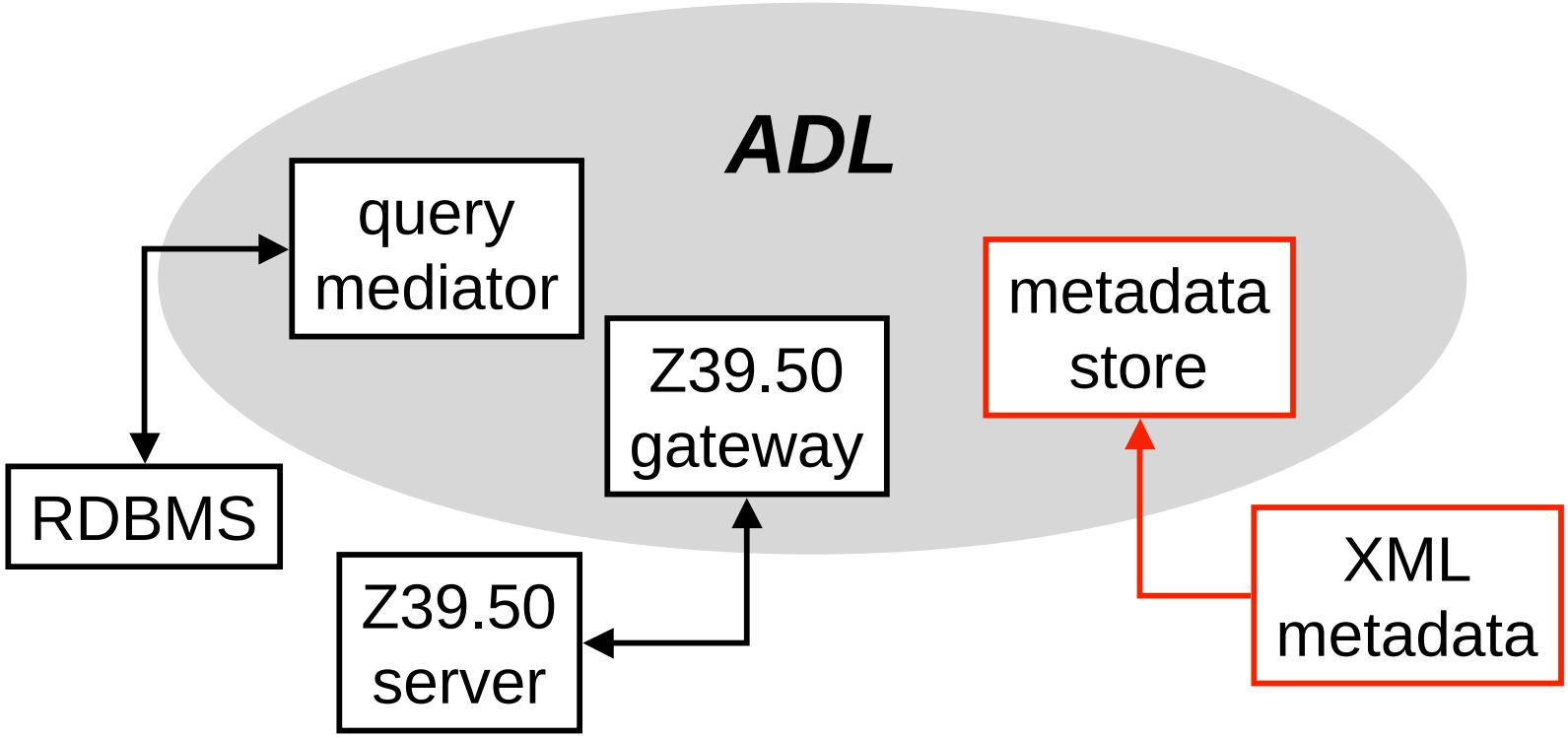
A Hybrid Declarative/Procedural Metadata Mapping Language Based on Python

Greg Janée & James Frew

University of California at Santa Barbara

ADL

- Federated digital library system
  - general framework for distributing queries
  - specific support for geospatial queries, data



ADL metadata views

- Bucket
  - mappings of native metadata to high-level search indices
  - comparable to Dublin Core
  - but strongly typed
- Browse
  - image thumbnails, abstracts, etc.
- Access
  - describes *how* item can be accessed
  - 4 different types of access points
  - 2 hierarchy mechanisms: decomposition, alternatives

Problem statement

- Metadata mapping
  - largely declarative in nature
    - **“A → B”**
  - need recourse to arbitrary computation
    - **e.g., conversion of geographic coordinates**
    - **deal with pervasive quality problems**
  - but problems are often consistent for given collection or provider

- Goal
  - simple, concise way of *specifying* mappings
  - that is executable

Related work

- Crosswalks
  - simplistic (“A is equivalent to B”)
  - no formal way to describe more complex mappings
  - not executable
- Many ad hoc solutions
  - Perl, XSLT
- XSLT
  - difficult to program
  - low-level, verbose
  - computationally hamstrung

Language for metadata mapping

- Mapping is an executable Python script
  - **`from ADL_mapper import *`**
    `input()`
    `...`*`statements`*`...`
    `output()`
- ~25 language statements
  - procedure calls
  - but are declarative in nature
- Query language
  - form tuples from Xpath expressions
- Inheritance mechanism

Two examples

❶
```
map("adl:titles", "/itemRecord/general/title")
```

```
def convertPresentKeyword (v):
  if v[1].lower() == "present"
    return (v[0], "9999")
  else:
    return v
```

❷
```
map("adl:dates",
    ["/itemRecord/temporalCoverages/timeInfo/timeAD"
     "begin@date", "end@date"],
    prefilters=convertPresentKeyword)
```

Partial equivalent in XSLT

```
<xsl:template match="itemRecord/temporalCoverages/timeInfo/timeAD">
  <temporal-value>
    <range>
      <begin><xsl:value-of select="begin/@date"/></begin>
      <end>
        <xsl:choose>
          <xsl:when
            test="translate(end/@date,
              'ABCDEFGHIJKLMNOPQRSTUVWXYZ',
              'abcdefghijklmnopqrstuvwxyz')='present'">
            <xsl:text/>9999<xsl:text/>
          </xsl:when>
          <xsl:otherwise>
            <xsl:value-of select="end/@date"/>
          </xsl:otherwise>
        </xsl:choose>
      </end>
    </range>
  </temporal-value>
</xsl:template>
```

Maya case study

- Maya: thematic collection of GIS and other materials with FGDC metadata

- Mapping problems:
  - poor titles, e.g., “mex250kr”
  - missing originator
  - misused FGDC field

- Solution
  - generic FGDC mapping
    - **23 statements**
  - Maya mapping derived from FGDC
    - **3 statements**

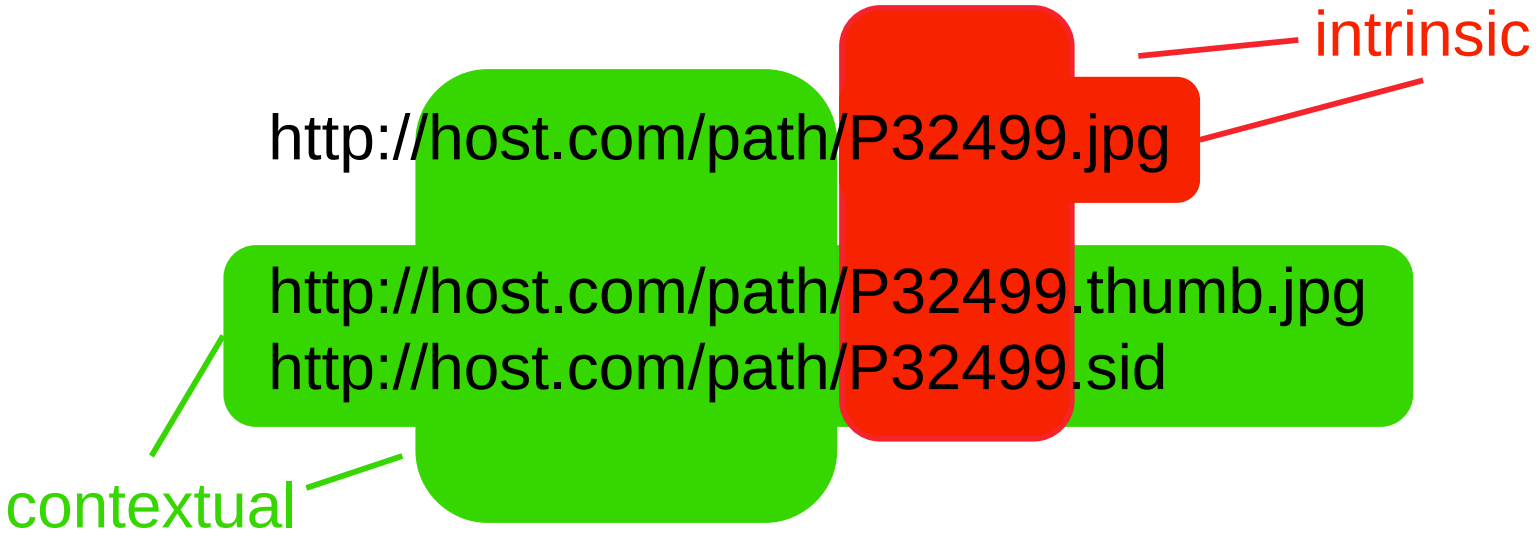
Actual Maya mapping

```
from ADL_mapper import *
input()

import FGDC_mapping

def constructNewTitle (v):
  form = get("/metadata/idinfo/citation/citeinfo/geoform")
  ...
  return ...

❶ appendPostfilter("adl:titles", constructNewTitle)

❷ unmap("adl:formats", id=2)

❸ mapConstant("adl:originators",
  "Anabel Ford, ISBER/MesoAmerican Research Center")

output()
```

Future work

- Mapping access-related metadata is problematic
  - poorly supported by metadata standards
  - only partially intrinsic to the item

- Need to add library context as an input



Summary

- Problem
  - create executable specifications
  - largely, but not entirely declarative
  - need ability to do arbitrary computation
- Solution
  - work within a high-level, procedural language
  - procedure calls act like declarations
  - resulting hybrid declarative/procedural language
- http://www.alexandria.ucsb.edu/mm/