Metacat - Bug #2084

Pathquery support for temporal search on date fields

05/20/2005 04:02 PM - Duane Costa

Status:	Resolved	**Start date:**	05/20/2005
Priority:	Normal	**Due date:**	
Assignee:	ben leinfelder	**% Done:**	0%
Category:	metacat	**Estimated time:**	0.00 hour
Target version:	2.0.0	**Spent time:**	0.00 hour
Bugzilla-Id:	2084		

Description

Metacat pathquery relational search modes ("greater-than", "less-than", etc.) do not currently support temporal searches on date fields. The reasons for this are described in the email correspondence to metacat-dev below. This enhancement would make it possible to do temporal searches using date ranges, which would be a important feature in an "Advanced Search" form (such as the one currently under development at LTER), and could also be added to the search dialog in Morpho.

On 5/17/2005, Duane Costa wrote:

Metacat supports the following pathquery search modes: contains, starts-with, ends-with, equals, isnot-equal, greater-than, less-than, greater-than-equals, less-than-equals.

For the search modes that are equivalent to relational operators (equals, isnot-equal, greater-than, less-than, greater-than-equals, less-than-equals), is it possible to use these search modes in EML fields that contain non-numeric string values? In particular, is it possible to use the relational search modes for date strings?

For example, here is a pathquery that attempts to find all documents with temporal coverage between January 1, 1900 and January 1, 2005. It reads like this: “Return all documents that have a beginDate or a singleDateTime greater than or equal to 1900-01-01, and an endDate or a singleDateTime less than or equal to 2005-01-01.”

```
<query>
<pathquery version="1.2">
<querytitle>LTER Query</querytitle>
<returnfield>dataset/title</returnfield>
<returnfield>originator/individualName/surName</returnfield>
<returnfield>creator/individualName/surName</returnfield>
<returnfield>originator/organizationName</returnfield>
<returnfield>creator/organizationName</returnfield>
<returnfield>keyword</returnfield>
<querygroup operator="INTERSECT">
<querygroup operator="INTERSECT">
<querygroup operator="INTERSECT">
<querygroup operator="UNION">
<queryterm searchmode="greater-than-equals" casesensitive="false">
<value>1900-01-01</value>
<pathexpr>temporalCoverage/rangeOfDates/beginDate/calendarDate</path
expr>
</queryterm>
<queryterm searchmode="greater-than-equals" casesensitive="false">
<value>1900-01-01</value>
<pathexpr>temporalCoverage/singleDateTime/calendarDate</pathexpr>
</queryterm>
</querygroup>
<querygroup operator="UNION">
<queryterm searchmode="less-than-equals" casesensitive="false">
```

<value>2005-01-01</value>
<pathexpr>temporalCoverage/rangeOfDates/endDate/calendarDate</pathex
pr>
</queryterm>
<queryterm searchmode="less-than-equals" casesensitive="false">
<value>2005-01-01</value>
<pathexpr>temporalCoverage/singleDateTime/calendarDate</pathexpr>
</queryterm>
</querygroup>
</querygroup>
</querygroup>
</querygroup>
</pathquery>
</query>

When I run this against a test Metacat with an Oracle database, this pathquery fails and the resultset contains zero documents. The Tomcat output shows that the pathquery triggers a SQL error in Oracle:

```
MetaCat: SQL Error in DBQuery.findDocuments: ORA-01722: invalid number
```

So my question is this: can this problem with temporal search be fixed in the same way that Sid fixed a similar bug for spatial search (Bug 1703, 1718), or is this a different situation because of the fact that the temporal fields contain non-numeric strings while the spatial fields contain numeric values? That is, is it illegal to use the relational pathquery modes for EML fields that contain non-numeric strings? If that is the case, it seems that there would be no practical way to use pathquery for a temporal search involving date ranges: is that correct?

On 5/18/2005, Chris Jones wrote:

Duane,

I've thought about this myself, and it seems that an internal metacat solution needs to happen where date/time strings are converted or linked to a universal date representation in order to do comparisons. This might be tough. Each vendor's database seems to store dates internally in very different ways. It would be difficult to create another column in the xml_nodes table that is of type 'date' (depending on the vendor), because an EML (or other XML) date string isn't necessarily recognizable as a date, whereas an integer or float is much more discernible (in the case of the nodedatanumerical column) in xml_nodes.

However, with strong typing used in XMLSchema, metacat could in theory glean 'date' strings as type 'date' by referring to the element definition in the XMLSchema document to which the instance document adheres. In that case, a nodedatadate column in xml_nodes might work out where, upon insert or update, the leaf node values get put into that column as a converted date. Of course, this leads to: which date formats will be supported for conversion?

Anyone have a better solution?

Chris

On 5/20/2005, Matt Jones wrote:

Chris and Duane,

I think Chris' analysis of the issues with adding support for datetime comparisons is just about right on. It would be a nice feature to have, but implementation would need to accomodate multiple database systems. I've even been contemplating adding support for Sleepycat XMLdb or other XML databases instead of relational backends, and this would further complicate the implementation issues for datetime values. But it would be worthwhile. Lets get it into bugzilla as a feature request and we'll see where it falls out in

the priority order.

Matt

History

#1 - 12/08/2005 10:14 AM - Saurabh Garg

Moving to 1.7 as not enough time to get this done for 1.6 deadline.

#2 - 03/16/2011 10:55 PM - ben leinfelder

Added nodedatadate column to the following tables: xml_nodes, xml_nodes_revisions, xml_path_index.
Included DB update statements for postgres and oracle.
Included Java upgrade utility (and mechanism for doing this in future releases) that converts date and dateTime nodedata values and inserts them into the nodedatadate column.
Formats supported for insert/searching are two common ISO 8601:
yyyy-MM-dd
yyyy-MM-ddThh:mm:ss

Note that yyyy is supported as a numerical comparison already.

#3 - 03/23/2011 04:50 PM - ben leinfelder

I'd like to expand the date formats we can accept to include all valid xs:dateTime formats (ISO 8601). I found I could use javax.xml.bind.DatatypeConverter.parseDateTime(String) to do all the heavy lifting - the only problem is that it will parse some pretty far-out dates and consider them valid rather than throwing an exception:
For example this random numeric value:
"1297467700008"
is parsed as:
"196973390-06-18 09:34:08.384000-0800 BC"
200 million years ago, indeed.
When I try to insert this date in Postgres (and likely other DBs) it complains about being out of range:
java.sql.SQLException: ERROR: timestamp out of range: "196973390-06-18 09:34:08.384000-0800 BC"
And it is, indeed, out of the range that postgres accommodates.
I am hesitant to add much date-range checking to the insert code for fear that it will be inappropriately draconian and/or a performance hit. But if I were to check the date, what range would be appropriate? I'd say it should be the intersection of postgres' and oracle's capabilities even though this can change from version to version and can be changed depending on the storage mechanism you configure if you're an advanced DB admin. Still it's a starting place.
Alternatively, we could attempt numeric parsing first, in which case this non-delimited ISO format would be considered simple numeric data rather than a date. We'd still get range-based query operations, but it would be numeric rather than date-based.

#4 - 03/23/2011 08:00 PM - Matt Jones

I agree that DatatypeConverter.parseDateTime(String) is a great way to go, ans supporting as much of the ISO 8601 as possible is best -- that's what we claim for EML, so it would match. If the DB can't handle the dates, it will throw an exception as you indicate and we can fall back to treating that particular value as a string (treating it as a number is probably not that useful). It seems to me that you have a good approach worked out.

#5 - 03/24/2011 09:25 AM - ben leinfelder

My concern is with catching the exception from the database level in that it will be a performance hit. But here's the plan:
1. Insert the node value as string (no numeric or date parsing yet).*
2a. Try parsing the node value as a dateTime.
2b. If parsing passes, we attempt to update the node row with this date value, otherwise carry on with 3.
2c. If this fails (date out of range) we carry on with 3.
3a. Parse node value as a number
3b. If it's a number we update the row with the numeric node value.

So in the case that I previously laid out we would have three prepared statements executing for this one xml node (insert, a failed update, and a successful update). But in most cases we would have only the insert because ISO spec is still pretty constrained and numeric parsing is quite strict. We'd still be catching Java ParseExceptions, but hopefully only a small number of SQLExceptions which are much more costly.

*I want to insert the node value as only a string initially so that the SQLException does not prevent the row from being written if/when there is a DB exception raised for a parsed date value.

#6 - 04/06/2011 11:39 AM - ben leinfelder

this is now in trunk and includes a utility method that runs during DB upgrade.

running it on dev I ran into OutOfMemory errors while upgrading - need to keep an eye on this and how the servers are configured WRT memory allocation.

#7 - 10/26/2011 04:30 PM - ben leinfelder

This should be in 2.0.0 since it is already in the trunk. The upgrade script is the only thing I am worried about -- but chunking up the modifications

should resolve it.

#8 - 10/27/2011 01:01 PM - ben leinfelder

Will exercise this during testing for release.

#9 - 06/06/2012 12:50 PM - gastil gastil

Invalid dates are allowed into Metacat 2.0.0 if the eml is 2.0.1.
I thought those were supposed to be caught.

If you want to test this, an example doc to use is
http://metacat.lternet.edu/knb/metacat/knb-lter-fce.515/lter
which is un-patched eml 2.0.1
or
http://lava.lternet.edu/knb/metacat/knb-lter-bug.515.1/lter
which is patched eml 2.0.1
or
https://demo2.test.dataone.org/knb/metacat/knb-lter-bug.515.1/default

This eml doc has
<rangeOfDates>
<beginDate>
<calendarDate>25569</calendarDate>
</beginDate>
<endDate>
<calendarDate>36891</calendarDate>
</endDate>
</rangeOfDates>

#10 - 09/05/2012 09:52 AM - ben leinfelder

Unfortunately, those values are "schema valid"

(In reply to comment #9)

> Invalid dates are allowed into Metacat 2.0.0 if the eml is 2.0.1.
> I thought those were supposed to be caught.
>
> If you want to test this, an example doc to use is
> http://metacat.lternet.edu/knb/metacat/knb-lter-fce.515/lter
> which is un-patched eml 2.0.1
> or
> http://lava.lternet.edu/knb/metacat/knb-lter-bug.515.1/lter
> which is patched eml 2.0.1
> or
> https://demo2.test.dataone.org/knb/metacat/knb-lter-bug.515.1/default
>
> This eml doc has
> <rangeOfDates>
> <beginDate>
> <calendarDate>25569</calendarDate>
> </beginDate>
> <endDate>
> <calendarDate>36891</calendarDate>
> </endDate>
> </rangeOfDates>

#11 - 03/27/2013 02:19 PM - Redmine Admin

Original Bugzilla ID was 2084