal.util.Preconditions.outOfBoundsCheckIndex(Preconditions.java:70)
    at java.base/jdk.internal.util.Preconditions.checkIndex(Preconditions.java:266)
    at java.base/java.util.Objects.checkIndex(Objects.java:361)
    at java.base/java.util.ArrayList.get(ArrayList.java:427)
    at com.goldencode.p2j.util.ArrayWithNegativeIndexes.get(ArrayWithNegativeIndexes.java:111)
    at com.goldencode.p2j.persist.RandomAccessCursor.getResult(RandomAccessCursor.java:1217)
    at com.goldencode.p2j.persist.RandomAccessCursor.getNext(RandomAccessCursor.java:942)
    at com.goldencode.p2j.persist.RandomAccessCursor.getNext(RandomAccessCursor.java:585)
    at com.goldencode.p2j.persist.CursorWrapper.getNext(CursorWrapper.java:659)
    at com.goldencode.p2j.persist.AdaptiveQuery.next(AdaptiveQuery.java:2277)
    at com.goldencode.p2j.persist.AbstractQuery.afterReposition(AbstractQuery.java:1787)
    at com.goldencode.p2j.persist.AdaptiveQuery.afterReposition(AdaptiveQuery.java:2190)
    at com.goldencode.p2j.persist.AdaptiveQuery.reposition(AdaptiveQuery.java:1676)
    at com.goldencode.p2j.persist.QueryWrapper.reposition(QueryWrapper.java:4081)
    at com.goldencode.p2j.ui.BrowseWidget.reposition(BrowseWidget.java:9636)
    at com.goldencode.p2j.ui.BrowseWidget.getRows(BrowseWidget.java:5581)
    at com.goldencode.p2j.ui.BrowseWidget.getRows(BrowseWidget.java:5179)
    at com.goldencode.p2j.ui.LogicalTerminal.getRows(LogicalTerminal.java:15220)
```

2. After all elements have been deleted from the scrolling query, the query may be refilled again from the underlying results.

Reproduction:

1. Press DOWN two times to select the second row.
2. Hold D - you'll see that the first row will loop.

Files

targetRepositionRow-single.p	859 Bytes	09/01/2026	Stanislav Lomany