Partition deletion detection catalog

Deletion detection by comparing data vault against the complete source data might become very expensive for large datasets or is even impossible, when the source only delivers partitions of its data on the interface.

In such cases a partitioned deletion detection will provide a way to get a correct image of the source data in the data vault.

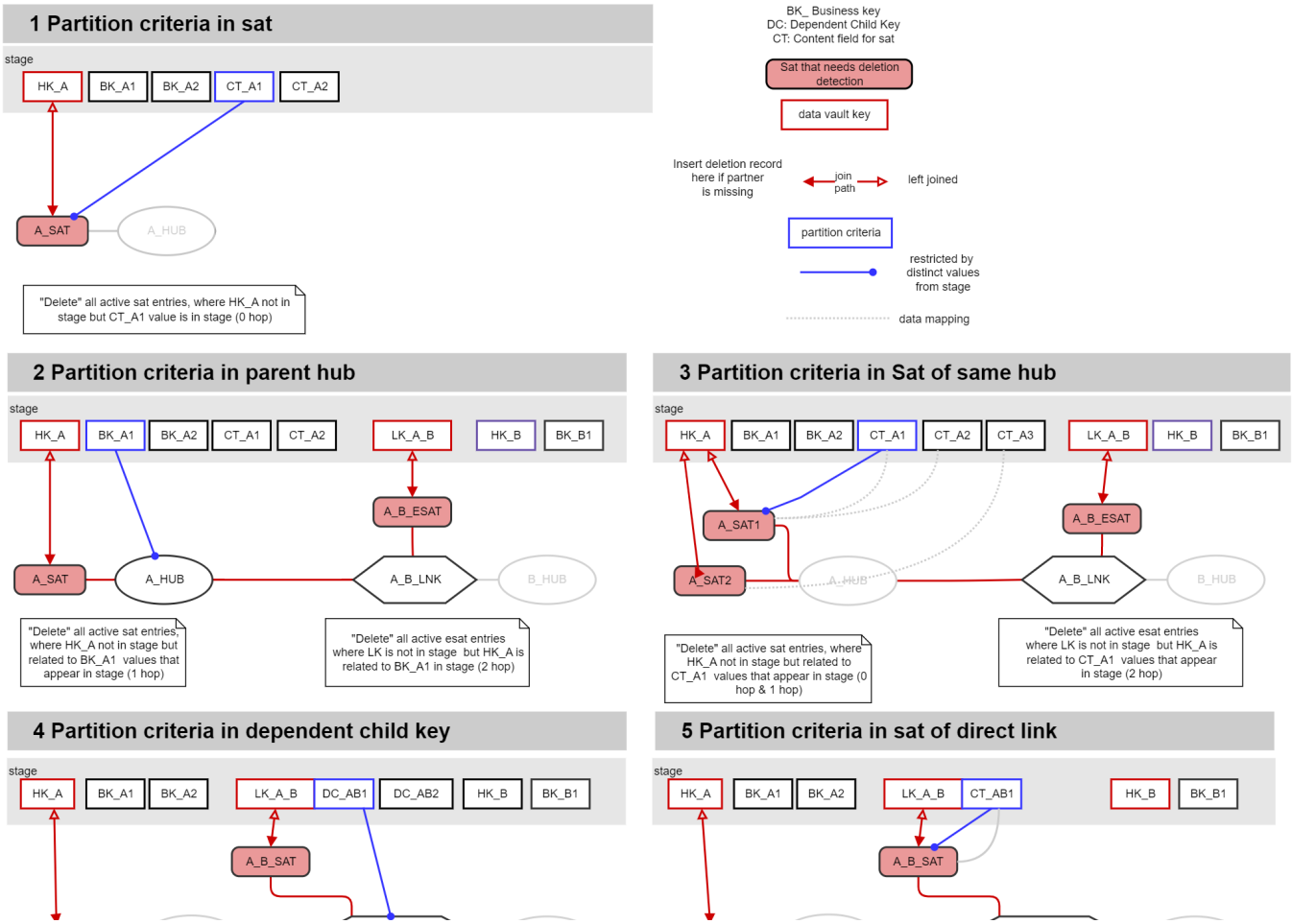
The basic prerequisits for using a partitioned deletion detection are:

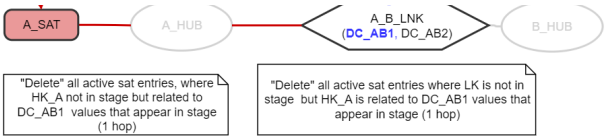
- partitions are identified by values in fields, that are stored in the data vault model
- the incoming increment of data contains all rows for all partitions, that are mentioned in the increment (or in pseudo sql language: The increment has always a complete set of rows for all values for SELECT DISTICT partitioning columns FROM increment)

The procedure for a single satellite is as follows:

- select all active "keys" from the satellite that belong to the partition
    - Join to the data vault tables, that contain the partitioning fields
    - constrain the partitioning field only to values that are in the stage table
- create deletion records for all "keys" of teh target satellit, that are missing in the stage
- apply enddating (when used)

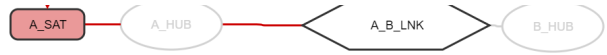
Selecting all active keys that belong to the partition can be a complex task, depending on the distribution of the partitioning columns in the data vault model. The following picture shows examples of common scenarios





"Delete" all active sat entries, where HK_A not in stage but related to DC_AB1 values that appear in stage (1 hop)

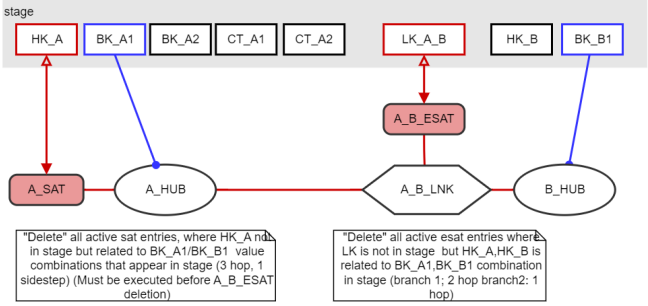
"Delete" all active sat entries where LK is not in stage but HK_A is related to DC_AB1 values that appear in stage (1 hop)



"Delete" all active sat entries, where HK_A not in stage but related to CT_AB1 values that appear in stage (2 hop)

"Delete" all active sat entries where LK is not in stage but LK_A_B is related to CT_AB1 (0 hop)

6 Partition criteria distributed

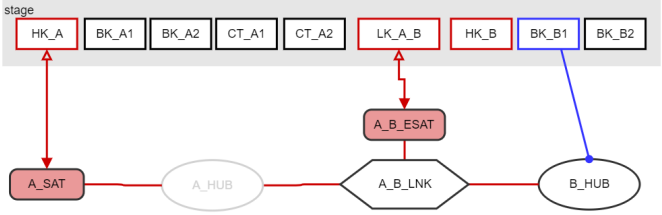


"Delete" all active sat entries, where HK_A not in stage but related to BK_A1/BK_B1 value combinations that appear in stage (3 hop, 1 sidestep) (Must be executed before A_B_ESAT deletion)

"Delete" all active esat entries where LK is not in stage but HK_A,HK_B is related to BK_A1,BK_B1 combination in stage (branch 1; 2 hop branch2: 1 hop)

XL Variations demonstrate the possible depth of complexity

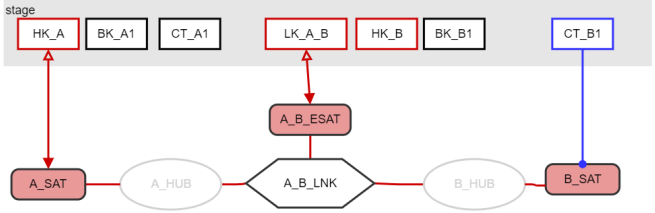
XL1 Partition criteria in linked hub



"Delete" all active sat entries, where HK_A not in stage but related to BK_B1 in stage via A_B_LNK and B_HUB (2 hop, 1 sidestep) (Must be executed bfore A_B_ESAT deletion)

"Delete" all active esat entries where LK is not in stage but HK_B is related to BK_B1 in stage (2 hop)

XL2 Partition criteria in sat of linked hub



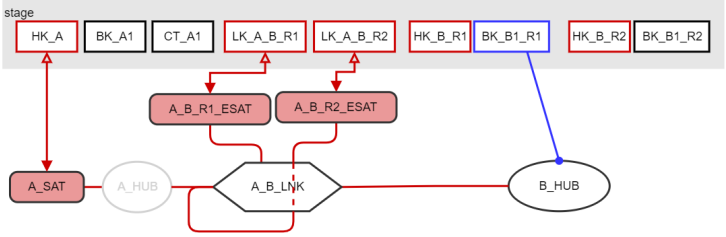
"Delete" all active sat entries, where HK_A not in stage but related to BK_A1 in stage (2 hop, 1 sidestep) (Must be executed before A_B_ESAT and B_SAT deletion)

"Delete" all active esat entries where LK is not in stage but HK_A is related to BK_A1 in stage (2 hop) (Must be executed before B_SAT deletion)

"Delete" all active sat entries where HK is not in stage but related to CT_B1 (0 hop)

XL3 Linked hub with 2 relations, only 1 relation is partioning

eg. orders of all customers of a specific country(=partitions) , where the paying customer is a second relation and might reside in another country



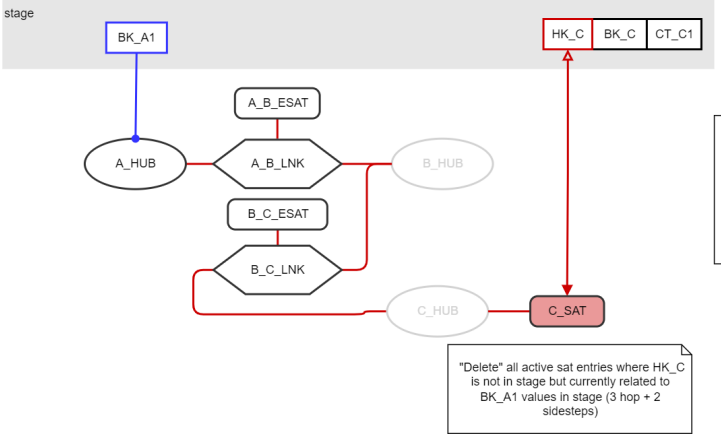
"Delete" all active sat entries, where HK_A not in stage but related to BK_B1_R1 in stage via A_B_LNK and A_B_R1_ESAT (2 hop)

R1: "Delete" all active esat entries where LK is not in stage but HK_B is related to BK_B1 in stage (2 hop)

R2: "Delete" all active esat entries where LK is not in stage but HK_B, when searched for same HK_A over AB_R1_ESATIs related to BK_B1 in stage (3 hop + 1 sidestep) (Must be executed before R1 deletion)

XL4 Sat related to the hub with partioning field via 2 links

eg. orderlines of all orders from a specific subdivision of the company



Chain of joins to the partition cirteria can be arbitrary long and might not be derivable from the relations in the model (e.g 2 esat on same link, 2 link between same hubs)

"Delete" all active sat entries where HK_C is not in stage but currently related to BK_A1 values in stage (3 hop + 2 sidesteps)

Scenario discussions

For naming conventing in these examples see appendix

1 Partition criteria in satellite

Needed Tables: only A_SAT

SQL to determine keys in partition:

```
SELECT HK_A
FROM A_SAT
WHERE CT_A1 IN
   (SELECT DISTINCT CT_A1
    FROM  STAGE_TABLE )
    AND A_SAT.META_VALID_BEFORE=far_future_date()
    AND NOT A_SAT.META_IS_DELETED
```

2 Partition criteria in parent hub

A_SAT Deletion

Needed Tables: A_SAT and A_HUB

SQL to determine A_SAT keys in partition:

```
SELECT HK_A
FROM A_SAT
JOIN A_HUB ON A_HUB.HK_A = A_SAT.HK_A
WHERE BK_A1 IN
   (SELECT DISTINCT BK_A1
    FROM  STAGE_TABLE )
   AND A_SAT.META_VALID_BEFORE=far_future_date()
   AND NOT A_SAT.META_IS_DELETED
```

A_B_ESAT Deletion

Needed Tables: A_B_ESAT, A_B_LNK and A_HUB

SQL to determine A_B_ESAT keys in partition:

```
SELECT LK_A_B
FROM A_B_ESAT
JOIN A_B_LNK USING (LK_A_B)
JOIN A_HUB USING (HK_A)
WHERE BK_A1 IN
```

```
   (SELECT DISTINCT BK_A1
    FROM  STAGE_TABLE )
   AND A_B_ESAT.META_VALID_BEFORE=far_future_date()
   AND NOT A_B_ESAT.META_IS_DELETED
```

3 Partition criteria in sat of same hub

A_SAT2 Deletion

Needed Tables: A_SAT2 and A_SAT1

SQL to determine A_SAT2 keys in partition:

```
SELECT HK_A
FROM A_SAT2
JOIN A_SAT1 ON A_SAT1.HK_A = A_SAT2.HK_A
           AND A_SAT1.META_VALID_BEFORE=far_future_date()
           AND NOT A_SAT1.META_IS_DELETED
WHERE CT_A1 IN
   (SELECT DISTINCT CT_A1
    FROM  STAGE_TABLE )
   AND A_SAT2.META_VALID_BEFORE=far_future_date()
   AND NOT A_SAT2.META_IS_DELETED
```

A_B_ESAT Deletion

Needed Tables: A_B_ESAT, A_B_LNK and A_SAT1

SQL to determine A_B_ESAT keys in partition:

```
SELECT LK_A_B
FROM A_B_ESAT
JOIN A_B_LNK USING (LK_A_B)
JOIN A_SAT1 ON  A_SAT1.HK_A = A_B_LNK.HK_A
           AND A_SAT1.META_VALID_BEFORE=far_future_date()
           AND NOT A_SAT1.META_IS_DELETED
WHERE CT_A1 IN
   (SELECT DISTINCT CT_A1
    FROM  STAGE_TABLE )
   AND A_B_ESAT.META_VALID_BEFORE=far_future_date()
   AND NOT A_B_ESAT.META_IS_DELETED
```

4 Partition criteria in dependent child key

A_SAT Deletion

Needed Tables: A_SAT, A_B_LNK and A_B_ESAT

SQL to determine A_SAT2 keys in partition:

```
SELECT HK_A
FROM A_SAT
JOIN A_B_LNK ON A_B_LNK.HK_A = A_SAT.HK_A
JOIN A_B_ESAT ON A_B_ESAT.LK_A_B = A_B_LNK.LK_A_B
           AND A_B_ESAT.META_VALID_BEFORE=far_future_date()
           AND NOT A_B_ESAT.META_IS_DELETED
WHERE DC_AB1 IN
   (SELECT DISTINCT DC_AB1
    FROM  STAGE_TABLE )
   AND A_SAT.META_VALID_BEFORE=far_future_date()
   AND NOT A_SAT.META_IS_DELETED
```

A_B_ESAT Deletion

Needed Tables: A_B_ESAT, A_B_LNK

SQL to determine A_B_ESAT keys in partition:

```
SELECT LK_A_B
FROM A_B_ESAT
JOIN A_B_LNK USING (LK_A_B)
WHERE DC_AB1 IN
    (SELECT DISTINCT DC_AB1
    FROM  STAGE_TABLE )
   AND A_B_ESAT.META_VALID_BEFORE=far_future_date()
   AND NOT A_B_ESAT.META_IS_DELETED
```

to be continued...

Appendix

Modelling convention

Satellites are enddated. The currently active rows are marked by setting the enddate column to a constant value in the far future.

Naming convention

- Data Vault table stereotype is provided as postfix
- Names of hash keys consist of a prefix followed by the table the hash belongs
    - HK_ prefix indicates the hash key of a hub
    - LK_ prefix indicates the hash key of a link
- table names of satellites begin with the table name of their parent
- table names of links contain all table names of the hubs, the link is connecting