Data Vault Pipeline Description - Introduction and orientiation

Data Vault Pipeline Description is a concept and syntax to declare all necessary metadata for fetching, parsing, hashing a data source and load it into a data vault model.

It provides a common interface between all steps and tools, that are needed during the design and implementation of data vault models and loading processes.

Even though it is generally an easy to use concept, the description of all provided elements and how to apply them needs some amount of words. Therefore the documentation is distributed over multiple articles.

This articles provides the overview, what you will find here.

PLEASE NOTE: **Knowledge about Data Vault modelling and loading procedures is essential to understand the concept.**

Please consult approriate literature, to learn Data Vault first.

The main articles

DVPD The Concept

The main article about the concept

- motivation of the concept
- Data Vault model and load requirements covered by the concept
- design decisions that have been made and are driving the syntax
- concepts behind the "non obvious purpose" syntax elements
- benefits of the concept for projects / consultants / tool developers

Data Vault method coverage and syntax examples

Provides examples for Data Vault models, especially all model elements from the book "Building A Scalable Data Warehouse With Data Vault 2.0" by Dan Linstedt and Michael Olschimke from 2016. This is to prove the completeness of the concept.

Besides from that, this is a good entry point to understand the core syntax by looking at examples.

Reference of code syntax elements

Defines and explains the DVPD core syntax and structure.

This article is highly formalized. The order of elements is not always supporting a learning process.

Reference of model profile syntax

Defines and explains syntax and structure of the model profile syntax.

Additional appliance descriptions

DVPD development workflows

Descibes different project scenarios, how the DVPD should be integrated in the development workflow.

Model and method investigations

Some requirements of the Data Vault method are not directly described in the books, but need to be discovered as imlplicit conclusions or from project experiences. The following articles are investigating different aspects in the data vault method and explain the requirements that lead to some syntax concepts and design descisions.

Model topologies and basic field mapping variations

This article provides a complete set of model topologies. The described topologies are an aggregation and combination of the basic patterns, described by the Data Vault modelling method.

Also it explains the basic variations how fields of the source record can be mapped.

Understanding both aspects is neessary to prove the completeness of the syntax via test cases.

Catalog of field mappings in relations

Loading relations to the data vault model is getting complex, when the same hub, link or satellite must be loaded more then once for a single record (e.g. when loading hierachical link structures). This article investigates the scenarios and provides all the background needed to understand the relation concept in DVPD.

It also is one of the main drivers of testcases.

Deletion detection catalog

Deletion detection is a complex problem on its own. The full investigation about all possible variationts of deletion detection and how to describe it in a syntax is placed in this document to shorten the central concept article.

Implementation of reference compiler and example rendering scripts

To proof the validity of the concept, this git procject contains a reference implementation of a DVDP Compiler and examples for scripts to render various code and documentation files from the DVPD or its compiled version, the DVPI.

Reference implementaion installation and usage

Instructions how to install and use all the provided scripts.

DVPDC automated testing

To proof, that extentions and changes to the compiler do not have any sideeffects to unchanged functionality, a large set of automated test cases is provided in the reposotory. How this works and can be extended, is documented here.