User Guide
Agile Data Vault Modeler

Release 1

Jon Nedelmann, 16.12.2016

1. Introduction

Agile Data Vault Modeler (We call it the modeler for short in this guide.) is a web application which can be used to create data vault models. Data vault is a modeling technique that can be used for data models of a core data warehouse. There have been for a long time two schools of thought how the core layer of a data warehouse shall be modeled. On the one side there was the Kimball philosophy that everything shall be designed in a dimensional manner. On the other side there was Inmon's idea to use a third normal form added with time information. Both approaches have there pros and cons, but both do not behave well when your data model changes due to new requirements. Data Vault is a new approach which overcomes this problem. It was established by Daniel Linstedt (see e. g. [Linstedt 2010], [Hultgren 2012], [Hahne 2014]) and developed further to Data Vault 2.0 as described in [LinstedtOlschimke2016]. Furthermore, there exists a visual modeling language Visual Data Vault, see www.visualdatavault.com for more details.

This user guide is not a tutorial for Data Vault 2.0 or Visual Data Vault. We assume that you have a basic understanding of these techniques, as presented e. g. in the chapters 4 to 6 of [LinstedtOlschimke2016]. We will discuss some aspects in detail just when the modeler approach differs from the one presented in that book.

The modeler follows the Visual Data Vault language, but it is not a strict implementation of this standard. For example, Visual Data Vault has several rules, how the line ends can be decorated. The modeler ignores these rules.

But the modeler also goes beyond the modeling techniques of Visual Data Vault. Each element can be configured according to its kind: For a hub you can define the fields of a business key, for a link you can define the link type, for a satellite you can define the satellite type and all business fields.

With this information the modeler can generate DDLs of all data vault tables and functions for the generation of hash values.

2. Installation

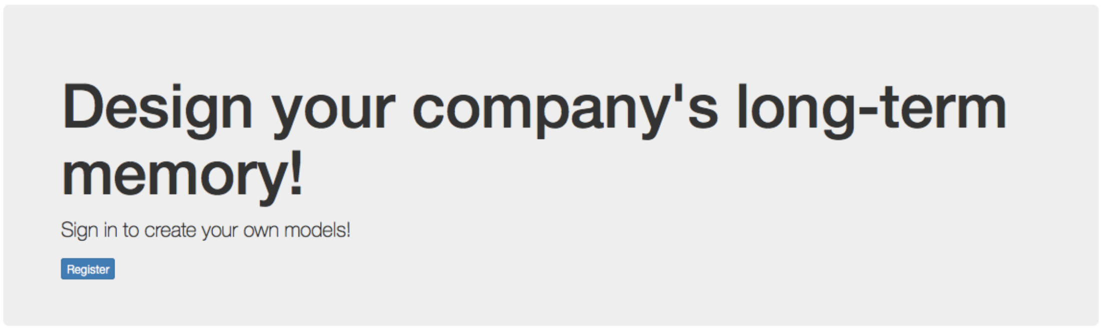
The modeler is a node express application which uses couchdb as database. The installation requires eight steps:

1. Install node.js
   Go to https://nodejs.org and follow the instructions there.
2. Install couchdb
   Follow the instructions on http://couchdb.apache.org/.
3. Database setup
    1. Create a database.
    2. Create a user that will be used for the connection from modeler to couchdb.
4. Get the modeler sources from the GitHub repository https://github.com/jon-agilebi/agile-bi.de.data-vault.modeler
5. Change to the root folder agile-bi.de.data-vault.modeler of the application. In the sub folder documents you will find three JSON-Files. These files have to be stored in the database with the following ids:
    1. user.json gets id user
    2. configuration.json gets id configuration
    3. design_vault.json gets id _design/<database name>
6. In the root folder agile-bi.de.data-vault.modeler of the application run the command **npm install**
7. Set the following environment variables:
   ADVM_ADMIN = <user id of the modeler admin>
   MODELER_PORT = <port where the modeler shall listen>
   DB_HOST= <host name of couchdb database>
   DB_PORT = <port where couchdb listens, default is 5984>
   DB_NAME= <name of the database>
   DB_USER= <name of the user created in step 3.2>
   DB_PWD= <password of the user created in step 3.2>
8. In the root folder agile-bi.de.data-vault.modeler of the application run the command **node agileVault.js**. The modeler listens now on the port defined in step 7.

In a production environment you will go to some more steps and use e.g. upstart or systemd in a linux environment for starting and stopping the application. You will find several articles in the web how you should run a node.js application in production. On www.agile-bi.de you will also find an installation script for an Ubuntu 14.04 Linux distribution and an upstart configuration file.

3. User Management

When you navigate to the modeler's main page you will find a register button.

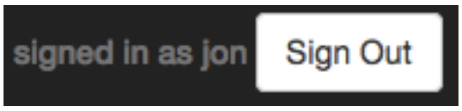


Design your company's long-term memory!

Sign in to create your own models!

Register

With this button you can open a registration form where you have to enter a user id and a password. After registration you find yourself again on the main page, but now signed in to the modeler application as you can see on the right side of the navigation bar:



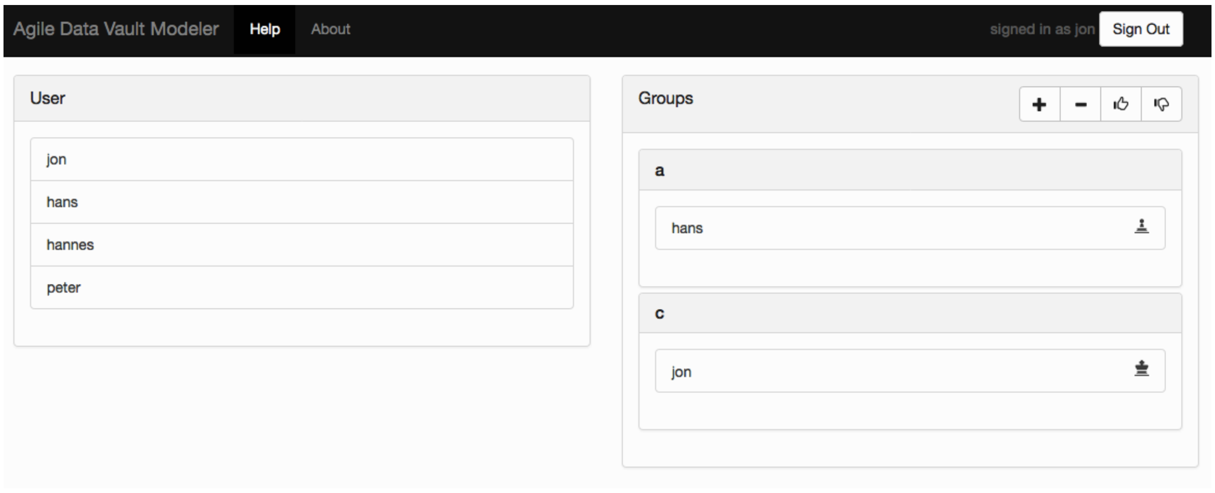
When you sign out, then you will find the required form to sign in again on the right side of the navigation bar.

After registration you have to wait until the administrator assigned you to at least one group (the communication between you and the administrator is out of scope of the modeler functionality). After this assignment and signing in again you will see all models which were created by members of one of your groups. When the administrator gave you the rights to create new models, you will also see a button „Add a new model".

This is all you have to know about user management as ordinary user.

The remainder of this section describes the possibilities of an administrator to manage user and groups. Create a new user with the same id as in the environment variable ADVM_ADMIN. This user has the administration role.

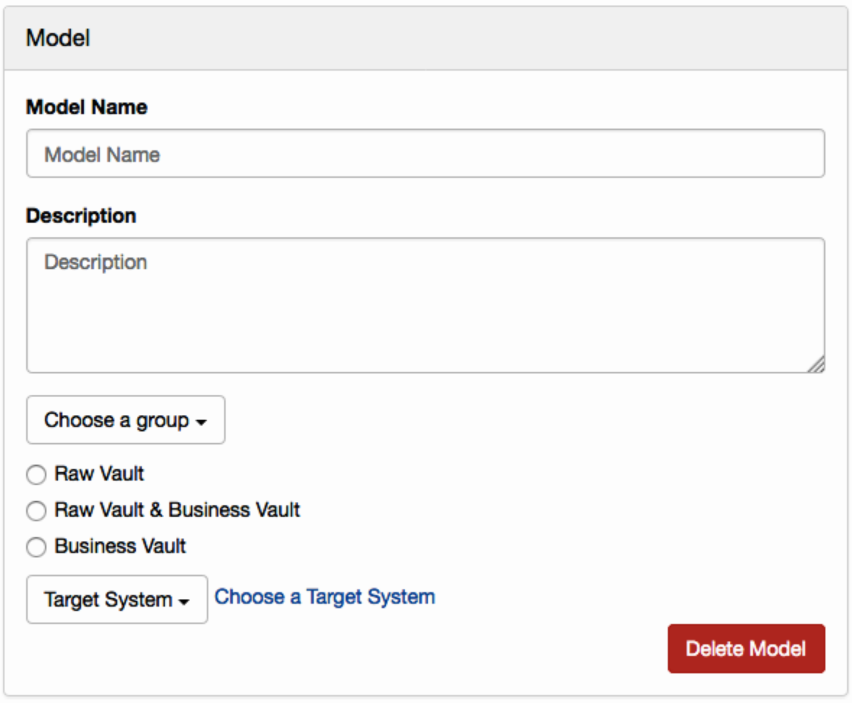
The administrator will find the link User Management in the navigation bar. Following this link the following page appears:



With the buttons + and - the administrator can create new groups or delete selected groups. After selecting a user and a group the administrator can assign the user to the group with the thumbs-up button. After selecting a user in a group he can remove him from the group with the thumbs-down button. After the assignment of a user to a group he will be displayed with a pawn icon on the right. This means, the user is an ordinary user and is not allowed to create or delete a model or to edit its properties. Clicking on the icon will change it to a king icon and the user now can fulfill these actions. A click on the king icon will make it a pawn again.

4. Create and Edit Models

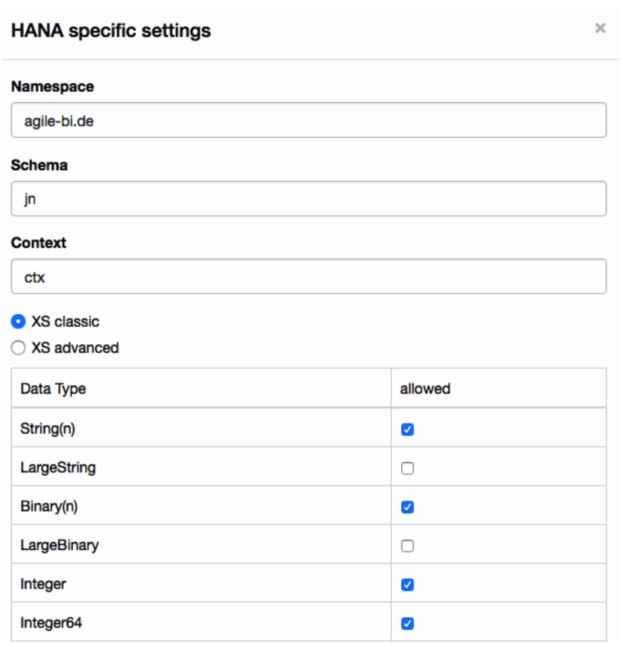
When you are allowed to create models, you will find a button **Create a new model** on the overview page. Clicking on this button will bring you to a dialogue where you can edit all properties of this model. These properties can be found in three different groups. In the first group you can define the name of the model and add a short description. Next, you choose the group the model shall belong to. In the select box you will find all groups you are a member of. Choose one of these groups. Each member of this group then can work with this model.



Furthermore, you can define whether the model represents a raw vault, a business vault or both. Then you can choose the target system. There are four options:

- Standard-SQL
- SQL Server
- Oracle
- SAP HANA

After the choice of a system, you see a link with the system name. With this link you can open a dialogue where you can set specific settings for this target system. For example, here you see a part of this dialogue for the target system SAP HANA:


HANA specific settings ×

Namespace
agile-bi.de

Schema
jn

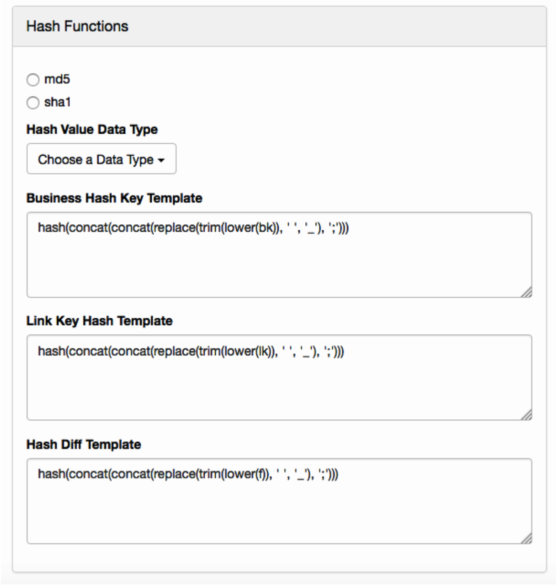
Context
ctx

[✓ XS classic]
XS advanced

Data Type	allowed
String(n)	✓
LargeString	☐
Binary(n)	✓
LargeBinary	☐
Integer	✓
Integer64	✓

Besides some specific settings you find in each of these four dialogues a list of all data types where you can select those types that are allowed in the model.

In the second group you can define all required information for the generation of the source code of functions for the creation of hash values.


Hash Functions

md5
sha1

Hash Value Data Type
Choose a Data Type ▾

Business Hash Key Template
```
hash(concat(concat(replace(trim(lower(bk)), ' ', '_'), ';')))
```

Link Key Hash Template
```
hash(concat(concat(replace(trim(lower(lk)), ' ', '_'), ';')))
```

Hash Diff Template
```
hash(concat(concat(replace(trim(lower(f)), ' ', '_'), ';')))
```

Typically, in Data Vault 2.0 there are three situations where hash values are computed:

1. For a hub a hash value for a business key is computed. The business key can contain of several fields.
2. For a link a hash value for the combination of all business keys of all connected hubs is computed.
3. For a satellite a hash value for all business attributes of the satellite is computed, so that one can easily check whether there is a change in at least one of the attribute values.

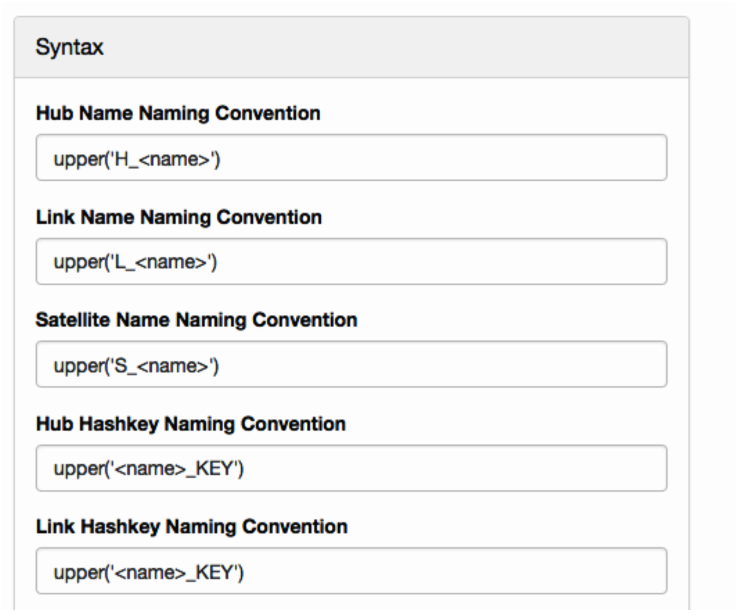
To define a function in one of these situations you have to determine at first the method for hash generation: either md5 or sha1. Furthermore, you have to define the data type of the return type.

To define the behavior of the function for each of the three above mentioned situations you can use a term whose syntax follows the following rules:

- The letter **f** represents all fields of a satellite, the expression **bk** all fields of the business key of a hub and **lk** all fields of all hubs connected to a link. Each term must have exactly one of these expressions.
- In order to transform a list of fields into a single field you can use the operator **concat(-).** In each term there must be exactly one application of this operator.
- A **literal** is an expression '-' in simple quotation marks
- **concat(-,-)** is also a binary operator, which just concatenates two strings
- The operators **trim(-)**, **lower(-)**, **upper(-)**, **replace(-,-,-)** represent the functions you will expect:
    - trim removes whitespace at the beginning and the end of a string
    - lower replaces uppercase letters with lowercase letters, upper replaces lowercase letters with uppercase ones
    - replace replaces each occurrence of the second operand in the first operand with the third operand
- The operator hash(-) represents the function that computes the hash value of a string, either by the md5 or the sha1-operator.

We will go through some examples for some function for the business key of a hub. The simplest term would be **concat(bk)**. If the business key of a product would consist of the product group 'A' and the product key '1234', we would get the string 'A123' as the output of our function. The term **hash(concat(bk))** would produce the hash value of 'A123', either by the md5 or sha1-method as defined in the configuration; hash(concat(lower(concat(bk,';')))) would produce a function with the hash value of 'a;123'.

Finally, in the third group you can define several naming conventions, field names and data types for the generic fields like the record source or the load date. For the naming conventions you can refer with <name> to the element name itself and you can also defined the functions upper(-), lower(-), trim(-). So for example, when you have a hub with the name Product and you want the hub table to have the name HUB_PRODUCT, choose the template upper(‚HUB_<name>'); if your hub table name shall be h_product, then choose as template lower(‚h_<name>'). If your element name contains the German letters ä, ö, ü or ß or some blanks, then there will be replacements to ae, oe,ue, ss and _. The hub with the name 'Vertrag Störung' will be implemented by a table HUB_VERTRAG_STOERUNG.


Syntax

Hub Name Naming Convention

upper('H_<name>')

Link Name Naming Convention

upper('L_<name>')

Satellite Name Naming Convention

upper('S_<name>')

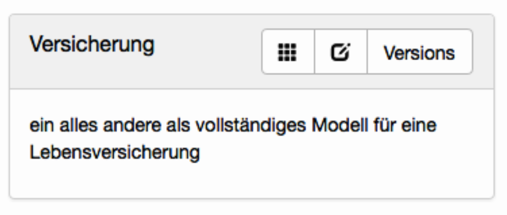
Hub Hashkey Naming Convention

upper('<name>_KEY')

Link Hashkey Naming Convention

upper('<name>_KEY')

After saving the model properties, you get back to the main page. But you will see now a new thumbnail for the new model.


Versicherung

ein alles andere als vollständiges Modell für eine Lebensversicherung

Besides the project’s title and description the thumbnail also has a small menu bar with three buttons: with the left one you can open the editor page where you can create the model elements. With the button in the middle you get back to the properties page that we described in this chapter. Finally, the right button opens a page with an overview of all created versions. There you can also create new versions.

5. Data Vault Modeling

After the creation of a model we can start to fill it with content. Therefore you have to navigate to the editor page. The modeling process consists of the following activities:

- Create or remove views
- Add elements or references to an element to the canvas
- Move and connect the elements on the canvas
- Configure the details of an element

A **model** consists of **elements** and the **connections** between elements. An element itself is defined by its name, its kind (hub, link, satellite, reference) and its configuration, which again depends on its kind, e. g. a hub has another configuration than a link.

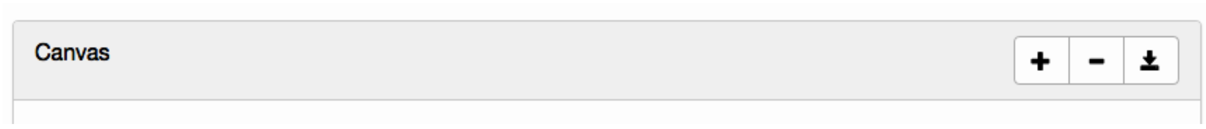
The name of an element must be unique with respect to its kind. For example, there can be a hub with the name Product and a satellite with the name Product, but no two hubs can have the same name. Connections can have a name, too. But the connection name is just a label and there are no constraints on connection names.

A **view** shows a part of the model. One could create simply one view and place all elements in it. Probably, you will run into space problems quite soon as the canvas has a limited size. It will be better to create several views and to name them according to some classification of yours. Each view has its own canvas and you get enough space to draw your model.

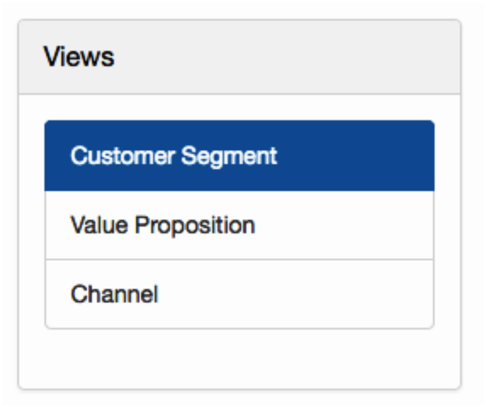
When an element is needed in a second view, then you can add a reference of this element. All connections of one element consist of all connections to the original element plus all connections to a reference of this element.

We will now have a closer look on the above mentioned activities:

Create Views



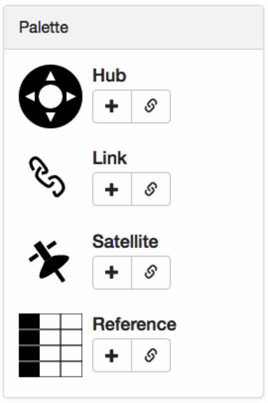
In the top right corner of the canvas you find the menu for views: add a new view, remove the selected view or save the selected view. On the left side you see all views of the model, the selected view has a blue background color.



As long as there is no view you cannot add elements to the canvas. So for the remainder of this chapter we assume that there is at least one view and a view is selected.

Add elements to the canvas

In the left lower corner you find the palette. It consists of the four element types hub, link, satellite and reference. Here you can add elements to the canvas - just click on the +-symbol or you can add the reference of an element, which has already been added to another view of the same model.

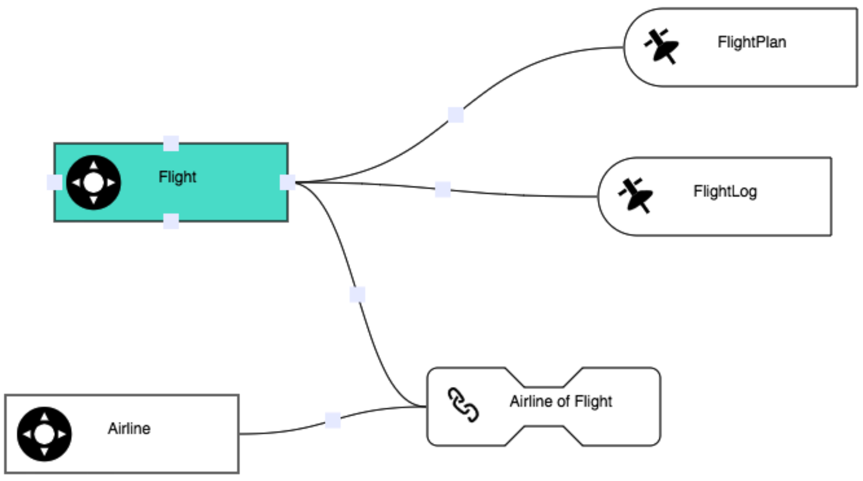


Move and connect the elements on the canvas

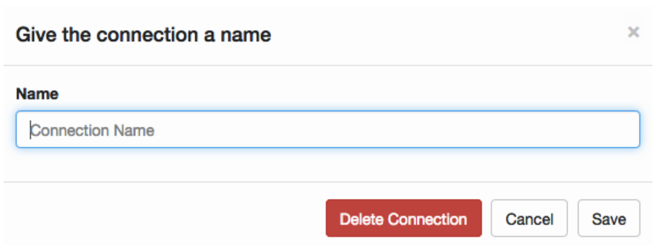
When you move the mouse pointer over an element on the canvas, it will change its background color - turquoise for an original element and gray for a reference to an element - and show four small rectangles. We call this element the **active element**. Press the left mouse button down, then you can move the element along the canvas. Release the mouse button to place the element where you want it to be. When you click on one of the four rectangles and let the mouse button be pressed down, then you can start to draw a line. Release the mouse button in another element and a connection between the two elements is created unless some constraint prevents this creation. The constraints are:

- Self-references are not allowed.
- There can be at most one connection between two elements.
- A satellite can have just one connection to a hub or a link but several to a reference table
- Two hubs cannot be directly connected.
- A reference table can only be connected to a satellite.

In the screenshot below the Flight hub is the active element. It is connected to the two satellites FlightPlan and FlightLog. It is connected to the hub Airline via the link Airline of Flight.



In the middle of each connection there is also a small quadrangle. Clicking on it, will open a dialogue.

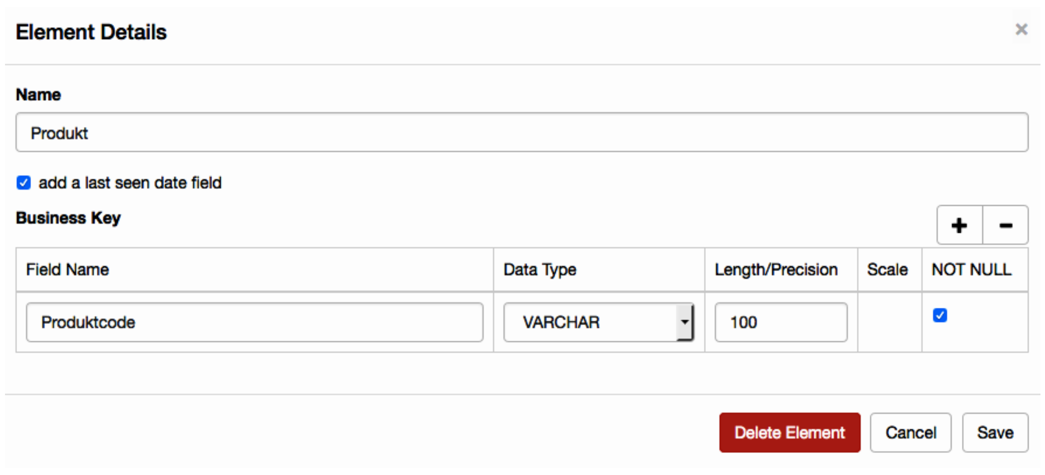


Here you can give the connection a name or you can delete the connection. One of the constraints above says that there can be at most one connection between two elements. But Visual Data Vault allows several connections between two elements. There can be e. g. two connections between a link Flight and a hub Airport, one for the origin and one for the destination. We can achieve this by giving the connection a semicolon separated list of names, Origin;Destination in the example above. In a physical instance of this model, the link would then have two referencing hash keys.

Configure the details of an element

A double-click on the active element opens a modal window, where you can define its configuration. The content of this window depends on the kind of the element. But there are also some points that have all configurations in common: You can give the element a name, you can delete the element, you can save the configuration or you can leave without saving. When the model properties say that you can decide for each element independently whether you want to have a last seen date field, then you find an appropriate checkbox.

In the following diagram you see the modal window for a hub configuration. you can define the business key which again can consist of several fields. With the +-button you can add fields, with the —button you can remove the selected field. For each field you can define its name, determine its data type and finally decide whether the field is nullable or not (in this case consider that it is very inconvenient to have a nullable part of a key).


Element Details

Name

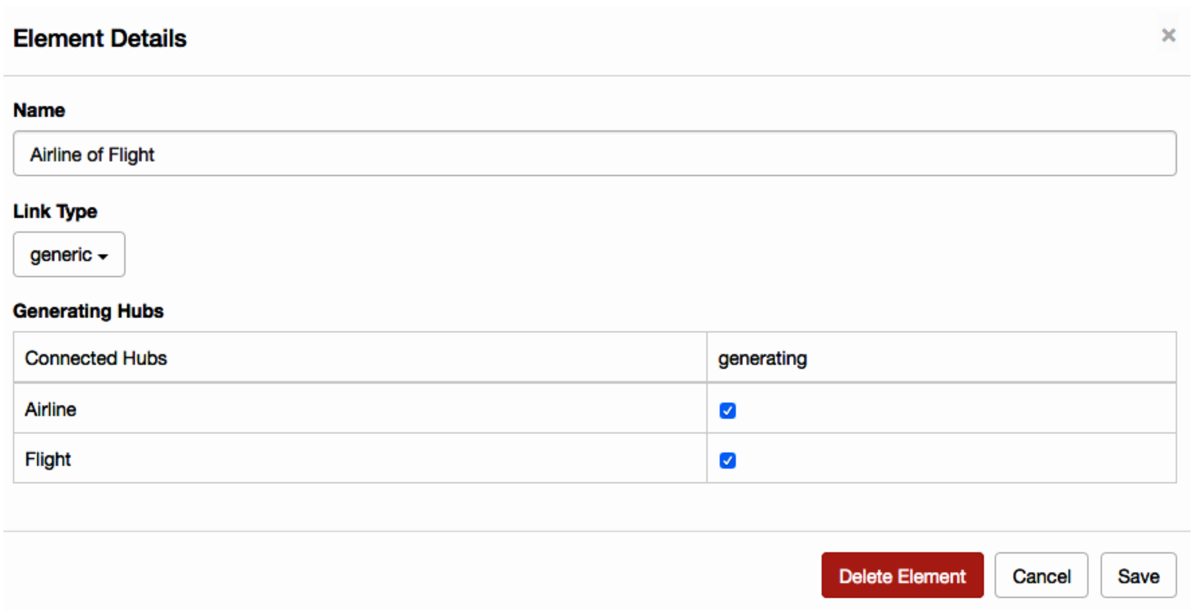
Produkt

☑ add a last seen date field

Business Key

Field Name	Data Type	Length/Precision	Scale	NOT NULL
Produktcode	VARCHAR	100		☑

Now let us have a look on the configuration of a link:


Element Details

Name

Airline of Flight

Link Type

generic

Generating Hubs

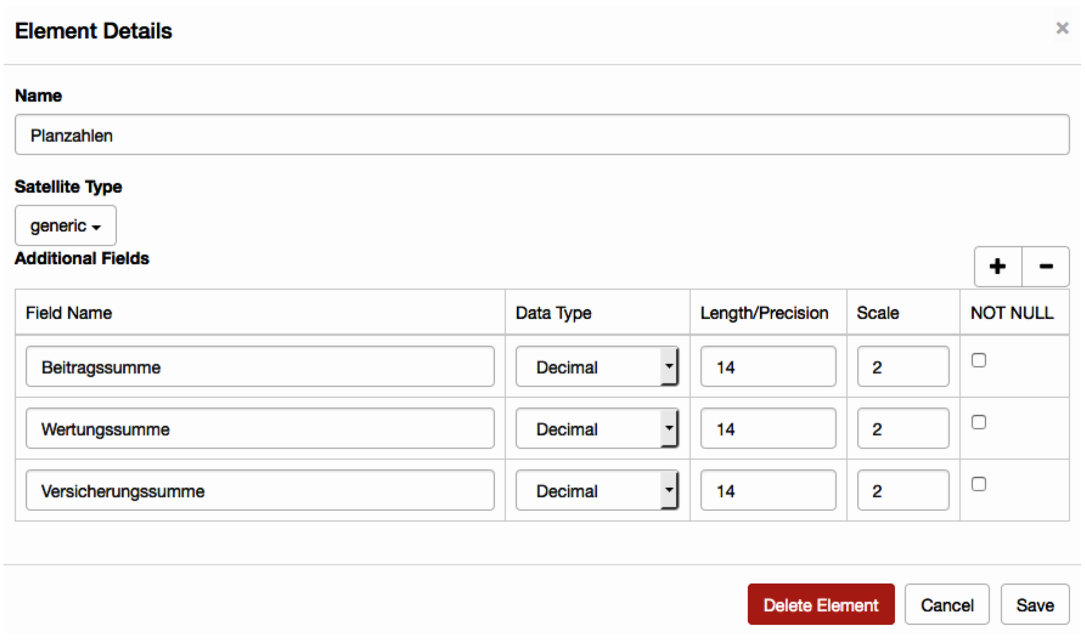
Connected Hubs	generating
Airline	☑
Flight	☑

Here you can choose the link type and decide which connected hubs are generating hubs. For a discussion of link types see [LinstedtOlschimke2016]. When a link type is generic, it just means, that it is not of a special link type. The choice of a link type does not influence the physical implementation of the link unless the type is one of the following two:

- same-as
  the table has two references (master and duplicate) to the connected hub

- hierarchical
  the table has two references (parent and child) to the connected hub

Here you see the configuration view of a satellite:



Element Details

Name

Planzahlen

Satellite Type

generic

Additional Fields

Field Name	Data Type	Length/Precision	Scale	NOT NULL
Beitragssumme	Decimal	14	2	☐
Wertungssumme	Decimal	14	2	☐
Versicherungssumme	Decimal	14	2	☐

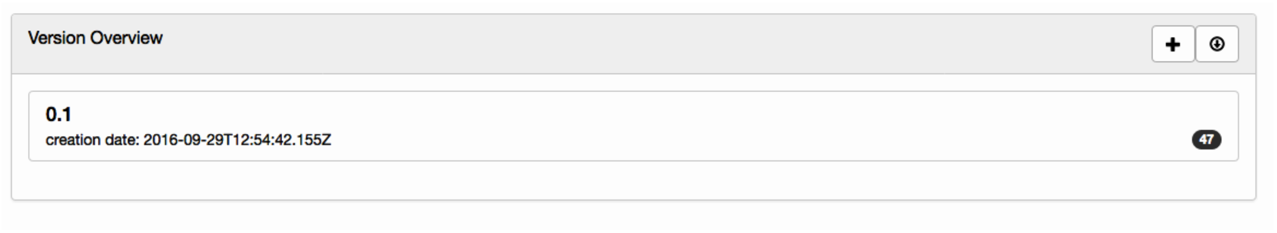
You can choose a satellite type and finally you can add, remove and edit the business fields of the satellite. We skip the configuration of a reference table, where you also can add, remove and edit the fields of the reference table.

The following features of Visual Data Vault are not implemented:

- bridge tables and PIT tables
  These tables are used for performance optimization. During a performance analysis one will find out which tables are required. As you will work with the physical implementation at that moment, it is easier to implement these tables immediately without the reference to a model.
- Several link and satellite types can be chosen but there is no effect on the structure of the satellite or link. This might change in future releases of the modeler.

6. Create Releases

It is quite simple to generate a release of a model. Click on the Versions-button of a model. Then a page with a version overview appears. In the example below just one version with the label 0.1 has already been generated. the icon on the right tells you how many elements the model has (47 in the example below).


Version Overview

0.1
creation date: 2016-09-29T12:54:42.155Z

To generate a new version just click on the +-button. Enter a version name and click on the Save-button. In the background the modeler generates a new version and the version list will get a new entry. With the download button you can download a selected version. The content of the release depends on the chosen target system:

- Standard SQL
  DDLs for all tables in one file
- Oracle & Microsoft SQLServer
  DDLs for all tables and functions for hash values in one file
- SAP HANA
  DDLs for all tables in one file and for each function for hash values a separate file, all together in a zip file

7. References

[Hahne 2014] Hahne, M., Modellierung von Business-Intelligence-Systemen, dpunktverlag, 2014.
[Hultgren 2012] Hultgren, H., Modelling the Agile Data Warehouse with Data Vault, New Hamilton Press, 2012.
[Linstedt 2010] Linstedt, D.: Super Charge Your Data Warehouse. Dan Linstedt, 2010.
[LinstedtOlschimke2016] Linstest, D., Olschimke, M., Building a Scalable Data Warehouse with Data Vault 2.0, Morgan Kaufmann Publishers, 2016.