Data Formats and APIs

WHAT’S TAKING YOU SO LONG ON THE PROJECT?

THE APPLICATION IS UNSTABLE BECAUSE THE DATA MODEL IS DRIVEN BY AN OVERLY COMPLEX RELATIONAL DATABASE AND THERE WAS NO INTEGRATION TESTING.

DOES ANY OF THAT MEAN THE SAME THING AS “LAZY”?

Mike Carey

mjcarey@ics.uci.edu
Announcements

• Keep watching the course wiki page (especially its attachments):
  • https://grape.ics.uci.edu/wiki/asterix/wiki/stats170ab-2018

• Ditto for the Piazza page (for Q&A):
  • http://piazza.com/uci/winter2018/stats170a/home

• Note: HW#3 is due tonight (11:45pm)
  • HW#4 should be available by then as well

• Today:
  • More PostgreSQL techniques and tips
  • Twitter APIs and Python’s Tweepy package
  • Beyond tables: XML and JSON
**XML**

- Stands for eXtensible Markup Language
- XML 1.0 – a recommendation from W3C, 1998
- Roots: SGML (a complex *document* markup language)
- After the roots: a format for sharing *data* as well
Why XML is of Interest

• XML is just syntax for data
  • (Note: we have no syntax for relational data!)
  • XML is not relational: it’s *semistructured*

• XML’s data syntax is exciting because:
  • Can translate *any* data to XML
  • Can ship XML over the Web (HTTP)
  • Can input XML into any application
  • *Thus*: Data sharing and exchange on the Web!

*(Note: JSON is another similar technology today.)*
HTML (a descendant of SGML)

<h1>Bibliography</h1>
<p><i>Foundations of Databases</i>  
Abiteboul, Hull, Vianu  
Addison Wesley, 1995
</p>
<p><i>Data on the Web</i>  
Abiteoul, Buneman, Suciu  
Morgan Kaufmann, 1999
</p>

HTML describes the presentation
XML describes the content
XML Terminology: Elements & Tags

• **Tags:** book, title, author, ...
• **Start tag:** `<book>`, **end tag:** `</book>`
• **Elements:** `<book>...</book>`, `<author>...</author>`
• Elements can be **nested**
• **Empty element:** `<red></red>` (abbreviated `<red/>`)
• **XML document:** single *root element*

*Well formed* XML document: matching/nested tags
More XML: Attributes

```xml
<book price="55" currency="USD">
    <title>Foundations of Databases</title>
    <author>Abiteboul</author>
    ...
    <year>1995</year>
</book>
```

Attributes are alternative ways to represent data.
More XML: Attributes Revisited

Attributes are best used to represent “metadata”!

<book>
  <title>Foundations of Databases</title>
  <author>Abiteboul</author>
  ...
  <year>1995</year>
  <price currency="USD">55</price>
</book>
XML Semantics: Tree of Data

Also: Order matters! (Or at least it can...)

Michael Carey/Padhraic Smyth, UC Irvine: Stats 170A/B, Winter 2018
XML Data

• XML is **self-describing**

• Schema information is part of the data
  • Consider a relational schema: `person(name, phone)`
  • In XML `<person>`, `<name>`, `<phone>` are part of the data (and are repeated for each person)

• Consequence: XML is much more flexible
  • Can have variations from instance to instance
  • Supports “schema later” (or “schema never”) methodology

• **XML = semistructured** data
Ex: Relational Data as XML

person relation:

<table>
<thead>
<tr>
<th>name</th>
<th>phone</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>3634</td>
</tr>
<tr>
<td>Sue</td>
<td>6343</td>
</tr>
<tr>
<td>Dick</td>
<td>6363</td>
</tr>
</tbody>
</table>

XML:

```
<person>
  <row>
    <name>John</name>
    <phone>3634</phone>
  </row>
  <row>
    <name>Sue</name>
    <phone>6343</phone>
  </row>
  <row>
    <name>Dick</name>
    <phone>6363</phone>
  </row>
</person>
```
XML is Semi-structured Data

• Missing elements and/or attributes:

```xml
<person>
  <name>John</name>
  <phone>1234</phone>
</person>
<person>
  <name>Joe</name>
</person>
```

• Could represent in a table with nulls:

<table>
<thead>
<tr>
<th>name</th>
<th>phone</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>1234</td>
</tr>
<tr>
<td>Joe</td>
<td>-</td>
</tr>
</tbody>
</table>
XML is Semi-structured Data

• Repeated attributes

```
<person>  <name> Mary </name>
          <phone> 2345 </phone>
          <phone> 3456 </phone>
</person>
```

• Impossible in tables (w/o normalization – due to 1NF)

<table>
<thead>
<tr>
<th>name</th>
<th>phone</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mary</td>
<td>2345</td>
</tr>
<tr>
<td></td>
<td>3456</td>
</tr>
</tbody>
</table>
XML is Semi-structured Data

• Attributes with different types in different objects

```xml
<person> <name> <first> John </first> <last> Smith </last> </name> <phone>1234</phone> </person>
```

← Structured name!

• Nested collections (not 1NF)
• Heterogeneous collections:
  • `<db>` containing both `<book>`’s and `<publisher>`’s
XML: So What Is It Again...?

- A standard, flexible, *self-describing* syntax used to represent and exchange data of all shapes and sizes
  - Regular, structured data (think *records*)
    E.g., a purchase order (customer info and line items)
    Record-like, typed, nested data values
  - Irregular, unstructured data (think *documents*)
    E.g., a book (title, author, chapters, and text)
    Text-like, untyped, variant, marked-up data values

- Uses include document storage, data exchange, Web service calls, B2B messaging, information integration, even configuration metadata...
XML: One Final Example

<?xml version="1.0" encoding="ISO-8859-1" ?>
<catalog>
  <book isbn="ISBN 1565114302">
    <title>No Such Thing as a Bad Day</title>
    <author>Hamilton Jordan</author>
    <publisher>Longstreet Press, Inc.</publisher>
    <price currency="USD">17.60</price>
    <review>
      <reviewer>Publisher</reviewer>: This book is the moving account of one man's successful battles against three cancers ...
      <title>No Such Thing as a Bad Day</title> is warmly recommended.
    </review>
  </book>
  <!-- more books and specifications -->
</catalog>
JSON

- **JavaScript Object Notation**
  - Born from JavaScript, now language-independent
- Minimal
  - Much (much!) simpler than XML
- Textual
  - Machine- and human-readable format
- Subset of JavaScript
  - But similar to many languages’ types (including Python)
Values

- Primitive values
  - Strings
  - Numbers
  - Booleans

- Structured values
  - Objects
  - Arrays

- A special “missing” value
  - null
  - (or a field can be altogether missing)
Numbers

• Integer
• Real
• Scientific

• No octal or hex

• No NaN or Infinity
  • Use null instead
Booleans

- true
- false

null

- A value that isn't anything
Object

• Objects are unordered containers of key/value pairs
• Objects are wrapped in \{ \}
• , separates key/value pairs
• : separates keys and values
• Keys are strings
• Values are any JSON values
  • Similar to struct, record, hashtable, object, dict, ...
Object Example

{
    "name": "Jack B. Nimble",
    "at large": true,
    "grade": "A",
    "format": {
        "type": "rect",
        "width": 1920,
        "height": 1080,
        "interlace": false,
        "framerate": 24
    }
}

Michael Carey/Padhraic Smyth, UC Irvine: Stats 170A/B, Winter 2018
Array

- Arrays are **ordered** sequences of values
- Arrays are wrapped in `[]`
- `,` separates values
- JSON does not talk about indexing
  - JSON is just a data format (not a language)
  - An implementation can start array indexing at 0 or 1
Array Examples

["Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"]

[
    [0, -1, 0],
    [1, 0, 0],
    [0, 0, 1]
]
Arrays vs. Objects

- Use objects when the key names are arbitrary strings – i.e., for record-like data
  - Similar to a dict in Python (slightly more restrictive)
- Use arrays when the key names are sequential integers – i.e., for indexed sequences
  - Similar to a tuple or an array in Python
## JSON vs. Relational (and CSV)

<table>
<thead>
<tr>
<th></th>
<th>Relational (and CSV)</th>
<th>JSON</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Structure</strong></td>
<td>Flat (Tables)</td>
<td>Nested (Complex Objects)</td>
</tr>
<tr>
<td><strong>Schema</strong></td>
<td>Per collection (and static)</td>
<td>Per object</td>
</tr>
<tr>
<td><strong>Query Support</strong></td>
<td>SQL standard</td>
<td>Varies (no standard)</td>
</tr>
<tr>
<td><strong>Ordering</strong></td>
<td>None (sets/bags)</td>
<td>Includes arrays</td>
</tr>
<tr>
<td><strong>Native System Support</strong></td>
<td>DB2, Oracle, SQL Server, SQLite, PostgreSQL, MySQL, ...</td>
<td>MongoDB, Couchbase Server, AsterixDB, ...</td>
</tr>
</tbody>
</table>
## JSON vs. XML

<table>
<thead>
<tr>
<th></th>
<th>XML</th>
<th>JSON</th>
</tr>
</thead>
<tbody>
<tr>
<td>Verbosity</td>
<td>Higher</td>
<td>Lower</td>
</tr>
<tr>
<td>Complexity</td>
<td>Higher</td>
<td>Lower</td>
</tr>
<tr>
<td>Use of Validation</td>
<td>Common (DTD, xsd)</td>
<td>Rare (JSON schema)</td>
</tr>
<tr>
<td>PL Friendliness</td>
<td>Low (impedance mismatch)</td>
<td>High</td>
</tr>
<tr>
<td>Query Support</td>
<td>XSLT, XPath, XQuery</td>
<td>JAQL, AQL, JSONiq, SQL++</td>
</tr>
</tbody>
</table>
Questions?

• Next time we’ll talk about data management technologies (databases and query languages) for “modern data”