Data Management Toolkit

Introduction

Whether via the SWEET or via ETL, data in eagle-i is always produced in conformance to the eagle-i ontology that is current at the time of entering the data. It is not uncommon, however, for the ontology to evolve in such a way that some data triples become unnecessary or incorrect. In addition to data migration, it is sometimes necessary to perform low-level data manipulations that are not currently supported by the SWEET UI -- for example, bulk modification of property values according to a regular expression substitution.

The data management toolkit provides system administrators with powerful tools to modify data at the triple level. It requires understanding of RDF and familiarity with the eagle-i ontology.

The toolkit is comprised of a series of Java utilities that are invoked from the command line, and is packaged as a jar file that must be present in the environment where the commands are executed. Only users with administrator privileges can issue data management commands; they should be always tested before applying in a production environment, as they bypass regular data validity controls and may result in non-conformant data.

Data migration as a result of ontology changes

Upon ontology changes, the eagle-i ontology team produces three migration tables that are packaged with the ontology release; they contain the changes that were made to the ontology since the previous ontology release, grouped by data element where the changed occurred: ontology classes, ontology predicates and ontology individuals.

An ontology release is in turn packaged with a general eagle-i software release. As part of the software upgrade procedure, the eagle-i node system administrator must run the `data-migration.sh` script, which results in the necessary data changes being performed to reflect the new ontology. Internally, this script invokes various data management commands according to the requirements of the particular ontology release (typically only `ChangeObject` and `ChangePredicate` are invoked). Hence, in the general case, a system administrator need not worry about executing individual commands.

If a system administrator upgrades to a non-contiguous release (e.g. from 1.7MS1 to 1.7MS4), applying the regular data migration procedure would miss some of the necessary data changes. The eagle-i team can produce a zip file with cumulative data migration tables (currently upon request). To apply the cumulative tables, the directory where this zip is expanded needs to be provided as a parameter to the data migration script, for example:

```
sh ${REPO_HOME}/etc/data-migration.sh adminUser adminPassword https://your.host.edu migration-tables-since-1.7
MS1
```

Data management commands

Data management commands are found in the java package:

org.eaglei.datatools.datamanagement.command

The Javadocs linked above provide more details on each command.

From a terminal window, commands may be invoked as:

```
java -cp ./eagle-i-datamanagement.jar org.eaglei.datatools.datamanagement.command.[Command] [parameters]
```

Issue the command without parameters to print a help message.

A few parameters are common to all commands:

-c admin credentials, in the form username:password

-r repository base URL, e.g. https://your.host.edu

-t optional URI of an RDF:type restriction, i.e. only apply changes to resources where RDF:type is the provided URI

Considerations

The commands in the data management toolkit make use of the `repository/graph` service to delete and upload entire RDF graphs obtained via `construct` queries. Since the changes don't go through the regular CRUD interface, they do not result in metadata changes for the resource (e.g last modified date or contributor). Some commands add a comment explaining what was done (prepended with the name OTTO KURATOR, so it is easy to SPARQL them), but most commands don't. A detailed log of changes is produced with each run, which contains the old triples (deleted) and the new triples (added). If a command was applied erroneously, it is sometimes possible to reverse it by looking at the logs, reintroducing the old triples and deleting the new triples. If the command is based on a translation table, it is also possible to re-apply it with a reversed translation table.

All commands are idempotent - they may be applied repeatedly without consequences.

Finally, because most commands do not update resource metadata, the changes will not be reported by `repository/harvest`, and will therefore not be visible in the search API. It is therefore necessary to perform a clean search re-indexing and sparqler synchronization after data management commands are executed. **This is yet another reason for using them sparsely.**