Variety-Aware OLAP of Document-Oriented Databases

Enrico Gallinucci, Matteo Golfarelli, Stefano Rizzi

DISI – University of Bologna, Italy

business intelligence group

26/03/2018

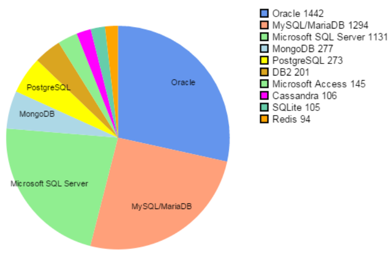
Introduction

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

Market growth, 2016 (Gartner)

- RDBMSs: +8,6%
- NoSQL: +76,6%

DB-Engines Ranking, 2018



26/03/2018

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

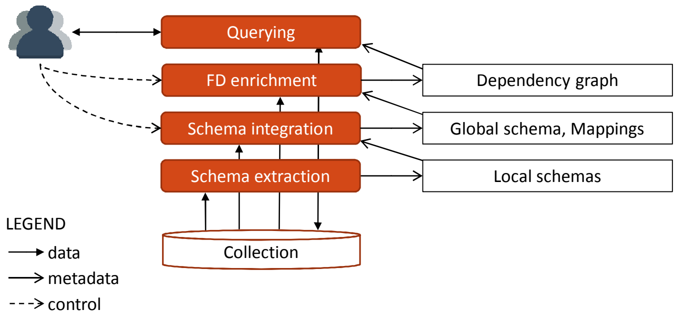
Introduction

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



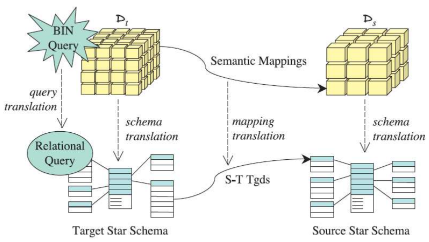
Overview - The BIN framework

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
  - [1] M. Golfarelli, F. Mandreoli, W. Penzo, S. Rizzi, E. Turricchia. ***OLAP query reformulation in peer-to-peer data warehousing***. Inf. Syst. (2012)

Compliance with the BIN data structures guarantees the correctness of the approach

This symbol will be used to highlight where the BIN framework is adopted



Schema extraction

Goal: introduce a notion of (local) schema for a document

Motivating example: real-world collection of workout sessions

```
[  {  "_id" : ObjectId("54a4332f44cfc02424f961d4"),
      "User" :
      {  "FullName" : "John Smith",
         "Age" : 42  },
      "StartedOn" : ISODate("2017-06-15T10:20:44.000Z"),
      "Facility" :
      {  "Name" : "PureGym Piccadilly",
         "Chain" : "PureGym"  },
      "SessionType" : "RunningProgram",
      "DurationMins": 90,
      "Exercises" :
      [  {  "Type" : "Leg press",
            "ExCalories" : 28,
            "Sets" :
            [  {  "Reps" : 14,
                  "Weight" : 60  },
                  . . .
            ] },
         {  "Type" : "Tapis roulant" },
         . . .
      ]
   } ,
   . . .
]
```

WS

_id
User.FullName
User.Age
StartedOn
Facility.Name
Facility.Chain
SessionType
DurationMins

Exercises

Exercises_id
Type
ExCalories

Sets

Sets_id
Reps
Weight

Schema extraction

Goal: introduce a notion of (local) schema for a document

```
[  {  "_id" : ObjectId("54a4332f44cfc02424f961d4"),
      "User" :
      {  "FullName" : "John Smith",
         "Age" : 42  },
      "StartedOn" : ISODate("2017-06-15T10:20:44.000Z"),
      "Facility" :
      {  "Name" : "PureGym Piccadilly",
         "Chain" : "PureGym"  },
      "SessionType" : "RunningProgram",
      "DurationMins": 90,
      "Exercises" :
      [  {  "Type" : "Leg press",
            "ExCalories" : 28,
            "Sets" :
            [  {  "Reps" : 14,
                  "Weight" : 60  },
                  . . .
            ] },
         {  "Type" : "Tapis roulant" },
         . . .
      ]
   } ,
   . . .
]
```

Objects are flattened

Schema of arrays is the union of the fields

WS

_id
User.FullName
User.Age
StartedOn
Facility.Name
Facility.Chain
SessionType
DurationMins

Exercises

Exercises_id
Type
ExCalories

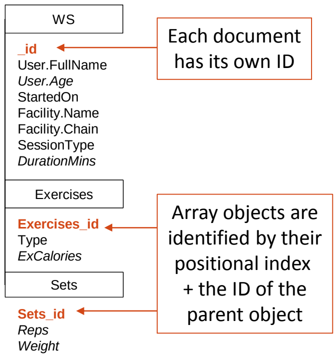
Sets

Sets_id
Reps
Weight

Schema extraction

Goal: introduce a notion of (local) schema for a document


```
[  {  "_id" : ObjectId("54a4332f44cfc02424f961d4"),
      "User" :
      {   "FullName" : "John Smith",
          "Age" : 42  },
      "StartedOn" : ISODate("2017-06-15T10:20:44.000Z"),
      "Facility" :
      {   "Name" : "PureGym Piccadilly",
          "Chain" : "PureGym"  },
      "SessionType" : "RunningProgram",
      "DurationMins": 90,
      "Exercises" :
      [  {   "Type" : "Leg press",
             "ExCalories" : 28,
             "Sets" :
             [  {   "Reps" : 14,
                    "Weight" : 60  },
                . . .
             ] },
         {   "Type" : "Tapis roulant" },
         . . .
      ]
   } ,
   . . .
]
```

Schema extraction

Goal: introduce a notion of (local) schema for a document

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

# records	DB size	Time
5 K	2 MB	4 sec
50 K	20 MB	33 sec
500 K	197 MB	6 min
5 M	1.7 GB	60 min

Consistent with related approaches that perform schema extraction [2]

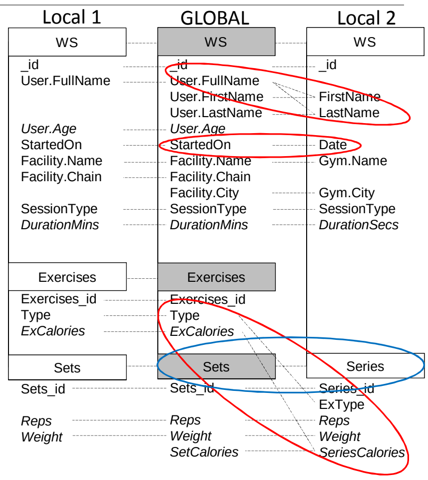
- [2] M. A. Baazizi, H. B. Lahmar, D. Colazzo, G. Ghelli, C. Sartiani. ***Schema Inference for Massive JSON Datasets***. In Proc. EDBT 2017.

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



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

Schema integration

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

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: *path(a)=path(b)* → *a=b*
- Start from the union of all fields
- Collapse fields such that *p(match(a,b))>1-ε* , ∃*a* → ∄*b* , ∃*b* → ∄*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 (*approximate*)

With FDs we define a *dependency graph*

WS
_id
User.FullName
User.FirstName
User.LastName
User.Age
StartedOn
Facility.Name
Facility.Chain
Facility.City
SessionType
DurationMins

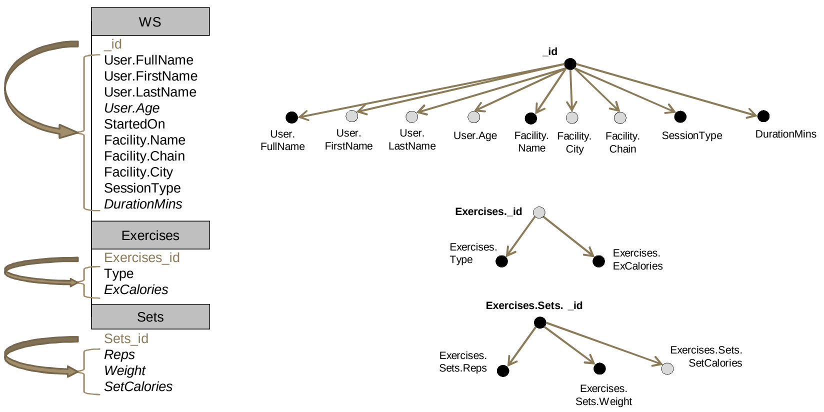
Exercises
Exercises_id
Type
ExCalories

Sets
Sets_id
Reps
Weight
SetCalories

FD enrichment

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

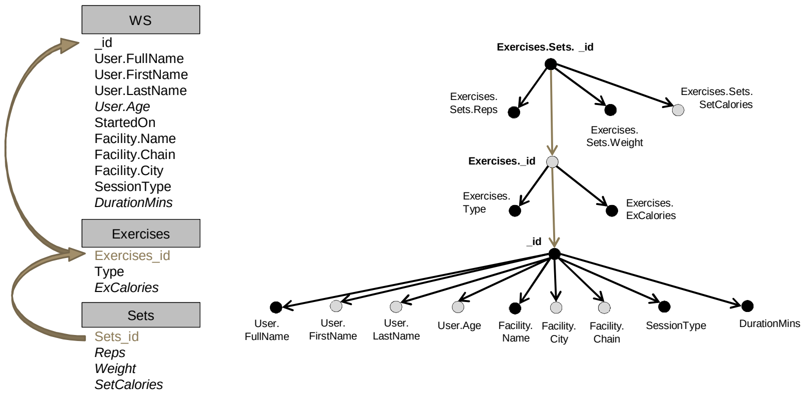
Intensional FDs, rule #1: every *id* determines the value of its primitives



FD enrichment

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

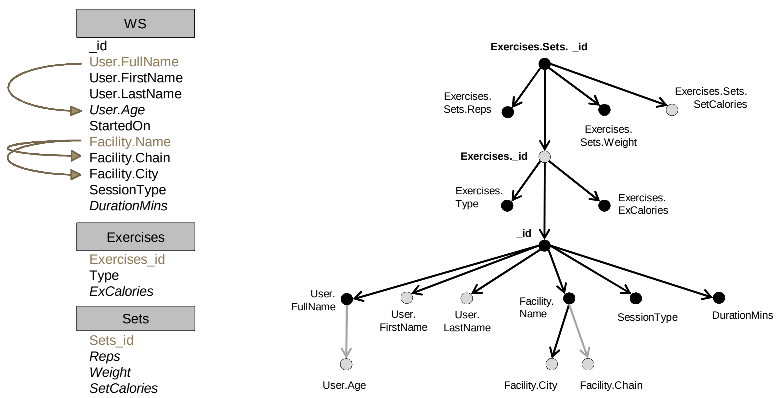
Intensional FDs, rule #2: every *id* determines the value of its *parent id*



FD enrichment

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

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

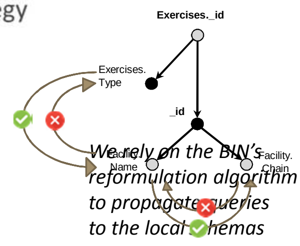
Dependency $a \rightarrow b$ is check only if:

- $\text{arr}(a) \geq \text{arr}(b)$
- $|a| > |b|$

Dependency $a \rightarrow b$ is ascertained if:

- $\text{acc}(a,b) = \frac{|a|}{|ab|} \geq \varepsilon$, $\text{acc}(a,b) \in [0,1]$
-

```
db.WS.aggregate([
  {$group: {
    {"_id": {
      "Facility.Name": "$Facility.Name",
      "Facility.Chain": "$Facility.Chain"
    }
  },
  {$group: {
    {"_id": null, "count": {$sum: 1} }
  }}
]}
```



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

BIN

26/03/2018

Querying

Goal: **formulate**, execute, evaluate, evolve

OLAP query

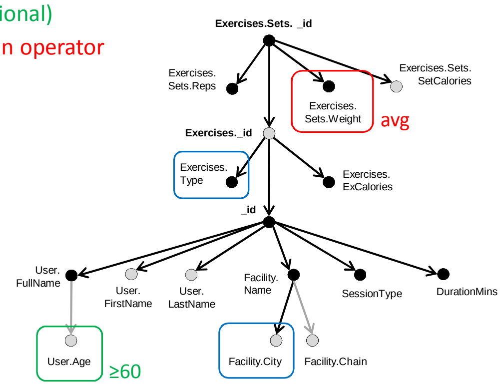
- Group-by set (non-empty)
- Selection predicate (optional)
- Measure and aggregation operator

Hard constraints

- Base integrity constraint
- Fact check

Soft constraints

- Summarization integrity constraint



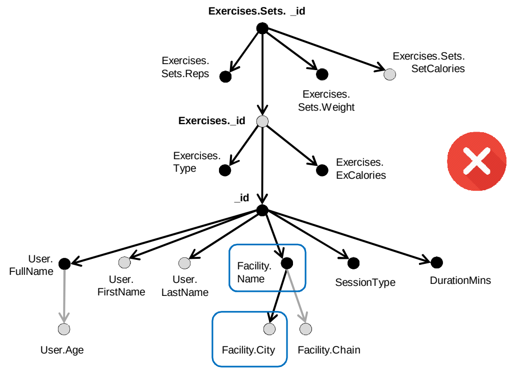
26/03/2018

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

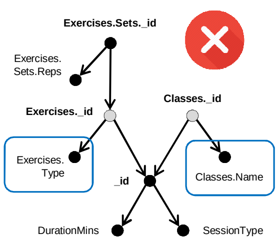


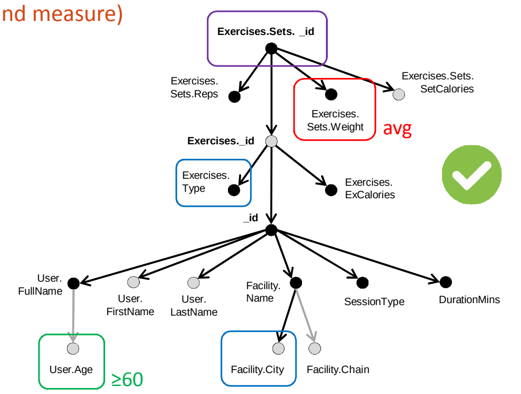
Querying

Goal: **formulate**, execute, evaluate, evolve

Hard constraint #2: fact check

- There must exist a field that determines all others (group-bys, predicates and measure)
- That's the fact



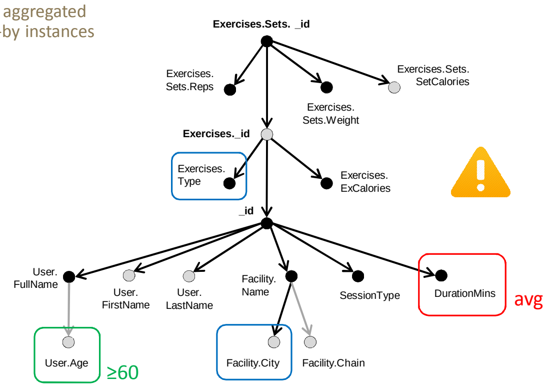


Querying

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



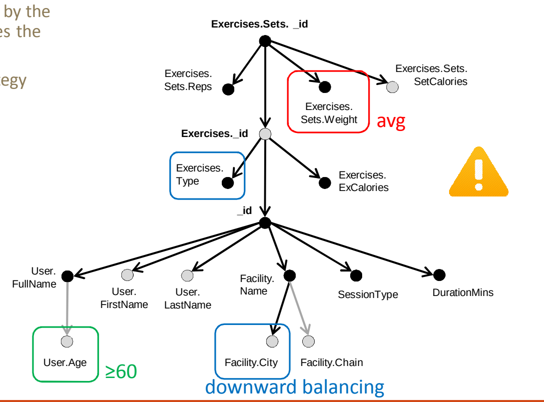
26/03/2018

Querying

Goal: **formulate**, execute, evaluate, evolve

Soft constraint: summarization integrity constraint

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



26/03/2018

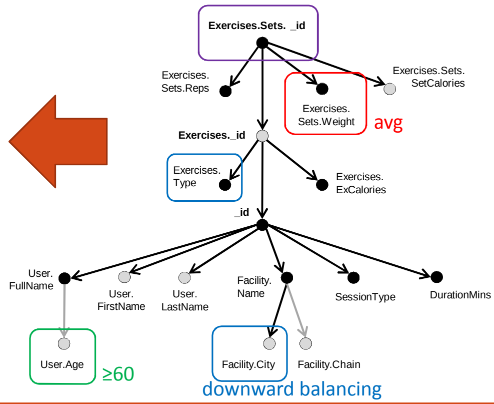
Querying

Goal: formulate, **execute**, evaluate, evolve

Execution requires to

- Define the query on the global schema in the DB's language

```
db.WS.aggregate({
  { $unwind: "$Exercises" },
  { $unwind: "$Exercises.Sets" },
  { $match: { "User.Age": { $gte: 60 } } },
  { $project: {
    "Facility.City": { $ifNull:
       ["$FacilityCity","$FacilityName"] }
    },
    "Exercises.Type": 1,
    "Exercises.Sets.Weight": 1,
    "balanced": {
       $cond: ["$FacilityCity",false,true]
    }
  } },
  { $group: {
    "_id": {
       "FacilityCity","$FacilityCity",
       "ExercisesType","$Exercises.Type",
       "balanced","$balanced"
    },
    "Exercises.Sets.Weight": {
       $avg: "$Exercises.Sets.Weight"
    },
    "count": { $sum: 1 },
    "count-m": { $sum: {
       $cond: ["$Exercises.Sets.Weight",1,0]
    } }
  } }
}
```

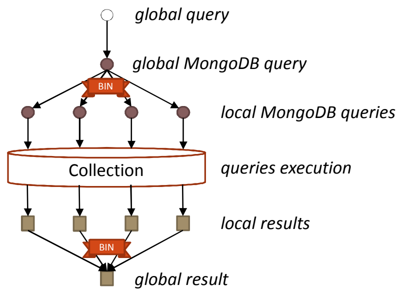


Querying

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



BIN

We rely on the BIN framework for the reformulation algorithm and for the final aggregation of the single results

Querying

Goal: formulate, execute, **evaluate**, evolve

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

$$compl(q) = \frac{\sum_{e \in E, !balanced(e)} |e|}{\sum_{e \in E} |e|}$$

- Group precision
  - Percentage of aggregated objects that actually contain a value for the measure

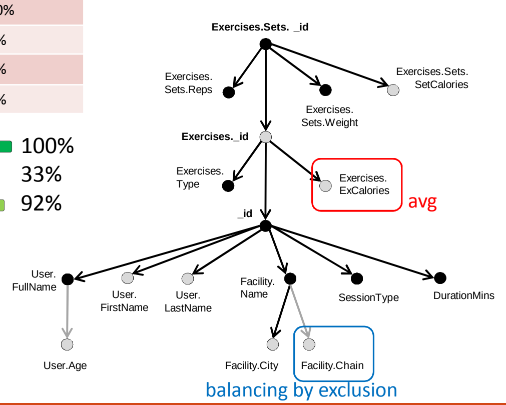
$$prec(e) = \frac{|e|_m}{|e|}$$

Querying

Goal: formulate, execute, **evaluate**, evolve

Facility.Chain	Exercises. ExCalories	precision
Chain1	35	100%
Chain2	58	95%
Chain3	44	98%
Others	21	90%

Selectivity: 100%
Completeness: 33%
Average precision: 92%



Querying

Goal: formulate, execute, evaluate, **evolve**

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

$$acc(q', q) = 1 - \prod_{\gamma \in \Gamma'} acc(\gamma)$$

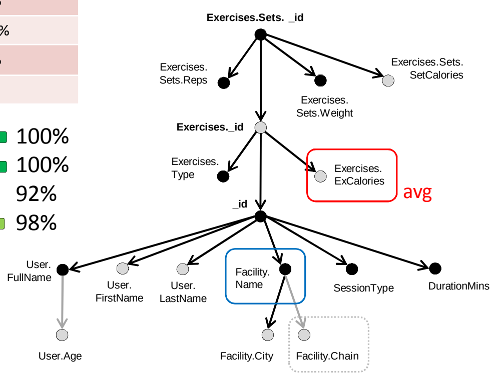
26/03/2018

Querying

Goal: formulate, execute, evaluate, **evolve**

Facility.Name	Exercises. ExCalories	precision
Facility1	30	99%
Facility2	32	100%
Facility3	48	98%
…	…	…

Selectivity: 100%
Completeness: 100%
Average precision: 92%
Accuracy: 98%



26/03/2018

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