CQL 2.0

# Use Cases for CQL 2.0

## Associated Object Retrieval

From: Imaging, TBPT, many others

Overview: Users need to be able to return a target object with one or more of its associations populated. Potentially also the association’s associations populated as well.

Example: A researcher would like to retrieve all Genes with a symbol like ‘BRCA%’ and then get the chromosome number affected. Currently, the researcher would have to perform two queries: The first to get all Genes with symbol like ‘BRCA%’, then another to get all Chromosomes who have an association to a Gene with one of the IDs of the genes from the first query. It’s cumbersome, slow, and causes unneeded network traffic. Additionally, such a two-step query is impossible (does not conform to the domain model) if the association from Gene to Chromosome is not bi-directional.

## Temporal Queries

From: TBPT

Overview: Expression of query attributes in terms of ‘age’.

Example: A researcher would like to find all Samples in a data service that are 30 days old or newer. While one could formulate such a query presently by doing something like “current day – 30”, the query cannot be saved for later re-use.

## Aggregate Queries

From: TBPT, few others

Overview: Return things like min, max, average values

Example: A researcher would like to know the max white blood cell count from a set of blood samples defined by some query.

## Strong Data Types in Query attributes

From: Everybody

Overview: CQL Attribute has a ‘value’ attribute, which is presently just a string. Users don’t know how to format dates, times, etc.

Example: Overview says it all. Adding this feature removes an ambiguity and isolates CQL formatting from the backend data source’s expected formatting.

## Complex Attribute Types

From: TBPT

Overview: Ability to query over ‘attributes’ which are actually a conglomeration of attributes.

Example: An Identifier field may be of the format “ssn:firstName:lastName” or similar. A researcher wants all Patients with identifier.firstName = “Foo”.

# Solutions

## Associated Object Retrieval

Thoughts: This is very common request, so it deserves high priority. I propose adding an optional element to the root of a CQL query (like Query Modifiers are now) to specify which associations are to be returned. The association population should be configurable to either populate ALL associations up to a certain number of levels, or a named association (or multiple associations). In the case of named associations, the query developer should be able to specify the names of sub-associations to populate as well, in a recursive fashion.

Proposal: The schema AssociationPopulationSpec.xsd fulfills the requirements. It allows for a choice of depth-based population or named association population. The naming is specified recursively, and the depth based population is a simple integer value. The schema presently allows for a flag to be set indicating “infinite” depth population, the usefulness of which needs to be evaluated.

To-do:

I. Evaluate the usefulness and practicality of implementation of the ‘infinite depth’ flag

II. Naming of elements and types in the schema needs some work

## Temporal Queries

Thoughts: In the TBPT case, this might be a modeling issue. Things like “age” should never be stored in a database, since they change as time progresses. From a query perspective, we would need a way to make a query “relative to” some other value. This gets into the area of joins, which CQL doesn’t really do. In this case, the value is relative to today’s date, so it’s a known value and not really a join but a value replacement on the server side.

To-do:

I. Evaluate some TBPT models to see if anybody actually stores “age” values.

II. Develop a specialized query type for temporal queries

A. Might be dependent on the strongly typed queries values request