LORE: A Database Management System for XML

Jennifer Widom, Roy Goldman, Jason McHugh

Stanford University

http://www-db.stanford.edu/lore

With System Design and Development Help From...

Serge Abiteboul, Brian Babcock, Andre Bergholz, Vineet Gossain, Kevin Haas, Erik Haugen, Matt Jacobson, Dallan Quass, Anand Rajaraman, Hugo Rivero, Michael Rys, Narayanan Shivakumar, Janet Wiener, Raymond Wong, Beverly Yang, Takeshi Yokokawa, Ramon Zazueta

Acronyms

- LORE: Lightweight Object REpository
- LOREL: LORE Language

History

- Project began 4 years ago with a couple of Ph.D. students
- By 1 year ago: a complete Database Management System (DBMS) for semistructured data
  - Hot topic in database research circles
- Today: a complete DBMS for XML
  - Hot topic everywhere

Well-Structured Data

- Today's DBMS's operate on well-structured data
  - Fixed schema (structure) defined in advance
  - All data conforms to schema
  - DBMS needs schema to:
    - Store and index data
    - Process queries and updates
  - Users need schema to formulate queries and updates

Semistructured Data

- Much of today's information is semistructured
  - Lack of regular structure, or structure evolves unpredictably
  - Data may be incomplete
  - DBMS and users don't know complete structure

- Sources of semistructured data
  - Data integration and exchange in heterogeneous environments
  - Data extracted from World-Wide Web
  - XML (eXtensible Markup Language)

Lore Research Areas

- Data model for semistructured data
- Querying semistructured data effectively
  - Query language
  - Formulating queries
  - Browsing results
- Indexing semistructured data
- Query processing and optimization

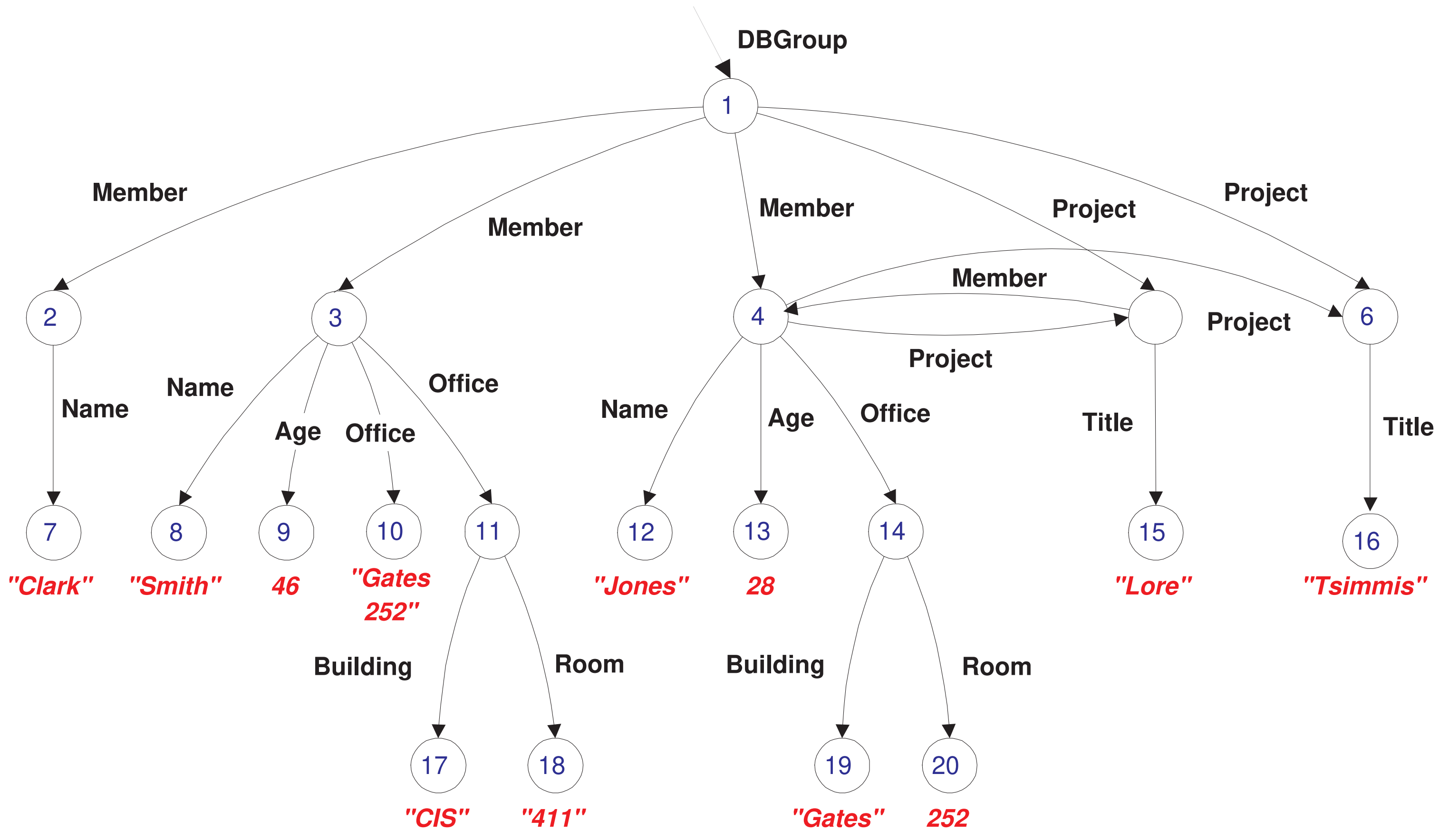
Lore Research Areas (cont'd)

- *DataGuide* – dynamic structural summary
- External data manager
- Keyword and proximity search
- Full migration to XML

Lore's Original Data Model

- Object Exchange Model (OEM)
- Simple nested object model
- Objects are self-describing via labels
- No fixed schema
- Directed labeled graph
  - Nodes are objects
  - Labeled edges denote object-subobject relationship
  - Atomic values in leaves (integer, real, string, gif, avi, etc.)

Example Database



Portion of Example Database in XML

```
<MEMBER>
   <NAME>Clark</NAME>
</MEMBER>
<MEMBER>
   <NAME>Smith</NAME>
   <AGE>46</AGE>
   <OFFICE>Gates 252</OFFICE>
   <OFFICE>
      <BUILDING>CIS</BUILDING>
      <ROOM>411</ROOM>
   </OFFICE>
</MEMBER>
<MEMBER>
   <NAME>Jones</NAME>
   <AGE>28</AGE>
   <OFFICE>
      <BUILDING>Gates</BUILDING>
      <ROOM>252</ROOM>
   </OFFICE>
</MEMBER>
<PROJECT>
   <TITLE>Lore</TITLE>
</PROJECT>
...
```

How OEM Differs from XML

- OEM mixes subelements, attributes, and IDREF's
- OEM is unordered (though Lore preserves ordering)
- OEM is schema-less: no Document Type Definitions (DTD's)

⇒ We are migrating Lore to an XML-oriented graph-based data model

LOREL – The LORE Language

- Basic principles
  - No errors (hints and warnings OK)
  - Gracefully handles irregular and incomplete data
  - User need not know full object structure
- Extension of OQL
  - No classes, no strict type-checking
  - Extensive automatic type coercion
    - Atomic values
    - Sets and singletons
  - Heterogeneous sets
  - Very general path expressions

Lorel: Query Examples

- Simple example

```
select DBGroup.Member.Project
where  DBGroup.Member.#.(Office%|Room%) like "%252"
```

- More complex example

```
select m.Name, (select p1.Title
                from m.Project p1
                where p1 <> p)
from   DBGroup.Member m, m.Project p
where  p.Title hasword "XML"
```

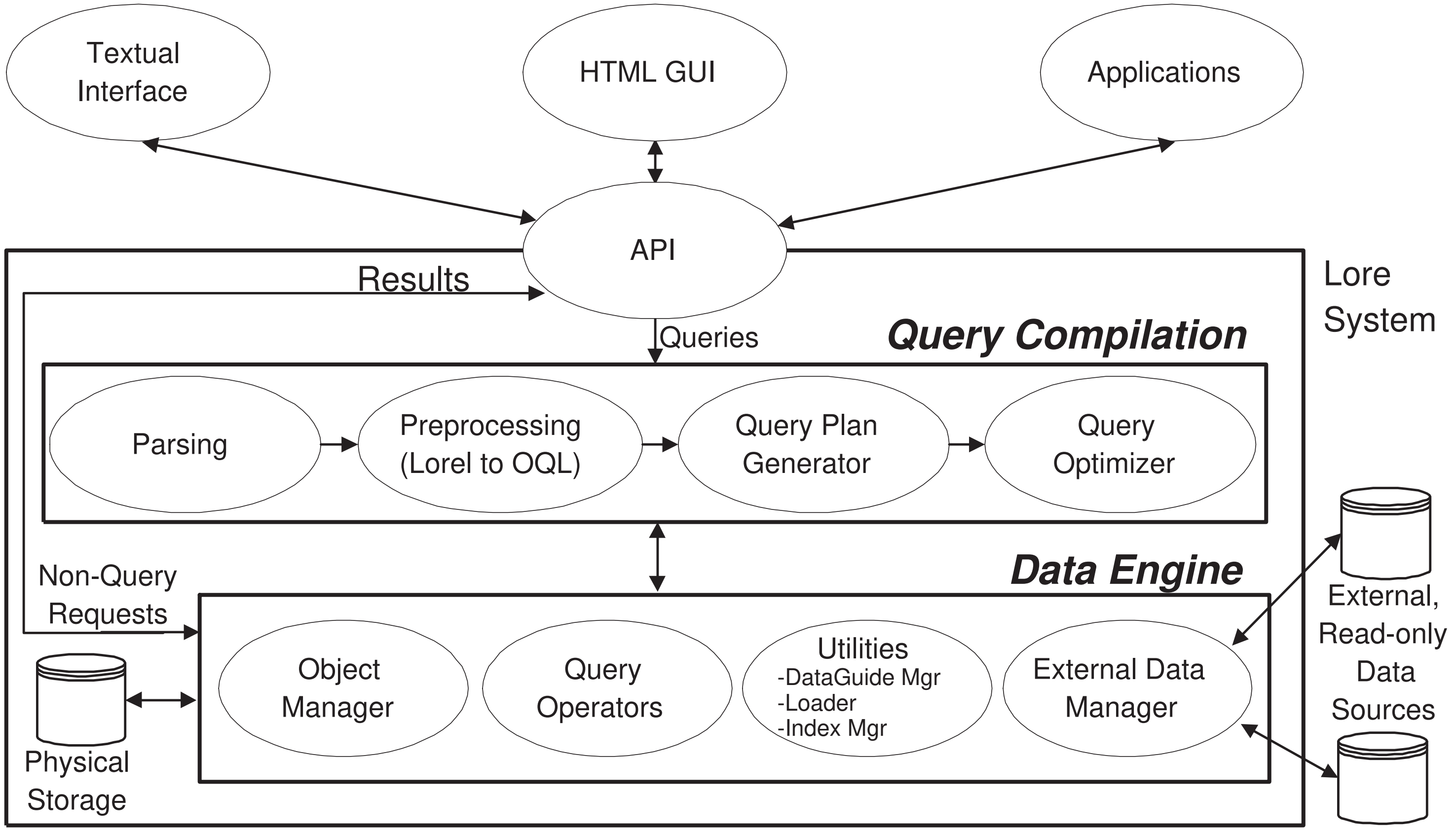
Lorel (cont'd)

- Declarative update language

```
update p.Member +=
  (select DBGroup.Member
   where DBGroup.Member.#.(Office%|Room%) like "%252")
from   DBGroup.Project p
where  p.Title hasword "XML"
```

- Also existential and universal quantification, label and path variables, grouping and aggregation, external functions and predicates, materialized views, bulk loading, ...

System Architecture



Indexing in Traditional Database Systems

- Relational systems provide attribute indexes
  - Quickly find all student tuples with `GPA < 2.0`
- Object-oriented systems add path indexes
  - Quickly find all student objects with `Advisor.Dept.Name = "CS"`

Indexing in Lore

- Indexes created/destroyed at any time to speed up query processing
- Value indexes
  - All atomic objects matching some predicate
  - Restricted by incoming label
  - Example: all `...(Score|Grade) > 95`
  - Multi-index scheme to accommodate coercion
- Link indexes
  - All [L-labeled] parents of a given object
  - Serves as back-pointers

Indexing in Lore (cont'd)

- Edge indexes
  - All L-labeled edges in database
- Text indexes
  - Full-text index over string values, IR-style `hasword` operator
  - Restricted by incoming label
- Path indexes
  - All objects via given path
  - Example: all `DBGroup.Project.Publication.Author`
  - Provided by DataGuide

Query Execution

- Basic goal: match path expressions in queries to paths in data

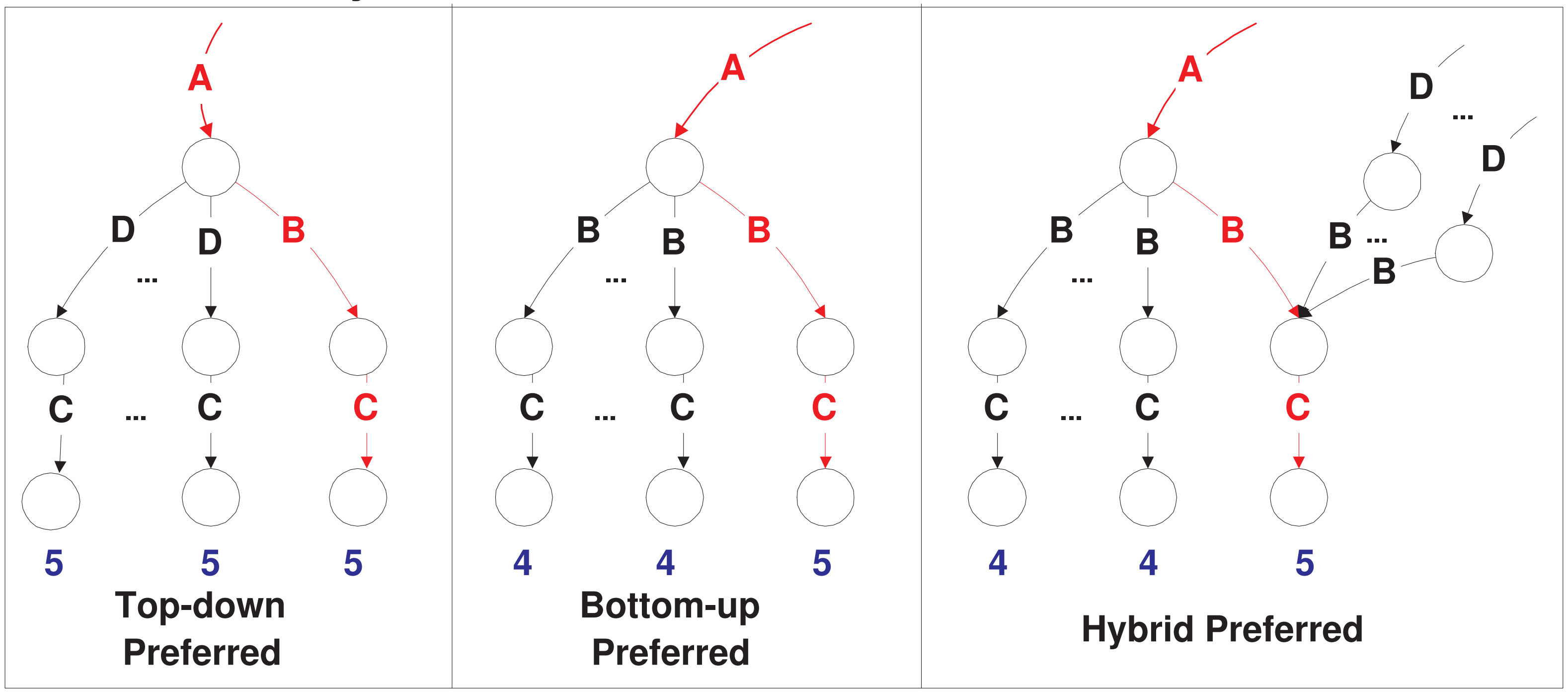
```
select DBGroup.Member.Project
where  DBGroup.Member.#.(Office%|Room%) like "%252"
```

- Naive approach: exhaustive top-down traversals

- Indexes enable many other approaches
  - Bottom-up
  - Hybrid
  - Inside-out
  - Path indexes

Database Shapes

Find all objects reachable via A.B.C whose value is 5



Query Optimization

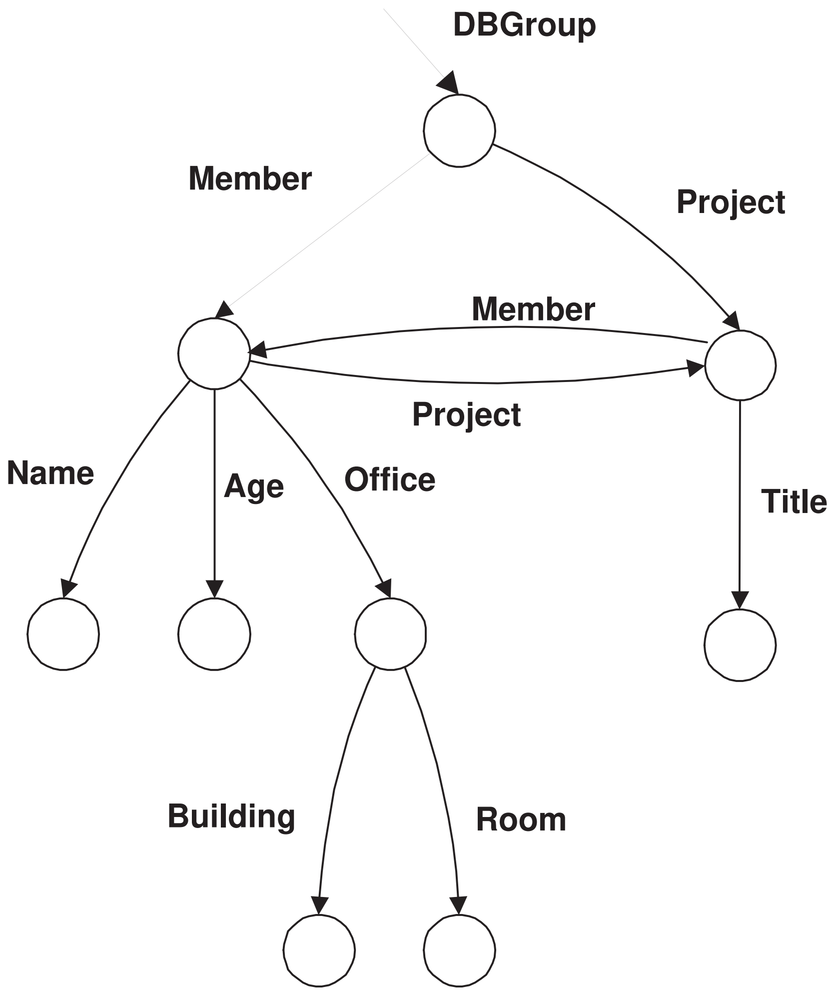
- Find "optimal" execution plan for given query and database
  - Optimal = lowest predicted running time
- Lore query optimizer
  - Cost-based, estimates # of page I/O's
  - Flexible logical query plans
  - Wide variety of candidate physical plans
  - Some locally optimal decisions, other heuristics
  - New kinds of statistics, kept in DataGuide

DataGuides

- No fixed schema

- DataGuide: dynamic structural summary of current database
  - Each label path in database appears once in DataGuide
  - No extraneous paths in DataGuide
  - Maintained incrementally as database evolves
  - Serves role of schema

DataGuide for Example Database



DataGuide Uses

- Stores statistics for query optimizer
- Serves as path index
- Central to graphical user interface
  - To browse structure
  - To see sample atomic values and statistics
  - To formulate queries "by example"

External Data Manager

- "Virtual" database objects from outside information sources
- Fetched on demand during query processing, cached for future queries
- Distinction between Lore-resident and external data invisible to user
- *Descriptors* stored in Lore
- *Arguments* sent to sources based on query, local data
- Optimizations to minimize external calls, fetched data

External Data Manager – Example

- Group member data within Lore
  - `Name`, `MachineName`
  - External `Status` objects with idle time, mail read, etc.

- Query

```
select DBGroup.Member.Status.Idle
where  DBGroup.Member.Name = "Jason"
```

- External call

```
status jason@shark
```

Keyword and Proximity Search

- Casual users like to type keywords
- Near operator important
  - Want proximity in graph, not in document
- Ex: FIND Publication NEAR Widom
  - Ranks objects matching keyword Publication by closeness to objects matching keyword Widom
  - Specialized *distance index*
- In the works: integration with Lorel

Lore System Status

- Approx. 120,000 lines of C++ code
- Multi-platform, multi-user
- Robust on most features described
- Demo and executables available on-line

  http://www-db.stanford.edu/lore

Current and Future Work

- Lots of XML stuff
  - Subelements vs. attributes vs. IDREF's
  - Ordering
  - DTD's
  - Data vs. documents
  - Etc...
- More query optimization
- *Approximate* DataGuides
- Clustering, other physical storage issues
- Integrating semistructured and structured data

Demos

- Lore – three ways to query
  - Using Lorel
  - "By example" using DataGuide
  - Using keyword/proximity search

- Microsoft auction demo
  - Original demo creates XML tree in memory, performs queries with hard-coded traversals, sends updates to relational back-end
  - Our demo uses Lore as XML store and query/update engine