370-F10

|  |  |
| --- | --- |
| **Subject:** | Re: [postgis-users] PostGIS data in the marine environment |
| **Date:** | Tue, 1 Dec 2015 21:11:13 +0000 (UTC) |
| **From:** | Brent Wood <pcreso@yahoo.com> |
| **Reply-To:** | Brent Wood <pcreso@yahoo.com> |
| **To:** | PostGIS Users Discussion <postgis-users@lists.osgeo.org>, terry.curran@shaw.ca <terry.curran@shaw.ca> |

Hi Terry,

We at NIWA make extensive use of Postgis for marine data. We have some 600,000,000 readings in a database storing underway readings on a research vessel that uses this approach, and we also have a CTD database, with observations at standard depths, doing much the same as you require,

I think you need to "normalise" your data model (don't just replicate the CSV structure as this creates large amounts of redundant data), and I don't think your proposed linestring approach will work - as you can only associate one value per attribute (eg: temperature) with the entire linestring - not with each of the vertices of the linestring.

There are two common approaches to storing observation data like this (simplified, hopefully not over simplified):

Use a Postgis column for the point location in a table describing each "station" - where a station = a deployment.

This might have attributes such as max\_gear\_depth, bottom depth, a trip or survey id, a station id, a start time, finish time, at least a start position (postgis point) and perhaps an end position (Postgis), etc. This table describes the deployment - but does NOT contain the instrument data that was captured

You then have a readings table, and here there are two approaches - one is very fixed around your current instrument setup, the other is very generic & more flexible about adding new types of observations.

Both have a station ID as a foreign key, linking each observation record to the station it was recoded at.

The fixed approach has a column (attribute) for each reading, so is much like a CSV. A new type of reading requires a change in database structure - add a new column to store the new values.

The more normalised approach has at least two tables for the obs data, one contains the columns:

record\_id, station\_id, instrument\_id, value (etc - ie: readings)

the other contains id, depth (or timestamp), name, description, (etc - ie: instruments/sensors)).

This structure allows you to add a new record in the second table to describe a new instrument (or reading to be captured), then you can record readings from that instrument in the same table, with the record referencing the instrument in the second table.

Assuming you use the second approach, then:

select s.trip\_code, s.station\_no, s.start\_time, ST\_X(s.statr\_point) as lon, ST\_Y(s,.start\_point) as lat, i.name, r.depth, r.value

from station s, instruments i, readings r

where s.id = r.station\_id

and i.id  = r.instrument\_id

and trip\_code='trip\_no\_1'

and station\_id=1

and i.name='conductivity'

order by depth;

will give you the conductivity readings & other station values for the first station in the first trip, with your X & Y's as coord values.

Similarly:

select s.trip\_code, s.station\_no, s.start\_time, s.start\_point, ST\_X(s.statr\_point) as lon, ST\_Y(s,.start\_point) as lat, i.name, r.depth, r.value

from station s, instruments i, readings r

where s.id = r.station\_id

and i.id  = r.instrument\_id

and trip\_code='trip\_no\_1'

and depth = 50

and i.name='conductivity';

will give you all the 50m conductivity values for a trip,

Data modelling & normalisation (very simplistically) is a process to deign a database - to define the real world entities you are describing in the db, determine the attributes you are recording for each, and the relationships between entities. There should be no values stored more than once, with shared values linked rather than duplicated.

Note that Geonetwork is a metadata catalogue - & I figure you would describe datasets here - so you could have one record describing your entire database for all sonde data, or you could have one record describing each survey.

Also - Geoserver really only works with 2D data - so would work well with your station point locations, but simply loads lots of points in the same place for records at the various depths for each station. You could use a query like the one above to create sets of points at each depth - and use these as "layers" served by Geoserver.

I suggest you look at a couple of other tools which may help with your data - if you turn each depth/survey into a coverage, RASDAMAN could be useful to store them as raster datasets, and if you want to manage your observations in a genuine time series based system, look at 52N, which supports the OGC SOS (Sensor Observation Service) standard for observation & time series data, much like Geoserver supports the OGC WMS/WFS standards & Geonetwork supports their CSW service.

Cheers,

Brent Wood

**From:** Terry <terry.curran@shaw.ca>  
**To:** postgis-users@lists.osgeo.org   
**Sent:** Wednesday, December 2, 2015 9:02 AM  
**Subject:** [postgis-users] PostGIS data in the marine environment

We have been investigating how to get our marine data into and out of our PostGIS - GeoServer - GeoNetwork on a CentOS system, and we are looking for some advice.

We often have the situation of marine data obtained from a sonde lowered vertically through the water column. The resulting data consists of a lat-long location, datetime, and sensors typically measuring water temperature, conductivity, dissolved oxygen and chlorophyll fluorescence as a function of pressure.

We have typically created CSV files and entered them into PostGIS. We create a spatial index based upon the 2D lat-long position.  This approach does not seem to maximally use the "relatedness" of the cast information.

We are considering creating for each horizontal location and datetime a series of linestrings of (pressure, parameter) tuples, and then reading these into PostGIS.  The resulting table records could then be up to a thousand times shorter, and searches potentially faster.This may imply that we need to write an application to form these linestrings, and another to create SQL commands.

Alternatively, we are looking for a nice way to create NetCDF data from our profile data, and then input that into PostGIS.

Any suggestions on contemplated approaches, or advice or links to prior technology for a similar situation?

-- terry