June 8, 2016 Cooking with CQL – All questions under General CQL – **SESSION 3**

Q: How does CQL simplify more than other traditional query languages?
A: CQL has dedicated constructs for the clinical domain, native support for intervals, and high-level phrasing around temporal relationships. This allows for as close to natural language expressions as possible without ambiguity.

Q: How does CQL compare to DROOLS and is there any sample available for CQL to DRL (drools) conversion?
A: DROOLS is a production rule language, whereas CQL is a functional language designed for expressing queries. Translators have been written to translate CQL to DROOLS, but they are not open source. Tooling exists to aid the translation of CQL to any target format in both the Java and .NET platforms. There are no publicly available examples. However, a pilot in the Clinical Quality Framework initiative built a translator from CQL to Drools.

Q: What is a library? Does it import something from another measure?
A: A CQL library is text document (.cql, .xml, and .json) that contains CQL expressions, definitions, functions and other declarations that can be used across measures. Each eCQM contains a primary library that defines the criteria used by the populations of the measure. The HQMF document references this library and defines the populations by identifying which expressions in the CQL library define each population (e.g., Inpatient Encounter, terminologies, etc.). Measures may include references to a shared library, which can be either referenced throughout a single measure or across different measures (and even clinical decision support rules) to share definitions and functions like "Hospitalization". This library sharing minimizes efforts both with measure implementation and development.

For more information on libraries, refer to the Using Libraries to Shared Logic section of the CQL Implementation Guide
(http://www.hl7.org/implement/standards/product_brief.cfm?product_id=400)

Q: What is the driving benefit of CQL in eCQMs and does it justify extensive vendor resources to re-write parsing tools?
A: CQL implementation is designed to address many of the issues encountered with previous eCQL specifications (using QDM-logic). It allows for increased precision (reduced ambiguities) and improved specificity, provides the ability to share between decision support rule and quality measures, and improves the ease of building language processing applications. CQL-based eCQMs are intended to provide as much opportunity for automated interpretation as possible. QDM-based logic measures have often been manually translated for implementation, which results in ongoing costs as measure specifications are updated annually. While programmatic implementation and interpretation of CQL-based eCQMs has an initial cost, that cost is then significantly reduced for ongoing maintenance, as systems will only have to ingest any updated specifications.

Learn more about the Benefits of CQL here
(https://ecqi.healthit.gov/system/files/Benefits_of_CQL_Updated_1.15.2018_508.pdf)

Q: In the multi-source example shown in slide 23, what is the result type?
The query in question is:
define "Emergency Department Encounters": ["Encounter, Performed": "Emergency Department Visit"] ED with "Inpatient Encounters" E
such that ED.dischargeDateTime 1 hour or less before E.admissionDateTime
A: The “with” and “without” operations in CQL do not change the result type of the query, so it remains "Encounter, Performed". These operators only filter the results based on the presence or absence of a related record (using the “such that” criteria to determine existence). They are equivalent to “semijoin” and “semiminus” in a relational query language.

Q: What's the purpose of the library02 here? Is this something you're creating for reuse in other measures?
A: Measures may include references to a shared library, which can be either referenced throughout a single measure or across different measures (and even clinical decision support rules) to share definitions and functions like "Hospitalization". This library sharing minimizes efforts both with measure implementation and development.

For more information on libraries, refer to the Using Libraries to Shared Logic section of the CQL Implementation Guide
(http://www.hl7.org/implement/standards/product_brief.cfm?product_id=400)

There are also at least two open source engine implementations of CQL that evaluate the measures and supporting logic.

JavaScript Open Source Engine:
https://github.com/cqframework/cql-execution

Java Open Source Engine:
https://github.com/dbcg/cql_engine

Q: Where can I find more detailed description on the temporal operators used by CQL?
A: The Clinical Quality Language Specification, available from HL7, has a complete description of all the operations and functions available in CQL. For the temporal operators specifically, section 2.5.5 of the Clinical Quality Language Specification provides a discussion of the available operators, and sections 9.8 and 9.9 of the reference define detailed semantics.
Link: http://www.hl7.org/implement/standards/product_brief.cfm?product_id=400

Q: Is the CQL Author's guide part of the HL7 documentation? Or where can this be accessed?

A: Yes, it's part of the CQL specification available on the HL7 site
Link: http://www.hl7.org/implement/standards/product_brief.cfm?product_id=400

Q: Was the goal of CQL to encourage modularity for re-usability?
A: Reusability was part of the goal, but also for ease in implementation. CQL's use of libraries enables sharing of logic expressions, definitions, functions and other declarations between different measures and clinical decision support rules, and provides the same language for data from different models.

Q: Do you see CQL implementation impacting how vendors write their code to extract the data elements?
A: It depends on the degree to which vendor's existing code evaluates logic. In QDM, the data elements and the logic to write queries with them are specified together, making a clean separation between those two functions difficult, and preventing them from evolving at different rates. A primary design goal of CQL is to enable that separation at the specification level, and the reason for the approach taken of moving only the logic portion to CQL while retaining the QDM data model representation was to allow vendors to pivot their architectures as well. If a given architecture already drew a line between the data access layer and the logic evaluation, that architecture will be less impacted by this change than one that implemented the logic and retrieval together.