An Approach for Efficient Querying of Large Relational Datasets with OCL-based Languages

Dimitrios S. Kolovos
Ran Wei
Konstantinos Barmpis
{dimitris.kolovos, rw542, kb634}@york.ac.uk
Motivation

• Data used in MDE likely found in non-model artefacts:
  – Spreadsheets
  – Databases
  – XML documents

• Such data needs to be converted for use in model transformations & queries
The ATM System

<table>
<thead>
<tr>
<th>origin</th>
<th>dest</th>
<th>depTime</th>
<th>arrTime</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABE</td>
<td>ATL</td>
<td>1557</td>
<td>1812</td>
<td></td>
</tr>
<tr>
<td>ABQ</td>
<td>BWI</td>
<td>0735</td>
<td>1252</td>
<td></td>
</tr>
<tr>
<td>ANC</td>
<td>ADQ</td>
<td>0804</td>
<td>0915</td>
<td></td>
</tr>
<tr>
<td>AZA</td>
<td>DEN</td>
<td>1556</td>
<td>1731</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

- 1 Table (Flight)
- > 200 Columns
- > 500,000 Rows
The ATM System

| origin | dest | depTime | arrTime | ...
|-------|------|---------|---------|-----
| ABE   | ATL  | 1557    | 1812    | ...
| ABQ   | BWI  | 0735    | 1252    | ...
| ANC   | ADQ  | 0804    | 0915    | ...
| AZA   | DEN  | 1556    | 1731    | ...

![ATM System Diagram]

- DTW
- ORD
- AUS
- SFO
- 550
- 518
- 620
- 589
- 126
- M2T
- M2M
- MV

29/09/2013
XM'13 Miami
The ATM System

| origin | dest | depTime | arrTime | ...
|--------|------|---------|---------|-----
| ABE    | ATL  | 1557    | 1812    |     |
| ABQ    | BWI  | 0735    | 1252    |     |
| ANC    | ADQ  | 0804    | 0915    |     |
| AZA    | DEN  | 1556    | 1731    |     |
| ...    | ...  | ...     | ...     |     |

Diagram showing flight connections with airports.

- DTW
- ORD
- AUS
- SFO
- M2T
- M2M
- MV

JAVA

Model

Airport

Route

numberOfFlights : EInt

outgoing 0..*

incoming 0..*

origin 0..1

destination 0..1

name : EString

Table: Flight Information

| origin | dest | depTime | arrTime | ...
|--------|------|---------|---------|-----
| ABE    | ATL  | 1557    | 1812    |     |
| ABQ    | BWI  | 0735    | 1252    |     |
| ANC    | ADQ  | 0804    | 0915    |     |
| AZA    | DEN  | 1556    | 1731    |     |
| ...    | ...  | ...     | ...     |     |
The ATM System

<table>
<thead>
<tr>
<th>origin</th>
<th>dest</th>
<th>depTime</th>
<th>arrTime</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABE</td>
<td>ATL</td>
<td>1557</td>
<td>1812</td>
</tr>
<tr>
<td>ABQ</td>
<td>BWI</td>
<td>0735</td>
<td>1252</td>
</tr>
<tr>
<td>ANC</td>
<td>ADQ</td>
<td>0804</td>
<td>0915</td>
</tr>
<tr>
<td>AZA</td>
<td>DEN</td>
<td>1556</td>
<td>1731</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
The ATM System

| origin | dest | depTime | arrTime | ...
|--------|------|---------|---------|------
| ABE    | ATL  | 1557    | 1812    | ...  |
| ABQ    | BWI  | 0735    | 1252    | ...  |
| ANC    | ADQ  | 0804    | 0915    | ...  |
| AZA    | DEN  | 1556    | 1731    | ...  |
| ...    | ...  | ...     | ...     | ...  |

Route
- numberOfFlights : EInt
  - outgoing 0..*
    - origin 0..1
    - destination 0..1

Airport
- name : EString
  - airports 0..*

Model

Diagram with nodes and edges representing flight connections and distances:
- DTW connected to ORD with distance 550
- ORD connected to SFO with distance 518
- SFO connected to AUS with distance 620
- AUS connected to M2T with distance 589
- M2M with distance 126

Nodes: M2T, M2M, MV
# The Epsilon Modeling Suite & EOL

<table>
<thead>
<tr>
<th>Task-specific languages</th>
<th>Model Refactoring (EWL)</th>
<th>Pattern Matching (EPL)</th>
<th>Model Validation (EVL)</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model Comparison (ECL)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Model Merging (EML)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Code Generation (EGL)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Model Migration (Flock)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The Epsilon Modeling Suite & EOL

<table>
<thead>
<tr>
<th>Task-specific languages</th>
<th>Model Refactoring (EWL)</th>
<th>Pattern Matching (EPL)</th>
<th>Model Validation (EVL)</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model Comparison (ECL)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Model Merging (EML)</td>
<td></td>
<td>Code Generation (EGL)</td>
<td>Model Migration (Flock)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Technology-specific drivers</th>
<th>Eclipse Modeling Framework (EMF)</th>
<th>Schema-less XML</th>
<th>Relational Store</th>
<th>NoSQL Store</th>
</tr>
</thead>
<tbody>
<tr>
<td>Meta Data Repository (MDR)</td>
<td>CSV</td>
<td>Bibtex</td>
<td>MetaEdit+</td>
<td>...</td>
</tr>
</tbody>
</table>
The Epsilon Modeling Suite & EOL

<table>
<thead>
<tr>
<th>Task-specific languages</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Model Refactoring (EWL)</td>
<td>Pattern Matching (EPL)</td>
</tr>
<tr>
<td>Model Comparison (ECL)</td>
<td>Model-to-model Transformation (ETL)</td>
</tr>
<tr>
<td>Model Merging (EML)</td>
<td>Code Generation (EGL)</td>
</tr>
</tbody>
</table>

Epsilon Object Language (EOL) \(\approx\) JavaScript + OCL

Epsilon Model Connectivity (EMC)

<table>
<thead>
<tr>
<th>Technology-specific drivers</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Eclipse Modeling Framework (EMF)</td>
<td>Schema-less XML</td>
</tr>
<tr>
<td>Meta Data Repository (MDR)</td>
<td>CSV</td>
</tr>
</tbody>
</table>

29/09/2013
XM'13 Miami
Challenges (1)

Taking the following OCL-like expression to retrieve the number of distinct airports:

\[ \text{Flight.allInstances.origin.asSet().size()} \]

We would need to:

1. Inspect the model and compute a collection of all model elements of type Flight;
2. Iterate through the contents of the collection (from step 1) and collect the values of the property \textit{origin} in a new collection;
3. Remove all duplicates from the collection (from step 2);
4. Compute the size of the collection computed in step 3.
Challenges (2)

The following issues arise if the information is stored in a relational database:

• Computing the *Flight.allInstances* collection requires the engine to perform a:

  \[
  \text{select } * \text{ from } \text{Flight}
  \]

  SQL query. For large tables (such as Flight) the returned set needs to be streamed from the database.

• Such streamed sets restrict us to:
  – Forward-only iteration
  – Size can only be calculated after exhaustive iteration
  – Only 1 set can be streamed at a time in a MySQL store.
Challenges (3)

The following issues arise if the information is stored in a relational database:

• The next step would be to iterate through all the rows of the Flight table through the streamed set and collect the values of origin.

• This is inefficient as using a:

  
  select origin from Flight

SQL statement would be orders of magnitude faster.
Challenges (4)

The following issues arise if the information is stored in a relational database:

- Eliminating duplicates is similarly inefficient and can be easily done using a
  
  \[
  \text{select distinct origin from Flight}
  \]

  SQL statement.

- Calculating the size of a streamed result-set without invalidating the result-set itself is an issue. By contrast, this could be computed in one step using a:

  \[
  \text{select count(distinct origin) from Flight.}
  \]

  SQL statement.
Solutions (1)

Calculate the average delay of flights flying from JFK to LAX on Sundays:

```
Flight.allInstances
  .select(f | f.origin="LAX")
  .select(f | f.dest="JFK"
      and f.dayOfWeek=1)
  .collect(f | f.delay)
  .avg()
```
Solutions (1)

Calculate the average delay of flights flying from JFK to LAX on Sundays:

```scala
Flight.allInstances
  .select(f | f.origin="LAX")
  .select(f | f.dest="JFK"
       and f.dayOfWeek=1)
  .collect(f | f.delay)
  .avg()
```
Solutions (1)

Calculate the average delay of flights flying from JFK to LAX on Sundays:

```java
Flight.allInstances
    .select(f | f.origin="LAX")
    .select(f | f.dest="JFK"
             and f.dayOfWeek=1)
    .collect(f | f.delay)
    .avg()
```
Calculate the average delay of flights flying from JFK to LAX on Sundays:

```
Flight.allInstances
  .select(f | f.origin="LAX")
  .select(f | f.dest="JFK" and f.dayOfWeek=1)
  .collect(f | f.delay)
  .avg()
```
Solutions (1)

Calculate the average delay of flights flying from JFK to LAX on Sundays:

```java
Flight.allInstances
    .select(f | f.origin="LAX")
    .select(f | f.dest="JFK" and f.dayOfWeek=1)
    .collect(f | f.delay)
    .avg()
```
Calculate the average delay of flights flying from JFK to LAX on Sundays:

```java
Flight.allInstances
  .select(f | f.origin="LAX")
  .select(f | f.dest="JFK" and f.dayOfWeek=1)
  .collect(f | f.delay)
  .avg()
```
Solutions (1)

Calculate the average delay of flights flying from JFK to LAX on Sundays:

```sql
Flight.allInstances
  .select(f | f.origin="LAX")
  .select(f | f.dest="JFK" and f.dayOfWeek=1)
  .collect(f | f.delay)
  .avg()
```

select avg(delay) from Flight where (origin="LAX") and (dest="JFK" and dayOfWeek=1)
Solutions (2)

EOL Engine Extension for SQL:

**.allInstances** Returns a streamed lazy collection (**ResultSetList**) backed by a `select * from <table>` SQL expression.

**.select(<iterator>|<condition>)** Translates the EOL condition to an SQL expression, and returns a new **ResultSetList**. Similarly for `exists()`, `forall()` and `reject()` OCL operations.

**.collect(<iterator>|<expression>)** Returns a streamed lazy collection of primitive values (**PrimitiveValuesList**). Calls to the `size()` method are interpreted as count SQL queries.

**asSet()** Returns a new **PrimitiveValuesList** backed by a distinct SQL query.
The ATM System

<table>
<thead>
<tr>
<th>origin</th>
<th>dest</th>
<th>depTime</th>
<th>arrTime</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABE</td>
<td>ATL</td>
<td>1557</td>
<td>1812</td>
</tr>
<tr>
<td>ABQ</td>
<td>BWI</td>
<td>0735</td>
<td>1252</td>
</tr>
<tr>
<td>ANC</td>
<td>ADQ</td>
<td>0804</td>
<td>0915</td>
</tr>
<tr>
<td>AZA</td>
<td>DEN</td>
<td>1556</td>
<td>1731</td>
</tr>
</tbody>
</table>

- DTW
- ORD
- AUS
- SFO
- M2T
- M2M
- MV
Extracted Facts

Analysis of this dataset reveals:

• Of the 306 airports, 68 (>20%) are connected directly to only 1 other airport;

• The most distant pair of airports are ABE and BRW. A passenger needs to change 4 flights (ABE-DTW-SEA-FAI-BRW);

• The Atlanta International Airport (ATL) is the busiest airport (# of flights going through it - 67,717), followed by ORD and DFW;

• ATL is the best-connected airport with direct flights to 148 other airports;

• >50% of all the flights go through the 18 busiest airports & >90% of all flights go through the 91 busiest airports.
Conclusion & Further Work

• MDE can greatly benefit from using technologies outside MOF and EMF
• If integrated correctly, relational datasets can be used to contain model data
• The challenges lay in identifying and optimising the way such stores are queried

• We aim at investigating the impact of compile-time static analysis on performance
• We aim at supporting multi-table querying (and hence transformations) by use of foreign keys
Questions?