Database - Support #6855

database/persistence tests

10/18/2022 06:49 PM - Greg Shah

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

Description

History

#1 - 10/22/2022 11:51 AM - Greg Shah

We need testcases that explore the following as comprehensively as possible. The intention is to provide complete coverage of the 4GL compatibility of the "database/persistence" support.

- Queries and Query-Integrated Blocks (all forms including multi-table forms)
  - FIND
    - FIRST/LAST/NEXT/PREVIOUS, plus the implicit unique find
    - FIND literal (i.e., the short form for find the unique record with the literal value on the primary (unique) index)
  - FOR blocks and loops including FIRST, LAST, EACH and no qualifier
  - DO PRESELECT
  - REPEAT PRESELECT
  - OPEN QUERY
    - with and without DEFINE QUERY
    - multiple OPEN QUERY statements sharing same DEFINE QUERY
  - other features
    - SCROLLING queries
      - NEXT, PREV, FIRST, CURRENT and LAST along with QUERY-OFF-END behavior
      - REPOSITION with ROWID, RECID, ROW, FORWARDS and BACKWARDS
      - INDEXED-REPOSITION
      - NUM-RESULTS
    - BREAK keyword generating groups including ACCUM, ACCUMULATE, FIRST-OF, LAST-OF
    - BY clause used for ordering is important
    - default ordering if no BY clause exists
      - single-table query on a table (without index)
      - single-table query on a table (with index)
      - multi-table query on tables (with/without index combinations)
      - DESCENDING
    - [LEFT] OUTER-JOIN clause
    - MAX-ROWS limit
    - FIELDS/EXCEPT options
    - all non-FIND types should have some multi-table tests:
      - all permanent tables
      - all temp-tables
      - mixture of permanent tables and temp-tables
- WHERE clauses (expression processing)
  - literals of all data types and unknown value
  - all field data types (some only are possible in temp-tables)
  - operators including precedence (including CONTAINS and :: which are database-specific)
    - logical OR
    - logical AND
    - bitwise OR
    - bitwise XOR
    - bitwise AND
    - logical NOT
    - bitwise NOT
    - =, EQ, <>, NE, <, LT, >, GT, ≤, LE, ≥, GE, MATCHES, BEGINS, **CONTAINS**
    - binary +, binary -
    - *, /, MODULO
    - unary +, unary -
    - : and **::**

        - ()
    - function calls including:
        - built-in functions (for which we have SQL UDFs)
        - built-ins which can't be UDFs (ECF: aren't there some 4GL built-in functions which cannot be or are not implemented as a database UDF?)
        - functions implemented in 4GL code:
            - which do not reference the query's buffer
            - which do reference the query's buffer
    - handle-based attributes and methods, including chaining
    - system handles, "instance" handles and system handles that are referenced by instance handles
    - OO properties, data members, class event references, method calls including chaining and both instance and static references
    - extent subscripting
        - for both complex expressions and literals
        - out of bounds subscripts (above and below)
        - ranges (e.g. var[1 for 2], to the degree they can be referenced in expressions)
- Buffer Scoping (make sure to start with the cases in our documentation, scroll down to "Record Scopes", these were used to initially identify the scoping rules)
- Flushing, Validation
    - the goal is to have a large variety of tests which together define the behavior of when newly created records, or records read from the database and updated, need to be validated and flushed (back) to the database; we need to understand:
        - are validation and flushing joined at the hip, or are there cases when they are performed separately?
        - what are the conditions under which a newly created record is validated? flushed?
        - what are the conditions under which an updated record is validated? flushed?
        - what is the relationship between validation/flushing of:
            - field updates?
            - index updates?
            - partial index updates?
            - does it differ between newly created records and records read from the database?
        - what are the conditions under which a write triggers should fire (or should not fire), and is this related to validation and flushing?
- Triggers
    - only schema based / only session based / both kind of triggers defined
    - trigger events: CREATE, DELETE, FIND, WRITE and ASSIGN
    - NEW BUFFER and OLD BUFFER in WRITE session-based triggers
    - ensure trigger is fired at the right time: when the buffer is released or the transaction ends (other cases)
    - triggers firing other triggers / how we avoid executing the same trigger in an infinite loop
    - the effect of return error/no-apply in database triggers
- Locking
    - NO-LOCK
    - SHARE-LOCK (including implicit usage)
    - EXCLUSVE-LOCK
    - NO-WAIT
    - how locks get released (implicitly and explicitly via RELEASE), including interactions with buffer scoping
    - how locks get downgraded, including interactions with buffer scoping
    - remote database tests
- CAN-FIND()
    - in a WHERE clause, nested at various levels (2, 3, 4)
        - using AND CAN-FIND(...)
        - using OR CAN-FIND(...)
    - outside of a WHERE clause
    - default lock (NO-LOCK)
    - SHARE-LOCK/EXCLUSIVE-LOCK
- Sequences
    - minimal/default definition
    - using all options
- Assignments
    - = assignment operator for fields
    - ASSIGN statement database-specific forms
    - batching (how multiple database field assignments differ in an ASSIGN statement versus implementing the same changes in separate assigment statements)
    - NO-ERROR
- Built-in Functions
    - all database built-ins
    - include "special" ones like the various record funcs which take non-regular syntax (things that cannot normally exist in an expression)
    - non-UDFs (database related functions that can only be called from business logic, not in WHERE clauses)
    - UDFs (built-ins used in WHERE clauses) ECF: this may not be different than above, I am including it here to be complete
- Table Copying, Emptying and Comparison
    - COPY-TEMP-TABLE handle based method
    - EMPTY-TEMP-TABLE statement
    - BUFFER-COPY statement and handle based method
        - same table and different tables with different structure
        - various options, including:
            - EXCEPT/USING
            - ASSIGN
            - NO-LOBS
    - BUFFER-COMPARE statement and handle based method

        - same table and different tables with different structure
        - various options, including
            - EXCEPT/USING
            - CASE-SENSITIVE/BINARY
            - SAVE [RESULT IN]
            - [EXPLICIT] COMPARES
            - NO-LOBS
- Parameter Passing
    - Modes
        - input
        - input-output
        - output
        - return (where possible)
    - Types
        - buffer
        - dataset
        - dataset-handle
        - table
        - table-handle
    - Other options
        - APPEND
        - BY-VALUE (default), BY-REFERENCE, BIND (for the types that support these options)
    - functions
    - internal/external procs
    - OO methods
    - extents
- before table and related method/attributes/scope
- Shared Resources
    - buffers
    - queries
    - temp-tables
- Other Extent Support
    - dynamic extents
    - range extents (outside of expressions)
    - unsubscripted extent references
    - initialization
    - assignment (including bulk and individual element)
- Dynamic Database
    - queries
    - temp-tables
    - validation expression
    - ::
    - buffer handle
    - buffer-field handle
- Word Indexes
- SAVE CACHE
    - not sure how to test this; we implement this feature differently than OE (see Non-Standard Save Cache Implementation)
- Handle and System Handles (including all of their attributes and methods)
    - buffer objects
    - buffer fields
    - data-relation
    - data-source
    - query object
- Security
    - (TODO IAS, OM: need test ideas here)
- I18N
    - UTF-8 and other CPs (stick to those we support currently)
- CLOB, BLOB, database-related COPY-LOB
- multi-tenant
    - TBD how these will look and what 4GL features we will support; currently, we are taking a database-per-tenant approach to implementation
- multi-database, database connect/disconnect, connect options
- cross-database joins
    - across permanent databases
    - across a permanent database and the temp-table database (note that in the FWD implementation, temp-tables and permanent tables are in separate databases; this affects the conversion of such joins, even if OE handles it differently)
- schema features
    - unique and primary constraints of indexes (is this already part of "Validation" bullet?)
    - mandatory fields (is this already part of "Validation" bullet?)
    - INITIAL, POSITION, ORDER, DECIMALS, DESCRIPTION, LABEL, FORMAT specification of fields
    - schema triggers
- ProDataSets, data sources, data relations
- XML and JSON serialization/deserialization
- two-phase commit (future; not implemented in FWD yet)

Some of the above items (e.g. ProDataSets) already have testcases and can be included once they are reworked in #6858. I'm just trying to have a

complete list that we can use to confirm when we are done.

I expect these tests to be split into smaller functional groupings that can be run on their own. I don't expect a single set of tests which includes all of these categories.

Initial priorities:

- validation/flushing
- triggers (especially write triggers)
- datasets
- open for discussion, in case starting with simpler constructs and building on them makes more sense...

#2 - 11/06/2023 04:23 AM - Marian Edu

- Status changed from New to WIP