Schema related points:

1. It's a really good thing that we now have strongly typed values as part of an attribute constraint. I was wondering if it's not possible to have strongly typed values in the gov.nih.nci.cagrid.cql2.results.TargetAttribute

class as well (instead of a String value)?

Maybe. Doing so would make the results document excessively verbose, since this would require unique elements for each potential return type (Integer, String, Boolean, etc).

I think I need to modify typed attribute values in queries, though, since anything other than string won’t work for ‘like’ queries. In CQL 1, if you want values like “10?”, which would match 100-109, you can put that value in a string. There’s no way to do that using a strongly typed integer query value in CQL 2 at the moment. I’ll mess with some variations on the predicates, inheritance, and xsd:choices and see where it takes me.

I wasn’t aware that one could use LIKE for non-string operands in CQL1. I don’t recollect any use-case where this feature is needed. We currently do “x like 10?” as “x >=100 and x<=109” (cf your example above).

Also, just noticed that float/double types aren’t present right now in CQL2; will they be present in the final version?

2. Refer the attached file "handCardSuit\_associationPopulation.xml". The query will return all Card's (and Suit's) of any Hand that contains the Ace of Spades; is that correct? (Note that since I've not executed any CQL2's I could be wrong about some of these points).

- <ns1:CQLQuery xmlns:ns1="http://CQL.caBIG/2/gov.nih.nci.cagrid.cql.Components">

- <ns1:CQLTargetObject className="gov.nih.nci.cacoresdk.domain.other.levelassociation.Hand">

- <ns1:CQLAssociatedObject className="gov.nih.nci.cacoresdk.domain.other.levelassociation.Card" sourceRoleName="cardCollection">

- <ns1:CQLGroup logicalOperation="AND">

- <ns1:CQLAssociatedObject className="gov.nih.nci.cacoresdk.domain.other.levelassociation.Suit" sourceRoleName="suit">

- <ns2:BinaryCQLAttribute name="name" xmlns:ns2="http://CQL.caBIG/2/gov.nih.nci.cagrid.cql.Attribute">

 <ns2:Predicate>EQUAL\_TO</ns2:Predicate>

- <ns2:AttributeValue>

 <ns2:StringValue>Spade</ns2:StringValue>

 </ns2:AttributeValue>

 </ns2:BinaryCQLAttribute>

 </ns1:CQLAssociatedObject>

- <ns3:BinaryCQLAttribute name="Name" xmlns:ns3="http://CQL.caBIG/2/gov.nih.nci.cagrid.cql.Attribute">

 <ns3:Predicate>EQUAL\_TO</ns3:Predicate>

- <ns3:AttributeValue>

 <ns3:StringValue>Ace</ns3:StringValue>

 </ns3:AttributeValue>

 </ns3:BinaryCQLAttribute>

 </ns1:CQLGroup>

 </ns1:CQLAssociatedObject>

 </ns1:CQLTargetObject>

- <ns4:AssociationPopulationSpecification xmlns:ns4="http://CQL.caBIG/2/gov.nih.nci.cagrid.cql.AssociationPopulationSpec">

- <ns4:NamedAssociationList>

- <ns4:NamedAssociation roleName="cardCollection">

 <ns4:NamedAssociation roleName="suit" />

 </ns4:NamedAssociation>

 </ns4:NamedAssociationList>

 </ns4:AssociationPopulationSpecification>

 </ns1:CQLQuery>

Actually, this query will return Hands which have associated Cards which are of the suit ‘Spade’ and have the name ‘Ace’. The returned Hands will have their cardCollection populated, and each card will have its suit populated.

But several user-defined queries require that only the Ace (Card) and

Spade(Suit) is returned; in other words the constraints on Card and Suit should be satisfied by the returned Card's and Suit's.

Example: Assume the following simplified model "Participant -> Specimen"; Specimen has attribute called "type".

Consider the query "Give me all male participants and their associated DNA specimens". Here, the returned specimens must be of type "DNA" and belong to "male" participants. Instead it seems that a corresponding CQL2 can only do the query "Give me all male participants that have an associated DNA specimen, and also return ALL their specimens".

Your assessment is correct.

From CQL2 schema perspective, we should be able to specify whether or not the Attribute-constraint "Specimen.type='DNA'" should be applied to the specimens that will be returned.

I’m not certain HQL can even do this, let alone the difficulty of implementing this functionality in other query languages.

I don’t know much about HQL; but the SQL I had in mind was

select \* from participant p join specimen s on (FK\_cond)

where p.gender=’male’ and s.type = ‘DNA’

Ofcourse, converting the relational results to objects is the difficult part: but I’d thought HQL would provide a way to do this…

Note that simply firing additional CQL2's will not solve this problem. This is because even the query "Give me DNA specimens for Participant with id=5"

can also not be performed.

Client-side processing will be needed to filter out non-DNA specimens.

This is an underlying side effect of the way the model is designed. If a bi-directional association existed between DNA specimen and Participant, this type of query would work. I think I could add an attribute to the Association query element which indicates the association is to be traveled in the “reverse” direction. So long as a role name is defined which can be traversed by the query processor, I think this will work ok.

Yes; all these problems arise only in the context of unidirectional associations. I couldn’t grasp your idea of specifying “reverse”.

A (far) related minor quirk: I prefer the word “roleName” to “sourceRoleName” as part of an Association tag. I find “sourceRoleName” a bit confusing…

3. It seems like if I specify AssociationPopulationSpec, then the results will always be of type CQLObjectResult. Is this correct?

Correct. I should probably make this clearer by using a choice between association population and query modifiers.

4. This is regarding the use-case on the wiki at [http://www.cagrid.org/wiki/Data\_Services:CQL2:Use\_Cases#Returning\_attributes](http://www.cagrid.org/wiki/Data_Services%3ACQL2%3AUse_Cases#Returning_attributes)

\_from\_multiple\_objects

Copied here for reference:

**From:** TBPT

**Overview:** Ability to return an attribute value from two (or more) different objects.

**Example:** A researcher would like all white blood cell counts from blood specimens, as well as dates of hospital visits for a set of patients defined by a query.

**Currently:** CQL does not provide support for such queries since it is strictly a hierarchical query language. Two queries could be issued and the results joined on the client side.

**Solutions:** Since CQL 2 is also a hierarchical language, support for this functionality cannot be directly included in the language. However, the implementation of association population may handle some portion of this use case. Client side tooling could also be developed which provides this functionality. DCQL / DCQL 2 may reevaluate this use case.

**Complexities:**

* How would one express a tuple of results from different objects?
* How are those tuples built? Cross join, left join?

I did not understand the points in this section. I thought that the requirement was equivalent to specifying a set of attributes under a NamedAssociation...

Note that is related to point 3 above: what should the results be if we allow NamedAttribute[] in a NamedAssociation?

It would have to be a new type (say CQLNonObjectResult) that is something like

class CQLNonObjectResult {

 TargetAttribute[] values;

 CQLNonObjectAssocResult[] assoc;

}

class CQLNonObjectAssocResult extends CQLNonObjectResult { // maybe composition is better?

 String roleName; //equivalent to roleName in NamedAssociation }

And results will be of type CQLNonObjectResult if any of the NamedAssociation's in the CQL2 contain a NamedAttribute.

This use case was presented as a way to support what amounts to the following HQL:

Select a.foo, b.bar from A as a and B as b where ….

Ah, I think I understand now; this was about classes that aren’t associated!

Since CQL and CQL 2 are hierarchical query languages, this kind of expression can’t exist. I mentioned association population might solve some subset of this use case, which would be true if B were associated to A somehow and was populated. Then one could simply retrieve the attributes needed by walking the object graph.

The CQL 2 results schema already handles returning things other than objects:

<xsd:complexType name="CQLQueryResults">

 <xsd:annotation>

 <xsd:documentation>Results from a CQL query executed against a caGrid data service</xsd:documentation>

 </xsd:annotation>

 <xsd:choice>

 <xsd:sequence>

 <xsd:element name="ObjectResult" type="res:CQLObjectResult" minOccurs="0" maxOccurs="unbounded"/>

 </xsd:sequence>

 <xsd:sequence>

 <xsd:element name="AttributeResult" type="res:CQLAttributeResult" minOccurs="0" maxOccurs="unbounded"/>

 </xsd:sequence>

 <xsd:sequence>

 <xsd:element name="AggregationResult" type="res:CQLAggregateResult" minOccurs="0" maxOccurs="1"/>

 </xsd:sequence>

 </xsd:choice>

 <xsd:attribute name="targetClassname" type="xsd:string" use="required"/>

 </xsd:complexType>

Attribute results get returned in an array of AttributeResult elements, aggregations get returned in their own type as well.

My imaginary CQLNonObjectResult above provides attribute-value pairs for both the target object and any Associated objects as well (as specified in NamedAssocation’s (imaginary) NamedAttribute[]). But, currently, AttributeResult only contains values of attributes of the target object, right?

----------------------------------------------------------------------------

-------------------------------------------

API related points:

1. It would be helpful to have utility a method that, in case of object results, parses the result xml and returns a map with

 key as attributeName/roleName and

 value as attributeValue/(nested map for associated object).

Note that this map is effectively the CQLNonObjectResult class described above.

(Of course, this parser can only used when the xml is created using the default caGrid serializer.)

I think the AttributeResult type fits this. I intend to implement tools similar to the existing CQLQueryResultsIterator in caGrid for CQL 2 once it’s incorporated into the main caGrid release, which will give you a nice object API and Iterator implementations for working with results.

Again, I think AttributeResult only contains values of attributes of one (target) class; the above map intends to hold values of attributes of associated classes as well. Am I missing something here?

2. We will still be able to use the generic DataServiceClient (or an

equivalent) which will now be able to fire CQL2's, right?

Yes. I think the plan is for both CQL and the CQL 2 APIs to live side by side for a time before dropping support for CQL, but there’ll be a standard data service client which can communicate with any CQL 2 data service, just like we currently have one for CQL 1 data services.