Designing Middleware to Facilitate the Analysis of Disparate Environmental Datasets

By Wesley Leonard

Thesis Defense

2008/10/13

Committee Members

Dr. Paul Albee

Dr. Michael Stinson

Dr. Tracy Galarowicz

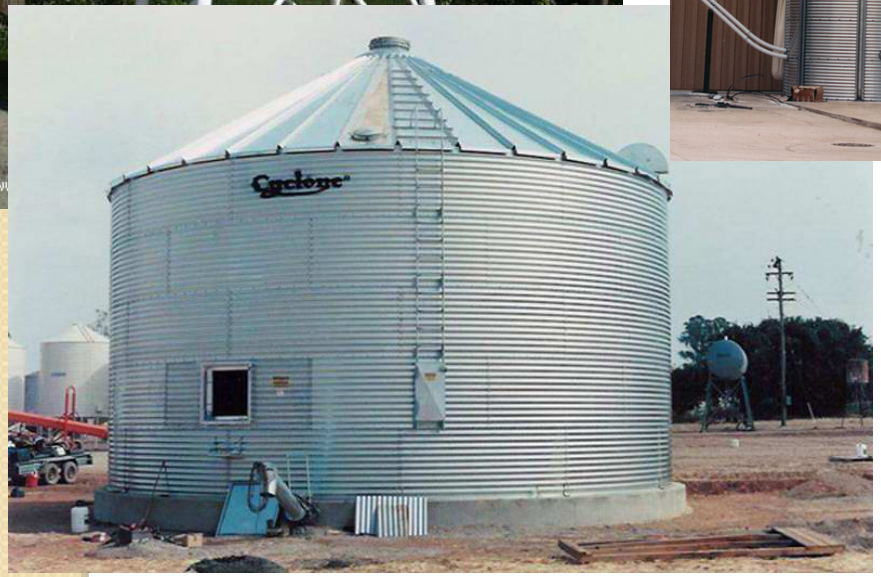
Guest:

Mr. Thomas Rohrer

The "Data Glut" Problem

- The amount of scientific data collected and stored is vast and ever increasing
- Traditional analysis methods infeasible
- Computer-based tools are needed to organize and analyze large datasets

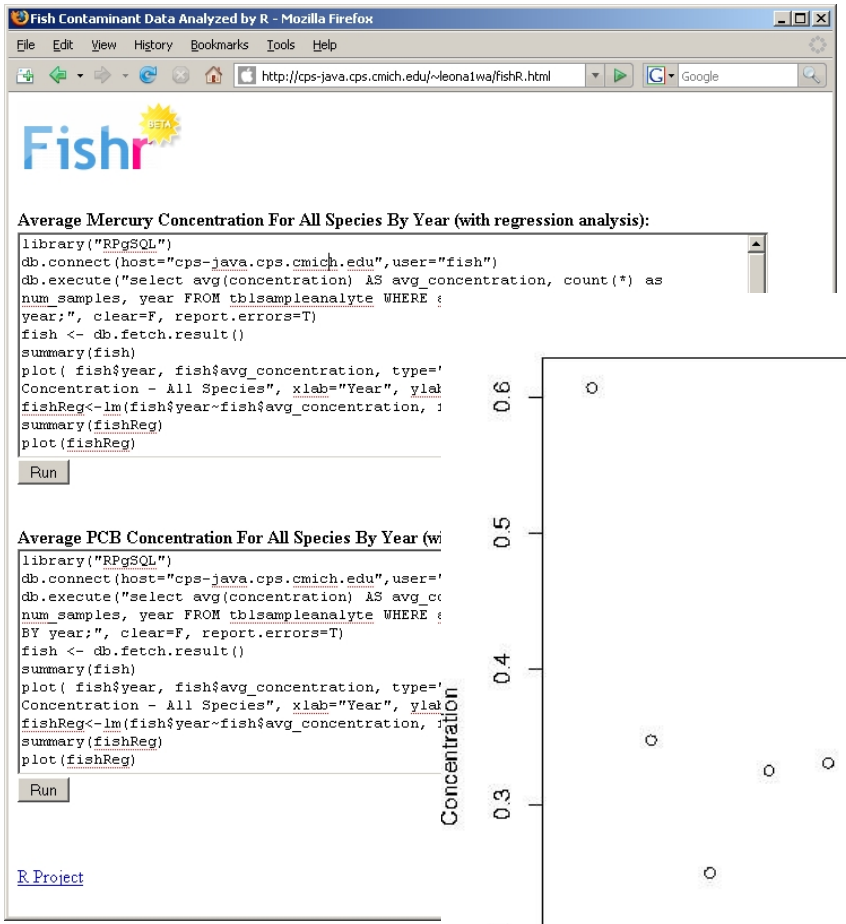
Isolated Data Storage

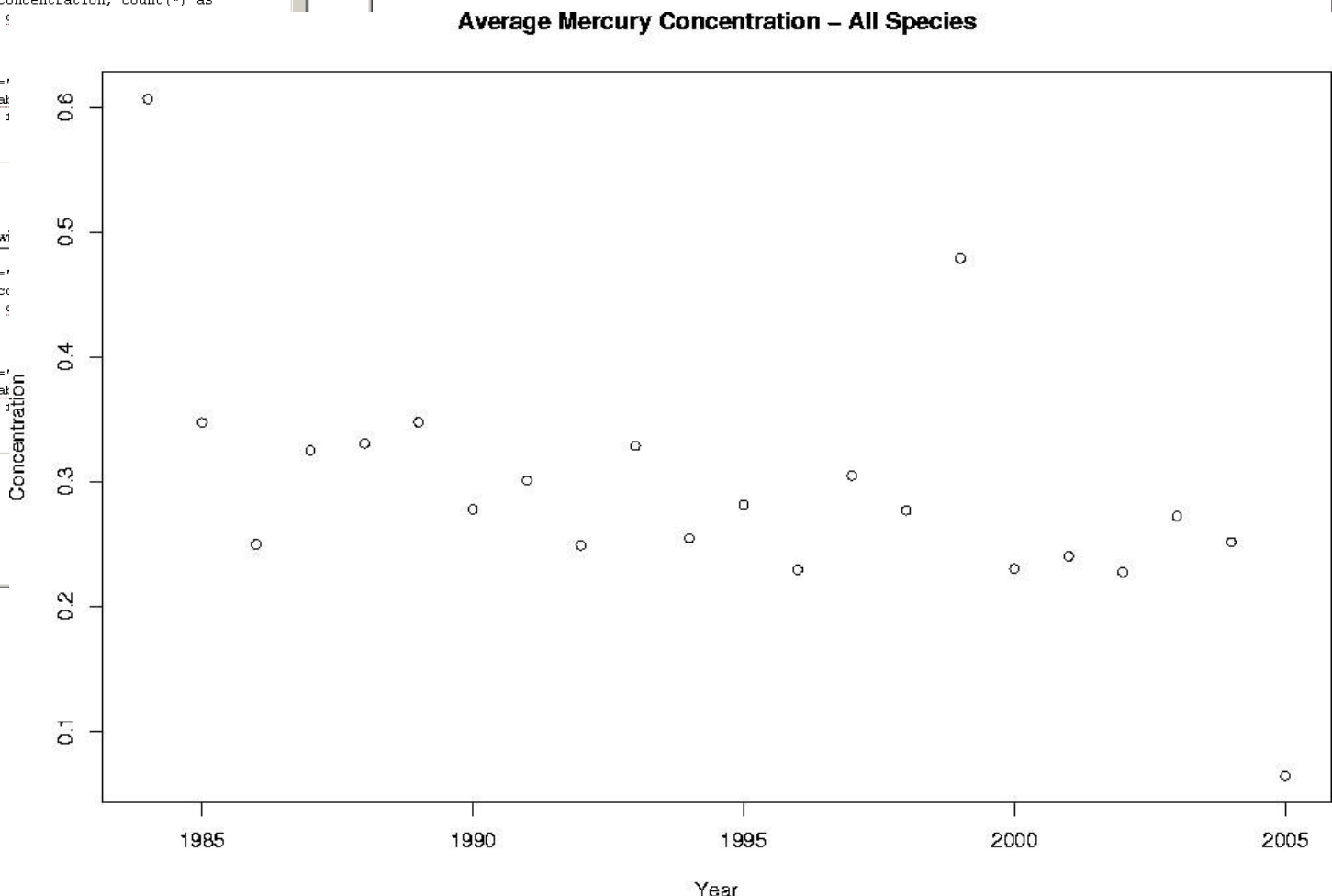


Michigan DEQ Data

- Dataset of fish contaminants
- Started collecting around 1980
- Details:
  - 29,033 samples
  - 28,586 fish
  - 68 different species of fish
  - 326 types of contaminants examined

Earlier Work with DEQ Data





FishR

- Provided web-based access to fish contaminant data
- Provided analysis toolset with graphical output (R)
- Limited to this one dataset
- Requires knowledge of R to perform analysis

Proposed Solution

- Develop a tool that can easily combine different datasets for analysis
- Further requirements:
  - Export data to commonly-used analysis tools
  - Retrieve data from remote source databases
  - Combine data from different databases in a linear fashion or by joining tables

Literature Review – Similar Work

- System developed by Ikoma et. al. is similar to proposed solution:
  - Design to integrate about 900GB if data from various environmental and climatological sources
  - Provide users with a more friendly interface to access and analyze data

Ikoma Data Mining Application

- Specific to analysis of particular data sources
- Focused on user interface, analysis & visualization tools
- Web-based interface

Other Literature

- Research areas:
  - Data mining and analysis
  - Data warehousing
  - Middleware
  - Web services and XML
  - Ruby on Rails

Literature on Data Mining & Analysis

- Definition: The process of sorting through data to pick out new, valuable, and non-trivial information
- Selection, preparation, and pre-processing of raw datasets is a necessary step
  - Particularly with scientific data

Literature on Data Warehousing

- Definition: A read-only database that creates a single logical view of data
- Primary purpose is analysis
- Separate from operational systems
- A variety of techniques and implementations used
- Metadata makes warehoused data useful
  - Dictionary, versions, statistics, aliases, security

Literature on Middleware

- Definition: Software layer that homogenizes an infrastructure
- Provides general access to data
- Makes it possible to connect heterogenous databases in a single view
- Isolate application from infrastructure

Literature on Web Services & XML

- Definition: A set of technologies that enable networked and modular applications
- Built upon XML standards
- XML is a self-describing semi-structured data format
- Provide a framework for interoperability between solutions
- "The Semantic Web"

Literature on Ruby on Rails

- Definition: "An open-source web framework that's optimized for programmer happiness and sustainable productivity" (www.rubyonrails.org)
- Rapid development
- Favors convention over configuration
- Model – View – Controller
- All object-oriented

Thesis Statement

A middleware system to facilitate the exploration, combination, organization, conversion, and analysis of data sets will be designed and a prototype will be built. The design and implementation of this system will be based on current research in the areas of data mining, data warehousing, middleware, and information sharing through web services. Research into these topics revealed the following criteria which will be used to gauge the effectiveness of the tool and its ability to contribute significantly to contemporary research:

The Seven Criteria

- Provide controls and management tools to access and view data
- Support a wide variety of operating environments
- Support a wide range of data sources
- Support the propagation of both incremental changes and full extracts
- Support source data transformations
- Support data cleansing (removing inconsistent records from the data set)
- Support documented SQL and other open interfaces for third-party and user-defined code

System Components

- Relational database
- Management interface
- Middleware layer
- Web service interface

Relational Database

- Store configuration information
- Store metadata
- Cached Data
- PostgrSQL chosen
  - Open source db engine
  - Free
  - Rich feature set
- Ruby on Rails (model)

Management Interface

- Web-based access to configuration
- Manage datasets
- Manage exports
- Manage transformations
- Manage user access
- Ruby on Rails (view)

Middleware

- Underlying functionality for management interface and web service interface
- Handles database connectivity
- Authenticate users
- Manage configuration information
- Manage cached data
- Apply data transforms
- Ruby on Rails (controller)

Web Service Interface

- Provide access to data in XML format
- Each export has a unique URI
- Exports may be from one or more datasets
- Data transforms and joins applied during export
- Ruby on Rails (view)

Data Caching

- Selected fields from data sources replicated in cache tables
- Transforms applied before caching
  - Improve response time
  - Save processing
- Caching required to accomplish joins between databases
- Side effect: improved performance

Joining Tables

- Major contribution of application
- Tables from different databases can be joined for export
- Interesting fields from remote datasets cached and joins performed locally
- Middleware layer processes, handles cache
- Management interface configures
- Web Service interface delivers results

Terms

- Dataset
- Export
- Exported Field
- Transformations (or Transforms)
- SQL

DM Algorithm – Part I

(1) **if** $Cache_{config} \wedge (Cache_{expired} \vee Cache_{dirty})$

(2) $\quad Cache_{update} \leftarrow true$

(3) **if** $\neg Cache_{config} \wedge J \neq \emptyset$

(4) $\quad Cache_{update} \leftarrow true$

(5) **if** $Cache_{config} \vee J \neq \emptyset$

(6) $\quad Cache_{read} \leftarrow true$

(7) $queries \leftarrow$ GENERATE_QUERIES$(export)$

(8) $T_c \leftarrow \emptyset$

Discussion of Algorithm

- Dynamic Merged cache / query
- Lines 1 – 6 determine if caching will be used and if cache needs to be updated
- Line 7 generates queries for export
  - *See sub-algorithm*

Sub-Algorithm: Generate Queries

GENERATE_QUERIES($export$)

(1) $Q \leftarrow \emptyset$

(2) $D \leftarrow$ GET_DATASETS($export$)

(3) **foreach** $d \in D$

(4) $T \leftarrow$ GET_TABLES_EXPORTED(d)

(5) **foreach** $t \in T$

(6) $q \leftarrow$"SELECT t_{fields} FROM t"

(7) **if** $Cache_{update} \wedge Cache_{type} = incremental$

(8) **if** $isset(table_{key\_field})$

(9) $max \leftarrow$ MAX($Cache_{key\_field}$)

(10) $q \leftarrow q+$ " WHERE key_field $> max$"

(11) $Q \leftarrow Q \cup q$

(12) **return** Q

DM Algorithm Continued

(9) **foreach** $q \in queries$

(10) **if** $\neg Cache_{read} \vee Cache_{update}$

(11) $results \leftarrow$ EXECUTE(q)

(12) **if** $Cache_{update}$

(13) STORE_IN_CACHE_TABLE(results)

(14) $T_c \leftarrow T_c \cup$ CACHE_TABLE_NAME(q)

(15) **else if** $\neg Cache_{read}$

(16) $result\_set \leftarrow results$

(17) **else if** $Cache_{read}$

(18) $T_c \leftarrow T_c \cup$ CACHE_TABLE_NAME(q)

Discussion of Algorithm

- Iterates through queries
- Executes queries
- Three exclusive conditions

DM Algorithm Concluded

(19) **if** $Cache_{read}$

(20) **if** $J \neq \emptyset$

(21) **foreach** $j \in J$

(22) $result\_set \leftarrow result\_set \cup j_{table1} \bowtie j_{table2}$

(23) $T_c - \{j_{table1}, j_{table2}\}$

(24) **foreach** $t \in T_c$

(25) $result\_set \leftarrow result\_set \cup \text{QUERY\_CACHE\_TABLE}(t)$

(26) $\text{RENDER\_XML}(result\_set)$

Discussion of Algorithm

- Joins performed
- Remaining cached tables read
- Complete result set rendered in XML

Discuss Implementation Details

- Server Environment
- Ruby on Rails
- ActiveRecord

Results

- Demo video

Reading Time

Total Records	Data Operation	Without Cache	With Cache
2000	none	1.332195	0.61517
2000	none	1.304521	0.611286
2000	none	1.292177	0.620324
2000	cache update	n/a	19.537353
1014	join	19.182868	0.51366
1014	join	19.233897	0.510997
1014	join	19.273681	0.509054
2000	transformation	5.562685	0.628351
2000	transformation	5.546536	0.614257
2000	transformation	5.585319	0.614521
2000	transformation	5.560045	0.613802
2000	transformation	5.567419	0.62572
2000	transformation and cache update	n/a	23.302291

Conclusions – How Criteria Met

- Provide controls and management tools to access and view data
- Support a wide variety of operating environments
- Support a wide range of data sources
- Support the propagation of both incremental changes and full extracts
- Support source data transformations
- Support data cleansing (removing inconsistent records from the data set)
- Support documented SQL and other open interfaces for third-party and user-defined code

Future Work

- The system developed could be expanded to include:
  - Export from custom SQL
  - Greater security
  - Greater cache expiration flexibility
  - Additional export options (csv, etc)
  - Additional data source options (file-based, ODBC, etc)
  - Data transforms with SQL
  - Distributed architecture