VegBank - Bug #2434

Profiling of XML Loader's Performance

05/09/2006 08:15 AM - Michael Lee

Status:	Resolved	**Start date:**	05/09/2006
Priority:	Immediate	**Due date:**	
Assignee:	Chad Berkley	**% Done:**	0%
Category:	userLoad	**Estimated time:**	0.00 hour
Target version:	1.1.0	**Spent time:**	0.00 hour
Bugzilla-Id:	2434		

Description

During the week of 5/8/2006, the XML Loader should be profiled to see why it takes as long as it does and whether there are places where we could improve it. This will also serve as a good way of getting to know the XML Loader and the data model.

http://vegbank.org/erd is a good resource for learning the model (click on a table to link to the data dictionary)

Related issues:

Blocked by VegBank - Bug #1860: XML Loader: need to read incoming accessionCo...	**Resolved**	**01/10/2005**
Blocked by VegBank - Bug #1859: XML Loader: need receipt for VegBranch to upd...	**Resolved**	**01/10/2005**

History

#1 - 06/02/2006 02:23 PM - Chad Berkley

I inserted a timer into the loading code. Below are the results for a very small xml file. A 0 in the timer means it took less than 1 second to perform.

Loaded Items:
covermethod (1)
stratummethod (1)
commclass (1)
observation (3)
plot (3)
party (4)
project (1)
commconcept (1)
reference (2)

TIMER: initing db :::total time: 0
TIMER: inserting commConcepts :::total time: 0
TIMER: inserting plantConcepts :::total time: 0
TIMER: inserting references :::total time: 0
TIMER: inserting parties :::total time: 0
TIMER: inserting observations :::total time: 24
TIMER: committing data to db :::total time: 0
TIMER: creating dataset :::total time: 0
TIMER: create accession codes :::total time: 0
TIMER: denormalizing sql :::total time: 0
TIMER: generate keywords :::total time: 1
TIMER: inserting vegbank package :::total time: 31
TIMER: Time for SAX parse :::total time: 33
TIMER: Time to process xml file :::total time: 34

So this seems to indicate that the most time is being spent on inserting observations into the db. I will profile that part of the load process for further results.

#2 - 06/05/2006 01:44 PM - Chad Berkley

It seems a lot of time is being spent in the LoadTreeToDatabase.insertTaxonObservations() method. For a run of the same simple xml file, here is the timing info:

TIMER: initing db :::total time: 0
TIMER: inserting commConcepts :::total time: 0
TIMER: inserting plantConcepts :::total time: 0
TIMER: inserting references :::total time: 0
TIMER: inserting parties :::total time: 0
TIMER: IOT1 :::total time: 0
TIMER: IOT2 :::total time: 0

TIMER: IOT3 :::total time: 0
TIMER: IOT4 :::total time: 0
TIMER: IOT5 :::total time: 0
TIMER: IOT6.0 :::total time: 14
TIMER: IOT6.1 :::total time: 0
TIMER: IOT6.2 :::total time: 0
TIMER: IOT1 :::total time: 0
TIMER: IOT2 :::total time: 0
TIMER: IOT3 :::total time: 0
TIMER: IOT4 :::total time: 0
TIMER: IOT5 :::total time: 0
TIMER: IOT6.0 :::total time: 1
TIMER: IOT6.1 :::total time: 0
TIMER: IOT6.2 :::total time: 0
TIMER: IOT1 :::total time: 0
TIMER: IOT2 :::total time: 0
TIMER: IOT3 :::total time: 0
TIMER: IOT4 :::total time: 0
TIMER: IOT5 :::total time: 0
TIMER: IOT6.0 :::total time: 5
TIMER: IOT6.1 :::total time: 0
TIMER: IOT6.2 :::total time: 0
TIMER: inserting observations :::total time: 22
TIMER: committing data to db :::total time: 0
TIMER: creating dataset :::total time: 0
TIMER: create accession codes :::total time: 0
TIMER: denormalizing sql :::total time: 0
TIMER: generate keywords :::total time: 1
TIMER: inserting vegbank package :::total time: 30

IOT6.0 represents the call time for insertTaxonObservations(). It accounts for 90% (20 of the total 22 seconds) of the total insert observations time and 73% of the overall loading time. Unfortunately, there are no easily identifiable sticking points within this method. It seems that the database structure that contains this info is very complex, requiring complex processing to parse the data into it. I will keep looking at this method to see if I can identify any areas that could be streamlined.

#3 - 06/05/2006 02:55 PM - Chad Berkley

Michael pointed out that the accessionCode field in several tables gets hit a lot. Since the field is pretty static, we added indices to all of the foreign keys. This has significantly improved the performance. The xml load for the standard test file is now only taking 10 seconds instead of 35. Here is the timer trace now:

TIMER: initing db :::total time: 0 seconds
TIMER: inserting commConcepts :::total time: 0 seconds
TIMER: inserting plantConcepts :::total time: 0 seconds
TIMER: inserting references :::total time: 0 seconds
TIMER: inserting parties :::total time: 0 seconds
TIMER: IOT1 :::total time: 0 seconds
TIMER: IOT2 :::total time: 0 seconds
TIMER: IOT3 :::total time: 0 seconds
TIMER: IOT4 :::total time: 0 seconds
TIMER: IOT5 :::total time: 0 seconds
TIMER: IOT6.0 :::total time: 0 seconds
TIMER: IOT6.1 :::total time: 0 seconds
TIMER: IOT6.2 :::total time: 0 seconds
TIMER: IOT1 :::total time: 0 seconds
TIMER: IOT2 :::total time: 0 seconds
TIMER: IOT3 :::total time: 0 seconds
TIMER: IOT4 :::total time: 0 seconds
TIMER: IOT5 :::total time: 0 seconds
TIMER: IOT6.0 :::total time: 0 seconds
TIMER: IOT6.1 :::total time: 0 seconds
TIMER: IOT6.2 :::total time: 0 seconds
TIMER: IOT1 :::total time: 0 seconds
TIMER: IOT2 :::total time: 0 seconds
TIMER: IOT3 :::total time: 0 seconds
TIMER: IOT4 :::total time: 0 seconds
TIMER: IOT5 :::total time: 0 seconds
TIMER: IOT6.0 :::total time: 0 seconds
TIMER: IOT6.1 :::total time: 0 seconds
TIMER: IOT6.2 :::total time: 0 seconds
TIMER: inserting observations :::total time: 0 seconds
TIMER: committing data to db :::total time: 0 seconds
TIMER: creating dataset :::total time: 0 seconds
TIMER: create accession codes :::total time: 0 seconds
TIMER: denormalizing sql :::total time: 0 seconds

TIMER: generate keywords :::total time: 2 seconds
TIMER: inserting vegbank package :::total time: 9 seconds
TIMER: Time for SAX parse :::total time: 10 seconds
TIMER: Time to process xml file :::total time: 10 seconds

#4 - 06/07/2006 01:24 PM - Chad Berkley

I've finished my analysis of the observation insertion part of LoadTreeToDatabase.insertVegbankPackage(). It is still taking a non-trivial amount of time, but no one part of the method is taking more than a second, so it looks to me that the processing time of this method is a function of the amount of work it takes to insert data into the database. Since the data and the database are both pretty complex, it takes more time to process it.

I'm now looking at the KeywordGen class which generates keywords from each dataset and also takes a non-trivial amount of time.

#5 - 06/08/2006 12:34 PM - Chad Berkley

I've been looking at possible ways to speed the loader up even more than we already have for the last 2 days. Basically, I think that to have a significant increase in performance, the java loading system itself is going to have to be revamped. This is non-trivial and might end up taking a long time (monthes). I'm not sure if this is high enough priority to get into such an effort.

If this were to take place, I would recommend the following:
1) The SAX parser parse and load per observation instead of per vegbankPackage. Right now the entire vegbankPackage is read into a hash table before the datagbase insertion even begins. For large xml files, this uses a lot of memory. I had to increase my base Tomcat memory just to get one of the larger files to load at all. Instead of loading the entire document into memory, it is entirely feasable to parse one observation at a time, insert that observation into the DB, then go onto the next. Observations tend to be much smaller datastructures than the entire vegbankPackage so I think this would reduce memory requirements and increase overall load time.

2) Handle the keyword creation outside of Java. Right now, keywords are generated for several tables (project, covermethod, stratummethod, commconcept, party, place, observation) to increase search speed on pertinent keywords within the package. Instead of doing this in Java, which requires lots of semi-large Hastables and Vectors, a postgres trigger should probably be written to handle the keyword creation. The trigger could be written in any number of scripting languages or could be pre-compiled C code. I'm not a fan of the C code idea since it would make installation on multiple environments a PITA, but it would be fast. This is also a non-trivial change to the system and could potentially take a month or more to complete.

3) Do some further analysis to find more fields like accessionCode that get accessed a lot and index them. I looked at the fields and the sql a bit, but since I'm still pretty unfamiliar with the datamodel, it's not clear to me what fields get the most hits. Michael could probably identify these easier than I can.

#6 - 06/08/2006 12:39 PM - Chad Berkley

Typo fixes for the last comment...oops.

(In reply to comment #5)
1) **Change the** SAX parser parse and load per observation instead of per vegbankPackage.
.....
reduce memory requirements and **decrease** overall load time.

#7 - 09/06/2006 03:12 PM - Chad Berkley

this is as done as it's going to get. the only way this is going to get faster is to significantly change the code base or the schema.

#8 - 03/27/2013 02:20 PM - Redmine Admin

Original Bugzilla ID was 2434