

MageBuilder: A Schema Translation Tool for Generating MAGE-ML from Tabular Microarray Data

William Martin <eco_bill@yahoo.com> and Robert M. Horton <rmhorton@attotron.com>
Attotron Biosensor Corporation

Abstract

A 'MageBuilder' object takes a set of 'MageMap' objects and a set of data streams as input, and produces a MAGEstk object representation, which is then serialized as MAGE-ML. A 'MageMap' object encapsulates the rules of how data records from an input stream relate to one MAGE object. Each input "stream" is an anonymous subroutine that supplies records whose fields represent columns in the input table. The input tables can be delimited text files, database queries, or essentially any source that can be coerced into a set of records with fixed fields.

1. Introduction

The MicroArray Gene Expression Markup Language (MAGE-ML [1]) is a standardized XML format for representing the results of microarray experiments. It is designed to include sufficient structural meta-information to make this data useful for purposes such as meta-analysis and data mining. The MAGE data model is sufficiently complex that specialized software tools, such as the MAGE software toolkit (MAGEstk), are invaluable for dealing with the structured data. Laboratories conducting microarray experiments typically store results in tabular format, either in spreadsheets or relational databases, and conversion of this tabular data to MAGE format may be daunting. Here we present tools to facilitate mapping from table-based schemas to MAGE-ML.

2. MageBuilder

MageBuilder [2] is a tool for constructing MAGEstk objects from various data sources. It is the engine that processes input record streams using MageMap objects to produce a MAGE-stk object structure. Although fairly flexible, this approach imposes a framework that requires the problem to be broken down into manageable sized, usually

simple pieces. These pieces are MageMaps, input record streams, and relationships between the maps.

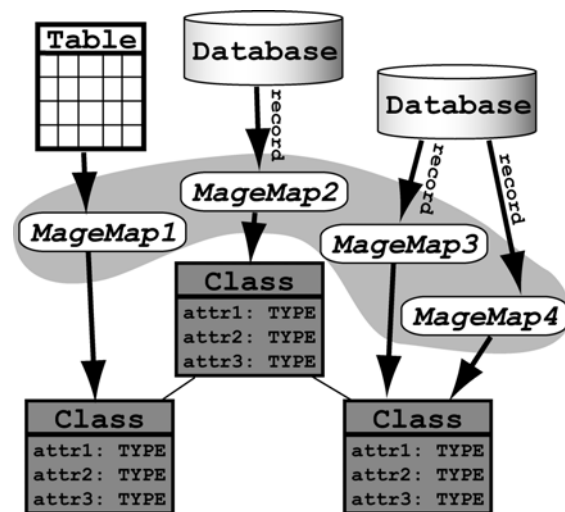


Figure 1. Microarray data stored in tabular format are mapped to a collection of MAGEstk objects. Input data may be delimited text from a spreadsheet or records from a database query. MageMaps convert a stream of records into corresponding objects; multiple streams can contribute to a single object.

A MageMap allows you to specify rules for how to determine values for attributes and associations for a particular class of MAGE objects based upon input records. Mapping rules may be as simple as copying a column to an attribute, while custom subroutines may be required to calculate attribute values for more complex mappings.

The mapping rules are output-centric, in that each map specifies creation of one output object, rather than the full set of rules for processing an input record. Although a map's perspective is limited to one input record, multiple maps can be applied to

one input stream, each describing how to create a different class of MAGE object. Multiple maps can be used for the same MAGE class using multiple input record streams.

Mapping Rules

A MageMap consists of rules specifying how to determine values for attributes and associations for a particular class of MAGE objects based upon input records. The MageMap rule for an attribute can be specified as:

- 1) The name of a column in the input record to be copied verbatim into an attribute
- 2) A string to be evaluated at run time (using Perl's 'eval' function); it could contain references to columns in the input record with `$INPUT->{column_name}` or to the counter with `$COUNTER->{MAGE_class}` or even the MAGE class name itself as `$MAGE_CLASS`.
- 3) An anonymous subroutine to calculate a value, given the input record. These subroutines can be closures, or they can reference their own package variables to maintain their own (static) context for richer functionality.

The MageMap rule for an association can be specified as:

- 1) Any of the ways an attribute rule can be specified; in this case the calculated string value is used as a key to implicitly find corresponding objects.
- 2) A MageMap object, in which case the MageMap will be applied to the current input record and the results will be assigned to this association.
- 3) A MAGE object, in which case it will simply be assigned to this association.
- 4) 'undef', in which case the engine will look for an association previously set up by another map using a pseudo-attribute.

A "pseudo-attribute" is a MageMap rule whose key (in lieu of an attribute name) is the name of an association from the perspective of another MAGE object. An index is created for the foreign object to find the mage objects produced by this MageMap. The MageMap rule for a pseudo-attribute looks just like a rule for an attribute. The distinction is that whereas an attribute name is a single word, a pseudo-attribute name is of the form "foreign_MAGE_class.foreign_association". The

key difference is in how the rule is used. Rather than assigning the calculated value to the current MAGE object, the value is used as the key to an index containing a reference to the current MAGE object. The calculated value for a pseudo-attribute is expected to be the 'identifier' for a foreign MAGE class that needs to reference this object in one of its associations.

To support generation of unique IDs on the fly, there is an automatic counter mechanism, one counter for each MAGE class. These counters can be referenced by the rules for the map. After processing each input record, each counter (that was used in the processing of that record) is incremented automatically.

Input Streams

An input record is a reference to a hash or pseudo-hash with a predefined set of keys (which are column names). The values are of course corresponding column values.

The input record stream is either a reference to an array of input records, or an anonymous subroutine that returns an input record with each call to it (and 'undef' at the end of the data).

Conclusion

Combining abstract input streams with flexible mapping of streams to objects provides an extensively customizable approach to generating MAGE-ML from sets of data tables.

References

1. Microarray Gene Expression Data Society: www.mged.org
2. Source code: cybertory.org/software/MageBuilder

Acknowledgements

Funded by NIH grant #R44 RR13645 02A2 to Attotron Biosensor Corporation. Inspired by the script "createmageml.pl" of Eric Deutsch <edeutsch@systemsbiology.org>, which is distributed with the MAGEstk.