A Categorical Framework on Multi-Model Databases

Jiaheng Lu
University of Helsinki, Finland

Oracle ST-Seminar  23.08.2019
Category theory can be used to model multi-model databases.

- Category theory provides a unified data model for multi-model databases.

- Categorical framework can provide powerful query language for multi-model DB.
A grand challenge on Variety

- Big data: Volume, Variety, Velocity, Veracity

- Variety: hierarchical data (XML, JSON), graph data (RDF, property graphs, networks), tabular data (CSV), etc
Motivation: one application to include multi-model data

- Relational data: customer databases
- Graph data: social networks
- Hierarchical data: catalog, product
- Text data: customer review

An E-commerce example with multi-model data
An example of multi-model data and query

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Credit</th>
<th>Phone</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Mary</td>
<td>5,000</td>
<td>{ &quot;type&quot;: &quot;home&quot;, &quot;number&quot;: &quot;212 555-1234&quot; }</td>
</tr>
<tr>
<td>2</td>
<td>John</td>
<td>3,000</td>
<td>{ &quot;type&quot;: &quot;fax&quot;, &quot;number&quot;: &quot;646 555-4567&quot; }</td>
</tr>
<tr>
<td>3</td>
<td>William</td>
<td>2,000</td>
<td>{ &quot;type&quot;: &quot;work&quot;, &quot;number&quot;: &quot;2198 -111-4321&quot;, &quot;1&quot; --&gt; &quot;34e5e759&quot;, &quot;2&quot; --&gt; &quot;0c6df508&quot; }</td>
</tr>
</tbody>
</table>

```json
{ "Order_no":"0c6df508", "Orderlines": [ { "Product_no":"2724f", "Product_Name":"Toy", "Price":66 }, { "Product_no":"3424g", "Product_Name":"Book", "Price":40 } ] }
```
Q: Return all products which are ordered by a friend of a customer whose credit > 3000

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Credit</th>
<th>Phone</th>
<th>Order No</th>
<th>Orderlines</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Mary</td>
<td>5,000</td>
<td>{ &quot;type&quot;: &quot;home&quot;, &quot;number&quot;:</td>
<td>34e5e759</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>&quot;212 555-1234&quot; }</td>
<td></td>
<td>[{ &quot;Product_no&quot;: &quot;2724f&quot;,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>&quot;Product_Name&quot;: &quot;Toy&quot;,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>&quot;Price&quot;: 66 },</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>{ &quot;Product_no&quot;: &quot;3424g&quot;,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>&quot;Product_Name&quot;: &quot;Book&quot;,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>&quot;Price&quot;: 40 } ]</td>
</tr>
<tr>
<td>2</td>
<td>John</td>
<td>3,000</td>
<td>{ &quot;type&quot;: &quot;fax&quot;, &quot;number&quot;:</td>
<td>0c6df508</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>&quot;646 555-4567&quot; }</td>
<td></td>
<td>[{ &quot;Product_no&quot;: &quot;2724f&quot;,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>&quot;Product_Name&quot;: &quot;Toy&quot;,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>&quot;Price&quot;: 66 },</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>{ &quot;Product_no&quot;: &quot;3424g&quot;,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>&quot;Product_Name&quot;: &quot;Book&quot;,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>&quot;Price&quot;: 40 } ]</td>
</tr>
<tr>
<td>3</td>
<td>William</td>
<td>2,000</td>
<td>{ &quot;type&quot;: &quot;work&quot;, &quot;number&quot;:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>&quot;2198-111-4321&quot; }</td>
<td></td>
<td>[{ &quot;Product_no&quot;: &quot;2724f&quot;,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>&quot;Product_Name&quot;: &quot;Toy&quot;,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>&quot;Price&quot;: 66 },</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>{ &quot;Product_no&quot;: &quot;3424g&quot;,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>&quot;Product_Name&quot;: &quot;Book&quot;,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>&quot;Price&quot;: 40 } ]</td>
</tr>
</tbody>
</table>
Let CustomerIDs = (FOR Customer IN Customers FILTER Customer.CreditLimit > 3000 RETURN Customer.id)

Let FriendIDs = (FOR CustomerID in CustomerIDs FOR Friend IN 1..1 OUTBOUND CustomerID Knows return Friend.id)

For Friend in FriendIDs
For Order in 1..1 OUTBOUND Friend Customer2Order
Return Order.orderlines[*].Product_no

Recommendation query:
Return all products which are ordered by a friend of a customer whose credit>3000
Multi-model databases

- A multi-model database is designed to support multiple data models against a single, integrated backend.

- Document, graph, relational and key-value models are examples of data models that may be supported by a multi-model database.
Multi-model databases:
One size fits multi-data-model
Need theoretical foundation for multi-model databases

Call for a unified theory for manipulating & transforming multi-model data

The theory of relations (150 years old) needs to be extended to mathematically describe transformation in multi-model databases
Categories Defined

A category $C$ is ….

- a collection of objects $\text{ob}(C)$ .. {X, Y, Z ....}
- a collection of morphisms \{f, g ....\}
  - A set of morphisms from object X into Y is denoted by $\text{Hom}(X, Y)$ or $X \rightarrow Y$.

\[
\begin{array}{ccc}
X & \xrightarrow{f} & Y \\
| & (g \circ f) & \downarrow g \\
& Z & \\
\end{array}
\]
Categories Defined (con’t)

- The category must satisfy the following rules

  Composition of morphisms:

  - for any three objects X, Y, and Z, we have
  - If \( f: X \rightarrow Y, \ g: Y \rightarrow Z, \) then there is a composed morphism \( f \circ g: X \rightarrow Z. \)

  - Having associative and identity laws
An example of multi-model data

**Table:**

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Credit</th>
<th>Phone</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Mary</td>
<td>5,000</td>
<td>{ &quot;type&quot;: &quot;home&quot;, &quot;number&quot;: &quot;212 555-1234&quot; }</td>
</tr>
<tr>
<td>2</td>
<td>John</td>
<td>3,000</td>
<td>{ &quot;type&quot;: &quot;fax&quot;, &quot;number&quot;: &quot;646 555-4567&quot; }</td>
</tr>
<tr>
<td>3</td>
<td>William</td>
<td>2,000</td>
<td>{ &quot;type&quot;: &quot;work&quot;, &quot;number&quot;: &quot;2198 -111-4321&quot; }</td>
</tr>
</tbody>
</table>

```
"1" -> "34e5e759"
"2" -> "0c6df508"
```

```json
{"Order_no":"0c6df508", "Orderlines": [ {
    "Product_no":"2724f",
    "Product_Name":"Toy",
    "Price":66 },
    {
    "Product_no":"3424g",
    "Product_Name":"Book",
    "Price":40 } ]}
```
Categorical model for multi-model data

- Customer
  - Phone type
  - Phone number
  - Buy
- Product
  - Price
- Name
- Credit
- Friend
**Instance categorical model**

Morphisms:
- Friend: 1 → 2, 3 → 2;
- Name: 1 → Mary, 2 → John, 3 → William ……
Difference between category model and Entity-Relation model (I)

1. Category model supports the composition of morphisms (relations)

- Friend (A,B), Friend(B,C) ⇒ Friend (A,C)

- Can define the reachability query in graph databases

- Can define the descendant axes in XML databases
SQL and logic

- E.F. Codd, 1970:
- A SQL query can be defined as a first order logic.

- For example: “List the set of employees’ names and their department whose chair is named Cher”

- **Select** Employee_name
- **From** Dep, Employee
- **Where** Employee.depID=Dep.ID and Dep.chair=“Cher”

- $\exists x, \exists y, \text{Dep}(x,"Cher") \land \text{Employee}(x,y)$
• Fagin, 1976: **Graph reachability is not expressible in first-order logic!**

There is no first-order formula $\phi(x, y)$ that says there is a path in graph $G$ from node $x$ to node $y$.

Categorical framework can present the reachability query.

• $\exists x, \exists y, \exists f, f(x,y)$
  where $f$ is a composed morphism in the category
Computation complexity

- Categorical query can support other complicated path query:
- Find persons $x$ and $z$ such that $x$ is parent of $y$ and $x$ know one of teachers of $z$ and $y$ (recursively) knows $z$
- $\exists x, \exists z \ p(x,y) \land k^*(y,z) \land (k \circ t)(x,z)$

<table>
<thead>
<tr>
<th>Query fragment</th>
<th>Evaluation</th>
<th>Containment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Categorical Path Query</td>
<td>NLOGSPACE-complete</td>
<td>PSPACE-complete</td>
</tr>
<tr>
<td>Categorical Path Query with Counting</td>
<td>#P-complete</td>
<td>???</td>
</tr>
</tbody>
</table>
2. Category model can define exponential object.

Given two objects, an exponential object is an object \( X^Y \) equipped with an evaluation map \( \text{ev}: X^Y \times Y \rightarrow X \) which is universal in the sense that, given any object \( Z \) and map \( e: Z \times Y \rightarrow X \), there exists a unique map \( u: Z \rightarrow X^Y \).
Difference between category model and Entity-Relation model (II)
2. Category model can define exponential object. Exponential object is different from the Relation attribute in ER.

Exponential object includes all morphisms between two objects, but relation attribute defines only one morphism.
Multi-rounds of games

<table>
<thead>
<tr>
<th>Player</th>
<th>Card</th>
<th>Start_time</th>
<th>End_time</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>9♥, 10♣</td>
<td>10.00</td>
<td>10.02</td>
</tr>
<tr>
<td>P2</td>
<td>7♠</td>
<td>10.02</td>
<td>10.03</td>
</tr>
</tbody>
</table>

Project 1

Round n

<table>
<thead>
<tr>
<th>Player</th>
<th>Card</th>
<th>Start_time</th>
<th>End_time</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>5♦</td>
<td>11.00</td>
<td>11.01</td>
</tr>
<tr>
<td>P2</td>
<td>A♥</td>
<td>11.02</td>
<td>11.03</td>
</tr>
</tbody>
</table>
ER for Multi-rounds of games

Project 1

Persons
Start_time

Cards
End_time

Project n
Category model for Multi-rounds of games

- Category model can define exponential object.

![Category model diagram]

- Players
- Jobs
- Cards
- Project Name
- Project leader
- Persons
Complicated query with exponential object

- Categorical query can support some complicated queries with exponential object
- Find two projects where nobody is assigned the same job
- Let \( \text{ev}: (\text{Cards}^{\text{Players}} \times \text{Players}) \rightarrow \text{Cards} \)
  - \( \exists r_1, \exists r_2, \forall p, \text{ev}(r_1,p) \neq \text{ev}(r_2,p) \land \text{E}(r_1) \land \text{E}(r_2) \)
  - \( \text{E} \) stands for the exponential object \( \text{Cards}^{\text{Players}} \)
- (This is NOT a rigorous formal expression. We need to introduce Heyting algebra to give a formal definition)
XPath 3.1 with higher-order function and categorical query

$$\exists r_1, \exists r_2, \forall p, \ f(r_1, p) \neq f(r_2, p) \land E(r_1) \land E(r_2)$$

E stands for the exponential object $\text{Cards}^{\text{Players}}$

Let $b := \text{TRUE}$ for each pair $(\text{game/round/player/cards}, \text{game/round/player/cards}, \text{fn}(a, b))$

(if ($a = b$) then let $b := \text{FALSE}$)

Return $b$
The category $C$ is called Cartesian closed if and only if it satisfies the following properties:

- Any two objects $X$ and $Y$ of $C$ have a product $X \times Y$ in $C$.
- Any two objects $Y$ and $Z$ of $C$ have an exponential $Z^Y$ in $C$.

A multi-model database can be modeled as a Cartesian closed category.
Query category with higher order logic

- SQL query is the first-order logic query language. SQL can be extended

- Based on Cartesian closed category (and topos theory), we can define higher-order logic query for XPath 3.1 and reachability graph query.

- Categorical framework build a unified query language

- More powerful, but more expensive to compute
Limitation of this category framework

• Provide a unified framework to incorporate different data models together

• But no hints for efficient algorithms

• Many problems are NP-hard, EXPTIME or even undecidable

• Efficient implementation and tractable subclasses for the future work
Related systems and work

- **Multidatabase systems** (or federated systems)
  - a few databases (less than 10)
  - Powerful queries (transaction and updates)

- **Object Relation database systems**
  - supports extension of the data model with custom data types and methods
  - Relation model is the first-class citizen, and ad-hoc support for other models of data

- **Polyglot persistence**
  - Integrated access to multiple, heterogeneous cloud data stores, such as NoSQL, HDFS, and RDBMS
Related systems and work (con’t)

- **Category theory and databases**
  - David I. Spivak propose: category = database schema
  - The data is a functor: $C \rightarrow \text{Set}$
  - But our work focuses on multi-model data and the unified query language
Conclusion

- Category theory can serve as a theoretical framework for multi-model databases

- Powerful high-order logic query interface
Conclusion (Cont.)

• We will demo how to use a functional programming language **Haskell** to query multi-model data

• Category theory is a theoretical foundation for functional programming.
Acknowledgement

• Thanks for the review and feedback from Dieter Gawlick, Zhen Hua Liu, Souripriya Das, Heli Helskyaho and Greg Pogossiants

• This research is partially funded by Academy of Finland and Oracle gift award.
References


4) Chao Zhang, Jiaheng Lu, Pengfei Xu, Yuxing Chen: UniBench: A Benchmark for Multi-model Database Management Systems. TPCTC 2018: 7-23 (Benchmark system work)

