Archiving Scientific Data

Susan B. Davidson
CIS 700: Advanced Topics in Databases
MW 1:30-3
Towne 309

http://www.cis.upenn.edu/~susan/cis700/homepage.html
Why archive?

• Data changes over time
  • New data is added
  • Mistakes are corrected
  • Old data is removed

• To enable *reproducibility* and *verifiability*, it must be possible to access the state of a database as of a certain point in time.
  • Also crucial for dereferencing citations

• May also want to ask questions about how the database has changed.
How to archive?

• Many databases periodically publish new versions
• Keep copy of each version
  • Allows data as of a certain time to be accessed quickly
  • May not be space efficient since very little may change between versions
  • Doesn’t allow efficient queries over the change history
• Keep a log of changes ("sequence of delta")
  • Space efficient
  • May be expensive to recompute data as of a certain time
  • May be expensive to query change history
Outline

• Versioning and citation: experiences with eagle-i
• Archiving XML datasets
• Conclusions
Our experience: eagle-i

• eagle-i is an RDF dataset which contains information about resources for translational research (e.g. software, cell lines, lab facilities)
• Each resource has an immutable eagle-i id; the subject of each resource triple is an eagle-i id
• Resources are classified using an ontology, and the citation depends on the classification of the resource.
• eagle-i talked about citation but didn’t automate it...
### Significance Tester for the Accumulation of Reads

**Algorithmic software component**

<table>
<thead>
<tr>
<th>Software Description</th>
<th>STAR was developed to identify regions enriched for a histone modification based on ChIP-Seq evidence, by identifying regions with a significant accumulation of reads.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software Additional Name</td>
<td>STAR</td>
</tr>
<tr>
<td>Used by</td>
<td>Computational Biology and Informatics Laboratory</td>
</tr>
<tr>
<td>Contact</td>
<td>Grant, Gregory R., Ph.D.</td>
</tr>
<tr>
<td>Related Technique</td>
<td>ChIP-seq assay</td>
</tr>
<tr>
<td>Software purpose</td>
<td>DNA modification site prediction objective</td>
</tr>
</tbody>
</table>
Significance Tester for the Accumulation of Reads

Algorithmic software component

Send message to resource contact
Cite this resource

Software

STAR was developed to identify regions enriched for a histone modification

eagle-i ID for this resource:
http://eagle-i.itmat.upenn.edu/i/0000013d-8d96-57e1-2ed7-105480000000

Click here for citation examples and more information.

Related Technique
ChIP-seq assay

Software purpose
DNA modification site prediction objective
Citing an eagle-i Resource

Citing eagle-i resources is an easy way to give credit.

The formats suggested below provide the minimum information necessary to identify and credit the resource provider, and are designed to provide a traceable, durable, and unambiguous reference for the resource being cited. These suggestions can and should be used along with those from other resource identifiers (i.e. Antibody Registry ID, Addgene, DSHB, RRID) or from the journal publishing your work.

Note that for all types, the names of Core Facilities or other ambiguously named organizations should be followed by the name of the affiliated eagle-i institution in order to disambiguate them (e.g. Flow Cytometry Core, Montana State University vs. Flow Cytometry Core, Dartmouth College).

Citation Guidelines

Although only the most commonly cited types are listed below, the same rules can be used to cite any eagle-i resource.
Citation architecture

- Isolator access
  - Access service
- Send message to resource contact
- Cite this resource

- Citation Generator
  - RDF Citation Model
  - Citation Formatter
  - Human-readable citation

- Citation Derenencer
  - Dereferencing Module
  - Citation Parser
  - Original cited object

- eagle-i system
  - eagle-i id
  - eagle-i triple store
  - eagle-i versioning system

- Versioning Manager
The latest copy of eagle-i is available on the website, but it is not “versioned”

We did a daily download since we didn’t know how frequently it changed (not frequently!)

Needed “time queries” to understand how the dataset changed over time
  • What triples were added/deleted in the period [t, t’]?
  • What was the object of triple X at time t?
  • When was triple Y first added/deleted
Example: versioning 2 RDF triples

Last version date: 12/30/2016
Today's date: 12/31/2016

First triple to be versioned:
<eagle-id: e2> <Developer> <Grant, G.>

Second triple to be versioned:
<ont: t1> <rdfs:label> <Software>

---

**triple_store**

<table>
<thead>
<tr>
<th>ID</th>
<th>subject</th>
<th>predicate</th>
<th>object</th>
<th>resource_type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>eagle-id: e1</td>
<td>Developer</td>
<td>Manduchi, E</td>
<td>eagle-i</td>
</tr>
<tr>
<td>2</td>
<td>ont: t1</td>
<td>rdfs:label</td>
<td>Software</td>
<td>ontology</td>
</tr>
<tr>
<td>3</td>
<td>eagle-id: e2</td>
<td>Developer</td>
<td>Grant, G.</td>
<td>eagle-i</td>
</tr>
</tbody>
</table>

---

**triple_log**

<table>
<thead>
<tr>
<th>ID</th>
<th>version_start_date</th>
<th>version_end_date</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>08/04/2016</td>
<td>12/30/2016</td>
</tr>
<tr>
<td>2</td>
<td>09/10/2016</td>
<td>12/30/2016</td>
</tr>
<tr>
<td>3</td>
<td>12/31/2016</td>
<td>12/31/2016</td>
</tr>
</tbody>
</table>
Versioning and citation

• When should versioning be triggered?
  • At least when a user cites an eagle-i resource

• What should be versioned?
  • At least changes to the resource being cited.

➢ If a version of a resource is not cited, it does not have to be stored.
➢ However, time-based queries will only detect changes with respect to citations rather than all changes.
Outline

• Versioning and citation: experiences with eagle-i
• Archiving XML
• Conclusions
Recall: approaches to archiving

- Keep copy of each new version of the database
  - Allows data as of a certain time to be accessed quickly
  - May not be space efficient since very little may change between versions
  - Doesn’t allow efficient queries over the change history

- Keep a log of changes (“sequence of delta”)
  - Space efficient
  - May be expensive to recompute data as of a certain time
  - May be expensive to query change history
Problem with diff-based approaches

- Ignores the “semantic continuity of keys” by focusing on minimal edit distance
Proposed approach in paper

• Focus on *hierarchical* scientific datasets
  • XML-based
  • Changes are primarily insertions
• Changes identified based on keys
• Version merging based on keys
• Inheritance of timestamps
  • Timestamp is stored at a child element only when it is different from the timestamp of its parent element

➢ “Key-based + merging” approach
Example: sequence of versions

Version 1
- db
  - dept
  - name
    - finance

Version 2
- db
  - dept
  - name
    - emp
      - fn
        - Jane
      - ln
        - Smith

Version 3
- db
  - dept
  - name
    - emp
      - fn
      - ln
      - sal
      - tel
      - marketing
      - fn
      - ln
      - 90K
      - 123–4567
      - John
      - Doe

Version 4
- db
  - dept
  - name
    - emp
      - fn
      - ln
      - sal
      - tel
      - fn
      - ln
      - 95K
      - 123–4567
      - Jane
      - Smith
      - 95K
      - 123–6789
      - tel
      - tel
      - 112–3456
      - John
      - Doe
Adding keys

![Diagram showing a database structure with keys for departments and employees.](image)
Example of an archive
Representing archive in XML

```xml
<T t="1-4">
  <root>
    <db>
      <T t="3">
        <dept>
          <name>marketing</name>
          <emp>
            <fn>John</fn> <ln>Doe</ln>
          </emp>
        </dept>
      </T>
      <dept>
        <name>finance</name>
        <T t="3-4">
          <emp>
            <fn>John</fn> <ln>Doe</ln>
            <T t="3"><sal>90K</sal></T>
            <T t="4"><sal>95K</sal></T>
            <tel>123-4567</tel>
          </emp>
        </T>
      </dept>
    </db>
  </root>
</T>
```
What is a key for XML?

- A key has form \((Q, \{P_1,\ldots,P_k\})\), where \(Q\) and \(P_i\) are path expressions
  - \(Q\) identifies the **target set**
  - \(P_i\) are **key paths**, analogous to key attributes in relations
- An XML document satisfies a key \((Q, \{P_1,\ldots,P_k\})\) if
  - From any node identified by \(Q\), every \(P_i\) exists uniquely
  - If two nodes \(n_1\) and \(n_2\) identified by \(Q\) have the same value at the end of each key path in \(\{P_1,\ldots,P_k\}\) then \(n_1\) and \(n_2\) are the same node.
Relative keys

• Since XML is hierarchical, we also need to specify keys relative to a context node
  • \((Q, (Q', \{P_1, \ldots, P_k\}))\)

• Examples
  • \((/ \text{db}, \{\})\). There is at most one \text{db} element below the root.
  • \((/\text{db} \text{dept}, \{\text{name}\})\). Every \text{dept} node within a \text{db} node can be uniquely identified by the contents of its \text{name} subelement.
  • \((/\text{db/dept} \text{emp}, \{\text{fn}, \text{ln}\})\). Every \text{emp} node within a \text{dept} node along the path /\text{db/dept} can be uniquely identified by the contents of its \text{fn} and \text{ln} subelements.
  • \((/\text{db/dept/emp}, \{\text{sal} \})\). There is at most one \text{sal} subelement under each \text{emp} node along the path /\text{db/dept/emp}. 
Assumptions:

- Every key defined for a node is relative to its parent, e.g. the key for `emp` is relative to its parent `dept` node

- Frontier nodes identify unkeyed portions of the document
• Recursively merge nodes in the incoming version (D) to nodes in the archive (A) that have the same key value, starting from the root.

• When a node \( y \) in D is merged with a node \( x \) from A, the timestamp of \( x \) is augmented with \( i \) (the new version number), and subtrees are recursively merged.

• Nodes in D that do not have nodes in A are simply added with \( i \) as the timestamp.
Further compaction under frontier node
• What is the database at \( t=1 \)?
• When did Joe Doe get a salary raise?
• What were the changes to the database between \( t=1 \) and \( t=3 \)?
Conclusions

• Versioning is important for many different applications

• While techniques are similar between different representations (e.g. files, relations, XML, RDF), differences in assumptions can be used to build more efficient solutions.

  • And the operations (e.g. queries) you wish to perform are important too!