EML - Bug #544

issues about storageType and attributeDomain

07/05/2002 11:59 AM - Dan Higgins

Status:	Resolved	**Start date:**	07/05/2002
Priority:	Normal	**Due date:**	
Assignee:	Chad Berkley	**% Done:**	0%
Category:	eml - general bugs	**Estimated time:**	0.00 hour
Target version:	EML2.0.0rc1	**Spent time:**	0.00 hour
Bugzilla-Id:	544		

Description

The beta9 version of the attribute module has an element called "storageType". As I understand it, the preferred use of this attribute is to contain the XMLSchema datatype of the attribute (e.g. 'string'). The attribute module also has a subtree named 'attributeDomain' with the three branches 'enumeratedDomain', 'textDomain', and 'numericDomain'.
It seems to me that the "storageType" and "attributeDomain" elements are logically related, but that relation is not indicated in the attribute module.
As an example, consider a storageType of 'string'. With XMLSchema datatypes, the concept of a datatype is limited using "facets". Thus a string can be further restricted using (for example) 'enumeration', 'maxLength, or 'pattern' constraining facets. Similarly, 'totalDigits' or other facets can be used to contrain a "decimal" datatype.
In the 'attribute' module of eml, however, such contraints are put into the 'attributeDomain' subtree. The 'enumeratedDomain' subelement does have the ability to enter code values and the associated definition (a code/definition facet is NOT available in XMLSchema datatypes), but the 'enumeratedDomain' subelement does NOT have a simple enumeration where one just lists allowed values for an attribute.

```
In summary, I would suggest that the enumeratedDomain element should have a
simple 'enumeration' child with the ability to just list allowed values (and not
require definitions), and/or we should combine the 'storageType' and
'attributeDomain' elements into something like the structure used with XMLSchema
datatypes and contraining 'facets'/
```

History

#1 - 09/02/2002 10:29 AM - Matt Jones

Thanks for the comments on these data typing issues, Dan. There are two distinct issues you raised, which I will address separately:

1) Enumerated domain doesn't allow a simple list without definitions

```
This is true, and intentional.  When data are distributed, it is critical to
   know the definitions for the string values that are present in the data
   entity. String values or enumerated lists are generally codes that represent
   some type of measurement (e.g., HIGH, MEDIUM, LOW), or are names of
   sampling locations (e.g., SUBPLOT4).
   In either case, it is critical to have the definition.  From a data re-use
   or data preservation perspective, can you show a case where it would be
   acceptable to not have a definition of an enumerated value?  If so, I would
   agree that we should consider relaxing this requirement, but for now I think
   it is a fundamental part of the definition of an enumerated attribute.
```

2) XML Schema data types used in storageType overlap with attributeDomain

```
Also true, but the two fields serve different purposes.
   The storageType of an attribute is an indication of the type that might be
   used to represent the value in a data management system, such as
   a database or programming language.  It is not actually an
   expression of the true domain, as it may in fact be defined slightly
   differently than the attributeDomain (e.g., storageType might be "character"
   while the domain might be a restricted list of character values).
```

```
That we recommend XML Schema Datatypes (which allow restrictions) for the
   storageType does not change the need for an independent specification of the
   domain.  If someone were to use a different type system for the storageType,
   especially one which didn't have the restriction capabilities that XML Schema
   Datatypes does, then the elimination of attributeDomain would be problematic.
   So, basically, attributeDomain is a required expression of the domain, while
   storageType is an optional expression of the likely type from some
   (hopefully common) type system (e.g., Oracle datatypes, Java datatypes,
   XML Schema data types). One might think of storageType as a hint to
   automated processing systems as to how one might represent the values of
   the attribute.  storageType was originally repeatable, and one might
   argue that it should be repeatable so that the type from multiple systems
   can be indicated.  I think that would be a positive change.
```

In summary, although you make cogent points, I don't think that we should make substantial changes to the model at this time. I will, however, revise the schemas to try to clarify the documentation with respect to these issues, and to make storageType repeatable. Comments? In the absence of further comments, I'll close this bug this week. Thanks.

#2 - 09/05/2002 04:34 PM - Matt Jones

Note from conference call:

some people think unit and attributeDomain should be optional, or that we need a good default value if they are required. For example, a default value for "unit" would be "undefined", and a default value for the attributeDomain would be a textDomain that matches a pattern of ".*". Need to figure out what the heck "dimensionless" is in STMML and how it relates to comment fields and stuff like that that aren't really data. I prefer the latter (required but with sensible defaults). Need a decision on this. Hopefully during tomorrow's conference call.

#3 - 09/09/2002 02:05 AM - Matt Jones

OK, we reached a decision on the conference call. Create a unit "undefined" that is the default for the "unit" field, and keep unit required. Keep attribute domain required, but make sure that; it also has a default of any string as the domain (basically, a textDomain with patter ".*"). I'll need to look into how to implement these decisions.

#4 - 09/11/2002 02:57 PM - Matt Jones

Chad -- can you handle these fixes for this bug? You're pretty familiar with the issues and the proposed solution, and I'm getting slammed before my trip. It has a better chance of getting done on your plate than on mine. That ok? Just need to define the "undefined" unit and the default attributeDomain (".*").

#5 - 09/11/2002 03:27 PM - Chad Berkley

made the default of unit "undefined". made attributeDomain optional but I put in the docs that if the element is omitted, the domain of the attribute is assumed to be the regular expression (.*).

#6 - 03/27/2013 02:14 PM - Redmine Admin

Original Bugzilla ID was 544