Towards Schema-independent Querying on Document Data Stores

H. BEN HAMADOU¹, F. GHOZZI², A. PENINOU¹, O. TESTE¹

¹IRIT, Université de Toulouse - France UT3, UT2J
²MIRACL, Université de Sfax - Tunisie ISIMS

hamdi.ben-hamadou@irit.fr

26-03-2018, DOLAP’18

Document-oriented Database

- **Data format**: Semi-structured documents, JSON, BSON . . .
- **Data model**: Schema-less
- **Advantage**: Big data support, Scalability, Availability
- **Example**: MongoDB, CouchDB
- **Applications**: Web, IoT, social media . . .
- **Interrogation**: JDBC, Drivers, API, Command line . . .

```json
{
  "firstName": "John",
  "lastName": "Smith",
  "age": 25,
  "address": {
    "streetAddress": "121 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021"
  },
  "phoneNumber": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "fax",
      "number": "646 555-4567"
    }
  ]
}
```
Modeling Multi-structured Data

Collection

\[ C = \{d_1, \ldots, d_c\} \]

Document

\[ d_i = (k_i, v_i) \]

- \( k_i \) is the document's identifier.
- \( v_i = \{a_{i,1} : v_{i,1}, \ldots, a_{i,n} : v_{i,n}\} \) is the document's value.

Document Schema

\[ s_i = \{p_1, \ldots, p_m\} \]

where \( p_i \) is a path leading to leaf node in document \( d_i \).

Collection Schema

\[ S = \bigcup_{i=1}^{C} s_i \]

Structural Heterogeneity

Document 1

```
{  
   "_id": 1,  
   "title": "Fast and furious",  
   "year": 2017,  
   "language": "English"
}
```

Document 2

```
{  
   "_id": 2,  
   "title": "Titanic",  
   "details":  
      {  
         "year": 1997,  
         "language": "English"
      }
}
```

Document 3

```
{  
   "_id": 3,  
   "title": "Despicable Me 3",  
   "year": 2017
}
```

Document 4

```
{  
   "_id": 4,  
   "title": "The Hobbit",  
   "versions":  
      [{  
         "year": 2012,  
         "language": "English"
      },  
      {  
         "year": 2013,  
         "language": "French"
      }]
}
```
Query Operators

Kernel of Unary Operators

\[ k = \{ \pi, \sigma \} \]

Projection Operator

\[ \pi(A)(C_{in}) = C_{out} \]

The project operator reduces the initial schemas of documents to a finite subset of attributes \( A \).

Selection Operator

\[ \sigma(P)(C_{in}) = C_{out} \]

The select operator retrieves only documents that match the selection condition \( P \) expressed in normal form (\( Norm_p \)).

Querying Multi-structured Data Problem

\[ \pi(\text{"title"}, \text{"year"})(C) \]

Document 1

```
{ 
  "_id": 1,
  "title": "Fast and furious",
  "year": 2017,
  "language": "English"
}
```

Document 2

```
"_id": 2,
"title": "Titanic",
"details": {
  "year": 1997,
  "language": "English"
}
```

Document 3

```
{ 
  "_id": 3,
  "title": "Despicable Me 3",
  "year": 2017
}
```

Document 4

```
{ 
  "_id": 4,
  "title": "The Hobbit",
  "versions": {
    "year": 2012,
    "language": "English"
  },
  { 
    "year": 2013,
    "language": "French"
  }
}
```
Querying Multi-structured Data Problem

\[ \pi(\text{"title"}, \text{"year"}) \](C)

Document 1

\{ 
  "_id": 1,
  "title": "Fast and furious",
  "year":2017
  "language":"English"
\}

Document 2

\{ 
  "_id": 2,
  "title": "Titanic",
  "details":
   { 
      "year":1997
      "language":"English"
   }
\}

Document 3

\{ 
  "_id": 3,
  "title": "Despicable Me 3",
  "year":2017
\}

Document 4

\{ 
  "_id": 4,
  "title": "The Hobbit",
  "versions":
    [{
      "year":2012
      "language":"English"
    },
    { 
      "year":2013
      "language":"French"
    }]
\}

\[ \pi(\text{"title"}, \text{"year"},\text{"details.year"}, \text{"versions.1.year"}, \text{"versions.2.year"}) \](C)
Querying Multi-structured Data Problem

\[ \pi(\text{"title"}, \text{"year"}, \text{"details.year"}, \text{"versions.1.year"}, \text{"versions.2.year"})(C) \]

**Document 1**

```
{  
  "_id": 1,  
  "title": "Fast and furious",  
  "year":2017  
  "language": "English" 
}
```

**Document 2**

```
{  
  "_id": 2,  
  "title": "Titanic",  
  "year":1997  
  "language": "English" 
}
```

**Document 3**

```
{  
  "_id": 3,  
  "title": "Despicable Me 3",  
  "year":2017 
}
```

**Document 4**

```
{  
  "_id": 4,  
  "title": "The Hobbit",  
  "versions":  
  [{  
      "year":2012  
      "language": "English" 
    },  
    {  
      "year":2013  
      "language": "French" 
    }  
}
```

---

**Plan**

1. Introduction
2. Querying Heterogeneous Documents
3. Experiments
4. Conclusion & perspectives
Physical data transformation

- Flattening data.
- Using additional databases.
- Introducing new structures.

\[ ([\text{Chasseuretal.}, 2013], (\text{Taharaetal.}, 2014)(\text{Taharaetal.}, 2014)] \]
⇒ Need to learn new schema.
⇒ Loss of initial document schemas/structures.
⇒ Need to re-build new schemas when structures are changed.

Virtual data transformation

- Inferring existing schemas.
- Building an unified schema.
- Tracking different schemas versions.

\[ ([\text{Baazizi et al.}, 2017],[\text{Ruiz et al.}, 2015],[\text{Wang et al.}, 2015]) \]
⇒ Need to learn new structures.
⇒ Querying is only limited to structural level.
⇒ Heterogeneity is manually managed to formulate application queries.
The dictionary $\text{dict}_C$ constructed from a collection $C$ is defined by

$$\text{dict}_C = \{(p_k, \triangle_k) \mid \forall p_k \in S_C\}$$

- $p_k \in S_C$ is a path leading to a leaf node which is present in at least one document;
- $\triangle_k = \{p_{p_k,1}, \ldots, p_{p_k,q}\} \subseteq S_C$, is a set of navigational paths leading to $p_k$;
Dictionary Construction Process

```
dict = { "year", "details.year", "versions.1.year", "versions.2.year" }
```
Dictionary

dict = {
  "title", {"title"} },
  "year", {"year", "details.year", "versions.1.year", "versions.2.year"},
  "language", {"language", "details.language", "versions.1.language", "versions.2.language"},
  "details", {"details"},
  "details.year", {"details.year"},
  "details.language", {"details.language"},
  "versions", {"versions"},
  "versions.1", {"version.1"},
  "versions.1.year", {"versions.1.year"},
  "versions.1.language", {"versions.1.language"},
  "versions.2", {"versions.2"},
  "versions.2.year", {"versions.2.year"},
  "versions.2.language", {"versions.2.language"}
}

---

**Algorithm for Automatic Query Extension**

**Algorithm 1:** Automatic extension for the original user’s query

**Input:** \( Q \)

**Output:** \( Q_{ext} \)

\[
Q_{ext} \leftarrow \text{id} \quad \text{// identity}
\]

**foreach** \( q_i \in Q \) **do**

**switch** \( q_i \) **do**

**case** \( \pi_{A_i} \)

**do**

\[
A_{ext} \leftarrow \bigcup_{\forall a_k \in A_i} \triangle_k; \quad Q_{ext} \leftarrow Q_{ext} \circ \pi_{A_{ext}}
\]

**end**

**case** \( \sigma_{\text{Norm}_p} \)

**do**

\[
P_{ext} \leftarrow \bigwedge_k \left( \bigvee_l V_{a_j \in \triangle_k, l} a_j \varpi_{k, l} (a_l) \right); \quad Q_{ext} \leftarrow Q_{ext} \circ \sigma_{P_{ext}}
\]

**end**

**end**
Extending project operator

\textbf{Algorithm 1: Automatic extension for the original user's query}

\begin{algorithmic}
  \Require $Q$
  \Ensure $Q_{\text{ext}}$
  \State $Q_{\text{ext}} \leftarrow \text{id}$  \hfill // identity
  \ForEach{$q_{i} \in Q$}
    \Switch{$q_{i}$}
      \Case{$\pi_{A_{i}}$}  \hfill // Projection operation
        \State $A_{\text{ext}} \leftarrow \bigcup_{\forall a_{k} \in A_{i}} \Delta_{k}$
        \State $Q_{\text{ext}} \leftarrow Q_{\text{ext}} \circ \pi_{A_{\text{ext}}}$
      \EndCase
      \Case{$\sigma_{\text{Norm}_{p}}$}  \hfill // $\text{Norm}_{p} = \bigwedge_{j}(\forall j \ a_{i,j} \ \omega_{i,j} \ v_{i,j})$ selection operation
        \State $P_{\text{ext}} \leftarrow \bigwedge_{k} \left( \forall j \ \forall a_{j} \in \Delta_{k,l} \ a_{j} \ \omega_{k,l} \ v_{k,l} \right)$
        \State $Q_{\text{ext}} \leftarrow Q_{\text{ext}} \circ \sigma_{P_{\text{ext}}}$
      \EndCase
    \EndSwitch
  \EndFor
\end{algorithmic}

\textbf{Attributes extensions}

$A_{\text{ext}} \leftarrow \bigcup_{\forall a_{k} \in A_{i}} \Delta_{k}$

\textbf{Example}

$\pi(\text{"title"}, \text{"year"})(C)$
Extending project operator

\[ \pi(\text{"title"}, \text{"year"})(C) \]

\[ \text{dict} = \{ \]
\[ (\text{"title"}, \{\text{"title"}\}), \]
\[ (\text{"year"}, \{\text{"year"}, \text{"details.year"}, \text{"versions.1.year"}, \text{"versions.2.year"}\}) \]
\[ (\text{"language"}, \{\text{"language"}, \text{"details.language"}, \text{"versions.1.language"}, \text{"versions.2.language"}\}) \]
\[ (\text{"details"}, \{\text{"details"}\}), \]
\[ (\text{"details.year"}, \{\text{"details.year"}\}), \]
\[ (\text{"details.language"}, \{\text{"details.language"}\}), \]
\[ (\text{"versions"}, \{\text{"versions"}\}), \]
\[ (\text{"versions.1"}, \{\text{"version.1"}\}), \]
\[ (\text{"versions.1.year"}, \{\text{"versions.1.year"}\}), \]
\[ (\text{"versions.1.language"}, \{\text{"versions.1.language"}\}), \]
\[ (\text{"versions.2"}, \{\text{"versions.2"}\}), \]
\[ (\text{"versions.2.year"}, \{\text{"versions.2.year"}\}), \]
\[ (\text{"versions.2.language"}, \{\text{"versions.2.language"}\}) \] \]

Attributes extensions

\[ A_{\text{ext}} \leftarrow U_{\forall k \in A_i} \Delta_k \]

Example

\[ \pi(\text{"title"}, \text{"year"})(C) \]

\[ A_{\text{ext}} \leftarrow \{\text{"title"}\} \cup \{\text{"year"}, \text{"details.year"}, \text{"versions.1.year"}, \text{"versions.2.year"}\} \]

Projection query extended

\[ \Rightarrow \pi(\text{"title"}, \text{"year"}, \text{"details.year"}, \text{"versions.1.year"}, \text{"versions.2.year"})(C) \]
Extending Select Operator

**Algorithm 1**: Automatic extension for the original user’s query

Input: \( Q \)
Output: \( Q_{\text{ext}} \)

\[
Q_{\text{ext}} \leftarrow \text{id} \quad // \text{identity}
\]

\[
\text{foreach } q_i \in Q \text{ do}
\]

\[
\switch q_i \text{ do}
\]

\[
\text{case } \pi A_i \quad // \text{Projection operation}
\]

\[
\text{do}
\]

\[
A_{\text{ext}} \leftarrow \bigcup_{k, l \in A_i} \Delta_k ; \quad Q_{\text{ext}} \leftarrow Q_{\text{ext}} \circ \pi A_{\text{ext}}
\]

\[
\text{end}
\]

\[
\text{case } \sigma_{\text{Norm}} \quad // \text{Norm} = \bigwedge_i \left( \bigvee_j a_{i,j} \land a_{i,j} \lor v_{i,j} \right) \text{selection operation}
\]

\[
\text{do}
\]

\[
P_{\text{ext}} \leftarrow \bigwedge_k \left( \bigvee_l \bigvee_{a_j \in \Delta_k} a_j \land a_{k,l} \land v_{k,l} \right) ; \quad Q_{\text{ext}} \leftarrow Q_{\text{ext}} \circ \sigma P_{\text{ext}}
\]

\[
\text{end}
\]

\[
\text{end}
\]

**Attributes extensions**

\[
P_{\text{ext}} \leftarrow \bigwedge_k \left( \bigvee_l \bigvee_{a_j \in \Delta_k} a_j \land a_{k,l} \land v_{k,l} \right)
\]

**Example**

\[
\sigma(“title” \neq \text{Null} \land “language” = “English”) (C)
\]
Extending Select Operator

Extending Selection’s Predicates

\[ P_{\text{ext}} \leftarrow \bigwedge_k \left( \bigvee_l \bigvee_{a_j \in \Delta_{k,l}} a_j \not\equiv_{k,l} v_{k,l} \right) \]

Example

\[ \sigma \left( \text{"title"} \neq \text{Null} \land \text{"language"} = \text{"English"} \right)(C) \]

Selection query extended

\[ P_{\text{ext}} \leftarrow \left( \bigvee_{a_j \in \Delta(\text{"title"})} a_j \not\equiv \text{Null} \right) \land \left( \bigvee_{a_j \in \Delta(\text{"language"})} a_j = \text{"English"} \right) \Rightarrow \sigma(P_{\text{ext}})(C) \]

dict = {
    ("title", {"title"}),
    ("year", {"year", "details.year", "versions.1.year", "versions.2.year"}),
    ("language", {"language", "details.language", "versions.1.language", "versions.2.language"}),
    ("details", {"details"}),
    ("details.year", {"details.year"}),
    ("details.language", {"details.language"}),
    ("versions", {"versions"}),
    ("versions.1", {"version.1"}),
    ("versions.1.year", {"versions.1.year"}),
    ("versions.1.language", {"versions.1.language"}),
    ("versions.2", {"versions.2"}),
    ("versions.2.year", {"versions.2.year"}),
    ("versions.2.language", {"versions.2.language"})
}
Extending Select Operator

Selection query extended

$$\sigma(\forall a_j \in \triangle("title") \cdot a_j \neq \text{Null}) \land (\forall a_j \in \triangle("language") \cdot a_j = "English") (C)$$

Rewritten Query

$$\sigma \left( ("title" \neq \text{Null}) \land ("language" = "English" \lor "details.language" = "English" \lor "versions.1.language" = "English" \lor "versions.2.language" = "English") \right) (C)$$
Experiments Experimental Protocol

Synthetic dataset

Figure: Flat Document d1 Describing Movies from IMDB

H. BEN HAMADOU et al. (IRIT) Schema-independent Querying 26-03-2018, DOLAP’18 18 / 28

Synthetic dataset

Figure: Document D1 after structural heterogenetiy injection

H. BEN HAMADOU et al. (IRIT) Schema-independent Querying 26-03-2018, DOLAP’18 18 / 28
Settings of the generated dataset

<table>
<thead>
<tr>
<th>Setting</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td># of schema</td>
<td>10</td>
</tr>
<tr>
<td># of grouping objects per schema</td>
<td>{5,6,1,3,4,2,7,2,1,3}</td>
</tr>
<tr>
<td>Nesting levels per schema</td>
<td>{4,2,6,1,5,7,2,8,3,4}</td>
</tr>
<tr>
<td>Percentage of schema presence</td>
<td>10%</td>
</tr>
<tr>
<td># of attributes per schema</td>
<td>Random</td>
</tr>
<tr>
<td># of attributes per grouping objects</td>
<td>Random</td>
</tr>
<tr>
<td>Collection size</td>
<td>10 GB, 25 GB, 50 GB, 100 GB</td>
</tr>
<tr>
<td>Number of documents per collection</td>
<td>12 M, 30 M, 60 M, 120 M</td>
</tr>
</tbody>
</table>

Table: Settings of the generated dataset

Queries predicates

<table>
<thead>
<tr>
<th>Predicate</th>
<th>Attribute</th>
<th>Type</th>
<th>Operator</th>
<th>Paths</th>
<th>Depths</th>
<th>selectivity</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>DirectorName</td>
<td>String</td>
<td>Regex{^A}</td>
<td>8</td>
<td>{8,2,3,9,6,5,4,7}</td>
<td>0.06 %</td>
</tr>
<tr>
<td>p2</td>
<td>Gross</td>
<td>Int</td>
<td>&gt; 100 k</td>
<td>7</td>
<td>{7,8,2,3,9,6,4}</td>
<td>66 %</td>
</tr>
<tr>
<td>p3</td>
<td>Language</td>
<td>String</td>
<td>= &quot;English&quot;</td>
<td>7</td>
<td>{7,8,3,9,6,5,4}</td>
<td>0.018 %</td>
</tr>
<tr>
<td>p4</td>
<td>Imdb_score</td>
<td>Float</td>
<td>&lt;4,7</td>
<td>8</td>
<td>{8,7,2,3,4,5,6,9}</td>
<td>29 %</td>
</tr>
<tr>
<td>p5</td>
<td>Duration</td>
<td>Int</td>
<td>≤ 200</td>
<td>7</td>
<td>{7,8,2,3,6,5,4}</td>
<td>77 %</td>
</tr>
<tr>
<td>p6</td>
<td>Country</td>
<td>String</td>
<td>≠ Null</td>
<td>6</td>
<td>{7,2,3,9,5,4}</td>
<td>100 %</td>
</tr>
<tr>
<td>p7</td>
<td>year</td>
<td>Int</td>
<td>&lt; 1950</td>
<td>7</td>
<td>{7,8,2,3,6,5,4}</td>
<td>23 %</td>
</tr>
<tr>
<td>p8</td>
<td>FB_likes</td>
<td>Int</td>
<td>≥ 500</td>
<td>7</td>
<td>{6,2,3,8,5,4,3}</td>
<td>83 %</td>
</tr>
</tbody>
</table>

Table: Query predicates
Queries

Q1/Q2
\[ \pi(\cdot)(\sigma(director\_name=\text{"A"} \ (\wedge/\vee)\ gross>100000)(C)) \]

Q3/Q4
\[ \pi(\cdot)(\sigma(director\_name=\text{"A"} \ (\wedge/\vee) gross>100000(\wedge/\vee)\ duration<200(\wedge/\vee) title\_year<1950)(C)) \]

Q5/Q6
\[ \pi(\cdot)(\sigma(director\_name=\text{"A"} \ (\wedge/\vee)\ gross>100000(\wedge/\vee)\ duration<200(\wedge/\vee) title\_year<1950 \ (\wedge/\vee) pays!=\text{Null}(\wedge/\vee) language=\text{English}(\wedge/\vee) imdb\_score<4(\wedge/\vee) \ cast\_total\_facebook\_likes>500(C)) \]
Experimental Results

Average time for conjunctive queries (Q1, Q3, Q5)

Average time for disjunctive queries (Q2, Q4, Q6)

Average time for overall queries
Data diversity effects on query rewriting time and dictionary size

<table>
<thead>
<tr>
<th># of schemas</th>
<th>Query rewriting in (s)</th>
<th>Dictionary size</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.0005</td>
<td>40 KB</td>
</tr>
<tr>
<td>100</td>
<td>0.0025</td>
<td>74 KB</td>
</tr>
<tr>
<td>1 K</td>
<td>0.139</td>
<td>2 MB</td>
</tr>
<tr>
<td>3 K</td>
<td>0.6</td>
<td>7.2 MB</td>
</tr>
<tr>
<td>5 K</td>
<td>1.52</td>
<td>12 MB</td>
</tr>
</tbody>
</table>

Dictionary online construction overhead

<table>
<thead>
<tr>
<th># of schemas</th>
<th>Load (s)</th>
<th>Load and dict. (s)</th>
<th>Overhead</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>201s</td>
<td>269s</td>
<td>33%</td>
</tr>
<tr>
<td>4</td>
<td>205s</td>
<td>277s</td>
<td>35%</td>
</tr>
<tr>
<td>6</td>
<td>207s</td>
<td>285s</td>
<td>37%</td>
</tr>
<tr>
<td>8</td>
<td>208s</td>
<td>300s</td>
<td>44%</td>
</tr>
<tr>
<td>10</td>
<td>210s</td>
<td>309s</td>
<td>47%</td>
</tr>
</tbody>
</table>

Table: Study of the overhead added during load time
Conclusion & perspectives

Plan

1. Introduction
2. Querying Heterogeneous Documents
3. Experiments
4. Conclusion & perspectives

Conclusion

EASYQ Advantages

- Overcoming the problem of querying documents with structural heterogeneity.
- Transparent rewriting mechanisms.
- Ensuring the coverage of latest structural changes. Therefore, the same query is rewritten at each execution

⇒ The heterogeneity is automatically handled.
Perspectives

- Employing real datasets
- Dealing with concurrent access
- Covering more operators