QBMetrics: a tool for evaluating and comparing document schemas

Evandro Miguel Kuszera[1,2], Letícia M. Peres[2], and Marcos Didonet Del Fabro[2]

[1] Federal University of Technology - Paraná, Dois Vizinhos, PR, Brazil
evandrokuszera@utfpr.edu.br
[2] C3SL Labs, Federal University of Paraná, Curitiba, PR, Brazil
{lmperes,marcos.ddf}@inf.ufpr.br

Abstract. Document stores are frequently used as representation format in many applications. It is often necessary to transform a set of data stored in a relational database (RDB) into a document store. However, it is difficult to evaluate which target document structure is the most appropriate for each scenario. In this article, we present a tool, called QBMetrics (Query-based Metrics), that assists on an RDB to NoSQL document conversion process, by calculating a set of query-based metrics for evaluating and comparing the created schemas against a set of existing queries. We represent the schemas and the queries as DAGs (Directed Acyclic Graphs), which are used to calculate the metrics. The metrics allow to evaluate if a given target document schema is adequate to answer the queries. We demonstrate the tool in an RDB to NoSQL conversion scenario, involving the creation of the schemas, queries and the metrics calculation.

Keywords: RDBs · Document stores · Metrics · Evaluation · Tool.

1 Introduction

Relational databases (RDB) are widely used to store data of several types of applications. However, they do not meet all requirements imposed by modern applications [8], that handle structured, semi-structured and unstructured data. Furthermore, RDBs are not flexible enough, since they have a predefined schema. NoSQL databases [6] emerged as an option. They differ from RDB in terms of architecture, data model and query language [6]. They are generally classified according to the data model used: document, column family, key-value or graph-based. One of the most used NoSQL format are document stores.

RDB and document stores will be used together, being necessary to investigate strategies to convert and migrate schema and data between them. Different approaches have been presented to convert RDB to NoSQL document stores [7,9,1,4,3]. Some of them consider just the structure of the RDB in the conversion process [7,9]. While others also consider the access pattern of the application [1,4,3]. However, none of the approaches is concerned with the evaluation and the comparison of the output document structure against the existing queries that

need to be adapted and then executed. The work from [2] presents eleven metrics to evaluate the structure of document stores. Such evaluation is important to guide the choice of an adequate document structure. However, the approach has no specific metrics for assessing the queries access pattern against document structure. Despite not having a formal schema, a document has a structure used by the queries to retrieve data. We consider that this document structure can be used as an abstraction to represent a schema.

We present a demo of the **QBMetrics** tool[3], which is used in RDB to NoSQL document conversion processes. It provides a graphical interface to calculate query-based metrics that assist on the choice of a most appropriate target NoSQL schema. The tool uses Direct Acyclic Graphs (DAGs) to represent both the target NoSQL schema and the set of queries, where the vertices are entities and the edges are relationships. More specifically, a DAG represent a collection structure for a schema and an access pattern for a query. DAGs as schema have already been used in a previous approach to convert RDB to NoSQL nested models [5].

This demonstration will show how the tool is used to calculate metrics to evaluate and compare candidate target NoSQL schemas, by executing the following steps:

- creation of the target NoSQL schema and queries;
- calculation of a set of schema scores and query scores;
- analysis of the results.

This paper is organized as follows: section 2 presents the architecture of our tool. In section 3 we present a validation scenario and the demonstration steps. Finally, conclusions are provided in section 4.

2 QBMetrics tool

The QBMetrics tool supports an RDB to NoSQL document conversion process by calculating metrics for evaluating and comparing target NoSQL schema options, prior to the data conversion. The tool has two components, the Converter and Metric components, as shown in Figure 1.

2.1 Converter component

This component defines conversion process, encapsulating the input RDB connection properties, the set of target NoSQL schemas and the set of queries. Both, schemas and queries are represented by DAGs (Directed Acyclic Graphs). A DAG is defined as $G = (V, E)$, where the set of vertices V is related with the tables of the RDB and the set of edges E with the relationships between tables. The direction of the edges defines the transformation flow. Each DAG may be seen as a tree, where the root vertex is the target entity. The path from

[3] The tool is available for download at: `https://github.com/evandrokuszera/nosql-query-based-metrics`

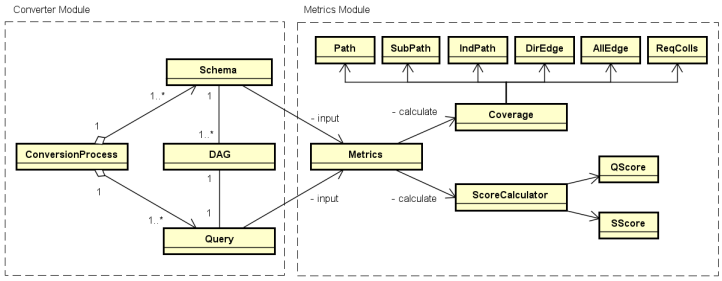

Fig. 1: Tool architecture.

one leaf vertex to the root vertex defines one transformation flow. Each vertex contains the metadata of its respective RDB table, including its name, fields and primary key. The edge between two vertices encapsulates relationship data between two tables, including primary and foreign keys and which entity is on the *one* or *many* side of the relationship. Through the DAG, we specify the de-normalization process from a set of related tables to produce a NoSQL collection. There are works with the same objectives, but with different strategies [3,9].

Considering a DAG as a NoSQL collection, the root vertex is the first level of the collection and the remaining vertices are the nested entities. The direction of the edges defines the direction of nesting between entities and encapsulates nesting type information, including embedded objects or array of embedded objects types. We represent a NoSQL schema through a set of DAGs, where each DAG represents the structure of a collection. We define a NoSQL schema as $S = \{DAG_1, ..., DAG_n | DAG_i \in C\}$, where C is the set of collections of S. The resulting S schema can be used by our Metamoforse framework [5] to migrate data from RDB to NoSQL document

2.2 Metric component

It receives as input a target NoSQL schema and a set of queries, both represented by DAGs. We developed six (6) query metrics and two (2) schema metrics.[4]

Query Metrics The tool calculates three metrics that measure the coverage of the query paths in relation to the schema collection paths, being $Path$, $SubPath$ and $IndPath$. The $Path(query)$ metric measures coverage considering path matching (leaf-to-root vertices), $SubPath(query)$ considering subpath matching and $IndPath(query)$ considering indirect path matching. An indirect path is the one where all its vertices and edges are contained in a regular path, but there are additional intermediate vertices.

[4] A detailed description of the metrics is available in a paper accepted for publication at this same conference ("*Query-based metrics for evaluating and comparing document schemas*").

The tool also calculates two metrics related to edge coverage. $DirEdge(query)$ measures the coverage of the edges of the query against the edges of the collection, considering the direction of edges (e.g. $a \rightarrow b$). On the other hand, $AllEdge(query)$ measures the edge coverage regardless of edge direction (e.g. $a \rightarrow b$ or $a \leftarrow b$). The last metric is called $ReqColls(query)$, which returns the smallest number of collections required to answer a given query.

To measure the coverage of the query paths and query edges relative to all the schema, the maximum value found when applying the metric for each collection is considered. However, for the $ReqColls$ is minimum value found.

Query Score (QScore) represents the query score for a given metric or set of metrics. The $QScore$ for $DirEdge$, $AllEdge$ and $ReqColls$ is the same value returned by the respective metric, for example, $QScore(DirEdge, query) = DirEdge(query)$. However, for the $Path$, $SubPath$ and $IndPath$ metrics, it is calculated as the maximum value between them and is defined as $QScore(Paths) = max(Path, SubPath, IndPath)$.

To calculate the value of $Path$, $SubPath$ and $IndPath$ in $QScore(Paths)$, we use the expression $(xPath(query) * w)/depth(query)$, where $xPath$ can be replaced for one of the three metrics above, w is the weight of each metric and $depth()$ is a function that returns the depth where the $xPath$ match was found in the schema. Different weights can be assigned to each $xPath$, prioritizing schemas with a specific type of structural correspondence. The match depth is used to penalize schemas, with less deep schemas being preferred.

Schema Score (SScore) denotes the sum of the $QScore$ values for all the queries for a given metric (except $ReqColls$), where each query q_i of the set of queries (Q) has a specific weight w_i, and the sum of all w_i is equal to 1. Following the same idea of the $QScore$, $SScore(Paths, Q)$ denotes the schema score for the metrics $Path$, $SubPath$, and $IndPath$. The $SScore$ for $ReqColls$ metric is a ratio between the number of queries and the number of collections required to answer them. A schema that answers each input query through only one collection has $SScore(ReqColls, Q) = 1$. It decreases when the number of collections increases.

To summarize, the $QScores$ shows the coverage provided by the schema for each query, where we can identify which queries require the most attention or are not covered by the schema. The $SScore$ field provides an overview of how well the schema fits the query set. Since the metrics are not independent, we do not define a single expression to calculate the overall score of the schema.

3 QBMetrics demonstration

3.1 Scenario

The tool's demonstration scenario consists of converting an existing RDB to NoSQL document. Figure 2 shows the E-R model of the RDB. Although the

RDB is composed of 7 tables, related to each other, in the demo only the *Customers*, *Orders* and *Orderlines* tables will be used. Generally, the RDB entities are converted to documents and the relationships to references, embedded documents or arrays of embedded documents. The decision on how the documents will be structured is not a trivial task and depends on the various aspects (application access pattern, redundancy, maintainability, etc.).

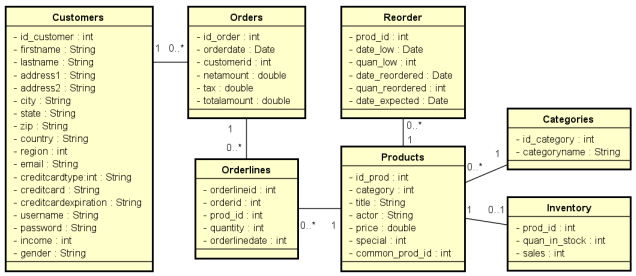

Fig. 2: Input RDB

Figure 3 shows three options for structuring the entities *Customers*, *Orders* and *Orderlines* as documents, named schemas A, B and C. In schema A we have a collection called *Customers*, where *Orders* and *Orderlines* are arrays of embedded documents. In schema B there is a collection called *Orders*, where *Customers* is an embedded document and *Orderlines* is an array of embedded documents. In schema C there are two collections, *Customers* and *Orders*.

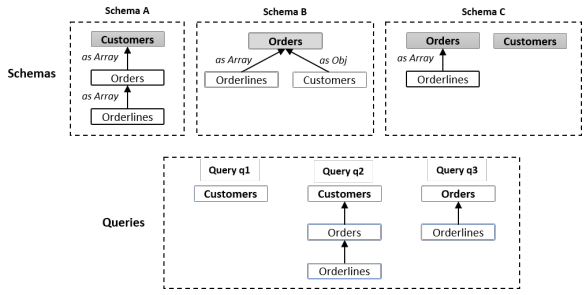

Fig. 3: Input NoSQL document schemas ($A - C$) and queries ($q_1 - q_3$).

In this demo, we consider the application access pattern to evaluate and compare the schemas $A-C$. The access pattern is represented by the queries q_1-q_3 of Figure 3. The goal is to calculate the metrics on the set of schemas and check which one provides greater coverage for the set of queries. The user can configure different weights for queries, prioritizing a certain access pattern. In addition, it

is also possible to assign different weights to the *Path*, *SubPath* and *IndPath* metrics, prioritizing schemas by path type. In the demonstration scenario, all queries will have the same weight (same priority), however, different weights will be assigned for path type, being $path = 1.0$, $subpath = 0.7$ and $indpath = 0.5$. In this way, schemas with greater *Path* coverage will be prioritized, followed by schemas with greater *SubPath* and *IndPath* coverage.

3.2 Demonstration

The tool provides support for defining a conversion process from RDB to NoSQL document, and also for evaluating and comparing possible NoSQL schemas through query-based metrics. Figure 4 shows the graphical interface of the tool. The execution flow is based on four steps: In (A) the input RDB connection parameters are provided. In (B) one or more NoSQL schemas are created from the entities of the input RDB. In (C), queries that represent the application's access pattern are defined. Finally, in (D) the metrics are calculated. Each step will be detailed below.

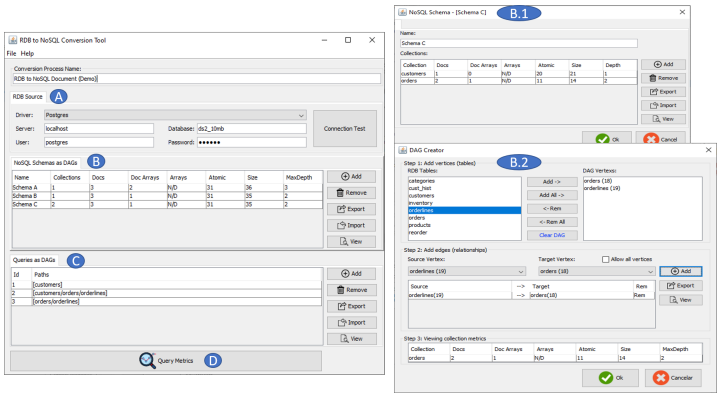

Fig. 4: The tool graphical interface

A: Input RDB The starting point is to define the connection properties of the input RDB (A). The tool accesses the RDB metadata to assist in creating schemas and queries. Currently, Postgres and MySQL databases are supported.

B: NoSQL schemas The user can create one or more NoSQL schemas from the input RDB. Each schema consists of one or more collections of documents, represented as DAGs. In Figure 4 (B), schemas $A - C$ and respective values for structural metrics are shown, such as number of documents, arrays of embedded documents, arrays of primitive types, primitive types and the maximum

collection depth. The user can add new collections or remove existing collections from the schema. In Figure 4 (B.1) schema C is shown. It is composed of the *Customers* and *Orders* collections. In Figure 4 (B.2) the screen for creating a collection (or DAG) is shown. To create a collection three steps are required:

- **Step 1 - Add vertices**: the user selects the vertices (RDB tables) to compose the DAG. The tool automatically loads the list of tables from the input RDB. For instance, in (B.2) the *Orders* and *Orderlines* vertices are selected.
- **Step 2 - Add edges**: the user adds edges between vertices. When selecting the source vertex, the tool automatically searches for possibly target vertex, based on RDB metadata. For example, in (B.2) an edge is added in the direction $Orderlines \rightarrow Orders$. The direction of the edges, together with metadata extracted from the input RDB (e.g. PK and FK) define how the entities will be nested. In this case, *Orderlines* (FK) will be nested as an array of documents embedded in the collection *Orders* (PK).
- **Step 3 - Collection metrics**: show the structural metrics of the collection.

The next step is to create the queries, which will be used to calculate the metrics on the NoSQL schemas.

C: Queries The process of creating queries is similar to the process of creating collections. Both, collections and queries, use DAG-based abstraction. Figure 4 (C) shows the three previously defined queries (from Figure 3). These queries represent the access pattern that will be used to evaluate and compare the schemas $A - C$.

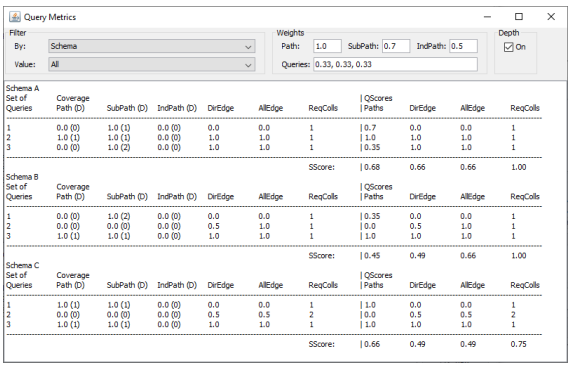

Fig. 5: Query metrics results by schema

D: Calculating Query Metrics After defining the candidate NoSQL schemas and queries, the user can calculate the metrics. Figure 5 shows the results of the

metrics for the schemas $A-C$ and queries q_1-q_3, including query coverage (left side), $QScore$ (right side) and $SScore$ (below each schema). Field (D) represents the depth where the correspondence between schema and query was identified. As a result, among schemas A, B and C, schema A has the highest $SScore$ for the $Paths$ (0.68) and $DirEdge$ (0.66) metrics, being the schema closest to the query access pattern. The schema C is in second place, but has the lowest $SScore$ for the $ReqColls$ (0.75) metric, which means it is necessary to join documents from different collections. Through the metrics, the user can evaluate and compare different NoSQL schema options before migrating the data, where the user can select one or more metrics that best meet the requirements of the application.

4 Conclusion

In this demo paper we presented the **QBMetrics** tool to support the conversion process from RDB to NoSQL document. Based on an input RDB, the user defines a set of candidate NoSQL schemas and a set of queries that represent the application's access pattern, both represented as DAGs. The tool calculates a set of metrics that measures the coverage that a schema provides for the set of queries. This information is used to decide which schema is most appropriate for the required access pattern, before migrating the data. As a future work, we intend to extend the set of metrics to consider aspects related to the implementation effort and execution time of queries in the target database.

References

1. Freitas, M.C.d., Souza, D.Y., Salgado, A.C.: Conceptual mappings to convert relational into nosql databases. In: Proceedings of the 18th ICEIS 2016
2. Gómez, P., Roncancio, C., Casallas, R.: Towards quality analysis for document oriented bases. In: Conceptual Modeling (2018)
3. Jia, T., Zhao, X., Wang, Z., Gong, D., Ding, G.: Model transformation and data migration from relational database to MongoDB. In: IEEE BigData. pp. 60–67 (2016)
4. Karnitis, G., Arnicans, G.: Migration of relational database to document-oriented database: Structure denormalization and data transformation. In: 2015 7th ICCI-CSN. pp. 113–118 (2015)
5. Kuszera, E.M., Peres, L.M., Fabro, M.D.D.: Toward RDB to NoSQL: Transforming data with metamorfose framework. In: Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing. pp. 456–463. SAC '19 (2019)
6. Sadalage, P.J., Fowler, M.: NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence. Addison-Wesley Professional, 1st edn. (2012)
7. Stanescu, L., Brezovan, M., Burdescu, D.D.: Automatic mapping of MySQL databases to NoSQL MongoDB. In: 2016 FedCSIS. pp. 837–840 (Sep 2016)
8. Stonebraker, M., Madden, S., Abadi, D.J., Harizopoulos, S., Hachem, N., Helland, P.: The end of an architectural era (it's time for a complete rewrite). In: Proc. of 33rd VLDB, University of Vienna, Austria, Sept 23-27, 2007. pp. 1150–1160 (2007)
9. Zhao, G., Lin, Q., Li, L., Li, Z.: Schema conversion model of sql database to nosql. In: 2014 Ninth 3PGCIC. pp. 355–362 (2014)