**Migration toolkit**

There are a couple of tasks in the toolkit but not all of them are needed for the migration to be ran. The minimum ones needed need to be executed in the following order:

1. Run\_Generate\_Project\_Directory
2. Run\_Metadata\_Extractor
3. Run\_Sql\_Generator
4. Run\_Row\_Counter
5. Run\_Data\_Migrator

The remaining tasks are:

1. Run\_CSV\_Generator
2. Run\_DDK\_Script\_Generator
3. Run\_Join\_Finder
4. Run\_Migration\_Checker
5. Run\_View\_Checker

Each step does something different but is reliant on some of the features of the previous step, needs some values declared in the migrationtoolkit.properties. I’ll describe some of these features bellow.

1. For helping you understand what sections of the toolkit I’m talking about I’m going to suggest you check out the foundation mantel of the migration toolkit:
2. <http://subversion.nsc/repos/MigrationToolkit/trunk/>
3. Do a svn checkout on the trunk and then open the project is NetBeans. Right click on the project root and click tasks>dist>dist
4. Click on the files tab open

migrationtoolkit[root]>dist>migrationtoolkit-1.0.0

1. Everything needs to be set in this directory and I will make references to this bellow

**First step**

You need to create a project directory like in Datanovata which is where the migration takes place.

Personally, I feel like it would be best to create a configuration manager at this step. Before we can run any other task, we need to set up the main features of the MigrationToolkit.properties.

If you open template>migrationtoolkit.properties and build.properties, this is where all the information is set for the toolkit.

Personally, I would like to move this outside of the dist directory in case we need to make any changes to the toolkit in the future. The migrationtoolkit doesn’t look directly in the migrationtoolkit. properties file it looks as the build.properties which directs it too the migration one. Whenever you perform a dist on the toolkit it these two files are reset back to normal so by moving them outside of the dist it will not be lost.

It is very tedious process filling in this file so for the configuration manger I think it would be best if we set the majority of the properties at this step.

List of the properties that need to be set at this point:

* outputDirectory=
* sourceDatabaseDriver=
* sourceDatabaseUrl=
* sourceDatabaseUser=
* sourceDatabasePassword=
* targetDatabaseDriver=
* targetDatabaseName=
* targetDatabaseUrl=
* targetDatabaseUser=
* targetDatabasePassword=
* sourceDatabaseDriverType=
* sourceDatabaseType=
* targetDatabaseDriverType=
* targetDatabaseType=
* maxThreads=
* datanovataTables=

The final step should run Run\_Generate\_Project\_Directory.

**Run\_Metadata\_Extractor**

Requirements to run:

* Have ran and tested the connection to the databases
* The following fields exist in the Outputs folder
  + Metadata\_report.txt
  + Metadata.xml
* A selected schema

The configuration manager for this step should ask the user to select the metadata\_report.txt and metadata.xml document select a schema. The Run\_Metadata\_Extractor can then be ran. I would like to see if it is possible to make a kind of loading bar for the extractor so that you can visually see the progress of this task.

It might be an idea to add a step that checks the connectivity to the source and target database before running the task. This can make use reading the properties file. The step probably isn’t needed but it might be useful to check.

The metadata extractor creates an xml document that is kind of like a description of the source database. The SQL generator and data migrator tasks use this as a quicker access point rather than constantly querying the database. At the end of the task it runs some sql that was created by Paul in the past. It creates 4 tables that are used for tracking the toolkit. These tables are needed for the Run\_Row\_Counter and data\_migrator.

**Run\_Row\_Counter**

Requirements to run:

* Ran and tested the connection to the databases
* The 4 dnv migration tables
* Specified row count folder
* Specified schema

The configuration manager for this step will need to have a folder specified for a report of the row counts. The row counts are read from the source database and sql is constructed in the background and ran against the target database. It populated the columns with values for each table within the database schemas selected.

It again might be a good idea to test the connections to the databases. It doesn’t need to be as a complex check as the ones used in DataNovata just a quick screen that will test the connections to the databases.

**Run\_Sql\_Generator**

Requirements:

* Connection to the databases
* The following files need to be specified:
  + Table.sql
  + Unique\_index.sql
  + Index.sql
* A specified schema

The configuration manager needs to specify the specified file locations above. it will also be a good idea to test the connection to the databases. The sql generator creates three sql files that each do there own thing. the table.sql is the one that creates the new migrated tables on the target database which has the same structure as the source table. The unique\_index.sql creates the sql that creates the unique identifiers for the database, these are what are used by DataNovata to generate a search for each table. The index.sql creates a similar query that can increase the read/write speed of the database. **The only one that should be ran pre migration in table.sql** the other two can be ran after. It might be an idea to have a feature that checks if the migration is complete, this would mean that you can’t run the index files if the migration isn’t complete.

I would like to try and make it possible for the UI to interact with the database and run the sql against the target database.

**Run\_Data\_Migrator**

Requirements:

* Connection to the databases
* Specified schema
* A created target database
* Specified number of threads

The data migrator is the actual process of migrating the databases. It makes use of something called threads. Threads are kind of like having its own processor for each thread. So, in the toolkit they are used by each table. So, if you specify 50 threads in the migration configurator it means that 50 tables can be migrated all at the same time.

I would also like to try and have some form of loading bar making use of the one in the metadata extractor if that it possible to do.