Cooking with CQL & FHIR Qs&As - Session 37

Thursday, July 25, 2019

Measure Logic in CQL

Q: What is the best way to specify in Clinical Quality Language (CQL) that a medication was present on admission for an inpatient encounter? There have been conversations recently about a "presentOnAdmissionIndicator" for diagnoses in Quality Data Model (QDM) 5.5, but not something equivalent for medications.

A: This will be documented as “Medication Active at Admission”.

```
define "Medications Active at Admission":
   ["Medication, Active"] M
        with ["Encounter, Performed"] E
            such that M.relevantPeriod contains start of E.relevantPeriod
```

The Medication, Active must have a relevant period that includes the start date of the encounter. Note also that this is only addressing the simplest case of an inpatient admission. It may need to be expanded to account for an immediately preceding emergency department visit, depending on measure intent. Also note that the Quality Data Model (QDM) specifies relevant period start for a Medication, Active as the time when the patient is first known to have been taking the medication. Whether this is captured by the electronic health records is a question for investigation.

Measure Logic in CQL

Q: How do you represent Structured Query Language (SQL) joins in Clinical Quality Language (CQL)?

A: In Structured Query Language (SQL), joins are used to combine data from multiple tables. In Clinical Quality Language (CQL), the focus is on simplest possible expression of single-source queries. But multi-source queries are possible as well. Some examples:

```
define "Semi-join Example":
  ["Encounter, Performed": "Outpatient"] Encounter
   with ["Laboratory Test, Performed": "Streptococcus Test"] Test
     such that Test.resultDatetime during Encounter.relevantPeriod


define "Semi-minus Example":
 ["Encounter, Performed": "Outpatient"] Encounter
   without ["Laboratory Test, Performed": "Streptococcus Test"] Test
    such that Test.resultDatetime during Encounter.relevantPeriod

define "Join Example":
from
  ["Encounter, Performed": "Outpatient"] Encounter,
  ["Laboratory Test, Performed": "Streptococcus Test"] Test
where Test.resultDatetime during Encounter.relevantPeriod
return { Encounter: Encounter, Test: Test }
```

```
define "Join Example with Select":
from
  ["Encounter, Performed": "Outpatient"] Encounter,
  ["Laboratory Test, Performed": "Streptococcus Test"] Test
where Test.resultDatetime during Encounter.relevantPeriod
return {
   encounterId: Encounter.id,
   encounterCode: Encounter.code,
   encounterRelevantPeriod: Encounter.relevantPeriod
   testId: Test.id,
   testCode: Test.code,
   testResultDatetime: Test.resultDatetime,
    testResult: Test.result
}
```

```
define "Cartesian-product Example":
from
  ["Encounter, Performed": "Outpatient"] Encounter,
  ["Laboratory Test, Performed": "Streptococcus Test"] Test

The Default result from a multi-source query is a tuple with an element for each query source
{ Encounter: "Encounter, Performed", Test: "Laboratory Test, Performed" }
```

```
define "Left-outer-join Example":
 ["Encounter, Performed": "Outpatient"] Encounter
   let Test: singleton from (["Laboratory Test, Performed": "Streptococcus Test"] LabTest
     where LabTest.resultDatetime during Encounter.relevantPeriod)
   return { Encounter: Encounter, Test: Test }
```

Note, there is an open source project that allows you to translate CQL to SQL. It was a pilot project in the Healthe-Decisions initiative. It is available on the Clinical Quality Framework Repository. It's a little outdated, but the structure is there.

You can have a native environment that runs SQL directly. There are vendor systems that can do this and three open source implementations of a SQL engine that runs SQL directly. On the GitHub Wiki, there is a community projects page that has links to all of those open source implementations.

Functions in CQL

Q: Does Clinical Quality Language (CQL) support a mechanism for randomly picking an item from a list?

A: The short answer is no. Clinical Quality Language (CQL) is purposely a deterministic language, so we don't have functions like Random(). This was a design decision in the very earliest phases of language development to ensure that systems could reliably cache intermediate results. Even functions like Today() and Now() are defined deterministically so that they return the same value within any given evaluation session. However, CQL 1.3 introduced the ability to define external libraries. From the perspective of the translator, external definitions are just

function signatures; it's up to the engine to resolve external definitions and provide the run-time implementation for them. For engines, this would be fairly straightforward to support, but

- It would tie the CQL that uses the external definitions to engines that understood those externals (reducing portability)
- The Java-based open source engine does support external libraries, but it's not a feature that has been well-tested.

Using FHIR

Q: How would some of the most commonly used datatypes in Quality Data Model (QDM) be expressed in a Fast Healthcare Interoperability Resource (FHIR)-based measure?
A: The common datatype examples below are expressed in Fast Healthcare Interoperability Resource (FHIR)-using Quality Data Model version 5.4, FHIR version 4.0.0, and include FHIRHelpers version 4.0.0.

1. QDM Diagnosis to FHIR Diagnosis from CMS72v8:

```
define "QDM Diagnosis":
  [Diagnosis: "Intravenous or Intra arterial Thrombolytic (tPA) Therapy Prior to
Arrival"]


FHIR Equivalent:
define "FHIR Diagnosis":
       [FHIR.Condition: "Intravenous or Intra arterial Thrombolytic (tPA) Therapy Prior
to Arrival"] PriorTPA
       where PriorTPA.clinicalStatus in "Active Condition"


NOTE: onset and abatement need to be considered, and there is an outstanding question
as to whether the clinical status element would always be consistent with onset and
abatement.
```

2. Next datatype is Encounter, Performed using the example from MATGlobalCommonFunctions:

```
define "QDM Encounter Performed":
       ["Encounter, Performed": "Encounter Inpatient"]


FHIR Equivalent:
define "FHIR Encounter Performed":
       [Encounter: "Encounter Inpatient"] EncounterInpatient
              where EncounterInpatient.status = 'finished'

NOTE: Encounter, Performed in QDM can include "in progress" encounter, so it's not
necessarily always just 'finished', the point is status needs to be checked consistent
with measure intent.
```

   QDM to QI Core Mapping would be a good starting point. This goes through all of the QDM datatypes and what the mapping to FHIR looks like.

3. Next datatype is Medication, Order using the example from VTE-1.

```
define "QDM Medication Order":
   ["Medication, Order": "Low Dose Unfractionated Heparin for VTE Prophylaxis"]

FHIR Equivalent:
define "FHIR Medication Order":
  ["MedicationRequest": "Low Dose Unfractionated Heparin for VTE Prophylaxis"] M
        where M.intent = 'order'
```

4. Last datatype is Procedure, Performed using the example from VTE-1.

```
define "QDM Procedure Performed":
  ["Procedure, Performed": "Comfort Measures"]

FHIR Equivalent:
Define "FHIR Procedure Performed":
 ["Procedure": "Comfort Measures"] P
        where P.status = 'completed'
```