Environment and organisms - Task #363

Assemble all GHCN data into a single database

02/23/2012 01:13 PM - Benoit Parmentier

Status:	In Progress	**Start date:**	05/09/2012
Priority:	High	**Due date:**	
Assignee:	Jim Regetz	**% Done:**	80%
Category:	Climate	**Estimated time:**	4.00 hours
Target version:			
Activity type:			

Description

Meteorological station data used in the project was downloaded from the Global Historical Climate Network (GHCN). This data is distributed in a variety of formats. The daily global data (GHCND) data for the year 2010 was used for the pilot test in Oregon. The dataset consists in a textfile with fields related to Tmin, Tmax, Precip and Snow depth. Station locations are stored separately so that data must be linked to the meteorological station locations using the Station ID. For the initial test, data related to the Tmax variable was selected and joined to the Oregon station locations. This process may need to be repeated for each variable (e.g. Tmin and Precip etc.).

The processing was done using an unix shell script but could be replicated through R, python or some SQL database system.

History

#1 - 02/23/2012 01:15 PM - Benoit Parmentier

- Assignee set to Benoit Parmentier

#2 - 05/09/2012 11:10 AM - Jim Regetz

- Subject changed from Preprocessing and formatting of downloaded meteorological station data to Assemble all GHCN data into a single database

- Status changed from New to In Progress

- Assignee changed from Benoit Parmentier to Jim Regetz

- Priority changed from Normal to High

- Start date changed from 02/23/2012 to 05/09/2012

- % Done changed from 100 to 0

- Estimated time set to 4.00 h

We decided to organize and load *all* available GHCN data (i.e., all dates and stations, globally) into SQLite database. This will simplify the data management, allow us to do some basic aggregation directly in the database via SQL, and perform arbitrary queries against it as needed from R, Python, etc.

Current data location: atlas:~layers/data/climate/ghcn/ghcnd_all.tar.gz

#3 - 05/09/2012 11:49 PM - Jim Regetz

- % Done changed from 0 to 80

Committed an R script that orchestrates the processing and loading of all 75000+ *.dly files into a SQLite database. I initially used core R functions read.fortran and reshape to do most of the preprocessing work in a straightforward manner. However, after initial tests indicated this would take a bunch of days to run, I replaced those with customized functions to speed things up a bit (including delegating to grep/awk/tr for initial filtering and pre-processing).

Current incarnation is at source:climate/extra/ghcn-to-sqlite.R@d4e63f5d

I'm currently running this to load all TMIN/TMAX records as a first pass to see what we get overnight.

#4 - 05/10/2012 08:49 AM - Jim Regetz

Data loading slowed to a crawl overnight, thanks to a record count query tucked inside in the bulk insert function that gets called on each daily file -- O(n) with number of records in the table. I removed the offending statement (3b8dd010) and restarted the load process where it left off.

#5 - 05/13/2012 11:14 PM - Jim Regetz

Belated update. The TMIN/TMAX data loaded in ~6 hours total, yielding a 25GB SQLite file with nearly 600 million rows.

#6 - 05/13/2012 11:16 PM - Jim Regetz

Vaguely concerned about performance and manageability of such a large SQLite database, I wrote a second script to load the data into a PostgreSQL database. Simply swapping out the RSQLite functions for roughly equal RPostgreSQL functions made things noticeably slower, but I was able to make up the difference (and then some) by switching to loading the wide format data into a PostgreSQL staging table and then using SQL UNIONs to transform these data into the long (one day per record) destination table.

Loading all TMIN, TMAX, and PRCP data into this database took ~11 hours, yielding ~1.4 billion rows. I've also built column indexes on 'station' and on 'element' -- each of which took half a day to complete. Presumably we'll only ever need to do processing on smallish subsets of stations at a time, and this should be reasonably fast with the station index.

Database holdings, by element type:

```
ghcn=# select element, count(*) from ghcn group by element;
 element |   count
---------+-----------
 PRCP    | 815002338
 TMAX    | 297159769
 TMIN    | 297320380
(3 rows)
```

(This query itself took ~21 minutes.)

The script as run is committed to the repo at source:climate/extra/ghcn-to-psql.R@c7235eae

#7 - 05/14/2012 10:00 AM - Jim Regetz

Quick example comparing SQLite vs PostgreSQL timings on a simple aggregate query that doesn't use any indexes. SQLite takes nearly twice as long even though it doesn't have the precip records and is thus less than half the size. Of course, both give you enough time to grab a quick lunch.

sqlite

```
$ time echo "select element, min(value) from ghcn where value<>-9999 group by element;" | sqlite3 ghcn_tmintma
x.db
TMAX|-5733
TMIN|-5733

real    54m10.936s
```

postgres

```
$ time echo "select element, min(value) from ghcn where value<>-9999 group by element;" | psql ghcn
 element |  min
---------+-------
 PRCP    | -1575
 TMAX    | -5733
 TMIN    | -5733
(3 rows)

real    31m22.431s
```

As an aside, this query suggests that there are some data quality issues. After excluding the documented missing value code (-9999), I wouldn't expect to see any precip values <0 (tenths of mm) or temperature value <-900 (tenths of degrees C).

#8 - 05/22/2012 08:25 AM - Adam Wilson

Perhaps it would make sense to keep the data in the "one-row-per-month" format in the database and transform it on the fly in the query? This would reduce the number of rows in the table by 1/30th...

#9 - 05/22/2012 10:14 AM - Jim Regetz

If useful for some specific purpose, we could generate relevant synoptic views or subsetted table dumps pivoted back into a more space-efficient format. However, I do think it's worth retaining our core database table in the long form, as this is a more natural representation in an RDBMS. I'm not actually worried about table size per se. I mostly mentioned it to point out that the SQLite file is large and not very portable, which IMHO removes the main advantage of using it over PostgreSQL. Storage capacity is not an issue, nor is performance. For example, this query returned in <0.5 seconds on the postgres database:

```
SELECT month, MIN(value), ROUND(AVG(value), 1) AS mean,
       MAX(value), ROUND(STDDEV(value), 1) AS sd, COUNT(*) AS n
    FROM ghcn
    WHERE element='TMAX'
```

```
        AND value>0
        AND qflag IS NULL
        AND station='USC00091665'
        GROUP BY month
        ORDER BY month;
```

```
 month | min  | mean  | max |  sd  |  n
-------+------+-------+-----+------+------
     1 | -178 | 109.9 | 261 | 61.5 | 2061
     2 |  -83 | 134.3 | 283 | 62.4 | 1903
     3 |  -33 | 181.8 | 306 | 57.1 | 2099
     4 |   28 | 234.8 | 339 | 47.6 | 2027
     5 |  106 | 273.3 | 378 | 38.2 | 2147
     6 |  167 | 308.0 | 406 | 32.4 | 2043
     7 |  189 | 321.2 | 411 | 28.4 | 2224
     8 |  178 | 317.1 | 422 | 28.8 | 2218
     9 |  133 | 287.4 | 411 | 39.0 | 2114
    10 |   33 | 234.5 | 378 | 43.8 | 2175
    11 |  -22 | 174.9 | 306 | 51.6 | 2080
    12 | -122 | 124.7 | 278 | 56.1 | 2110
```

The filtering and aggregation aspects of the SQL statement above are typical of what we'd like to be able to do, and they can be applied more nimbly to yield results like the above when applied to data in the 'long' rather than 'wide' form.

We could also knock out an appreciable number of rows by eliminating -9999 values altogether, and for that matter records with undesirable qflags. But so far I've kept them in to be conservative, and prefer to keep it that way as long as the overhead on-the-fly filtering isn't a problem.

On an unrelated note, before closing out this ticket I'd like to load in the stations table (with geographic coordinates, etc) in order to enable queries on the basis of that information as well.

#10 - 05/25/2012 03:21 PM - Jim Regetz

The daily data files currently in the database were obtained back in Oct 2010. Benoit and I discussed simply augmenting this with more recent temp/precip records downloaded via the yearly CSVs provided by GHCN, but it looks like >5000 new stations have been added to GHCN in the intervening time, along with a number of other changes. Let's just grab the current version of the complete GHCN-Daily dataset and re-do the entire database load, which will ensure up-to-date holdings and also give us full historical records for the newly added stations.

The data are accessible here:
ftp://ftp.ncdc.noaa.gov/pub/data/ghcn/daily/

#11 - 05/30/2012 03:38 PM - Jim Regetz

I wrote and ran a little bash script to download the latest GHCN data and generate a basic log file for future reference. The script is on the jr/ghcndb branch at source:/climate/procedures/download-ghcn-daily.sh@fb35e17b, and will be merged into master in due course.

The newly downloaded GHCN data and metadata files, along with a log file generated by the download script, are now all on atlas at:
/home/layers/data/climate/ghcn/v2.92-upd-2012052822/

PostgreSQL database (re)loading is underway.

#12 - 06/01/2012 09:57 AM - Jim Regetz

Finished loading the new TMAX, TMIN, and PRCP data (just under 12 hours) and creating the station index (just over 13 hours). I'm not going to bother with the other two indexes I did before until we determine that we actually need them. The new table currently stands at 1.48 billion records.

#13 - 06/05/2012 12:35 AM - Benoit Parmentier

A short R script was added to select stations data from the postgres database created by Jim. The user must provide a polygon of the study area as well as the ghncd station locations. I ran one query for year 2010 and the period 2000 to 2010. Quality flags are preserved so we can now experiment with various screening options.

The codes can be viewed here 95d521f9

#14 - 06/05/2012 11:23 AM - Benoit Parmentier

I did a 30 year query during the night for the 1980-2010 period for the Oregon study area. I just checked the results this morning. The table contains:

-248 stations (compared to 186 for year 2010)
-2,026,222 rows (compared to 67,053 rows for year 2010)

Note that the spatial query gives a total 1093 stations within Oregon for the entire GHCND record.

We can use the 30 year table to calculate the 30 year climatology for every month/day and stations.

#15 - 07/03/2012 09:45 AM - Benoit Parmentier

There is new update to the code for the extraction of data from the POSTGRES database:
b8bf4042.

The code now allows for a choice of projection for the output study area and saves results in both text file and shape file formats. This is a general code that can be used to query: tmin, tmax or precipitation.