Lore: A Database Management System for Semistructured Data

Authors from Stanford University
Introduction

- Traditional DBMS are relational
- Force data to adhere to rigid schema
- Data can be irregular (null values?)
- Difficult to decide in advance of single correct schema
- Where should we store unstructured data?!
In Comes Lore

- Takes advantage of structure where it exists
- Handle irregular/unstructured data gracefully
- Uses the OEM (Object Exchange Model) as the data model
- Uses the Lorel query language
- Uses DataGuides in place of a standard schema
This Presentation

- OEM (Object Exchange Model)
- The Lorel Query Language (and OQL)
- High Level System Architecture
- Query Plans and Data Flow
- Query Operators
- Query Plan Construction
- Query Optimization and Indexing
- Index and Update Query Plans
- Physical Storage
- Data Guides
The Object Exchange Model

- Labeled directed graph
- Atomic objects - leaf/edge vertices
- Complex objects - vertices with outgoing edges
- *Names* serve as entry points

Figure 1: An OEM database
The Lorel Query Language

- Simple Path Expressions
  - DBGroup.Member.Office

- Rewritten into OQL style

```sql
QUERY
select DBGroup.Member.Office
where DBGroup.Member.Age > 30

select O
from DBGroup.Member M, M.Office O
where exists A in M.Age : A > 30
```
The Lorel Query Language

**QUERY**

```
select DBGroup.Member.Office
where DBGroup.Member.Age > 30
```

**RESULT**

- Office "Gates 252"
- Office
- Building "CIS"
- Room "411"

*Figure 1: An OEM database*
The Lorel Query Language

- Complex Path Expressions

```
QUERY
    select DBGroup.Member.Name
    where DBGroup.Member.Office(.Room|.Cubicle)?
        like "%252"

RESULT
    Name "Jones"
    Name "Smith"
```
The Lorel Query Language

- Subqueries

```plaintext
QUERY
   select M.Name,
       ( select M.Project.Title
           where M.Project.Title != "Lore" )
   from DBGroup.Member M
   where M.Project.Title = "Lore"

RESULT
   Member
      Name "Jones"
      Title "Tsimmis"
```
The Lorel Query Language

- Updates
  - insertion/removal of edges
  - creation of vertices
  - modifications of atomic values
  - modifications of name assignments
  - no object deletion (handled by garbage collector)

```plaintext
update P.Member +=
  ( select DBGroup.Member
    where DBGroup.Member.Name = "Clark"
  )
from DBGroup.Project P
where P.Title = "Lore" or P.Title = "Tsimmis"
```
High Level System Architecture

Figure 2: Lore architecture
Query Plans and Data Flow

**QUERY**

```sql
SELECT DBGroup.Member.Office
WHERE DBGroup.Member.Age > 30
```

- Execution begins at the top
- Iterator approach avoid creation of temporary relations
- Each OA slot holds the oid of a vertex

**Figure 3**: Example Lore query plan

<table>
<thead>
<tr>
<th>OA0</th>
<th>OA1</th>
<th>OA2</th>
<th>OA3</th>
<th>OA4</th>
</tr>
</thead>
<tbody>
<tr>
<td>(DBGroup)</td>
<td>(OA0.Member)</td>
<td>(OA1.Office)</td>
<td>(OA1.Age)</td>
<td>(true/false)</td>
</tr>
</tbody>
</table>
Query Operators

**QUERY**

```sql
select DBGroup.Member.Office
where DBGroup.Member.Age > 30
```

- Scan
- Join
- Select
- Aggregate
- Project
- SetOp
- ArithOp
- CreateSet
- GroupBy

![Diagram of query plan](image)

**Figure 3:** Example Lore query plan

<table>
<thead>
<tr>
<th>OA0 (DBGroup)</th>
<th>OA1 (OA0.Member)</th>
<th>OA2 (OA1.Office)</th>
<th>OA3 (OA1.Age)</th>
<th>OA4 (true/false)</th>
</tr>
</thead>
</table>

**Figure 4:** Example object assignment
select M.Name, count(M.Publication) from DBGroup.Member M where M.Dept = "CS"
Query Optimization and Indexing

- Lacks sophisticated query planning
- Selections are pushed down
- Two types of indexes:
  - Lindex (Parent Link Index)
  - Vindex (Value Index)
- Lindexes implemented using linear hashing
- Vindexes implemented using B+-Trees

<table>
<thead>
<tr>
<th>arg1</th>
<th>arg2</th>
<th>string</th>
<th>real</th>
<th>int</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td></td>
<td></td>
<td>string → real</td>
<td>both → real</td>
</tr>
<tr>
<td>real</td>
<td></td>
<td></td>
<td>-</td>
<td>int → real</td>
</tr>
<tr>
<td>int</td>
<td>both</td>
<td>real</td>
<td>int → real</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 1: Coercion for basic comparison operators
Index Query Plans

**QUERY**

```sql
select DBGroup.Member.Office
where DBGroup.Member.Age > 30
```

![Diagram of query plan](image)

*Figure 6: A query plan using indexes*
Update Query Plans

```
update P.Member +=
    ( select DBGroup.Member
        where DBGroup.Member.Name = "Clark"
    )
from DBGroup.Project P
where P.Title = "Lore" or
    P.Title = "Tsimmis"
```

**Figure 7: Example update query plan**

Query plan to find all projects with the title "Lore" or "Tsimmis", results placed in OA1

Query plan to find all members with name "Clark", results placed in OA5
Physical Storage

- Each page on disk has slots
- One object in each slot
- First-fit algorithm used
- Object forwarding mechanism
- Large objects span many pages
- Object clustering is depth first
- Garbage collector for orphans
- External data also supported

Figure 8: The logical and physical views of the data
Figure 9: A DataGuide for Figure 1
Thank You! Questions?