Database - Feature #2220

dynamic query methods which require implementation

01/07/2014 12:27 PM - Eric Faulhaber

Status:	Closed	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Stanislav Lomany	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:	Cleanup and Stablization for Server Features		
billable:	No	**vendor_id:**	GCD

Description

History

#1 - 01/07/2014 12:27 PM - Eric Faulhaber

The following need implementation in QueryWrapper:

```
public character indexInformation()
public character indexInformation(int n)
public character indexInformation(NumberType n)
public logical createResultListEntry()
public logical deleteResultListEntry()
```

#2 - 03/30/2014 02:51 PM - Eric Faulhaber

- Assignee set to Stanislav Lomany

#3 - 04/18/2014 09:15 AM - Stanislav Lomany

- Status changed from New to WIP

Eric, some issues about indexInformation:
1. There may be some differences in index selection rules with 4GL, e.g.

```
def temp-table tt
            field f1 as char
            field f2 as char
            index idx1 f1
            index idx2 f2.

def temp-table tt2
            field f21 as char
            field f22 as char
            index idx21 f21
            index idx22 f22.

...
qh:query-prepare("for each tt, each tt2 by f2 by f22").
```

P2J picks indexes idx2 and idx22. 4GL picks idx2 and idx21.

2. We do not support multiple index selection case, while this function supports it. Should we handle this case?
3. For static queries I see three ways of solution:
- analyze where/sort clause at runtime and select an index using documented index selection criteria. Cons: code duplication with index_selection.rules.
- analyze where/sort clause, build, basically, original 4GL AST and pass it to index_selection.rules. Cons: seems complicated.
- my favorite: select indexes at conversion stage and, say, pass to query constructors. Cons: less pretty code.

Also you should take into consideration that this function provides 1. debug information 2. which usefulness is questionable since the tables are SQL-mapped.

#4 - 04/18/2014 11:21 AM - Constantin Asofiei

Stanislav, if I'm putting your example into a OPEN QUERY case, the code can not compile in 4GL - it says that f2 can be resolved to multiple fields in tt2. I guess this is because it assumes f2 is an abreviation for either f21 or f22. Please test your scenario by either disambigating each field used in the query-prepare clause or renaming the fields so there are no abreviation collisions. Also, I think you should check how OPEN QUERY behaves in this case, too - you will need an associated DEFINE QUERY for INDEX-INFORMATION to work.

#5 - 04/18/2014 11:48 AM - Stanislav Lomany

Constantin, yes, for static query it for some reason requires explicit fields specifications while works fine for dynamic query. Result is the same: 4GL picks idx21.

#6 - 04/19/2014 01:38 PM - Stanislav Lomany

Guys, about deleteResultListEntry - how it should be implemented? My idea is to make query-specific list of recids that should be skipped on query iteration.
createResultListEntry doesn't present in the customer project. Should I implement it?

#7 - 04/23/2014 02:48 PM - Stanislav Lomany

About deleteResultListEntry - it can be applied to
1. queries with preselected result set
2. scrolling queries or queries linked to browse.
While in preselected queries a deleted entry can be referenced thru record ids, for adaptive queries its not the case - we can meet this record later in the result set. As it seems to me, for adaptive queries progress has some preselected cache of ids, from which it removes an entry, so it's not clear how should I reference a deleted entry and it which cases we will be able to meet it again. I'll look deeper into that, but considering that we do not completely reproduce adaptive behavior, that can be a problem to implement it accurately. Do you have any ideas about a simple solution?

#8 - 04/23/2014 03:09 PM - Eric Faulhaber

Stanislav Lomany wrote:

> Eric, some issues about indexInformation:
> 3. For static queries I see three ways of solution:
> - analyze where/sort clause at runtime and select an index using documented index selection criteria. Cons: code duplication with index_selection.rules.
> - analyze where/sort clause, build, basically, original 4GL AST and pass it to index_selection.rules. Cons: seems complicated.
> - my favorite: select indexes at conversion stage and, say, pass to query constructors. Cons: less pretty code.

I would like to avoid passing additional information into the constructor for the index in use, because we essentially already do this in the form of the sort clause. For every table/buffer involved in a static query, we have chosen an index to use during conversion, based on the index selection rules we have determined. This is represented in converted code as the sort clause applied to that query (or query component). We can (in fact, we already do, for FindQuery/RandomAccessQuery) map this clause back to an index (see the IndexHelper class). The missing piece is to map the selected index back to its legacy name, which we now can do based on the index annotations we have in the DMO classes. Ideally, we would only do this work on demand, in response to an indexInformation call, and cache it, rather than for every query.

> Also you should take into consideration that this function provides 1. debug information 2. which usefulness is questionable since the tables are SQL-mapped.

This is true, but we can't know what decisions any given application will make based on the output of this attribute.

#9 - 04/23/2014 03:32 PM - Eric Faulhaber

Stanislav Lomany wrote:

> createResultListEntry doesn't present in the customer project. Should I implement it?

No. I cannot find a reference to it in the GUI project, either.

#10 - 04/23/2014 03:59 PM - Eric Faulhaber

Stanislav Lomany wrote:

> About deleteResultListEntry - it can be applied to
> 1. queries with preselected result set
> 2. scrolling queries or queries linked to browse.
> While in preselected queries a deleted entry can be referenced thru record ids, for adaptive queries its not the case - we can meet this record later in the result set. As it seems to me, for adaptive queries progress has some preselected cache of ids, from which it removes an entry, so it's not clear how should I reference a deleted entry and it which cases we will be able to meet it again. I'll look deeper into that, but considering that we do not completely reproduce adaptive behavior, that can be a problem to implement it accurately. Do you have any ideas about a simple solution?

First, please explain what you mean by the adaptive behavior we are not completely reproducing. I guess you are referring to OPEN QUERY FOR ... cases (as opposed to OPEN QUERY PRESELECT ...), but we have taken great lengths to reproduce the same, dynamic behavior of Progress in this regard (in terms of changes to an index by the current session or by a separate session which trigger changes to the results an active query sees).

What behavior are you seeing in Progress that we are not performing, such that it is causing concern about how to implement this method?

#11 - 04/23/2014 06:28 PM - Stanislav Lomany

> I would like to avoid passing additional information into the constructor for the index in use, because we essentially already do this in the form of the sort clause. For every table/buffer involved in a static query, we have chosen an index to use during conversion, based on the index selection rules we have determined. This is represented in converted code as the sort clause applied to that query (or query component). We can (in fact, we already do, for FindQuery/RandomAccessQuery) map this clause back to an index (see the IndexHelper class).

Eric, I thought about it, but it seems unreliable to me. E.g.:

```
def temp-table tt
     field f1 as integer
     field f2 as integer
     field f3 as integer
     index idx1 f1
     index idx2 f2.

def query q for tt.
open query q for each tt where f1 > 0 by f2.

message query q:handle:index-information.
```

Selected index is idx1 and it cannot be restored using sort clause only.

#12 - 04/23/2014 06:56 PM - Eric Faulhaber

Yes, I see. The conversion seems correct, but the sort clause is lossy in this case, with respect to the selected index.

Go with your favored approach, then. Note that you will need to change a lot of constructors. Since this will be a mandatory parameter, you will want to put it near the front of the list of parameters, since there are so many variants when all the optional parameters are taken into account. Are there any concrete implementations of P2JQuery for which this parameter will not apply (i.e., with which indexInformation cannot be used)?

#13 - 04/24/2014 06:53 AM - Stanislav Lomany

> Are there any concrete implementations of P2JQuery for which this parameter will not apply (i.e., with which indexInformation cannot be used)?

AFAIK, no.

One more issue - if the index do not have bracketing, the first entry in the list is the CHARACTER string “WHOLE-INDEX,” and the second entry in the list is name of the index. E.g.:

```
def temp-table tt
     field f1 as integer
     field f2 as integer
     field f3 as integer
     index idx1 f1
     index idx2 f2.

def query q for tt.
open query q for each tt.

message query q:handle:index-information.
```

Output: WHOLE-INDEX,idx1. In this case I think I'll pass this string to c'tor.
If there are no indexes then output is WHOLE-INDEX,default. I think that is a rare case and I'll pass this string to c'tor too.

#14 - 04/24/2014 08:22 AM - Constantin Asofiei

Stanislav Lomany wrote:

> Are there any concrete implementations of P2JQuery for which this parameter will not apply (i.e., with which indexInformation cannot be used)?

> AFAIK, no.

I think the 4GL query blocks (i.e. FOR EACH) do not allow access to INDEX-INFORMATION - currently, P2JQuery is the root class for all legacy-style queries (including blocks, FIND, etc). In #2120 (notes 71/72) there was some talk about enhancing the query infrastructure so that the functionalities accessed only via query resources are removed from P2JQuery and brought into another interface - we should have a task for this, if it's still in play.

#15 - 04/24/2014 08:25 AM - Stanislav Lomany

Right. I thought we are talking about only about queries which can be used in OPEN QUERY and forgot that P2JQuery has a broader scope.

#16 - 04/24/2014 02:35 PM - Stanislav Lomany

> I guess you are referring to OPEN QUERY FOR ... cases (as opposed to OPEN QUERY PRESELECT ...),

By preselect queries I mean ones with use preselected result set rather than index walking (i.e. OPEN QUERY PRESELECT + subset of OPEN QUERY FOR).

> What behavior are you seeing in Progress that we are not performing, such that it is causing concern about how to implement this method?

```
def buffer xbook for book.

do transaction:
   for each book:
     delete book.
   end.

  create book. book.book-id = 1. book.isbn = "isbn1".
  create book. book.book-id = 2. book.isbn = "isbn2".
  create book. book.book-id = 3. book.isbn = "isbn3".
  create book. book.book-id = 5. book.isbn = "isbn5".
  create book. book.book-id = 6. book.isbn = "isbn6".
end.

def query q for book scrolling.

open query q for each book.
def var i as integer init 1.
repeat:
  get next q no-lock.
  if not avail(book) then leave.

  if i = 2 then do transaction:
     find xbook where xbook.book-id = 1.
     xbook.book-id = 4.
  end.

  i = i + 1.
end.

do transaction:
   find xbook where xbook.book-id = 4.
   xbook.book-id = 1.
end.

get last q no-lock.
repeat:
  message string(book.book-id) + " " + string(book.isbn).

  get prev q no-lock.
  if not avail(book) then leave.
end.
```

4GL output:

```
6 isbn6
5 isbn5
1 isbn1
3 isbn3
2 isbn2
1 isbn1
```

P2J output:

```
6 isbn6
5 isbn5
3 isbn3
2 isbn2
1 isbn1
```

#17 - 05/20/2014 09:45 AM - Eric Faulhaber

Do the results differ if the query is not set to be scrolling?

#18 - 05/20/2014 01:04 PM - Stanislav Lomany

Yes, they differ and match P2J.

#19 - 06/04/2014 06:34 PM - Stanislav Lomany

- File svl_upd20140604a.zip added

Update for review with indexInformation implemented.

#20 - 06/24/2014 06:45 PM - Eric Faulhaber

I am reviewing the 0604a update now (sorry it took so long for me to get to this). Could you please post some test case output, so I can see what the effect on the code is?

#21 - 06/24/2014 07:13 PM - Eric Faulhaber

Code review 0604a:

Looks good, though I would still like to see sample output. Note there is another version of IndexHelper in play right now (#2186, note 50). Please merge with that version (it just has a method removed), then regression test. Please re-convert the customer server project as well, and spot check the converted source diffs there.

#22 - 06/25/2014 11:34 AM - Stanislav Lomany

Testcase:

```
def temp-table tt
     field f1 as integer
     field f2 as integer
     field f3 as integer
     index idx1 f1
     index idx2 f2.

def temp-table tt2
     field f4 as integer
     field f5 as integer
     field f6 as integer
     index idx4 f4
     index idx5 f5.

def var i as integer init 0.
def var b as logical.
```

```
def query q for tt.
def query q2 for tt, tt2.
open query q for each tt where f1 > 0 by f2.
message "1: " + query q:handle:index-information.

open query q for each tt by f2.
message "2: " + query q:handle:index-information.

open query q preselect each tt where f1 > 0.
message "3: " + query q:handle:index-information.

open query q2 preselect each tt where tt.f1 > 0, each tt2.
message "4: " + query q2:handle:index-information(1) + " (2) " + query q2:handle:index-information(2).

open query q2 preselect each tt where tt.f1 > 0, each tt2 where tt2.f4 > i.
message "5: " + query q2:handle:index-information(1) + " (2) " + query q2:handle:index-information(2).

open query q preselect each tt where f1 > 0 by if b then f2 else f2.
message "6: " + query q:handle:index-information.

open query q2 for each tt where tt.f1 > 0, each tt2 where tt2.f4 > 0.
message "7: " + query q2:handle:index-information(1) + " (2) " + query q2:handle:index-information(2).

open query q2 for each tt where tt.f1 > 0, each tt2 where tt2.f4 > 0 by if b then f2 else f2.
message "8: " + query q2:handle:index-information(1) + " (2) " + query q2:handle:index-information(2).

open query q2 for each tt where tt.f1 > 0, first tt2 where tt2.f4 > 0 by if b then f2 else f2.
message "9: " + query q2:handle:index-information(1) + " (2) " + query q2:handle:index-information(2).
```

Converted code:

```
query0.assign(new PreselectQuery(tt, "tt.f1 > 0", null, "tt.f2 asc, tt.f1 asc", "idx1"));
query0.open();
message(concat(new character("1: "), query0.indexInformation()));
query0.assign(new AdaptiveQuery(tt, (String) null, null, "tt.f2 asc", "WHOLE-INDEX,idx2"));
query0.open();
message(concat(new character("2: "), query0.indexInformation()));
query0.assign(new PreselectQuery(tt, "tt.f1 > 0", null, "tt.f1 asc", "idx1"));
query0.open();
message(concat(new character("3: "), query0.indexInformation()));
query1.assign(new PreselectQuery("tt.f1 asc, tt2.f4 asc"));
query1.addComponent(tt, "tt.f1 > 0", "idx1");
query1.addComponent(tt2, ((String) null), "WHOLE-INDEX,idx4");
query1.open();
message(concat("4: ", query1.indexInformation(1), " (2) ", query1.indexInformation(2)));
query1.assign(new PreselectQuery("tt.f1 asc, tt2.f4 asc"));
query1.addComponent(tt, "tt.f1 > 0", "idx1");
query1.addComponent(tt2, "tt2.f4 > ?", "idx4", new Object[]
{
    i
});
query1.open();
message(concat("5: ", query1.indexInformation(1), " (2) ", query1.indexInformation(2)));
query0.assign(new PresortQuery(tt, "tt.f1 > 0", null, "tt.f1 asc", "idx1"));
query0.addSortCriterion(byExpr0);
query0.open();
message(concat(new character("6: "), query0.indexInformation()));
query1.assign(new CompoundQuery(true, false));
query1.addComponent(new AdaptiveQuery(tt, "tt.f1 > 0", null, "tt.f1 asc", "idx1"));
query1.addComponent(new AdaptiveQuery(tt2, "tt2.f4 > 0", null, "tt2.f4 asc", "idx4"));
query1.open();
message(concat("7: ", query1.indexInformation(1), " (2) ", query1.indexInformation(2)));
query1.assign(new PresortQuery());
query1.addComponent(tt, "tt.f1 > 0", "idx1");
query1.addComponent(tt2, "tt2.f4 > 0", "idx4");
query1.addSortCriterion(byExpr0);
query1.open();
message(concat("8: ", query1.indexInformation(1), " (2) ", query1.indexInformation(2)));
query1.assign(new PresortCompoundQuery());
query1.addComponent(new PreselectQuery(tt, "tt.f1 > 0", null, "tt.f1 asc", "idx1"));
query1.addComponent(new RandomAccessQuery(tt2, "tt2.f4 > 0", null, "tt2.f4 asc", "idx4"), QueryConstants.FIRST
```

```
);
query1.addSortCriterion(byExpr0);
query1.open();
message(concat("9: ", query1.indexInformation(1), " (2) ", query1.indexInformation(2)));
```

#23 - 06/25/2014 11:58 AM - Stanislav Lomany

> Looks good, though I would still like to see sample output. Note there is another version of IndexHelper in play right now (#2186, note 50). Please merge with that version (it just has a method removed), then regression test.

OK.

> Please re-convert the customer's server project as well, and spot check the converted source diffs there.

I've done that before posting the update.

#24 - 06/25/2014 12:35 PM - Eric Faulhaber

Sample looks good, thanks.

#25 - 07/31/2014 08:15 AM - Stanislav Lomany

Regarding the question if scrolling adaptive queries are modified with DELETE-RESULT-LIST-ENTRY in the customer app: I cannot say definitely because the code is complicated, but it seem to be the case. So it looks like we have to properly implement adaptive scrolling queries.

#26 - 07/31/2014 08:49 AM - Stanislav Lomany

About caching. So far I can state that:
1. Adaptive scrolling queries have recid-based cache which is populated as we iterate records.
2. Once populated:
  2.1 records do not change their position if indexed fields are updated;
  2.2 deleted records are not displayed;
  2.3 new records are not added.
3. DELETE-RESULT-LIST-ENTRY deletes only one entry even if then same record is represented by several entries in the result list.

#27 - 08/01/2014 10:12 AM - Stanislav Lomany

BTW, there is an issue with dynamic queries you probably aware of. Creation of a record using a backing buffer of a dynamic query may lead to

incorrect results.

```
def temp-table tt field f1 as integer
                  index idx1 f1.

def query q for tt.
def var i as integer.

create tt. tt.f1 = 1.
create tt. tt.f1 = 2.
create tt. tt.f1 = 3.

open query q for each tt.
get next q.

create tt. tt.f1 = 4.

get next q.
message string(tt.f1).
```

Correct output: 2. P2J output: 1.
It doesn't affect my current work and I'm not aware if it affects the customer app.

#28 - 08/01/2014 02:51 PM - Stanislav Lomany

Strange case. Testcase 1:

```
def temp-table tt field f1 as integer
                  index idx1 f1.
define buffer xtt for tt.

def query q for tt.
def var i as integer.

create tt. tt.f1 = 1.
create tt. tt.f1 = 2.
create tt. tt.f1 = 3.

open query q for each tt.
create xtt. xtt.f1 = 0.

repeat:
  get prev q.
   if query-off-end("q") then do:
      message "N/A". leave.
   end.
   else message string(tt.f1).
end.
```

4GL output: 3 2 1 N/A

Testcase 2:

```
def temp-table tt field f1 as integer
                  index idx1 f1.
define buffer xtt for tt.

def query q for tt.
def var i as integer.

create tt. tt.f1 = 1.
create tt. tt.f1 = 2.
create tt. tt.f1 = 3.

open query q for each tt.

def var created as logical init false.

repeat:
   if not created then do:
      create xtt. xtt.f1 = 0.
      created = true.
   end.

   get prev q.
   if query-off-end("q") then do:
      message "N/A". leave.
   end.
   else message string(tt.f1).
end.
```

4GL output: 3 2 1 0 N/A
Looks like 4GL bug. There is no difference in testcases output if the queries are iterated in the forward direction (i.e. get next q is used).

#29 - 08/01/2014 03:06 PM - Eric Faulhaber

I think what's happening here is that nothing is releasing/flushing the newly created record from buffer xtt before or during the repeat loop in the first

case, so it doesn't appear in the query results. I believe if you assign a non-default value to the indexed field f1 (say, -1), or change the default value for f1 to something non-0, that would trigger a flush on the assignment. Also, an explicit release or validate statement after the assignment should do it.

#30 - 08/01/2014 03:08 PM - Eric Faulhaber

Stanislav Lomany wrote:

> There is no difference in testcases output if the queries are iterated in the forward direction (i.e. get next q is used).

Are you saying they both include the 0 record or they both exclude it?

#31 - 08/01/2014 03:19 PM - Stanislav Lomany

You are right - it is about flushing.

> Are you saying they both include the 0 record or they both exclude it?

Include. But in the NEXT-case I also has value 4 assigned instead of 0. That's why it worked.

BTW, P2J includes records in both PREV cases.

#32 - 08/01/2014 03:24 PM - Eric Faulhaber

Stanislav Lomany wrote:

> BTW, P2J includes records in both PREV cases.

I am working on a flush/validate fix that I hope will make this more consistent with Progress. Is not having this now getting in your way? I don't want us both working on the same problem.

#33 - 08/01/2014 03:55 PM - Stanislav Lomany

No problem about having this issue. I just need to consider it when making testcases.

#34 - 08/01/2014 04:44 PM - Stanislav Lomany

This issue I have to fix. LAST just do not work for adaptive queries.

```
def temp-table tt field f1 as integer
                  index idx1 f1.
define buffer xtt for tt.

def query q for tt.
def var i as integer.

create tt. tt.f1 = 1.
create tt. tt.f1 = 2.
create tt. tt.f1 = 3.

open query q for each tt.
get last q.

message string(query-off-end("q")).
message string(avail(tt)).
```

P2J output: no no.

#35 - 08/01/2014 04:51 PM - Eric Faulhaber

OK, but that doesn't look like a flushing issue to me, so I guess we will stay out of each others' way on this one.

#36 - 08/04/2014 08:55 AM - Stanislav Lomany

While working on the LAST issue, I've looked into invalidation logic for adaptive queries. I think it's a bit doubtful, e.g.:

```
def temp-table tt field f1 as integer
                  index idx1 f1.
define buffer xtt for tt.

def query q for tt scrolling.
def var i as integer.

create tt. tt.f1 = 1.
create tt. tt.f1 = 2.
create tt. tt.f1 = 3.

open query q for each tt.
get next q.
get next q.
get next q.

create xtt. xtt.f1 = 4.

get prev q.

if avail(tt) then message string(tt.f1).
else message "N/A".
```

4GL output: 2
P2J output: 4.
Overall I'm concerned how deep down the rabbit hole can I go before returning to the original deleteResultListEntry issue.

#37 - 08/06/2014 12:07 PM - Stanislav Lomany

Found this in AdaptiveQuery javadoc:

```
* <b>WARNING:</b>  currently, the multi-table implementation is not
* functional.
* <p>
* TODO:  support multiple table queries.  The support is partially in place
* in the DynamicQuery inner class;  however, the instance of
* CompoundQuery which is created to drive the continuation of
* the query after invalidation currently does not have its state set
* correctly.
* <p>
```

#38 - 08/06/2014 12:32 PM - Eric Faulhaber

Stanislav Lomany wrote:

> Found this in AdaptiveQuery javadoc:
> [...]

Yes, I wrote that. All such cases are converted to use CompoundQuery instead, so you won't see a multi-table AdaptiveQuery come out of conversion currently. The multi-table implementation would be an optimization, but it is quite a complicated solution. Since we convert multi-table cases to CompoundQuery, there is no functional gap here, just a missing optimization.

#39 - 08/12/2014 05:18 AM - Stanislav Lomany

- File svl_upd20140812a.zip added

Fixes for testcases in notes 28 (not about flushing part, about PREV) and 34 (LAST issue). I didn't performed extensive testing with old testcases, I prefer you to review it first.
Now I'm moving to the adaptive scrolling queries cache.

#40 - 08/12/2014 08:44 AM - Eric Faulhaber

Code review 0812a:

I think it looks OK, but this whole area of the code is tricky to assess with just a review. I'll feel more comfortable after the more extensive testing you have planned (plus regression testing eventually) is successful.

#41 - 08/14/2014 11:41 AM - Stanislav Lomany

I finally got caching rules for adaptive queries (single-table):

1. Cache is advanced in forward (if the first iteration was FIRST) or reverse (the first iteration was LAST) direction as we move thru the result set using GET NEXT/PREV or REPOSITION.
2. LAST resets cache if it was advanced in forward direction and changes caching direction to opposite. FIRST - if in reverse.
3. Reaching the opposite off-end freezes cache: new records can't be added to the end of the result set, cache can't be reset using FIRST/LAST.
4. REPOSITION TO ROW resets cache if it was advanced in reverse direction and changes caching direction to forward.
5. REPOSITION FORWARD/BACKWARD/TO ROWID doesn't change caching direction.
6. For adaptive scrolling queries NUM-RESULTS doesn't make any iterations and returns the number of records in this cache.
7. Rules from note 26 apply.

Some P2J concerns:

1. In P2J REPOSITION FORWARD/BACKWARD/TO ROWID doesn't always correctly detect off-end.
2. In P2J NUM-RESULTS returns the total number of records and I can't easily change its behavior to correct because it is used by P2J browse and some other places.

#42 - 08/14/2014 11:58 AM - Eric Faulhaber

Stanislav Lomany wrote:

> I finally got caching rules for adaptive queries (single-table):
>
> 1. Cache is advanced in forward (if the first iteration was FIRST) or reverse (the first iteration was LAST) direction as we move thru the result set using GET NEXT/PREV or REPOSITION.
> 2. LAST resets cache if it was advanced in forward direction and changes caching direction to opposite. FIRST - if in reverse.
> 3. Reaching the opposite off-end freezes cache: new records can't be added to the end of the result set, cache can't be reset using FIRST/LAST.
> 4. REPOSITION TO ROW resets cache if it was advanced in reverse direction and changes caching direction to forward.
> 5. REPOSITION FORWARD/BACKWARD/TO ROWID doesn't change caching direction.
> 6. For adaptive scrolling queries NUM-RESULTS doesn't make any iterations and returns the number of records in this cache.
> 7. Rules from note 26 apply.

Nice work. Please report the test cases you used to determine these rules and check them into the testcases project if you haven't done so already.

How close is P2J to matching these rules, or better said, how much effort do you think is required to get it to this point?

> Some P2J concerns:
>
> 1. In P2J REPOSITION FORWARD/BACKWARD/TO ROWID doesn't always correctly detect off-end.
> 2. In P2J NUM-RESULTS returns the total number of records and I can't easily change its behavior to correct because it is used by P2J browse and some other places.

If browse and other runtime components depend upon the current implementation of NUM-RESULTS, we can refactor that implementation to another method to preserve it for internal runtime purposes, but create a correct implementation of NUM-RESULTS for use by converted code.

Are these last two points incidental issues you found along the way, or are they blocking the implementation of deleteResultListEntry?

#43 - 08/14/2014 05:03 PM - Stanislav Lomany

> Please report the test cases you used to determine these rules and check them into the testcases project if you haven't done so already.

See uast/adaptive_scrolling directory.

> How close is P2J to matching these rules, or better said, how much effort do you think is required to get it to this point?

Single table - about 2-3 days.

> Are these last two points incidental issues you found along the way, or are they blocking the implementation of deleteResultListEntry?

I think they do not block neither cache implementation nor deleteResultListEntry.

#44 - 08/21/2014 06:33 PM - Stanislav Lomany

Rules from note 41 apply to multi-table queries as well.

#45 - 09/25/2014 12:51 PM - Stanislav Lomany

- File svl_test20140925a.zip added

Test update for adaptive scrolling cache. Mixed with svl_upd20140812a (posted above, reviewed, not committed). Note that my decision to remove Cursor.offEnd field probably was a mistake.

#46 - 09/26/2014 11:32 AM - Stanislav Lomany

About off-end after reposition. It may have a quirky state. As far as I've checked the rules are following:

1. off-end is "yes" if the position is < -0.5 of > cacheSize - 1 + 0.5 (note that strict operators are used), except the following case:
2. off-end is "no" if we have the full cache and the current off-end state is "no".

However I have two concerns:
1. Some code in browse widget and queries use _isOffEnd() and isOffEnd() functions to detect if we have reached the last result (i.e. position is < 0 of > cacheSize - 1). My suggestion is to make _isOffEnd() return "useful" off-end and isOffEnd() - 4GL value. Of course we should decide if we want to implement off-end quirks an this point.
2. There is the case for which I cannot find a logical explanation (and I don't know how to implement it):

```
def temp-table tt field f1 as integer
```

```
          index idx1 f1.
def query q for tt scrolling.

create tt. tt.f1 = 1.
create tt. tt.f1 = 2.
create tt. tt.f1 = 3.

open query q for each tt. get first q. get prev q.
reposition q to row 0.
put stream s string(query-off-end("q")) " " string(current-result-row("q")) skip. /* yes */

open query q for each tt. get first q. get prev q.
reposition q to row 1.
put stream s string(query-off-end("q")) " " string(current-result-row("q")) skip. /* yes */

open query q for each tt. get first q. get prev q.
reposition q to row 2.
put stream s string(query-off-end("q")) " " string(current-result-row("q")) skip. /* yes */

open query q for each tt. get first q. get prev q.
reposition q to row 3.
put stream s string(query-off-end("q")) " " string(current-result-row("q")) skip. /* no */

open query q for each tt. get first q. get prev q.
reposition q to row 4.
put stream s string(query-off-end("q")) " " string(current-result-row("q")) skip. /* no */

open query q for each tt. get first q. get prev q.
reposition q to row 5.
put stream s string(query-off-end("q")) " " string(current-result-row("q")) skip. /* yes */

open query q for each tt. get first q. get prev q.
reposition q to row 100.
put stream s string(query-off-end("q")) " " string(current-result-row("q")) skip. /* yes */
```

#47 - 09/28/2014 03:21 PM - Stanislav Lomany

- File svl_upd20140928a.zip added

Update for adaptive scrolling cache. Fixed forward/backward. Doesn't reproduce 4GL off-end state issues noted in the previous note. Passed regression testing.

#48 - 09/30/2014 04:41 PM - Stanislav Lomany

Eric, there is a quirk in 4GL DELETE-RESULT-LIST-ENTRY implementation: for adaptive non-scrolling queries if DELETE-RESULT-LIST-ENTRY has been called before, following DELETE-RESULT-LIST-ENTRY calls will return "yes" (once) if the query is off-end. Should I implement this quirk at this point?

#49 - 09/30/2014 04:46 PM - Eric Faulhaber

If in your estimation it is not more than a few hours of effort, then yes -- it doesn't sound like it would be a lot of effort from your description. Best to get it right now than to trip over it later.

#50 - 10/01/2014 02:43 PM - Stanislav Lomany

I've checked the presence of PresortCompoundQuery in the customer app. It presents only in FOR EACH form, so there is no need to implement DELETE-RESULT-LIST-ENTRY for this type of queries. And even if I needed to implement it, I don't think this would be easy - I have some concerns about the way PresortCompoundQuery is implemented.

#51 - 10/01/2014 04:46 PM - Eric Faulhaber

OK, please thoroughly document the problem in a separate issue in this project, including why it is specific to PresortCompoundQuery, details of your concerns about the way PresortCompoundQuery is implemented, and your ideas about how an appropriate solution should be implemented. Provide enough detail that you or someone else can pick up this work later. Do not set an assignee, target, or start date. Link it as a related issue to this issue.

Other than this, is DELETE-RESULT-LIST-ENTRY fully implemented now? Is there anything else left to do on this task besides review and regression test?

#52 - 10/01/2014 11:13 PM - Eric Faulhaber

Code review 0928a:

I think the changes are OK, but please see my questions and comments below.

In note 45 above, you wrote:

> Note that my decision to remove Cursor.offEnd field probably was a mistake.

It is still removed in this update. Why was it a mistake, and how did you compensate/correct for it? Have we lost any functionality as a result?

Please list the test case names you used for your manual testing and be sure they are checked into the testcases project. The Cursor logic is too complicated to know whether the changes are correct just from a review, so I will have to rely on the results of your manual testing and regression testing. However, I noticed that your changes to the Cursor.reposition(Serializable[], boolean) method are not reflected in the javadoc, which still indicates that the methods throws IllegalStateException. Please review the other methods you have changed and update their javadoc as necessary (and that of the class, if needed) to reflect any changes you have made to the logic.

Why are you calling Collections.reverse(results) at Cursor:851?

Why did you make Cursor.next() public (from private)? If this needs to stay this way, please group it with the other public methods.

As with Cursor, the AdaptiveQuery logic is too complicated to confirm your changes from just a review, so I have to rely on your testing. However, I noticed your additions to AdaptiveQuery deal with composite rows (i.e., arrays of ids to represent multi-table row sets). However, AdaptiveQuery is currently limited to a single table implementation -- addComponent enforces this at runtime, and conversion will not create multi-table instances of AdaptiveQuery. What was the intention behind making your changes multi-table? In your estimation, how much effort is left to fully enabling multi-table support? If we could get this right, it should be much more performant than CompoundQuery in many cases.

#53 - 10/02/2014 01:52 AM - Stanislav Lomany

> Other than this, is DELETE-RESULT-LIST-ENTRY fully implemented now? Is there anything else left to do on this task besides review and regression test?

Adaptive non-scrolling queries remain.

#54 - 10/02/2014 07:47 AM - Stanislav Lomany

> Note that my decision to remove Cursor.offEnd field probably was a mistake.

> It is still removed in this update. Why was it a mistake, and how did you compensate/correct for it?

The problem with off-end is described in note 46, i.e. in 4GL off-end state doesn't always match "logical" off-end. And placing this quirky logic into Cursor may be a solution.

> Have we lost any functionality as a result?

Neither old nor new implementations doesn't fully conform 4GL because P2J just doesn't implement 4GL logic. If we want full conformity - we should just spent some time to implement it. Overall off-end behavior is better here and there, and sometimes it may be worse. But getOffEnd() is used internally by P2J, where it is meant for "logical" off-end, which it represents now, so from the practical point of view it became better.

> Please list the test case names you used for your manual testing and be sure they are checked into the testcases project.

See uast/adaptive_scrolling and uast/delete_entry folders.

> Why are you calling Collections.reverse(results) at Cursor:851?

That is the point at which cache built starting from the last element reaches the first element (i.e. we have the full set of rows). I found it easier for handling to change numeration from backward to forward. It is not an operation that I can't get rid of, just personal design preference.

> I noticed your additions to AdaptiveQuery deal with composite rows (i.e., arrays of ids to represent multi-table row sets). However, AdaptiveQuery is currently limited to a single table implementation -- addComponent enforces this at runtime, and conversion will not create multi-table instances of AdaptiveQuery. What was the intention behind making your changes multi-table?

I had no intention of making AdaptiveQuery multi-table. I've connect it with Cursor which is multi-table, so it turned out that way.

> In your estimation, how much effort is left to fully enabling multi-table support? If we could get this right, it should be much more performant than CompoundQuery in many cases.

Note that for the most part I've been modifying scrolling part of the AdaptiveQuery, which uses cache. And when we need to add a new cache row, we turn to core adaptive processing. I haven't looked into core processing, and cannot give estimates.

#55 - 10/02/2014 12:11 PM - Eric Faulhaber

OK, so my understanding of where this all stands now (and please correct me if I'm missing something) is that we have 3 unfinished areas that do not directly impact the implementation of DELETE-RESULT-LIST-ENTRY as used by the current project, but that need to be addressed ultimately:

- An issue with the PresortCompoundQuery implementation.
- Fixing adaptive, non-scrolling queries.
- Off-end behavior still not fully compatible with 4GL.

Please create separate Redmine issues for each of these, in this project (no assignee, target version, or start date). You can relate them to this issue, but each one should have all the relevant details one would need to understand and work on the problem.

Other than that, please make the minor changes (javadoc, method ordering) requested in note 52, then commit and distribute your update. These minor changes should have no impact on functionality, so you don't need to re-run regression testing.

#56 - 10/02/2014 05:00 PM - Stanislav Lomany

That is correct except for:

- Fixing adaptive, non-scrolling queries.

If you mean improving it for multi-table use, then yes, that is a separate task. But also I need to finish up DELETE-RESULT-LIST-ENTRY for this type of queries.

> Other than that, please make the minor changes (javadoc, method ordering) requested in note 52, then commit and distribute your update. These minor changes should have no impact on functionality, so you don't need to re-run regression testing.

I made several fixes for the adaptive cache along the way, so I will have to re-run regression (unless you want me to check in the existing update).

#57 - 10/02/2014 05:48 PM - Eric Faulhaber

Stanislav Lomany wrote:

> That is correct except for:
>
> - Fixing adaptive, non-scrolling queries.

> If you mean improving it for multi-table use, then yes, that is a separate task. But also I need to finish up DELETE-RESULT-LIST-ENTRY for this type of queries.

No, I misunderstood that you were saying there was something else that needed to be fixed for adaptive, non-scrolling queries. Please go ahead and finish the DELETE-RESULT-LIST-ENTRY implementation.

> > Other than that, please make the minor changes (javadoc, method ordering) requested in note 52, then commit and distribute your update. These minor changes should have no impact on functionality, so you don't need to re-run regression testing.

> I made several fixes for the adaptive cache along the way, so I will have to re-run regression (unless you want me to check in the existing update).

No, don't check it in. Please make the requested code changes, open the other two issues (for PresortCompoundQuery and the remaining, non-conforming, off-end behavior), finish with the DELETE-RESULT-LIST-ENTRY implementation, then repost an update for review.

#58 - 10/15/2014 01:34 PM - Stanislav Lomany

- File svl_upd20141015a.zip added

Final update for review. Latest changes are related to DELETE-RESULT-LIST-ENTRY and peekNext/peekPrevious in compound query.

#59 - 10/15/2014 03:22 PM - Eric Faulhaber

Code review 1015a:

I found nothing problematic, though it is quite a large and complicated update, so it is hard to tell. Please regression test.

#60 - 10/21/2014 10:06 AM - Stanislav Lomany

- File svl_upd20141020a.zip added

Update with fixed regressions for review. One regression was minor overlook, two were related to FilteredResults which are extending modified SimpleResults. Passed regression testing (except three tc_job_clock tests).

#61 - 10/21/2014 10:16 AM - Eric Faulhaber

Code review 1020a:

Change look good.

> Passed regression testing (except three tc_job_clock tests)

Do you have another run in which the failing tests have passed (except for tc_job_002, step 40)?

#62 - 10/21/2014 10:29 AM - Stanislav Lomany

I have two runs with tc_job_clock_002 and tc_job_clock_004 failed. So I'll run regression one more time.

#63 - 10/22/2014 02:39 AM - Stanislav Lomany

- Status changed from WIP to Review

Committed to bzr revision 10633.

#64 - 10/25/2014 10:40 PM - Eric Faulhaber

- % Done changed from 0 to 100

- Status changed from Review to Closed

#65 - 11/16/2016 12:06 PM - Greg Shah

- Target version changed from Milestone 11 to Cleanup and Stablization for Server Features

Files

svl_upd20140604a.zip	193 KB	06/04/2014	Stanislav Lomany
svl_upd20140812a.zip	36.6 KB	08/12/2014	Stanislav Lomany
svl_test20140925a.zip	156 KB	09/25/2014	Stanislav Lomany
svl_upd20140928a.zip	156 KB	09/28/2014	Stanislav Lomany
svl_upd20141015a.zip	164 KB	10/15/2014	Stanislav Lomany
svl_upd20141020a.zip	169 KB	10/21/2014	Stanislav Lomany