Database - Feature #2115

Feature # 1581 (Closed): add conversion and runtime support for certain metadata constructs

add runtime support for _file and _field metadata tables

04/08/2013 10:33 PM - Eric Faulhaber

Status:	Closed	**Start date:**	04/17/2013
Priority:	Normal	**Due date:**	05/08/2013
Assignee:	Eric Faulhaber	**% Done:**	100%
Category:		**Estimated time:**	120.00 hours
Target version:	Runtime Support for Server Features		
billable:	No	**version:**	
vendor_id:	GCD		

Description

History

#1 - 04/08/2013 10:37 PM - Eric Faulhaber

- Target version set to Milestone 7

#2 - 04/17/2013 01:04 PM - Eric Faulhaber

Minimum fields we need to support in _File table due to customer use:

- _file-name
- _dump-name
- _file-label
- _desc
- _hidden
- _crc
- _file-number
- _prime-index
- _tbl-type

In addition, we most likely will need the following fields in the _File table for internal runtime use:

- _numfld

#3 - 04/17/2013 01:57 PM - Eric Faulhaber

Minimum fields we need to support in _Field table due to customer use:

- _field-name
- _file-recid
- _label
- _data-type
- _extent
- _order

In addition, we most likely will need the following fields in the _Field table for internal runtime use (primarily to implement BUFFER-FIELD attributes):

- _initial
- _format
- _col-label
- _valmsg
- _valexp
- _mandatory
- _field-physpos or _field-rpos (presumably, for POSITION attribute)

#4 - 04/17/2013 02:11 PM - Eric Faulhaber

We need a runtime facility to map between legacy (4GL) file/field names and converted (P2J) file/field names, for things like error messages, dynamic query and temp-table support, embedded SQL support. One fairly clean way to provide this is to add some P2J-specific fields (e.g., _java-name, _sql-name) to the _File and _Field (also I suppose, the _Index and _Index-field) metadata tables.

#5 - 04/17/2013 02:53 PM - Eric Faulhaber

As alluded to above, it appears that the metadata tables are the backing data store for many of the handle-based attributes of BUFFER and BUFFER-FIELD. We need some 4GL tests to confirm that, at least for the fields we care about (i.e., in notes 2 & 3 above), the following assumptions hold true:

- a change to a writable attribute changes the corresponding, backing metadata record;
- a change to a metadata record changes the corresponding, readable, handle-based attribute;
- dynamically creating a temp-table adds the corresponding records to the _file, _field, _index, and _index-field metadata tables (and deleting the temp-table removes these records);
- all metadata for all databases an application uses are shared in a single metadata database instance (there is a _db table, and _file has a _db-recid field, so this suggests a single metadata database instance -- the database qualifier to _file references in application code must internally trigger a join between the _file and _db tables).

The current plan is to build an embedded, in-memory, H2 database to back the _file, _field, _index, _index-field, _lock, _user [and _db?] metadata tables. The [_db,] _file, _field, _index, and _index-field tables will be initialized/loaded at server startup time from static data, either gathered at conversion or exported from the application (which way is yet TBD), or some combination of both. Runtime code which dynamically creates/destroys temp-tables must make the corresponding metadata record updates.

Application access to these metadata tables already should be properly synchronized by record locking in the converted business logic; however, access from the runtime will need to be synchronized as well, using manual record-locking.

Currently, the metadata DMOs are created at conversion time, but we should create cut-down, hand-written Java classes for these which include the fields we care about from notes 2 & 3 above, plus the custom additions (e.g., javaName, sqlName) in note 4.

We will need a new class to implement this core functionality (MetadataManager?). It will be called from Persistence.initialize to initialize the metadata, and will be used by other parts of the persistence runtime for dynamic temp-table support, attribute services, and other support purposes.

#6 - 04/17/2013 02:57 PM - Eric Faulhaber

- Assignee set to Ovidiu Maxiniuc

- Estimated time set to 120.00

#7 - 04/17/2013 03:15 PM - Eric Faulhaber

Proposed development approach:

- perform testing in note 5 to determine whether assumptions are correct;
- implement MetadataManager to create embedded, in-memory, H2 database and initialize it (will involve adding conversion-time resources, like .p2o files or 4GL export files to application jar file to enable loading of metadata).

MetadataManager may need to provide some helper methods for those who need runtime metadata services, but I envision specific metadata queries being handled by those who actually implement these services (i.e., that is not part of this task, but may require some coordination among team members).

For an example of using an embedded H2 database with shared tables, see the com.goldencode.p2j.persist.dirty package, though I'm not sure if we should use temporary or regular tables.

#8 - 04/19/2013 08:58 AM - Eric Faulhaber

- Assignee changed from Ovidiu Maxiniuc to Eric Faulhaber

#9 - 04/25/2013 10:48 AM - Eric Faulhaber

- Start date set to 04/17/2013

- Due date set to 05/08/2013

#10 - 08/16/2013 06:57 AM - Vadim Nebogatov

CREATE DATABASE statement (not supported now) will be one more place for creating metadata tables.

#11 - 08/20/2013 05:18 AM - Vadim Nebogatov

Will tables _File, _Field, _Index and so on have some unique fields like file_id, field_id, index_id, ... for joining these tables when needed? For example, in order to find table for concrete index from _Index, field from _Field.

#12 - 08/30/2013 02:11 PM - Eric Faulhaber

Sorry I missed this earlier. Yes, these tables have a RECID field, generally of the form {table_name}-recid. But, I plan on hiding this behind the MetaSchema API. Is there a reason you would need to join them directly to support some runtime feature that is not well suited to using the MetaSchema interface?

#13 - 08/30/2013 02:16 PM - Vadim Nebogatov

No reasons, I also think they could be hidden behind MetaSchema API.

#14 - 08/30/2013 02:22 PM - Vadim Nebogatov

I asked only to understand tables structures.

#15 - 10/18/2013 01:03 PM - Eric Faulhaber

- File ecf_upd20131015a.zip added

This is an interim drop of my implementation of this feature. It has failed regression testing so far, but contains several updates that may be needed by other developers before I am able to commit the full update to bzr.

#16 - 10/28/2013 12:18 AM - Eric Faulhaber

- File ecf_upd20131024a.zip added

This update passed regression testing and is committed to bzr rev 10407. This drop does not include the MetaSchema interface (nor its implementation), and does not resolve an existing flaw in recording natural joins between metadata tables (which operate differently than joins between "regular" 4GL tables; the meta tables use a primary-foreign key join, rather than the "implicit" joining that other tables do).

#17 - 10/28/2013 12:37 AM - Eric Faulhaber

- % Done changed from 0 to 70

- Status changed from New to WIP

#18 - 01/17/2014 04:31 PM - Eric Faulhaber

- Status changed from WIP to Closed

- File ecf_upd20140115a.zip added

- % Done changed from 70 to 100

This update fixes metadata foreign key detection, completes basic lock metadata support (#2117), removes underscore from Meta_User DMO name. It also fixes a conversion memory leak during record scope processing. It has passed regression testing and is committed as bzr rev 10441.

#19 - 11/16/2016 11:42 AM - Greg Shah

- Target version changed from Milestone 7 to Runtime Support for Server Features

Files

ecf_upd20131015a.zip	398 KB	10/18/2013	Eric Faulhaber
ecf_upd20131024a.zip	390 KB	10/28/2013	Eric Faulhaber
ecf_upd20140115a.zip	85.3 KB	01/17/2014	Eric Faulhaber