Cooking with CQL Qs&As

Thursday, April 23, 2020

Using Quality Data Model (QDM)

Q: In the Quality Data Model (QDM) 5.5, what does the expand operator do? Can it be used to make a list of each day in the measurement period?

A: QDM 5.5 does not have an expand operator, QDM is the data model. The CQL expand operator is an operator that takes a list of intervals and returns the unit intervals in that input. As shown in Example 1, if you expand the interval from 1 to 10, the result would be a list of the intervals from 1 to 1, 2 to 2, and so on.

Example 1

```
expand { Interval[1, 10]} // { Interval[1, 1], Interval[2, 2] ...}
```

There is also a per clause, shown in Example 2. If you expand the measurement period, you can get intervals of milliseconds by saying per day then you get intervals of days. If you said per month, you would get 12 months and then you could count the encounters in that month.

Example 2

```
define "Measurement Period Days":
                 expand { "Measurement Period" } per day
```

Measurement Period is a calendar calculation. If you are using Measurement Period Days and the measurement period includes a leap year, that will be accounted for in the calculation. Similarly, if you said month, you will get the calendar month intervals. You could also ask for the months, the counts in the days, and the highest count of days in that month.

Measure Logic in CQL

Q: In the example below, N (line 8) is the alias of Ambulatory Numerator. Does that definition return a list of the dates?

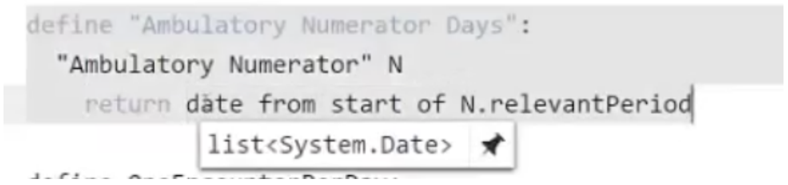
```
define "Ambulatory Numerator Days":
     "Ambulatory Numerator" N
       return date from start of N.relevantPeriod

define OneEncounterPerDay:
     "Ambulatory Numerator Days" D
       return Last(
         "Ambulatory Numerator" N
           where N.relevantPeriod starts same day as D
           sort by end of relevantPeriod
       )
```

A: Ambulatory Numerator is an alias in the CQL using list of elements, each described by a QDM "Encounter, Performed". In the first line of the example, the Ambulatory Numerator Days definition is used to range over the days that had an encounter that ended. Since we are dealing with ambulatory, it's unlikely that the encounter will be more than one day, but it is a possibility.

Q: When using the Quality Data Model (QDM) 5.5 as shown in the example, does this return a date?

```
define "Ambulatory Numerator Days":
      "Ambulatory Numerator" N
        return date from start of N.relevantPeriod
```

A: Yes, the return is saying for every entry in the `"Ambulatory Numerator" N` input, return this `date from start of N.relevantPeriod` output. Since the Ambulatory Numerator is a list of encounters, this expression will be evaluated for each encounter in that list so you'll get the dates. The result of that expression is then a list of System.Date as shown in the popup:



```
define "Ambulatory Numerator Days":
  "Ambulatory Numerator" N
    return date from start of N.relevantPeriod
                  list<System.Date>
```

Functions in CQL

Q: Encounters do not end in the Electronic Health Record (EHR) until closed. Completion of documentation could occur 3 days after the patient has left even though they were only there for 20 minutes. Most implementers figure out how to say the encounter has ended. They either say Departure Time and apply that to the ended encounter or they automatically end all encounters for the measure calculation at the end of the day. While Fast Healthcare Interoperability Resource (FHIR) or Quality Data Model (QDM) allows you say the end, that is not really where the EHR sees it as an end. The implementer has to figure that out and it can get a little complicated. Would it be reasonable to say the beginning of the encounter starts on that day, if you are talking ambulatory, because you're looking for the number that started that day? Is that significantly different? I am asking that question to decrease implementer variability.

A: Thank you for bringing up this concern. The QDM User Group addressed this issue on June 19, 2019 (link to minutes) based on the QDM-235 Jira tracker item. The QI-Core Implementation Guide STU 4 (v4.0.0 for FHIR 4.0.1) also includes encounter timing guidance. The resulting guidance for QDM "Encounter, Performed" *relevantPeriod endTime* is that implementation sites determine local mechanisms to overcome the issue. Feedback on the QDM User Group identified two approaches. Some use the check-in and check-out time for the actual encounter timing noting that the encounter may actually be opened ahead of the visit to prepare before the patient's arrival and remain open long after the patient departs, perhaps days later. Another implementer automatically assigns an encounter end time as 23:59 of the day the encounter occurred. Your approach to limit measure expressions to compare actions only to ambulatory encounter start times may work. However, the intent of the measure may require an action after the patient departs. As you note, this is a complicated workflow issue. Hopefully the discussion in the QDM User Group minutes and the QI-Core documentation will be helpful. There are different implementation variations happening and that is the challenge.

Using Fast Healthcare Interoperability Resources (FHIR)

Q: When using Clinical Quality Language (CQL) with Fast Healthcare Interoperability Resource (FHIR), FHIR has some things called Extensions and Slices. What's the difference between them? In which scenario should we use them?

A: An extension is a special kind of slice since it is a repeating element and is always discriminated by a URL. Whereas a slice can be discriminated by any of the elements.

In the blood pressure profile example below, you have a component. If you are not saying something different, but simply slicing to put some constraints on it, then you use a slice. However, if you want to say something new, then you use an extension.

```
define TestSlices:
      [Observation: "Blood Pressure"] BP
        let
          SystolicBP: singleton from (BP.component C where C.code ~ "Systolic blood
    pressure"),
          DiastolicBP: singleton from (BP.component C where C.code ~ "Diastolic blood
    pressure")
        where SystolicBP.value < 140 'mm[Hg]'
          and (DiastolicBP.value < 90 'mm[Hg]'


define TestSlices:
      ["observation-bp"] BP
        where BP.SystolicBP.value < 140 'mm[Hg]'
          and BP.DiastolicBP.value < 90 'mm[Hg]'
```

Q: For the current Fast Healthcare Interoperability Resource (FHIR) resource version, implementers created an extension due to some missing information that was required for Quality Improvement Core (QI-Core). For the next FHIR version, will this extension be replaced by a formal element?

A: Extensions are defined to support use cases that are not covered by the base resource. If the extensions are broadly applicable, they could be submitted back to the base specification to be considered for inclusion as an element on that base resource. The committee that stewards the base resource would need to go through the consensus process before that extension can be replaced by a formal element. During the consensus process, the extension would be evaluated to determine whether it is broadly applicable and if so, it would be replaced by a formal element in the resource. Extensions are not automatically included in the next FHIR version until they go through this consensus process to ensure they are applicable universally.

Q: Regarding Slices in Quality Improvement Core (QI-Core) version 4.0.0, how do you get the blood pressure profile in the example? Is it using the name or the code? Because it is not really defined with the condition we are going to follow. Right?

```
define TestSlices:
      [Observation: “Blood Pressure”] BP
        let
          SystolicBP: singleton from (BP.component C where C.code ~ "Systolic blood
    pressure"),
          DiastolicBP: singleton from (BP.component C where C.code ~ "Diastolic blood
    pressure")
        where SystolicBP.value < 140 'mm[Hg]'
          and (DiastolicBP.value < 90 'mm[Hg]'
```

```
define TestSlices:
      [“observation-bp”] BP
        where BP.SystolicBP.value < 140 'mm[Hg]'
          and BP.DiastolicBP.value < 90 'mm[Hg]'
```

A: The slice is defined in the profile and this profile’s code is Blood Pressure (BP). You would use the print name. It is important to note that typically when we say BP in a measure, we are talking about more than one code. It is a value set with a specified list of codes that constitutes all the values of BP. However, for observation BP, Fast Healthcare Interoperability Resources (FHIR) states “observations shall use LOINC code…” We are relying on the definition of the profile to provide the slice. The bottom expression is saying give me observation BP and then I can treat it like a BP and it behaves and manifests in Clinical Quality Language (CQL) using the slicing. As part of the CQL to ELM translation, the top expression is the output. The underlying Expression Logical Model (ELM) runs against the base FHIR. Therefore, systems do not change what they are doing on the implementation side. This is just an authoring simplification.

Q: Where can I find the `using` statement that references Quality Improvement Core (QI-Core) 4.0.0?

A: It is located on the Clinical Quality Framework Github repository in the Clinical Quality Language (CQL)-to-Expression Logical Model (ELM) translator. Quality Improvement Core (QI-Core) is one of the models that are in line as resources in that translator.

Q: Is the Quality Information and Clinical Knowledge (QUICK) data model Extensible Markup Language (XML) using Fast Healthcare Interoperability Resources (FHIR) Helpers?

A: The QUICK 1.6 version of FHIR was manually built and is not using any type of FHIR Helpers. It is still being developed to secure a more automatic process. There is also QUICK 3.0 XML and 3.0.0 XML which are in various stages of completion, but not ready for use and are not using FHIR helpers.

Q: Does the Quality Improvement Core (QI-Core) model info 4.0.0 Extensible Markup Language (XML) use Fast Healthcare Interoperability Resources (FHIR) Helpers?

A: It does not. QI-Core maps are in the model info. Example: the CQL to ELM translator is outputting gender.value, but the authoring environment is just gender.

Q: Which model should measure developers use while coding within Fast Healthcare Interoperability Resources (FHIR)?

A: The Quality Improvement Core (QI-Core) 4.0.0 model is the correct one at this point. Testing is still being done to cover all of the use cases.

Q: Over the last year, there have been changes to CQL. Is the version of CQL that was used to build the Expression Logical Model (ELM) available?

A: Yes, it is included in the latest ELM output version of the CQL-to-ELM translator.

Q: Once Quality Improvement Core (QI-Core) becomes stable, will HL7 publish an update to include the new model information and replace the Quality Information and Clinical Knowledge (QUICK) model tab?

A: Yes, this could be an update since it is not changing QI-Core. At the moment, an update is still in the exploration phase. The team is trying to provide simple and consistent approaches as well as making sure it can be maintained and implemented based on the authoring model. The CQL to ELM translator v1.4.9 snapshot supports using QI-Core. The translator is being used in the Atom plugin and the Measure Authoring Tool (MAT)-on-Fast Healthcare Interoperability Resources (FHIR) to make sure there is consistent usage of Clinical Quality Language (CQL) across the board

Q: In regard to Slices in Quality Improvement Core (QI-Core), where do you find the naming to define a new QI-Core profile more directly? How do you define values that you need to put in? For example, how is the "observation-bp" defined and applied?

```
define TestSlices:
     [Observation: "Blood Pressure"] BP
       let
         SystolicBP: singleton from (BP.component C where C.code ~ "Systolic blood
   pressure"),
         DiastolicBP: singleton from (BP.component C where C.code ~ "Diastolic blood
   pressure")
       where SystolicBP.value < 140 'mm[Hg]'
         and (DiastolicBP.value < 90 'mm[Hg]'
```

```
define TestSlices:
     ["observation-bp"] BP
       where BP.SystolicBP.value < 140 'mm[Hg]'
         and BP.DiastolicBP.value < 90 'mm[Hg]'
```

A: That is a good question. Navigate to the latest published QI-Core Implementation Guide (IG), go to Section 1.2 Contents and click on Profiles. This will take you to the QI-Core Profiles page. If the profile has QI-Core written in front of it, just use the name (e.g., AdverseEvent) as in the example:

> 2 QI-Core Profiles
> The table lists the QI-Core profiles that are part of the IG, from which USCore profile they are derived, if any, and the underlying Fast Healthcare Interoperability Resources (FHIR) resources:

QI-Core Profile	USCore Profile	Base Resource
QICoreAdverseEvent		AdverseEvent

However, if the element you are defining is pulled in from an underlying source and does not have QI-Core written in front of it, you would select the element (e.g., FHIR Vital Signs) from the QI-Core Profile table, as in the example below:

2 QI-Core Profiles
The table lists the QI-Core profiles that are part of the IG, from which USCore profile they are derived, if any, and the underlying FHIR resources:

QI-Core Profile	USCore Profile	Base Resource
	FHIR Vital Signs	Observation

This will take you to the element's profile page. Scroll to and select the profile name of choice (e.g., Blood pressure systolic and diastolic). This will bring you to the Content tab of the StructureDefinition page. Click on the Differential Table tab to locate the ID of the profile (e.g., observation-vitalsigns).

Side note: As Bryn was answering this question; he realized that he did not have all of the QICore 4.0.0 elements displayed on the StructureDefintion pages. He will post all of the definitions on the Clinical Quality Language Test QICore github page for users. He will submit this as a tracker to surface the name of the profile. The list of elements he will display and track are:

```
define TestAdverseEvent: ["AdverseEvent"]
      define TestAllergyIntolerance: ["AllergyIntolerance"]
      define TestBodyStructure: ["BodyStructure"]
      define TestCarePlan: ["CarePlan"]
      define TestCareTeam: ["CareTeam"]
      define TestClaim: ["Claim"]
      define TestCommunication: ["Communication"]
      define TestCommunicationNotDone: ["CommunicationNotDone"]
      define TestCommunicationRequest: ["CommunicationRequest"]
      define TestCondition: ["Condition"]
      define TestCoverage: ["Coverage"]
      define TestDevice: ["Device"]
      define TestDeviceNotRequested: ["DeviceNotRequested"]
      define TestDeviceRequest: ["DeviceRequest"]
      define TestDeviceUseStatement: ["DeviceUseStatement"]
      define TestDiagnosticReport: ["DiagnosticReportLab"]
      define TestDiagnosticReportNote: ["DiagnosticReportNote"]
      define TestEncounter: ["Encounter"]
      define TestFamilyMemberHistory: ["FamilyMemberHistory"]
      define TestFlag: ["Flag"]
      define TestGoal: ["Goal"]
      define TestImagingStudy: ["ImagingStudy"]
      define TestImmunization: ["Immunization"]
      define TestImmunizationEvaluation: ["ImmunizationEvaluation"]
      define TestImmunizationNotDone: ["ImmunizationNotDone"]
      define TestImmunizationRecommendation: ["ImmunizationRecommendation"]
      define TestImplantableDevice: ["USCoreImplantableDeviceProfile"]
      define TestLaboratoryResult: ["USCoreLaboratoryResultObservationProfile"]
      define TestLocation: ["Location"]
      define TestMedication: ["Medication"]
      define TestMedicationAdministration: ["MedicationAdministration"]
      define TestMedicationAdministrationNotDone: ["MedicationAdministrationNotDone"]
      define TestMedicationDispense: ["MedicationDispense"]
      define TestMedicationNotDispensed: ["MedicationDispenseNotDone"]
      define TestMedicationNotRequested: ["MedicationNotRequested"]
      define TestMedicationRequest: ["MedicationRequest"]
      define TestMedicationStatement: ["MedicationStatement"]
      define TestObservation: ["Observation"]
      define TestObservationNotDone: ["ObservationNotDone"]
      define TestOrganization: ["Organization"]
      define TestPatient: ["Patient"]
      define TestVitalsPanel: ["observation-vitalspanel"]
      define TestRespRate: ["observation-resprate"]
      define TestHeartRate: ["observation-heartrate"]
      define TestOxygenSat: ["observation-oxygensat"]
      define TestBodyTemp: ["observation-bodytemp"]
      define TestBodyHeight: ["observation-bodyheight"]
      define TestHeadCircum: ["observation-headcircum"]
      define TestBodyWeight: ["observation-bodyweight"]
      define TestBMI: ["observation-bmi"]
      define TestBP: ["observation-bp"]
      define TestSmokingStatus: ["USCoreSmokingStatusProfile"]
      define TestPulseOximetry: ["USCorePulseOximetryProfile"]
      define TestPediatricBMIForAge: ["USCorePediatricBMIforAgeObservationProfile"]
      define TestPediatricWeightForHeight:
      ["USCorePediatricWeightForHeightObservationProfile"]
      define TestPractitioner: ["Practitioner"]
      define TestPractitionerRole: ["PractitionerRole"]
      define TestProcedure: ["Procedure"]
      define TestProcedureNotDone: ["ProcedureNotDone"]
      define TestRelatedPerson: ["RelatedPerson"]
      define TestServiceNotRequested: ["ServiceNotRequested"]
      define TestServiceRequest: ["ServiceRequest"]
      define TestSpecimen: ["Specimen"]
      define TestSubstsance: ["Substance"]
      define TestTask: ["Task"]
```