|  |  |
| --- | --- |
| Module | rDQ |
| Sub-Module | rDQ Data Model |
| Author | D Twaddell |
| Date | 27th Aug 17 |
| Reviewer |  |
| Review Date |  |
| Review Status |  |
| Review Comments |  |

# Design Summary

## Design Log

Contents

[Design Summary 1](#_Toc491640516)

[Design Log 1](#_Toc491640517)

[Sub-Module Description 1](#_Toc491640518)

[Data Model 2](#_Toc491640519)

[Database tables 2](#_Toc491640520)

[Table: sysParameters – stores information about parameters used in the system 2](#_Toc491640521)

[Table: rdqModule -reference data, list of rDQ modules and sub-modules 2](#_Toc491640522)

[Table: conDataConnections - reference data, list of connection to data sources of interest to the user 2](#_Toc491640523)

[Table: antTableProfile – stores the results of a table profile 3](#_Toc491640524)

[Table: anaAttributeProfile – stores the results of a column profile 3](#_Toc491640525)

[Table: datDataTable – stores the metadata about a table or dataset 3](#_Toc491640526)

[Table: daeDataElement – stores the metadata about a column or attribute 3](#_Toc491640527)

[Table: gloGlossary – The ‘data dictionary’ with definitions of concepts and data elements 4](#_Toc491640528)

[Table: refReferenceTable – Stores all the reference data used in the system. 4](#_Toc491640529)

[Table: refReferenceItem – Stores all the reference data used in the system. 4](#_Toc491640530)

[Table: dqrDataQualityRule – List of data quality rules that are used in the system. 5](#_Toc491640531)

[Table: dqtDataQualityTest – List of data quality rules that are used in the system. 7](#_Toc491640532)

[Table: hryHierarchy – Models hierarchical relationships between things. 7](#_Toc491640533)

[Appendix 1 – rDQ Module List 8](#_Toc491640534)

## Sub-Module Description

**rDQ Data Model –** define and describe the data structures used throughout rDQ. Use this document in conjunction with the data model maintained in mySQL.

The data model will be implemented in SQLite, so that the data quality state can be persisted across sessions. This means that data quality scores can be accumulated over time to view trends. Other data that is persisted includes:

* **System Parameters** – for example, User Name, Organisation, Database Path, and other things it is useful for the system to know about.
* **Glossary of Terms** – the ‘data dictionary’ that explains what all the data actually is
* **Data Connections** – a list of available data connections, locations where data of interest resides. This may be network folders, web locations, databases, XML/RDF feeds, etc.
* **Data tables and data elements** – the system records certain facts about data tables and data elements, for example the table or data element name. Tables and elements can be linked to the glossary, to describe the data. There is also a link from a data element to some data lineage information. rDataQuality may add a module to automatically derive data lineage from R code. But for now this will be a link to a diagram or some other lineage repository.
* **Data table and attribute Profile Results** – the ‘analyser’ module looks at data to determine certain facts about it, e.g. how many rows, what sort of data is in the table. But the user can also record comments about the data, for example a description of what the data is used for. All this makes up the ‘profile’ of data.
* **Reference Data** – the system records any reference data that is either used in data definitions, e.g. a fixed set of values that a data element can take. Or is reference data used in the system, e.g. the set of data quality dimensions. This reference data is used to define and implement data quality rules. Note that some reference data is hierarchical, for example: Country – Region – Continent. A special ‘hierarchy’ table records the relationships in hierarchical reference data, and could be used to record other hierarchical relationships.
* **Data Quality Rules and Tests** – A DQ rule is a template expression of a test that can be applied to any data element or data table (assuming the rule is relevant to the type and use of the data element or table). A test can involve more than one data element / table, but the result is always attributed to one data element /table. A DQ test is a DQ rule that is bound to a particular data element / table. The DQ rule is expressed in in pseudocode, whereas a DQ test is expressed as executable R code. An example of a data element test is to test for the completeness of the value – i.e. a blank value fails the test. An example of a table-level test is to test for the completeness of the table – i.e. missing rows will fail the test.

## Data Model

Add a new relational diagram and define the data structures. Commit the diagram to GitHub.

## Database tables

Here are some started designs. Primary keys are in **red** and foreign keys in **green**. Add these to the mySQL diagram:

### **Table: sysParameters** – stores information about parameters used in the system

|  |  |  |
| --- | --- | --- |
| Attribute | M/O | Definition |
| parId | M | Unique identifier for the parameter. System assigned |
| parDate | M | Date the parameter was created or updated |
| parName | M | Name of the parameters |
| par\_modId | M | The scope of the parameter – links to an rDQ module |
| parDefault | O | Any default value. User can select “Re-apply default value” to overwrite their custom value back to the default |
| parDataType | M | Select from: Integer, Numeric (real), Character, Boolean, Percentage, image (blob) |
| parValue\_integer | O | Parameter value when the parDataType is integer |
| parValue\_numeric | O | Parameter value when the parDataType is numeric |
| parValue\_character | O | Parameter value when the parDataType is character |
| parValue\_boolean | O | Parameter value when the parDataType is boolean |
| parValue\_percentage | O | Parameter value when the parDataType is percentage |
| parValue\_blob | O | Parameter value when the parDataType is blob |

Likely system parameters include:

* **spaUserName** – The name of the system user (will need to change this when we implement multi-user access)
* **spaUserOrganisation** – the organization that the user works for
* **spaUserOrgLogo** – an image representing the user organization – used in reports and in the UI
* **spaFunctionPath** – path to where data derivation functions are stored

### **Table: modModule** -reference data, list of rDQ modules and sub-modules

|  |  |  |
| --- | --- | --- |
| Attribute | M/O | Definition |
| modId | M | Unique identifier for the module. System assigned |
| modDate | M | Date the module details were created or updated |
| modName | M | Name of the module |
| modPurpose | O | Score of the parameters. Global parameters belong to “rDataQuality”. |
| mod\_parent\_modId | O | Module structure is hierarchical, so this is the modId of any parent module. All modules except rDataQuality will have a parent. |

#### Functions:

Note – I am not sure if the data table needs to be passed as the first parameter to these functions. The function always operates on the same database table, so probably no need to pass it. Included since this seems to be R’s preferred parameter structure, but do omit it if it is unnecessary.

modCreate (mod = modModule, modName,modPurpose,modParent){

# check module doesn’t already exist. If it does return error

# Otherwise create the module, setting modDate to system datetime.

# Return the modId

}

modUpdate (mod = modModule, modId, modName,modPurpose,modParent){

# check module exists. If it doesn’t return error

# Otherwise update the module, setting modDate to system datetime.

# If update successful return TRUE else return FALSE.

}

modDelete (mod = modModule, modId, deletechild = FALSE){

# check module exists. If it doesn’t return error

# If deletechild = FALSE then check if child modules exist. If they do then return error “Cannot delete as child modules exist”.

# Otherwise delete the module, setting modDate to system datetime.

# If delete successful return TRUE else return FALSE.

}

modList (mod = modModule, modId, modSearchString){

# If modId passed, then check module exists. If it doesn’t return error

# Otherwise update the module, setting modDate to system datetime.

# If update successful return TRUE else return FALSE.

}

### **Table: conDataConnections** - reference data, list of connection to data sources of interest to the user

|  |  |  |
| --- | --- | --- |
| Attribute | M/O | Definition |
| conId | M | Unique identifier for the connection. System assigned |
| conDate | M | Date the connection details were created or updated |
| conName | M | Name of the data connection |
| conDescription | O | Optional description of the data connection |
| conString | O | Connection string – that enables R to connect. Includes user/password, etc. |

#### Functions:

conCreate (con=conDataConnections, conName,conDescription,conString){

# check connection name doesn’t already exist. If it does return error

# Otherwise create the connection, setting conDate to system datetime.

# Test the conString. It should be obvious from the format of the string if it is a folder, database

# connection string, web URL, etc. If connection not successful print “Connection created but

# connection not successful”. Else print “connection successful”

# return conId

}

conUpdate (con=conDataConnections, conId, conDescription,conString){

# check module exists. If it doesn’t return error

# Otherwise update the connection, setting conDate to system datetime.

# Test the conString. It should be obvious from the format of the string if it is a folder, database

# connection string, web URL, etc. If connection not successful print “Connection created but

# connection not successful”. Else print “connection successful”

# If update successful return TRUE else return FALSE.

}

conDelete (con=conDataConnections, conId){

# check connection exists. If it doesn’t return error

# Otherwise delete the connection, setting conDate to system datetime.

# If delete successful return TRUE else return FALSE.

}

### **Table:** **antTableProfile** – stores the results of a table profile

|  |  |  |
| --- | --- | --- |
| Attribute | M/O | Definition |
| antId | M | Unique identifier for the connection. System assigned |
| antDate | M | Date of the profile operation |
| ant\_tabId | M | Key to the DataTable that was profiled |
| antProfileDescription | O | Optional description of the profile operation |
| antRows | O | Number of rows in the dataset |
| antCols | O | Number of attribute columns in the dataset |

### **Table: anaAttributeProfile** – stores the results of a column profile

|  |  |  |
| --- | --- | --- |
| Attribute | M/O | Definition |
| anaId | M | Unique identifier for the attribute profile. System assigned |
| anaDate | M | Date the attribute was profiled |
| ana\_daeId | M | Key to the data element profiled |
| anaDescription | O | Optional description of the data attribute profile |
| anaDetailReport | O | A parameter string that allows a detailed report of the attribute profile to be run. Expecting to pass this to DataExplorer to get the report |

### **Table: datDataTable** – stores the metadata about a table or dataset

|  |  |  |
| --- | --- | --- |
| Attribute | M/O | Definition |
| datId | M | Unique identifier for the connection. System assigned |
| datDate | M | Date the table was last updated |
| datTableName | M | Name of the dataset profiled |
| datTableDescription | O | Optional description of the dataset profiled |
| dat\_gloId | O | Key to corresponding glossary term that defines the data table. This will be a data concept type glossary term. |

### **Table: daeDataElement** – stores the metadata about a column or attribute

|  |  |  |
| --- | --- | --- |
| Attribute | M/O | Definition |
| daeId | M | Unique identifier for the attribute profile. System assigned |
| daeDate | M | Date the attribute was profiled |
| daeName | M | Optional description of the attribute profiled |
| daeDescription | O | Optional description of the data attribute |
| daeDataType | O | The type of data. Read from metadata or determine by inspecting values |
| dae\_gloId | O | Key to corresponding glossary term that defines the data |
| daeLineage | O | Link to or description of the lineage of the data element. (We may develop a module to derive the lineage directly from R code, and visualize it in the UI.) |
| dae\_rftId | O | Key to the reference data table that describes the valid values for the data element |

### **Table: gloGlossary** – The ‘data dictionary’ with definitions of concepts and data elements

|  |  |  |
| --- | --- | --- |
| Attribute | M/O | Definition |
| gloId | M | Unique identifier for the glossary term. System assigned |
| gloDate | M | Date the term was updated |
| gloTermType | M | Type of glossary term. Can be one of: **Data Element** or **Data Concept**. Data Elements all have an association to one-and-only-one Data Concept. |
| gloName | M | Term name -the name everyone uses to refer to the concept or data element |
| gloDataUse | M(DE) | Refers to a code that describes what the data is used for. Read from system reference table Data Type. |
| gloDefinition | O | The unambiguous definition of the term. |
| gloDefSource | O | A link to or description of the source of the definition |
| glo\_rft\_validvalues | O | Key to the reference table that describes the single set of valid values for this term. Where there is more than one set of valid values, this is described on the daeDataElement table. |
| glo\_AlternateNames | O | List of alternative names used to refer to the main term. E.g. synonyms |
| glo\_gloConcept | M(DE) | Mandatory for type data element. Refers to the glossary term that is the data concept that ‘own’s the element. |

### **Table: refReferenceTable** – Stores all the reference data used in the system.

|  |  |  |
| --- | --- | --- |
| Attribute | M/O | Definition |
| rftId | M | Unique identifier for the attribute profile. System assigned |
| rftDate | M | Date the attribute was profiled |
| rftTableName | M | Optional description of the attribute profiled |
| rftTableDescription | O | Optional description of the data attribute |
| rftSource | O | Link or description of the source of the reference table |
| rftSystem | M | Boolean flag to indicate if system reference table. System ref tables cannot be deleted. |

System reference tables (*create a function to set the system tables up*)::

1. Data Use
2. Data Quality Dimension

### **Table: refReferenceItem** – Stores all the reference data used in the system.

|  |  |  |
| --- | --- | --- |
| Attribute | M/O | Definition |
| rfiId | M | Unique identifier for the attribute profile. System assigned |
| rfiDate | M | Date the attribute was profiled |
| rfi\_rftId | M | Key to the reference table that ‘owns’ the reference item |
| rfiTableDescription | O | Optional description of the data attribute |
| rfiSystem | M | Boolean flag to indicate if system reference item. System ref items cannot be deleted. |

System Reference Items (*create a function to set the system values up*):

1. Datetime (Data Use)
2. Commentary (Data Use)
3. Amount (Data Use)
4. Classifier (Data Use)
5. Identifier (Data Use)
6. Completeness (DQ Dimension)
7. Validity (DQ Dimension)
8. Accuracy (DQ Dimension)
9. Consistency (DQ Dimension)
10. Integrity (DQ Dimension)
11. Timeliness (DQ Dimension)

### **Table: dqrDataQualityRule** – List of data quality rules that are used in the system.

|  |  |  |
| --- | --- | --- |
| Attribute | M/O | Definition |
| dqrId | M | Unique identifier for the data quality rule |
| dqrDate | M | Data the rule was updated |
| dqrRequirement | M | The business expression of the data quality rule |
| dqrTest | M | The R expression of the data quality rule. Contains markers for data table, data element and reference data references. These markers are replaced by actual references when the rule is used to create a test. |
| dqrSystem |  | Boolean flag to indicate if system DQ rule. System rules cannot be deleted. |

System data quality rules (*create a function to set the system rules up*):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| DQ Rule | DQ Dimension | Element/ Table | Rule expression | Notes |
| Truth | Accuracy | E | The **<data element value>** is asserted as a true value, either by using the **<corresponding primary key>** to reconcile the **<data element value>** using a **<transformation formula>** to a **<guaranteed accurate table>** or by **<manual assertion>** following manual tests. | The simplest <transformation formula> is simply a test that the value is the same. But if a transformation is involved then this needs to be taken into account when reconciling the value. |
| Data Value | Completeness | E | If no **<conditional statement>** exists, or if a **<conditional statement>** evaluates TRUE, then check that the **<data element value>** is not null or blank. | The conditional statement allows for the rule to be applied only in certain circumstances. For example, it might not apply in certain countries, or on certain days of the week, etc.. |
| Specificity | Completeness | E | Check that the <data element value> does not contain a non-specific value | e.g. "Dummy", "99/99/9999", "NOT AVAILABLE", etc. If necessary, establish values to test for by profiling the data. |
| Reconciliation | Consistency | E | The **<data element value>** corresponds to an <expected value> derived by using the **<corresponding primary key>** to reconcile the **<data element value>** using a **<transformation formula>** to a **<trusted table>**. | The <trusted table> will usually have checks and controls around it, and may feed data to many processes. |
| Foreign Key | Integrity | E | The **<data element value>** must refer to one and only one row in the **<foreign table>** to which it refers. |  |
| Primary Key | Integrity | E | The **<data element value>** must refer to one and only one row in the **<table>** in which it is an attribute |  |
| Format | Validity | E | Check that the **<data element value>** corresponds to a particular **<format mask>** | Define the mask using R syntax |
| Range | Validity | E | Check that the **<data element value>** is higher than or equal to the **<low value>** and lower than or equal to the **<high value>**. |  |
| Reference value | Validity | E | Use the **<data element value>** to read the **<corresponding reference table>**. If a **<corresponding row exists>** then the test passes. Otherwise it fails. | Select from reference tables, or create a new reference table |
| Dataset | Completeness | T | The dataset contains all the data it should, based on rules about *what* data should be included. The scope of data is usually expressed in terms of other data elements, and can be checked by referring to other data tables. | For example*, the dataset should contain all individual human customers, and exclude customers that are organisations*. Or, *the dataset should contain all US data, and not from any other country*. |
| Dataset | Timeliness | T | The dataset contains all the data it should, based on rules about *when* data should be available. | Can be difficult to test. But, for example, all bank customers should be screening with 24 hours of opening an account. Therefore the table containing customers to be screened should contain all new customers opening an account within 24 hours |

#### Data Quality Rule/test Notes

A note on the order of applying tests. Normally the table-level tests are applied first, then the element-level. The completeness (data value) test is always applied first to a data element, if it is defined. If the completeness test fails then all other data element tests are not tested, because they will of course also fail. If the completeness test passes, then all other tests can be run in any order.

If the data is used as a *classifier*, i.e. it is reference data, then normally there will be a validity(reference) test and no need for any other validity test. This is on the basis that if the value exists in reference data then it can be assumed to be of the correct format, in an acceptable range, and to be specific.

The primary and foreign key tests are usually only applied to data that is used as an *identifier*.

Note that the user can change the association of rule to DQ dimension. For example, could make the specificity test a validity test, instead of completeness. But the table gives the system defaults.

### **Table: dqtDataQualityTest** – List of data quality rules that are used in the system.

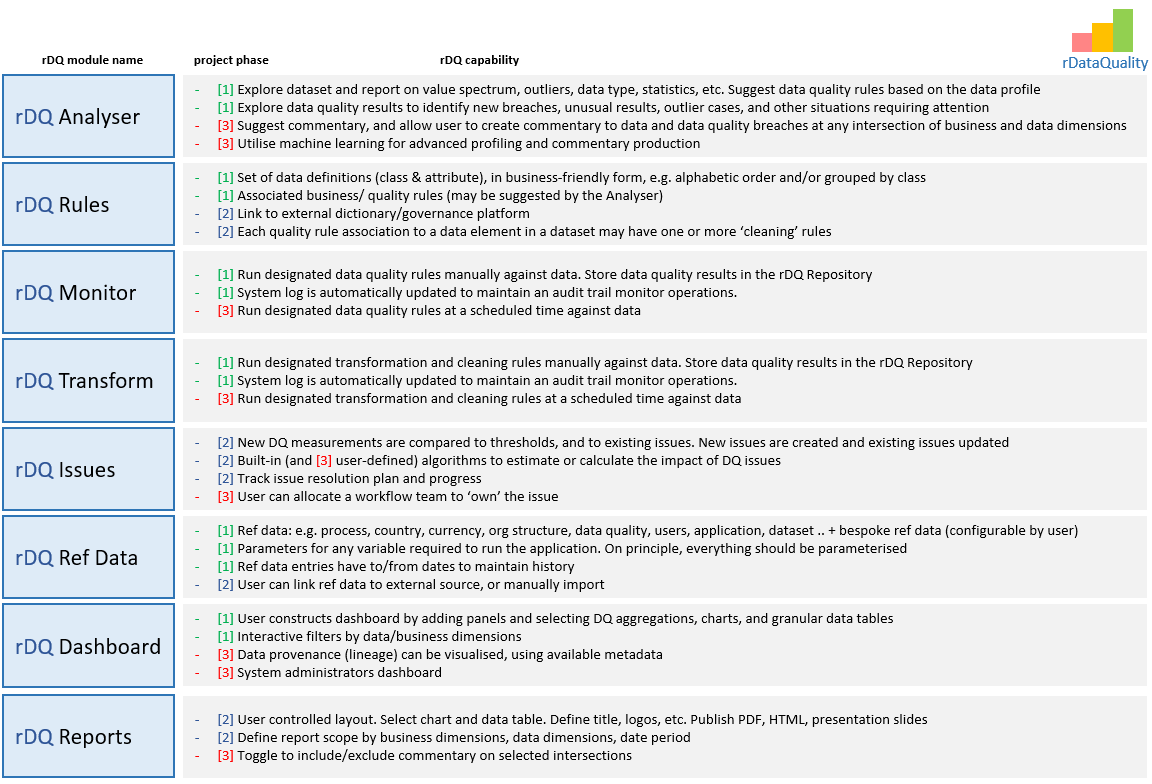
|  |  |  |
| --- | --- | --- |
| Attribute | M/O | Definition |
| dqtId | M | Unique identifier for the data quality test |
| dqtDate | M | Data the test was updated |
| dqt\_dqrId | M | Key to the data quality rule that is applied to the data element |
| dqt\_daeId | M | Key to the data element rule that is being tested |
| dqtTest | M | The R expression that implements the data quality for the data element |

### **Table: hryHierarchy** – Models hierarchical relationships between things.

This is usually reference data. For example, country-region-continent. Hierarchies can be referred to in data quality rules, for example to test that a value is a member of a hierarchy given the presence of another value.

|  |  |  |
| --- | --- | --- |
| Attribute | M/O | Definition |
| hryId | M | Unique identifier for the hierarchy member |
| hyrDate | M | Date the hierarchy member was updated |
| hryScheme | M | Unique name of the hierarchy scheme. For example: **Geographic Area**, or **Legal Entity Structure**. |
| hryLevel | M | Level in the hierarchy: 1 to N |
| hry\_hriID\_parent | M (if Level >1) | Pointer to the parent hierarchy member, where the present member does not exist at the top level. A member can only have one parent in a given hierarchy. |
| hry\_rfiId | M | Points to the corresponding reference item. Note that each item can be part of many hierarchies. |

# Appendix 1 – rDQ Module List



In addition to these ‘public’ modules we may need some private modules to help organise work, e.g.:

* **rDQDataManagement** – common functions relating to data management, dealing with files and database operations. Define functions that are used in other modules. Define global variables
* **rDQUI** – implements the user interface, using Shiny