fhiso

CFPS 19
(Call for Papers Submission number 19)

Proposal to Create a Standard Run-time Object Model

Submitted by:	Proctor, Tony
Type:	An area requiring standardisation
Created:	2013-02-25
Last updated:	2013-04-20
URL:	Most recent version: http://fhiso.org/files/cfp/cfps19.pdf This version: http://fhiso.org/files/cfp/cfps19_v1-1.pdf
Description:	Proposal to have a standard object model to support scripting and query languages.
Keywords:	Object-model, Scripting, Queries

Contents

Abstract

The standard Data Model will prescribe the structure of genealogical data, but not its indexed representation, e.g. database schemas. There are good cases where people or software will need to manipulate or query genealogical objects within a given dataset. This proposal is to create a related standard Object Model.

1. Goal

There are two main goals to this proposal.

1.1 Scripting Languages

A number of people have discussed the provision of libraries for manipulating GEDCOM datasets. The References section cites one such example that advocated the use of AWK to manipulate GEDCOM files directly. There are multiple fallacies associated with this approach:

- AWK is a record-by-record text-processing language. This makes it useless for manipulating genealogical entities that are defined by blocks rather than lines, or which have links to other entities such as Persons, Places, and Events.
- Processing a dataset via its text is fraught with problems in ensuring that you process the syntax exactly, and do not accidentally recognise false syntax in comments, notes, or names.
- The AWK code would be specific to the GEDCOM serialisation format (i.e. file format). If, for instance, GEDCOM and GedXML were equivalent representations of a common model then the AWK code could not be used for both formats.

The cited thread also refers to using SQL Stored Procedures but this has just as many fallacies:

- The SQL code is tied to a specific SQL schema, and is of no use for other SQL schemas or non-SQL representations.
- The processing of dates, and multiple date calendars, cannot be adequately handled by SQL operations. Genealogical dates also have granularity and uncertainty to deal with.
- The processing of personal names cannot adequately be handled by SQL operations. A person may have multiple names, and these may be time-dependent. Processing must take account of case-blind, accent-blind, and composed/decomposed Unicode forms.
- The processing of place references cannot adequately be handled by SQL operations. Place references may have time-dependent hierarchies, and the name-matching shares issues with personal names.

What is needed is a generic Object Model that can reliably manipulate compiled genealogical entities, i.e. objects. This may be used to achieve custom manipulations within the context of a commercial product, or independently of any such product through a scripting language.

1.2 Query Languages

This Object Model should also be applicable to query languages. A well-known problem when handling massive amounts of data is how to handle queries efficiently. A client program may know what it wants to select but it cannot simply pull all the data up to the local machine in order to execute a selection algorithm on it. Languages like SQL and MDX push query expressions down to the data server so that they can be executed remotely, where the data resides, and thus reduce the amount of data transmitted back to the local machine. Stored Procedures are a way of putting more complex, custom code into the database that can be executed during a query. As we've just explained, though, SQL operations are inappropriate for generic genealogical queries as they're dealing with a specific physical schema.

2. Outline Proposal

We should define a set of Classes corresponding to the genealogical entities in the standard Data Model. These should include standard methods, each with standard semantics, and these should correspond with the defined processing of entities. For instance:

- Matching a date, an approximate date, or a date range.
- Handling different world calendars.
- Matching a personal name against any of the ones defined for a person.
- Matching a place reference against a hierarchical place authority.

The Object Model could also support methods for generating all serialisation formats defined by the Data Model. This also means that a section of script could even achieve a format conversion.

3. Not Covered or Not Required

An Object Model can be defined without prescribing a specific language syntax. In principle, many different language syntaxes could be used to make such a model accessible.

While Inheritance (in the context of OOP, or Object Oriented Programming) is widely regarded as a positive feature, a fundamental and explicit dependence on it could prove to be divisive and reduce the advantages of a common Object Model. Even where the concept exists, it could be either “interface inheritance” or “implementation inheritance” which are widely different.

4. Illustration

A simple illustration involves a requirement to look at all the events in a timeline, then look at all the people sharing those same events, and then to select just the ones whose name(s) have the element "Jesson" in them.

This example uses a java-like syntax.

```
Person me = New Person("Tony Proctor", 1956);
for (Event e: me.allEvents()) {
    for (Person other: e.allPersons()) {
        if (other.nameContains("Jesson")) {
            ...do something with this other person...
        }
    }
}
```

This example uses a VB6-like syntax.

```
Dim me As New Person
me.setPersonName (“Tony Proctor”)
me.setDateOfBirth (1956)
Dim e As Event
Dim other As Person
For Each e In me.allEvents()
    For Each other in e.allPersons()
        If (other.nameContains(“Jesson”)) Then
            ...do something with this other person...
        End If
    Next other
Next e
```

The syntax is different but the essential elements of the Object Model, such as class names and method names, are the same in the two examples.

5. Use Cases

Scripting is required for expressing queries that maybe outside of the scope of ones preferred product, or for automating processing tasks related to ones genealogical data.

Within queries (including some type of genealogical stored procedure), the same object model may be used to support complex queries devised by a client product. For instance, some type of validation

While we should not prescribe the actually language syntax, we must ensure that there is a standard Object Model that is specific to our Data Model, and independent of the schema or serialisation format currently in use.

6. References

Thread on **soc.genealogy.computing** discussing a related requirement through direct processing of the text in a data file:
https://groups.google.com/d/topic/soc.genealogy.computing/_DzCQ_lt4Zo/discussion.