Database - Feature #1999

add conversion support for builtin functions in a WHERE clause

02/15/2013 04:47 PM - Eric Faulhaber

Status:	Closed	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Vadim Nebogatov	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:	Conversion Support for Server Features		
billable:	No	**vendor_id:**	GCD

Description

History

#1 - 02/15/2013 05:07 PM - Eric Faulhaber

The following builtin functions are used within a WHERE clause in the current project (in order of most to least used):

- recid()
- today()
- can-do()
- trim()
- rowid()
- entry()
- integer()
- substring()
- lookup()
- if()
- string()
- can-find()
- to-rowid()
- length()
- available()
- weekday()
- program-name()
- chr()
- date()
- year()
- caps()
- input()
- left-trim()
- day()
- index()
- month()
- quoter()
- time()
- dynamic-function()
- maximum()
- decimal()
- minimum()
- logical()
- num-entries()
- random()
- fill()
- replace()

Today we convert and have runtime implementations for many of these. For those cases, the only thing to do is to identify unsupported options.

We need to add conversion support and runtime stubs for the remaining functions.

Builtin functions generally can be inlined into the WHERE clauses of FIND, FOR, QUERY, etc. database statements. This often requires special handling during conversion. Depending on the nature of the parameters and context of the calls, function calls sometimes convert as query substitution parameters (evaluated in the P2J application server), and sometimes as server-side function calls embedded in the HQL string passed to a P2JQuery implementation class' method or constructor (the calls are then evaluated in a JVM running within the database server).

Please use the examples of the existing, supported functions when adding support for a new one.

Rule sets to look at include convert/builtin_functions.rules and annotate/where_clause*.rules.

#2 - 02/18/2013 12:58 PM - Vadim Nebogatov

- Status changed from New to WIP

#3 - 02/20/2013 10:41 AM - Eric Faulhaber

Please provide a summary of which functions are supported by conversion already and which need work. We need to get a better sense of how much work there is to do here.

#4 - 02/21/2013 12:34 PM - Vadim Nebogatov

- File issue1999.ods added

Spreadsheet with testing results is attached.

#5 - 02/21/2013 12:39 PM - Vadim Nebogatov

Briefly, all functions are supported with all options excepting

1) can-find() - works in general, but at least FIRST/LAST, SHARE-LOCK/NO-LOCK/NO-WAIT/NO-PREFETCH and possible some more options are ignored
2) chr() - supported without options target-codepage/source-codepage

#6 - 02/21/2013 07:19 PM - Eric Faulhaber

Vadim Nebogatov wrote:

> Briefly, all functions are supported with all options excepting
>
> 1) can-find() - works in general, but at least FIRST/LAST, SHARE-LOCK/NO-LOCK/NO-WAIT/NO-PREFETCH and possible some more options are ignored

I'm pretty sure I remember adding support for lock options in a CAN-FIND, but not necessarily with a NO-WAIT. There is one use of a CAN-FIND with a SHARE-LOCK NO-WAIT option in the current project.

Please create and convert a test case with CAN-FIND(FIRST ... WHERE ... SHARE-LOCK) and another with CAN-FIND(FIRST ... WHERE ... SHARE-LOCK NO-WAIT). Report what the result is.

> 2) chr() - supported without options target-codepage/source-codepage

These options are not used in the current project.

#7 - 02/21/2013 08:30 PM - Vadim Nebogatov

CAN-FIND: SHARE-LOCK and NO-WAIT
Yes, you are right, it is supported. I did not notice that WHERE clause is generated separately.

```
FIND CURRENT hTemp WHERE (CAN-FIND(FIRST hTemp WHERE hTemp.ctChar = "hhh" SHARE-LOCK)).

P2J code:

  WhereExpression whereExpr0 = new WhereExpression()
   {
      public logical evaluate(final BaseDataType[] args)
      {
         return new FindQuery(htemp, "upper(htemp.ctchar) = 'HHH'", null, (String) null, LockType.SHARE).hasAn
y();
      }
   };
            new FindQuery(htemp, (String) null, whereExpr0, "htemp.itint asc, htemp.ctchar asc").current();

FIND CURRENT hTemp WHERE (CAN-FIND(FIRST hTemp WHERE hTemp.ctChar = "hhh" SHARE-LOCK NO-WAIT)).

P2J code:

   WhereExpression whereExpr1 = new WhereExpression()
   {
      public logical evaluate(final BaseDataType[] args)
      {
         return new FindQuery(htemp, "upper(htemp.ctchar) = 'HHH'", null, (String) null, LockType.SHARE_NO_WAI
T).hasAny();
      }
   };
new FindQuery(htemp, (String) null, whereExpr1, "htemp.itint asc, htemp.ctchar asc").current();

FIND CURRENT hTemp WHERE (CAN-FIND(FIRST hTemp WHERE hTemp.ctChar = "hhh" NO-LOCK)).

P2J code:

            new FindQuery(htemp, "exists(from TempRecord2 as htemp where upper(htemp.ctchar) = 'HHH')", null,
"htemp.itint asc, htemp.ctchar asc").current();
```

CAN-FIND: FIRST/LAST
It seems it is not supported properly, but differs in case when this option is presented or absent(hasAny/hasOne)

```
FIND CURRENT hTemp WHERE (CAN-FIND(LAST hTemp WHERE hTemp.ctChar = "hhh" SHARE-LOCK)).

P2J code:
   WhereExpression whereExpr2 = new WhereExpression()
   {
      public logical evaluate(final BaseDataType[] args)
      {
         return new FindQuery(htemp, "upper(htemp.ctchar) = 'HHH'", null, (String) null, LockType.SHARE).hasAn
y();
      }
   };
new FindQuery(htemp, (String) null, whereExpr2, "htemp.itint asc, htemp.ctchar asc").current();

FIND CURRENT hTemp WHERE (CAN-FIND(hTemp WHERE hTemp.ctChar = "hhh" SHARE-LOCK)).

P2J code:

  WhereExpression whereExpr3 = new WhereExpression()
   {
      public logical evaluate(final BaseDataType[] args)new FindQuery(htemp, (String) null, whereExpr3, "htemp
.itint asc, htemp.ctchar asc").current();
      {
         return new FindQuery(htemp, "upper(htemp.ctchar) = 'HHH'", null, (String) null, LockType.SHARE).hasOn
e();
      }
   };
new FindQuery(htemp, (String) null, whereExpr3, "htemp.itint asc, htemp.ctchar asc").current();
```

#8 - 02/21/2013 10:28 PM - Eric Faulhaber

Vadim Nebogatov wrote:

> CAN-FIND: SHARE-LOCK and NO-WAIT
> Yes, you are right, it is supported. I did not notice that WHERE clause is generated separately.

Well, yes, it is supported, but by using a temp-table in your test case, you have highlighted an interesting point. When no lock option is specified, we convert a CAN-FIND embedded in a WHERE clause to HQL which contains a subselect. The work of the CAN-FIND is all done on the database server. None of this WhereExpression nonsense is emitted in that case.

However, when a SHARE or EXCLUSIVE lock option is used, we can't use a subselect and we convert to this strange mechanism, with a separate WhereExpression instance that is passed to the FindQuery constructor. This is needed because the question of whether a found record can be locked cannot be answered on the database server with a subselect; it must be answered at the application server, where the runtime lock manager resides.

This convoluted approach is correct when using a permanent table (i.e., non-temp-table). However, it is unnecessarily complicated for a temp-table, because locking a temp-table record is a no-op. Temp-tables are local to a session, not global across sessions. So, the question of whether a lock can be obtained on a temp-table record is a meaningless one. If a record can be found, it can be locked, because no one else will ever hold a lock on that record; it is private.

So, we could optimize this conversion (i.e., CAN-FIND on a temp-table with a lock option) to convert to the more straightforward way. Probably the simplest way to do this is just to hide the lock option. This is a low priority; I'll open a separate issue for it.

> CAN-FIND: FIRST/LAST
> It seems it is not supported properly, but differs in case when this option is presented or absent(hasAny/hasOne)

That's interesting. We use hasAny when the FIRST or LAST keyword is present. This was intentional and it is correct when there is no lock, because when one is simply trying to determine whether the first or last record can be found, it really doesn't matter whether it is the first or last record that's found, just than **any** record can be found. However -- and I hadn't considered this before now -- when a lock is involved, then the specific record (i.e., first or last) **does** matter, because the request is to determine not only whether a specific record can be found, but also whether it can be locked. We'll have to fix this ultimately, but not for M4. I'll open a separate issue for this as well.

#9 - 02/21/2013 11:15 PM - Eric Faulhaber

- % Done changed from 0 to 100

- Status changed from WIP to Closed

I am closing this issue. While it brought up some interesting issues with our CAN-FIND support (which are now documented separately), all of the other functions in the list are already supported.

#10 - 11/16/2016 11:08 AM - Greg Shah

- Target version changed from Milestone 4 to Conversion Support for Server Features

Files

issue1999.ods	18.2 KB	02/21/2013	Vadim Nebogatov