9 Evaluation of the Semantic Data Quality Management Framework (SDQM)

In this chapter, we evaluate the proposed SDQM approach. The evaluation methodology of SDQM is separated into three parts. The first part is concerned with the evaluation of precision and recall of SDQM's data quality monitoring and assessment algorithms. The second part evaluates the practical applicability of SDQM by applying the framework to three different use cases, namely one business use case on material master data of a large organization, one Semantic Web use case with data from DBpedia[68], and one use case that examines the capability of SDQM to automatically identify inconsistent data requirements. In the third part of the evaluation, SDQM is compared to a conventional data quality tool.

9.1 Evaluation of Algorithms

9.1.1 Algorithm Evaluation Methodology

In this section, we will apply the notions of recall and precision from the field of Information Retrieval to data quality management and use them as indicators for the performance of our approach (cf. Batini & Scannapieco, 2006, pp. 125-127; Buckland & Gey, 1994; Raghavan et al., 1989). This is based on the idea that essentially our algorithms attempt to retrieve all requirement violations. Precision can be defined as the degree to which an information retrieval result contains relevant information (cf. Buckland & Gey, 1994, p. 12f.). It is measured via the ratio between true positives (TP) and the sum of true positives (TP) and false positives (FP) (Batini & Scannapieco, 2006, p. 126). True positives are thereby instances that are correctly identified to be relevant (Batini & Scannapieco, 2006, p. 125f.). False positives are relevant instances that were incorrectly identified to be relevant (Batini & Scannapieco, 2006, p. 125f.). In our case, true positives are correctly identified data requirement violations and false positives are requirement violations that have not been identified.

[68] http://dbpedia.org

C. Fürber, *Data Quality Management with Semantic Technologies,*
DOI 10.1007/978-3-658-12225-6_9, © Springer Fachmedien Wiesbaden 2016

Hence, precision in our case measures how many of the identified data requirement violations have been identified correctly, i.e. really violate a data requirement (Batini & Scannapieco, 2006, p. 126).

$$Precision = \frac{\mathrm{TP}}{TP + FP}$$

Recall is a measure that represents the ratio between the retrieved relevant instances and all relevant instances (cf. Buckland & Gey, 1994, p. 12). In our case, the equivalent is the number of correctly identified requirement violations (TP) and all requirement violations including false negatives (FN), i.e. requirement violations that have not been identified by the algorithms. Recall, therefore, measures how many data requirement violations have been identified by the algorithm compared to the whole population of data requirements violations (cf. Batini & Scannapieco, 2006, pp. 125-127).

$$Recall = \frac{\mathrm{TP}}{TP + FN}$$

Since our algorithms attempt to identify all data quality problems related to a certain data requirement, the scores for precision and recall should be equal to one in the ideal case.

9.1.2 Application Procedure

In order to identify the required variables correctly, we created a small test data set with product and location data that contains all instance-related single-source data quality problem types as listed in table 5 of section 3.6.1. Additionally, we created 49 self-defined data requirements for the data, such as “Every instance of class `Location` must have a ZIP code.” The full set of rules that were used to evaluate SDQM’s algorithms can be found in appendix B. The full test data set including the reference data that was used in the evaluation can be found in appendix C. All requirement violations in the test data set were known, so that we were able to exactly identify all false positives and false negatives. In sum, we tested all 64 algorithms of SDQM for data quality monitoring and data quality assessment.

9.1.3 Results

As expected all tested algorithms returned perfect results for precision and recall. These perfect results are necessary before we apply the algorithms to real data, in order to make sure that they are able to identify all types of data quality problems. It must be stressed that the queries related to “Functional Dependency Reference Rules” return instances that miss one or more dependent values or properties as requirement violations, i.e. true positives, although the correct value may be located in a different attribute. E.g. the record with LOCID equal to 3 with city value “Nantes” and state value “France” returned as true positive since the correct dependent value “France” was not located in the property country, but located in the wrong property state. A full list of the algorithm evaluation results of SDQM can be found in appendices D and E. In summary, the evaluation results show that SDQM’s algorithms are able to identify data requirement violations and assess the state of data quality correctly.

9.2 Use Case 1: Evaluation of Material Master Data

The first use case deals with a real business scenario that is concerned with the quality of master data of an information system. According to ISO 8000-102:2009 master data is defined as “data held by an organization that describes the entities that are both independent and fundamental for that organization and that it needs to reference in order to perform its transactions” (ISO, 2009). Hence, correct and complete master data is essential for the execution of business processes and, therefore, the organizational success. This first use case shall illustrate how the SDQM framework can be applied for master data quality management in real-world settings. We thereby evaluate SDQM especially regarding the following criteria:

- Ability of SDQM to represent the organization’s data requirements
- ability to process the organization’s data requirements to create data quality reports, and
- performance of SDQM’s data quality reports

9.2.1 Scenario

A large public organization uses an ERP system to support its logistic processes. The system contains material master data as a source for process-relevant information that is used for process execution. The system uses the material master data to automate tasks such as the placement of purchase orders, storage management, or to inform people, e.g. about appropriate handling of materials. In order to avoid process failures, it is necessary to assure that the master data provided by the ERP system is of sufficient quality. Therefore, the organization seeks for a system that identifies data quality problems, i.e. instances with data that violate the organization's requirements, and that allows the quick evaluation of the overall quality state of data items.

9.2.2 Setup and Application Procedure of SDQM

The SDQM framework is used in the context of the above scenario to (1) represent data requirements, (2) identify requirement violations, and (3) evaluate the quality state of data items of the data source. Therefore, SDQM was set up with the data of the organization on a local server as explained in section 8.1. The server used is an AMD Athlon II X4 630 Processor 2.80 GHz with 8 GB RAM on a Windows 7 64bit operating system. The Fuseki server thereby received 4,600 megabyte of the RAM and the SDQMgr 1,536 megabyte of RAM. The capturing of data requirements and the execution of data quality measurement reports was performed as described in section 8.2. The organization provided 19 data requirements for their general material master data. The source data was stored in single table of a relational database. We converted the data into an N-Triples file via D2RQ[69] and imported the N-Triples file into the triplestore via the user interface of the Fuseki server[70]. In the relational database, the source table had 3.3 million records. Together with the data requirements the triplestore contained 53,077,730 triples. Before executing SDQM's reports, the hardware setup was optimized by comparing the execution time of a simple SPARQL query that counts all triples of the Jena TDB published by the Fuseki server. In the mentioned configuration, the COUNT query performed best and executed within 41,713 milliseconds. Table 20 shows the rules that have been collected from experts

[69] http://d2rq.org/ (Last accessed on 30.08.2014)
[70] http://jena.apache.org/documentation/serving_data/ (Last accessed on 30.08.2014)

of the organization and were applied on their material data to identify data quality problems.

Table 20: Data requirements that were collected and applied for use case 1[71]

Report	Rule
Missing values and properties (5 property requirements)	The following fields must have a value for all materials: - Lab/Office - Material group - Base unit of measure - Manufacturer part number - Material type
Conditional missing values and properties (1 requirement)	If the material type is set for non-valuated materials, then the field “Installation type” must always have a value.
Syntax violations (1 property requirement)	The field “Internal material number” must always have 9 digits.
Illegal values (Legal value rules) (6 property requirements)	The following fields can only obtain specific values: - Installation type - External material group - Material condition management - Serial number profile - Lab/Office - Material type
Out of range values (1 property requirement)	The field “Standard price” must not be lower than 0.02 € and not higher than 999,999,999.00 €.
Duplicate instances (3 equal values) (1 duplicate instance requirement)	If the material text, the manufacturer part number and the standard price have the same value for two or more instances, then the instances are potential duplicates.
Functional dependent value rule (4 requirements)	Furniture materials must have a specific installation type value. Certain material types are always in ownership of a specific office. Materials with a specific external material group are always in ownership of a specific office. Materials with a certain installation type must always have a price greater than 4,999.00 €.

[71] The rules are described on an abstract level in order to assure the anonymity of the organization.

9.2.3 Results and Findings

As shown in table 20, the data requirements delivered by the organization covered syntax rules, legal value rules, duplicate instance rules, property completeness rules, legal value range rules, and functional dependency rules. The standard forms of SDQM's data requirements wiki were expressive enough to cover all of the organization's data requirements. All data requirements were represented in the data requirements wiki and could be processed by the SDQMgr to generate reports about requirements violations and reports that reflect the overall quality state of the organization's data items. Figure 47 shows the data quality monitoring report with instances that violate a legal value range requirement of a certain property.

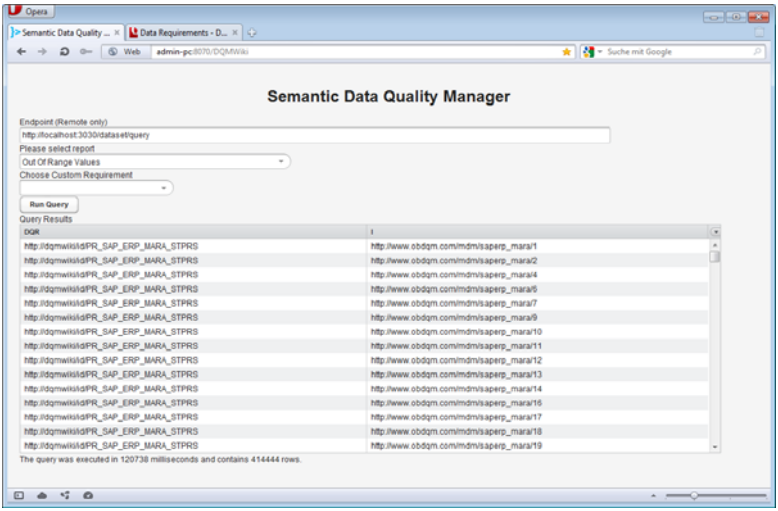

Figure 47: Report with legal value range violations

Figure 48 shows the accordant data quality assessment report which contains a score about the overall semantic accuracy of the property. The score has been computed based on the legal value range requirement which contains an upper and lower legal value for the property.

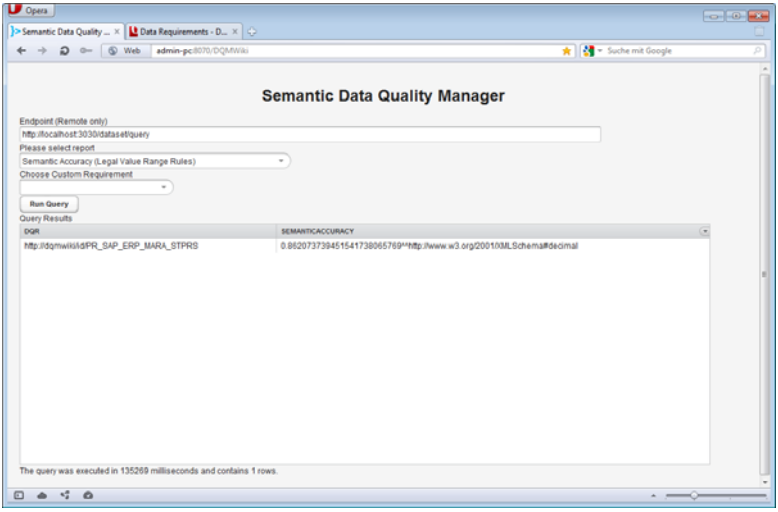

Figure 48: Report with semantic accuracy score based on value range requirement

The overall performance of the reports that were executed with the SDQMgr showed mostly sufficient results as shown in table 21. One exception was discovered during the execution of the report that indicates duplicate instances. The accordant query of SDQMgr was designed to compare certain property values of each instance with each other. In our use case, duplicate instances should be identified in a class with roughly 3,000,000 instances. This resulted in $(3{,}000{,}000 - 1)^2 / 2$ comparisons which was not processable in a sufficient time with the current setup. However, the data quality assessment reports showed also sufficient results regarding their performance as illustrated in table 22.

Table 21: Evaluation results of SDQMgr's data quality monitoring reports (use case 1)

Report	**Result**	**Execution Time (in min:sec.ms)**
Missing Values and Properties (5 requirements)	311,821 rows	10:02.901
Conditional Missing Values and Properties (1 requirement)	56 rows	01:43.038
Syntax violations (1 requirement)	7 rows	03:54.431
Illegal Values (Legal Value Rules) (6 requirements)	23,724 rows	18:35.353
Out of Range Values (1 requirement)	414,444 rows	02:00.738
Duplicate Instances (3 Equal Values) (1 duplicate instance requirement)	Did not finish	Did not finish
Functional Dependent Value Rule (4 requirements)	71 rows	02:02.784

Table 22: Evaluation results of SDQMgr's data quality assessment reports (use case 1)

Report	Result	Execution Time (in min:sec.ms)
Completeness (5 requirements)	Property 1: 100 % Property 2: 99,05 % Property 3: 93,05 % Property 4: 97,53 % Property 5: 100 %	15:59.841
Conditional Completeness (1 requirement)	Property 6: 99,93 %	01:50.137
Syntactic Accuracy (Syntax Rules) (1 requirement)	Property 7: 99,99 %	02:08.727
Syntactic Accuracy (Legal Value Rules) (6 requirements)	Property 8: 99,95 % Property 9: 100 % Property 6: 99,99 % Property 4: 99,97 % Property 10: 99,28 % Property 5: 100 %	27:18.928
Semantic Accuracy (Legal Value Range Rules) (1 requirement)	Property 11: 86,20 %	03:04.716
Semantic Accuracy FDV (1 Condition) (4 requirements)	FDV 1: 100 % FDV 2: 100 % FDV 3: 99,96 % FDV 4: 99,77 %	02:54.406

In summary, the evaluation results show that SDQM is basically capable to be used for quality management of master data in real-world business settings. However, there is room for improvement in several areas. In particular, future work on SDQM should regard the following options to increase performance:

- Jena's in-memory technology could be used to load the whole Jena TDB of SDQM into the computer's main memory before execution of SDQMgr's reports.
- The execution of queries and generation of data quality reports could be decoupled from each other. E.g. the queries could be executed at night and the reports would only access a cached query result.
- The CPU and main memory capacity could be extended to provide more resources for SDQM's applications.
- An authorization system could be added that requires user's login before the execution of data quality reports to avoid inappropriate use.

Moreover, SDQM's mechanisms for representing and processing duplicate instance requirements should be optimized to be applicable to larger data sets, e.g. by adapting duplicate detection algorithms as proposed in (Monge & Elkan, 1997) or (Herschel et al., 2011). For example the performance of SDQM's duplicate checking algorithm can be improved by adjusting the algorithm to search for duplicates only in a sorted neighborhood (Bitton & DeWitt, 1983) or by building clusters based on the transitivity of the "isDuplicateOf" relationship and thereby avoiding unnecessary comparisons (Monge & Elkan, 1997).

Despite the successful application of SDQM in this use case, it must be stressed that this is only a first step to prove SDQM's practical applicability. A longer practical application of SDQM in a realistic business setting would be needed to evaluate the strengths and weaknesses of SDQM with higher precision. For example the amount of data requirements will most likely increase over time and easily exceed the number of data requirements as applied in this use case. Furthermore, more complex functional dependencies may exist that may not be represented with the standard forms of SDQM.

9.3 Use Case 2: Evaluation of Data from DBpedia

The second use case attempts to investigate the applicability of SDQM for tasks related to data quality in Semantic Web scenarios. As for the evaluation, we chose DBpedia (Bizer, Lehmann, et al., 2009), a publicly available Semantic Web data source that contains structured information from Wikipedia. As DBpedia data stems from the open environment of Wikipedia where anyone can edit content, it raises new challenges for a data quality management tool especially regarding the heterogeneity of data and data requirements.

9.3.1 Scenario

Wikipedia is a public encyclopedia that can be edited by anyone who has access to the internet (cf. Voss, 2005, p. 1). As of June 2012 the English Wikipedia contains over 3.9 million articles about persons, locations, movies, species, and many other things[72]. The DBpedia project extracts the structured part of Wikipedia's articles regularly and publishes the data in the Semantic Web (cf. Kobilarov, Bizer, et al., 2009, p. 35f.) where it can be used by anyone for multiple different purposes. Due to the amount of data, it is not feasible to identify data quality problems manually. Thus, means are required to efficiently identify data quality problems and to evaluate the quality state of DBpedia's data items for the following purposes:

- Administrators of DBpedia and Wikipedia may want to efficiently identify vandalism caused by the openness of Wikipedia.
- Data consumers may want to evaluate the quality state of certain parts of DBpedia before relying on it.

In the following, we evaluate whether SDQM may help in these tasks.

[72] http://en.wikipedia.org/wiki/Main_Page (Last accessed on June 10th 2012)

9.3.2 Specialties of Semantic Web Scenarios

Data quality tasks in open environments such as the Semantic Web underlie different conditions than data quality management tasks of information systems in closed settings. Since data can be edited and used by anyone, the degree of heterogeneity is much larger in open settings than in closed settings (cf. Batini & Scannapieco, 2006, p. 15; Bizer, 2007, p. 44). Heterogeneity thereby does not only reflect on data, but also on data requirements due to different subjective preferences and different use cases, in which the data is used (Bizer & Cyganiak, 2009, p. 2). Hence, the definition of the characteristics of high quality data may be much more contrary in open settings, since it is more difficult to achieve agreement in a large and diverse environment such as the Web. In consequence, the goal of data quality management tasks is usually not primarily the correction of data according to specific requirements of single users. A consensual agreement would have to be first established about a data requirement before requirement violations can be corrected in the data source. Due to heterogeneous world views and ways of expression, it is not realistic to satisfy everyone's requirements.

9.3.3 Setup and Application Procedure

First of all, we downloaded the DBpedia ontology, the ontology infobox types, the property data including the specific properties, and the titles data which are all available at http://dbpedia.org/Downloads37. The downloaded data sets were extracted from the English Wikipedia in July 22nd 2011 and contain 35,823,373 million triples in summary. The data was loaded into SDQM's triplestore. We thereby used the same hardware configuration as in use case one. We also again used the application procedure as describe in figure 46 to create the requirement metadata for the data quality management tasks. Since (to the best of our knowledge) there is currently no community that establishes agreement among data requirements in Web environments such as DBpedia, we created our own subjective data requirements. It must be stressed that, therefore, the ability of SDQM to represent data requirements cannot be fully evaluated. However, this second use case rather focuses on collecting first evidence for the applicability of SDQM in Semantic Web environments. Table 23 lists the assumed data requirements for this use case.

Table 23: Assumed data requirements of use case 2

No.	Requirement Description
1	The property http://dbpedia.org/ontology/gender can only obtain the values http://dbpedia.org/resource/Female and http://dbpedia.org/resource/Male.
2	The property http://dbpedia.org/ontology/populationTotal can only obtain values between 0 and 7,000,000,000.
3	The property http://dbpedia.org/ontology/populationTotal can only obtain numeric values.
4	The property http://dbpedia.org/ontology/populationTotal should exist in all instances of the class http://dbpedia.org/ontology/PopulatedPlace.
5	The property http://www.w3.org/2003/01/geo/wgs84_pos#long must exist in all instances of class http://dbpedia.org/ontology/Place.
6	The property http://www.w3.org/2003/01/geo/wgs84_pos#long must have a specific syntax (Regular expression: "^(\-?\d+(\.\d+)?)").
7	The property http://www.w3.org/2003/01/geo/wgs84_pos#lat must exist in all instances of class http://dbpedia.org/ontology/Place.
8	The property http://www.w3.org/2003/01/geo/wgs84_pos#lat must have a specific syntax (Regular expression: "^(\-?\d+(\.\d+)?)").
9	Country – Capital combinations in DBpedia must match the country capital combinations of Geonames.

We focused on data requirements relevant for data usage of data from the DBpedia classes `dbo:Place`[73], `dbo:PopulatedPlace`[74], `dbo:Country`[75], and `dbo:Person`[76]. It must be stressed that the data requirements as listed above are the

[73] http://dbpedia.org/ontology/Place
[74] http://dbpedia.org/ontology/PopulatedPlace
[75] http://dbpedia.org/ontology/Country
[76] http://dbpedia.org/ontology/Person

subjective requirements of the author and do not necessarily represent a commonly accepted understanding of high-quality data in DBpedia.

9.3.4 Results and Findings

Our analyses identified several requirement violations. E.g. requirement no. 1 revealed that there are eight other values for the property http://dbpedia.org/ontology/gender in instances of the class http://dbpedia.org/ontology/Person besides “Male” and “Female” in the English Wikipedia as of July 2011, namely “Man”, “Nerd”, “Cylon (Battlestar Galactica)”, “Elves (Shannara)”, “Puppet”, “Sex”, and “Pantomime horse”. Figure 49 shows the results as identified by the SDQMgr.

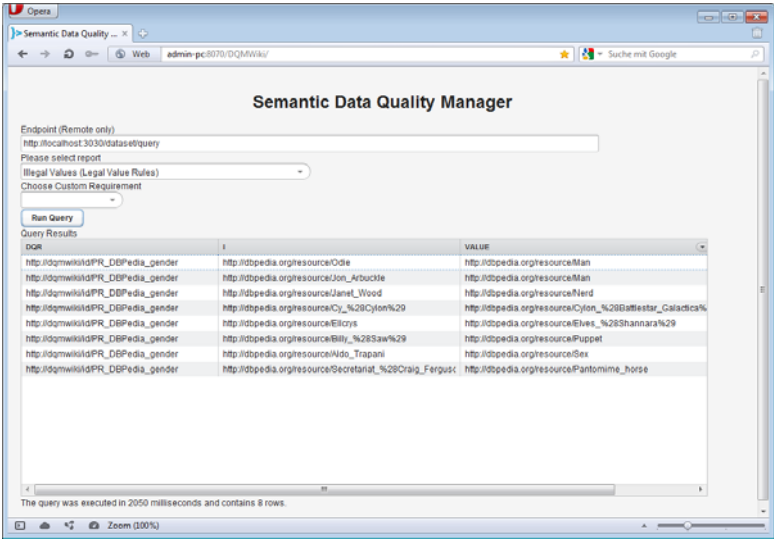

Figure 49: Result of legal value requirement analysis in DBpedia

An additional random check confirmed the usage of these values in the English version of Wikipedia. Figure 50 reveals that the Wikipedia page of the television character “Janet Wood” has been subject to assignment of the value “Nerd” as gender. In the meanwhile the value for gender has been changed by the Wikipedia community to “Female”. This reflects agreement to the author’s understanding of legal values for the properties representing the gender of a person.

```
{{Infobox character
| name       = Janet Wood
|image       = [[Image:Janet Wood 1982.png|220px]]
| caption    = Joyce DeWitt as Janet Wood
| first      = "A Man About the House"
| last       = "Friends and Lovers"
| nickname   =
| alias      =
| species    =
| gender     = [[Nerd|Female]]
| occupation = Florist, Aerobics instructor
| title      =
| family     = Roland Wood (father)<br>Ruth Wood (mother)<br>Jenny Wood (sister)<br>unnamed brother
| children   =
| relatives  =
| portrayer  = [[Joyce DeWitt]]
| creator    =
}}
```

Figure 50: Infobox source code of Wikipedia page “Janet Wood” as of June 27, 2011

However, the analysis results contain other requirement violations that point to less agreement about the correct gender value. Figure 51 shows a page about the robot “Cy” from the television series “Galactica 1980” which indicates the Gender “Cylon” for “Cy” until today[77].

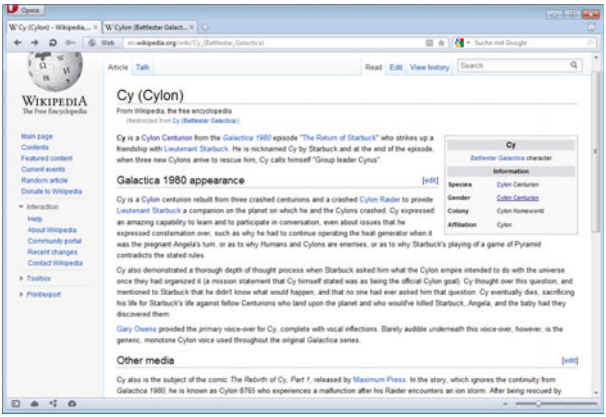

Figure 51: Wikipedia page "Cy (Cyclon)” as of June 10, 2012

To the best of our knowledge, there is no commonly accepted truth about the real gender of Cy. Therefore, the gender “Cylon” may be seen as valid. However, from our subjective perspective it is not harmful to regard “Cylon” as invalid value for representation of a gender. But most likely we are not able to change the value for “Cy” permanently to “Male” in Wikipedia without convincing the community. This example

[77] Today in this context equals June 10th 2012.

emphasizes the special problems related to data quality management in open environments such as the Web.

Moreover, we were able to detect obviously incorrect values for the property http://dbpedia.org/ontology/populationTotal. We found 47 instances of the class http://dbpedia.org/ontology/Place which contain a population value greater than 7,000,000,000. Figure 52 shows SDQMgr's report on out of range violations according to our data requirement No.2 of table 23.

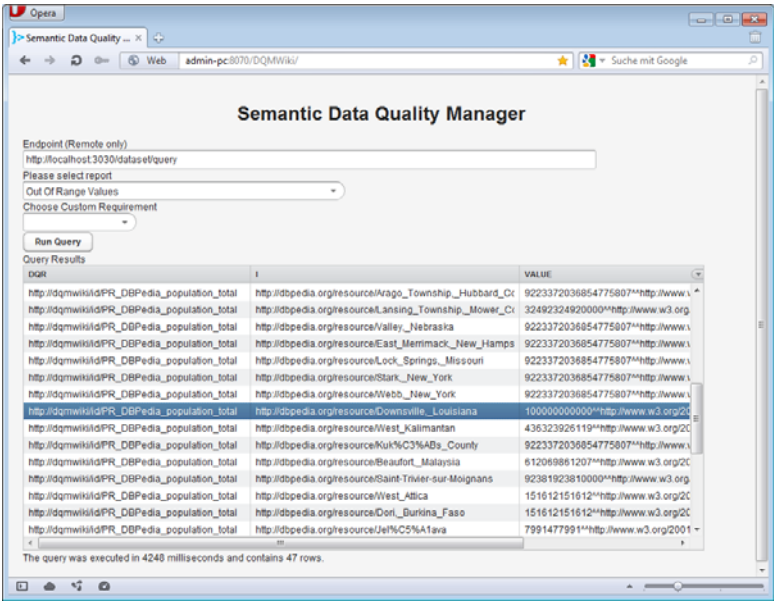

Figure 52: Out of range values for property "population" in DBpedia

The highlighted row in the result table shows that "Downsville Louisiana" has a population value of "100,000,000,000". The accordant Wikipedia page from June 19th 2011 confirms this result as illustrated in figure 53.

Figure 53: Wikipedia page "Downsville, Louisiana" as of June 19th 2011

In the meanwhile, the population value for Downsville (Louisiana) has been corrected to 141 inhabitants[78]. The syntactic requirements for the property http://www.w3.org/2003/01/geo/wgs84_pos#long and the property http://www.w3.org/2003/01/geo/wgs84_pos#lat did not return any violations in the SDQMgr.

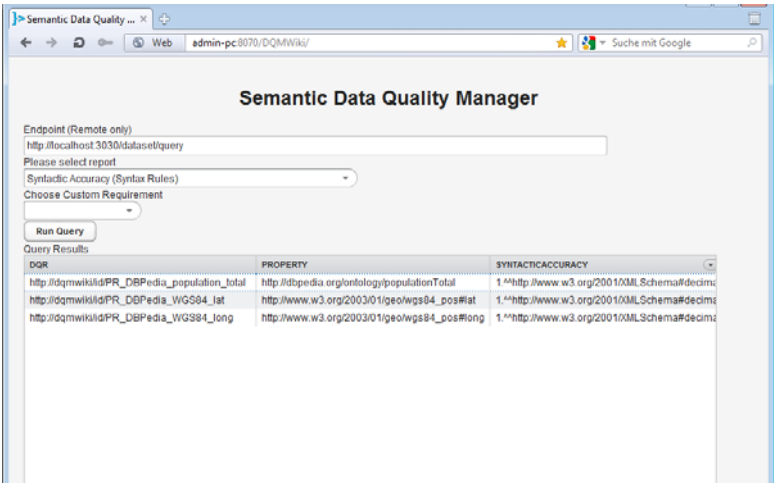

Figure 54: Data quality assessment report displaying syntactic accuracy results

[78] http://en.wikipedia.org/wiki/Downsville,_Louisiana (Last accessed on June 10th 2011)

Moreover, we generated data quality assessment reports to each of the requirements which are shown in table 24.

Table 24: SDQMgr's data quality assessment results on DBpedia

Report	**Result**	**Execution Time (min:sec.ms)**
Completeness (Requirement no. 4, 5, 7)	Population total: 61,21 % Latitude: 65,79 % Longitude: 65,79 %	01:27.221
Syntactic Accuracy (syntax rules) (Requirement no. 3, 6, 8)	Population total: 100 % Latitude: 100 % Longitude: 100 %	01:02.057
Syntactic Accuracy (legal value rules) (Requirement no. 1)	Gender: 99,99 %	00:47.565
Semantic Accuracy (out of range rules) (Requirement no. 2)	Population: 99,98 %	00:14.773
Semantic Accuracy (functional dependency reference rule) (Requirement no. 9)	Country Capital Combinations (Variant 1: Class Country): 0,07 % Country Capital Combinations (Variant 2: Class CurrentCountry): 46,22 %	00:06.100 00:01.701

It must be stressed that the interpretation of the above results must be performed very carefully. For example the analysis results show that DBpedia and, therefore, most likely also Wikipedia provides data on population, latitude, and longitude for almost two thirds of the documented places or populated places respectively. This does not mean

that it makes sense to provide such data for all of Wikipedia's places and populated places, since these high level classes may combine different concepts. For example, the data quality monitoring report with missing latitude and longitude values contains a lot of rivers which do not have specific latitude and longitude values. Moreover, we identified almost perfect results regarding our syntactic requirements except for the gender values that were mentioned earlier. The semantic accuracy of the population values that were tested with help of a legal value range (requirement no. 2) is also on a very high level. The 0.02 % requirement violations are all caused by population values beyond 7,000,000,000 which have partly already been removed in Wikipedia as shown earlier. Finally, we tested country related data of DBpedia against Geonames[79], a publicly available data source for geographic data. We thereby downloaded the country info data of Geonames[80] as of June10th 2012 which contains information about 252 countries, such as population, capital, currency, format of postal codes, etc. The Geonames data was converted to be matched against data from DBpedia's `dbo:Country` class as trusted reference to check valid combinations of country names and its capital cities. The first run showed insufficient results as only 0.07 % of DBpedia's country data matched with the data in Geonames. One of the major reasons for this poor result was the fact that DBpedia represents current and historic countries while Geonames only represents current countries. Thus, we adjusted our data requirement by creating a new class `CurrentCountries` that contains all instances of DBpedia without a property value for `dbpedia-owl:dissolutionDate` or `dbpedia-owl:dissolutionYear`. In consequence, the semantic accuracy score raised up to 46.22 %. The remaining requirement violations are in majority caused by different naming, e.g. "Bogota" versus "Bogotá" or "China" versus "People's Republic of China". But besides these heterogeneities, there are also real errors. For example, DBpedia contains a triple that says that "La Paz" is the capital of "Bolivia". In fact, "Sucre" is the constitutional capital of Bolivia, while "La Paz" is only the seat of government. However, in cases where the seat of government is also regarded as capital, the combination "La Paz" and "Bolivia" would have to be added to the trusted reference.

In summary, SDQM indicates that it can be used in Semantic Web environments, such as DBpedia, (1) to spot potential data quality problems according to one's requirements

[79] http://www.geonames.org (Last accessed on June 2nd 2011)
[80] Available at http://download.geonames.org/export/dump/countryInfo.txt (Last accessed on June 10th 2011)

and (2) serve data consumers to quickly analyze a Semantic Web data source regarding their own quality perception. Moreover, the performance of SDQM showed promising results. But we also discovered several problems which have to be considered when using SDQM in Semantic Web settings:

- Agreement about data requirements is much harder to achieve in Web environments than in closed settings due to a greater heterogeneity of world views.
- Heterogeneity and different world views may lead to inconsistent data requirements. E.g. one may define “Cylon” as valid value for gender, while another person defines “Cylon” as invalid value for gender.
- Correction of an open data source, such as Wikipedia, usually requires agreement from the community to persist.
- Heterogeneity makes the definition of data requirements more complicated, since it raises the amount of acceptable states of values.
- The classes of the DBpedia ontology only barely distinguish between real entities and fictitious entities. This again complicates the definition of data requirements. For example the robot “Cy” from the television series “Battlestar Galactica” is considered as a person in DBpedia and, therefore, should have a gender.
- The classes of the DBpedia ontology do not distinguish between historic and currently existing entities. For example the German Democratic Republic is member of the class “Populated Place” in DBpedia.

As part of future work, SDQM could be deployed to the Web to generate commonly accepted data requirements by the Semantic Web community. Therefore, it can efficiently support data quality management on Web-scale and the improvement of Semantic Web data.

9.4 Use Case 3: Consistency Checks Among Data Requirements

In this use case, we intend to demonstrate how SDQM facilitates the automated identification of inconsistent data requirements.

9.4.1 Scenario

A large organization that performs data quality management has many data requirements which are used to improve data quality. The organization uses SDQM. The organization's data requirements have been previously represented via the data requirements wiki of SDQM. The organization seeks for an efficient automatic way to identify conflicting data requirements.

9.4.2 Application Procedure

In SDQM, all data requirements are represented in a common structure that is provided by the DQM vocabulary. The data requirements are themselves represented as data in RDF format. Therefore, we can use standard SPARQL queries to manage the quality of data requirements. In general, there are two different types of inconsistencies between data requirements, namely (1) duplicate, but consistent requirements, and (2) contradicting requirements (cf. Oliveira, Rodrigues, & Henriques, 2005, p. 8). Duplicate requirements typically refer to the same schema elements, i.e. classes and properties, which are tested for requirement violations. Contradicting requirements are two or more requirements about the same schema elements that oppose each other and, therefore, cannot be applied at the same time. In the following, we will provide some example queries that are based on fictitious data requirements. The data requirements are based on the test data with information about suppliers. The examples are separated according to the different types of data requirements, since they require different application procedures.

SDQM's property requirements can in general not become inconsistent due to the enforced naming convention of wiki pages in the data requirements wiki. By convention the property requirement title in the wiki is concatenated from the class and property name. Hence, if the tested class and property is only registered under one name in the data requirements wiki, it will not be possible to create duplicate property requirements. However, the naming convention may be modified to create duplicate requirements for the same property if the use case required capturing heterogeneous and potentially inconsistent requirements. In such cases, the same property may be associated to multiple different requirements. Due to the annotation of each requirement with the

“testedClass” and “testedProperty” properties and their representation in RDF, it is possible to identify duplicate requirements and duplicate inconsistent requirements with standard SPARQL queries. To prove this, we created three property requirements for the property http://www.example.org/suppliers#supplierID. The first property requirement “PR Organization FOO Supplier ID” defines that unique values are required for this property in all instances of the class http://www.w3.org/2006/vcard/ns#Organization. The second property requirement “PR Organization EXAMPLE Supplier ID” refers to the same class and property, but does not define that unique values are required. Thus, the property requirement “PR Organization EXAMPLE Supplier ID” is not consistent with the original requirement “PR Organization FOO Supplier ID”. The third property requirement “PR Organization Supplier ID” consistently defines that unique values are required for this property in all instances of the class http://www.w3.org/2006/vcard/ns#Organization. All of the three requirements make statements about the same tested class and property, but use different representations of the property http://www.example.org/suppliers#supplierID, since the same property has been registered with three different names in the data requirements wiki. Figure 55 shows a generic SPARQL query that identifies duplicate property requirements and its result based on our test data.

```
PREFIX dqm:<http://purl.org/dqm-vocabulary/v1.1/dqm#>
SELECT (?tclassURI1 AS ?Class) (?tpropURI1 AS ?Property) (?dr1 AS ?DataRequirement)
WHERE {
   ?dr1 a dqm:PropertyRequirement .
   ?dr1 dqm:testedClass ?tclass1 .
   ?tclass1 dqm:hasURI ?tclassURI1 .
   ?dr1 dqm:testedProperty1 ?tprop1 .
   ?tprop1 dqm:hasURI ?tpropURI1 .
}
ORDER BY ?tclassURI1 ?tpropURI1 ?dr1
```

Class	Property	DataRequirement
"http://www.w3.org/2006/vcard/ns#Organization"	"http://www.example.org/suppliers#supplierID"	<http://localhost/mediawiki-1.18.2/index.php/Special:URIResolver/PR_Organization_EXAMPLE_Supplier_ID>
"http://www.w3.org/2006/vcard/ns#Organization"	"http://www.example.org/suppliers#supplierID"	<http://localhost/mediawiki-1.18.2/index.php/Special:URIResolver/PR_Organization_FOO_Supplier_ID>
"http://www.w3.org/2006/vcard/ns#Organization"	"http://www.example.org/suppliers#supplierID"	<http://localhost/mediawiki-1.18.2/index.php/Special:URIResolver/PR_Organization_Supplier_ID>
"http://www.w3.org/2006/vcard/ns#Organization"	"http://www.w3.org/2006/vcard/ns#country-name"	<http://localhost/mediawiki-1.18.2/index.php/Special:URIResolver/PR_VCARD_Country_Name>
"http://www.w3.org/2006/vcard/ns#Organization"	"http://www.w3.org/2006/vcard/ns#fn"	<http://localhost/mediawiki-1.18.2/index.php/Special:URIResolver/PR_VCARD_Formatted_Name>

Figure 55: SPARQL query and result displaying duplicate property requirements

In general, it is possible to identify only such duplicate requirements that are inconsistent with each other. Figure 56 shows a SPARQL query and its result that can be used to identify inconsistent unique value rules, in case the requirements have been represented in the DQM vocabulary.

```
PREFIX dqm:<http://purl.org/dqm-vocabulary/v1.1/dqm#>
SELECT (?dr1 AS ?UniqueValueRequirement) (?dr2 AS ?InconsistentRequirement)
WHERE{
?dr1 a dqm:UniqueValueRule .
?dr1 dqm:testedClass ?class1 .
?class1 dqm:hasURI ?class1URI .
?dr1 dqm:testedProperty1 ?prop1 .
?prop1 dqm:hasURI ?prop1URI .
OPTIONAL{
   ?dr2 a dqm:PropertyRequirement .
   ?dr2 dqm:testedClass ?class2 .
   ?class2 dqm:hasURI ?class2URI .
   ?dr2 dqm:testedProperty1 ?prop2 .
   ?prop2 dqm:hasURI ?prop2URI .
   FILTER(str(?prop1URI) = str(?prop2URI) && str(?class1URI) = str(?class2URI) && ?dr1 != ?dr2)
   MINUS{
      ?dr2 a dqm:UniqueValueRule
   }
}
FILTER(bound(?prop2URI))
}
```

UniqueValueRequirement	InconsistentRequirement
<http://localhost/mediawiki-1.18.2/index.php/Special:URIResolver/PR_Organization_FOO_Supplier_ID>	<http://localhost/mediawiki-1.18.2/index.php/Special:URIResolver/PR_Organization_EXAMPLE_Supplier_ID>
<http://localhost/mediawiki-1.18.2/index.php/Special:URIResolver/PR_Organization_Supplier_ID>	<http://localhost/mediawiki-1.18.2/index.php/Special:URIResolver/PR_Organization_EXAMPLE_Supplier_ID>

Figure 56: SPARQL query for identification of inconsistent property requirements

9.4.3 Summary

The above queries are domain independent and can be reused to identify inconsistencies among unique value requirements in a data quality management system that represents its data requirements with the DQM vocabulary. Therefore, data quality management with SDQM is especially useful in large environments with distributed knowledge where it is important to identify inconsistent data requirements that have been created and maintained by several different individuals. However, the demonstrated duplicate and consistency checks are only first steps and do not prove that every data requirement type can be checked for consistency. For example, consistency checks among conditional requirements, timeliness requirements, and functional dependency reference rules have not been evaluated, yet. Moreover, as soon as reasoning is enabled, the identification of duplicates and conflicts may become more complex. Further research is needed in this area, to provide reliable information about the scope of consistency checks that is currently possible with SDQM. But the current results based on this evaluation are a promising first approach that may probably be extendable to other data requirement types.

9.5 Comparison with Talend OS for Data Quality

In this section, we compare SDQM with Talend Open Studio for Data Quality (Talend OS for Data Quality), a conventional data quality software tool from the software company Talend[81]. Talend OS for Data Quality can be used for analyzing the quality of data. It is open-source software that is freely available for download. The comparison is focused on the following issues:

- Representation of data requirements
- consistency checks among data requirements
- data quality monitoring and assessment reporting, and
- performance of data quality analyses

It must be stressed that Talend OS for Data Quality offers many more features, e.g. in the area of data profiling, that are beyond the scope of SDQM and, therefore, not subject of this comparison.

9.5.1 Representation and Management of Data Requirements

In Talend OS for Data Quality, data requirements can be represented with so called "SQL business rules". In order to represent a data requirement with Talend OS for Data Quality, the following three high-level steps are required (cf. Talend, 2012, p. 140ff.):

(1) Create SQL business rule
(2) Create new analysis
(3) Run analysis

As the name implies, SQL business rules are based on the relational query language SQL. The data requirement is thereby represented in SQL code which is later automatically embedded into the WHERE clause of an SQL query. Figure 57 shows an SQL business rule for the identification of missing values in the attribute "city".

[81] http://www.talend.com (Last accessed on June 2nd 2012)

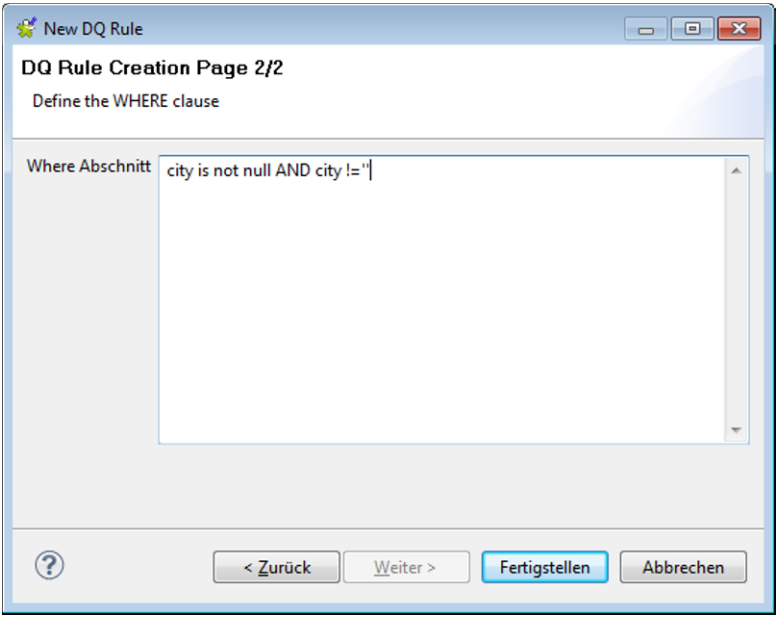

Figure 57: SQL business rule in Talend OS for Data Quality

After the data requirements have been represented as SQL business rules, they have to be attached to a so called analysis. Therefore, a new business analysis object has to be created in Talend OS for Data Quality. The tool provides a wizard for the creation of the analysis object in which the relevant table and the relevant SQL business rules can be chosen from a list. The latter is shown in figure 58. Based on these inputs the analysis can be run to identify requirement violations.

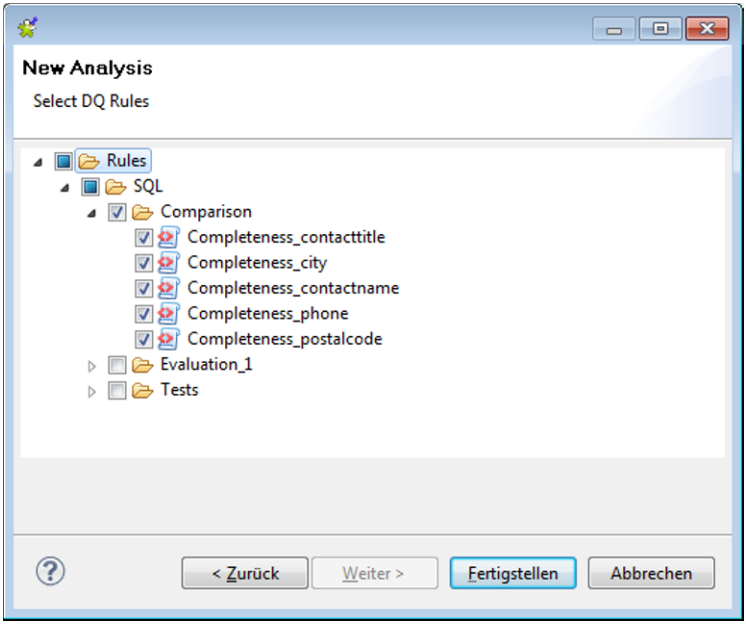

Figure 58: Selecting SQL business rules in Talend OS for Data Quality

In the area of data requirements management, there are three major differences between Talend OS for Data Quality and SDQM. The first difference lies in the way of representing data requirements. Talend OS for Data Quality uses plain SQL coding, while SDQM uses forms to capture data requirements which are automatically converted into RDF data. Other than the users of Talend OS for Data Quality, SDQM's users do not have to know any query language to create data requirements, since they just have to fill in wiki-based forms. The second difference is the location in which the data requirements are created and maintained. In Talend OS for Data Quality data requirements are typically created and maintained on the client of the software installation. Since SDQM uses the data requirements wiki to manage data requirements, they can be created and maintained at Web scale by anyone who has sufficient access rights. Lastly, due to the representation of the data requirements in RDF, it is possible to check consistency among data requirements with SDQM by using standard SPARQL queries. To the best of our knowledge, this is not possible with the data requirements represented in Talend OS for Data Quality, since the requirements are represented in plain SQL. Finally, in Talend OS for Data Quality the data requirements are hard-wired to the actual schema elements of the data source,

whereas SDQM provides a level of abstraction which allows the reuse of the same type of algorithm for multiple different schema elements. Table 25 summarizes the findings of the comparison in the area of data requirements management.

Table 25: Qualitative comparison of SDQM and Talend OS for Data Quality regarding data requirements management

Criterion	**Talend OS for Data Quality**	**SDQM**
Representation of data requirements	SQL	Forms / Wikipage
Location of data requirements	Local	Web
Consistency checks among data requirements	No	Yes
Binding to schema of data source	Direct	Abstract

9.5.2 Data Quality Monitoring and Assessment Reporting

In this section, we compare the data quality reporting capabilities of Talend OS for Data Quality and SDQM. SDQM provides separate reports for data quality monitoring, i.e. the identification of instances with requirements violations, and for data quality assessment, i.e. the computation of dimensional quality scores. In Talend OS for Data Quality, these reports are combined. After data requirements have been represented and integrated into an analysis object, the execution process of Talend OS for Data Quality first computes a score which indicates the percentage to which the requirement has been met. Figure 59 shows such a report in which the completeness scores for five different attributes are shown. Based on this assessment report, it is possible to drill down to the tuples that violate data requirements via the context menu in the red box as shown in figure 59.

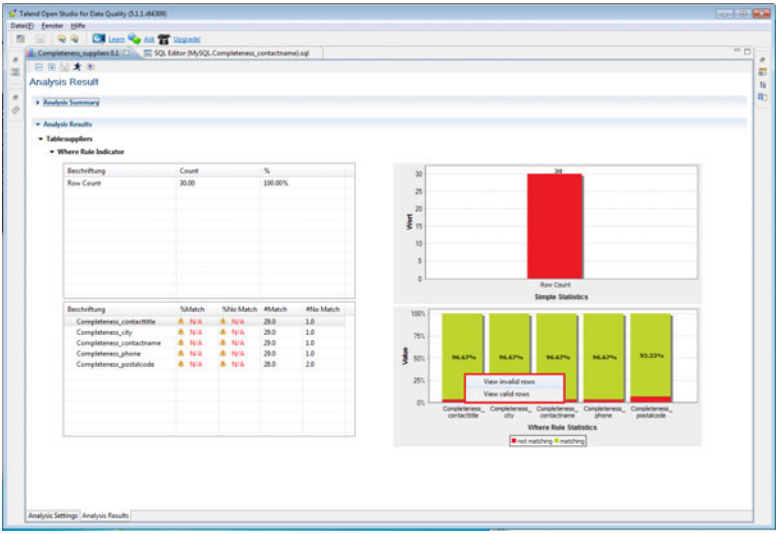

Figure 59: Data quality assessment report in Talend OS for Data Quality

When hitting the menu option “View invalid rows”, an SQL query is automatically executed which retrieves the tuples violating the requirements. Figure 60 shows the result of such a query which can be viewed as the data quality monitoring reports of Talend OS for Data Quality.

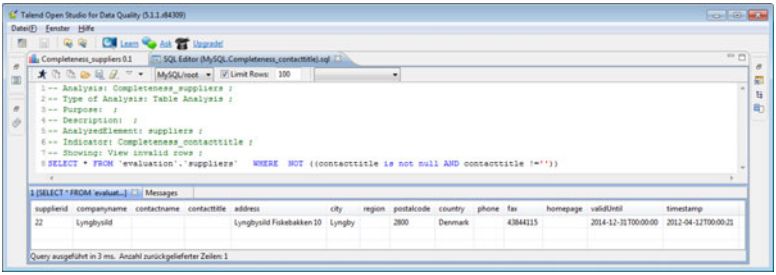

Figure 60: Data quality monitoring report of Talend OS for Data Quality

Hence, in summary we can say that Talend OS for Data Quality and SDQM almost provide the same reports for data quality monitoring and assessment. However, both differ in two issues:

(1) In opposite to the current version of SDQM, Talend OS for Data Quality also visualizes the data quality assessment reporting by providing bar charts.
(2) The reports of SDQM can be made available on the Web, while the reports of Talend OS for Data Quality are only available locally.

Table 26 summarizes the qualitative comparison of Talend OS for Data Quality and SDQM.

Table 26: Qualitative comparison of Talend OS for Data Quality and SDQM regarding data quality reporting

Criterion	**Talend OS for Data Quality**	**SDQM**
Identification of requirement violations	Yes	Yes
Automated computation of data quality scores	Yes	Yes
Graphical visualization of data quality scores	Yes	No
Availability of reports	Local	Web-scale

Moreover, we compared the performance of a DQM architecture with Talend OS for Data Quality and our SDQM architecture. The Talend OS for Data Quality architecture uses a 64bit MySQL database and 4,600 megabytes buffer size. Moreover, we assigned 1,536 megabytes of main memory to Talend OS for Data Quality. This shall represent a similar configuration as used in use case one for the SDQM architecture. For the evaluation of the performance we used the same data corpus for both architectures with one exception: the Talend architecture processed the data in relational format, while SDQM processed it in the triple structure. We executed the same data requirements and created data quality assessment reports in both cases. The results of the performance analysis are listed in table 27.

The performance analysis shows that SDQM still has a significant performance drawback compared to conventional DQM architectures. But it must be stressed that SDQM is an early prototype, while the conventional DQM architecture with Talend OS for Data Quality and MySQL has already matured through practical experience over several years. However, we expect that with the optimization of SDQMgr's queries and with increasing maturity of triplestores the performance gap between both architectures will decrease.

Table 27: Results of performance analysis between Talend OS for Data Quality and SDQM

Report	**Talend OS for Data Quality**	**SDQM (in mm:ss.ms)**
Completeness (5 requirements)	00:23.790	15:59.841
Conditional Completeness (1 requirement)	00:07.800	01:50.137
Syntactic Accuracy (Syntax Rules) (1 requirement)	00:09.937	02:08.727
Syntactic Accuracy (Legal Value Rules) (6 requirements)	00:29.937	27:18.928
Semantic Accuracy (Legal Value Range Rules) (1 requirement)	00:07.504	03:04.716
Semantic Accuracy FDV (1 Condition) (4 requirements)	00:32.402	02:14.406

9.5.3 Summary

In summary, we can say that both architectures, the SDQM architecture and the conventional DQM architecture, have strengths and weaknesses and none of the architectures is superior in general. The strengths of SDQM lie in data requirements management. While Talend OS for Data Quality requires SQL knowledge to create data requirements, SDQM only requires users to fill in wiki-based forms which is much less time consuming and more convenient for business experts who often do not have programming skills. Also, in contrast to DQM tools based on the state-of-the-art, SDQM can identify inconsistencies among data requirements automatically. Moreover, SDQM provides a Web-based user interface for the management of data requirements which facilitates collaboration and the generation of agreement. A shared understanding of data requirements promises a more sustainable and effective improvement of data quality. A local data quality tool, such as Talend OS for Data Quality, hides data requirements in SQL code of client software which hinders the generation of a common understanding about data requirements. SDQM's data requirements are audit-proof due to its version-based storage in Semantic MediaWiki and they can be combined with other information due to the wiki architecture. A major weakness of SDQM compared to the conventional DQM architecture is currently the comparatively slow speed of execution. The current performance of SDQM is acceptable, but far away from the performance of a conventional DQM architecture. As mentioned earlier, the growing use of SDQM and the increasing maturity of triplestore technology will decrease this gap over time. Moreover, the use of Jena's in-memory features may close this gap in the future.