Database - Feature #2126

add support for table-level locking (share and exclusive) for runtime internal use

04/17/2013 10:27 PM - Eric Faulhaber

Status:	Closed	**Start date:**	06/27/2013
Priority:	Normal	**Due date:**	07/01/2013
Assignee:	Vadim Nebogatov	**% Done:**	100%
Category:		**Estimated time:**	16.00 hours
Target version:	Runtime Support for Server Features		
billable:	No	**version:**	
vendor_id:	GCD		

Description

Related issues:

Related to Database - Feature #2043: add runtime support for some embedded SQ...	**Closed**	**05/13/2013**	**05/31/2013**
Related to Database - Feature #1588: add conversion and runtime support for e...	**Closed**	**02/14/2013**	**05/31/2013**

History

#1 - 04/17/2013 10:40 PM - Eric Faulhaber

This feature is needed for permanent tables to enable bulk delete with embedded SQL and with delete loops optimized by conversion. The idea is that any record-level share or exclusive lock must first obtain a share lock on the associated table and any bulk delete operation must first obtain an exclusive lock on the table.

The feature will require changes to the LockManager interface (addition of a lockTable(LockType) method) and to the implementation of InMemoryLockManager (to implement lockTable(LockType) and internally, to obtain/release table locks properly when obtaining/releasing record locks).

To avoid deadlock, the table lock must always be obtained first and released last.

#2 - 04/25/2013 03:54 PM - Eric Faulhaber

- Due date set to 07/01/2013

- Start date set to 06/27/2013

- Assignee set to Eric Faulhaber

#3 - 07/31/2013 05:29 PM - Eric Faulhaber

- Assignee changed from Eric Faulhaber to Vadim Nebogatov

Correction: lockTable(LockType lockType) should actually be lockTable(String table, LockType lockType), where table is a name normally used to create a RecordIdentifier instance.

Internally, InMemoryLockManager should use a special RecordIdentifier to represent the table being locked. Use a constant, non-integer recordID parameter for the RecordIdentifier c'tor, as in:

```
private static final String TABLE_ID = "T";
...
public void lockTable(String table, LockType lockType)
{
   RecordIdentifier ident = new RecordIdentifier(table, TABLE_ID);
   ...
}
```

Before obtaining any record lock, InMemoryLockManager should call lockTable(table, LockType.SHARE) to acquire a share lock on the table. After releasing any record lock, it should call lockTable(table, LockType.NONE) to release that lock. These operations should be atomic for a given thread.

Code which requires a table-wide operation (like delete from) should call lockTable(table, LockType.EXCLUSIVE) (and of course release that lock when the operation is done).

#4 - 08/01/2013 05:43 PM - Vadim Nebogatov

Is the any sense in adding also boolean update parameter like for lock method?

#5 - 08/02/2013 05:00 AM - Eric Faulhaber

Yes, that's a good point. The update parameter will need to be passed through from the lock method.

#6 - 08/05/2013 10:30 AM - Vadim Nebogatov

Several records could be locked from one context and require SHARE-LOCK for the same table.
I suggest to add one more map to class LockStatus:

```
 /** Map of reference counts for records for given context */
private final Map<SessionToken, Integer> recordsCounterMap =  new HashMap<SessionToken, Integer>();
```

which will contain

- if it is table lock thanks to record locking: count of locked records per each context. If table is already locked, newly locked record will only increment this amount. Lock will be tried to release only if last record from given context is unlocked (as usually taking to account if table is locked from other contexts.)
- empty (or null?) if it is record lock or usual table lock (happened not from record locking)

#7 - 08/05/2013 07:43 PM - Eric Faulhaber

Good point. Please see the class RecordLockContext, which already deals with this issue (it is a bit more complicated than you describe) at the record level. I think you will need to integrate your solution with this class.

#8 - 08/07/2013 02:59 PM - Eric Faulhaber

It seems it might be best to leave the LockManager interface and InMemoryLockManager class alone after all.

Given the context issue, I think it would be better to implement a TableIdentifier subclass of RecordIdentifier. The TableIdentifier c'tor would accept only a table name, and would assign a static (non-integer) identifier (like "T") for the recordID.

I think the desired effect can be achieved with changes to RecordLockContext and RecordBuffer (in setPinnedLockType and in the bulk delete methods). Those changes would track the pinned lock state of a table, or lock/unlock/downgrade a table lock, using an instance of TableIdentifier, at the same time the lock state data of an associated RecordIdentifier were changed.

Make sense?

#9 - 08/07/2013 03:16 PM - Vadim Nebogatov

Eric Faulhaber wrote:

> It seems it might be best to leave the LockManager interface and InMemoryLockManager class alone after all.
>
> Given the context issue, I think it would be better to implement a TableIdentifier subclass of RecordIdentifier. The TableIdentifier c'tor would accept only a table name, and would assign a static (non-integer) identifier (like "T") for the recordID.

I made with close way

```
public static RecordIdentifier createTableIdentifier(String table)
{
   return new RecordIdentifier(table, TABLE_ID);
}
```

```
public boolean isTableIdentifier()
{
   return TABLE_ID.equals(recordID);
}
```

In case of separate TableIdentifier class we will need to use instance of TableIdentifier to differ TableIdentifier from RecordIdentifier.

> I think the desired effect can be achieved with changes to RecordLockContext and RecordBuffer (in setPinnedLockType and in the bulk delete methods). Those changes would track the pinned lock state of a table, or lock/unlock/downgrade a table lock, using an instance of TableIdentifier, at the same time the lock state data of an associated RecordIdentifier were changed.
>
> Make sense?

I will check. I am still in investigation stage so far.

#10 - 08/07/2013 03:21 PM - Eric Faulhaber

Vadim Nebogatov wrote:

> In case of separate TableIdentifier class we will need to use instanceof TableIdentifier to differ TableIdentifier from RecordIdentifier.

All the existing lock-related APIs are set up to deal with RecordIdentifier. Where/why would you need to differentiate instance types?

#11 - 08/07/2013 03:31 PM - Vadim Nebogatov

Eric Faulhaber wrote:

> Vadim Nebogatov wrote:
>
> > In case of separate TableIdentifier class we will need to use instanceof TableIdentifier to differ TableIdentifier from RecordIdentifier.
>
> All the existing lock-related APIs are set up to deal with RecordIdentifier. Where/why would you need to differentiate instance types?

I thought about reusing lock() method in lockTable() like

```
  public void lockTable(String table, LockType lockType, boolean update)
   throws LockUnavailableException
   {
      RecordIdentifier ident = RecordIdentifier.createTableIdentifier(table);
      lock(lockType, ident, update);
   }
```

and implement all table lock checks and changes in lock() method itself like

```
   public void lock(LockType lockType, RecordIdentifier ident, boolean update)
   throws LockUnavailableException
   {
…
                if (update)
                {
                   status = (trace
                             ? new TraceLockStatus(ident, lockType, locker)
                             : new LockStatus(ident, lockType, locker));

                   if (!ident.isTableIdentifier())
                   {
                      // any record-level share or exclusive lock must first obtain a share lock on
                      // the associated table
                      lockTable(ident.getTable(), LockType.SHARE, true);
                   }
…
}
```

#12 - 08/08/2013 10:53 AM - Vadim Nebogatov

Eric Faulhaber wrote:

> I think the desired effect can be achieved with changes to RecordLockContext and RecordBuffer (in setPinnedLockType and in the bulk delete methods). Those changes would track the pinned lock state of a table, or lock/unlock/downgrade a table lock, using an instance of TableIdentifier, at the same time the lock state data of an associated RecordIdentifier were changed.
>
> Make sense?

I think yes. Do we still keep FOR-EACH loop based RecordBuffer.deleteAll implementation?

#13 - 08/08/2013 03:59 PM - Vadim Nebogatov

Condition on the line RecordBuffer, line 7747

```
if (currentRecord == null || !sameRecord)
```

is always true, because above

```
if (currentRecord != null)
{
    ...
    if (sameRecord)
   {
       ...
    return;
    }
 }
```

#14 - 08/08/2013 04:21 PM - Eric Faulhaber

OK, but if you remove the conditional test, please be sure to add comments so we don't lose information. This conditional, while apparently unnecessary from a logical standpoint, is somewhat helpful in documenting the expected state of the buffer at the time we are transitioning the lock.

#15 - 08/08/2013 05:18 PM - Vadim Nebogatov

Eric, could you describe your idea about using table pinned lock for bulk delete in some more detail? I still not good understand it. setPinnedLockType works now on record level and saves pinned locks in RecordBuffer. Do you want to add adding table lock to RecordLockContext inside

setPinnedLockType?

#16 - 08/08/2013 06:11 PM - Eric Faulhaber

I've been thinking about this most of the afternoon here, and actually, after some discussions with Greg, I'm leaning back toward an approach that is much closer to what you originally proposed (sorry to steer you in a bad direction temporarily).

The LockManager is used both by converted code via RecordBuffer/RecordLockContext and by hand-written code that developers write after conversion. So, our change needs to be as simple to use as possible for the developer hand-writing Java after conversion. The RecordLockContext approach I was thinking about does not do this. However, it's probably good that you spent some time getting familiar with this class, since the compatibility layer is highly dependent upon it.

The bulk delete feature is going to be rarely used, and so it is OK if that path is a bit more complicated to use. Since the developer who is hand-writing code uses the LockManager directly, our core table-locking strategy has to be implemented here. Since converted code only uses the LockManager via the RecordLockContext class, we know that calls from converted code to lock/unlock/change record locks will only happen after RecordLockContext has decided it is appropriate. Thus, we can base our decisions as to when to acquire and release table-level locks accordingly.

The basic rules about table-level locks are:

1. a share lock on a table must be obtained by a context before the first time any record in that table is locked in that context (regardless of lock type);
2. the share lock on a table held by a context must be released when the last record in that table is released by that context;
3. an exclusive lock on a table is needed for bulk delete.

The first 2 items should be handled transparently when calls are made to LockManager.lock(LockType, RecordIdentifier, boolean). This can be done in InMemoryLockManager by adding a map to the Context inner class which maps table name to number of records currently locked *in that table*, and by updating that map every time a record lock is acquired or released. A table share lock is acquired when the count is 0 (on a record lock acquisition request) and is released when the count is 0 (on a record lock release request). The table lock must always be acquired before the record lock and must always be released after the record lock.

I would suggest moving the code from within the InMemoryLockManager.lock(LockType, RecordIdentifier, boolean) method into a new, private method, lockImpl(LockType, RecordIdentifier, boolean), and calling that worker method from the original lock method, which would now just be a wrapper which checks the Context's table lock counter map and calls out to the worker method accordingly:

```
if lock type is not NONE
   if this context has 0 record locks in the current table
      call lockTable to SHARE-lock the table

call lockImpl for normal record lock processing
increment or decrement record lock count for the current table in this context

if lock type is NONE
   if this context has 0 record locks in the current table
      call lockTable to release lock on the table
```

The LockManager.lockTable method would only be used by bulk delete. Its implementation in InMemoryLockManager would look something like this:

```
public LockType lockTable(LockType lockType, TableIdentifier ident, boolean update)
{
   store current lock type for this context
   call lockImpl(lockType, ident, update)
   return previous lock type (so that calling code knows what to set it back to after bulk delete)
}
```

Finally, bulk delete would look something like this:

```
get EXCLUSIVE table lock, remembering current lock level
try
   do bulk delete work
catch
   process exceptions
finally
   set table lock back to previous level
```

Please let me know if that makes sense, and if you have any questions.

#17 - 08/08/2013 06:14 PM - Eric Faulhaber

Vadim Nebogatov wrote:

> Do we still keep FOR-EACH loop based RecordBuffer.deleteAll implementation?

Yes, it is the simplest and most tested approach. If we find it causes performance issues, we can revisit the savepoint implementation.

#18 - 08/09/2013 11:15 AM - Vadim Nebogatov

Eric, thanks, I have no questions so far.

#19 - 08/14/2013 02:56 PM - Vadim Nebogatov

- File vmn_upd20130814a.zip added

- File vmn_upd20130814b.zip added

#20 - 08/14/2013 02:57 PM - Vadim Nebogatov

Update vmn_upd20130814a.zip is attached.

1) lockTable implemented
2) delete all corrected with lockTable
3) sort string is fixed in delete all. First index is selected so far because currently not possible recognize primary index in runtime.

Before commit I will split it to 2 different updates like Eric wrote.

Merged with latest revision 19376

#21 - 08/14/2013 05:25 PM - Vadim Nebogatov

- File deleted (vmn_upd20130814a.zip)

#22 - 08/14/2013 05:26 PM - Vadim Nebogatov

- File vmn_upd20130814a.zip added

#23 - 08/30/2013 11:15 AM - Eric Faulhaber

Code review 20130814a:

I think the implementation of table locking in InMemoryLockManager generally is OK, but some of the comments are confusing:

- The javadoc for the lock method states that the table lock and record lock acquisition/release operations are atomic for a given thread. I don't think that's right. A thread may be preempted or interrupted between these operations, preventing atomicity.
- There is an inline comment in lock that states: "No table lock check is required if update == false". This is not right. The update flag is false when business logic is checking the availability of a record lock, but will not actually lock it. The unavailability of a table lock will prevent acquisition of a record lock within that table, so a table lock check actually is necessary. However, the comment doesn't really match what the code is doing, which is just storing the existing table lock type in case we need to undo the lock in the event of an error. I think the code is correct, but the comment doesn't really match, which confused me.

Minor formatting point: please follow the coding standards with respect to putting extends and implements phrases on a separate line.

In RecordBuffer.deleteAll, instead of calling getPrimaryIndex, please stub out a method for this in MetaSchema and call that instead. Then remove RecordBuffer.getPrimaryIndex.

RemoteLockManager.lockTable needs an implementation. It should use the same idiom as all the other methods in that class, which simply delegate to LockManagerMultiplexer (which will need a corresponding method). Please also correct the format of the throws clause.

I don't think any of the modified classes have changed in bzr, but please confirm and merge with the latest tips if necessary. After you've addressed the above items, please regression test again.

#24 - 09/02/2013 04:36 AM - Vadim Nebogatov

I have executed regression two times.

Regression 20130831_172455, 4 FAILED tests:

1. gso221_session1 FAILED failure in step 5: 'timeout before the specific screen buffer became available'
2. gso221_session2 FAILED failure in step 1: 'Timeout while waiting for event semaphore to be posted.'
3. tc_dc_slot_029 FAILED failure in step 8: 'timeout before the specific screen buffer became available (Mismatched data at line 4, column 9. Expected '' (0x0000 at relative Y 4, relative X 9) and found '1' (0x0031 at relative Y 4, relative X 9).)'
4. tc_job_002 FAILED failure in step 40: 'Unexpected EOF in actual at page # 1.'

Regression 20130901_143336, 6 FAILED tests:

1. gso_17 FAILED Mismatched data at absolute row 13, relative row -1, page # -1, page size -1, last page false, column index 3. Expected '3' (0x33) and found '1' (0x31).
2. gso_238 FAILED failure in step 4: 'timeout before the specific screen buffer became available (Mismatched data at line 19, column 8. Expected '(' (0x0028 at relative Y 19, relative X 8) and found 'I' (0x0049 at relative Y 19, relative X 8).)'
3. tc_dc_slot_024 FAILED failure in step 8: 'timeout before the specific screen buffer became available (Mismatched data at line 4, column 9. Expected '' (0x0000 at relative Y 4, relative X 9) and found '1' (0x0031 at relative Y 4, relative X 9).)'
4. tc_dc_slot_025 FAILED failure in step 8: 'timeout before the specific screen buffer became available (Mismatched data at line 4, column 9. Expected '' (0x0000 at relative Y 4, relative X 9) and found '1' (0x0031 at relative Y 4, relative X 9).)'
5. tc_job_002 FAILED failure in step 40: 'Unexpected EOF in actual at page # 1.'
6. tc_job_clock_004 FAILED failure in step 3: 'Timeout while waiting for event semaphore to be posted.'

Because of gso_38, I have started to repeat regression one more time.

#25 - 09/02/2013 04:54 PM - Vadim Nebogatov

Regression 20130902_061345, 3 FAILED tests:

1. tc_job_002 FAILED failure in step 40: 'Unexpected EOF in actual at page # 1.'
2. tc_job_clock_002 FAILED failure in step 20: 'timeout before the specific screen buffer became available (Mismatched data at line 5, column 18. Expected '' (0x0000 at relative Y 5, relative X 18) and found ' ┌' (0x250C at relative Y 5, relative X 18).)'
3. tc_job_clock_005 FAILED failure in step 23: 'timeout before the specific screen buffer became available (Mismatched data at line 5, column 18. Expected '' (0x0000 at relative Y 5, relative X 18) and found ' ┌' (0x250C at relative Y 5, relative X 18).)'

So, regression seems passed successfully.

#26 - 09/02/2013 05:41 PM - Vadim Nebogatov

- File vmn_upd20130831a.zip added

I have uploaded update vmn_upd20130831a.zip replacing vmn_upd20130814a.zip with fixes according to note 23.

#27 - 09/02/2013 05:43 PM - Vadim Nebogatov

Regressions (note 24-25) were executed for vmn_upd20130831a.zip.

#28 - 09/03/2013 11:37 AM - Eric Faulhaber

The changes look good. Please add a header entry to LockManagerMultiplexer.java. As MetaSchemaImpl is a temporary stub class, please put a TODO into RecordBuffer.deleteAll, noting that the code which calls it is temporary and needs to be replaced when a true MetaSchema implementation is available.

BTW, to make your life easier: you don't need to change the date in the header of a file between updates, if nothing else has changed in the file (e.g., RecordIdentifier).

#29 - 09/03/2013 11:48 AM - Eric Faulhaber

Since my requested changes do not affect the code itself, no need for another round of testing. Assuming it still compiles after the changes, please commit to bzr and distribute.

#30 - 09/03/2013 12:29 PM - Eric Faulhaber

- Status changed from New to WIP

- % Done changed from 0 to 90

#31 - 09/03/2013 04:11 PM - Vadim Nebogatov

Eric, should I commit also MetaSchemaImpl?
Or simply comment current MetaSchemaImpl method call and write P2JIndex primaryIndex = null for compile?

#32 - 09/03/2013 04:13 PM - Eric Faulhaber

Please don't commit MetaSchemaImpl and comment out its usage for now, adding a TODO to fix it later.

#33 - 09/03/2013 05:06 PM - Vadim Nebogatov

- File vmn_upd20130903a.zip added

Update vmn_upd20130903a.zip committed to bzr revision 10379.

#34 - 11/15/2013 12:16 PM - Eric Faulhaber

We have decided not to move forward with the MetaSchema interface, and instead to use Java annotations in the DMO implementation classes to

meet our runtime DMO and legacy table metadata needs.

So, we will have to address the last remaining TODO of this task -- i.e., determining the legacy, primary index for a DMO -- using this approach. For issue #2120, Constantin already has introduced some index-related annotations. See the LegacyIndex and LegacyIndexes interfaces in the persist package, and look at any converted DMO implementation class to see how they have been added.

I think the best way to add the primary index information is to enhance the LegacyIndexes annotation type by adding an annotation type element declaration for primary(). You will need to emit the associated primary = "<index name>" during DMO conversion, then enhance IndexHelper to give you the information you need about that primary index for your purposes in RecordBuffer.deleteAll.

#35 - 11/15/2013 12:19 PM - Eric Faulhaber

Note that we will not need the primary annotation element for temp-tables (at least not for bulk delete purposes), but I am not familiar enough with annotations to know whether we can leave this out of the temp-table DMO implementation classes or not. In general, I want to add as little clutter to the DMOs as possible, so if we can leave it out, let's do so. If not, so be it.

#36 - 11/20/2013 10:11 AM - Vadim Nebogatov

I am trying to add optional annotation “primary” and generate DMO impl annotation like

```
@LegacyTable(name = "vehicle")
@LegacyIndexes(
   primary = true,
   value ={
      @LegacyIndex(name = "pi-make-model", converted = "vehicle_pi_make_model"),
      @LegacyIndex(name = "si-vin", converted = "vehicle_si_vin")
   }
)
```

with corresponding changes in LegacyIndexes:

```
public @interface LegacyIndexes
{
   /** The list of {@link LegacyIndex legacy table indexes.} */
   LegacyIndex[] value();

   boolean primary() default false;
}
```

For this I tried to correct “legacy_indexes_annotation” template

```
   <!-- add legacy indexes annotations with primary-->
   <ast-root name="legacy_indexes_annotation" terse="true" ast_class="com.goldencode.p2j.uast.JavaAst">
      <ast id="0" type="ANNOTATION" text="LegacyIndexes">
         <ast id="0" type="INITIALIZER"/>
         <ast id="0" type="ASSIGN" text="=">
            <ast id="0" type="REFERENCE" text="primary" />
            <ast id="0" type="STRING" text="$(primary_name)" />
         </ast>
      </ast>
   </ast-root>
```

But annotation is generated wrongly:

```
@LegacyIndexes(
```

```
{
   @LegacyIndex(name = "pi-make-model", converted = "vehicle_pi_make_model"),
   @LegacyIndex(name = "si-vin", converted = "vehicle_si_vin")
}primary = "index_name")
```

As I see, problem is in brew.xml code which seems works correctly only with one property in annotation container (LegacyIndexes in our case).

#37 - 11/20/2013 10:23 AM - Constantin Asofiei

Vadim/Eric: The LegacyIndexes annotation is used only because Java 7 does not allow multiple annotations with the same name. Java 8 will allow this, and at a certain point I would like to get rid of the LegacyIndexes parent annotation; the code will look cleaner after this.

The primary mark should be at the specific LegacyIndex annotation to which it belongs; I don't see a benefit in specifying the primary index name. Just add primary = true at the LegacyIndex annotation and at runtime, when legacy indexes are resolved, throw an IllegalStateException if multiple indexes are marked primary (and save the primary index name somewhere if you need to look it up later).

More, I think for temp-tables the primary (and other info like this) is emitted to the dmo_index.xml file too; if this is the case, that should be removed, else we will have the same information in two places.

#38 - 11/20/2013 10:33 AM - Vadim Nebogatov

Actually I also started with adding boolean primary to LegacyIndex, but noticed that Eric suggested to add index name to LegacyIndexes, so I meant it is more convenient in usages.

#39 - 11/20/2013 10:35 AM - Vadim Nebogatov

...in order to find primary index without search in array for example.

#40 - 11/20/2013 10:40 AM - Constantin Asofiei

Vadim Nebogatov wrote:

> ...in order to find primary index without search in array for example.

Yes, but you can always resolve this when the legacy indexes are initially resolved and cache it somewhere... so you don't have to search in the array each time you need to look it up. Also, in case the user adds a hand-written DMO and somehow ends up with two indexes marked as primary, you will need to catch this and throw an exception.

#41 - 11/20/2013 10:50 AM - Eric Faulhaber

I am OK with Constantin's approach, especially if we are going to get rid of the @LegacyIndexes parent with Java 8.

#42 - 11/20/2013 11:04 AM - Eric Faulhaber

Constantin Asofiei wrote:

> More, I think for temp-tables the primary (and other info like this) is emitted to the dmo_index.xml file too; if this is the case, that should be removed, else we will have the same information in two places.

This is a bigger problem/task, as it affects more than just this one piece of information. I have added #2203 for this, but it is not a priority at the moment.

#43 - 11/25/2013 01:53 PM - Vadim Nebogatov

- File vmn_upd20131125b.zip added

- File vmn_upd20131125a.zip added

Update vmn_upd20131125a.zip is attached.

1) Using java annotations for retrieving primary index and deleteAll().
2) NPE fixed, initialize() moved in deleteAll()

Attached vmn_upd20131125b.zip contains tests for permanent and temporary tables.

Merged with latest revision 10414.

#44 - 11/26/2013 03:35 PM - Vadim Nebogatov

- File vmn_upd20131126a.zip added

- File vmn_upd20131126b.zip added

Update vmn_upd20131126a.zip is attached.

Fixed P2JIndex.getOrderByClause(), used dmoAlias instead of table name.
Attached vmn_upd20131126b.zip contains corrected tests for permanent and temporary tables.

Merged with latest revision 10416.

#45 - 11/27/2013 06:10 PM - Eric Faulhaber

Code review 20131126a:

- The new legacy_primary_index_annotation in java_templates.tpl is largely redundant with legacy_index_annotation. The same could be accomplished by adding a peernode (TRPL) annotation instead of the LABEL AST, under the ANNOTATION AST. Then, in dmo_common.rules, conditionally add a peer LABEL node at that location only if the index represents the primary index. See how other templates containing the peernode annotation are used in rules as an example.
- In IndexHelper.getPrimaryIndex, the primaryIndexByEntity map will never contain any entries, because nothing is ever put into it. Please also remove the blank line between the javadoc comment and the method.

- Also regarding IndexHelper.getPrimaryIndex, I'm pretty sure this is the case, but have you confirmed that it is impossible in Progress to have a permanent table with no primary index defined? We are throwing PersistenceException in this case, which raises an error condition in RecordBuffer.deleteAll.

#46 - 11/28/2013 08:00 AM - Vadim Nebogatov

Eric Faulhaber wrote:

> - In IndexHelper.getPrimaryIndex, the primaryIndexByEntity map will never contain any entries, because nothing is ever put into it. Please also remove the blank line between the javadoc comment and the method.

Thanks, fixed.

#47 - 11/28/2013 08:43 AM - Vadim Nebogatov

Eric Faulhaber wrote:

> - Also regarding IndexHelper.getPrimaryIndex, I'm pretty sure this is the case, but have you confirmed that it is impossible in Progress to have a permanent table with no primary index defined? We are throwing PersistenceException in this case, which raises an error condition in RecordBuffer.deleteAll.

It seems it is true. I tried to remove primary index and system did not allow it and asked to reassign primary index to some another one. Not primary index is deleted with no problems. Also I have read:

ltu.pdf

```
The primary index is the one that Progress uses by default. Each table has
one and only one primary index. Progress uses the primary index when
retrieving records or ordering records for a list (like a report) if you don’t
 specify another index. You want your primary index to reflect the most
common or natural sort order.
```

ABL_triggers_and_indexes.pdf

```
You can change the name of an index at any time. You can also delete nonprimary
indexes. However, before letting you delete a primary index, OpenEdge requires that
you first designate another index as primary.
If there is only one index, you must create a new index before deleting the existing
index. You cannot change any of the component definitions of an index. Instead, you
must delete the index and re-create it using the modified component definitions.
Remember that OpenEdge assumes that the first index you create is the primary index,
so create your primary index first.
```

http://www.scribd.com/doc/31065006/Progress-Database-Design-Guide

```
Every table should have at least one index, the primary index. When you create the first indexon any table in
a 4GL implementation, Progress assumes it is the primary index and sets thePrimary flag accordingly.
```

#48 - 11/28/2013 09:45 AM - Vadim Nebogatov

But I found out on testing one strange fact. Data Dictionary allows to create permanent table with no fields, default primary index is also generated in this case with no fields in index. And CREATE and DISPLAY commands work with such table...

#49 - 11/29/2013 02:37 PM - Vadim Nebogatov

- File vmn_upd20131129a.zip added

- File vmn_upd20131129b.zip added

Update vmn_upd20131129a.zip/vmn_upd20131129b.zip attached.

Additionally to previous update vmn_upd20131126a.zip:

1. peerNode used for primary index annotation;
2. Fixed primaryIndexByEntity map issue in IndexHelper.getPrimaryIndex;
3. Added support for custom conversion of fields with extents when hinted names amount is not equal to original extent (#2134, note 27). Tested different situations (equal/less/more, incremented name already exists if less and so on);
4. Using ELEM_EXTENT_NAME instead of ELEM_EXTENT_NAMES for custom conversion of fields with extents (#2134, note 27) - updated hints examples are attached in vmn_upd20131129b.zip.

Merged with latest revision 10417.

#50 - 12/05/2013 03:33 PM - Eric Faulhaber

Code review 20131129a/b:

The changes look good. I am expecting an update for #1654 to be checked in today. Once that is done, please merge with it and conduct a (hopefully final) round of regression testing, then commit and distribute. I don't need to review the merged update again, unless you find some significantly problematic conflict in the merge.

#51 - 12/06/2013 02:45 PM - Vadim Nebogatov

- File vmn_upd20131206a.zip added

Update vmn_upd20131206a.zip attached as result of merge vmn_upd20131129a.zip with revision 10418 (corresponding to committed #1654). Only one text conflict was found in comments.

Regression started.

#52 - 12/12/2013 02:31 PM - Vadim Nebogatov

- File vmn_upd20131212a.zip added

Merged with latest revision 10419.

#53 - 12/13/2013 07:59 AM - Vadim Nebogatov

20131212_212328 regression results:

FAILED tc_job_002
FAILED (only timeouts) tc_dc_slot_014, tc_dc_slot_016,tc_dc_slot_017, tc_dc_slot_018, tc_dc_slot_024, tc_dc_slot_025, tc_dc_slot_026, tc_dc_slot_027, tc_dc_slot_028, tc_job_clock_004, tc_dc_slot_023
FAILED_DEPENDENCY - tc_dc_slot_019, tc_dc_slot_020

Regression started second time.

#54 - 12/13/2013 03:15 PM - Vadim Nebogatov

Second 20131213_113947 regression results:

FAILED tc_job_002
FAILED tc_job_clock_002 failure in step 20: 'timeout before the specific screen buffer became available (Mismatched data at line 5, column 18. Expected '' (0x0000 at relative Y 5, relative X 18) and found ' ┌' (0x250C at relative Y 5, relative X 18).)'

Regression passed.

#55 - 01/07/2014 12:42 PM - Eric Faulhaber

- % Done changed from 90 to 100

- Status changed from WIP to Closed

#56 - 11/16/2016 11:42 AM - Greg Shah

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

Files

vmn_upd20130814b.zip	479 Bytes	08/14/2013	Vadim Nebogatov
vmn_upd20130814a.zip	148 KB	08/14/2013	Vadim Nebogatov
vmn_upd20130831a.zip	154 KB	09/02/2013	Vadim Nebogatov
vmn_upd20130903a.zip	153 KB	09/03/2013	Vadim Nebogatov
vmn_upd20131125a.zip	118 KB	11/25/2013	Vadim Nebogatov
vmn_upd20131125b.zip	711 Bytes	11/25/2013	Vadim Nebogatov
vmn_upd20131126a.zip	118 KB	11/26/2013	Vadim Nebogatov
vmn_upd20131126b.zip	716 Bytes	11/26/2013	Vadim Nebogatov
vmn_upd20131129a.zip	134 KB	11/29/2013	Vadim Nebogatov
vmn_upd20131129b.zip	443 Bytes	11/29/2013	Vadim Nebogatov
vmn_upd20131206a.zip	135 KB	12/06/2013	Vadim Nebogatov
vmn_upd20131212a.zip	136 KB	12/12/2013	Vadim Nebogatov