Variety-Aware OLAP of Document-Oriented Databases

Enrico Gallinucci, Matteo Golfarelli, Stefano Rizzi
DISI – University of Bologna, Italy

Introduction

In recent years, schemaless DBMSs have progressively eroded the predominance of RDBMS

Market growth, 2016 (Gartner)
- RDBMSs: +8.6%
- NoSQL: +76.6%
Introduction

Why NoSQL?
- Better scaling
- Low latency: no ACID transactions; absence of a unique schema

Schemaless feature grants flexibility to operational applications...
- ...but more complexity to analytical applications
- ...in terms of querying, interpretation, trust

Our proposal: an original approach to MD querying and OLAP on schemaless sources
- Focus on document-oriented databases

Variety-aware
- Welcome data heterogeneity and schema variety as an inherent source of information wealth

Distinguishing features
- First approach for approximated-OLAP on document-oriented databases
- No cube/DW materialization
- Covering inter-schema and intra-schema variety
  - Missing/additional attributes
  - Different names for an attribute
  - Different structures for instances
Overview

Overview - The BIN framework

We rely on the BIN (Business Intelligence Network) framework [1] to handle schema mappings and query reformulation

- It is an approach to enable OLAP on a P2P data warehousing architecture


Compliance with the BIN data structures guarantees the correctness of the approach
Schema extraction
Goal: introduce a notion of (local) schema for a document

Motivating example: real-world collection of workout sessions

Objects are flattened

Schema of arrays is the union of the fields
Schema extraction
Goal: introduce a notion of (local) schema for a document

Implemented as a customized version of the free tool variety.js

<table>
<thead>
<tr>
<th># records</th>
<th>DB size</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 K</td>
<td>2 MB</td>
<td>4 sec</td>
</tr>
<tr>
<td>50 K</td>
<td>20 MB</td>
<td>33 sec</td>
</tr>
<tr>
<td>500 K</td>
<td>197 MB</td>
<td>6 min</td>
</tr>
<tr>
<td>5 M</td>
<td>1.7 GB</td>
<td>60 min</td>
</tr>
</tbody>
</table>

Consistent with related approaches that perform schema extraction [2]

Schema integration

Goal: integrate the local schemas to obtain a single, comprehensive view

Integration through mappings
- **Primitive mappings**
  - Only exact mappings
  - Transcoding functions required
- **Array mappings**
  - Define the context of primitive mappings

We used a subset of exact mappings, which enables non-approximate query reformulation

Building the global schema requires to adopt:
- An integration methodology (e.g. the *ladder* strategy)
- A technique for finding mappings (plenty of choice)

Our approach: manual (so far)

Future work: **automation**
- Define a new integration methodology
- Basic assumption: \( \text{path}(a) = \text{path}(b) \Rightarrow a = b \)
- Start from the union of all fields
- Collapse fields such that \( p(\text{match}(a,b)) > 1 - \varepsilon, \exists a \Rightarrow \exists b, \exists b \Rightarrow \exists a \)
  - A mapping from \( b \) to \( a \) is materialized
  - Adoption of free tool COMA 3.0 for match determination
FD enrichment

Goal: propose a MD view of the global schema to enable OLAP analyses

Identify hierarchies  
Identify functional dependencies (FDs)

FDs can be identified

- From the schema (intensional)
- From the data \textit{(approximate)}

With FDs we define a \textit{dependency graph}

Intensional FDs, rule #1: every \textit{id} determines the value of its primitives
FD enrichment
Goal: propose a MD view of the global schema to enable OLAP analyses
Intensional FDs, rule #2: every id determines the value of its parent id

Approximate FDs: detected by checking the data
FD enrichment
Goal: propose a MD view of the global schema to enable OLAP analyses

Approximate FD detection follows a smart strategy

Dependency $a \rightarrow b$ is check only if:
- $arr(a) \geq arr(b)$
- $|a| > |b|

Dependency $a \rightarrow b$ is ascertained if:
- $acc(a, b) = \frac{|a|}{|ab|} \geq \varepsilon$, $acc(a, b) \in [0,1]$
- `db.WS.aggregate()`

We rely on the BIN’s reformulation algorithm to propagate queries to the local schemas

Querying
Goal: formulate, execute, evaluate, evolve

OLAP query
- Group-by set (non-empty)
- Selection predicate (optional)
- Measure and aggregation operator

Hard constraints
- Base integrity constraint
- Fact check

Soft constraints
- Summarization integrity constraint
Querying
Goal: formulate, execute, evaluate, evolve

Hard constraint #1: base integrity constraint
- The levels in the group-by set must be functionally independent of each other

Hard constraint #2: fact check
- There must exist a field that determines all others (group-bys, predicates and measure)
- That’s the fact
Querying

Goal: formulate, execute, evaluate, evolve

Soft constraint: summarization integrity constraint

- **Disjointness**
  - The measure instances to be aggregated are partitioned by the group-by instances
  - Fail leads to double counting

- **Completeness**
  - The partitioning of measures by the group-by instances constitutes the entire set
  - Fail requires a balancing strategy to be declared
Querying

Goal: formulate, execute, evaluate, evolve

Execution requires to

- Define the query on the global schema in the DB's language
- Translate it to one query per local schema exploiting mappings
- Execute each query
- Collect and aggregate the results

We rely on the BIN framework for the reformulation algorithm and for the final aggregation of the single results.
We introduce indicators to evaluate the quality of an OLAP query (after it has been executed):

- **Selectivity**
  - Selectivity of the selection predicate
  \[ sel(q) = \frac{\sum_{e \in E} |e|}{|fact(q)|} \]

- **Completeness**
  - Percentage of the queried objects that have not been affected by the balancing strategies
  \[ comp(q) = \frac{\sum_{e \in E, \text{balanced}(e)} |e|}{\sum_{e \in E} |e|} \]

- **Group precision**
  - Percentage of aggregated objects that actually contain a value for the measure
  \[ prec(e) = \frac{|e_m|}{|e|} \]

<table>
<thead>
<tr>
<th>Facility.Chain</th>
<th>Exercises.ExCalories</th>
<th>precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chain1</td>
<td>35</td>
<td>100%</td>
</tr>
<tr>
<td>Chain2</td>
<td>58</td>
<td>95%</td>
</tr>
<tr>
<td>Chain3</td>
<td>44</td>
<td>98%</td>
</tr>
<tr>
<td>Others</td>
<td>21</td>
<td>90%</td>
</tr>
</tbody>
</table>

Selectivity: 100%
Completeness: 33%
Average precision: 92%
Consistently with an OLAP scenario, a query can evolve into another with the application of an OLAP operation. Roll-ups and drill-downs imply navigating of the dependency graph:

- Navigating an AFD with accuracy lower than 1 leads to a violation of the roll-up semantics.
- The results of the second query will not be a correct (de)composition of the results of the first query.

Another indicator evaluates the quality of an OLAP roll-up or drill-down:

- **Accuracy**
  - Quantifies the accuracy of the aggregated results of a query during an OLAP session with respect to the results obtained from the previous query.
  
  $$\text{acc}(q', q) = 1 - \prod_{\gamma \in \Gamma^*} \text{acc}(\gamma)$$

### Exercises

<table>
<thead>
<tr>
<th>Facility.Name</th>
<th>Exercises.ExCalories</th>
<th>precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>Facility1</td>
<td>30</td>
<td>99%</td>
</tr>
<tr>
<td>Facility2</td>
<td>32</td>
<td>100%</td>
</tr>
<tr>
<td>Facility3</td>
<td>48</td>
<td>98%</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

- **Selectivity:** 100%
- **Completeness:** 100%
- **Average precision:** 92%
- **Accuracy:** 98%
Conclusion

We have presented an original approach to approximate OLAP on document-oriented databases

Future work:
- Build a fully-functioning implementation
- Thoroughly evaluate the performance and scalability of the approach
- Switch from a single machine environment to a cluster
- Consider schema profiling techniques to enhance the support given to the user at query time

Thanks