Outline

- ADEPT overview
- Core library architecture
  - Metadata interoperability
  - Query translation
  - Collection discovery
- Concept modeling & educational applications

Quick project overview

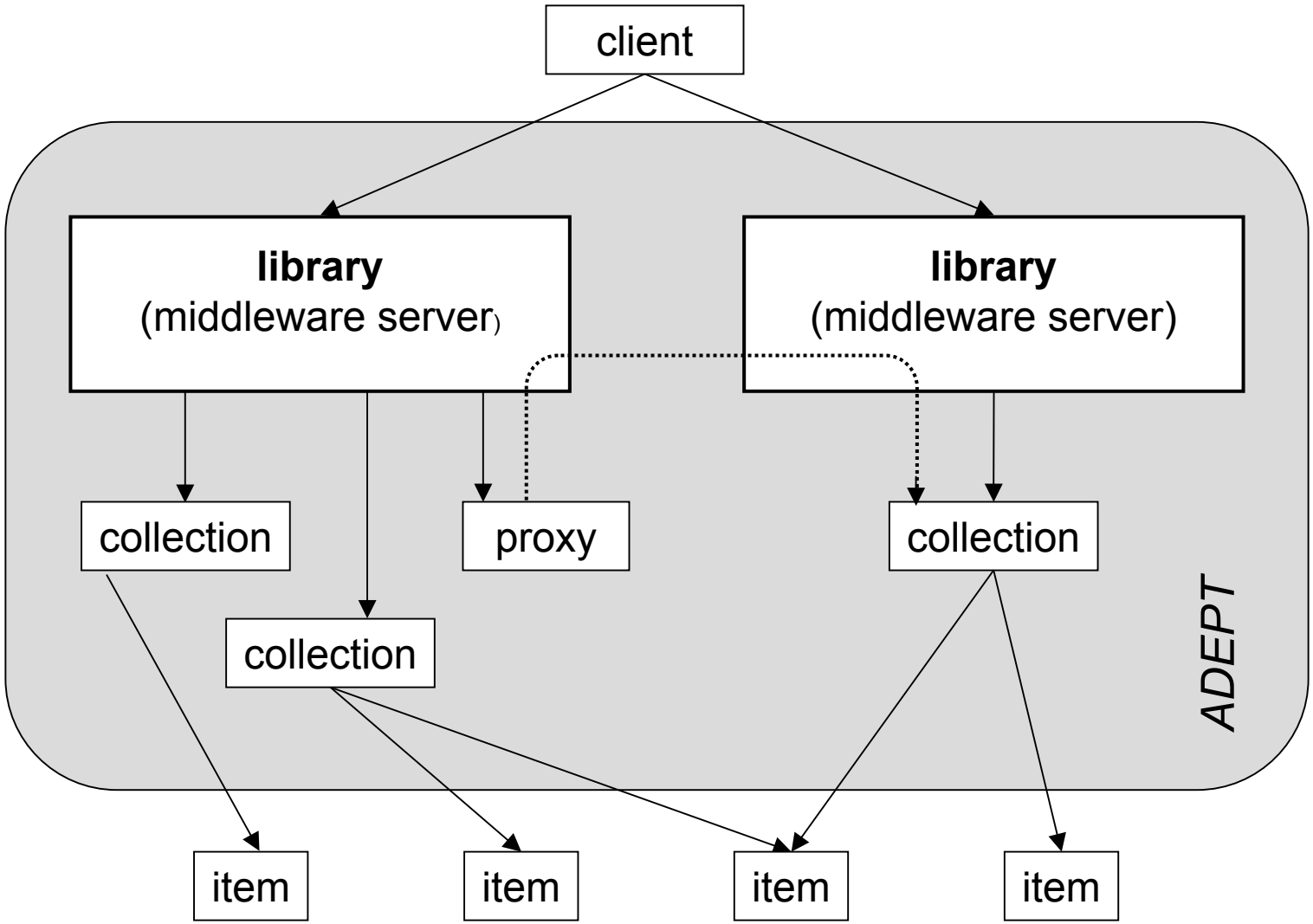
- Third layer of Internet
  - "library'" layer
  - persistence, accessibility, and organization
- Collections characterized by metadata
  - for items
  - for collections
- Models of DL organization
  - harvesting/central metadata
  - distributed peer-to-peer DLs (ADEPT model)
- Services for
  - discovering/accessing knowledge
  - using knowledge
  - creating knowledge
- GRID computing + DLs

ADEPT Core Library Architecture

Core architecture goals

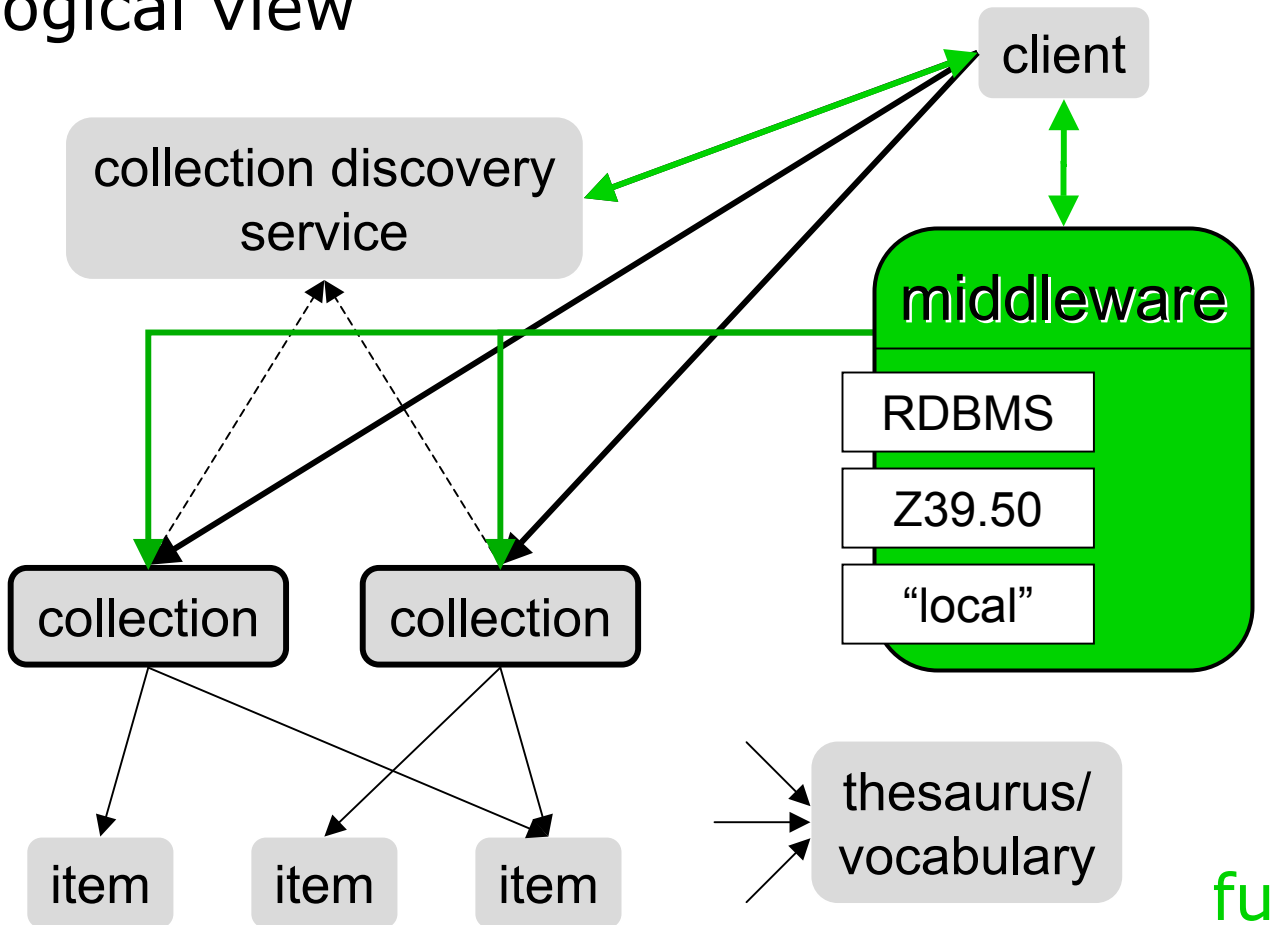
- Goals
  - distributed digital library for georeferenced information
  - services supporting DL federation and interoperation
  - "personalized learning spaces"
- Scalability
  - many collections
  - collections, very large to very small
  - extreme heterogeneity

Big picture


client
library
(middleware server)
library
(middleware server)
collection
proxy
collection
collection
ADEPT
item
item
item
item

Middleware server

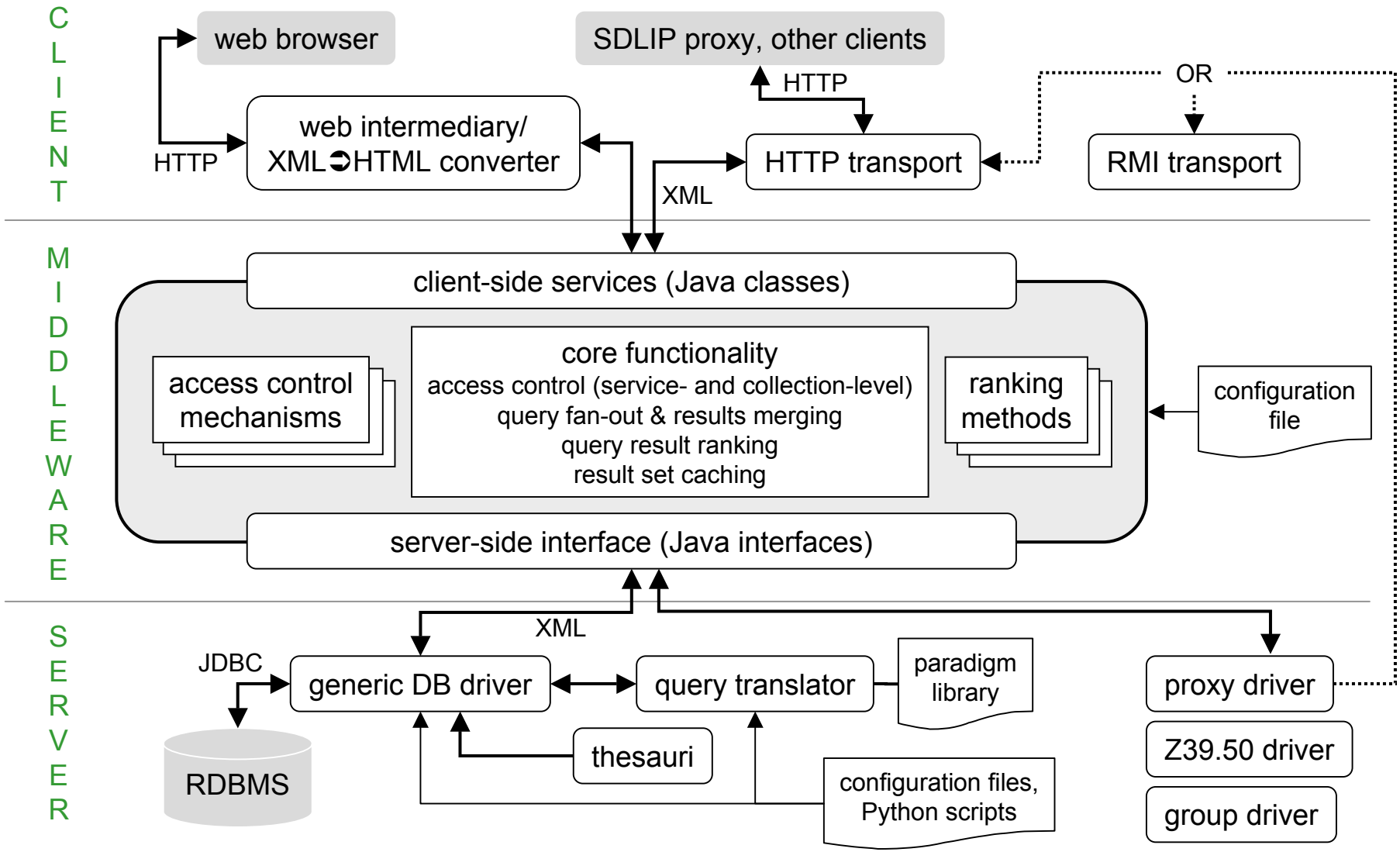
logical view



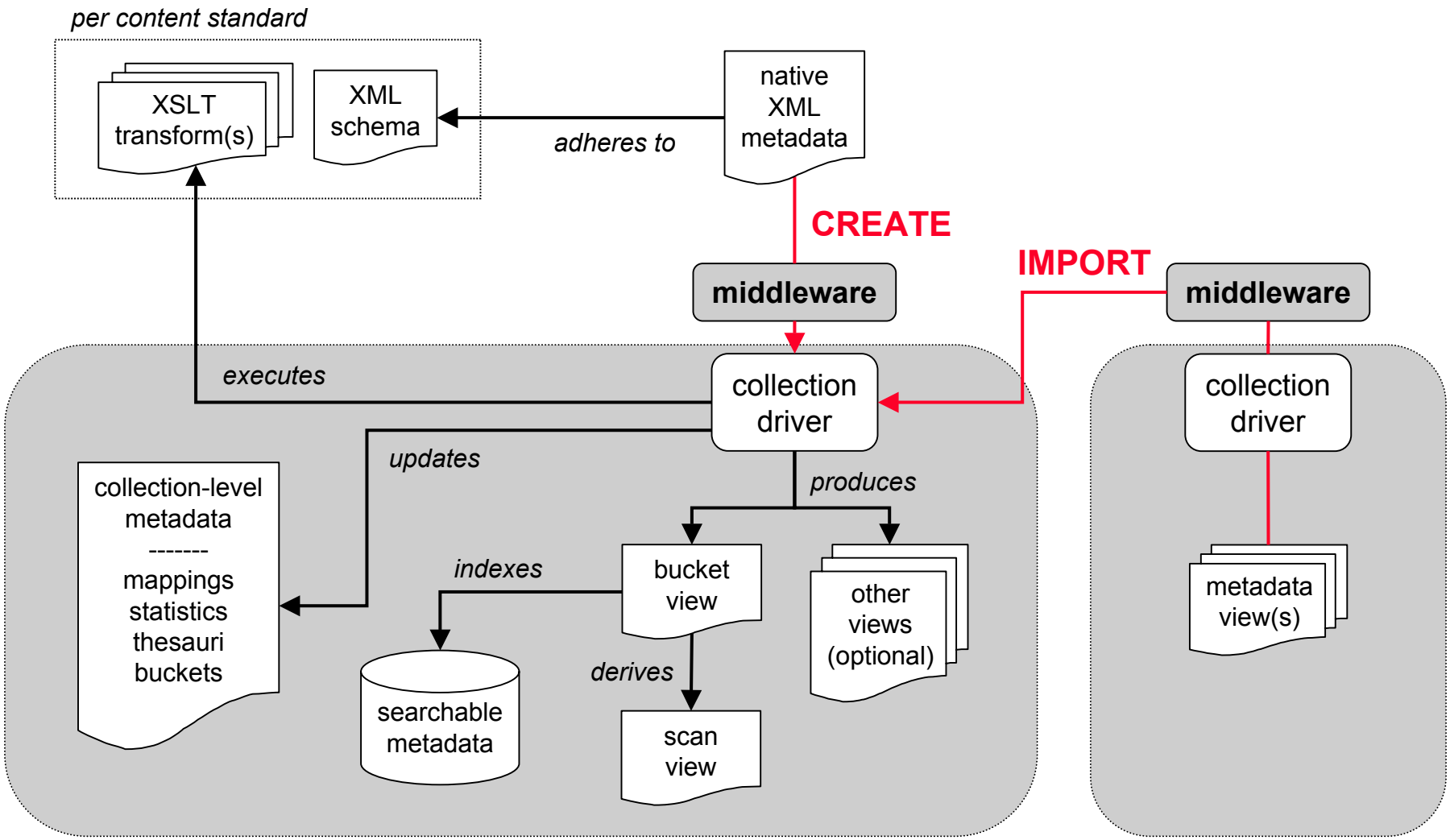
- local access point
- standard services
- access control
- thin client support
- distributed search
- brokering of queries & results
- proxying of collections & items
- creation & organization of “local” collections

functional view

Boxology


CLIENT
web browser
SDLIP proxy, other clients
HTTP
OR
HTTP
web intermediary/
XML➲HTML converter
HTTP transport
RMI transport
XML
MIDDLEWARE
client-side services (Java classes)
access control
mechanisms
core functionality
access control (service- and collection-level)
query fan-out & results merging
query result ranking
result set caching
ranking
methods
configuration
file
server-side interface (Java interfaces)
SERVER
XML
JDBC
generic DB driver
query translator
paradigm
library
proxy driver
Z39.50 driver
group driver
RDBMS
thesauri
configuration files,
Python scripts

"Local" collection population



Metadata Interoperability

ADEPT's interoperability problem

- Distributed, heterogeneous collections
  - locally, autonomously created and managed
- Minimal requirements on collection providers
  - allow use of native metadata
- Provide uniform client services
  - common high-level interface across collections
  - structured means of discovering and exploiting (possibly collection-specific) lower-level interfaces
- Assumptions
  - items have metadata
  - items have sufficient, "good" metadata
  - i.e., this is a metadata interoperability problem

What is a bucket? (1/2)

- Strongly typed, abstract metadata category with defined search semantics to which source metadata is mapped
- Key properties
  - name
    - **Coverage date**
  - semantic definition
    - **The time period to which the item is relevant.**
  - data type (strictly observed)
    - **calendar date or range of calendar dates**
  - syntactic representation (strictly observed)
    - **ISO 8601**

What is a bucket? (2/2)

- Source metadata is mapped to buckets
  - buckets hold *not* just simple values
    - **"2001-09-08"**
  - but rather, explicit representations of mappings
    - **(FGDC, 1.3, "Time period of content", "2001-09-08")**
  - multiple values may be mapped per bucket
- Bucket definition includes search semantics
  - defines query terms
    - **ISO 8601 date range**
  - defines query operators
    - **contains, overlaps, is-contained-in**
  - semantics are slightly fuzzy in certain cases to accommodate multiple implementations

Collection-level aggregation

- Collection-level metadata describes
  - buckets supported by the collection
  - item-level metadata mappings
  - statistical overviews
    - **item counts**
    - **spatiotemporal coverage histograms**
- Example (de-XML-ized)
  - in collection *foo*, the *Originator* bucket is supported and the following item fields are mapped to it:
    - **(FGDC, 1.1/8.1, "Citation/Originator") [973 items]**
    - **(USGS DOQ, PRODUCER, "Producer") [973 items]**
    - **(DC, Creator, "Creator") [1249 items]**
    - **unknown [6 items]**

Searching collections

- Bucket-level
  - uniform across all collections
  - example
    - **search all collections for items whose *Originator* bucket contains the phrase "geological survey"**
- Field-level
  - collection-specific
  - but discovery and invocation mechanisms are uniform
  - functionally equivalent to searching the entire bucket plus additional constraint
  - example
    - **search collection *foo* for items whose *FGDC 1.1/8.1* field within the *Originator* bucket contains the phrase...**

Bucket types (1/7)

- 6 bucket types: spatial, temporal, hierarchical, textual, qualified textual, numeric
- Type captures the portion of the bucket definition that has functional implications
  - data type & syntactic representation
  - query terms
  - query operators
- Complete bucket definition
  - name
  - semantic definition
  - bucket type

Bucket types (2/7)

- Spatial
  - data type: any of several types of geometric regions defined in WGS84 latitude/longitude coordinates
  - syntax: defined by ADEPT
  - query terms: WGS84 box or polygon
  - operators: contains, overlaps, is-contained-in
  - example query:
    - **<spatial-constraint**
      bucket="geographic-location"
      operator="overlaps">
      <box north="37.5" south="30.0" east="-110"
      west="-140"/>
      </spatial-constraint>

Bucket types (3/7)

- Temporal
  - data type: calendar date or range of calendar dates
  - syntax: ISO 8601
  - query term: range of calendar dates
  - operators: contains, overlaps, is-contained-in
  - example query:
    - **<temporal-constraint**
      bucket="coverage-date"
      operator="contains"
      from="1970-01-01" to="1979-12-31"/>

Bucket types (4/7)

- Hierarchical
  - data type: term drawn from a controlled vocabulary (thesaurus, etc.)
  - one-to-one relationship between hierarchical buckets and vocabularies
  - query term: vocabulary term
  - operator: is-a
  - example query:
    - **<hierarchical-constraint**
      bucket="feature-type"
      operator="is-a"
      vocabulary="ADL Feature Type Thesaurus"
      term="populated place"/>

Bucket types (5/7)

- Textual
  - data type: text
  - query term: text
  - operators: contains-all-words, contains-any-words, contains-phrase
  - example query:
    - **<textual-constraint bucket="subject-related-text" operator="contains-all-words" text="orthophotograph"/>**

Bucket types (6/7)

- Qualified textual
  - data type: text with optional associated namespace
  - query term: same
  - query operator: matches
  - example query:
    - **<qualified-textual-constraint**
      bucket="identifier"
      operator="matches"
      text="90-70002-34-5"
      namespace="ISBN"/>

Bucket types (7/7)

- Numeric
  - data type: real number
  - query term: real number
  - query operators: standard relational operators
  - example query:
    - **<numeric-constraint**
      bucket="minimum-feature-size"
      operator="less-than"
      value="1.0"
      unit="meters"/>

Bucket types vs. buckets

- Bucket *types* are defined architecturally
- Buckets in use are defined by collections and items
  - need standard buckets, defined conventionally, to support cross-collection uniformity
- ADL core buckets
  - simple; universal; easily & broadly populated; useful
- Bucket descriptions in the following slides:
  - bucket type
  - semantic definition
  - effective treatment of multiple values in searching
  - comparison to Dublin Core

ADL core buckets (1/6)

- Subject-related text
  - Title
  - Assigned term
- Originator
- Geographic location
- Coverage date
- Object type
- Feature type
- Format
- Identifier

ADL core buckets (2/6)

- Subject-related text
  - type: textual
  - description: text indicative of the subject of the item, not necessarily from controlled vocabularies
  - superset of *Title* and *Assigned term*
  - multiple values: concatenated
  - compare: DC.Subject
- Title
  - type: textual
  - description: the item's title
  - subset of *Subject-related text*
  - multiple values: concatenated
  - compare: DC.Title

ADL core buckets (3/6)

- Assigned term
  - type: textual
  - description: subject-related terms from controlled vocabularies
  - subset of *Subject-related text*
  - multiple values: concatenated
  - compare: qualified DC.Subject
- Originator
  - type: textual
  - description: names of entities related to the origination of the item
  - multiple values: concatenated
  - compare: DC.Creator + DC.Publisher

ADL core buckets (4/6)

- Geographic location
  - type: spatial
  - description: the subset of the Earth's surface to which the item is relevant
  - multiple values: unioned
  - compare: DC.Coverage.Spatial
- Coverage date
  - type: temporal
  - description: the calendar dates to which the item is relevant
  - multiple values: unioned
  - compare: DC.Coverage.Temporal

ADL core buckets (5/6)

- Object type
  - type: hierarchical
  - vocabulary: ADL Object Type Thesaurus (image, map, thesis, sound recording, etc.)
  - multiple values: unioned
  - compare: DC.Type
- Feature type
  - type: hierarchical
  - vocabulary: ADL Feature Type Thesaurus (river, mountain, park, city, etc.)
  - multiple values: unioned
  - compare: none

ADL core buckets (6/6)

- Format
  - type: hierarchical
  - vocabulary: ADL Object Format Thesaurus (loosely based on MIME)
  - multiple values: unioned
  - compare: DC.Format
- Identifier
  - type: qualified textual
  - description: names and codes that function as unique identifiers
  - multiple values: treated separately
  - compare: DC.Identifier

Summary

- A *bucket* is a strongly typed, abstract metadata category with defined search semantics to which source metadata is mapped
- Supports discovery/search across distributed, heterogeneous collections that use metadata structures of their choosing
- Uses high-level search buckets for cross-collection searching and supports "drill-down" searching to the item-level metadata elements

Challenges

- Metadata is like life: it refuses to follow the rules
  - unknown semantics; inconsistent typing/syntax; unknown or unidentifiable sources; poor quality; inconsistent quality; proliferation of overlapping vocabularies; ...
- Realities of the marketplace: Dublin Core won
  - adapt approach to qualified Dublin Core
  - incorporate either fallback mechanism or polymorphism
    - **e.g, treat fields as thesauri/controlled vocabularies or as text**

Query Translation

ADEPT query language

- Domain
  - a collection of items
  - each item has unique ID and 1+ fields
  - field = (name, value)
  - bucket = (name, union or concatenation of fields)
- Queries
  - atomic constraint: (attribute name, operator, target)
    - **semantics: return items that have 1+ values for the attribute, for which at least one value matches the target**
  - arbitrary boolean combinations
    - **AND, OR, AND NOT**

The problem

- Algorithmically translate ADEPT queries to SQL
  - ideally, accommodate all possible SQL implementations
  - configuration must be possible by mere mortals
  - must generate "reasonable" SQL
  - e.g., an unacceptable approach:
    - **(A, op, V) -> SELECT id FROM table WHERE cond(V)**
    - **(A1, op1, V1) B (A2, op2 , V2) ->**
      SELECT id FROM table1 where cond1(V1)
      B
      id IN (SELECT id FROM table2 WHERE cond2(V2))
  - ideally, could incorporate optimization considerations

Approach

- Python-based translator
  - ~1500 lines
- Employs extensible system of "paradigms" for describing atomic translation techniques
  - 15 paradigms
  - Each paradigm ~100 lines (50 Python code, 20 assertions, 30 documentation)
- Uses rules (intrinsic & explicit) to combine booleans
  - preferentially unifies; then JOINs; then self-JOINs, etc.
- Configuration file describes:
  - buckets, fields, paradigms, paradigm configuration
  - boolean override rules
  - misc: external identifier table, optimizer clauses

Translation paradigms

- Paradigm:
  - translateBucketAtomic (constraint) -> query
  - optional:
    - **translateBucketBoolean (boolOp, constraintList)**
    - **translateFieldAtomic, translateFieldBoolean**
    - **"adaptors" for standard field techniques**
- Example: Hierarchical_IntegerSet
  - SELECT id FROM table WHERE column IN (codelist)
  - codelist obtained via separate thesaurus interface
  - configuration: table, id, column, thesaurus info, cardinality
- Cardinality: 1, 1?, 1+, 0+
  - row multiplicity (really functional dependence on identifier)
  - optionality

Intermediate query form

- Query:
  - 1+ tables, expression
  - table: name
  - main table: table + id, cardinality
    - **IDs assumed to be equi-joinable**
  - qualified main table: main table + qualification condition
  - aux table: table + join condition
- Structure necessary for
  - analysis of unification, JOIN possibilities
  - translation correctness
    - **SELECT [t] cond1 …AND NOT…**
      SELECT [t, taux] joincond AND cond2
    - **-> SELECT [t, taux] joincond AND cond1 AND NOT cond2**

Combining queries

- Consider T(v) =
  SELECT id FROM t WHERE c IN (codelist(v))
- T(v1) AND T(v2)
  - if cardinality 1 or 1? can unify:
    SELECT id FROM t WHERE c IN (codelist(v1)) AND c IN (codelist(v2))
  - else: self-join or subquery
- In general:
  - Query({tables}, expression) boolOp
    Query({tables}, expresion)

Future work

- Paradigm system works well
- Boolean processing seems amenable to a more formal treatment
  - should have taken that DB course in college!
- Large, relevant literature
  - Qian & Raschid: algorithmic translation of XSQL to SQL
    - **very complex; not for mere mortals**
  - ADEPT query language is much simpler
    - **and common (Z39.50, WebDAV basicsearch, …)**
- Challenge: generate consistently good SQL
  - stupid things like order of tables & conditions matter
  - make up for DB deficiencies
  - tackle the JOIN problem

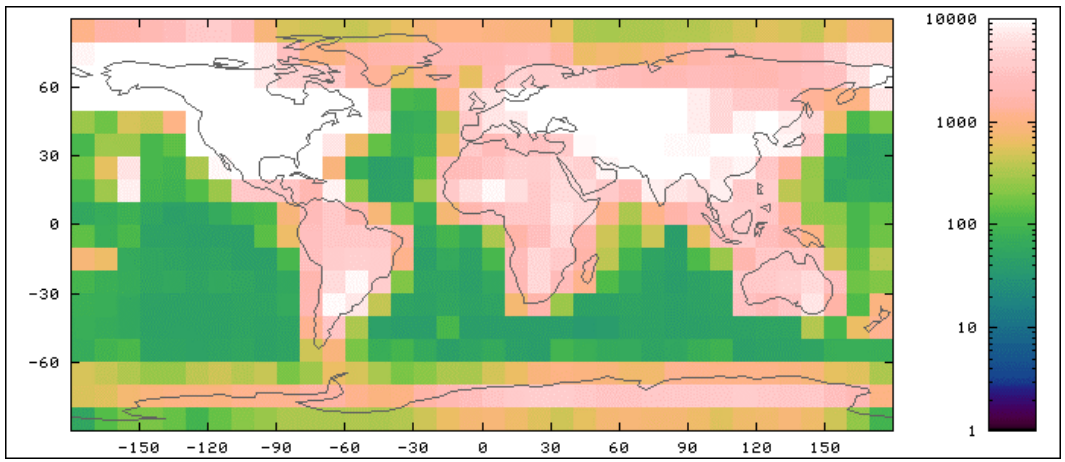
Collection Discovery

The problem

- Distributed queries: necessary evil
  - necessary to achieve scalability
    - **performance**
    - **autonomy**
  - introduce scalability, performance, and reliability problems
- Amelioration strategies
  - increase server performance/reliability
    - **replication, DIENST connectivity regions**
  - turn into offline problem
    - **Web search engines, OAI harvesting model**
  - identify relevant collections to query (ADEPT)
    - **analogous to Web search engine**
- Challenge: identify relevant collections

Approach

- Build on collection-level metadata
  - spatial & temporal density histograms; item counts broken down by collection categorization schemes
    - **more is better**



- Upload periodically to central server
- Replace histograms with Euler histograms to support range queries

Challenges

- Relevance is not necessarily boolean
  - worldwide, petabyte, 1cm resolution database = world map drawn on napkin?
  - introduce resolution/minimum feature size
    - **but sometimes you want the napkin**
- The problem with JOINs
  - statistics are computed independently
- Integrating text overviews
  - STARTS?

Concept Modeling and Education Applications

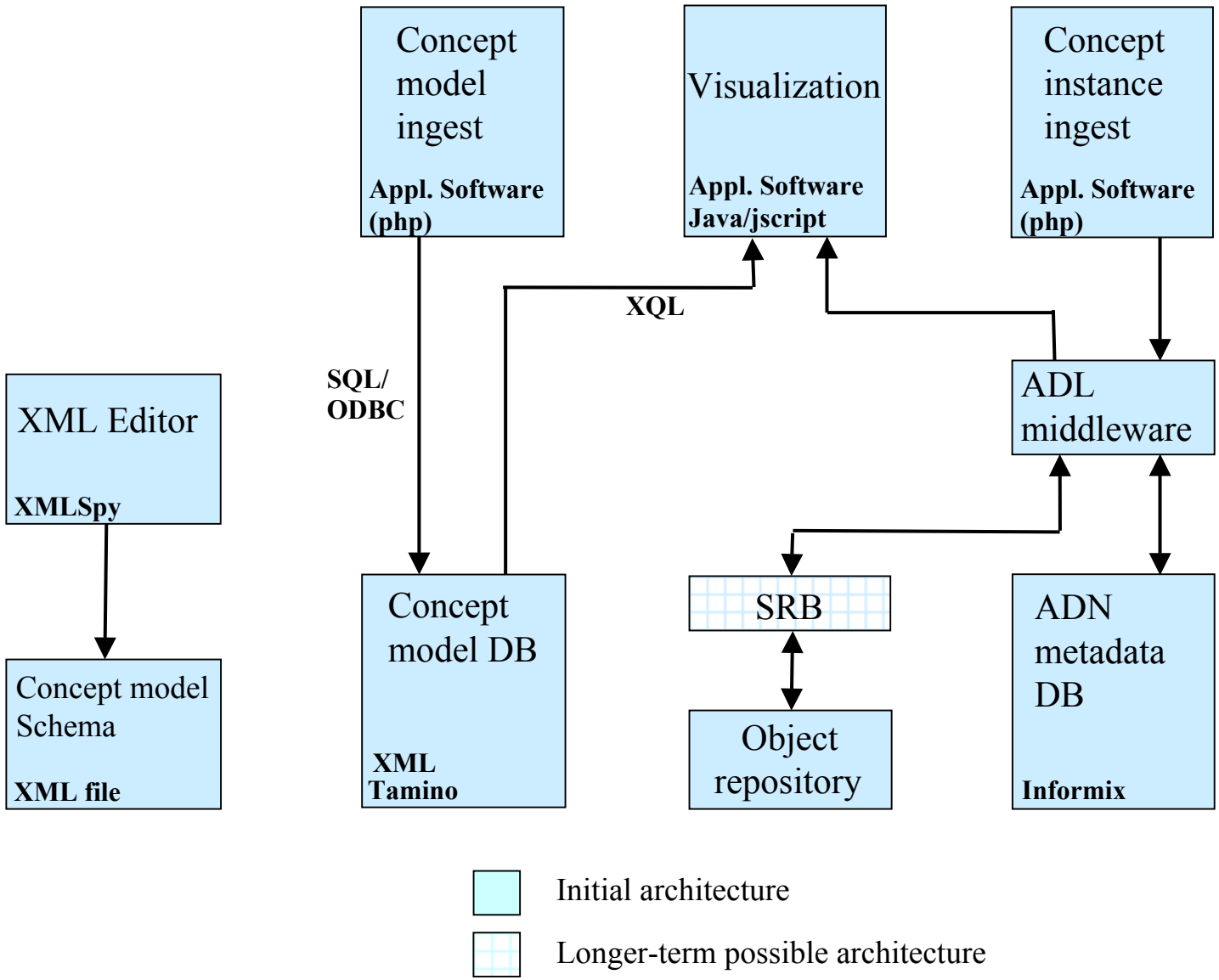
Concept-Based (CB) Learning Spaces: I

- Basic ideas:
  - **Scientific understanding based on large body of concept**
    - ``objective'' representations/interpretations
    - represent phenomena in terms of concepts/relations
    - example of mathematics
  - **Implicit treatment of concepts in learning environments**
    - No explicit model of a concept
      - Glossaries of terms at end of textbook chapters
      - Difficult for students to attain global conceptual views
  - **Structured representations of concepts**
    - Gardenfors models of concepts
      - Connectionist -> concept-level (MD spaces) -> symbolic
    - Concept level representations
      - structured model of concepts
      - inference-rich representations
  - **knowledge base (KB) of concepts as basis for teaching**

Support for CB Learning Spaces

- Organized collections
  - **KB of structured concept representations**
  - **Collections of illustrative materials**
    - Accessible by concept/property
- Services
  - **Creation/editing of KB/collection items**
  - **Search over KB and collections**
    - Real-time access
  - **Graphic/textural views of KB/collections**
    - Highlevel views of topic sequence (course as trajectory)
    - ‘’concept map’’ views of issues within topics
    - Views of concepts and illustrative materials
  - **Creation of Personalized collections**
    - Trajectory through KB/collection

ADEPT Concept Base Architecture version 3: objective for March 31 2002



M Freeston 14 Feb 2002

Concepts: structured representations

- ID
- TERM(S)
- DESCRIPTION(S)
- HISTORICAL ORIGIN(S)
- RELATIONS TO OTHER CONCEPTS
  - **KNOWLEDGE DOMAINS**
  - **SCIENTIFIC USES**
  - **REPRESENTATIONS**
    - Explicit full
    - Explicit partial
    - Implicit full
    - Implicit partial
  - **DEFINFINING OPERATIONS**
  - **PROPERTIES**
  - **HIERARCHIES**
  - **CAUSALITY**
  - **CO-RELATIONS**
- EXAMPLES

Application to learning environments

- Course as a trajectory (personalized collection)
  - **through space of concepts**
  - **through space of illustrative materials**
- Instructor provides framework
  - **Motivations with topics**
  - **Set of topic related issues**
  - **Representing/manipulating representations of issues**
    - Using concepts/illustrative material
- Three concurrent hierarchically-organized views
  - **KB items**
  - **Collection items**
  - **Course/lecture structure**
- Static (trajectory) and dynamic (real-time access) views