DataKnots: A Framework for Building Domain Specific Query Languages

Clark C. Evans; Kyrylo Simonov, PhD

DataKnots[1] is an extensible data processing framework. It lets collaborative research teams build their own domain specific query languages (DSQLs) that reflect their conceptual models and vocabularies. DataKnots is based upon an algebraic framework, Query Combinators[2], which formalizes how query components can be defined and combined in a consistent manner. While DataKnots includes standard query operations (group, filter, sum, etc.), they are not given special treatment over domain specific operations. One can add new query components by encapsulating existing queries, lifting Julia language subroutines to queries, and authoring novel transformations using a pipeline construction interface. DataKnots is a platform for creating an ecosystem of mutually interoperable DSQLs, so that collaborative research groups could easily customize their query systems to fit the kinds of data sources and analysis methods they use.

To show that ergonomic, high-performance DSQLs could be rapidly constructed, we built a DSQL inspired by the Clinical Quality Language (CQL) and used it to implement CMS124v7 “Cervical Cancer Screening”. Specifically, we constructed a processing pipeline that converts JSON encoded Fast Healthcare Interoperability Resources (FHIR) to its clinical quality measure (CQM) score. The pipeline loads JSON to an in-memory FHIR representation, converts to an intermediate Quality Data Model (QDM) form, then implements CMS124v7 logic to calculate the CQM score. This layered approach separates concerns, giving us a place to put encoding logic specific to FHIR that doesn’t belong in a CQM, as well as a place to isolate electronic health record vendor differences.

DataKnots4FHIR[3] took 27 working days to implement. New measures can now be added with incremental effort. We benchmarked it using synthetic patient bundles from Synthea. Computation of CMS124v7 over 1,000 patients averages 76ms per patient with a single core on a i7-4770 desktop computer, while the reference implementation[5] averaged 607ms per patient. Our bottleneck is memory usage, with a high-water mark of 11mb per patient. JSON parsing is expensive (17ms/patient). These benchmarks motivate future work on a custom JSON parser, suitable to our vectorized representation, that extracts FHIR lazily based upon the exact fields needed for a given computation.

CQL is a highly-targeted DSQL, created specifically for implementing clinical quality measures and decision logic. Using DataKnots, we were able to rapidly construct a DSQL that matches CQL with comparable functionality and ergonomics. Moreover, we were able to represent not only a CQM calculation, but the entire processing pipeline, including conversion from JSON to FHIR and conversion of FHIR to QDM. For this application, DataKnots was extended with relevant data types, including concepts and value-sets from clinical vocabularies. Informed by clinical quality measure guidelines, we also defined a datetime interval with operators, such as **`and_previous`** and **`during`**. Critically, these domain specific operators are treated no differently from built-in operations such as **`filter`**. Because its formalized approach permits new query operators to be seamlessly integrated, DataKnots can be used to build distinct DSQLs, each having a conceptual model and vocabulary responsive to its research domain and audience.

```
define "PapTest Within 5 Years"
 ("Pap Test with Results" PapTestOver30YearsOld
  with ["Patient Characteristic Birthdate"] BirthDate
    such that Global."CalendarAgeInYearsAt"(
           BirthDate.birthDatetime,
           start of PapTestOver30YearsOld.relevantPeriod) ≥ 30
    and PapTestOver30YearsOld.relevantPeriod 5 years or
                less before end of "Measurement Period")
```

CMS124v7 fragment using CQL with QDM

https://ecqi.healthit.gov/sites/default/files/ecqm/measures/CMS124v7.html

```
@define PapTestWithin5Years =
  let birthDate ⇒ PatientCharacteristicBirthdate.
                    BirthDateTime,
      previous5years ⇒ interval(MeasurePeriod.end).
                          and_previous(5years)
    PapTestWithResults.
    filter(years_between(relevantPeriod.start, birthDate) ≥ 30
          && relevantPeriod.during(previous5years))
 end
```

an equivalent using DataKnots – cms124.jl

https://github.com/rbt-lang/DataKnots4FHIR.jl/blob/master/doc/src/cms124v7.jl

DataKnots is MIT/Apache licensed, is well documented, and has extensive regression tests. Our approach has multiple applications: we have additionally prototyped a DSQL for Observational Health Data Sciences and Informatics (OHDSI) cohort construction[4]. We are actively searching for a pilot project.

1. Evans CC, Simonov K, DataKnots Query System for Julia (https://github.com/rbt-lang/DataKnots.jl/)
2. Evans CC, Simonov K, Query Combinators (https://arxiv.org/abs/1702.08409)
3. Evans CC, DataKnots4FHIR : Query Adapters for FHIR (https://github.com/rbt-lang/DataKnots4FHIR.jl/)
4. Evans CC, Simonov K, DSQLs for Medical Research (https://www.biorxiv.org/content/10.1101/737619v2)
5. Database Consulting Group, CQL Measure Processing Component (https://github.com/DBCG/cqf-ruler)