OBIS Portal
Development & Architecture

Ei Fujioka
Marine Geospatial Ecology Lab
Nicholas School, Duke University

A DECADE OF DISCOVERY
CENSUS OF MARINE LIFE 2010
Amazon Cloud

Internet

DiGIR

Web site
Simón Bolívar

Search Interface
Duke Univ.

Amazon Cloud
A replicated set for development

Database
Rutgers Univ.

DiGIR or File transfer

DiGIR

Data providers

Amazon Cloud
(R&D environment supported via Amazon grant to Duke)

Ongoing R&D
Duke Univ.

Team members also data providers:
• OBIS-SEAMAP, Duke University
• Caribbean NRIC, Simón Bolívar
Ideas of Future Infrastructure

Internet (Asia and others?)

INCOIS Node

Data providers

Replica DB

Mirror

INCOIS Node

Production Search Interface & Drupal

Production Database

Service DB

Replica Database (App dev)

DiGIR/OGC/REST (non-interactive services)

R&D

Amazon Cloud

Code repository At MGEL

Search Interface Development

Duke Simón Bolívar

VLIZ Node

Rutgers

Production Database (Data dev)

Master Database (Data dev)
Responsibility Categories

**Coordination**
Oversee activities, coordinate collaboration and make plans

**System maintenance**
- Daily machine maintenance
  - Monitoring (performance, network, disk space etc.)
  - Patches / Trouble-shooting

**Database maintenance**
- Data updates / daily operations
  - Data registration (DiGIR crawls, etc.)
  - Communication with providers

**Content management**
- Contents updates on Drupal site
  - News
  - Translation

**Customer services**
- Respond to user questions
  - Q & A
  - Help for usages
  - Help for advanced queries

**Application maintenance**
- Daily operations
  - Bug fixes / Trouble-shooting
  - Small improvements

**Collaborative researches**
- Research & analysis using OBIS
  - Grant proposals
  - Conduct researches and developments

**Application development**
- Major upgrades
  - New features / services
  - New versions

**Partner care**
- Collaboration with partners
  - Service change notice
  - Build and maintain custom services (e.g. REST, OGC)

**Infra improvements**
- Major H/W, S/W upgrades
  - New features
  - New versions

**Content management**
- Database maintenance
  - System maintenance
  - Application maintenance
  - Customer services
  - Coordination
  - Collaborative researches
  - Application development
  - Partner care
  - Infra improvements
Data Access

Search Interface

OpenLayers
EXTJS stores Javascript

WMS requests
WMS images

Mapping Engine
GeoServer
OBIS Store

PHP

AJAX calls
JSON

ADODB

Database

PostGIS
PostgreSQL
Database Schema – Basic ideas

Raw data (DiGIR table)

Normalized tables
- Taxon info
- Positions
- Resources
- Additional attributes

Calculated point data
- Denormalized for better performance
- Oceanographic data using WOA
- Keys to external layers such as EEZ

Summarized tables
- Per species
- Per species and resource
- Gridded using Csquare
- 5/1/0.5/0.1 degree(s)

Calculated summary data
- Denormalized for better performance
- 5/1/0.5/0.1 degree(s)

Scripted

Taxonomy
- Based on multiple taxon database
  - WoRMS
  - ITIS
  - IRMNG
  - COL

Supplemental
- EEZ
- World Ocean Atlas
- CSquare

Search Interface

GeoServer

Schema for Search Interface

Scripted
Structure Details – Server side

- GeoServer
- Database
- Browser

- Javascript / CSS files
- index.php
- portal_main.php
- download.php
- species_info.php
- dataset_info.php
- record_info.php

- OGC requests through OpenLayers
- AJAX calls

- PHP/Apache
- connections/obis.php
- advanced_panels_xx.html

- Classes/clsObisDB.php
  - Construct SQLs to extract data

- Individual information
- Download requests
- Data extraction
Structure Details – Client side

- **rs_front_page_xx.js**
- **xml**
- **prototype.js**
- **portal_objects.js**
- **front_page.js**

- **Internationalization**
- **EXTJS**
- **Handle user actions**
- **Render grids, panels etc.**
- **Define WMS layers**
- **Build CQL to WMS**
- **CSS files**
- **Styles**

- **AJAX calls**
- **GeoServer**
- **OGC requests**
- **OpenLayers GeoExt**
- **OGC features**
- **PHP/Apache**
- **Database**
GeoServer

Database

advanced_panels_xx.html
Layouts of the page and static contents
xx: language code

GeoServer

Javascript & CSS

rs_front_page_xx.js
Internationalization
xx: language code

OGC requests

front_page.js
Handle most user actions
Render grids, panels etc.
Define WMS layers

download.php
Accept download requests

portal_map.php
Accept AJAX calls

species_info.php
dataset_info.php
record_info.php

clsObisDb.php

index.php

Database
GeoServer

Database

Browser

Javascript
- prototype.js
- EXTJS
- OpenLayers
- GeoExt
- common.js
- rs_front_page_xx.js
- portal_objects.js
- map_classes.js
- front_page.js

CSS
- EXTJS
- GeoExt
- Override for EXTJS
- portal_common.css

Index.php

PHP/Apache
- portal_map.php for data extraction requests
- download.php for download requests
- classes/clsObisDb.php defines SQLs
- Connection/obis.php Connection to DB

Index.php

AJAX calls

OGC requests through OpenLayers

GeoServer

Database
Parameterized layers allow to extract data based on user inputs
- Implemented as vendor-specific parameters for OGC requests
- Add a parameter as %param_name% in the layer definition SQL
- Parameter values are added to a WMS request like:
  viewParams=param_name:value;...

Definition of Parameterized Layer

<table>
<thead>
<tr>
<th>Name</th>
<th>Default value</th>
<th>Validation regular expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>where</td>
<td>valid_id=4567890</td>
<td>[-?\d/\w\s]+$</td>
</tr>
<tr>
<td>count_column</td>
<td>nextcl</td>
<td>\d+$</td>
</tr>
<tr>
<td>table</td>
<td>dist_sp_1deg</td>
<td>\d+$</td>
</tr>
</tbody>
</table>
Development Environment (1)

Amazon Cloud
- Production Servers
  - Search Interface
  - Database

Development Servers
- Search Interface
- Database

Production version

Trunk for testing

Subversion repositories
- trunk
- Production versions

Development Server
- Search Interface
- Debugger-enabled Apache & PHP

Development Client
- PHP Dev Environment (Zend Studio)
- Firefox with Firebug
- Subversion client

MGEL at Duke

Rutgers

Database updates
Development Environment (2)

TortoiseSVN
Firefox with Firebug
Zend Studio (subversion support)
Subversion Control

Developer 1
Developer 2
Developer 3

Check out / update working copy
Commit changes

Repository

trunk
branches

tags

Final release
Version 1.0
Version 1.1
Version 2.0

Merge

New feature / Major upgrade
French translation
New graph types

Development
Production
Multi-language Support

Español
(Spanish)
Simón Bolívar

日語
(Japanese)
Duke Univ.

русский
(Russian)
Nat’l Academy of Sciences of Ukraine

French
(French)
Bruno Danis
DFO

More to come…
Multi-language Support

Drupal Standard language settings

- English
- Japanese
- Russian
- Spanish

Search Interface Javascript resource files

- English resource file (rs_front_page_en.js)

Or http://www.iobis.org/mapper/?language=en
OBIS Team at Rutgers and MGEL at Duke have been migrating the OBIS DiGIR Provider from its classic framework (old DiGIR version, old ADODB Lib, older PHP, Oracle, Java) to the new environment (latest DiGIR version, latest ADODB Lib, PHP5.3, PostgreSQL).

Classic OBIS DiGIR

Admin tools with Java

DiGIR Provider on older PHP

DiGIR table Oracle

Rutgers machine

New OBIS DiGIR

Admin tools with PHP, R

DiGIR Provider on PHP5.3

DiGIR table PostgreSQL

VLIZ or INCOIS or both

Old ADODB Lib

Latest ADODB Lib
DiGIR Configuration

- OBIS DiGIR is under subversion control.  
  https://code2.nicholas.duke.edu/svn/obis_portal/digir

- Configuration files required maintenance & updates
  - Provider information (providerMeta.xml)
  - List of datasets available (resources.xml)
  - Individual dataset metadata (*dataset_name*.xml)

- Edward’s script generates dataset metadata xml files
Performance

- Cache period
- Need to cache per client?
- A dedicated machine?
- The larger a dataset grows, the more extremely it gets slow. EurOBIS took almost 2 days to fetch the biggest dataset in OBIS-SEAMAP (1.1 million records)
  - Up to 1,000 records at a time => Need 1,100 requests
  - Every time, it has to sort 1.1 million records to return a correct set of records
  - Caches are in no use as a request is different every time.

Usability

- List of taxa included
- Resource metadata for individual resources
  Currently DiGIR returns metadata for all resources.
OGC Consideration 1

Criteria
- Allow all possible criteria (by taxon, dataset, region, time period, oceanography)?
- Implement some limitation (e.g. one taxon, rectangular region only)?

Capabilities & List of available layers
- How to deal with the fact that GeoServer layers are defined as parameterized layers.
- How to inform users of available layers and vendor-specific parameters (search criteria)

Client support
- Browser
- OGC compliant client software
- ArcGIS (not good at adding vendor-specific parameters)
- Custom-made scripts (e.g. MGET); user-specific customization?

Performance
- Data extraction and rendering image/generating XML can take a long time or end up with a time-out
- The resulting WFS XML can be huge (e.g. ~100MB)
- Caching frequent requests?

Logging
- What and where to log the requests
- How to use the log info? (what you want to know from log?)
Example of performance issue

- WFS for Humpback whale
  - 23,831 records
  - XML file: 39MB for essential columns (1MB zipped)
  - XML file: 80MB in 2 minutes 20 seconds for full OBIS schema (2.2MB zipped)
**Technical Issues (1)**

**Database**

- Change field type for latitude/longitude from ‘real’ to double or numeric
  - Real can hold up to 6 digits in total
  - Numeric would be better as it holds the number as it is provided (no 10.9999999999 for the value of 11.0000).
- `resources_actors` does not have all contacts. Instead, it has duplicate entries for a person. Check the following SQL.
  ```sql
  select *
  from obis.resources_actors
  JOIN obis.actors ON resources_actors.actor_id = actors.id
  LEFT JOIN obis.roles ON resources_actors.role_id = roles.id
  where resource_id = 130
  ```
- `obis.roles` and `obis.resources_actors`: role_id in resources_actors are mostly empty.
- Need to figure out how to label types of a record and meaning of the count:
  - Avoid overestimate of abundance for telemetry/acoustic data (or OBIS does not collect acoustic data??)
- How to avoid duplication of a dataset (e.g. direct submission from a provider, upload from OBIS-USA or OBIS-SEAMAP for the same dataset)
  - Potentially it can be tripled if OBIS-USA harvests from OBIS-SEAMAP and uploads it to OBIS.
Portal Schema

- How to build/maintain the portal schema?
  - Originally Ei wrote a bunch of SQLs to build the schema and put them into Subversion.
  - Edward modified them. Ei also updated them. \( \Rightarrow \) inconsistency.
  - I’ve found that an index on taxon_tree in portal.species_summary is missing due to this inconsistency. (My script creates it but the one in the DB does not)
Metadata

- How to distinguish original providers/collectors and data portals such as OBIS-SEAMAP?
- More flexible ways to list providers as well as credits or acknowledgements to funding sources, collaborators etc.

Links to OBIS from partners

- What page to let them link to? All go to the Search Interface with a taxon id?
  - Possibility:
    - Search Interface,
    - Static species info page (http://iobis.org/mapper/species_info.php?id=609834)
    - WMS images (on their web site)
    - Distribution by Taxon on the Drupal site (?)
- How to notify them that the URLs have changed?
  - e.g. ITIS still links to the classic Portal URL. http://www.iobis.org/IndexSearch?sciname=Hyperoodontidae&sciButton=Scientific+name&category=all&names=all
Multi-resolution distribution maps

Humpback whale as an example
Higher taxon with/without branches

Annelida as an example; By default, the distribution map for a higher taxon includes observations for branch taxa.
Extracting data with region layers (e.g. EEZ, LME)

Humpback whale observations in Bahamas EEZ
Browsing taxonomic tree

Selecting a taxon brings up taxon information to the right panel.
Live search for species name

- Search by scientific or common name
- Matches with 'begin with' or 'contains'
- By selecting a taxon from candidate list, you are taken to the corresponding taxon tree node.
Multiple view of the search results

Individual records

List of taxa extracted

List of contributing datasets

- List of taxa and contributing datasets can be saved as a csv file
- Individual records are downloadable ([Download] tab)
Some Cool Features

Advanced graphing – Time series plots for oceanographic variables

- Zoom in/out at various temporal scales (century/decade/year/month/day; four seasons/12 months)
- Can hide empty periods for a compact view
- Supported variables: temperature, salinity, nitrate, oxygen, phosphate, silicate, sample depth
Some Cool Features

Advanced graphing – Histogram for oceanographic variables

Histogram of temperature
(5-degree bin)

2-degree bin

- Zoom in/out at various bin sizes (bin sizes can vary variable by variable)
- Graph data can be saved
Advanced graphing – Time series plots of #records

Zoom in/out at various temporal scales (century/decade/year/month/day; four seasons/12 months)
Can hide empty periods for a compact view
### Technical Details

#### Major tables

<table>
<thead>
<tr>
<th>Individual records</th>
<th>Species summary</th>
<th>Gridded distribution</th>
<th>Diversity indices</th>
</tr>
</thead>
<tbody>
<tr>
<td>Record ID</td>
<td>Taxon ID</td>
<td>CSquare code</td>
<td>CSquare code</td>
</tr>
<tr>
<td>Taxon ID</td>
<td>Taxon info</td>
<td>Taxon ID</td>
<td>Shannon</td>
</tr>
<tr>
<td>Taxon info*</td>
<td>Taxon tree</td>
<td>Taxon tree</td>
<td>Simpson</td>
</tr>
<tr>
<td>Resource ID</td>
<td>Statistics</td>
<td>#records w/ branches</td>
<td>ES(50)</td>
</tr>
<tr>
<td>Taxon ID</td>
<td>(#records,</td>
<td>#records w/o branches</td>
<td>#records</td>
</tr>
<tr>
<td>Taxon tree</td>
<td>extent, WOA</td>
<td>Geometry</td>
<td>#Species</td>
</tr>
<tr>
<td>Geometry</td>
<td>etc.)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WOA values</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- * indicates partial information (denormalized for better performance; avoid JOIN)
- Region layers such as EEZ and LME are imported into PostGIS tables
- Resolutions are 10 / 5 / 1 /0.5 /0.1 degree(s); 10 degrees are not used online
- World Ocean Atlas data are imported and associated with CSquare codes.
Example SQLs

Extracting records for a taxon with its branch taxa

```sql
SELECT
    id, sname, sauthor, tname, tauthor, resource_id, resname, to_char(datecollected, 'YYYY-MM-DD') as datecollected,
    latitude, longitude, lifestage, basisofrecord, datelastcached, dateprecision, datelastmodified, depth, depthprecision, temperature, salinity, nitrate, oxygen, phosphate, silicate
FROM portal.drs_with_woa
WHERE
    (valid_id in (458788))
    or
    UPPER(storedpath) LIKE '%739909|738303|741923|762719|766931|766932|642142 |778875|781762|778804| 696387|753566|769769|458788|%'
```

Kogia (ID: 458788) as an example
This SQL returns records for Kogia as well as Kogia breviceps (ID: 458789) and Kogia sima (ID: 458790). Three of them share the same parent storedpath.

CREATE INDEX idx_drs_with_woa_storedpath
ON portal.drs_with_woa
USING btree
(upper(storedpath::text));

`storedpath` column is indexed

- Taxon tree information is built as a series of parent taxon IDs with a delimiter.
- To extract records of branch taxa, search for those with the same taxon tree.
- UPPER(storedpath) LIKE `(taxon tree)` uses an index and reasonably fast.
Example SQLs

**Extracting data for time series chart (over years)**

```sql
DROP TABLE IF EXISTS graph_table CASCADE;

CREATE TEMP TABLE graph_table AS
SELECT date_trunc('year', datecollected)::date as x_value, count(*) as y_value
FROM portal.drs_with_woa
WHERE
  (valid_id in (698592)
or
  UPPER(storedpath) LIKE '^[739909|738303|741923|762719|766931|766932|642142|778875|781762 |778804|
  696387|752492|739487|469012|698592]%')
and
datecollected is not null
GROUP BY date_trunc('year', datecollected)::date
ORDER BY date_trunc('year', datecollected)::date;

CREATE INDEX idx_graph_table_x_value ON graph_table USING btree (x_value);

SELECT generate_series as x_value, y_value
FROM (SELECT generate_series (min, max, '1 year')::date as generate_series
FROM (SELECT min(x_value) as min, max(x_value) as max
FROM graph_table ) foo ) hoo
LEFT JOIN graph_table ON hoo.generate_series = graph_table.x_value;
```

- This set of SQLs is run on the fly when you switch to [Graph] tab.
- Use a temporary table to pre-calculation.
- Generate time series sequence (generate_series in PGSQL 8.4) to fill the empty periods.
Example SQLs

Extracting data for histogram of temperature

```sql
DROP TABLE IF EXISTS graph_table CASCADE;

CREATE TEMP TABLE graph_table AS
SELECT (floor(temperature \(\mod 5\)) \* 5)::int as x_value, count(*) as y_value
FROM portal.drs_with_woa
WHERE (temperature is not null
and (valid_id in (698592) or UPPER(storedpath) LIKE '^[739909|738303|741923|762719|766931|766932|642142|778875|781762|778804|696387|752492|739487|469012|698592|\%$')
GROUP BY floor(temperature \(\mod 5\)) \* 5
ORDER BY floor(temperature \(\mod 5\)) \* 5;

CREATE INDEX idx_graph_table_x_value ON graph_table USING btree (x_value);

SELECT generate_series as x_value, y_value
FROM (
SELECT generate_series(min, max, 5) as generate_series
FROM (
SELECT min(x_value) as min, max(x_value) as max
FROM graph_table ) foo ) hoo
LEFT JOIN graph_table ON hoo.generate_series = graph_table.x_value;
```

- **Same idea as time series data with different calculation.**
- **Generate bin sequence (generate_series in PGSQL 8.4) to fill the empty bins.**
EXTJS 3.x uses Yahoo graph component (flash object)

Extracting data for histogram of temperature

EXTJS 3.1.1 uses the slightly old version of the graph component.
To make possible to add secondary axis….

- Same idea as time series data with different calculation.
- Generate bin sequence (generate_series in PGSQL 8.4) to fill the empty bins.
## Drupal vs. Plone

<table>
<thead>
<tr>
<th></th>
<th>Drupal</th>
<th>Plone</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overall performance</td>
<td>Light</td>
<td>Heavy</td>
</tr>
<tr>
<td>Page load</td>
<td>Load fast generally</td>
<td>Load slow generally</td>
</tr>
<tr>
<td>Backend server</td>
<td>Apache</td>
<td>Zope</td>
</tr>
<tr>
<td>Good things about…</td>
<td>Multi-language support</td>
<td>Flexible account mgmt.</td>
</tr>
<tr>
<td></td>
<td>Rapid learning curve</td>
<td>Powerful workflow</td>
</tr>
<tr>
<td>Programming</td>
<td>PHP – Drupal components</td>
<td>Plone-native template/DTML etc. Python</td>
</tr>
<tr>
<td></td>
<td>Haven’t done much</td>
<td>Painful to debug!!</td>
</tr>
</tbody>
</table>

- As for content management, it seems Drupal is simpler than Plone. Haven’t done much with Drupal, so can’t say for sure but Plone tends to be frustratingly complex when it comes to customization.
- Unless you have clear, uncompromisable reasons to use Plone (e.g. implement scientific workflow using Plone’s workflow framework), Drupal would be sufficient for most purposes and usages.