IN43A-1394
2010 Fall AGU Meeting

STOQS: The Spatial Temporal Oceanographic Query System

McCann, Michael P (mccann@mbari.org), Richard Schramm (rich@mbari.org)
Monterey Bay Aquarium Research Institute, 7700 Sandholdt Road, Moss Landing, CA 95039-9644 United States

Abstract

The Spatial-Temporal Oceanographic Query System (STOQS) has been developed at the Monterey Bay Aquarium Research Institute to improve access and visualization of a multi-decadal archive of upper water column observations. STOQS consists of a set of applications, operational procedures, and a geospatial relational database. Borrowing a database schema from the Geographic Information System community we've implemented a database that is tuned for efficient queries across several dimensions of the data model. An Object Relational Mapping (ORM) tool was used to hide the complexity of SQL that results from our highly normalized data model. The Python scripting language is used to write the Extract Translate Load (ETL) programs for populating the database with data from our long-term operational archives. These archives include collections of Climate Forecast convention netCDF files of mooring and autonomous underwater vehicle data and other special purpose relational databases. This poster describes the specific tools and techniques used to implement STOQS. Though still in development the system already provides benefits to users through a Google Earth interface and an ability to conduct fast queries across multiple previously non-interoperable data sets.

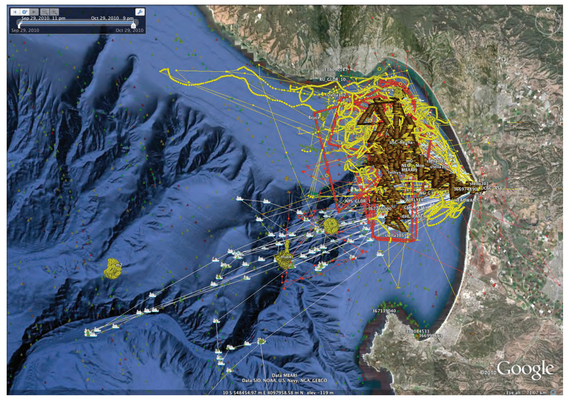

Figure 1. Screen shot from Google Earth showing dozens of platform tracks.from the October 2010 BloomEx study in Monterey Bay.

NetCDF good for for getting data along coordinate dimensions

Relational/Geospatial database good for querying along any “dimension”

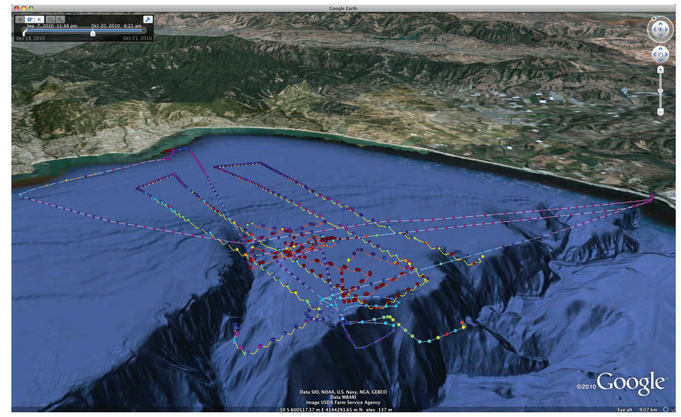

Figure 2. Screen shot from Google Earth showing Chlorophyll fluorescence observations from three different platforms. Just the upper 10 meters of observations are color coded with red indicating high values.

Python scripts load and get data using object notation

Entity Relationship Diagram

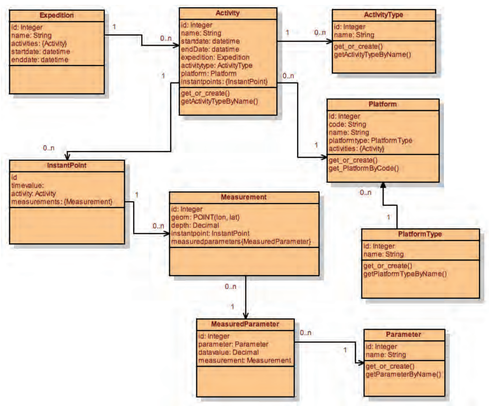

Diagram1.. Data model for the STOQS database. This model derives from the Arc Marine data model as described in Wright *et al*. All data values are stored in the MeasuredParameter table. The Parameter table contains a row for each kind of thing (e.g. Temperature, Chlorophyll Fluorescence) that we measure.

Code to query database

```
def xySlice(parm, upper = 0, lower = 10000, start = '1900-01-01 00:00:00',
    end = '3000-12-31 23:59:59'):
  '''Return data from STOQS on a horizontal slice restricted as:
       lower < depth < upper
       start < timevalue < end
       parameter == parm
  A list of (time, lon, lat, depth, parm, datavalue, platform) tuples is returned.
  Default input values basically allow unbounded limits on depth and time.
  '''
  dataList = []
  for (mp, pl) in session.query(MeasuredParameter, Platform).join('measurements',
    'instantpoints', 'activity', 'platform') .join('parameter').filter(
      and_(
       Measurement.depth >= upper, Measurement.depth <= lower
       , InstantPoint.timevalue > start, InstantPoint.timevalue < end
       , Parameter.name == parm
      )).order_by(InstantPoint.timevalue):

  dataList.append((mp.measurement.instantpoint.timevalue, mp.measurement.lon,
   mp.measurement.lat, mp.measurement.depth, mp.parameter.name, mp.datavalue, pl.name))

  return dataList
```

Listing 1.. Python code written with the SQLAlchemy Object Relational Mapper interface to the STOQS database. Table joins and constraint expressions are expressed using simple object notation. This method returns a list of the selected parameter within specified depth and time bounds.

Data Flow Diagram

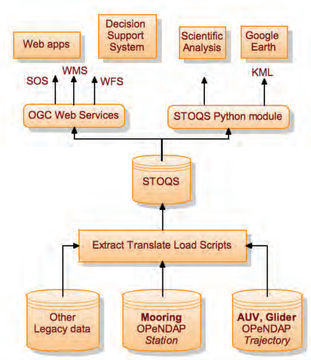

Diagram2.. Data flow from archive data stores at the bottom of the diagram through ETL scripts to the geospatial database STOQS. The STOQS database implements the data model shown in Diagram 1.

Code to build KML

```
def buildKMLpoints(plat, data, clt, clim):
  '''Build KML Placemarks of all the point data in `list` and use colored styles
  `data` are the results of a query, say from xySlice()
  `clt` is a Color Lookup Table equivalent to a jetplus clt as used in Matlab
  `clim` is a 2 element list equivalent to clim in Matlab
  Return strings of style and point KML that can be included in a master KML file.
  '''

  styleKml = ''
  for c in clt:
    ge_color = "ff%02x%02x%02x" % ((round(c[2] * 255), round(c[1] * 255), round(c[0] * 255)))
    style = '''<Style id="%s"><IconStyle><color>%s</color><scale>0.6</scale><Icon>
     <href>http://maps.google.com/mapfiles/kml/shapes/dot.png</href></Icon></IconStyle></Style>
     ''' % (ge_color, ge_color)
    styleKml += style

  pointKml = ''
  for row in data:
    (dt, lon, lat, depth, parm, datavalue, platform) = row
    coordStr = "%.6f, %.6f,-%.1f" % (lon, lat, depth)
    clt_index = int(round((float(datavalue) - clim[0]) * ((len(clt) - 1) / float(numpy.diff(clim)))))
    if clt_index < 0: clt_index = 0;
    if clt_index > (len(clt) - 1): clt_index = int(len(clt) - 1);
    ge_color_val = "ff%02x%02x%02x" % ((round(clt[clt_index][2] * 255), round(clt[clt_index][1] * 255),
     round(clt[clt_index][0] * 255)));
    placemark = """<Placemark><styleUrl>#%s</styleUrl><TimeStamp><when>%s</when>
     </TimeStamp><Point><altitudeMode>absolute</altitudeMode><coordinates>%s</coordinates></Point>
     </Placemark> """   % (ge_color_val, time.strftime("%Y-%m-%dT%H:%M:%SZ", dt.timetuple()), coordStr)
    pointKml += placemark

  return (styleKml, pointKml)
```

Listing 2. Python code to create Keyhole Markup Language text with dots colored according to value of the parameter. It takes as input the list that is produced by the code in listing 1. This code was used to produce the visualization of Chlorophyll measurements shown in figure 2.

References

- Arc Marine data model: Wright, D.J., Blongewicz, M.J., Halpin, P.N. and Breman, J., 2007. Arc Marine: GIS for a Blue Planet, Redlands, CA: ESRI Press, 202 pp. ISBN 978-1-58948-017-9.http://dusk.geo.orst.edu/djl/arcgis/book.html
- PyDAP: Python module for reading from OPeNDAP servers. http://www.pydap.org
- Django: Python module for Object Relational Mapping. http://www.djangoproject.com
- Postgres: Open source relational database. http://www.postgresql.org
- PostGIS: Relational database with geospatial support. http://postgis.refractions.net
- KML: Keyhole Markup Language. http://code.google.com/apis/kml/

Acknowledgements

This research is supported by the Monterey Bay Aquarium Research Institute through funding from the David and Lucile Packard Foundation.

Presented at: The Future Directions for Earth Science Data AccessTechnologies poster session at the American Geophysical Union meeting in San Francisco, CA on 16 December 2010

Prepared by: Mike McCann on 6 December 2010