QUIPU4 use cases

Introduction
QUIPU4 was launched in April this year at the BI & DW Summit. During the launch, the team behind QUIPU4 promised that the new modular approach would enable the quick development of modules with new functionality for use by clients. This blog focuses on two of these modules that were delivered before the summer holidays.

Model-to-model transformation
QUIPU4 is a platform that supports model-to-model transformation. This is how the product sets itself apart from other commonly accepted ETL tools and other data warehouse automation tools. Although such tools usually work well as far as the transformation and manipulation of data at the entity or attribute level is concerned, they lack the support needed to define and develop transformations at the model level. It is at this higher level that patterns can be found and applied, and therefore can be generated. According to QOSQO, this results in a much more efficient way of developing new data solutions quickly.

Use case: Detecting data model changes
The Model Delta Detector is one of our new modules. This model, added back in the spring of 2019, makes it possible to detect and report data model changes.

Figure 1. Model Delta Detector module

All kinds of data models are stored in the QUIPU4 repository. Source models as well as target models generated by QUIPU4. After re-retrieving the metadata from source models (reverse engineering) and generating target models, a new version of these data models is added to the repository. For organisations, it is thus of very real importance to be able to see what changes there are in relation to earlier versions. This simplifies maintenance and administration and can be used, for example, for the timely detection of new structures software suppliers have introduced into their applications. It enables the business to identify more quickly new data that has become available for managing the organisation.

Currently, this module generates an HTML file that reports the differences between two model versions indicated by the user. The output can be read in a web browser. For example, the report generated shows the entities, attributes, keys and relationships that have been removed, changed or added. Figure 2 shows the table of contents of this output and, alongside, a summary of the changes detected. Further on in the output, more details are given regarding the entities, attributes, keys and relationships.

Reference model: Dvdstore (version: 0.1, build number: 1, Local date/time: 2019-08-22T14:51:25+02:00[Europe/Berlin])

Compared model: Dvdstore (version: 0.1, build number: 3, Local date/time: 2019-08-22T15:24:33+02:00[Europe/Berlin])

- Summary
- Entities
- inventory (Added)
- orderline (Added)
- orders (Added)
- customer_history (Added)
- product (Added)
- customer (Added)

Summary

Reference model	Dvdstore
Compared model	Dvdstore
Total number of deltas	79
Total number of Entity deltas	6
Total number of Attribute deltas	43
Total number of EntityConstraint deltas	7
Total number of AttributeConstraint deltas	11
Total number of EntityRelation deltas	6
Total number of AttributeRelation deltas	6

Figure 2a. Table of contents.

Reference model: Dvdstore (version: 0.1, build number: 1, Local date/time: 2019-08-22T14:51:25+02:00[Europe/Berlin])

Compared model: Dvdstore (version: 0.1, build number: 3, Local date/time: 2019-08-22T15:24:33+02:00[Europe/Berlin])

- Summary
- Entities
- inventory (Added)
- orderline (Added)
- orders (Added)
- customer_history (Added)
- product (Added)
- customer (Added)

Entity: orderline

Go to attributes
Go to constraints
Go to relations

Attributes

Attribute name: **product_id**
Delta type: Added

Attribute property	Value in reference model	Value in compared model
Position		3
Data type		bigint
Precision		null
Data scale		null
Default value		null
Position		3
Nullable		false
Description		null

Attribute name: **orders_id**
Delta type: Added

Attribute property	Value in reference model	Value in compared model
Position		2
Data type		bigint
Precision		null

Figure 2b. Sample model change.

An attentive reader immediately understands that this module is also a step towards the automated implementation of detected model changes in the data solutions generated. We call this ‘adaptive design’. With a view to enabling this kind of automated processing of these differences in the future, this module can already generate output in JSON format.

Use case: Accessing data and metadata with OData

OData is a REST-based data access protocol. Among other things, it can be used to retrieve data from web-based relational sources. Thanks to OData, metadata from these relational sources can be retrieved easily. It is then possible to extract the data and store it in a data warehouse, for example.

It goes without saying that this is an increasingly common use case among our clients. More and more applications are web based and the underlying databases can increasingly be accessed via OData. The capacity to read from OData data sources is therefore a high priority for us. The direct

reason came from a client using the TOPdesk online system. That is why, before the summer, we also developed a module for reading and importing OData metadata. This module was released in the latest version of QUIPU4.

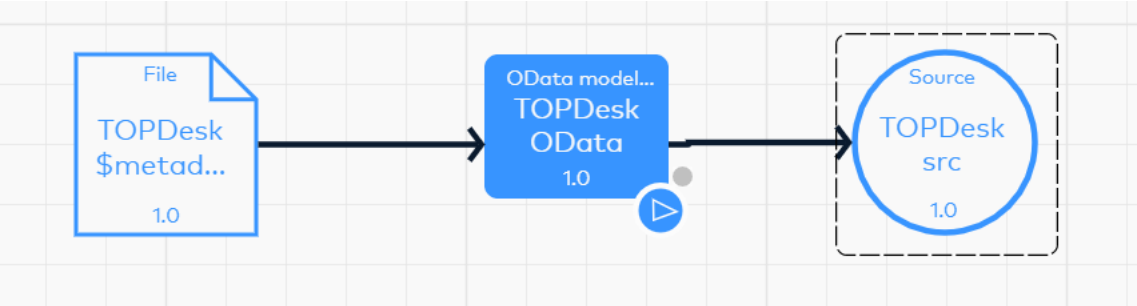

Figure 3. OData data access module

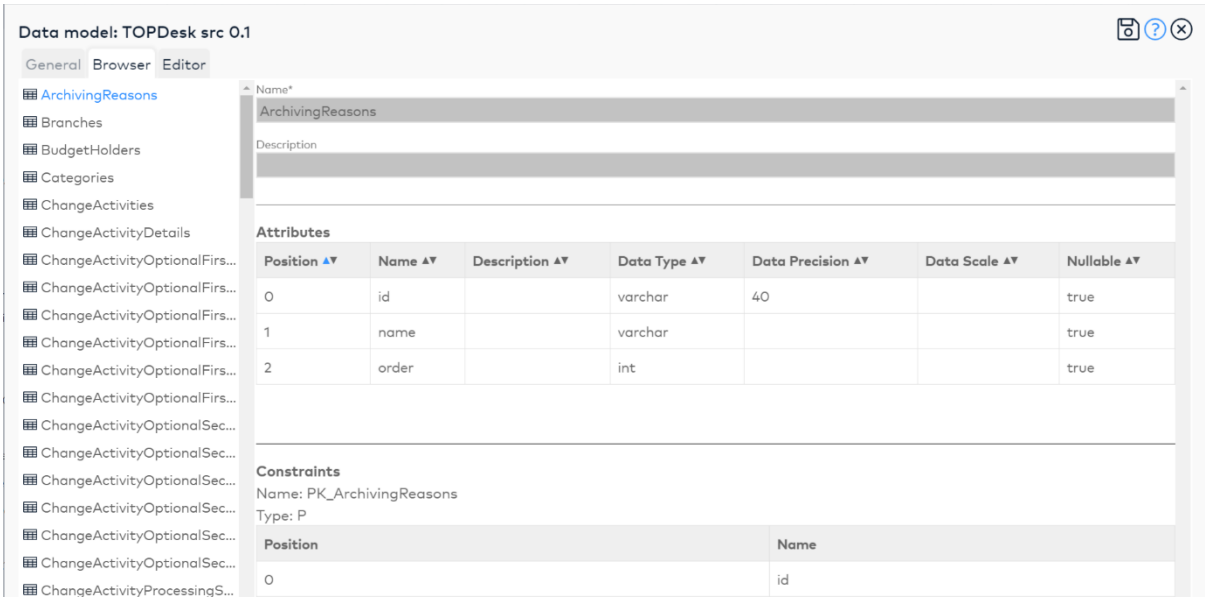

Figure 4. Illustration of the TOPdesk data model as retrieved via the OData module.

For more information, a white paper and a comprehensive manual visit www.quipu.nl. Interested in QUIPU4? You can request a free, one-month trial licence on our website. If desired, we can arrange for a demo.