Transcript – Clinical Quality Language (CQL) Basics Video Short

(Excerpt from 2019 EtoE webinar broadcast)

Published October 2020

Alright, so let's go ahead and get started with some of the CQL Basics to set the foundation.

So this diagram is one of the ways we represent the evolution of the standards when creating the specifications for eCQMs. There are three components to the eCQM. First you have the metadata. The HQMF or Health Quality Measure Format is the basic electronic specification for the measure, in other words, the metadata and the population structure.

Second, you have the data model, the QDM, which is the data model used for 2018 and 2019 reporting.

Third is the logic. We use the QDM for the logic in 2018 reporting, However, as of January 1, 2019, we transition the logic from QDM to CQL and the transition to CQL has been a big undertaking, but it does have its benefits.

So, from a technical standpoint, CQL simplifies the logic, making it easier for the system to compute. But, more importantly, from a clinical perspective, CQL allows for more flexibility in the logic so it has better timing precision so that the logic can be better aligned with the clinical intent of the measure.

This is a screenshot of the human readable file, and for those that are new to eCQMs, a Human Readable is the file format of the measure specification that a person can read.

The first thing I want to point out is if you look under the initial population, it's only referring to a line TJC.encounter with principle diagnosis and age. And so, this is a major difference from the QDM version, where you had lines and lines of logic stated. With CQL, the logic is simplified into what we call definitions. And that makes it easier to read.

And we title these definitions using more of a natural language to capture the meaning of the logic it represents and then use these definitions to build each of the population criteria.

The second thing to note is the definitions and functions sections. These are bracketed together and these act like building blocks, which I'll talk about a bit more in the next slide. But, collectively, these are all the definitions used to build the population criteria here at the top of the page.

So, to give you a better understanding of the CQL structure and a CQL definition, this diagram depicts a visual representation of a very simple construct for the initial population.

So if you imagine with me for a moment that you have a set of LEGOs, and if you've never played with them before, you're given a book of directions, which instructs you on how to connect several blocks together in order to build some sort of structure.

Well, we can take that same concept into CQL, where we have the clinical specifications as directions to build definitions. And those definitions are like LEGOs or building blocks, where we use one definition to build upon another definition until you have a completed structure.

So, looking at the diagram, the definition we're building is the initial population, which is defined as C.

Now, remember, each block then represents a definition and each definition has a name and logic to represent a population parameter.

So, in Block A, this is looking for a specific encounter type, for example, an inpatient or an ED encounter.

You'll also want to note that A is the largest block to represent the largest patient population.

So, the idea is, as you start building our population criteria, we want to start narrowing down the patient population.

So, now in block B rather than repeating encounter logic again, we can simply pull in definition A into the logic and then add next criteria we are looking for: in this case, a diagnosis.

This is how we connect the definitions together, by nesting one into the other.

So then definition B now includes the encounter type and the diagnosis.

Now, to complete the structure, we need an age block, so to speak. So we follow that same pattern where we connect B with age, to create, C.

Hopefully, this helps to show how we build off of a definition, so that each successive definition helps to further constrain the logic.

So now in C, we have all the criteria we defined: the encounter type, the diagnosis, and the age of the patient. And that's how C becomes the initial population.

Looking at this concept in more concrete terms, here in STK-2, we have three definitions to build the initial population. To start we have the TJC.Non Elective Inpatient encounter. Notice how we pull in the definition title, non elective inpatient encounter and then add diagnosis criteria for hemorrhagic and ischemic stroke.

Then, in the next block up, encounter with principle diagnosis and age, again, we pull in all stroke encounter into the logic and add age criteria.

So, then, in the population criteria, we use encounter with principle diagnosis and age directly as the initial population. And as you can see in real terms, how complex a definition can be if we were to try to squeeze all this logic into one statement.

Now, there are a lot more terms used in CQL, but these are just some of the basics.

So, as I mentioned, CQL uses definitions, and all definitions have a name. And that name should encompass the meet, or the meaning, of the definition and should be unique, so we can identify it within a library.

Libraries are created at the measure-level. So each measure builds its own library of definitions as they are written. But we also have global libraries that can be shared across all measures, which is the purpose of creating a library, so that we can easily share definitions across measures, rather than restating each time.

So, to identify if the definition is coming from a library, we use a library alias.

If you look at the yellow highlight, a library alias is denoted as a prefix in the name of a definition, although some definitions may not come from a specific library, you will not see that denotation.

Next, we have an expression. And an expression refers to the contents of the definition, which we use quite a bit throughout the logic.

Looking at the basic constructs of an expression. Expressions use data types. Data types describe a part of the clinical care process, which refers to a specific category in QDM.

For instance, the encounter performed data type used in this example is in the encounter category. A medication administered data type would belong to a medication category. Each data type has their own set of attributes, and attributes provide specific details about their data types.

So, in the example, dot.relevant period is an attribute of the encounter performed data type, meaning the relevant period is used to define a start and end time for the encounter.

Next, we have a value set, which defines, or specifies the kinds of data that we're looking for, or codes, relative to the QDM category. Ultimately, these are the codes and the logic we are looking for within the EHR.

So in this example, the value set of non-elective inpatient encounters consist of a SNOMED code. To satisfy that logic. We would expect to see that SNOMED code somewhere in the patient's record.

Lastly, we have aliases. Aliases and expressions are used to give a source a name so that the object can be referred to easily within the expression to avoid restating. We try to be very concise and meaningful so that it makes sense to the reader. So you'll be seeing this often throughout the measure specifications.

Just briefly, there is a Terminology section in the Human Readable, where you can see all the value sets and direct reference codes used in the measure. The Value Set Authority Center is where you would go to verify the codes listed in each value set, and we use a unique object identifier, which is the quickest way to locate the correct value set.

One of the most common questions we get is, “What does union mean?”

Union is a CQL Operator, and it's used to combine two or more lists together. So, looking at the diagram on the left, any element in list A, or list B will satisfy the condition. So, anything in red. And just to clarify what lists are: lists are the result of what the logic is looking for in the EHR.

So, again, in the example, the logic is combining a list of all diagnosis codes that represent each one of the diagnoses listed. And what union does, is it combines these lists so that if the patient encounter has any of the diagnosis codes from this list, it will satisfy the condition.

Another common operator in CQL is intersect, which is only looking for the common elements between List A and list B.

So in the diagram, the red depicts the shared elements between the two lists. To add some context, let's look at the example.

List A returns a list of all inpatient encounters with age 18 years and older. And List B returns a list of all encounters with a CBC level. Then the result of this intersection is a list of all inpatient encounters, aged 18 years and older, and a CBC level.