ER 2019

Salvador, Bahia, Brazil

Unified Management of Multi-Model Data

Irena Holubová, Martin Svoboda, Jiaheng Lu

svoboda@ksi.mff.cuni.cz

November 7, 2019

Charles University, Prague, Czech Republic

University of Helsinki, Helsinki, Finland

Introduction

Motivation

- **Multi-model data**
  - We often need to work with **multiple logical models** at the same time within a given application / information system
  - This brings a **non-trivial complexity**

Objective

- Illustrate the reasons for this complexity
  - Using practical examples
- **Identify key challenging research areas**
  - So that they can be appropriately figured out

Data Variety

- **Logical models**
  - Relational, key-value, wide column, document, graph, …
- **Data formats**
  - XML or JSON for the document model
- **Schemas**
  - DTD or XML Schema schema languages
- **Vocabularies**
  - Names of XML elements or attributes
- **Technologies**
  - Databases, protocols, interfaces, …
- **Query languages**
  - Syntax, constructs, expressive power

RDF Store

Sample **RDF** data in Turtle notation

- Data about products, their names and other features

```
@prefix p: <http://www.myshop.cz/products/> .
@prefix c: <http://www.myshop.cz/countries/> .
@prefix i: <http://www.myshop.cz/schema#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
p:banana18 rdf:type i:Product ;
           i:name "Cavendish Banana" ;
           i:producer c:India , c:Ecuador , c:China ;
           i:color "yellow" .
p:melon5 rdf:type i:Product ;
         i:name "Watermelon" ;
         i:producer c:China , c:Turkey ;
         i:color "red" .
p:melon13 rdf:type i:Product ;
          i:name "Cantaloupe Melon" ;
          i:producer c:China , c:Iran ;
          i:color "orange" .
```

RDF Store

Sample **SPARQL** query

- Items produced in China or Egypt

```
PREFIX c: <http://www.myshop.cz/countries/>
PREFIX i: <http://www.myshop.cz/schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?color ?product ?name
FROM <http://www.myshop.cz/products>
WHERE
  {
    ?product rdf:type i:Product ;
             i:name ?name ;
             i:producer ?country ;
             i:color ?color .
    FILTER (?country = c:China || ?country = c:Egypt)
  }
ORDER BY ASC(?color) DESC(?name)
```

Relational Database

Sample **relational** data

- Data about changes in the stock of products

product	date	time	quantity	unit
melon5	2019-08-15	13:45:00	150	kg
banana18	2019-08-15	13:45:30	50	kg
melon5	2019-08-15	15:15:00	-5	kg
melon5	2019-08-15	18:00:00	-2	kg
banana18	2019-08-15	18:30:00	-4	kg
melon13	2019-08-16	09:15:00	30	pc
melon5	2019-08-16	11:15:00	-2	kg
melon13	2019-08-16	11:15:00	-1	pc
banana18	2019-08-16	11:15:00	-2	kg

Relational Database

Sample **SQL** query

- Overall quantities of sold items during August 2019

```
SELECT
  product,
  SUM(ABS(quantity)) AS sales,
  unit
FROM
  stock
WHERE
  (YEAR(date) = 2019) AND (MONTH(date) = 8) AND
  (quantity < 0)
GROUP BY
  product, unit
ORDER BY
  sales DESC, product ASC
```

JSON Database

Sample **JSON** data in **MongoDB** database

- Data about registered clients

```
{
  _id: "client32",
  name: { first: "Jane", last: "Williams" },
  age: 25,
  email: [ "jane@company.com", "williams@hotel.org" ]
}
```

```
{
  _id: "client26",
  name: { first: "Peter", last: "Smith" },
  age: 30,
  email: [ "peter@somewhere.net" ],
  phone: "+420 777 123 456",
  address: {
    street: "Long 35", city: "Prague", zip: "12116", country: "CZE"
  }
}
```

JSON Database

Sample **MongoDB** query

- Clients older than 20 years from Prague in the Czech Republic

```
db.clients.find(
  {
    age: { $gt : 20 },
    address: {
      $elemMatch: { city: "Prague", country: "CZE" }
    }
  },
  {
    _id: false, name: true, address: true
  }
).sort(
  {
    name.last: 1, name.first: -1
  }
)
```

XML Database

Sample **XML** data

- Data about purchases made

```
<?xml version="1.1" encoding="UTF-8"?>
<orders>
  <order id="order105" date="2019-08-15" time="15:15:00">
    <client ref="client32">Jane Williams</client>
    <items>
      <item product="melon5" qty="5" unit="kg" name="Watermelon"/>
    </items>
  </order>
  <order id="order127" date="2019-08-16" time="11:15:00">
    <client ref="client26">Peter Smith</client>
    <items>
      <item product="melon5" qty="2" unit="kg" name="Watermelon"/>
      <item product="melon13" qty="1" unit="pc" name="Cantaloupe Melon"/>
      <item product="banana18" qty="2" unit="kg" name="Cavendish Banana"/>
    </items>
  </order>
</orders>
```

XML Database

Sample **XQuery** query

- HTML table with statistics of sold products

```
<table>
  <tr>
    <th>Product</th><th>Name</th><th>Quantity</th>
  <tr>
  {
    for $product in distinct-values(/orders/order/items/item/@product)
    let $items := //item[@product = $product]
    let $quantity := sum($items/@qty)
    order by $quantity descending, $product ascending
    return
      element tr {
        <td>{ $product }</td>,
        <td>{ data(($items)[1]/@name) }</td>,
        <td>{ $quantity }</td>
      }
  }
</table>
```

Graph Database

Sample **property graph** data in **Neo4j** database

- Data about clients and their orders

```
(p1:PRODUCT { id: "banana18", name: "Cavendish Banana", color: "yellow" })
(p2:PRODUCT { id: "melon5", name: "Watermelon", color: "red" })
(p3:PRODUCT { id: "melon13", name: "Cantaloupe Melon", color: "orange" })
```

```
(c1:CLIENT { id: "client32", name: "Jane Williams", age: 25 })
(c2:CLIENT { id: "client26", name: "Peter Smith", age: 30 })
```

```
(c1)-[e1:PURCHASE { quantity: 5, unit: "kg" }]->(p2)
(c2)-[e2:PURCHASE { quantity: 2, unit: "kg" }]->(p1)
(c2)-[e3:PURCHASE { quantity: 2, unit: "kg" }]->(p2)
(c2)-[e4:PURCHASE { quantity: 1, unit: "pc" }]->(p3)
```

Graph Database

Sample **Cypher** query

- Names of clients with above average purchases of watermelons

```
MATCH (c:CLIENT)-[e:PURCHASE]->(p:PRODUCT)
  WHERE p.id = "banana18"
WITH avg(e.quantity) AS average
MATCH (c:CLIENT)-[e:PURCHASE]->(p:PRODUCT { id: banana18" })
  WHERE e.quantity > average
RETURN c.name
  ORDER BY c.age DESCENDING, c.name ASCENDING
```

Existing Strategies

Polyglot persistence

- Different databases for different data models
- Accessed independently or using an integrating mediator
- E.g. DBMS+, BigDAWG

Multi-model databases

- One database for multiple different data models
- Provides a fully integrated backend
- More than 20 representatives
- E.g. OrientDB, ArangoDB, MarkLogic, Virtuoso, …

Open Problems

Main issues of multi-model databases

- Specifics of the **original underlying model**
  - Existing solutions…
    - originate mainly from the IT industry
    - were originally single-model systems
    - only later were adapted to other models
    - and so are determined and limited by these models
- Support for true **cross-model processing**
  - Varies greatly
    - Query constructs, index structures, query optimization, …
- Lack of **necessary formal background**
  - Data model itself
  - Syntax and semantics of the query language

Key Challenges

Only too many **models, formats, technologies, query languages**, …

- $\Rightarrow$ only too high **complexity**
- $\Rightarrow$ not sustainable in long-term perspective
- $\Rightarrow$ **unification** is essential

Challenging areas

- Conceptual modeling
- Schema inference
- Unified querying
- Evolution management
- Autonomous systems

Conceptual Modeling

Existing top-down approaches

- **UML**
  - Standardized, but data-oriented and concealing details
- **ER**
  - Several notations, constructs better grasping the real-world

Observations

- **Common principles** but also **specifics** of individual models
- **Links** between distinct models
  - I.e. foreign keys, pointers, references, …

Objectives

- **Unified conceptual modeling approach for multi-model data**
- **Mapping rules and transformations for individual models**

Schema Inference

Levels of schema support

- **Schema-full**
  - Description of data structure is provided explicitly
  - Its requirements must be satisfied
- **Schema-less**
  - Schema is neither provided nor required
    - However, in reality, **implicit schema** exists nevertheless

Observations

- Possible extensions of **single-model solutions**
  - Not straightforward because of the **links**
  - We also want near **real-world schemas**

Objective

- **Universal multi-model schema inference method**

Unified Querying

Existing query languages

- **Standardized**, **proprietary**
- **Single-model**, attempts of **multi-model** languages
- **Expressive power** varies greatly
- **Specifics**, but also **common principles**
  - Tables as results in SQL but also SPARQL or Cypher
  - Statement **clauses**
    - Names and purpose often inspired by SQL
    - Structure often fixed, almost arbitrary chaining in Cypher
  - **Sub-query expressions** in SQL or SPARQL
  - **Functional querying** in XPath or XQuery

Objective

- **Unified conceptual query language for multi-model data**

Evolution Management

Observations

- **Structure of data can change in time**
  - $\Rightarrow$ impact on data instances, indices, query expressions, …
- Manual approach
  - Skilled database administrators required
  - Error-prone job
- Existing **single-model solutions**
  - Can be used for **intra-model** changes
  - Cannot be easily extended to **inter-model** changes

Objectives

- **Multi-model data evolution management framework**
- **Operations for semi-automatic propagation of changes**

Autonomous Systems

We can go even one step further…

- …towards **autonomous management** of data
  - Selection of suitable logical models
  - Handling of data transformations
  - Evolution of data changes
  - Etc.
- It should be the **responsibility** of the database system itself to **find the best way how data should be organized**

Objective

- **Autonomous multi-model database management technique**

Conclusion

Variety of multi-model data

- Logical models, data formats, schemas, vocabularies, database representatives, query languages

Identified **challenges** for **unified** processing of **multi-model data**

- Conceptual modeling
- Schema inference
- Unified querying
- Evolution management
- Autonomous systems

General requirements

- **Formal background**, **practical impact**, **user-friendliness**

Thank you for your attention…