Data Management Strategies Report

for the

Cosumnes Research Group II (CRGII)

David P. Waetjen [1, 3]
Joshua H. Viers [1]
Allan Hollander [1]
Shaozhi Ye [2]
James F. Quinn [1]

1. Department of Environmental Science & Policy
2. Department of Computer Science
and
3. Geography Graduate Group
University of California, Davis

in affiliation with

Information Center for the Environment (ICE)
John Muir Institute of the Environment
Center for Watershed Sciences
NBII: National Biological Information Infrastructure
CAIN: California Information Node
&
UC Natural Reserve System

This research was supported in part by
California Bay-Delta Authority
Ecological Restoration Program (Award # ERP-01-NO1).

Table of Contents

Introduction

The Cosumnes Research Group II (CRG) has employed open data management standards as part of an overall data management strategy and in the process has created a data storage framework that can serve as the basis for future data transactions. We feel that is was an important data architecture decision from the inception of CRG. This document describes critical data architectural components that we feel meet the objectives of a robust implementation of data capture, storage, manipulation, and retrieval in support of environmental sciences.

Our methods are now being integrated into the California Information Node (CAIN) of the National Biological Information Infrastructure (NBII), providing access to a broad spectrum of environmental data over multiple spatial and temporal domains. The use of open data standards ensures that we can communicate and collaborate with our colleagues worldwide and that access to information will not be hindered by proprietary data systems.

We constructed the CRG website (http://baydelta.ucdavis.edu) using the Drupal Content Management System (http://www.drupal.org) integrated with the Coppermine Photo Gallery (http://coppermine.sf.net). While this document describes this implementation, it also goes beyond this scope and provides a background to the underlying extensible architecture, sensor database systems, and the importance of metadata for this system.

General Overview

The objective of our data management strategy, in connection with the Semantic Online Data Access (SODA) module, is to provide a central repository for field data and to avail these data via the Internet. Real-time or historical data, collected in a field setting, often must be transmitted via radio or area-network to a central data repository where it can be processed, uploaded, and stored within a dedicated database. These data also require access from a front-end user application, often in the form of an Internet browser and portal.

Our outline consists of several areas of information research and development. These areas are based on a project's technical requirements and subsequent architecture, which are often designed to be distinct units. These modules include data collection and transport, data storage and management, and website services with front-end applications. Outlined in this document is a description of each module, describing in detail the overall design approach and various technical details where appropriate.

Central to the execution of a standardized data collection and storage project is both flexibility and serviceability. The flexibility of the system must be evident in the handling of different data types (including qualitative data, numerical data feeds, images, and more), support for all operating environments and computing platforms, scalable architecture, as well as overall system extensibility. Insofar as serviceability is concerned, a data system must be easy to maintain and not require a dedicated administrator; adding and removing data feeds should be simple and straightforward.

The appendices contained in this document provide supplemental information for the application and implementation of similar data storage enterprises, including summary tables and data flow schematics.

Sensor Networks and Database Systems

The definition of a sensor network is broad and far reaching. Data collection devices range from small Tidbit sensors which collect temperature only, to weather stations collecting a broad range of parameters, to video surveillance systems, to satellite imagery, to even human data collection—sensors do not need to be electronic devices. Data is transported via a wrapper or transportation mechanism to a data repository or warehouse where it can be utilized. We make a distinction between sensors, which actively send data without request to a hub or mediator (termed 'push'), and sensors that request from others, most probably a data center, that is needed to initiate the process of data transmission (termed 'pull').

The widely deployed sensor network applications challenge traditional data management systems in that processing can now be done on the sensor nodes (depending upon the capability). The distinguishing characteristics of sensor databases comparing to the traditional distributed databases are as follows:

- Communications between a data center and sensor nodes are often limited, both in terms of flexibility and bandwidth. Additionally, the response time of different nodes varies greatly, from several seconds, to days, or to even months, which makes queries against these data challenging.
- The computation and storage capacity in sensor nodes is limited. No complex query processing operations can be performed in sensor nodes.
- The network members are unstable due to harsh environmental conditions (i.e., nodes within the system may become active or inactive at any time, often without warning).
- Data (permanent records of information) are updated incrementally (i.e., it is an append-only system).
- Heterogeneous nodes have different underlying data organizations (schemas), which complicate their integration into a unified system. Furthermore, newer versions of sensor node devices are likely to have significant differences in the parameters it collects, its data definitions and schema, and even its method of delivery (e.g., Bluetooth).

Brief Example

We illustrate here an example of a data system architecture and procedures that elucidate the nature of the problem and potential solutions:

Located on the experimental floodplain of the Cosumnes River, an instrument is measuring water temperature and dissolved oxygen. The instrument has the ability to wirelessly send these data to a nearby field station. The field station computer collects an hour's worth of data, separates water temperature from dissolved oxygen in the data streams, and in turn sends it – via secure

copy – to a data server upload directory located at the Information Center for the Environment. Embedded within the command sequence is a temporal procedure to for the data server to check the upload directory for any new files that might have arrived on a fixed interval (e.g., 60s). When data server finds a new data file from the field sensor network, a program is executed which can add these data to a standardized database. As soon as these data are recognized within the database, a web application will recognize that the time series has extended and subsequently shows users a new data frame. These data are then available for download and analysis, as they are provided in a standard format (e.g., comma-separated value or CSV) that can be imported to a variety of analytical (e.g., Microsoft Excel) or statistical packages.

As outlined, numerous sensors can be placed in a river to capture various water quality parameters, including temperature, pH, dissolved oxygen, specific conductance, and turbidity. Data would be captured hourly and uploaded monthly from the sensors to the data repository. A pull technique is used to extract the data and the transport agent is often an actual person (i.e., field data technician) who downloads the data to a temporary storage device and uploads it to the data warehouse.

An important lineage problem, relating to how a researcher utilizes the data for various statistical analyses, is having a fixed dataset after a query so that if an analysis is rerun, the same results are guaranteed. When a scientist requests data from the system, a text file is generated with a timestamp. These request files are stored for a period of time and the researcher can identify files for longer term storage. It does replicate data, but it is the easiest way to provide guarantee. In the dynamic nature of a database, it is often the case where a missing set of data might be plugged into the middle of a previously run query, changing the resultant tuples (rows returned) – so executing the same query routine would create different results. Keeping a compressed copy of the original result will preserve the data provenance (or history of that data usage) if there is need for data verification.

Architecture

A scalable data architecture is one that can grow both in utility and in service. As new data are collected, the system must be able to handle data additions and losses. Moreover, as more data are accessed, a reallocation of computational work must be accommodated. For these reasons, data applications need to be designed as distinct components that can be run as separate processes on different servers. For example, as the website traffic rises, a database may be physically moved to a different server to manage all data services and requests.

Generalize Data Flow

In a data centric view, we can view data flowing from left to right, from collection to storage to publication and reuse (see Illustration 1 below).

Illustration 1 A simplified schematic showing the various components of the system

In the case of digital imagery, GPS data, and other data collected by a person or device, we can build a sensor database network which enables other researchers, scientists, educators, and other interested parties, to utilize the data that has been extracted. This free flow of information, and data availability, is essential for environmental studies so that future investigations can build upon previous efforts.

Sensor Database Network

The architecture of a simple sensor database system is shown in the figure below (Illustration 2). In this architecture, a data collector transports data from sensor nodes to the data warehouse. Each homogeneous sensor node has an underlying data storage structure (often shared), which is expressed in the data dictionary. In most field environments, the metadata associated with sensor data is as important as the data itself, so we recommend special emphasis on metadata in this section. Initiated by a user or as a batch process, requests for data are served by the query processor, which returns the result to the requester in a timely fashion.

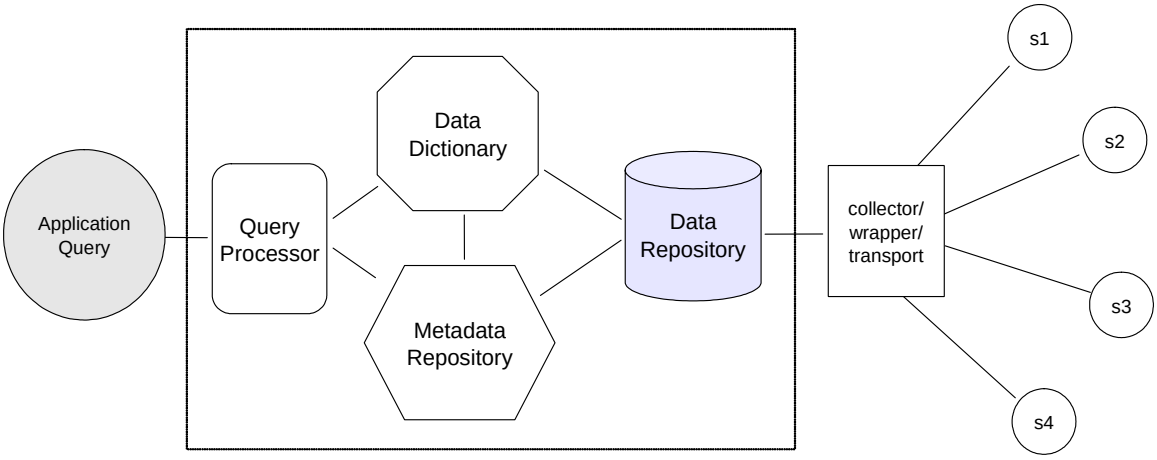

Illustration 2: Sensor database system from user's persepctive.

Data Collection and Transport

Data collected by remote collection devices need to be transported to a data server, often housed in a dedicated facility such as the Information Center for the Environment (ICE). Some transformation of data is likely to occur, which can be done on a remote computer or on a ICE data server. The section below describes these processes in detail.

Transport to ICE Data Server

When there is more than one remote collection device, it may be more efficient to initially collect the data on a remote field station computer, package it for transport, and send it all to the data server (Illustration 3). The remote field station computer could potentially transform data to a needed format, relieving the data server from some computational processing.

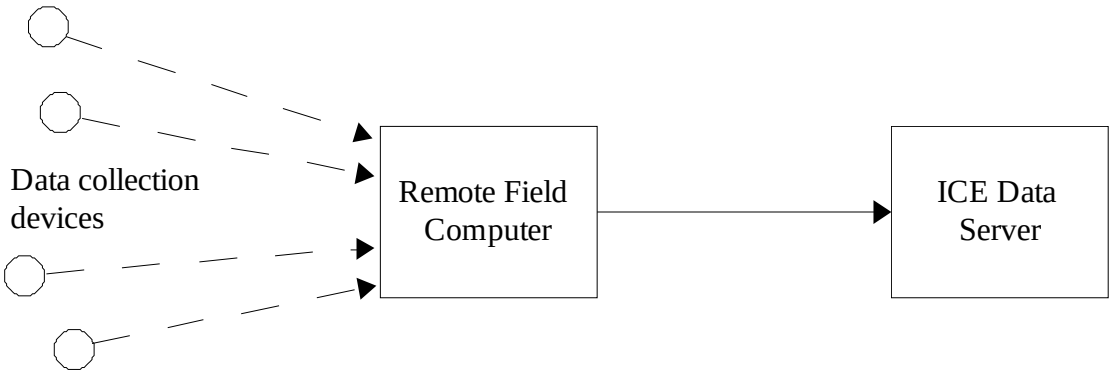

Illustration 3 Remote data collection devices send their data wirelessly to a computer located in a nearby field station, which then transports the data over a network or radio to the ICE data server.

Alternatively, remote collection nodes can send their data directly to a data server, which can process each data stream individually (Illustration 4). This adds a bit more work on the data server, but unless there are hundreds of such devices, computer responsiveness won't be impacted.

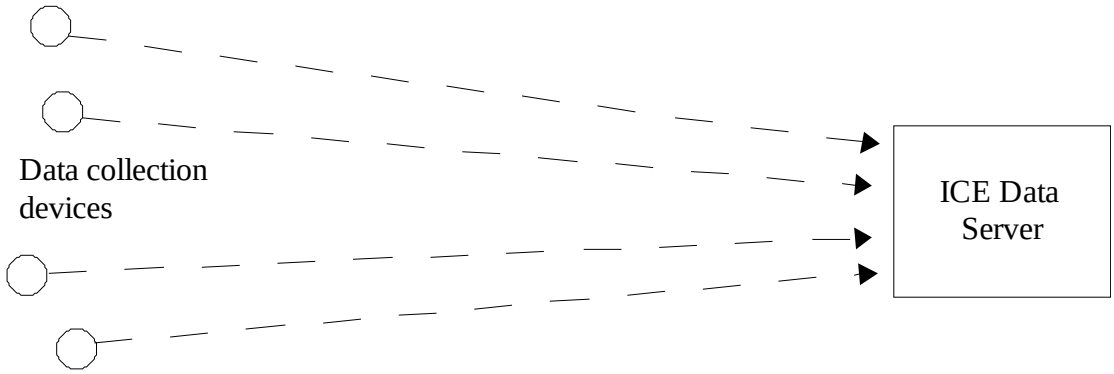

Illustration 4 Remote data collection devices send their data directly to the ICE data server via wireless or networked transport.

Transformation Tools

A set of tools (for example, small Python or Java programs) can handle the transformation from a data collection format to one that a database can more easily incorporate into its schema. These transformation tools often reside on a data server, but can also be implemented from a remote field computer.

Data Import Tools

A similar set of programs and tools are needed to upload transformed data into the desired database. These import tools check for duplicate records and improperly coded data streams. Often, a metadata table will contain information about how these data should appear and which import tools are needed to cleanse data streams.

Incidentally, it is more efficient to make these checks for erroneous data while records are being added rather than executing a periodic procedure against the entire database, which is often too large to accomplish efficiently. The latter would inevitably check the same records each time the procedure is run, while the former routine checks the record only once. As such, it is important to recognize that as more records are entered into the database, the more important this initial screening procedure becomes.

Sensors

From the moment the data are captured, they are in a procedural pipeline to an end user or permanent repository. However, there are many transformational stages between collection and analysis. It is during this process that we capture information for future use. Sensors capture data – often continuously – but it is not until data are analyzed do they transform into information.

In this section, we address the following design considerations regarding sensor nodes:

- Stationary vs. Mobile Sensor Nodes
- Static vs. Dynamic Member
- Homogeneous vs. Heterogeneous Sensor Nodes
- Flat vs. Hierarchical Sensor Networks
- Data Transportation: Push vs. Pull

Stationary vs. Mobile Sensor Nodes

A static placement of sensors is that after a sensor is established at a given location, the sensor remains fixed or stationary. Such is often the case for weather stations, many web-cameras, and river monitoring sensors, to name a few. Dynamic node placement, on the other hand, occurs when the sensors are mobile, and are not held in one position. Examples include sensors attached to animals to understand their movement and migration patterns, orbiting satellites, and traffic helicopters getting video footage of an accident. We generally consider the static varieties

since they are somewhat easier to manage and think about, although moving sensors can often be substituted.

When we look at data quality, we suggest using the spatial context of a sensor to help identify problems with the data. Grouped sensors often have a recognizable spatial autocorrelation present within their data and this artifact can be used to help test data quality. For example, if there are three weather stations that collect data only a few miles apart, and two stations report the air temperature to be 20° C, while one is reporting 10° C, we would question the quality of third sensor's data stream.

Static vs. Dynamic Member

The members in the sensor network may be fixed, as in the sensor network is stable and no new nodes are added. Once any of the current nodes are out of order, it must be fixed quickly to maintain network coverage. For example, a series of temperature loggers on a river reach may be viewed as a sensor network with static members. Conversely, there are many sensor network applications whose members are dynamic. For example, a network of tagged fish (recording position) is hard to remain stable since the sensors can be lost, and new sensors are constantly being added.

The dynamics in membership challenge sensor network database systems. First, there is no guarantee that the answer to a query will be returned from a particular node. Second, after new nodes are added to the network, some queries must be modified to correspond to new information retrieved, and thus previous queries may no longer be valid. Finally, since query processing in sensor networks takes time, we need to identify and handle cases when nodes are non-responsive.

Homogeneous vs. Heterogeneous Sensor Nodes

Within a homogeneous sensor network, all the nodes share the same data organization, which eases the management of the underlying storage system and the overall maintenance of nodes. Although this may be the ideal case (i.e. having nodes of all the same type), generally sensor networks have heterogeneous nodes. Heterogeneous nodes come in two flavors. First, they collect different parameters, e.g. some to monitor temperature and others to detect the wind direction and speed. Secondly, even for the nodes collecting the same parameters, a different data organization may be present, e.g. one device collects temperature and dissolved oxygen, and the newer device collects temperature, dissolved oxygen, and turbidity. This is especially true for newer version of the same type of sensor devices since added functionality drives the cost and purchasing decision.

Flat vs. Hierarchical Sensor Networks

Flat sensor nodes work in the same role and are independent from the other nodes on the network. Hierarchical sensor nodes have embedded dependencies on the other nodes and often perform different functions in the network, e.g. some nodes may gather the data from their

neighbors and transmit the combined data package to the data center while others are isolated from the data center and can only transmit data to a local hub. In some application scenarios, some nodes are located far away from the data center and have to use other sensors to relay their data. In this case, sensors are part of the transportation medium. Since different nodes have different functionalities, there are numerous applications based on the hierarchy.

Data Transportation: Push vs. Pull

The interactions between a given sensor to a data center is a way of classifying the transport techniques. Does the data center passively wait for sensor nodes to push the data to it, or actively pull data from sensor nodes? In the former case, data are usually collected on basis of a given time interval. Since the data center has no control of the initiation, estimation or smoothing techniques may be used to approximate the data during the gap if a request comes in seeking that data. In the latter case, query optimization may be applied to reduce the overall traffic and process cost of the sensor network.

In the push scenario, the data center takes a passive role and waits for the data to arrive. This could produce temporary or permanent gaps in data lineage, which may be resolved by estimation or smoothing techniques. But those techniques might not always be appropriate, depending on the size of the gaps and the application accuracy requirements. For example, if a query takes the average temperature between 10:30am and 3:30pm, and the sensor does not send any data between 11:00am and 2:00pm (for whatever reason), then the hottest part of the day might be missing and thus skewing estimated temperatures. Therefore, given known data patterns, we must determine whether a query is satisfactory. First, the collection behavior of sensors has to be included as part of metadata repository and the query processor needs to know how to handle gaps in the data accordingly. Secondly, some method or model has to be employed to check whether the current data is sufficient to answer the query, which does not happen in traditional distributed database systems.

Similarly, in the pull scenario, what happens when data is requested but no sensor device is found? This can occur if a sensor node goes down, creating a gap in the collection stream. For example, when monitoring the temperature of a river, if the sensor node usually reports temperatures every hour but goes offline for a full day, how would someone get an average temperature reading for the week? A good portion (1/7) of the data is missing! One way to resolve this is by using a bootstrapping approach where the captured data (6/7) is averaged and used in place of the missing day, but this is not a true record of what really happened, only an approximation. For some applications, this smoothing approach might be sufficient, but it often depends on the application and the environmental target.

Therefore, in sensor database systems, besides computing results with the present data, some source information related to the data or results must be provided as well for users, i.e. data lineage.

Metadata

Metadata is data about data. In order for there to be any kind of "intelligent" processing from a robust data system, solid and complete metadata are essential. Examples include full textual descriptions of the data parameters, units of measure for all quantitative data captured, spatial references (location), and theoretical high and low values for each parameter.

Metadata provide the semantics (meaning) to a data system so that it can respond better to human interaction. For example, metadata enable a query interface to be populated with contextual information that aid a researcher in how to manipulate the data for analysis.

Metadata can be extremely useful when users want to apply rules to measure the quality of data. A data dictionary is a type of metadata about the data and describes what is stored in the data repository and the relationships among data entities, although it is not considered a complete data description. The data dictionary provides information about the characteristic of a dataset, but does not have any knowledge about what the data represents.

There is a distinction here between metadata that can be accumulated automatically (e.g., through a script or program) and metadata that must be entered by a knowledgeable person (i.e., an expert) who understands the data stream. In the former case, a program can search all the records in a table and calculate, for example, the highest, lowest, and mean values for a given attribute. But in reference to the latter, a computer program cannot know what the theoretical upper and lower bounds would be, or what a *normal* value would be for a given situation. More concretely, only through metadata can we instruct computers to recognize that water temperatures of -10° C are impossible. Therefore, it is important to track metadata elements so that a data system can make some basic decisions based on the available data. In the context of a sensor database, metadata can help determine the availability of the sensors data for the time period in question, and the quality of the data.

Dublin Core Metadata

The Dublin Core metadata element set is slowly becoming a standard set of tags which can be associated with objects – often documents, images, and other digital media – at varying level of granularity. That is to say, a data set can be associated with an entire database, or a single query. This helps other computer-borne processes identify dataset contents. The Dublin Core contains a total of 15 standard tags, including copyright and contact information. It is fairly easy to envision a set of metadata tags, such as Dublin Core, to be apart of the future of digital storage.

Element	Description
title	name given to the resource
creator	entity primarily responsible for making the content of the resource
subject	topic of the resource's content (controlled vocabulary)
description	An account of the content of the resource.

Element	Description
publisher	entity responsible for making the resource available
contributor	entity responsible for making contributions to the content of the resource
date	date of an event in the lifecycle of the resource
type	nature or genre of the content of the resource
format	physical or digital manifestation of the resource.
identifier	unambiguous reference to the resource within a given context
source	reference to a resource from which the present resource is derived
language	language of the intellectual content of the resource
relation	reference to a related resource
coverage	extent or scope of the content of the resource
rights	Information about rights held in and over the resource

Table 1. This table lists the 15 standard Dublin Core elements and describes their purpose.

Data Dictionary

A data dictionary is embedded within all relational database systems and is the primary source of metadata for the system to function properly. These are often tables or other dynamic data structures which track the size and type of each data element. If the data dictionary is lost, the data itself may be at risk since there is no structure telling the system how to access the data.

Whether we are working with a sensor database or a normal relational database, the data dictionary is an integral part of the system. It is constantly being referenced by the system to know which tables to extract data from, where to apply updates, where to find a stored procedure to execute, how to apply locks, and manage constraints. The data dictionary is, in some sense, the most important component of the system. It needs to be recognized as an important element, but also one that is left to the system to manage. The metadata repository, on the other hand, is where we can define new elements to describe our data and turn them into information.

Data Storage and Management

A database schema often includes a set of standard and non-standard tables. The standard tables contain more general data that are used for command and control of the data system. Specialized data might have a dedicated table, but this would occur only in extreme circumstances. By keeping this option open, the system focus on flexibility is maintained.

Data Repository

Data from sensor nodes are stored in a raw data repository. It is the storage mechanism of the data, in terms of both logical and physical storage. Potentially, some of the data are summarized or aggregated, creating a data warehouse which can speed up query processing and serve a larger number of requests, but it is not the primary intent of the repository.

Adding a new collection node

Adding a new collection node is conducted via an administrative interface. A procedural call is executed differently if a new data node has the same type of data that is already being stored or if the new data to be added is unique. If the new data feed is of the same data type that the database already stored, additional node configuration is straightforward and easily accomplished, as is just a matter of adding the node information through established forms. Behind the scenes, a few records will be added to the database and a file will be written to that tracks file types in the upload directory. The data system recognizes its data type is already a stored "standard", which is currently in the collection set and the addition of a new node is a simple record addition to the database.

If the node to be added is non-standard data, or the underlying database schema has to change, other procedures are necessary. This might involve building a data import program or a data transformation program to handle this case. Additionally, we can separate the data streams so that the data is stored in a more singular fashion. For example, if the instrument collects temperature and dissolved oxygen (DO), we can store temperature in its own table and DO is another.

Centralized vs. Distributed Query processing

A query processor receives requests from client applications (or dynamic queries issued by a user) and returns the requested results. The query processor interfaces with the data dictionary to know what data to return as well as where to find it within the repository. In a centralized query processing model, the sensor nodes just collect and send the raw data to the data center. The query processor then works against the data center, which is simple and efficient since all the data are locally stored. In some applications, however, the communication cost is high, thus distributed or in-network, query processing should be used. In-network query processing distributes some processing work on sensor nodes directly, in that the sensor nodes not only collect data but also apply some query processing, reducing the volume of data to be transferred to the data center. This architectural decision is a trade-off between communication and computation cost. When the traffic between sensor nodes and data center is reduced, more computation must be done by sensor nodes. Thus, a sensor database system must be designed to consider both network bandwidth and sensor nodes processing capability and ultimately whether to use in-network query processing as a procedural technique.

Security

Some data might need to initially be password protected, but still available online for retrieval from the data's owner. Security should be built into database design, as well as the front-end website application. Security management can easily become a time consuming process in terms of administration as well as development time. Security should be kept somewhat simplified, yet robust and flexible enough to meet the requirements of the project.

Backup

Backup of the data should be accomplished as a CRON job which would copy the data files to a remote location elsewhere on the network. Backup can be done as a full data image (complete), incremental (files will be included which have changed since the last backup), or differential (files will be included which have changed since the last full backup). Backup files can be compressed for leaner storage requirements. As mentioned above, the raw data files can easily be used to restore the database to a current state. The other method of data restoration is often from a timed backup, such as a regularized push.

Website Application Framework

The website will have a clean interface, clear navigation, and maintain flexibility and expandability as a project like this requires. We can consider building the application on an existing framework, such as Mambo or another CMS. It will be important that the website is sleek and fast so that the management of data can remain the primary function of the server.

The site will be coded in PHP, an open source web scripting language that provides dynamic web content. It will interface with the database through a database abstraction layer (likely ADOdb), enabling the database to be independent of a particular vendor.

Open Standards

Creating standards ensure compatibility between components; open standards guarantee the availability worldwide. When seamless global collaboration is the objective, the importance of standards become paramount. Therefore, we need to look to international technology neutral organizations, such as the World Wide Web Consortium (W3C) and the Open Geospatial Consortium (OGC), to set the precedent in terms of such standards. Additionally, to ensure availability and "fair use" of content, we need to welcome the efforts of the Conservation Commons[1] and Creative Commons[2]. Open Source Software is a contract, or license, ensuring that source code is available to the general public (worldwide) and that the compiled version is distributed freely (without cost). The combination of open standards, open source, and the availability of content (and data) ensures that our colleagues across the globe can participate in the science we conduct.

[1] http://www.conservationcommons.org
[2] http://www.creativecommons.org

Community-based initiatives and Open Source tools

> "The hardest fact to grasp about the Internet and the I-way [Information Superhighway or Infobahn] is this: It isn't a thing; it isn't an entity; it isn't an organization. No one owns it; no one runs it." (Gleick, 2002).

The Open Source Definition includes ten points that are meant to ensure the freedom of a software product. Many believe this just means free software. It does mean free software, but it is designed to be much more. It includes the right to the source code for personal modification, ensures that derived works credits the original author(s), it does not discriminate against a person, group, or fields of endeavor, and it ensures that all derived work will include a similar license agreement. In essence, this agreement embodies what academicians are striving to do: build upon others' work, provide credit to the original author, and to ensure that this freedom is preserved over time.

Semantic Web

Tim Berners-Lee, James Hendler, and Ora Lassila, wrote an article in the May 2001 Scientific American and defined the semantic web as such: "The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation."

The semantic web provides a framework, or a set of tools, to intelligently make decisions about a specific problem. Pulling together distributed resources, it can classify data and content that enable more pertinent searches for this information. To many computer scientists, the semantic web is an extension of computer artificial intelligence and decision support systems. It uses RDF and OWL, both XML-based extensions, to build a set of interpretive subject/object constructs and ontologies. Other tools are built to process these subject/object constructs, such as the SPARQL query language.

The semantic web can be built using Open Source tools, and it is an ideal methodology to demonstrate the need for continued access to data and content. It certainly has the potential to further progress in academic fields, such as ecology. An important question which this development will attempt to answer is how can a long term dataset be expressed in a semantic context?

Global Implications

A communication network is established that spans the globe. The Internet decreases the distances between people, yet simultaneously, the technology also makes people rely more on information that can be "googled" or accessed effortlessly. It creates links across vast distances. Cyberspatial is a coined term to represent the Internet landscape and how distances are affected by the advent of the Internet. Email, websites, instant messaging, and now RSS are all technologies that bring the expanse of the world to one's desktop.

Like the Open Source definition states, there should be no discrimination between persons or group when it comes to accessing information and data, both of which lead to knowledge. This

is a global concept. For improved global collaboration and communication to occur – and not just in scientific circles – we each need access to the same information. For example, to study global climate change, having access to world data centers, where long term datasets can be accessed, is critical. Science is based on sensible data transformed into information, which must be accomplished collectively.

CRG II Implementation

The CRG II website was built with a content management system (CMS) called Drupal[1] and a data/image repository called Coppermine[2]. These two applications communicate with one another using a utility called Coppermine Fetch[3]. Both Drupal and Coppermine use a MySQL[4] database backend to organize and store data (both application data as well as CRG related content) and both were coded with PHP[5], a web scripting language. All components (CMS, gallery, database, programming language, utility programs) are open source, licensed with a GNU Public License (GPL).

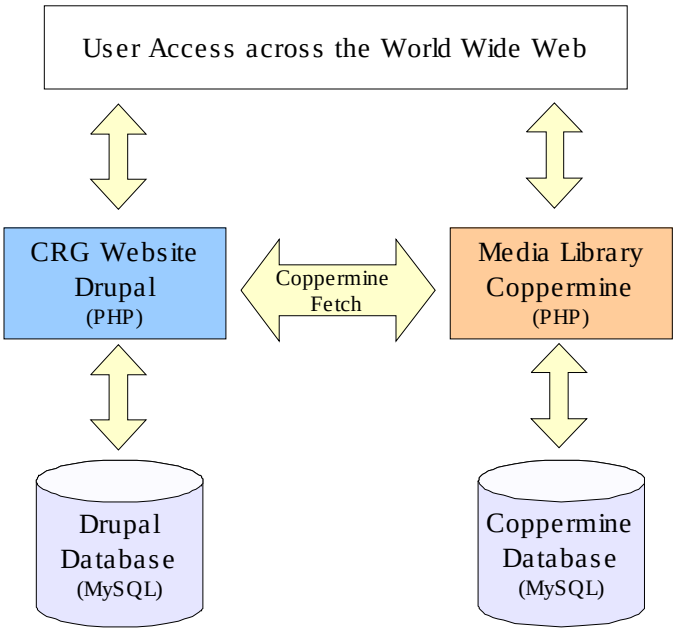

Illustration 5: CRG integrated website architecture

[1] http://www.drupal.org
[2] http://coppermine.sf.net
[3] http://www.fistfullofcode.com/projects/copperminefetch/
[4] http://www.mysql.com
[5] http://www.php.net

The figure above (Illustration 5) shows how each of these is related. The system has an extensible and flexible architecture, and incorporates tagged metadata recognizable within semantic contexts. The front end is web-based, so access is worldwide.

See Appendix D for a list of Coppermine Elements.

Selected Bibliography and References

Benjelloun, O., Sarma, A.D., Halevy, A., Widom, J.: ULDBs: Databases with uncertainty and lineage. Technical report, Stanford University (2005)

Berge, Erling (2003). Environmental Protection in the Theory of Commons. Presented at the "Trans-nationalizing the commons and the politics of civil society." Chiang Mai, Thailand, 11-14, July 2003.

Berners-Lee, Tim, Hendler, James, Lassida, Ora (2001). The Semantic Web: A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities, Scientific American, May 2001.

Bonnet, P., Gehrke, J., Seshadri, P.: Towards sensor database systems. In: Proceedings of the Second International Conference on Mobile Data Management. (2001) 3–14

Bose, R., Frew, J.: Lineage retrieval for scientific data processing: a survey. ACM Computing Surveys 37(1) (2005) 1–28

Buneman, P., Khanna, S., Tan, W.C.: Why and where: A characterization of data provenance. In: The eighth International Conference on Database Theory(ICDT'01). (2001) 316–330

Cui, Y., Widom, J.: Lineage tracing for general data warehouse transformations. In: Proceedings of the 27th International Conference on Very Large Data Bases(VLDB'01). (2001) 471–480

Dublin Core Metadata Initiative: http://dublincore.org.

Gleich, James (2002). What Just Happened: A Chronicle of the Information Frontier, Pantheon Books.

Jeffery, S.R., Alonso, G., Franklin, M.J., Hong, W., Widom, J.: A pipelined framework for online cleaning of sensor data streams. In: Proceedings of the 22nd International Conference on Data Engineering(ICDE'06). (2006) 140

IUCN (The World Conservation Union) (2004). Sharing Information with Confidence: "The Biodiversity Commons": past experience, current trends and potential future directions.

Lessig, Lawrence (2002). The Architecture of Innovation. Duke Law Journal, Vol. 51, No. 6, 2002.

Ludaescher, B., Lin, K., Bowers, S., Jaeger-Frank, E., Brodaric, B., Baru, C.: Managing scientific data: From data integration to scientific workflows. In: GSA Today, Special Issue

on Geoinformatics. (2005)

Moritz, Thomas Daniel (2004). Conservation Partnerships in the Commons? Sharing data and information, experience and knowledge, as the essence of partnership. Museum International, No. 224, Vol. 56, No. 4.

ROADNet Project: http://roadnet.ucsd.edu/.

Schweik, Charles M., Evans, Tom P., Grove, J. Morgan (2005). Open Source and Open Content: A Framework for Global Collaboration in Social-Ecological Research. Ecology and Society, forthcoming.

Scientific American Editors (2005), Beyond the Big ©, SA Perspectives, Scientific American, February, 2005.

Widom, J.: Trio: A system for integrated management of data, accuracy, and lineage. In: Proceedings of the Second Biennial Conference on Innovative Data Systems Research (CIDR'05). (2005)

Yao, Y., Gehrke, J.: Query processing in sensor networks. In: First Biennial Conference on Innovative Data Systems Research (CIDR'03). (2003)

Appendix A: Technologies and Tools

Technical Area	***Technology and notes***
Operating System:	The GNU/Linux operating system will be used to manage system services, although there is nothing about this project that would prevent it from running under Microsoft Windows.
Database (RDBMS):	PostgreSQL or MySQL would be the initial choices, but the system could also be run on other RDBMS, including Oracle and MS SQL Server. Microsoft Access is not an option.
Data Migration Tools:	Python, a platform independent scripting language, will be used for some of the data transformation processes. A database abstraction layer will be used so that several backend databases are supported.
Web Server:	Apache webserver is, once again, platform independent. It provides flexibility, reliability, excellent community-based support, and is a proven product.
Website:	PHP is a web-based scripting language that extends HTML and allows for dynamic websites. PHP runs on both GNU/Linux and Windows and is feature rich. A database abstraction layer, ADOdb, will be used to communicate with the database, perhaps making the database interchangeable.

Main Content Channels

Content	***Description***
Natural Reserve System and Field Stations	This area is designed to provide a reference on the areas of the data collection. This can include the geomorphology, vegetation, mammals, water resources, human impact, and whatever else we want.
Data Access	Online data access and views. This includes HTML tables, comma-separated values (CSV), graphs, and other visual models.
Data Analysis	Descriptions on the types of analysis that can be done with such data. This is designed for educational purposes and will include sample data sets that demonstrate certain techniques.
Equipment List	Description of the remote data collection equipment.
Image Library	Online image library of the sites in question.

Appendix B: Outline of Data Flow

1. Data collection at remote sites
   a) Field data server assembles individual remote node data and packages it for transport.
   b) Data is sent directly from the collection device to the ICE data server.
2. Data transport to ICE data server
   a) Real-time data is sent via radio or network to data server. Deposited in upload directory.
      - Data is either formatted by the field data server to adhere to a particular format (interface) which can be brought into the database automatically.
      - ICE data server can also parse the upload files and convert to a format that can be automatically inserted to ICE data server. Inevitably we will need both methods although the former is preferred as it move some of the processing off of the data server.
   b) Historical data added by same mechanism, that is, placed in the upload data directory.
3. Upload to data repository
   a) A timed routine will periodically check the upload directory for new files. Once a new file is identified, it will use a cross reference table to determine the proper routine to call that will upload the data to the correct database table.
   b) If a file is found that the server cannot identify, it will move the file to a new location which the administrator can look at later.
   c) A log file of actions is recorded to the log directory.
   d) Once a file has been added to the database, it is moved to an archive directory. If the database needs to be restored, one can simply move all of these files back into the upload directory and the timed handler will know to process them.
4. Once the data resides in the database, it is immediately available for online viewing.
5. A web application provides access to the data.
   a) Security is managed by the web application to various online data.
   b) The web application will have various methods of extracting or viewing the data, which include a graph, a web-based table, a comma delimited file to download, and a summarized view.

Appendix C: Open Source GIS Organizations, Tools, and Related Websites

Organizations and Standards

Open Geospatial Consortium

http://www.opengeospatial.org/

Web Map Service (WMS)
Approved ISO Standard
http://www.opengeospatial.org/

Web Feature Service (WFS)

Specification for interoperable geographic features over the Internet (using XML and GML).
http://cite.occamlab.com/test_engine/wfs_1_0_0/wfs_1_0_0.html

GIS

GRASS

Spatial data management.
http://grass.itc.it/

Quantum GIS (QGIS)

Friendly UI for managing maps and other GIS Layers.
http://qgis.org/

uDig

User-friendly Desktop Internet GIS.
http://udig.refractions.net/confluence/display/UDIG/Home

Databases and Spatial DB Extensions

PostgreSQL

Open Source Object Relational Database.
http://www.postgresql.org

PostGIS

Spatial extension to PostgreSQL.
http://www.postgis.org

MySQL

Open Source RDBMS. Supports spatial data.
http://www.mysql.com
http://dev.mysql.com/doc/refman/5.0/en/spatial-extensions-in-mysql.html

Webtools

MapServer

Web based mapping tool from UMN.
http://mapserver.gis.umn.edu/

Maptools.org

Provides great installations of mapserver for Linux/Windows.
http://www.maptools.org/index.html

GeoServer

Open Source WFS incorporating WMS.
http://docs.codehaus.org/display/GEOS/Home

Conferences

Geoinformatics 2006

(FOSS) Free and Open Source Software
Sept 12-15 2006, Lausanne, Switzerland
http://www.foss4g2006.org/

Other Tools and Data Formats

R-spatial

Spatial data extension to the R Statistical programming language.
http://r-spatial.sourceforge.net/

Proj.4

Cartographic Projections Library (used by Mapserver and other tools)
http://www.remotesensing.org/proj/

GDAL

Geospatial Data Abstraction Library for raster data formats
http://www.gdal.org/

GML

Geographic Markup Language
http://opengis.net/gml/

GeoTIFF

Cartographic data embedded in Tiff (Raster) graphic files.
http://www.remotesensing.org/geotiff/geotiff.html

Other Sources of Information

The State of Open Source GIS

White Paper
http://www.refractions.net/white_papers/

Open Source GIS

Provides a listing of many Open Source GIS Projects.
http://opensourcegis.org/

FreeGIS.org

Another listing of Open Source GIS tools.
http://freegis.org/

RemoteSensing.org

Open Source tools related to remote sensing.
http://remotesensing.org/tiki-index.php

SlashGeo

Slashdot-like website focusing on spatial issues and interests.
http://slashgeo.org/

Appendix D: CRG II Data/Image Repository Contents Outline

Coppermine Photo Gallery / Site Map

CRG Reports and Publications: Articles and Reports on the Cosumnes River & Floodplain

- Cosumnes Research Group: Phase I: Quarterly reports, conference posters, publications, regional maps and other related documentation.
    - Aquatic Resource
    - Data Analysis
    - Geomorphology
    - Hydrology
    - McCormack Williamson Tract - CalFed
    - Water Quality
- Cosumnes Research Group: Phase II: Quarterly reports, conference posters, publications, regional maps and other related documentation.
    - Publications
    - Quarterly Reports
    - Published Maps
    - Linking Aquatic and Terrestrial Ecosystems
    - Floodplain Restoration and Monitoring
    - Data Management
    - Archived Reports

CRG Image Library: Images from the Cosumnes River Watershed, California, USA

- Cosumnes River Preserve Vegetation: Vegetation types from the Cosumnes River Preserve
    - Virtual Herbarium
    - Castello Restoration
    - Valensin Forest
    - Triangle Floodplain
    - Mac World
    - Shaw Forest
    - Tall Forest
    - Two Oaks Restoration
    - CVH
- Cosumnes Research Group Activities: Investigations and activities of the Cosumnes Research Group
    - Cosumnes Research Group - Site Visit
    - Cosumnes Research Group - Berkeley
    - Cosumnes Research Group - PRBO
    - Cosumnes Research Group - Hydrology
    - Cosumnes Research Group - Aquatic Invertebrates

    - Cosumnes Research Group - ICE
    - Cosumnes Research Group - Work
- Cosumnes River Floodplain: Images of the Cosumnes River Floodplain
    - February 2004 Flooding
    - Before and After
    - Carson Jeffres on 2005.03.24
    - Floodplain Winter 2005

CRG Data & Video Repository: Data and Media Library

- Cosumnes Research Group Data:Cosumnes river and floodplain data, including remotely sensed imagery.
    - Floodplain Data
    - GIS
    - Remote Sensing
- CRG Video Library: Media library
    - MOV Format
    - AVI Format