Conclusions and Observations about Data Vault models and loading

(C) Matthias Wegner, cimt ag

The rules, defined by the Data Vault concept itself, allow various kinds of models but also imply constraints, that are not explicitly formulated

Conclusions and obeserveraions about these variations and constraints are taken into account, when designing the DVPD syntax.

This article explains these aspects, to provide a better understanding about the decisions of the syntax design.

Deletion detection variations

Deletion detection is a complex problem. See Deletion detection catalog for a full investigation about all possible variations of deletion detection and how to describe it in a syntax
The document contains explanations about: partitioned deletion detection

Driving keys for a satellite must be applied by all loading processes

From the perspective of the data consumer, it es expected, that the satellite provides a consistent representation of the relation, it stands for. When the satellites contains a 1:n relation, managed through driving keys, this must be true for all elements. Therefore, when multiple loading processes write to the same link satellite, all processes must apply the same driving key rule. To help, keeping consistency between different DVPD the driving key directive is part of the satellite table definition (good chance of copy/paste repitition)

Why do we discuss this so intense:

The procedure to manage a driving key constraint might look like a partitioned deletion detection, where the driving key is the partition criteria. But seeing the driving key deletion as a variation of deletion detection means, that the appliance of it is source specific and that there could be sources, which would explicitly provide the setting of the new and removal of the old relation. This is not the case. When driving key logic is applied, the source is providing the change of the relation by setting the foreign key value in the record to the new target. This implicitly ends the old relation. The need for the extra deletion operation in the satellite comes from the data vault mechanics, not from the source delivery behaviour.

preseving order of multitactive data

For multi active data it might be necessary to preserve the order of the data in the transmission format, since the might be a meaning to it.

To preserve the order of multiactive data in the satellite, when the incoming data sets have no field with the order, an artifical field with the "row number" must be added during the parsing.

This lead to the syntax of the ${ROW_NUMBER...} placeholders for the "field_value" parameter.

historization patterns for multi active data

Depending on the final pattern, how to separate the data versions, when reading the satellite, the loading process needs additional declarations beside the normal satellite loading parameters. The following options of historical data separations for multiactive satellites are possible:

- every value set/diff hash for a key is enddated individually.
- every diff hash for key is evaluated and managed individually during retrieval.
- subkey evaluation and insertion/deletion
- the full group of value sets of a changing load for a key have the same load date

Individual enddating

With individual enddating, determining the valid data for a specific point in time only needs to check the validity interval.

```
hub key| load date |end date  |diff|  content 1    | content 2

9a78raf| 2023-05-01|2999-09-09|qw2j| 1st delivery | still in the set
9a78raf| 2023-05-01|2023-05-05|k301| 1st delivery | gets changed in 5th delivery
9a78raf| 2023-05-05|2023-05-08|f298| 1st delivery | gets removed in 8th delivery
9a78raf| 2023-05-05|2999-09-09|asd9| 5th delivery | still in the set

gfo1721| 2023-05-07|2023-05-10|8faj| 7th delivery | gets totally removed in 10th delivery
```

The loading procedure needs an efficient way to

- insert unknown data. This can be an additional value set or an additional repition of an already known value set
- enddate missing: This can be a value set, that is not in the source ata anymore, or with less repititions

Only the usual satellite parameters are needed to run this procedure. An enddate must be configured.

Individual diff hash insertion

Without an enddate the valid data for a specific point in time is determined by following the chain of load dates for every diff hash and excluding deletion flagged rows.

```
hub key| load date |deleted   |diff|  content 1   | content 2

9a78raf| 2023-05-01|false     |qw2j| 1st delivery | still in the set
9a78raf| 2023-05-01|false     |f298| 1st delivery | removed in 8th delivery
9a78raf| 2023-05-08|true      |k301| #missing#    | #missing#
9a78raf| 2023-05-05|false     |asd9| 5th delivery | still in the set

gfo1721| 2023-05-07|false     |8faj| 7th delivery | gets totally removed in 10th delivery
gfo1721| 2023-05-10|true      |8faj| #missing#    | #missing#
```

The loading procedure needs an efficient way to

- insert unknown data. This can be an additional diff hash or an additional repition of an already known diff hash
- insert deletion record for missing: This can be a diff hash, that is not in the source data anymore, or with less repititions

Only the usual satellite properties are needed to run this procedure. A diff hash must be configured.

subkey evaluation and insertion/deletion

This pattern will only be available on data sources, where you can define a subkey, that will create uniqueness between rows of the same key.

Without an enddate the valid data for a specific point in time is determined by following the chain of load dates for every **sub key in a key of the satellite** and excluding deletion flagged rows. This needs knowledge about the subkey columns while quering the data. How this is documented depends on the project.

```
hub key| load date |deleted |diff|subkey|  content 1   | content 2
9a78raf| 2023-05-01|false   |qw2j| 1.1  | 1st delivery | still in the set
9a78raf| 2023-05-01|false   |k301| 1.2  | 1st delivery | gets changed in 5th delivery
9a78raf| 2023-05-05|false   |f298| 1.2  | 1st delivery | gets removed in 8th delivery
9a78raf| 2023-05-08|true    |----| 1.2  | #missing#    | #missing#
9a78raf| 2023-05-05|false   |asd9| 2.1  | 5th delivery | still in the set

gfo1721| 2023-05-07|false   |8faj| 3.1  | 7th delivery | gets totally removed in 10th delivery
gfo1721| 2023-05-10|true    |----| 3.1  | #missing#    | #missing#
```

The loading procedure needs an efficient way to

- insert unknown data. This can be an additional subkey, or changed diff hash for a known subkey
- insert deletion record for missing, when a subkey isn't delivered any more

For this procedure to work, the fields, that belong to the subkey in the satellite must be declared. This done with to the "is_muli_active_key" syntax in the target table mapping.

Beside this special handling during loading, the same information about the role as key is needed during retreival of the data, since it is needed to determin, wich row belong to actual data and wicht to historized data.

full group insertion

Without an enddate the valid data for a specific point in time is determined by following the chain of distinct load dates for the **key of the sattellite** and excluding deletion flagged rows. All valid rows in the interval will have the same loaddate.

```
hub key| load date |deleted   |diff|  content 1   | content 2

9a78raf| 2023-05-01|false     |qw2j| 1st delivery | still in the set
9a78raf| 2023-05-01|false     |k301| 1st delivery | gets changed in 5th delivery
9a78raf| 2023-05-05|false     |qw2j| 1st delivery | still in the set
9a78raf| 2023-05-05|false     |f298| 1st delivery | gets removed in 8th delivery
9a78raf| 2023-05-05|false     |asd9| 5th delivery | still in the set
9a78raf| 2023-05-08|false     |qw2j| 1st delivery | still in the set
9a78raf| 2023-05-08|false     |asd9| 5th delivery | still in the set

gfo1721| 2023-05-07|false     |8faj| 7th delivery | gets totally removed in 10th delivery
gfo1721| 2023-05-10|true      |----| #missing#    | #missing#
```

To achieve this pattern all previous methods to determine a change in the rows for a specific satellite key can be used with the modification, that all data for the satellite key with at least one changed row is inserted.

full group insertion with group diff hash

The full group insertion pattern can also be achieved by using a group diff hash load pattern.

- calculate a group hash for every incoming satellite key, by concatenating all rows of the same satellite key
- use the group hash as diff hash for loading
- insert the whole group, when the last group hash in the satellite differs from the group hash in stage for a key

Resulting satellite will only differ in the pattern of the diff hashes.

```
hub key| load date |deleted   |diff|  content 1   | content 2

9a78raf| 2023-05-01|false     |e1r1| 1st delivery | still in the set
9a78raf| 2023-05-01|false     |e1r1| 1st delivery | gets changed in 5th delivery
9a78raf| 2023-05-05|false     |h235| 1st delivery | still in the set
9a78raf| 2023-05-05|false     |h235| 1st delivery | gets removed in 8th delivery
9a78raf| 2023-05-05|false     |h235| 5th delivery | still in the set
9a78raf| 2023-05-08|false     |whsu| 1st delivery | still in the set
9a78raf| 2023-05-08|false     |whsu| 5th delivery | still in the set

gfo1721| 2023-05-07|false     |8faj| 7th delivery | gets totally removed in 10th delivery
gfo1721| 2023-05-10|true      |----| #missing#    | #missing#
```

To keep the group hashes constant, when the incoming data has different row order for every delivery, an artifical orderng must be implemented. This is supported by the syntax elements: "prio_for_row_order" and "row_ordr_direction".

more documents about model variations

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.