quipu
data warehouse management

# QUIPU v3: Overview

This document delivers some background information on QUIPU data warehouse generation software developed by QOSQO. It aims at creating understanding of the position of QUIPU in the BI landscape and development cycle.

WHITEPAPER

## TABLE OF CONTENTS

# INTRODUCTION

## QUIPU

| | |
|---|---|
| This Document is not a manual | This document delivers some background information on QUIPU data warehouse generation software developed by QOSQO. It aims at creating understanding of the position of QUIPU in the BI landscape and development cycle. It is *not* a manual, and does not replace release notes. The document may not always be 100% in sync with the latest release of the software, so please refer to the release notes for definitive information. |
| Professional Edition | After the initial release of QUIPU in the Open Source, QUIPU has found its way to its intended users and organizations small and large many of which have reported to benefit from it.<br><br>As of version 3.0 Quipu is released only in a licensed (Professional) version.<br><br>The Professional version 3.1 contains additional functionality:<br>• Extending existing data warehouse models<br>• Generating optimized code for specific database platforms<br><br>Future Professional features will include<br>• modifying the QUIPU generation templates<br>• free exchange of custom templates between paying customers (market place)<br>• support for multiple project teams working on a data warehouse<br>• ..and much more<br><br>We continue to invite the feedback from the user community, as it will be the basis for further development from our side. We may have already recognized the need for certain features, but we would love to be surprised with unexpected use cases that require features currently not present. |
| Introduction videos | In the past year several introduction videos were posted to youtube. You can find them on Channel @Quipu_DWM. (https://www.youtube.com/user/Quipu_DWM). |

## History

| | |
|---|---|
| Data Vault modeling | Consultants of our sister company Nippur as well as from QOSQO have been actively participating in data warehouse and business intelligence projects for many years. They have contributed or managed projects with large multinational companies based in the Netherlands as well as abroad. They have used the Data |

Vault modeling technique for many years now as best practice for modeling data in a way that fits business needs best.

## From best practice to software development

The Data Vault model is particularly strong in ensuring lineage of data warehouse data back to their source systems, thus enabling a complete audit trail. At the same time the Data Vault model is very easy to adapt to changing requirements and extremely scalable. The strong architectural foundation of Data Vault and the simplicity of its design open the way to automation. Our consultants have created quite a few customer specific generator scripts to create and populate a data warehouse.

## Early automation efforts

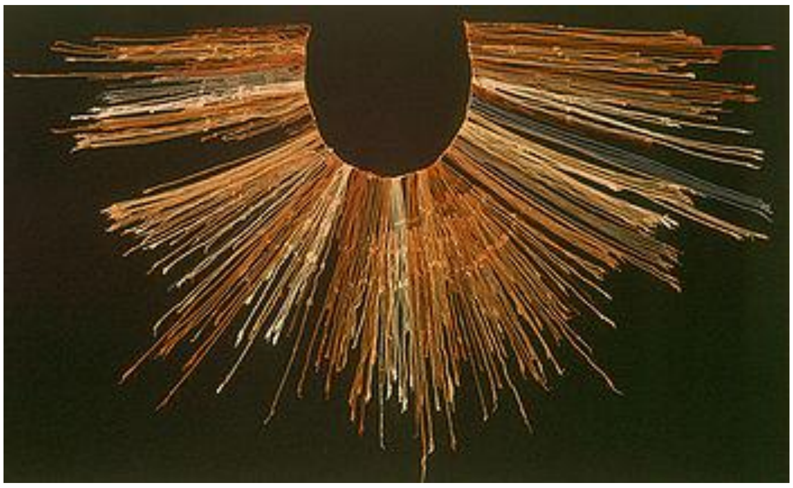
In some cases we created tailor-made fully automated data warehouse management solutions. Complete source and target models, ETL code specifications, scheduling and documentation are managed and maintained using a single repository. The lead architect of QUIPU was involved in the design and realization of one such effort of a large Netherlands based Bank / Insurance company. Currently, QOSQO is responsible for the support of this software.

## Development of QUIPU

This specific solution, combined with our best practices at other customers sparked the idea to develop a data warehouse management software solution back in 2008.

## QUIPU: an ancient Inca recording device

The quipu or khipu (sometimes called talking knots) was a recording device used in the Inca Empire and its predecessor societies in the Andean region. A quipu usually consisted of colored spun and plied thread or strings from llama or alpaca hair. It could also be made of cotton cords. The cords contained numeric and other values encoded by knots in a base ten positional system. Quipus might have just a few or up to 2,000 cords.



Quipucamayocs (Quechua khipu kamayuq, "khipu-authority"), the accountants of Tawantinsuyu, created and deciphered the quipu knots. Quipucamayocs could carry out basic arithmetic operations such as addition, subtraction, multiplication and division. They kept track of mita, a form of taxation. The Quipucamayocs also tracked the type of labor being performed, maintained a record of economic output, and ran a census that counted everyone from infants to "old blind men over 80." The system was also used to keep track of the calendar.

[source and images: wikipedia, http://en.wikipedia.org/wiki/quipu]

## DATA WAREHOUSE ARCHITECTURE

### Starting points

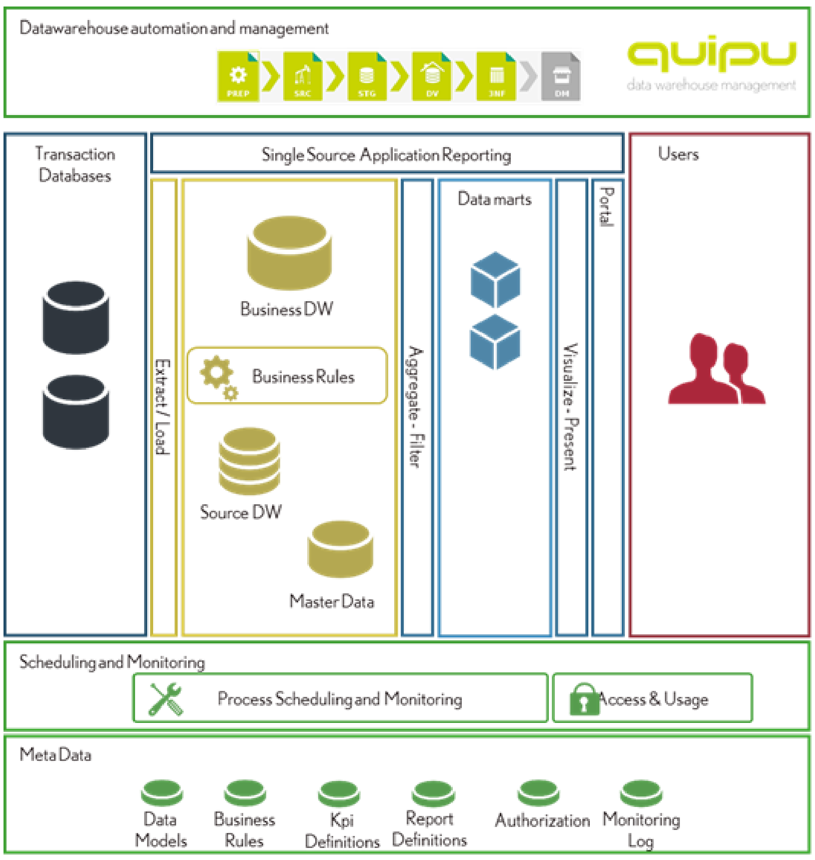
**Layered data warehouse architecture**

A data warehouse (DW) architectures consist of several layers. The first one is the source layer, representing the source systems that contain the data to be used for reporting and analysis (business intelligence) purposes. The second layer is the actual data warehouse where selected (or in rare cases: all) source data is offloaded from source systems and collected in the 'memory' of the organization. Subsets of data are then created to satisfy needs of specific functional areas or domains in the third layer. This layer is called the data mart layer. The fourth layer presents the relevant information to end users, the presentation layer.

**QUIPU generates models and load code**

QUIPU aims at generating the generic core components in the data warehouse layers of the architecture. It is particularly strong in generating, maintaining and populating (database) structures to store historical data, both transactional data and reference data. Apart from the data structures QUIPU also generates the (SQL) code to load the data correctly in the data warehouse. QUIPU is designed to operate in a multitude of environments.

.



At the core of QUIPU lies a data transformation engine that transforms typical relational data models into a data model that is optimized for data warehousing:

the Data Vault. This core engine can be used to transform a source database into a Data Vault, creating what is called a 'source data vault'. But the same engine can also be used to generate a data vault version of an enterprise data model, creating what is called a 'business data vault'. And obviously both usages can be combined to create a more complex, layered data warehouse architecture.

QUIPU contains several switches that allow the data warehouse architect to choose variations on the 'official' data vault specification. An example is the 'Historical Data Archive': a data model using 'technical' keys instead of business keys. This HDA can be implemented in stead of a source data vault. See advanced topics for more details.

**No business rules in QUIPU**

As QUIPU's added value lies with the generic modules, it implies all *solution specific* modules (mainly the data transformations that may be required) will have to be realized outside of the QUIPU product.

In most situations:

- the raw source data is not fit to fulfil the needs of the business. Sometimes complex business rules are required to integrate data from different sources (e.g. match external customer or supplier data to internal data structures).
- hierarchies are added to the source data structures, and complex KPI's are calculated for different levels of the hierarchy.

All of these derivations need to be implemented outside QUIPU, either using ETL tools or BI front-end tools.

## Modeling

**Data Vault - efficient, expandable historical record**

QUIPU is based on the Data Vault model. See http://danlinstedt.com/solutions-2/data-vault-basics/ for more information and background.

The key strengths of this model are:

- it allows capturing an historical record of any data ('transaction' as well as 'master' data)
- it is optimized for (near) real-time updates (allows massive parallel processing, late arriving transactions etc.)
- it isolates changes in the model and is thus flexible (can be adapted to changing business needs with relative ease)

## Basic use of QUIPU

**Business Data Warehouse**

In its most basic form QUIPU can be used to generate a single, integrated data warehouse based on a single, harmonized model. This model, the 'business model', is a single, comprehensive and consistent representation of all entities (with their relations) as recognized by the business users.

This business model does not normally exist in any source system: it must be developed in close cooperation with the business to reflect the terms and definition of the data that the business chooses to work with. It identifies the business keys that identify the various business entities and their inter-relations. It also specifies all relevant attributes and facts related to these business entities that are required for management reporting, (predictive) analysis, etc.

This business model, when provided to QUIPU, will be transformed into a Data Vault that can be completely time-variant: capturing not only business events when they occur in time, but also capturing the changing reference data. Thus the data warehouse based on Data Vault can reproduce the business entities and their relations as they are defined at any point in time.

3NF business data model

The business data model should be in 3NF (third normal form) and can be created using one of many database modeling tools. The business model cannot be imported in QUIPU directly (in the current version). The easiest way to get the business data model into QUIPU is by physically implementing the -empty- database in one of the many RDBMS'es supported by QUIPU and using QUIPU's reverse engineering function.

Once the model is available in QUIPU, the following functions are available:

Generate Staging area

- Generate Staging area. The source data will have to be delivered in a format that is more or less dictated by the business model into a staging area. The tables are generated by QUIPU, as well as template SQL statements that can fill the staging area from the source tables -assuming these tables exist physically in the same RDBMS system-. It is the task of an external ETL tool to load the data in the staging area if the source does not exist in this exact form or lives in a different database platform.

Generate Data Vault model

- Generate Data Vault model. QUIPU sports a number of powerful algorithms that analyze the source model's tables, keys and relations in order to generate a valid data vault model. This model can be further optimized through direct interactive manipulation (or in future versions via export and import functions).

Generate Load functions

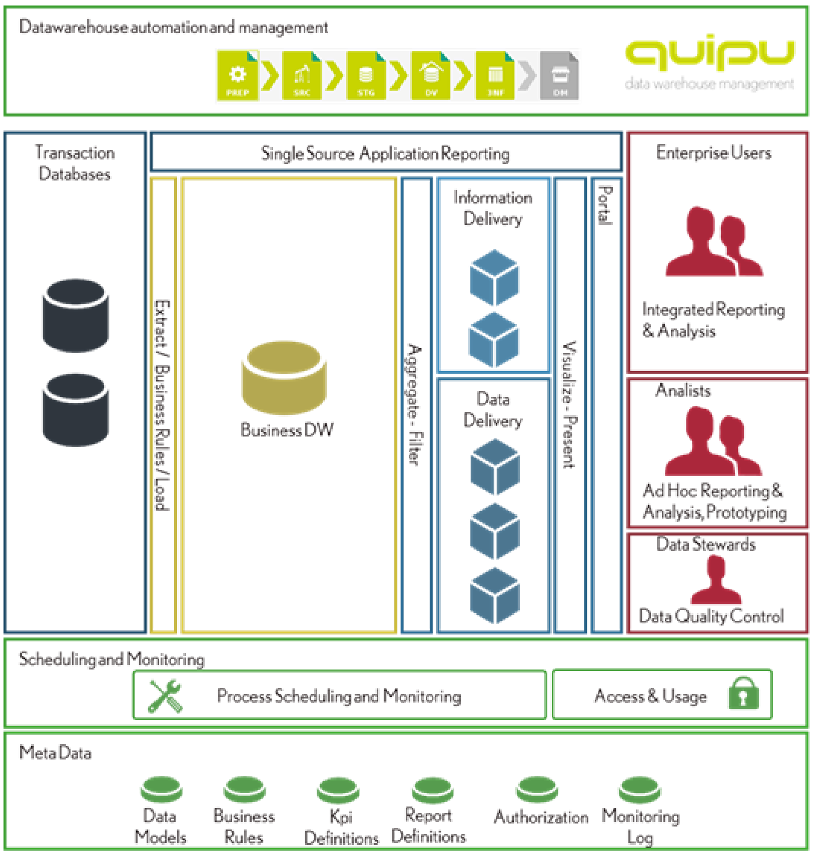
- Generate Load Functions. The data delivered into the staging area are loaded into the Data Vault tables. Extensive validation and logging assures that the data warehouse always retains a valid state. All data elements in a load are marked with a unique load id (or alternatively a source id and timestamp), allowing various housekeeping functions on the data loaded (e.g. archiving data or removing faulty loads). QUIPU also generates code to load the staging area, but this should be regarded as demo code as in practice the staging area is often loaded directly from the source systems. Generation is based on templates that can be adapted to exploit the features of various database management systems.

Generate views

- As a first step towards supporting BI, QUIPU generates views that reproduce the 3NF source model (thus hiding all the complexities of the Data Vault) for query purposes:
    - current state (showing actual data)
    - last known state (actual data including the last known state of deleted data)
    - point-in-time state (same as 'current state, but then for a specified point in -past-time)

The picture below shows the complete architecture.

*NOTE*: Data Marts are required to deliver data to the business. Functionality to assist the construction of Data Marts will be implemented in a future version of QUIPU.



*Business Data Warehouse*

## Enterprise DW Architecture

| | |
|---|---|
| Enterprise Requirements | QUIPU is designed to support the Enterprise Data Warehouse concept. It can certainly be used in less complex environments but is designed with the requirements of larger enterprises in mind. Thus, it is capable of capturing and integrating information simultaneously from multiple sources and scaling to large data volumes. |
| BI stack | QUIPU can integrate with Open Source or commercial products, to offer a complete solution. The solution then includes a modeling tool, ETL and a BI front-end stack. |
| Near-real-time | QUIPU is aware of the shifting role of the DW in today's businesses. More and more the DW is considered a key component in the information architecture, supporting not only the strategic information requirements of senior management, but also the tactical and even operational requirements at the shop floor. The latter require frequent data updates (near real-time). |
| Implement a multi-layered version | With QUIPU it is possible to implement a multi-layered version of the Data Vault. |
| Source Data Warehouse | The 'raw' source data is captured in a 'source Data Warehouse'. These are generated as Data Vaults and are directly derived from the (logical) data models of the sources. They are ideal for operational, single-source reporting and analysis.<br><br>Where business keys can be recognized they can be the first step towards building the 'business Data Warehouse' as they can bridge data from multiple source systems. If business keys cannot be identified easily it is possible to build a source Data Warehouse without the hubs and links: a Historical Data Archive (HDA). |
| Business Data Warehouse | The business needs a unified model to report from: the 'business Data Warehouse'. It is built for integrated, business wide reporting and analysis.<br><br>The Business Data Warehouse complements the data in the Source Data Warehouse with integrated data (through cleansing and mapping algorithms) and derived data (calculated using business rules). It is based on a consistent data model, specifically designed to support the business.<br><br>In most cases the integration requires mapping of business keys and transformation of the source data. Although it would be possible to distribute the transformed data directly into data marts, this is often not desirable:<br><br>• if the transformed data is considered business data in its own right: even when the transformation rules change, the data needs to remain available as it was historically derived (transformed).<br>• if transformations need to be implemented only once in order to safe-guard consistency. It can be difficult to enforce that the same business rule is correctly applied when building multiple data marts. It is often much easier to achieve consistency when the result is stored in a second Data Warehouse layer: the Business Data Warehouse. |

The business Data Warehouse is in its essence a logical layer: it is a complete model (describing *all* business entities) but some entities may simply refer to (be views on) the underlying source Data Warehouse layer. In most cases only a small subset of the entities in the business Data Warehouse model need to be implemented physically.

## Data Mart layer

QUIPU typically creates more tables than the original 'point-in-time' 3NF source models. As such it creates a model quite opposite of a star-schema and not particularly strong in query- performance on the dominant RDBMS systems (Oracle, MS SQLServer, DB2, MySQL, etc.).

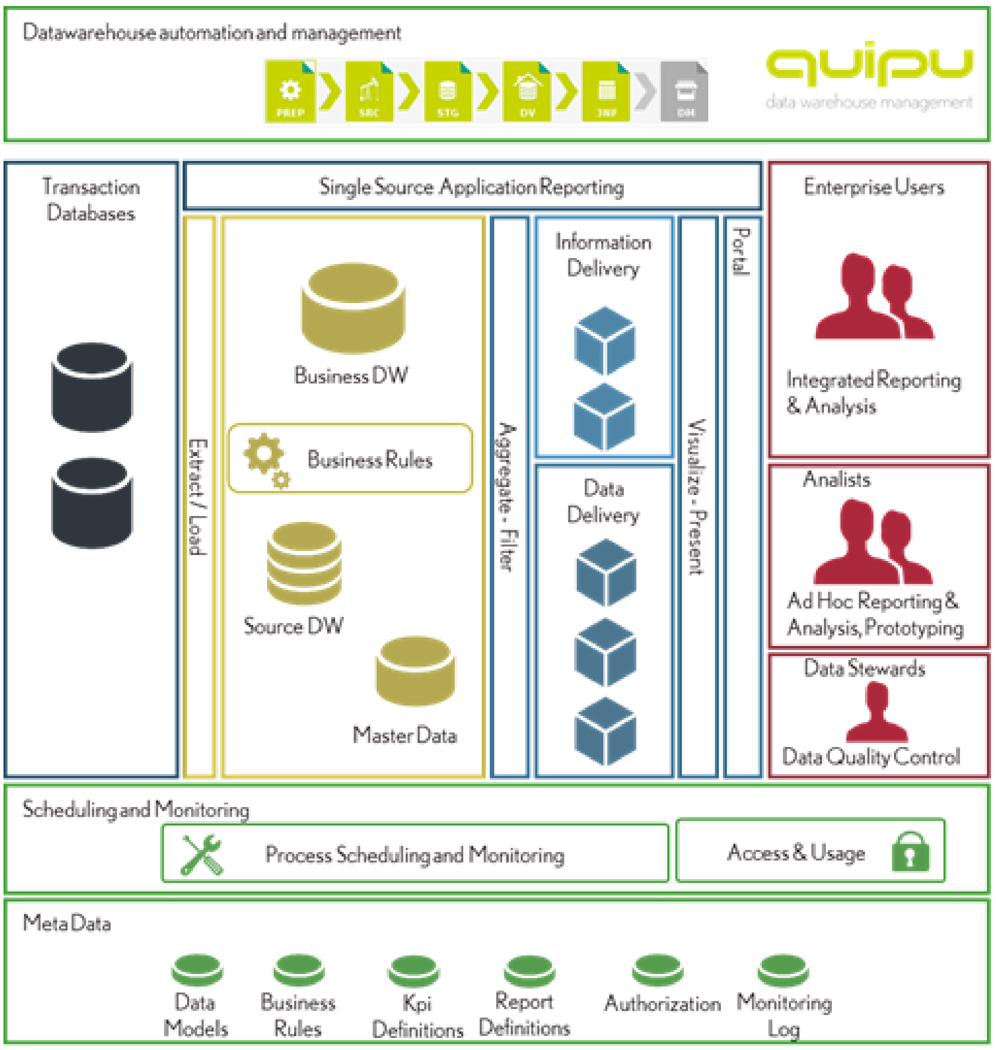
This is the technical reason why on most platforms a Data Mart layer is required.

The data is stored in 'raw' (source) format in the data warehouse, so all of the original data is available. This raw format of the data is good for operational reporting, but not optimal for other reporting and analysis purposes, as there are no derived data (e.g. KPI's) available and integration of data from different sources can be hard.

This is the business reason for creating datamarts.

Summarizing: the Data Mart layer contains a subset of the data and is optimized for query performance: data is integrated, aggregated and pre-calculated, often stored in a star-schema.

The picture belows shows the complete Enterprise DW Architecture.



*Extended DW Architecture*

# DATA VAULT

## Basics of the Data Vault

### Concept

The Data Vault concept has been developed by Dan Linstedt. A good starting point for learning more on the Data Vault is the dedicated wiki page http://en.wikipedia.org/wiki/Data_Vault_Modeling.

The Data Vault structure consists of three basic table types: Hubs, Links and Satellites. To satisfy specific requirements QUIPU additionally supports Reference tables.

## Hubs

### List of business keys

Hubs contain a simple list of *business keys*. They *must* be meaningful for the business and *uniquely identify* an entity (or object) recognized by the business (like customer, product, supplier, invoice, etc.).

Sometimes -ideally- these business keys can be found in source systems as primary keys in databases, or unique indices.

*The business keys - and thus the hubs- are the anchor points of the entire model!*

### Business keys must be unique

It is important to stress the requirement for *global and historical uniqueness* of the business key.

This means that the business key should not be used in different parts of the company to mean different things (often the case in larger organizations, where e.g. *different* customers with the *same key* may be created in different countries).

It also means business keys *should never be reused* (often seen in older systems, where a limited set of keys is re-used after a certain period of time). In both cases the Data Vault will treat the data it stores from these different business entities a belonging to the same entity. This will confuse users of the data warehouse, as it probably confuses the business in the operational processes.

So it is extremely important to find and analyze the business keys before constructing the Data Vault. Analysis should include inspection of source data (data profiling) and verification with business domain experts.

### Inserts only

The hub does *not* contain any other business data. As the business keys themselves do not change, the hub is not time variant. Records will *never ever be updated or deleted*. The table will only see record insertions (which guarantees a very efficient loading process).

### Hub structure

The hubs contain:

- the business key

- the source system that first reported the business key to the data warehouse
- the timestamp when it was first recorded
- optionally a meaningless id (may perform better when business keys are compounded and/or very large)
- some technical fields (like the meaningless ID generated by the data warehouse if chosen and an audit id identifying the process that created the record).

In QUIPU hubs generated from different source-models are separated physically and logically. This means shared hubs need to be created virtually, by (manually) defining views over multiple physical hubs. As an alternative the architect may tweak the QUIPU generated ETL code so all references to a shared hub point to a single physical implementation of that hub. Future versions of QUIPU will support shared hubs more transparently.

## Hub Satellites

At least one satellite per hub per source

When the hub only contains the business keys, the relevant data *belonging to* a business key is stored in the satellites. Each hub has at least one satellite that records at least the business validity of the business key.

As a principle a separate satellite for each source system should be created, recording the data that this specific source holds on the business key, over time. Each of these satellites contains the 'voided' attribute that indicates the business key is no longer valid in a particular source system.

Source system clones can feed into a single satellite

Generally a single implementation of a transaction system should be considered a source system. So an organization that has implemented an ERP system for each of its division should consider each as a separate source.

However, under certain conditions it may be possible -and then often also desirable- to consider multiple transaction systems as a single source. This is the case when:

- all systems share the same logical and technical (data) design (multiple clones of the same system)
- master data management is implemented across all systems, assuring business keys are uniquely defined and shared across all systems.

Under these conditions all systems can feed into the same source Data Vault, possible greatly reducing the number of tables and load processes and at the same time simplifying querying the data.

Still one needs to realize that this simplification may come at a price. For example parallel loading from multiple source systems becomes more complex or even impossible. There are very likely to be other consequences.

**At least one record per business key**

In this minimal case the satellite would only contain technical fields, and minimally a single record per business key that is known by this source (creation/first seen).

When the business key is deleted in the source another record is inserted indicating the business key is no longer valid. This demonstrates that in the satellite the history is stacked: new data is recorded by inserting new records with a Start Timestamp.

**Hub entries can be generated from transaction tables**

Ideally hub entries (Business Keys) are generated when a source table is loaded containing the master data on this Business Key, *before the transactions are processed*. Thus, the information on the BK is available when transactions arrive.

This cannot always be guaranteed, mostly because of timing issues (e.g. master data may be loaded only once a day, whereas transactions are loaded as they arrive during the day).

In the Data Vault this is not an issue, as the Hub will be updated when the transaction arrives. The corresponding record in the hub-satellite will be absent until the master data is loaded.

**Inserts Only**

As changes in the source data are recorded in *new* entries in the satellites also this table can be refreshed with only insert statements.

**End-dating**

Although logically not required, for performance reasons on specific database management systems it may be necessary to end-date a record when a new record is created in the satellite. An additional End Timestamp can be generated. This is the only reason for a record to be updated in the satellite.

**Satellite Structure**

The satellites contain:

- the unique key of the hub it refers to (either the business key or meaningless id)
- the 'start' timestamp: from what time is the data valid (usually the data was recorded in the data warehouse)
- optionally an 'end' timestamp: until what time is the data valid.
- the source system that delivered the data to the data warehouse
- some technical fields (like the audit id identifying the process that created the record, the voided flag that marks the data as deleted in the source).

## Links

**Links record relations between business keys**

Business entities interact in many ways, and these interactions are reflected in (logical) relations between business keys in source systems. In the Data Vault these relations are recorded in Link tables.

Examples of these relations are all kind of structures and classifications in the master data (e.g. chart of accounts, product structure, organizational structure, customer classifications, etc.) and of course the facts registered in the source (e.g. a sale of a product to a customer by a sales representative on a specific date).

Similar to the hub the link table only contains the business keys or meaningless id's, when these are preferred. All data linked to the relation are stored in a satellite table (link satellite).

### Link structure

The link contains:

- the unique key of the hubs it refers to (either the business key or meaningless id): at least 2.
- the timestamp: when was the data recorded in the data warehouse
- the source system that first reported the relation to the data warehouse
- some technical fields (like the audit id identifying the process that created the record).

## Link Satellites

### Similar to Hub satellites

Link satellites share all the characteristics of a hub satellite.

The difference is that a link satellite contains data describing a relationship. For links between master data elements (e.g. the organizational structure) these satellites contain little or no data and provide only information on the validity of the relationship.

But when facts are registered in the relation (e.g. a sale) the link satellite will contain all the details of the fact (e.g. number of products sold, the unit price, total value, etc.)

## Reference tables

### Reference tables can lead to an excessive number of links

A reference table is generally speaking a very simple, rather static table that is often referenced from many other tables. It is often used in source systems to define a restricted value set (domain) for a certain attribute. E.g. the list of currencies, countries, colors available for a product, etc.

By default (and following strict Data Vault modeling principles) a hub and satellite are created for each of these tables, as well as link tables to all hubs that reference these tables.

Querying these tables can then become rather complex, as in all cases the satellites of the reference tables need to be joined in (via the link tables) when the current value is requested for a given business key.

Also the model can become more cluttered due to the abundance of link tables that are generated.

**Links are not generated for reference tables**

To alleviate this problem it is possible to mark a source table as a reference table. The result is that a hub and satellite will be created for the reference table as normal, but *no link tables* between the reference hub and the referring hubs are generated.

**Joins to reference tables must be added in the queries**

The reference *to* the reference table must of course still be stored somewhere. This is now a function of the satellite of the referring entity.

Of course the data mart architect will have to understand what reference tables exist, and create the appropriate joins in his queries.

**History is kept for reference tables**

In the QUIPU implementation of reference tables, with a 'regular' hub and hub-satellite all history of (changes to) the reference table are kept. A further simplification would be to simply overwrite changes to the reference table, as they are very likely to occur only sporadically. Of course dropping history violates a basic concept of the data warehouse (always store all history) and potentially poses an auditing risk.

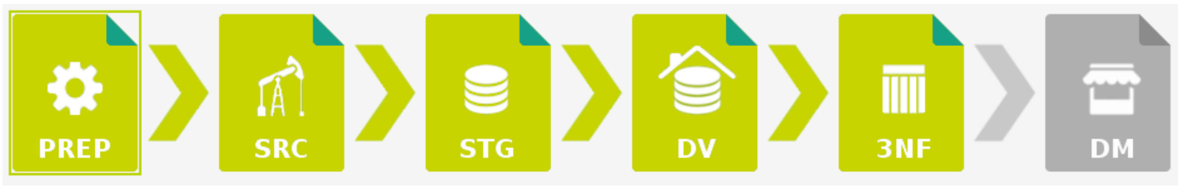
Reference tables without history are not (directly) supported by QUIPU.

**Caution: use with care**. The Reference table feature should be used with proper caution, as it breaks one of the key aspects of the Data Vault. In a strict Data Vault all relations are modeled as links. This makes it extremely easy to assess the impact of change. The existence of references from satellites to the hubs of reference tables breaks with this concept. This means changes to reference tables can impact a larger part of the Data Vault model. The benefits of simplifying the Data Vault model are very solid (for load processes, query processes and queries) so it is common practice to implement reference tables whenever it makes sense.

# MODEL GENERATION

## Data Vault Generation

| | |
|---|---|
| Five steps to Data Vault generation | The generation of a Data Vault from a source system follows a seven step approach:<br><br>• Import or reverse engineer a source model<br>• Generate the staging area as a (near) 1 to 1 copy<br>• Generate a Data Vault proposal from analysis of the staging model<br>• Evaluate the result and influence the generation process by:<br>  • (Re-)Define the business keys<br>  • (Re-)Define the relations between source tables<br>  • Ignore source columns that contain invalid or irrelevant data<br>• Regenerate the Data Vault model<br>• Final customization<br>  • Rename tables and/or columns if required to get an easier to understand model and add meta data to document the model<br>  • Split satellites, move attributes between satellites to group logically related attributes and/or separate frequently updated attributes from more static attributes<br>• Generate the scripts |



## Data Vault generation step 1: Import model

| | |
|---|---|
| Reverse engineer RDBMS | Reverse engineering is supported for a large number of relational database servers and file based 'databases' (Excel, XML, CSV, etc.). Requirements are:<br><br>• availability of a JDBC connector<br>• a connection between the QUIPU back-end and the database (or file) server<br>• sufficient rights to read the database catalog.<br><br>The reverse engineer module will import tables, columns, relationships and indices. |

**No direct support for database modeling tools**

Direct import of the model from a database modeling tool (e.g. ERWin) is currently not supported. The easiest way to transfer a model from such a tool to QUIPU is via generating a DDL script in the tool and using it to create a dummy database in any of the many supported database systems. This database can then be reverse engineered by QUIPU.

## Data Vault generation step 2: Generate Staging

**Staging**

From the imported source model a staging area model is generated. This model is basically a one-to-one translation of the source model where data-types are converted when required, but otherwise no changes are made.

**Copy of source model**

Staging models generated by QUIPU are almost 1-to-1 copies of their source models. The only difference between the two is that the staging model contains some extra administrative fields. These fields are:

- A timestamp to keep track of the time data entered the staging area.
- A sequence number to logically order data entries in the staging area.
- A voided field indicating whether a data entry contains a deletion in a source system. This is needed to handle delta loads.

Audit fields to keep track of the source of the data entered in the staging area.

## Data Vault generation step 3: Generate Data Vault proposal (Analyze staging)

**Rule based generation**

The Data Vault is generated from the staging model based on a fixed decision tree implemented in the code.

The first rule is that the table must have a unique key. It consists of a single or multiple columns. And each column can contain a technical or business key. If a unique key does not exist, the table is ignored.

Further rules:

*A source table is a Hub unless proven otherwise*: the general principle in the generation of the Data Vault is that all tables in the source become hubs, unless it can be determined (using a set of rules) that the table should be treated as a link or satellite table. This is generally the case when the principles ruling the Data Vault can be recognized in the source system.

A clear example is the many-to-many tables found in most relational systems that register transactions (e.g. a sale may be registered in a table with as primary key a reference to the product table, the customer table and a timestamp). Such a table should be recognized and translated to link and link satellite construction.

**Analysis options**

It happens that one (or more) of the business keys in a table is not a real business key. For instance: a table contains data with a business key and extra date column as its unique key. Strictly speaking, such tables should translate into link-satellites, with the extra (in this case: date) column representing a second Business Key placed in its own Hub. But often these extra columns have little business value and are NOT linking pins to other parts of the data model. The resulting Data Vault is then overly complex and hard to query.

QUIPU offers two distinct options to prevent this situation:

**Multi-active satellite:** Allow creating satellites with a different cardinality then its hub. The table is mapped to a satellite on the hub, but the satellite contains (potentially) multiple records for each business key at any given time (hence the name: multi-active). The records are distinguishable by the extra key field(s).

**Allow peg legged links:** Allow links that link only to one hub. Now the extra key field(s) are in the link table, as one would expect, but the corresponding hub is not implemented. All data on this link is kept in a link-satellite. See http://en.wikipedia.org/wiki/Data_Vault_Modeling for more information.

## Log of the results of the decision tree

The result of evaluation of each node in the decision tree is recorded in a log file and is shown in the GUI. This allows the architect to evaluate the results and make the necessary changes to the source model to arrive to the desired result within the limits of the decision tree.

## Business keys derived from primary or alternate keys

The Data Vault decision tree uses a primary key or unique index constraint to find the candidates for business keys.

In many cases the primary key is a simple code that is actually meaningful to the business, providing a natural business key.

The source system may use meaningless keys, however, that have little or no value to the business. The real business key is then often defined as an alternate key and/or a unique index is defined on the real business key.

## Source system blues

QUIPU will always create a Data Vault that is technically correct. It will however not always be the most efficient model, nor will it always be the model that you would like to have.

The reason for this is that the optimal Data Vault model cannot be derived by analyzing the source system *structure* as it is registered in the catalog of the RDBMS.

There are plenty of potential problems, including but not limited to:

- Many source systems are implemented without foreign key relationships guarded by the RDBMS. The database consistency is then assured by rules in the software. Without foreign key relations, it is not possible for QUIPU to create link tables. NOTE: these relations can be added manually in QUIPU to aid automated generation.
- Table and column names in the source may be generated, or for other reasons be impossible to interpret. It is clearly undesirable to import this interpretation problem into the Data Vault model.

- Tables are often 'overloaded'. Multiple logical entities are mapped into a single table.
    - Example 1: A column in the table indicates to what entity the row belongs. This is often found in source systems as implementation for reference tables, consisting of a code and description.
    - Example 2: Table are not fully normalized, so a 'parent' table (e.g. product group) is redundantly stored in a 'child' table (e.g. product).
- Columns may be used for other purposes then their names imply. Again, importing these confusing names into the Data Vault model is undesirable.
- Logical entities may be split over multiple physical tables (e.g. transaction per month)

All of these issues (and many more) will lead to the generation of a sub-optimal Data Vault. Issues related to the *structure* can -within limits- be resolved within QUIPU by creating or modifying relations, renaming tables and columns, defining business keys. But issues related to the *contents* of tables (like overloading) cannot currently be resolved by QUIPU. If you want to resolve these, you will have to create some ETL to split or combine the physical tables and find a way to offer the logical 3NF model to QUIPU (e.g. by creating a dummy database with the correct structure).

Data Warehouse Architect is in control

Ultimately the data warehouse architect needs to check the result of the Data Vault generation process and manually intervene if necessary.

*NOTE:* The initial release for QUIPU will generate a separate Data Vault for each source model offered. So QUIPU will *not* integrate different source model into a single, linked Data Vault.

## Data Vault generation step 4: Evaluate and modify generated Data Vault proposal (Prepare staging)

Including / Excluding columns or tables

Specific tables and/or columns can be excluded from the generation process by marking them as 'to be ignored'. This is particularly important when generating a source data warehouse from a source that contains tables or columns that are not used (properly) by the business.

History

By default all columns are set to track history after generating the Data Vault. However for some columns you might want to disable history tracking and include only the latest provided data. This may be the case for highly volatile, operational data (like process status-information), or on the opposite side of the spectrum: for very stable, descriptive information (like the descriptive name for organizations, equipment, etc.). In those cases you can be sure that people will only ever be interested in the actual values.

QUIPU allows switching history tracking off for particular columns.

NOTE: switching off history breaks the Data Vault. You will no longer have a full audit trail for this data in your data warehouse. So use it wisely!

| | |
|---|---|
| Adjusting the business key | The QUIPU analysis routine may not always be able to correctly find the Business Key(s) from the structural information it gets from reverse engineering the database. You can add and remove columns to the business key to correct these situations.<br><br>Columns that are ignored cannot be part of the business key.<br>Every table needs to have a business key. If not the data vault generation process will automatically ignore the table.<br>We don't recommend using technical (or generated) columns as business keys. While this will result in a technically correct Data Vault model, you can encounter problems when creating datamarts. |
| Reference tables | A reference table is a table that will be included when generating a Data Vault, but all foreign key relationships connected to this table will be ignored. This will result in a reduced amount of link tables in the generated data vault. Tables can be marked as reference table in the QUIPU GUI. |
| Add / remove relationships | QUIPU may not always get the correct relationships from the reverse engineering routine. These relationships are essential to the Data Vault generation routine, as QUIPU will map them to hub-satellites, links and/or link-satellites. It is therefore essential that QUIPU allows relations to be edited.<br><br>Each relationship should have a sensible, descriptive name, as this name is used when the relationship is transformed to a link as the link name. |
| Relationship groups | As a default QUIPU will generate a link table between 2 hubs for each relationship it finds. Very often it is more sensible to combine several relationships into a single link table, linking multiple hubs together.<br><br>This can be achieved by defining a 'Relationship group'. |

## Data Vault generation step 5: Regenerate Data Vault proposal

| | |
|---|---|
| Implementation options | When the logical model has been defined to satisfaction, it is time to start considering the -physical- implementation options. These options have no impact on the data content of the data warehouse, but may have an impact on the performance in a particular database system or affect the possibilities for scheduling, monitoring and data maintenance tasks. |
| Model level implementation options | **Record source and timestamp:** Choosing this option will add a record source and timestamp column to each table in the staging area (default).<br><br>**Audit ID:** Choosing this option will add an audit ID column to each table in the staging area. The audit ID can point to a record in an audit table with record source and timestamp information, but could also contain additional information about the loading process (e.g. on extraction date, # of records in the export, operator, etc.). |

| | |
|---|---|
| | *NOTE:* QUIPU will load these fields into the data warehouse for later reference, but will NOT fill these fields (in the generated ETL code to load the staging area). Loading the staging area is outside the scope of QUIPU. |
| Model/Table level implementation options | **Generate surrogate keys:** QUIPU can provide each data vault hub and link with a generated primary key (hub_id or link_id). Subsequently this key will be used in the relationships that connect these tables to each other and to their satellites. When this option is disabled the business key of the staging table will be used as primary key in the Data Vault. Using generated keys can greatly reduce the size of the data warehouse and improve query performance, especially when business keys are multi-field and large (strings). It comes at a cost, however: loading the data warehouse becomes (potentially a lot) slower, as all business keys need to be translated to the correct surrogate key. Queries may become harder to build and read, as the surrogate keys have no meaning to the business.<br><br>**Use end-dating:** By default all satellites in the generated data vault will get an extra column called "load_date" that will contain the date of inserting. This column is required and sufficient for keeping the history. It is therefore mandatory and part of the primary key.<br><br>Retrieving the correct historical record from such a table may be difficult for a relational database: a complex query may be required that could also perform badly. QUIPU offers the option to add another column called "load_date_end" that can be used to line up all records in chronological order. This field will be filled when a new version of the record is entered in the database, thus 'closing' the previous record. It can be used to simply single out the correct record with a SQL construct (load_date_start <= date < load_date_end). |

### Data Vault generation step 6: Modify / add metadata

| | |
|---|---|
| Modify the generated Data Vault | When the Data Vault is generated, there may be cause to modify some aspects; to simplify the model or improve performance. |
| Split a satellite | The fields in a satellite may not change at the same rate. Also, some fields may always change as a set, not individually.<br><br>Example: An employee table may contain info on name, sex, date and place of birth and address details. The personal identification attributes will (in general) never change. The address details may change, but if they do the individual attributes (street, city) often change together.<br><br>This is a typical case where splitting the satellite could result in a better model: attributes that 'belong together' and 'change together' are placed in a separate satellite. This makes the model easier to read and interpret when at the same time requiring less disk space and offering better performance. |
| Merge satellites | Sometimes it makes sense to combine multiple satellites in a single one. Thus, one can consolidate information from multiple source tables in a single table in the data warehouse. The model becomes simpler at the (potential) cost of storage requirements. |

| | |
|---|---|
| | *NOTE:* Combining satellites from different source systems is not recommended. Different source systems have very often different characteristics in terms of completeness, correctness, timeliness etc. It is wise to store the data apart, so these characteristics can be easily assigned to a complete satellite table in the data warehouse (as opposed to columns in a combined satellite). |
| Rename objects | You can rename all tables and columns in QUIPU, so you can apply any naming convention. This will never alter any mappings. |

## Data Vault generation step 7: Script generation

| | |
|---|---|
| Template based generation | QUIPU uses templates to generate scripts to generate databases and load code. These templates can be customized to support specific features of the various RDBMS's. So for each SQL dialect a template could be provided. The initial release comes with a template for standard ANSI SQL 92 intermediate code. Additional templates will be released when they become available through development at QOSQO or in the community. |
| DDL | QUIPU generates Data Definition Language scripts to create:<br>● The staging area of the data warehouse<br>● The Data Vault area of the data warehouse<br>● 3NF-views on the Data Vault (mimicking the original source model) |
| ETL | QUIPU generates the load code for the staging area and the Data Vault.<br><br>The staging area load code assumes that the source database is situated in the same RDBMS, and the specified connection to this RDBMS has sufficient rights in the source database. This is of course often not the case. In these situations the code may serve as a sample demonstrating the logic to fill the staging area. |
| Delta vs Full Load | The Data Vault load code implements a delta detection mechanism. This means the code will function properly on both full data sets and delta data sets (only containing the changes in the source), *and you are not required to indicate whether you offer full or partial data sets*.<br><br>The problem with delta data sets is that QUIPU will not be able to detect *deletions* in the source (unless deleted records are marked by an attribute and offered in the delta set). This is the *one and only reason* for QUIPU to offer an option indicating that the data set offered is a full data set: *any record not in the full data set is considered deleted from the source.*<br><br>QUIPU will end-date the current satellite record(s) and create a new record marked as deleted. Also, any hub-entry not found in the source will be marked as deleted. |
| Staging Load table | In practice, delta and full loads are often required intermittently for a specific table. It is common practice to process the smaller delta data sets on a daily basis, |

accepting that deletes are not recorded. Once every week or month a full data set is processed to record the deletes that occurred in this period.

To facilitate this practice QUIPU implements a special table in the staging area with a record for each source table. The data warehouse manager can indicate for each source table whether a full load or a partial load is required.

QUIPU generates code that will check this table and automatically perform the full or partial load as specified in this table.

## Splitting the load process

It is possible to split the load process in multiple jobs. For instance in many cases it makes sense to load the master data only once a day (with the full load option), but load the transaction data multiple times a day (to get more actual data and prevent very large load jobs at the end of the day).

This works well, with the one consequence that for NEW master data only the business key will be available until the all master data is refreshed.

All of this is not directly supported by QUIPU, but can be achieved by either splitting up the generated ETL scripts or building a scheduling solution that directly addresses the QUIPU repository.

## Re-starting load jobs

When the same data set is offered multiple times (either full or delta sets), no change will be found, and thus no changes will be recorded in the Data Vault.

This means interrupted data loads can be restarted without any adverse effects.

All tables in a single export can be loaded in any particular order (and even in parallel), although it makes sense to load master data before loading transaction data.

Please note that offering jobs out-of-order (older extracts after newer extracts) will cause incorrect results in the Data Vault!

*NOTE:* that the ETL scripts will only implement mapping from source tables to target tables. No transformations can be specified.

## End-dating

There is an option in QUIPU to add end-dates to the satellites. The end-date allows simpler SQL (and often: faster) queries on the Data Vault, using the SQL construct "WHERE load_date_start <= point-in-time < load_date_end"

End-dating the satellites means that the each record in the satellite needs to be updated with the start-date of a later record. If no later record exists, the end-date gets a date very far in the future (31-12-9999).

End-dates can be updated when new entries arrive, but this slows down the data load processes, which can be a problem in certain scenario's (e.g. for near-real-time updates for operational BI). That is why QUIPU implements end-dating as a separate process that should be executed *after* loading the data.

## Views

Some views will be generated by QUIPU that will present an easier to interpret (and query) representation of the Data Vault.

The views will come close to rebuilding the 3NF data model that was used for the Data Vault generation (unless manual interventions have changed the model).

As the Data Vault stores historical versions of all data, and the 3NF model in general does not, the views will need a point-in-time to indicate which historical view is requested.

Standard views included are:

- the current status (using today as the point-in-time).
- the current status including all deleted data (using today as the point-in-time)
- the full historical record

## ADVANCED TOPICS

### Historical Data Archive

**'Informal' business keys**

In our practice working with Data Vault models for our clients we have learned that is not always practical or even possible to identify the 'true business key' for all business objects. Many systems work internally on technical keys. Business keys are often available, but seldom guarded as vigorously as the technical keys. This leads to a variety of data quality issues with these 'informal' business keys.

There may be any or all of the following problems:

- missing keys (no entry)
- duplicate keys for different objects (e.g. due to re-use of keys)
- multiple keys for the same object (e.g. due to spelling faults)
- missing keys in the master data system (new keys are entered in slave systems)
- expired keys in slave systems

**Bad business keys break business processes**

Problems with business keys are a problem in operations, as it introduces ambiguity in the business processes. The resulting confusion is the cause of many inefficiencies in the business.

**Bad business keys lead to inconsistent BI**

But apart from that they seriously reduce the capability of BI to produce reliable integrated reports spanning multiple systems and processes. Reports on the same business facts from different systems will produce different results!

**Master Data Management is a pre-requisite**

Master Data Management practices need to be introduced and enforced to tackle these problems. In most cases that will mean operational procedures and systems need to be adapted and possibly a dedicated MDM system needs to be implemented.

The business cannot wait for such efforts to bear fruit before they can get access to their vital management information.

**Historical Data Archive (HDA) built on technical keys**

What can be done when business keys are not available? The next best thing is to store the historical data linked to the technical keys of the sources. This at least assures that a true record of the historical changes of the source can be kept.

This can easily be achieved using QUIPU, by simply following the default process. When QUIPU analyzes a source model it will always look for business keys as primary keys or unique indices of the source tables. If it finds such a key it will suggest to use this key as the business key. Of course in practice it often is a technical key, not a business key.

## Risk: Disconnected source Data Vaults

In the most extreme case (where all source systems have only disconnected technical keys) this approach results in a set of disconnected source data vaults, one for each source.

The value of a hub in this environment can be disputed, as the primary role of the hub is to serve as a connection point between systems/processes. When a hub contains technical keys that are only known in a single system this value evaporates.

## HDA: satellites only

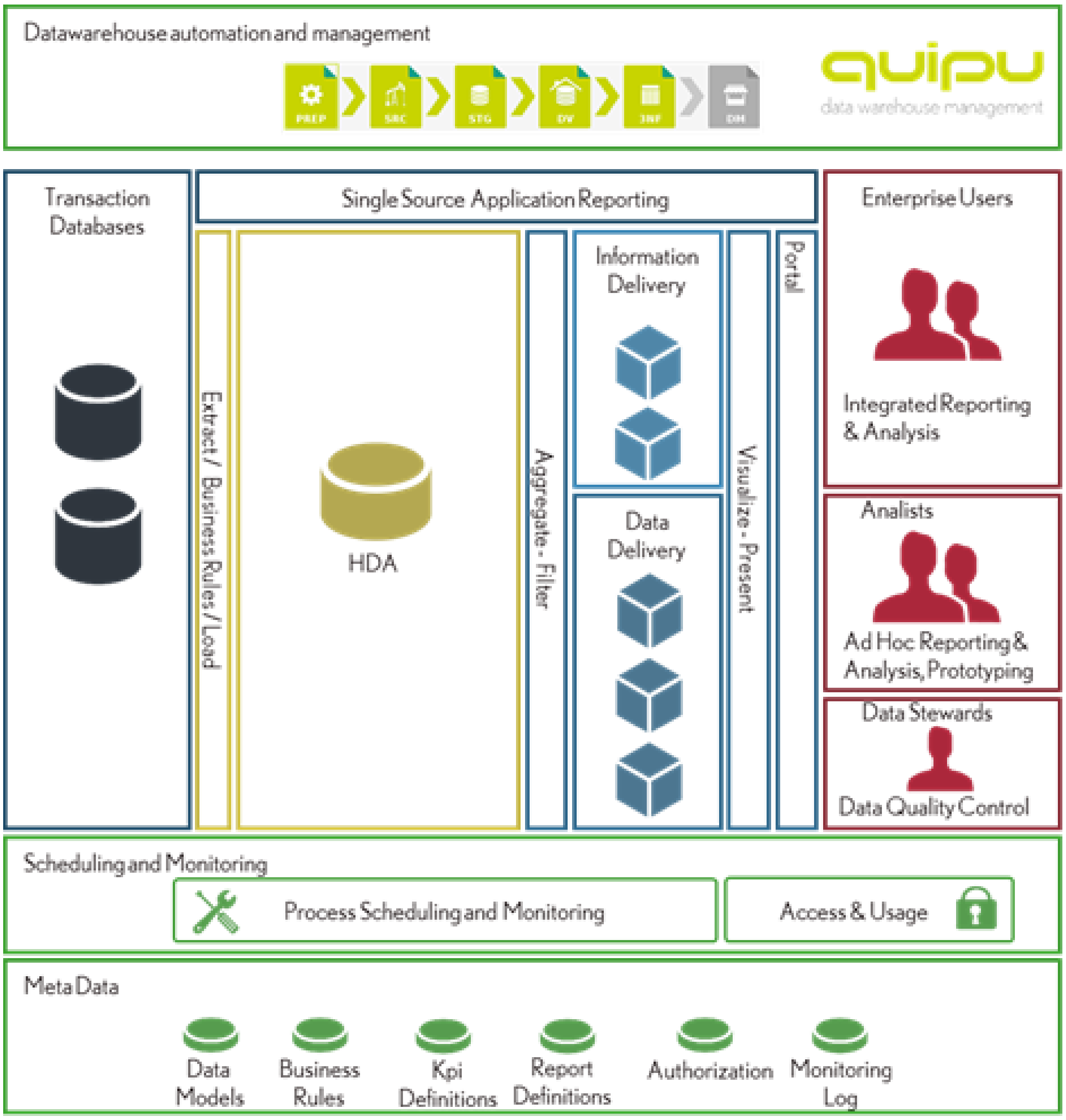
It is therefore logical to dispel the the hubs (and links) and only implement satellites.

The resulting environment is of course no longer a Data Vault, but is sometimes known as a Historical Data Archive (HDA). You could also label it as a Persistent Staging Area.

## Building the Data Warehouse on the HDA

The advantage of implementing the HDA akin to Data Vault satellites is that is still possible to later find/define the true business keys and then start building a 'Data Vault'-like data warehouse by adding hubs and links. In some cases it may be possible to link these new hubs directly to the existing satellites.

The picture below shows the complete architecture based on the HDA.



## *Historical Data Archive*

## Support for distributed data warehouses: Hash keys

### Big Data challenges

In the past years a new phenomenon has received a lot of attention, under the umbrella term 'Big Data'. With the arrival of new data capturing and processing technologies, in particular NoSQL databases and HADOOP, it became feasible to build and analyze huge data sets that are far too large for traditional relational databases.

Often these data sets contain unstructured or semi-structured data that make it hard to combine the data (or results of the analysis) with data available in data warehouses. But there is potentially great value in achieving just that: combine the well-controlled and structured data residing in data warehouses with the new data sources residing in HADOOP and similar databases.

One of the requirements for achieving this is linking the data in the Big Data stores to the business keys that form the anchor points in the (Data Vault) data warehouses. These links effectively expand the Data Vault concept to include data from Big Data stores.

The challenge then becomes to:

1. identify the business keys in the Big Data sets and
2. assign them the meaningless keys that are used in (most) data vaults.

Both steps can be a challenge indeed.

The first because data is often not sufficiently structured and does not contain the controlled business keys in the data warehouse. This means that classification algorithms must be designed that can reliably add the business keys to the data in the Big Data stores: manual classification -as is usual in the structured sources of the data warehouse- is almost never an option as the data volumes are too large and/or the sources are not controlled by the organization. What is 'reliable' will depend largely on the intended use of the data. For many purposes (like marketing or user satisfaction assessment) relatively crude classification is sufficient.

The second step is traditionally solved by lookup's in the data warehouse: the meaningless key is retrieved from the data warehouse if the business key already exists, or a new meaningless key is generated. In the latter case the business key - meaningless key pair is added to the data warehouse. This simple process can become a bottleneck however when processing huge data streams seen in Big Data environments in parallel environments.

## Hash Keys for performance improvement

An elegant way to solve this issue is by using a hash algorithm to *calculate* the unique meaningless key from the business key. This algorithm can be implemented on all -distributed- systems so they can process the incoming data stream in parallel. An additional benefit of this approach is that even in traditional data warehouses a hash key can often be computed very efficiently thus reducing data load times.

The theoretical problem with this approach is that there is an information overload in hash keys. Or, put differently- there exist a (very small) chance that the same meaningless key is computed for two *different* business keys. That would obviously result in a corrupted data warehouse.

There is much debate on whether or not this theoretical chance is something to take into account in the design and implementation of a data warehouse. If one is concerned about this happening it is possible to implement simple queries that will detect this collision in the data warehouse and then devise corrective actions.

As of version 3.1 QUIPU supports the use of hash keys as meaningless keys, for those that need it.

## Model extensions

### A typical data warehouse is constantly adapted to changes in the environment

QUIPU can generate all scripts required to generate the data structures for the data warehouse: staging and data warehouse (source data vault of HDA, business data vault). QUIPU also generates the scripts that load the data warehouse from the staging area and even a starting point for loading the staging area from the source(s).

The generated data warehouse can be loaded periodically with new data (typically daily), quickly building up potentially large volumes of data for reporting and analysis.

In a typical environment at some point in time changes to the data warehouse need to be made. This can be caused by changes in business processes, business applications or (often) both. It can also be the result of different reporting and analysis requirements or the correction of flawed business rules.

In all of these cases it may be necessary to make changes to the data warehouse data structures. The best possible model will then be found when re-generating the data models from scratch using QUIPU and the modified source model or enterprise data model. But this is not always the best approach, or even possible.

### Regeneration is a dead end alley

The main disadvantage of the regeneration approach are:

- even the simplest changes (like adding new data elements to existing tables) will lead to new tables to be generated, leaving 'dead' tables in the data warehouse: the outdated tables will no longer be refreshed. This means that all data marts (or reports directly accessing the data warehouse) relying on these old data structures will have to be modified, even when these reports do not now or ever require the newly added data elements.
- new or modified data structures will be empty when recreated. In order to preserve previously loaded data a data migration script might be required to initialize the empty structures with data from now outdated structures. This step can be rather complex, labor-intensive and time intensive.
- large data warehouses (in terms of complexity and data volumes) may require vast data migrations that will drive the cost for a change to levels that are inhibitive. In these cases the data warehouse becomes highly resistant to change, completely frustrating the business and defeating the original design goals of the data warehouse.

Most real-life data warehouses reach the point that regeneration is no longer an option within about a year after initial go-life.

### Closed for change, open for extension

The data warehouse must be flexible and changes should be implemented incrementally, limiting changes to existing structures to a minimum. By keeping data structures in tact (and 'live') whenever possible the impact of the change on the existing reporting and analysis applications is minimized.

In short: the data warehouse should be *closed* for change but *open* for extension.

For this reason QUIPU supports model extensions. Key feature of model extensions in QUIPU is the ability to lock tables (by default: all existing tables) so they will *not* be regenerated. QUIPU will then figure out how to complement the existing data vault with new structures to capture the changes.

## No data migration supported (yet)

The current release of QUIPU does not support data migration. So the newly created data structured will be empty and must still be populated.

## Change types supported by QUIPU

The following table shows the different type of changes that QUIPU currently supports, and how they are resolved.

| Object | Change Type | Structure (DDL) | Loading (ETL) |
|---|---|---|---|
| Attribute (non-key) | Add | Add Sat | • Add load code for Sat |
| | Remove | No change | • Remove load code for column in Sat |
| Attribute (key) | Add | Add Hub / Link / Sat | • Add load code for Hub/Link/Sat;<br>• Remove load code for obsolete structure |
| | Remove | Add Hub / Link / Sat | • Add load code for Hub/Link/Sat;<br>• Remove load code for obsolete structure |
| Attribute (all) | Data type | User is informed during reverse engineering. He has to take actions on what is best to do. Currently Quipu does not have any extra support. | |
| Table | Add | Add Hub / Link / Sat | • Add load code for Hub/Link/Sat |
| | Remove | No change | • Remove load code for Hub/Link/Sat |
| | Class (H,L,S) | Add Hub / Link / Sat | • Add load code for Hub/Link/Sat;<br>• Remove load code for obsolete structure |
| Relationship | Add | Add Link / Sat | • Add load code for Hub/Link/Sat;<br>• Remove load code for obsolete structure |
| | Remove | No change | • Remove load code for Hub/Link/Sat |

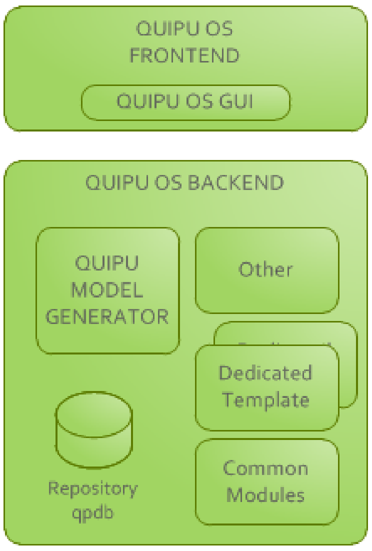
## QUIPU TECHNOLOGY

### Client Server Architecture

Internal structure

QUIPU is internally structured with a strict separation between components and (front- and back-end) layers based on open interfacing standards in order to:

- isolate the generic, data warehouse and data vault specific logic from the specific optimizations and implementation details of the technology used
- allow concurrent development of new functionality
- allow new technologies to be integrated and optimized (through add-in components like plugins, external libraries etc.)



Thin client concept

QUIPU has been designed with a thin client in mind. This means all core functions are implemented in the backend, and the client only implements the GUI component of the application.

Template based generation

The generation of DDL and ETL code is template based. This allows QUIPU to generate optimized code for a great variety of target (database) systems. The standard templates produce ANSI SQL code that will run unchanged on many RDBMS systems. For some database systems optimized templates are available.

### Back-end: Groovy / Grails

Java Virtual Machine

A key technology decision for QUIPU is to have the backend implemented on the Java Virtual Machine.

The Java Virtual Machine that executes the Java (and Groovy) code is the premier platform for portable applications (that will run on almost any hardware platform

| | |
|---|---|
| | and operating system) and is also well known for its capabilities for web applications.<br><br>Running the QUIPU backend on the Java Virtual Machine assures that this key component of QUIPU can run in all conceivable technical platforms, from stand-alone pc's to enterprise server environments.<br><br>The latter is specifically relevant as for more advanced implementations production systems need direct access to the QUIPU backend. The QUIPU backend must then be able to run in the same protected environment as these production systems. |
| Groovy programming language | The drive to develop Groovy comes from the perception that Java is not so very productive due to the enormous (coding) overhead it requires for very mundane tasks. Groovy is much more permissive and requires much less code to achieve the same functionality. |
| Grails Framework | GRAILS stands for Groovy on Rails and borrows a lot of the concepts and mechanisms of the well-known Ruby on Rails environment. Grails is a framework that bundles a number of key Java libraries and puts convention over configuration.<br><br>Groovy -as programming language- allows the programmer to build nearly all perceivable functionality. But of course that would require a lot of time and effort. For many generic tasks Java libraries have been built and many of those are in the public domain. Grails has selected and bundled a number of these (e.g. Hibernate to interact with databases, or Spring to generate some basic GUI functionality) into the Grails package. A common problem associated with the use of such libraries is the need to configure them correctly. The more powerful the package, the more options can be configured in configuration files and the more difficult it becomes to make them work.<br><br>Grails solves this problem by pre-configuring the libraries to assume the programmer adheres to a set of well described *conventions*. These conventions consist of rules on how to structure the code (in files) and where to put them (a set of directories with a specific purpose each) as well as naming conventions. When the programmer adheres to these conventions virtually all need to configure the libraries is removed. Of course, the configuration files are still available to the programmer if certain -default- behavior needs to be tweaked. |

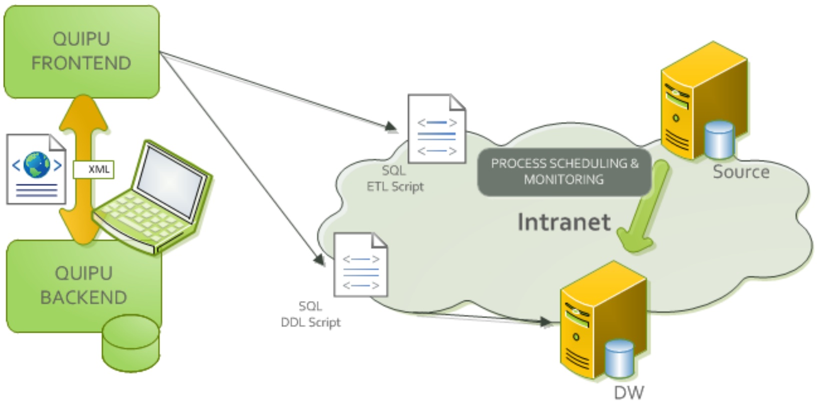
### Front-end: C# / .Net

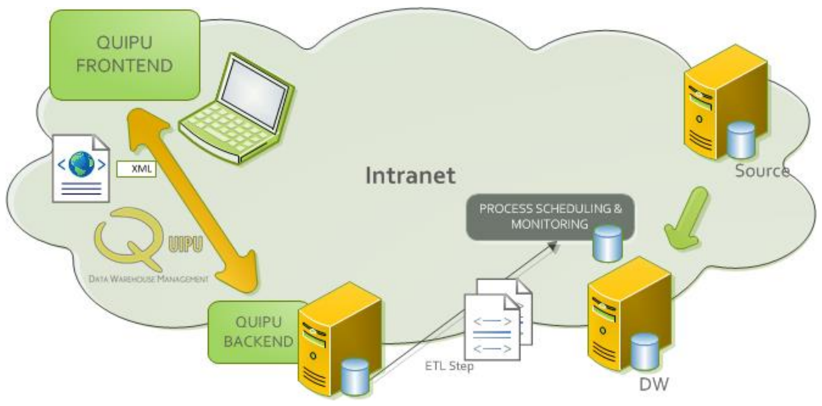
| | |
|---|---|
| Thin C# client | The front-end for QUIPU has been developed in C#, and requires the Microsoft .NET runtime modules to be installed on a pc with Windows installed. The front-end has deliberately been kept as thin as possible, so most functionality of QUIPU is implemented in the back-end. Front-end and back-end communicate through standard http calls. This makes it entirely feasible to develop alternative front-ends (e.g. using AJAX, Silverlight, etc.) or complimentary front-ends (e.g. scripting certain processes to execute without interaction). |

## Database support

| | |
|---|---|
| Broad RDBMS support for Quipu Repository | QUIPU stores all the information it manages in a (relational, 3NF) repository. This repository can exist in a very large number of database management systems (courtesy of Hibernate). See Hibernate documentation for a list of all supported databases and versions.<br><br>This broad support of database systems allows the QUIPU repository to be installed in the target database environment, which is required for the (future) monitoring and scheduling functionality. QUIPU configures itself upon installation: only an empty database needs to exist with a valid connection and proper authorization. QUIPU will then automatically create an empty repository.<br><br>'Out of the box' QUIPU contains a file based RDBMS (HSQLDB) so no separate RDBMS is required to install and run QUIPU. |

## QUIPU deployment options

| | |
|---|---|
| Standalone / Prototyping setup | In this minimal scenario QUIPU front-end and back-end are installed on the same machine (PC). QUIPU is used to generate the DDL scripts and ETL scripts that can be copy-pasted into any text editor. Scripts are available for bulk export .<br><br>For each physical data model (schema) a script file will be generated that will create or fill the entire model.<br><br> QUIPU FRONTEND<br>XML<br>QUIPU BACKEND<br>SQL ETL Script<br>SQL DDL Script<br>PROCESS SCHEDULING & MONITORING<br>Intranet<br>Source<br>DW<br><br>It is up to the architect to deploy these scripts in the production environment, to create the DW structures and then fill these structures with data. |
| Integration in server environment | A more advanced option is to install the QUIPU back-end on a machine that can be -reliably and securely- accessed by the process scheduling and monitoring tool (often a component of the ETL software). |



In this setup it is possible for the ETL software to access individual ETL step generated by QUIPU and execute them separately. This allows for more elaborate monitoring and exception handling.

The setup can be made fully generic, so few or no modifications are required in the process scheduling and monitoring setup are required when new version of the ETL are generated by QUIPU.

This setup has been successfully deployed in combination with Microsoft SQL Server Integration Services (SSIS) and open source Pentaho components (Data Integration, formally known as Kettle).

# MORE INFORMATION

## About QOSQO

| | |
|---|---|
| Our services | QOSQO is founded in 2008 as a sister company of Nippur (founded in 2002). QOSQO is the leading company behind QUIPU, and provides data warehouse services for companies using QUIPU or other Data Vault modeled data warehouses. |

## License model

| | |
|---|---|
| Subscription model | QUIPU is licensed for a yearly fee, including product support and product updates. More information can be found on http://www.datawarehousemanagement.com. |

QOSQO BV
Spoordonkseweg 7 5688 KB Oirschot, The Netherlands
Phone: +31 (0)499 577 562
E-mail: info@QOSQO.nl,
http://www.QOSQO-services.com
IBAN: NL72 ABNA 0421 6436 17