Semantic Technologies for Data Analysis in Health Care

Robert Piro[1(✉)], Yavor Nenov[1], Boris Motik[1], Ian Horrocks[1], Peter Hendler[3], Scott Kimberly[2], and Michael Rossman[2]

[1] University of Oxford, Oxford, UK
robert.piro@cs.ox.ac.uk
[2] Kaiser Permanente, Oakland, USA
[3] IHTSDO, Copenhagen, Denmark

Abstract. A fruitful application of Semantic Technologies in the field of healthcare data analysis has emerged from the collaboration between Oxford and Kaiser Permanente a US healthcare provider (HMO). US HMOs have to annually deliver measurement results on their quality of care to US authorities. One of these sets of measurements is defined in a specification called HEDIS which is infamous amongst data analysts for its complexity. Traditional solutions with either SAS-programs or SQL-queries lead to involved solutions whose maintenance and validation is difficult and binds considerable amount of resources. In this paper we present the project in which we have applied Semantic Technologies to compute the most difficult part of the HEDIS measures. We show that we arrive at a clean, structured and legible encoding of HEDIS in the rule language of the RDF-triple store RDFox. We use RDFox's reasoning capabilities and SPARQL queries to compute and extract the results. The results of a whole Kaiser Permanente regional branch could be computed in competitive time by RDFox on readily available commodity hardware. Further development and deployment of the project results are envisaged in Kaiser Permanente.

1 Introduction

Modern healthcare critically depends on data analysis, particularly in the context of quality assurance. In the US, the National Committee for Quality Assurance (NCQA)[1] specifies a wide range of quality measures; these include, e.g., the proportion of diabetic patients having regular eye examinations, because diabetes can cause retinal damage and eventually blindness. Health Maintenance Organisations (HMOs) are required to demonstrate satisfactory performance w.r.t. NCQA measures if they wish to participate in government funded healthcare schemes such as Medicare that cover more than 48 million patients in the US and represent a substantial share of the healthcare market.

Relevant quality measures can depend on many factors, and their computation may require complex analysis of the data. Moreover, data may be derived

[1] http://ncqa.org/.

P. Groth et al. (Eds.): ISWC 2016, Part II, LNCS 9982, pp. 400–417, 2016.
DOI: 10.1007/978-3-319-46547-0_34

from multiple sources and have heterogeneous structure. Currently, a combination of SAS programs and SQL queries is used to compute quality measures. The resulting software systems are complex, inefficient, and difficult to validate and maintain—a critical issue given that quality measures are regularly revised and augmented.

Semantic technologies offer a possible solution to this problem: RDF can be used to integrate data from heterogeneous sources, ontologies can provide flexible and adaptable schemas, and declarative rules can be used to capture relevant quality measures. A triple store could then be used to apply the rules to the data, with SPARQL queries[2] being used to return the results.

To test this hypothesis, the Knowledge Representation and Reasoning (KRR) group at the University of Oxford, together with the US HMO Kaiser Permanente[3] (KP), undertook a joint project in which they used declarative rules to capture a particularly complex set of quality measures relating to diabetes care, and used these rules with the RDFox [5] triple store in order to compute the corresponding quality measures for the 466,000 patients in KP's Georgia region. The results were extremely encouraging: firstly, only 174 rules were required, compared to the roughly 3,000 lines of complex and hard to maintain SQL code of their previously used solution, which has since been replaced by a vendor product. Secondly, RDFox was easily able to handle the relevant patient data (which amounted to approximately 1.6 billion triples), and computed the quality measures via application of the rules in approximately 30 min. The KP data analyst in charge of quality assurance confirmed that this was fast in comparison to their existing solution, and was also impressed with the small number of iteration cycles needed to check the correctness of our results—a consequence of the relative legibility of the declarative rules.

2 Motivation

The NCQA maintains and publishes the Healthcare Effectiveness Data and Information Set (HEDIS)[4], which uses (relatively) precise natural language to define sets of measures concerning the performance of HMOs in areas such as cancer screening, immunisation and Comprehensive Diabetes Care (CDC). The measures are usually expressed as a percentage of a population of interest and are designed to facilitate performance comparisons across multiple HMOs. In the case of CDC, the quality measures concern diabetic patients in the age range of 18 to 75; one measure, for example, is the percentage of the patients who received an eye exam during the relevant reporting period.

To compute the quality measures, the data first needs to be aggregated from various patient data systems. This typically involves the invention of one or more ad hoc schemas into which the data is cast. Such schemas are designed to facilitate analysis rather than to accurately model the domain, and so they are

[2] http://www.w3.org/TR/sparql11-query/.
[3] http://www.kaiserpermanente.org.
[4] http://www.ncqa.org/HEDISQualityMeasurement.aspx.

difficult to maintain and are prone to inconsistent interpretation by the members of the data analysis team.

Computation of NCQA measures over the aggregated data is typically done via SAS-programs or a sequence of SQL-queries. This process is also complex and error prone; for example, as already mentioned, computation of the CDC measures uses roughly 3,000 lines of SQL code. As a result, existing systems are costly, unreliable, and difficult to maintain.

2.1 Overview of Project

The aim of the project with Kaiser Permanente was to evaluate the effectiveness of Semantic Technologies for computing NCQA quality measures. The power of Semantic Technologies lies in the clearly defined declarative formalisms with which complex relationships can be expressed. One such formalism are Datalog-like rule languages that are supported by many triple stores and that can be used, e.g., to perform OWL 2 RL reasoning [4]. Rules can express relationships via intuitive if-then-statements such as

```
[?Pat, aux:countedFor, aux:measureEyeExam] :-
            [?Pat, rdf:type, aux:diabeticPatient], [?Pat, aux:has, aux:EyeExam] .
```

which says that '*if* the patient has diabetes and an eye exam, *then* the patient is to be counted for the measure Eye Exam'. These statements are succinct and relatively close to natural language.

In this project we used the RDFox triple store with its RDFox-Datalog rule language. Our goal was to investigate whether RDFox-Datalog is expressive enough to encode HEDIS specifications, if RDFox could handle datasets of the necessary size and provide competitive performance w.r.t. existing solutions, and if the resulting semantic technology solution could overcome some of the shortcomings of existing solutions. We decided to implement HEDIS CDC, as its variety of logical connexions between the data is rich, and it is particularly difficult to implement with traditional methods such as SQL and SAS; HEDIS CDC therefore makes an impressive use case for data analysts who conduct HEDIS measurements in HMOs.

The project is split into three tasks. The first task is to create a coherent and extensible data model into which the relevant patient data can be transformed. We used a data model that is close to human conceptualisation as this makes the data easier to understand. The data model makes it also easier to develop and maintain the rules that capture HEDIS measures (see Sect. 3). The second task is the development of such rules. We implemented the HEDIS CDC specification as a Datalog ontology (rule set), but had to augment the rules with SPARQL-queries to fully compute the measures (see Sect. 4). Finally, the approach was evaluated on data provided by the Kaiser Permanente Georgia region. We translated this data into RDF-triples according to our data model, computed the HEDIS CDC measures using RDFox, and compared our results with those computed using existing systems (see Sect. 5).

3 Healthcare Data Modelling in OWL

In this section we describe the conceptual model that we developed to describe the clinical and administrative data in KP. When designing our model we tried to satisfy the following three requirements. Firstly, the model had to be as close as possible to domain expert conceptualisation so as to facilitate the faithful representation of domain knowledge. Secondly, the model had to be sufficiently flexible to uniformly capture the diverse business processes that take place in a typical healthcare organisation. Thirdly, the model had to be readily amenable to Semantic Web technologies. We identified a healthcare data modelling standard from the field of healthcare informatics, called HL7 RIM, that satisfies the first two requirements. To satisfy the third requirement, we used the methodology behind the HL7 RIM to build a conceptual model in OWL. In the following two sections we give a short overview of the HL7 RIM standard and a description of how it was used to build an OWL ontology describing our data.

3.1 The HMO Data Model

Healthcare data modelling is an important topic in the field of healthcare informatics, and a number of standards have been developed to facilitate the exchange of clinical and administrative data between HMOs and other third party organisations. One such standard is the Reference Information Model (RIM)[5], which is issued by the international organisation for standardisation Health Level Seven International (HL7)[6].

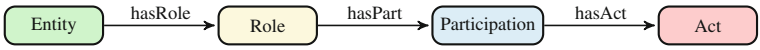

Fig. 1. Core concepts in the RIM, using the standard RIM colour scheme: Act: red, Participation: blue, Role: yellow, Entity: green (Color figure online)

The RIM standard models a wide range of healthcare business processes, including clinical processes, such as clinical visits and laboratory tests, as well as administrative processes of HMOs, such as patient enrolment and insurance plan authoring. All business processes in the RIM are uniformly represented using the notions *entity*, *role*, *participation*, and *act*. Each act is characterised by the participation of entities each of which fulfils a particular role (see Fig. 1). Acts are used to represent business processes, and participations are used to describe the different parties involved in an act, such as the performer of an act and the subject of an act. Entities are used to describe physical things, such as persons and organisations, while roles describe the different competencies of entities, such as employee and patient, in the case of a person, and insurer, in

[5] http://www.hl7.org/implement/standards/rim.cfm.
[6] http://www.hl7.org/.

the case of an organisation. For further details on the RIM standard, please refer to [2,9].

The scope of the RIM data model far exceeds the needs of this project, so we used its underlying design principles to build a simplified data model that better suits our needs. We modelled the different types of entities, roles, participations and acts as OWL classes that are specialisations of the classes Entity, Role, Participation, and Act, respectively. Similarly, we modelled the relationships between these notions as object properties whose domains and ranges are as specified in Fig. 1.

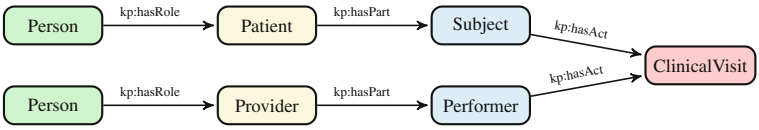

Fig. 2. The model of a clinical visit

Consider for example the part of our conceptual model depicted in Fig. 2, which describes the clinical visits of a patient to their health care provider. The clinical visit is modelled as the class ClinicalVisit, which is a subclass of Act. Similarly, the two entities involved in a clinical visit are modelled as members of the class Person, which is a subclass of Entity. One person in the role of a Provider participates in the clinical visit as a Performer, while the other person in the role of a Patient participates in the clinical visit as a Subject.

Our model also describes properties relevant to clinical visits, such as diagnoses and clinical procedures. In healthcare informatics these concepts are represented by *codes* from standard vocabularies such as ICD-9 [7], which describes diagnosis, CPT, which describes clinical procedures, and SNOMED-CT, which describes clinical terms in general. For example, ICD-9 assigns the code 250.60 to the diagnosis 'diabetes with neurological manifestations', and the code 250.70 to the diagnosis 'diabetes with peripheral circulatory disorders'. We model the ICD-9 concepts using the class ICD9Term, and we connect its instances to the ClinicalVisits in which they occur using the object property kp:hasDiag (see Fig. 3). In healthcare informatics, broader clinical concepts are often modelled as collections of codes, which are commonly referred to as *value sets*. For example, HEDIS defines the term 'Diabetes Diagnosis' as a value set that contains amongst others the ICD-9 codes 250.60 and 250.70. We model value sets using the class ValueSet, and the associations between codes and value sets are realised using the object property kp:hasValueSet. Hence, in our model, the instance of ICD9Term representing the ICD-9 code 250.60 is connected via the object property kp:hasValueSet to the instance of ValueSet that represents 'Diabetes Diagnosis'.

Finally, for each class in the model we introduce datatype properties that are used to specify relevant values. For example, every Person has a specified name,

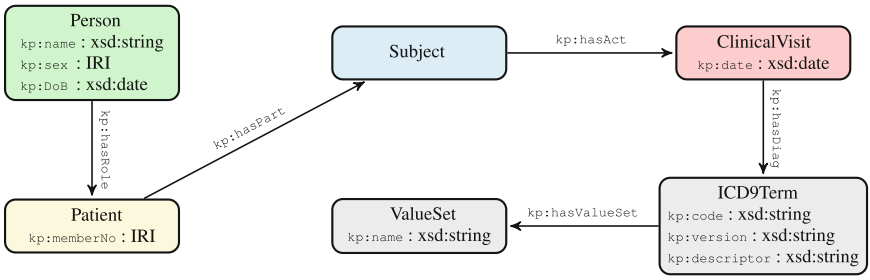

Fig. 3. Extended upper row of Fig. 2 showing how we capture health record data according to the developed schema

sex, and date of birth, every Patient has a member number with the HMO, every ClinicalVisit has a date, every ICD-9 term has a code, a version and a description, and every value set has a name (see Fig. 3).

3.2 Translating KP Data into RDF

We shall refer to the data provided by the Kaiser Permanente regional branch as *raw data*. This is the same data that serves as input to the NCQA approved vendor product, so it was already appropriately aggregated and curated. The data is obtained from KP's internal relational database and consists of delimited text files each of which represents a relational table. For example the file that stores clinical visits looks as follows.

```
VID   MBR     SERV-DT      ...   DIAG-1   ...   DIAG-22   PROVNBR
101   M4711   2013-09-10   ...   250.70   ...   NULL      P8736
```

Each line in this file specifies the visit ID as primary key, the patient's member number, the service date, up to twenty two ICD-9 diagnosis codes, and finally a provider number.

Before translating KP's raw data into RDF-triples, we first had to establish a naming scheme that assigns IRIs to the different objects participating in our model. We chose a naming scheme that allows us to easily map IRIs to the objects in the raw data that they represent. To this end we used IRIs that capture both the types and the identities of the encoded object. This was particularly important when we had to correct formatting errors in our translation, and in the recapitulation stage of the project in which we had to justify our results with the raw data.

For most types of objects the assignment of IRIs was relatively straightforward. For example, we encoded the patient with member number 'M4711' using the IRI <http://www.kp.org/Patient/M4711> and the provider with provider number 'P8736' using the IRI <http://www.kp.org/Provider/P8736>. Similarly, we encoded the ICD-9 code '250.70' using the IRI <http://www.kp.org/ICD9Term/250.70>. Note that, as mentioned above, each IRI captures both the type and the identity of the encoded object.

The assignment of IRIs to clinical visits was slightly more involved. In the initial translation we assumed that each clinical visit is described in a single database record, and thus we used the primary key VID of the record to identify each visit. In the recapitulation stage of the project, however, it became clear that this assumption was wrong, as clinical visit may reside in multiple database records. The data analyst in KP clarified that the identity of a visit is uniquely determined by the date of the visit, the member number of patient, and the provider number. To correct the translation, we therefore encoded a clinical visit using IRIs of the form <http://www.kp.org/Visit/UID>, where UID encodes the slash-separated values of the date, the member number, and the provider number. So, for example, for the clinical visit listed in the record above, UID is equal to 2013-09-10/M4711/P8736. Finally, since there is a one-to-one correspondence between a visit and its subject and a visit and its performer, we use a visit's UID in the IRIs of its subject and performer. Hence, the subject and the performer of the visit in our example are encoded using the IRIs <http://www.kp.org/Subject/UID> and <http://www.kp.org/Performer/UID>, respectively.

Having assigned IRIs to the different objects, we can now easily translate each record of a clinical visit into RDF-triples by simply referring to the data model described in Fig. 3. Some of the triples encoding the clinical visit record in the previous example are given below.

```
<http://www.kp.org/Patient/M4711>rdf:type<http://www.kp.org/Patient>.
<http://www.kp.org/Patient/M4711>kp:hasPart<http://www.kp.org/Subject/UID>.
<http://www.kp.org/Subject/UID>kp:hasAct<http://www.kp.org/Visit/UID>.
<http://www.kp.org/Visit/UID>kp:date"2013-09-10"^^xsd:date.
<http://www.kp.org/ICD9Term/250.70>rdf:type<http://www.kp.org/ICD9Term>.
<http://www.kp.org/Visit/UID>kp:hasDiag<http://www.kp.org/ICD9Term/250.70>.
                                    ...
```

Since the translation of each clinical visit for a given patient uses the same patient IRI, the patient's entire medical history is connected in a contiguous RDF-graph.

Note that our choice of naming scheme allows us to translate each database record independently of other records. As a result, in addition to being relatively simple, the translation could also be easily executed concurrently, as it maintains no global state. Observe, however, that the record-by-record translation results in repetition of triples. For example, the triple asserting that <http://www.kp.org/Patient/M4711> is a member of the class <http://www.kp.org/Patient> will be generated once for every database record that mentions that patient. Similarly, there may be repetitions involving each provider and each diagnosis. As we will see in Sect. 5, this redundancy increases the number of triples by a factor of 5.5.

Finally, we also had to add to our RDF-graph triples related to the HEDIS specification. Firstly, as discussed in the previous section, the HEDIS specification defines a number of value sets. The membership of codes to value sets is naturally encoded using assertions for the object property kp:hasValueSet. Additionally, the HEDIS specification refers to the begin and end dates of the current measurement year, which in our case is the year 2013, as well as to the measurement period, which in our case consists of the years 2012 and 2013. Instead of

explicitly referring to these dates and years in our rules, we exploit the following triple encoding of the relevant information.

```
kp:HEDIS kp:measuredPeriod 2012.  kp:HEDIS kp:beginDate "2013-01-01"^^xsd:date.
kp:HEDIS kp:measuredPeriod 2013.  kp:HEDIS kp:endDate "2013-12-31"^^xsd:date.
```

4 Encoding HEDIS CDC and Its Challenges

This section describes how we encoded the HEDIS specification in RDFox-Datalog. The resulting RDFox-Datalog ontology defines the different patient classes stipulated by the HEDIS specification, e.g. 'patient with eye exam'; we then use RDFox to compute class membership for all patients. Simple SPARQL counting queries determine the number of patients in each class. These numbers are used to calculate the percentage of the population of interest, which is then reported to the NCQA.

As we show in the following, capturing the HEDIS specification involved the use of recursive datalog rules, and hence went beyond what could be achieved via SPARQL query answering alone. We also needed value manipulation, stratified negation, and stratified aggregation, which are not commonly supported reasoning features. In standard materialisation-based triple stores, these features can be simulated by iteratively answering full SPARQL queries, adding the query results to the store, and applying the rules with respect to the enriched data. Since RDFox supports BIND and FILTER constructs in rule bodies, we had to simulate only stratified negation and aggregation.

4.1 Encoding Basic Concepts

HEDIS CDC is specified using natural language. The following extract (which we will refer to as *extract 1*) is drawn from the chapter that defines which patients are diabetic.

> [Diabetics are those patients] who met any of the following criteria during the measurement year [2013] or the year prior to the measurement year [2012] (count services that occur over both years):
> – At least two outpatient visits (Outpatient Value Set), observation visits (Observation Value Set) or nonacute inpatient visits (Nonacute Inpatient Value Set) on different dates of service, with a diagnosis of diabetes (Diabetes Value Set). Visit types need not be the same for the two visits.
> – ...

We first encode some basic concepts, starting with the notion of a *diabetes diagnosis*. Extract 1 specifies that a clinical visit has a diabetes diagnosis if it has a code in the value set named "Diabetes". The following rule classifies such clinical visits by deriving a triple of the form [?CV, rdf:type, aux:diabetesDiagnosis]

where ?CV is an instance of ClinicalVisit and the prefix aux indicates a derived property or class.

```
[?CV, rdf:type, aux:diabetesDiagnosis] :-[?CV, kp:hasDiag, ?ICD9],
                [?ICD9, kp:hasValueSet, ?VS],[?VS, kp:name, "Diabetes"] .
```

We add similar rules to classify outpatient visits, observation visits and non-acute inpatient visits, which derive triples of the form [?CV, rdf:type, aux:outpatient], etc.

We can now associate each patient with their "admissible visits" using triples of the form [?Pat, aux:admissibleVisit, ?CV]. According to Extract 1, a patient's visit counts as admissible if it has a diabetes diagnosis and is also either an outpatient, a non-acute inpatient, or an observation visit. For outpatient visits we thus use the following rule:

```
[?Pat, aux:admissibleVisit, ?CV] :-[?Pat, aux:patientHasAct, ?CV],
        [?CV, rdf:type, aux:outpatient], [?CV, rdf:type, aux:diabetesDiagnosis] .
```

For non-acute inpatient visits and the observation visits we use analogous rules.

Abstractions such as aux:admissibleVisit help to keep subsequent rules shorter and more easily legible. The declarative nature of Datalog allows to introduce such abstractions without the explicit creation of tables. The flexible RDF-schema is simply extended and no data needs to be copied into the new schema.

Finally, we need to ensure that there are at least two admissible visits on different dates in the relevant measurement period. We achieve this with SPARQL BIND and FILTER constructs which RDFox supports in rules bodies. Note that these features can also be simulated by interrupting the reasoning process and computing the relevant values using SPARQL queries, as in the case of stratified negation (see Sect. 4.3).

```
[?Pat, rdf:type, aux:diabeticPatient]:-
         [?Pat, aux:admissibleVisit, ?CV0], [?Pat, aux:admissibleVisit, ?CV1],
         [?CV0, kp:date, ?date0],           [?CV1, kp:date, ?date1],
         BIND( YEAR(?date0) AS ?y0 ),       BIND( YEAR(?date1) AS ?y1 ),
         [kp:HEDIS, kp:measurePeriod, ?y0], [kp:HEDIS, kp:measurePeriod, ?y1]
         FILTER ( ?date0 != ?date1 ) .
```

This rule says that a patient is a diabetic patient if they had two admissible visits in the years ?y0 and ?y1 (computed using BIND), each of which is either the measurement year or the year prior to that, and that the visits occurred on different dates (established using FILTER). The measurement year and the year prior to that are retrieved from the dataset as described in Sect. 3.2. Note that this rule is also non-treeshaped, and cannot be expressed in OWL 2 or its fragments. The non-treeshapedness is unavoidable since we need to compare for each patient the dates of each pair of admissible visits.

4.2 Recursion

We were able to encode all notions discussed so far by using only non-recursive Datalog rules, which means that we could also compute these notions using (large and complex) SPARQL queries. However, as we show next, HEDIS CDC also

contains notions that require genuine recursion, and thus cannot be computed using SPARQL queries alone.

A period in which a patient is insured with a HMO is called an enrolment. Patients often have multiple consecutive enrolments within the measurement year, which is due to changes in circumstances such as retirement, change of workplace or switching between health plan packages. A patient may also have gaps in their enrolment history, because they switched HMOs or were uninsured. However, the NCQA requirements on HMOs apply only to patients who have a *continuous enrolment* with the HMO which, according to the HEDIS CDC specification, is when they have:

> no more than one gap in enrolment of up to 45 days during the measurement year. [...Patients must be insured with the HMO on] December 31 of the measurement year.

To determine whether a patient has a continuous enrolment we proceed as follows. First, we identify as *connected* all enrolment acts that are connected to the end date of the measurement period via a sequence of enrolment acts without any gaps. Second, we identify as *gap-connected* all enrolment acts that are connected to the end date of a measurement period via a sequence of enrolment acts with one gap. Finally, we identify that a patient has a continuous enrolment if they have a (gap-)connected enrolment act containing the begin date of the measurement period.

The notion of connected enrolment act has the following recursive definition. An enrolment act is connected (1) if its period contains the end date of the measurement year, or (2) if it is directly succeeded by a connected enrolment. Case (1) of the definition is handled by the following rule.

```
[?Enr, rdf:type, aux:connEnr] :-
                  [kp:HEDIS, kp:endDate, ?anchor], [?Pat, aux:hasEnr, ?Enr],
                  [?Enr, kp:beginDT, ?dateB],      [?Enr, kp:endDT, ?dateE],
                  FILTER ( ?dateB <= ?anchor && ?dateE >= ?anchor ) .
```

For the recursive case (2), we need to identify the connecting successor. This involves date manipulation because we have to compute the date of the previous day. Unfortunately, SPARQL BIND does not provide arithmetic on the data type xsd:date and instead we had to compute the previous day during data translation which was stored using the data value property kp:beginDT-1.

```
[?Enr, rdf:type, aux:connEnr]:-
                 [?Pat, aux:hasEnr, ?Enr],   [?Pat, aux:hasEnr, ?SuccEnr],
                 [?Enr, kp:beginDT, ?dateB], [?SuccEnr, rdf:type, aux:connEnr],
                 [?Enr, kp:endDT, ?dateE],   [?SuccEnr, kp:beginDT-1, ?prev],
                 FILTER ( ?dateB <= ?prev && ?dateE >= ?prev ) .
```

The notion of gap-connected enrolment can again be defined recursively. An enrolment act is gap-connected (1) if it has a gap of at most 45 days to a connected enrolment act, or (2) if it is directly succeeded by a gap-connected enrolment. Similarly to before, during our translation we precomputed the date that is 46 days earlier than the start date of an enrolment act and stored it using the data property kp:beginDT-46.

Note that we compare all pairs of enrolments of each patient, which is quadratic in the number of patient's enrolments. To reduce the workload, we restricted aux:connEnr to enrolments whose period intersects the measurement year. We measured the outdegree of aux:connEnr, which was maximally 6 and thus manageable.

Finally, we determine if a patient was continuously enrolled using two simple rules that identify all patients having a connected or a gap-connected enrolment act whose period contains the begin date of the measurement period. We are thus able to encode this HEDIS section using just 6 recursive Datalog rules. This compares to 500 lines of heavily commented and involved SQL-code previously used by the Kaiser Permanente.

4.3 Stratified Negation

Stratified negation is a feature that is not commonly supported by RDF-triple stores but that can be very useful when conclusions need to be drawn based on the lack of some information. We next give an example of a HEDIS CDC measure whose computation requires negation, and we describe how it was computed before RDFox was extended to handle stratified negation.

HbA1c is a special type of haemoglobin, whose level is used as an indicator for average blood glucose levels over three months and whose healthy level is below 7 %. HMOs are required to pursue good levels of HbA1c, but only in patients without severe health issues, such as by-pass operations, etc. Concretely, the measure for HbA1c control is computed as $\frac{\#\text{patients in HbA1c denom. with HbA1c}<\%7}{\#\text{patients in the HbA1c denominator}}$, where the HbA1c denominator contains those patients in the population of interest that have no exclusions. For example, HEDIS CDC states in the definition of the HbA1c denominator:

> Exclude members [from the pop. of interest] who meet any of the following criteria:
> - IVD [Ischemic Vascular Disease]. Members who met at least one of the following criteria during both the measurement year and the year prior to the measurement year. Criteria need not be the same across both years.
>   - At least one outpatient visit (Outpatient Value Set) with an IVD diagnosis (IVD Value Set).
>   - At least one acute inpatient encounter (Acute Inpatient Value Set) with an IVD diagnosis (IVD Value Set).

From what we have seen earlier, it is not difficult to imagine how to write rules which identify patients with IVD and with other excluded properties. All final rules computing the excluded patients have the head [?Pat, rdf:type, aux:HasExclusion] and thus mark a patient excluded from the HbA1c denominator for the respective reporting year. Yet, computing the HbA1c denominator requires selecting all patients from the population of interest who do *not* have an exclusion and thus it requires negation.

Stratified negation [1,6] is a well established extension of recursive Datalog and is sufficient for our purposes. However, RDFox did not support stratified negation at the time, so we applied a well-known work-around [3,8] that uses the FILTER NOT EXISTS construct in SPARQL. After populating the class aux:Denominator with the population of interest and the class aux:HasExclusion with the part of the population that needs to be excluded, we halt RDFox's reasoning process and execute the following query.

```
SELECT ?Pat rdf:type aux:HbA1cDenom WHERE {
              ?Pat rdf:type aux:Denominator .
              FILTER NOT EXISTS { ?Pat rdf:type aux:HasExclusion }.}
```

We save the answers as triples into a file, which we then load back into RDFox.

This solution, however, is not optimal in a setting where transparency and proximity to the natural language specification is a major selling point. RDFox has since been extended to support stratified negation. The query can now be expressed as the following rule that can be listed and evaluated together with the other axioms:

```
[?Pat, rdf:type, aux:HbA1cDenom]:-
      [?Pat, rdf:type, aux:Denominator], not [?Pat, rdf:type, aux:HasExclusion].
```

4.4 Aggregates

Aggregate functions collapse multiple inputs into one single value, like 'max', 'count', 'average' but also 'list' or 'set'. The HEDIS CDC specification requires for measurement results always the latest and 'best' reading, if more than one measurement was taken on the same date. For example HEDIS CDC requires to

> [...] identify the most recent BP reading taken during an outpatient visit (Outpatient Value Set) or a non-acute inpatient encounter (Nonacute Inpatient Value Set) during the measurement year. The member is numerator compliant if the BP is <140/80 mmHg. The member is not compliant if [...] the systolic or diastolic level is missing. If there are multiple BPs on the same date of service, use the lowest systolic and lowest diastolic BP on that date as the representative BP.

We first use a rule to classify a clinical visits which has both systolic and diastolic measurements as instances of aux:HasCompleteBP. Amongst these the latest, i.e. the date maximal, measurement has to be determined. Since RDFox had no support for aggregate functions at the time, we used a workaround which incidentally shows the connexion between aggregates such as max, min etc. and negation: We first mark all those visits which have a later visit using the rule

```
[?CV0, rdf:type, aux:HasLaterVisit]:-
   [?Pat, aux:patientHasAct, ?CV0],     [?Pat, aux:patientHasAct, ?CV1],
   [?CV0, rdf:type, aux:HasCompleteBP], [?CV1, rdf:type, aux:HasCompleteBP],
   [?CV0, kp:date, ?date0],             [?CV1, kp:date, ?date1],
   FILTER ( ?date0 < ?date1 ).
```

and we use, as done in Sect. 4.3, a SPARQL query to determine all those clinical visits that do not have a later visit:

```
SELECT ?CV rdf:type aux:latestBP WHERE {
              ?CV rdf:type aux:completeBP .
              FILTER NOT EXISTS {?CV rdf:type aux:HasLaterVisit}.}
```

The answers of the this query are added to the running store in RDFox. We can then populate the class aux:latestBP-140-80 with clinical visits from the class aux:latestBP. Note that our work-around computes aux:hasLaterVisit using quadratically many rule instantiations in the number of the blood pressure measurements per patient per year, and that a native implementation of the aggregate function max can achieve the same in linear time by iterating through all measurements and retaining the binding with the latest date.

5 Performance and Evaluation

We evaluated our approach on a commodity server provided by the Kaiser Permanente data centre. The server was security compliant according to the sensitive nature of the data. All tests were performed on this server and none of the provided data has been transferred outside of the security compliant infrastructure. The server runs Linux RedHat, has 8 Intel Xenon E5-2680 CPUs, clocked at 2.7 GHz and has 64 GB RAM. In what follows we shall first discuss data translation then the computation of HEDIS CDC using RDFox and finally the reconciliation of the results.

5.1 Data Translation

Kaiser Permanente provided the data in several files listed in Table 1. A multi-threaded Scala application translated the data into an RDF-graph. As discussed in Sect. 3.2, the application produced many duplicate RDF-triples expanding the number of triples by a factor of 5.5 from 293 M triples to 1.6 G triples. The translation took 47 min and produced 8 GZip files amounting to 8.8 GB.

Table 1. Files provided by Kaiser Permanente regional branch in Georgia

Content	Records	Size	Content	Records	Size	RDF-graph
Providers	113 k	6.8 MB	Prescriptions	8.9 M	892 MB	Total triples: 1.6 G triples
Members	466 k	84 MB	Labs	28.3 M	1.4 GB	Unique triples: 293 M triples
Enrolment	3.3 M	332 MB	Visits	54 M	8.6 GB	Translation time: 45 min (8 CPUs)

5.2 Computing HEDIS CDC

RDFox imported the 1.6 G RDF-triples in 11 min using 8 threads (Table 2, first row), which, due to duplicate elimination, resulted in a store containing 293 M

unique triples. RDFox's importation process comprises reading, parsing, resolving the IRIs in an internal dictionary, eliminating duplicates and populating the store and its index structures.

The Datalog encoding of the HEDIS CDC specification consists of 174 rules of which approximately 65 % can be expressed in OWL 2 RL. Many of the OWL 2 RL expressible rules contain at most two body atoms whilst longer rules tend to be not tree-shaped and are thus not OWL 2 RL expressible, as the examples in Sect. 4 show.

The evaluation of larger rules, such as those for computing continuous enrolments, incur a high work-load, which leads to unacceptable computation times when applied to the whole RDF-graph. We therefore apply the full HEDIS Datalog ontology on a much smaller subgraph, which we compute using Datalog reasoning, and which contains all the data for the population of interest. To this end, we first identify the patients defined in extract 1 by evaluating the relevant rules on the full RDF-graph. These are simple rules, which RDFox can evaluate efficiently. Next, we use rules to mark all relevant triples connected to the identified patients. Finally, using a SPARQL query, we load the marked triples into a new store, on which we evaluate the full HEDIS Datalog ontology.

This strategy considerably reduces total computation time from 1 h 45 min to 30 min (sum of times in Table 2). Computing and extracting the subgraph on the full RDF-graph takes 13 min (795 s) using all 8 CPUs (Table 2, second row). Just before the subgraph extraction, the memory consumption peaks at 28 % at which RDFox uses 53 Bytes per RDF-triple. In dropping the store that contains the full RDF-graph, we release 18.1 GB of RAM. RDFox imports the subgraph in 32.4 s and consumes 2.5 % of the available RAM (Table 2, third row). The subgraph contains 14,000 patients of which almost all belong to the population of interest. This effectively reduces the original RDF-graph from 293 M triples to 23.4 M triples or to 8 % of its original size. Evaluating the full HEDIS Datalog ontology on the subgraph as well as running the counting queries then takes 4.5 min (258 s) on 8 CPUs (Fig. 2, fourth row). We could not properly compare our performance to the vendor product's performance. The vendor product not only computes HEDIS CDC but all HEDIS measure sets in approximately 8 h. The vendor product generally acts as black box and it is not possible to separate all different stages of computation from outside. Loading and initialising the database takes the vendor product 1 h. Then it executes a 4 h long pre-processing step which also includes the computation of the expensive continuous enrolment. The following stage contains an 18 min phase which can be associated with HEDIS CDC. Lastly the computation times of the vendor product were achieved on the more powerful licensed production server which has 16 CPUs but was not at our disposal.

5.3 Reconciliation of Results

For each category, we compared the membership numbers output by RDFox with those output by the vendor solution, and we found differences. The results are reported in Table 3. The row 'RDFox' reports the number of patients computed by RDFox. 'RDFox+' reports the number of excess patients which were not

Table 2. Computing HEDIS CDC with RDFox

	Patients	RDF-triples	RAM	% of 64G B	Time
Import	466 k	1.6 G (293 M)	17.8 GB	28 %	661 s
Extract 1 and extraction	466 k	367 M	18.1 GB	28 %	795 s
Import subgraph	14 k	23.4 M	1.6 GB	2.5 %	32.4 s
CDC numerators/counting	14 k	32.0 M	1.6 GB	2.5 %	258 s

included in the results of the vendor solution. Analogously 'Vendor+' shows the number of patients computed by the vendor that were not included in the RDFox results. 'CDC denominator' reports the population of interest, whilst each other category is a subset of the CDC denominator. The RDFox excess of 4 reported for the CDC denominator propagates through all other categories. We therefore indicate with the second summand for each category, how many of the excess patient were contained in the excess of the CDC denominator.

All results that were computed by RDFox were approved by the HEDIS data analyst. We shall shortly explain why discrepancies still remain. For each derived triple, RDFox can provide a proof tree that shows the rule instances and the RDF-triples in the RDF input graph which contributed to its derivation. Using these RDFox explanations and the information encoded in the IRIs (see Sect. 3.2), we can easily look up the records in the raw data and find the diagnosis codes in question. We can thus argue the correctness of RDFox's deviation directly. We showed, for instance, that RDFox's CDC denominator excess is actually correct and should also be output by the vendor solution. It is however much more difficult to argue why an excess in the Vendor product occurs. The vendor product only gives hints as to why it counts a patient into a certain category. For example it prints out the relevant visit date which is meant to help looking up the triggering visit. However in the case of Nephrological Attention, these visit dates of patients in the vendor excess could not be found in the raw data and it was not possible to explain the origin of these dates. The lack of explanations is a clear and typical short coming of traditional solutions.

Table 3. Computed numbers by RDFox and deviations

Results	CDC Denom	LDL-C		BP		Eye exam	Neph attent	HbA1c			HbA1c <7 %	
		Lab	<100 mg/dl	<140/80	<140/90			Lab	<8 %	>9 %	Denom	Lab
RDFox	14402	13217	7952	8963	11442	5430	13204	13474	9465	3132	8939	3702
RDFox+	0 + 4	0 + 3	0 + 2	3 + 3	0 + 4	0	2 + 3	0 + 3	0 + 3	0 + 1	1 + 3	0 + 2
Vendor+	0	0	0	0	0	1692	230	0	0	0	13	5

Since the HbA1c <7 % denominator uses negative information, the roles were reversed. We could show using RDFox explanations that all 13 patients in the vendor product's excess had an explicit exclusion and should not be counted. This excess propagates into HbA1c <7 % Lab. The minor RDFox excess in BP

140/80 could be traced to us interpreting a rule in a different way, which was subsequently approved by the HEDIS help-desk. The large discrepancy in Eye Exam was due to data that was not delivered by Georgia region.

6 Lessons Learnt

The project was very successful and we have learnt useful lessons in particular with regards to encoding and representing the data. However, the project also revealed some limitations of Semantic Technologies and suggested several ways in which they could be adapted to better fit data analysis applications of this kind.

Expressivity of the Ontology Language. The project revealed that OWL 2 alone is insufficient to compute the HEDIS measures. As we saw in Sect. 4.1, non-treeshaped expressions are necessary to determine the diabetic population. However, OWL 2, and consequently its tractable fragment OWL 2 RL, prohibits such expressions in order to ensure decidability. Furthermore, OWL 2 supports neither stratified negation nor stratified aggregation. As witnessed in Sects. 4.3 and 4.4 the absence of such constructs necessitates the introduction of non-declarative workarounds that make the behaviour of the system as a whole more difficult to understand. Finally, as we saw in Sect. 4.2, value manipulation during reasoning is an important language feature for data analysis applications. Although unrestricted value manipulation endangers the termination, non-recursive value manipulation preserves the termination guarantee and, and as seen in this project, is sufficient to encode the HEDIS specification.

Use of RDF as Data Format. RDF restricts the user to triples which correspond to unary and binary predicates. Hence rule bodies feature a large number of atoms, as n-ary relations have to be reified. Within rules this amounts to a named parameter perspective, since predicate names appear as constant in every rule body atom. It is therefore helpful to have meaningful predicate names which also indicate whether or not a triple is derived, as this makes rules legible and comprehensible. We also applied naming conventions that indicate which class of the data model an individual instantiates and from which data it originates. Both allowed us to debug and judge the correctness of the HEDIS encoding much faster. The resulting large number of joins that need to be performed in order to evaluate rules with many body atoms is not prohibitive in practical applications as RDF-triple stores are optimised for computing these large numbers of joins.

Due to the flexible schema of RDF, the data has a fully normalised representation. In particular, our data does not contain any null values and, for example in the case of clinical visits allows a variable number of diagnosis per clinical visit. The flexibility of RDF also helped us in the recapitulation stage of the project, in which, as discussed in Sect. 3.2, it became apparent that the data pertaining to a given clinical visit might be spread over multiple records in the raw data. This led to the misclassification of certain patients. The solution was simply to modify the assignment of IRIs to clinical visits in the data translation

phase, which effectively merged database records referring to the same visit. Due to the flexibility of RDF, we did not have to change the conceptual schema or the way in which we compute the HEDIS measures.

The RIM modelling standard. Successful deployment of (semantic) technology also requires addressing 'soft' issues such as user expectations and familiarity. In this project it was crucial to win the support of the domain experts, who are the future users. This can be achieved by exploiting modelling standards in the respective field as we did with HL7 RIM. Following these conceptualisations makes it easier to argue clarity and intuitiveness of Semantic Technologies which are their major selling points.

We also used the RIM modelling standard to structure the types of clinical processes that occur in our project, which allowed us to uniformly represent healthcare data regardless of whether it describes visits, prescriptions or lab results. Due to the uniformity, the data model could be more easily memorised which facilitated rule authoring as it was not necessary to frequently refer to the data model documentation

7 Conclusion

In this paper we described the project conducted in collaboration with Kaiser Permanente to investigate the benefits of using Semantic Technologies in data analysis. Using the RIM modelling standard, we developed a schema ontology that mirrors how domain experts conceptualise business processes in healthcare, and we translated the raw data into an RDF-graph following this schema. With this data model in hand we encoded in RDFox-Datalog the HEDIS CDC specification which is renowned for its complexity. The declarative nature of RDFox-Datalog allowed us to succinctly express HEDIS CDC in a rule ontology which is close to the language of the specification. During the development of the rule ontology RDF proved to be a flexible data format that keeps the vocabulary explicit and thus confers its legibility to the rules.

The process of evaluating the rules on the patients and reconciling the results exceeded our and the HEDIS data analyst's expectations. The data of 466 000 patients fit easily into memory and the results were computed on modest resources within 30 min using the highly efficient triple store RDFox. Due to RDFox's good scalability we are confident that we could significantly reduce this time on a machine with more threads such as the vendor licensed production server.

The HEDIS data analyst noted that we had very few discrepancies from the outset and appreciated the ease with which changes and amendments could be done, not least because the data model and the rules provided comprehensible context. The explanation facilities in RDFox allowed us to easily trace discrepancies into the raw data. This reduced the number of development cycles of our application and we even discovered problems with the vendor solution. All results computed by our solution using RDFox were approved by the HEDIS analyst.

With this project we have successfully demonstrated the advantages of Semantic Technologies over traditional solutions in the context of data analysis

in healthcare, and we are planning a further project with Kaiser Permanente in which the approach will be extended to all of HEDIS and all of their regions. The project also shows a possible avenue for applications of Semantic Technologies in encoding regulatory corpora in the field of data analysis in general, and demonstrates that it has the potential to cut development and maintenance costs in business settings.

Acknowledgments. The project was funded by the DBOnto Platform Grant, the MaSI^3 Fellowship, and Kaiser Permanente. Thanks are particularly due to Alan Abilla, Andy Amster, Patrick Courneya, Paul Glenn, Peter Hendler, Joseph Jentzsch, Scott Kimberly, and Mike Sutten, without whom this project would not have been possible.

References

1. Apt, K.R., Blair, H.A., Walker, A.: Towards a theory of declarative knowledge. In: Foundations of Deductive Databases and Logic Programming. pp. 89–148 (1988)
2. Benson, T.: Principles of Health Interoperability HL7 and SNOMED. Springer, New York (2010)
3. Chaussecourte, P., Glimm, B., Horrocks, I., Motik, B., Pierre, L.: The energy management adviser at EDF. In: Alani, H., et al. (eds.) ISWC 2013. LNCS, vol. 8219, pp. 49–64. Springer, Heidelberg (2013)
4. Krötzsch, M.: OWL 2 profiles: an introduction to lightweight ontology languages. In: Eiter, T., Krennwallner, T. (eds.) Reasoning Web 2012. LNCS, vol. 7487, pp. 112–183. Springer, Heidelberg (2012)
5. Nenov, Y., Piro, R., Motik, B., Horrocks, I., Wu, Z., Banerjee, J.: RDFox: a highly-scalable RDF store. In: Arenas, M., et al. (eds.) ISWC 2015. LNCS, vol. 9367, pp. 3–20. Springer, Heidelberg (2015)
6. Ross, K.A.: Modular stratification and magic sets for datalog programs with negation. In: Proceedings of the ACM Symposium on Principles of Database Systems, pp. 161–171 (1990)
7. Slee, V.N.: The international classification of diseases ninth revision (ICD-9). Ann. Intern. Med. **88**(3), 424–426 (1978). http://dx.doi.org/10.7326/0003-4819-88-3-424
8. Tao, J., Sirin, E., Bao, J., McGuinness, D.L.: Integrity constraints in OWL. In: Proceedings of the 24th AAAI Conference, AAAI 2010, Atlanta, GA, USA (2010). http://www.aaai.org/ocs/index.php/AAAI/AAAI10/paper/view/1931
9. Vizenor, L., Smith, B.: Speech acts and medical records: the ontological nexus. In: Proceedings of the International Joint Meeting EuroMISE 2004 (EuroMISE, Prague, CZ) (2004)