Dani Schnider

Wieviel Zeit braucht mein Data Warehouse?

Callista

About me

Dani Schnider

- Working for Callista
- Oracle ACE Director
- Member of DDVUG
- Hobby: Craft Beer Brewing

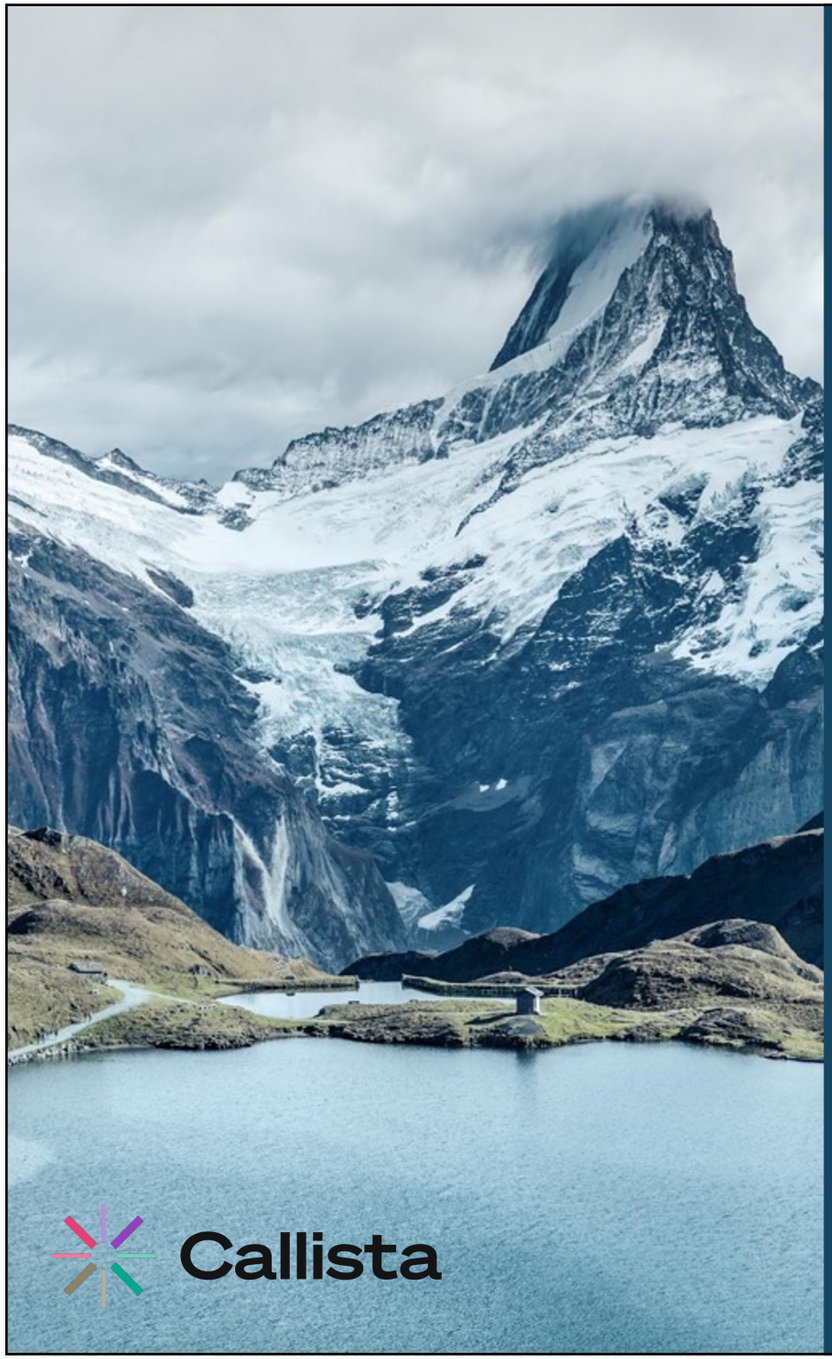
Callista

Oracle ACE Director

Deutschsprachige Data Vault User Group e.V.

MONSTER BRÄU

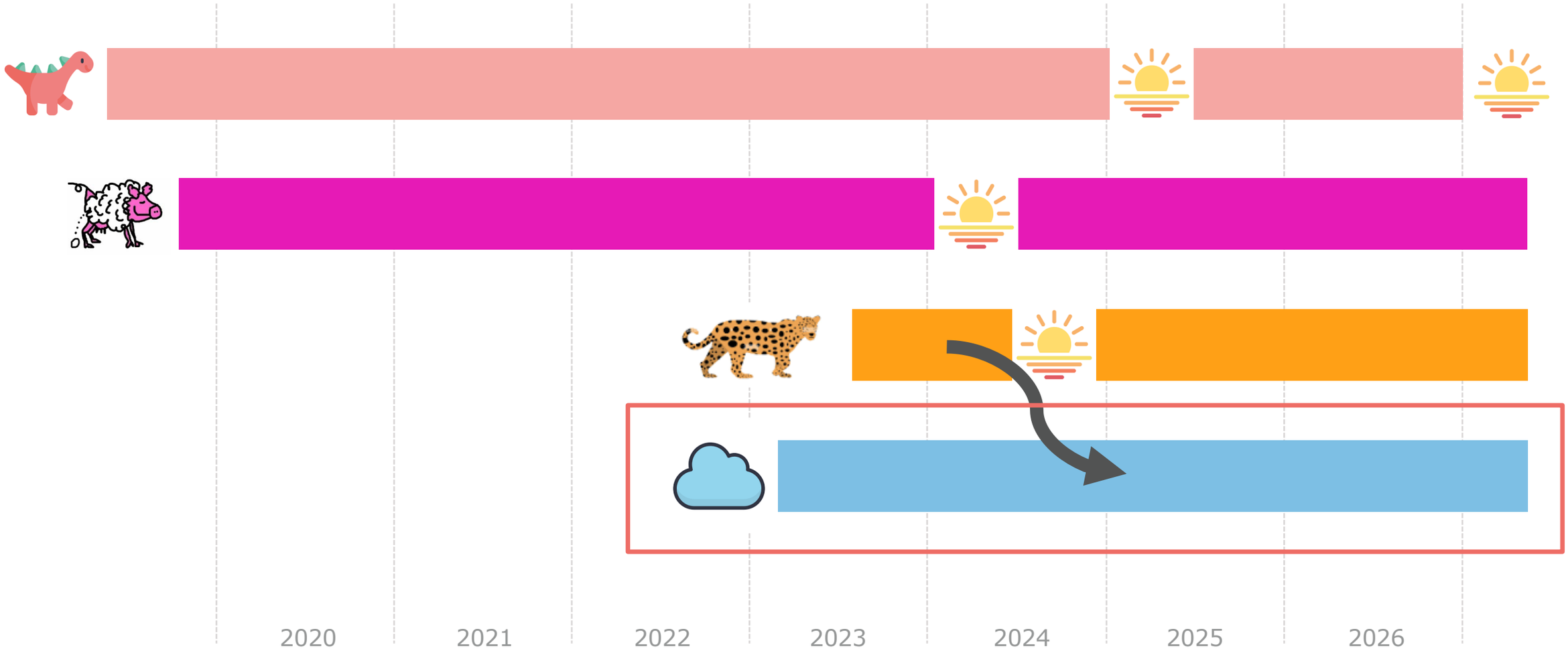
@danischnider.bsky.social danischnider.wordpress.com www.linkedin.com/in/danischnider/



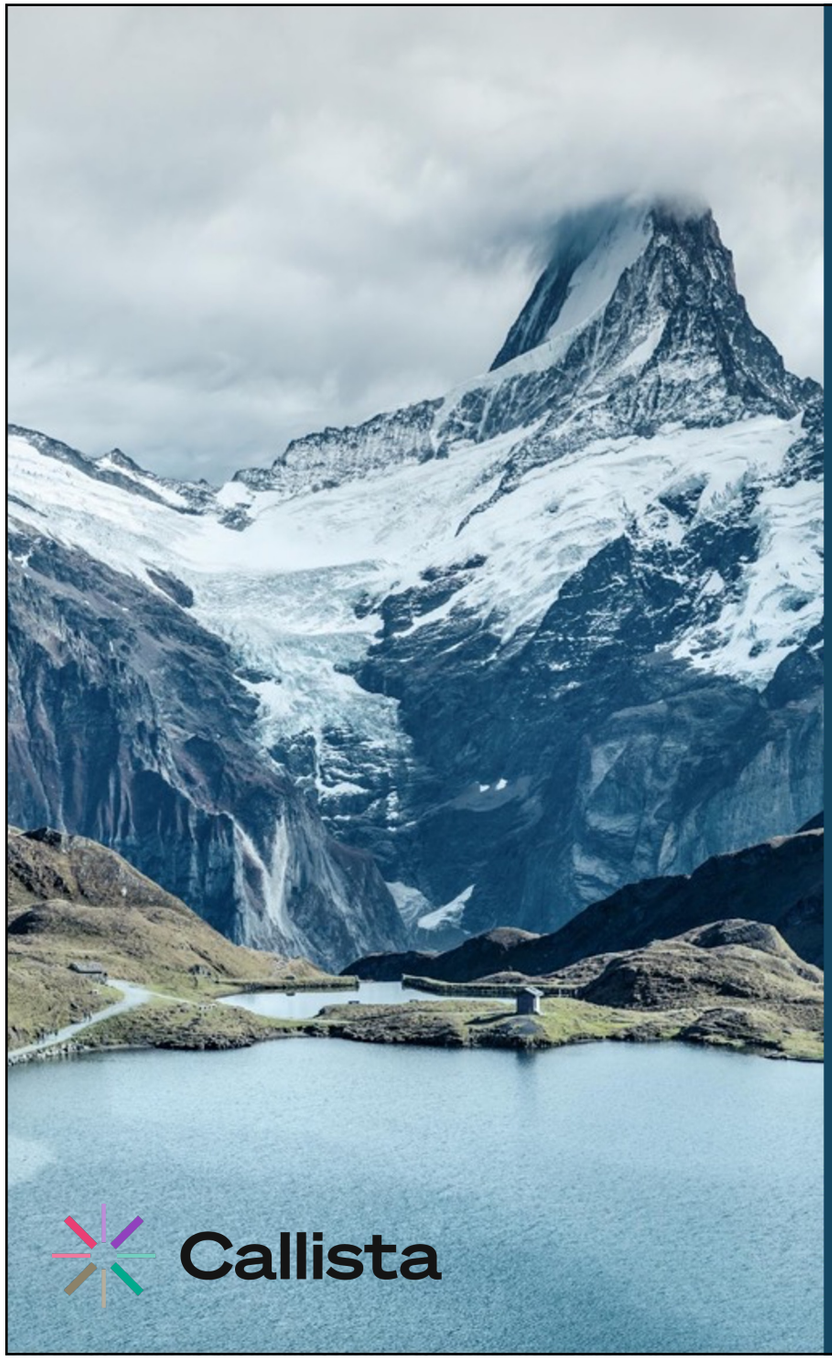
Callista

Ausgangslage

DWH Timeline



Callista



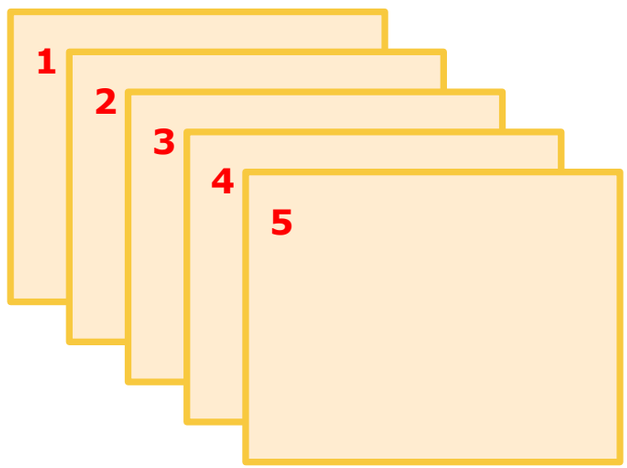
Callista

Historisierungs-konzepte

Historisierung Ladezeitpunkt

A

	HASH_KEY	LOAD_TIMESTAMP
1	A13CC05A0103051D	2024-08-30T18:53:18.124
2	A13CC05A0103051D	2024-10-07T03:10:46.345
3	A13CC05A0103051D	2024-10-11T02:48:44.023
4	A13CC05A0103051D	2024-10-14T02:48:54.500
5	A13CC05A0103051D	2024-10-23T03:01:07.379



Standard in Data Warehouses:

- Historisierung über Ladezeitpunkt
- Delta-Loads von Stammdaten
- Mit jeder Änderung wird neue Version geschrieben
- Laden von Bewegungsdaten

Eine Zeitachse:

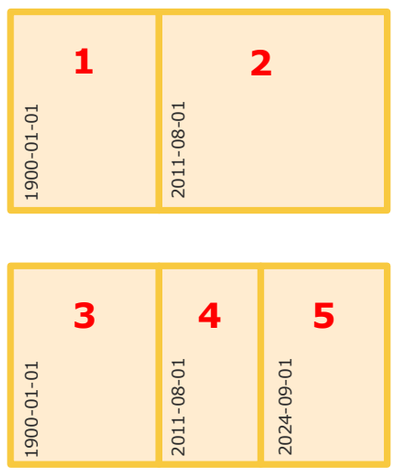
- Ladezeitpunkt

Callista

Fachliche Historisierung

B

	HASH_KEY	LOAD_TIMESTAMP	BEGIN_DATE	END_DATE
1	F4F291F16C359BCF	2024-08-30T14:57:10.330	1900-01-01	2011-07-31
2	F4F291F16C359BCF	2024-08-30T14:57:10.330	2011-08-01	2099-12-31
3	F4F291F16C359BCF	2024-09-11T02:17:13.581	1900-01-01	2011-07-31
4	F4F291F16C359BCF	2024-09-11T02:17:13.581	2011-08-01	2024-08-31
5	F4F291F16C359BCF	2024-09-11T02:17:13.581	2024-09-01	2099-12-31



Zusätzliche Anforderung:

- Fachliche Historisierung im Quellsystem
- Gültigkeitsintervalle für Stammdaten

Zwei Zeitachsen:

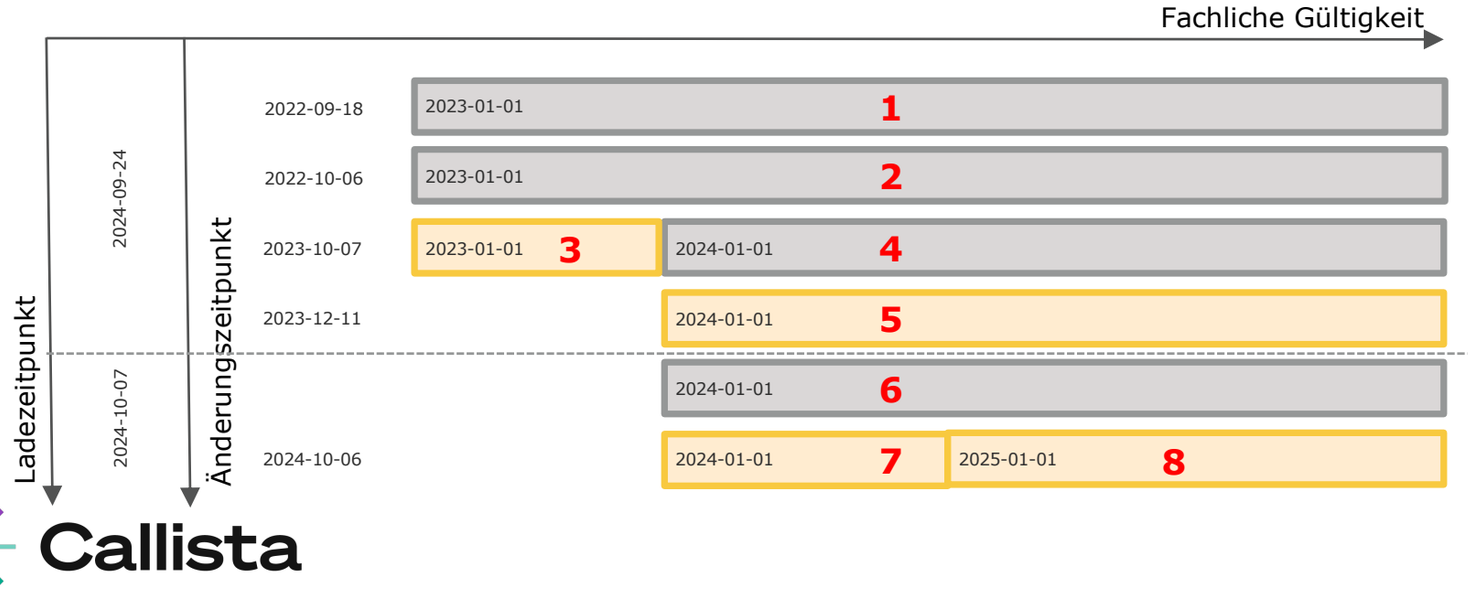
- Ladezeitpunkt
- Fachliche Gültigkeit (von/bis)

Callista

Bi-temporale Datenquelle

C

	HASH_KEY	LOAD_TIMESTAMP	BEGIN_DATE	END_DATE	CREATED_AT	REPLACED_AT
1	002272C8BE18D58B	2024-09-24T14:20:08.138	2023-01-01	2099-12-31	2022-09-18T08:43:22.9...	2022-10-06T02:39:39.1...
2	002272C8BE18D58B	2024-09-24T14:20:08.138	2023-01-01	2099-12-31	2022-10-06T02:39:39.1...	2023-10-07T10:02:24.5...
3	002272C8BE18D58B	2024-09-24T14:20:08.138	2023-01-01	2023-12-31	2023-10-07T10:02:24.5...	null
4	002272C8BE18D58B	2024-09-24T14:20:08.138	2024-01-01	2099-12-31	2023-10-07T10:02:24.5...	2023-12-11T14:41:16.0...
5	002272C8BE18D58B	2024-09-24T14:20:08.138	2024-01-01	2099-12-31	2023-12-11T14:41:16.0...	null



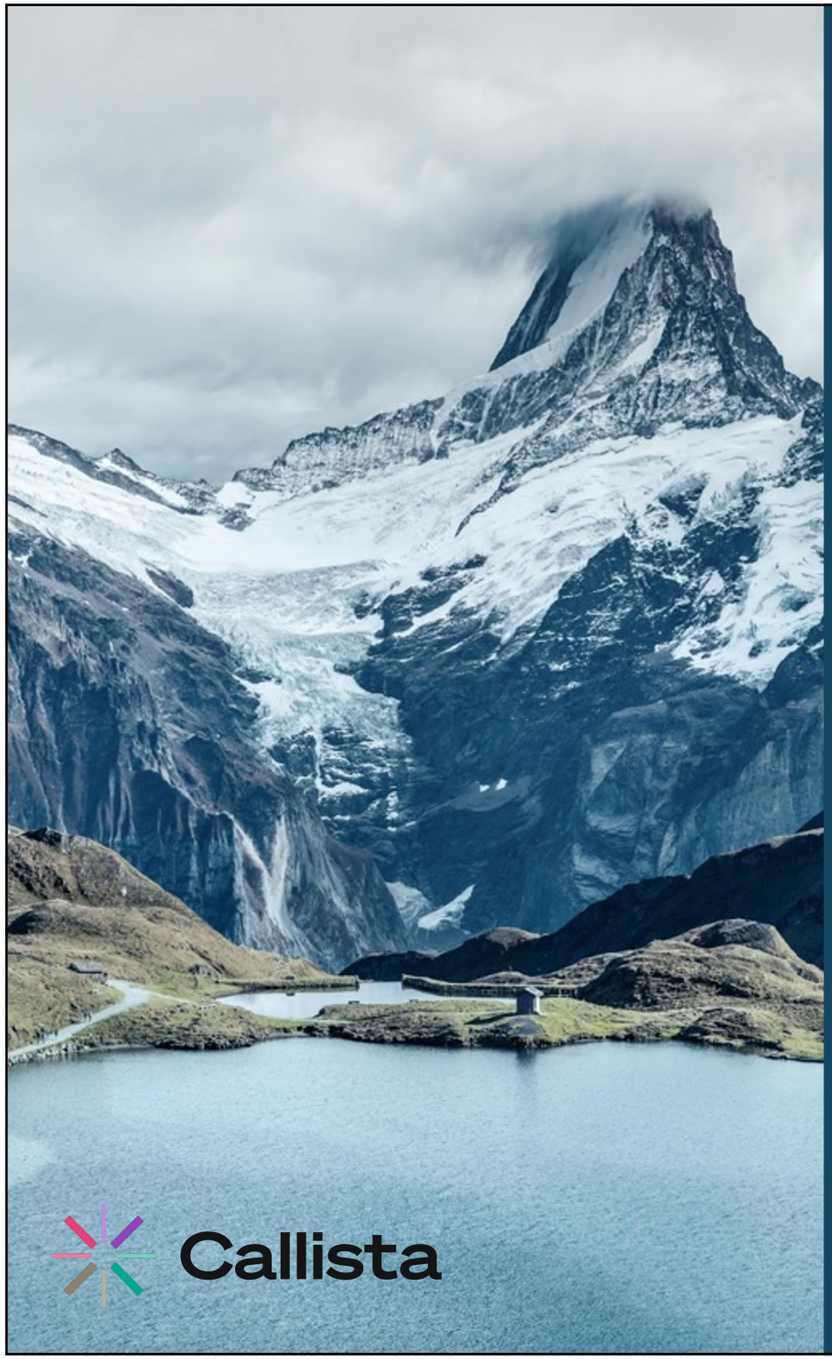
Callista

Bi-temporale Historisierung:

- Fachliche und technische Historisierung im Quellsystem
- Zusätzlich werden Daten im DWH historisiert
- Somit tri-temporale Historisierung

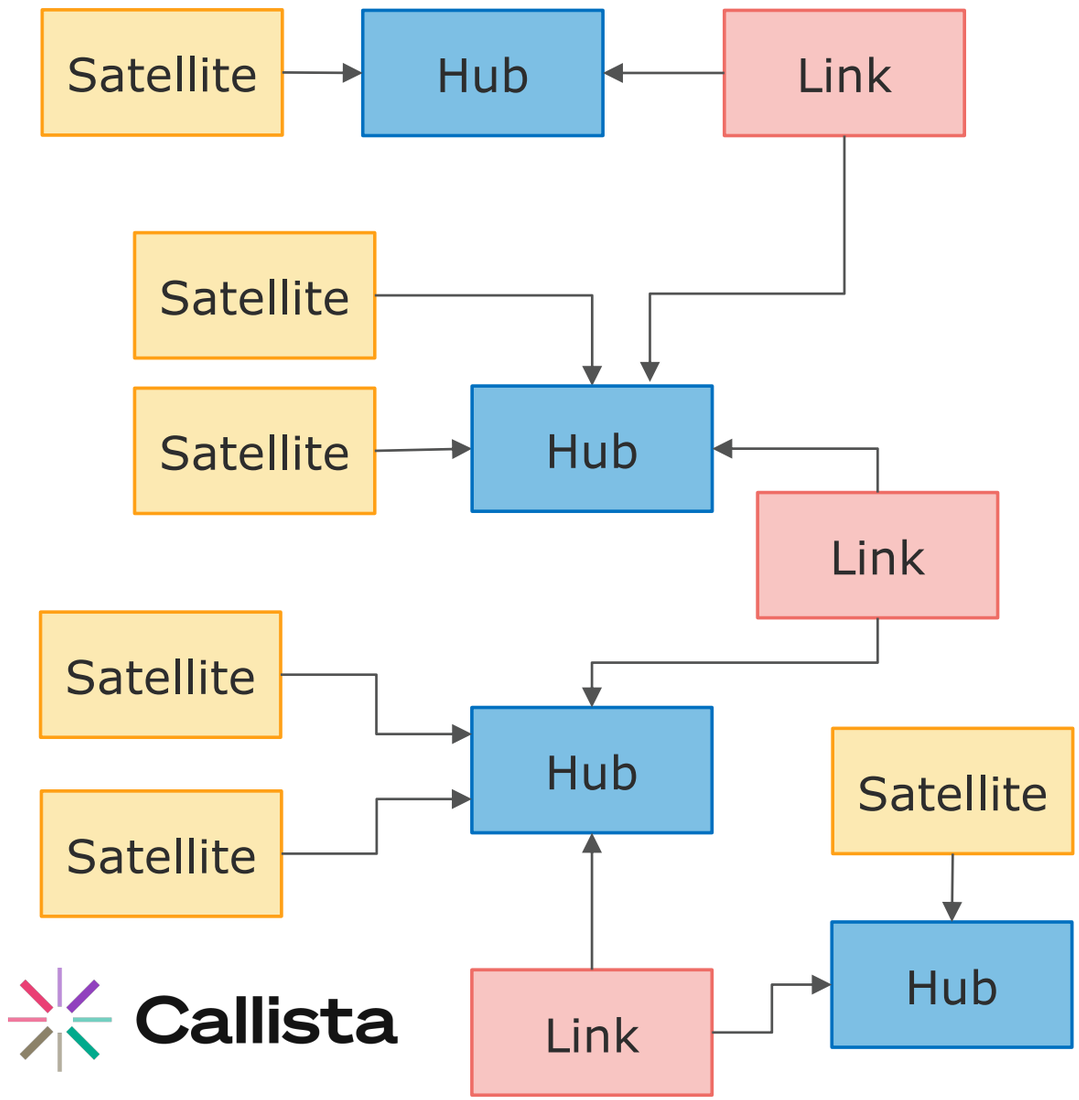
Drei Zeitachsen:

- Ladezeitpunkt
- Fachliche Gültigkeit (von/bis)
- Technische Historisierung im Quellsystem



Multi-temporale Historisierung mit Data Vault

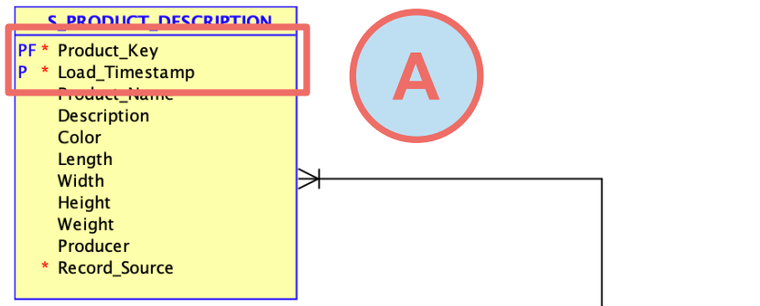
Data Vault

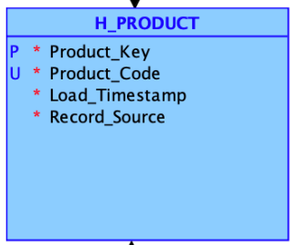


Callista

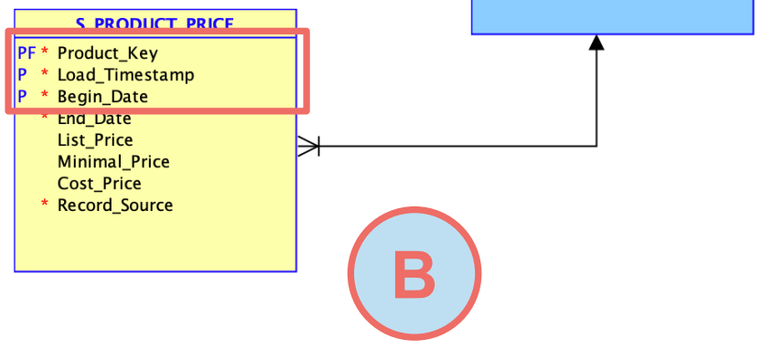
Data Vault: Historisierte Satelliten

HASH_KEY	LOAD_TIMESTAMP
A13CC05A0103051D	2024-08-30T18:53:18.124
A13CC05A0103051D	2024-10-07T03:10:46.345
A13CC05A0103051D	2024-10-11T02:48:44.023
A13CC05A0103051D	2024-10-14T02:48:54.500
A13CC05A0103051D	2024-10-23T03:01:07.379





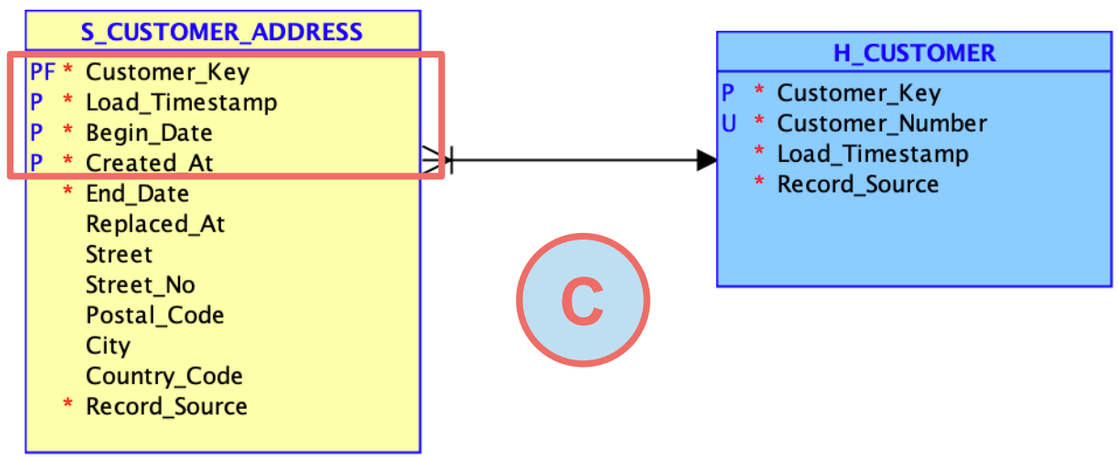
HASH_KEY	LOAD_TIMESTAMP	BEGIN_DATE	END_DATE
F4F291F16C359BCF	2024-08-30T14:57:10.330	1900-01-01	2011-07-31
F4F291F16C359BCF	2024-08-30T14:57:10.330	2011-08-01	2099-12-31
F4F291F16C359BCF	2024-09-11T02:17:13.581	1900-01-01	2011-07-31
F4F291F16C359BCF	2024-09-11T02:17:13.581	2011-08-01	2024-08-31
F4F291F16C359BCF	2024-09-11T02:17:13.581	2024-09-01	2099-12-31



Callista

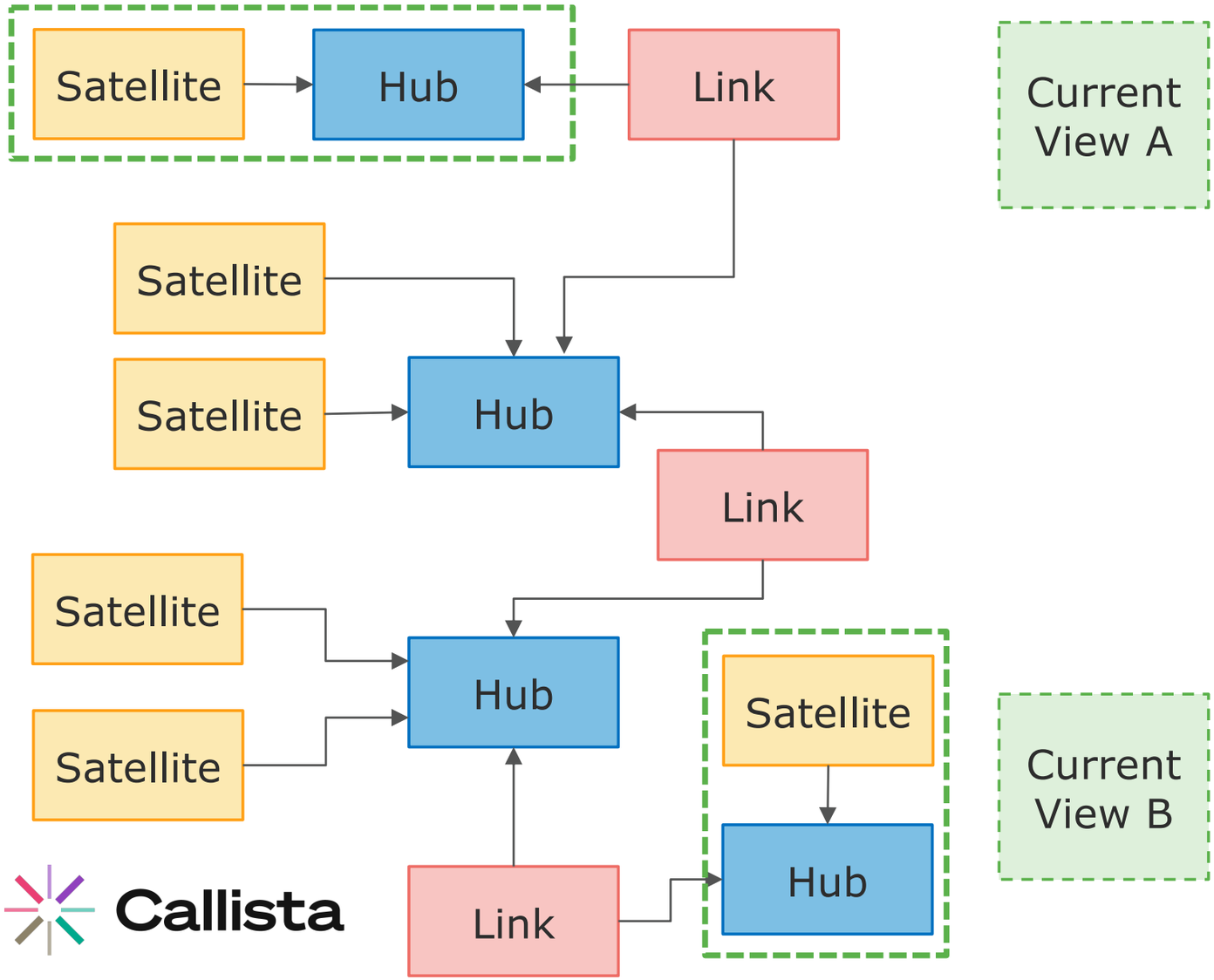
Data Vault: Historisierte Satelliten

HASH_KEY	LOAD_TIMESTAMP	BEGIN_DATE	END_DATE	CREATED_AT	REPLACED_AT
002272C8BE18D58B	2024-09-24T14:20:08.138	2023-01-01	2099-12-31	2022-09-18T08:43:22.9...	2022-10-06T02:39:39.1...
002272C8BE18D58B	2024-09-24T14:20:08.138	2023-01-01	2099-12-31	2022-10-06T02:39:39.1...	2023-10-07T10:02:24.5...
002272C8BE18D58B	2024-09-24T14:20:08.138	2023-01-01	2023-12-31	2023-10-07T10:02:24.5...	null
002272C8BE18D58B	2024-09-24T14:20:08.138	2024-01-01	2099-12-31	2023-10-07T10:02:24.5...	2023-12-11T14:41:16.0...
002272C8BE18D58B	2024-09-24T14:20:08.138	2024-01-01	2099-12-31	2023-12-11T14:41:16.0...	null
002272C8BE18D58B	2024-10-07T03:10:46.345	2024-01-01	2099-12-31	2023-12-11T14:41:16.0...	2024-10-06T00:25:10.8...
002272C8BE18D58B	2024-10-07T03:10:46.345	2024-01-01	2024-12-31	2024-10-06T00:25:10.8...	null
002272C8BE18D58B	2024-10-07T03:10:46.345	2025-01-01	2099-12-31	2024-10-06T00:25:10.8...	null



Callista

Data Vault + Current Views



Filtern auf gültige Versionen

A

HASH_KEY	LOAD_TIMESTAMP
A13CC05A0103051D	2024-08-30T18:53:18.124
A13CC05A0103051D	2024-10-07T03:10:46.345
A13CC05A0103051D	2024-10-11T02:48:44.023
A13CC05A0103051D	2024-10-14T02:48:54.500
A13CC05A0103051D	2024-10-23T03:01:07.379

B

HASH_KEY	LOAD_TIMESTAMP	BEGIN_DATE	END_DATE
F4F291F16C359BCF	2024-08-30T14:57:10.330	1900-01-01	2011-07-31
F4F291F16C359BCF	2024-08-30T14:57:10.330	2011-08-01	2099-12-31
F4F291F16C359BCF	2024-09-11T02:17:13.581	1900-01-01	2011-07-31
F4F291F16C359BCF	2024-09-11T02:17:13.581	2011-08-01	2024-08-31
F4F291F16C359BCF	2024-09-11T02:17:13.581	2024-09-01	2099-12-31

Pro Hash Key:

- Letzte geladene Version

Pro Hash Key und Beginndatum:

- Letzte geladene Version

Callista

Current View für Hub mit einem Satelliten

A

```
CREATE OR REPLACE VIEW example1_curr AS
WITH last_version AS (
   SELECT sat.*
        , RANK() OVER (PARTITION BY hash_key
                       ORDER BY load_timestamp DESC) rnk
     FROM example1_sat sat
)
SELECT h.hub_key
     , ...
     , ...
  FROM example1_hub h
  LEFT JOIN last_version s
    ON s.hash_key = h.hash_key
   AND s.rnk = 1
```

HASH_KEY	LOAD_TIMESTAMP
A13CC05A0103051D	2024-08-30T18:53:18.124
A13CC05A0103051D	2024-10-07T03:10:46.345
A13CC05A0103051D	2024-10-11T02:48:44.023
A13CC05A0103051D	2024-10-14T02:48:54.500
A13CC05A0103051D	2024-10-23T03:01:07.379

Callista

Current View für Hub mit einem Satelliten

B

```
CREATE OR REPLACE VIEW example2_curr AS
WITH last_version AS (
   SELECT sat.*
        , RANK() OVER (PARTITION BY hash_key, begin_date
                       ORDER BY load_timestamp DESC) rnk
     FROM example2_sat sat
)
SELECT h.hub_key
     , s.begin_date
     , s.end_date
     , ...
     , ...
  FROM example2_hub h
  LEFT JOIN last_version s
    ON s.hash_key = h.hash_key
   AND s.rnk = 1
```

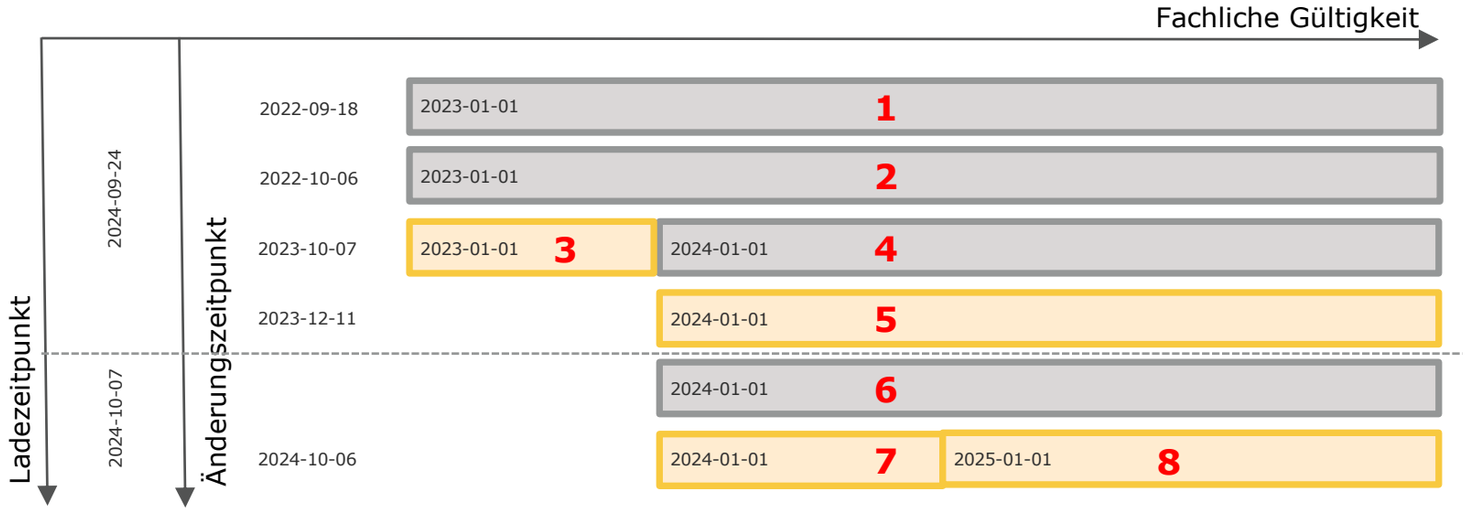
HASH_KEY	LOAD_TIMESTAMP	BEGIN_DATE	END_DATE
F4F291F16C359BCF	2024-08-30T14:57:10.330	1900-01-01	2011-07-31
F4F291F16C359BCF	2024-08-30T14:57:10.330	2011-08-01	2099-12-31
F4F291F16C359BCF	2024-09-11T02:17:13.581	1900-01-01	2011-07-31
F4F291F16C359BCF	2024-09-11T02:17:13.581	2011-08-01	2024-08-31
F4F291F16C359BCF	2024-09-11T02:17:13.581	2024-09-01	2099-12-31

Callista

Filtern auf gültige Versionen

C

	HASH_KEY	LOAD_TIMESTAMP	BEGIN_DATE	END_DATE	CREATED_AT	REPLACED_AT
1	002272C8BE18D58B	2024-09-24T14:20:08.138	2023-01-01	2099-12-31	2022-09-18T08:43:22.9...	2022-10-06T02:39:39.1...
2	002272C8BE18D58B	2024-09-24T14:20:08.138	2023-01-01	2099-12-31	2022-10-06T02:39:39.1...	2023-10-07T10:02:24.5...
3	002272C8BE18D58B	2024-09-24T14:20:08.138	2023-01-01	2023-12-31	2023-10-07T10:02:24.5...	null
4	002272C8BE18D58B	2024-09-24T14:20:08.138	2024-01-01	2099-12-31	2023-10-07T10:02:24.5...	2023-12-11T14:41:16.0...
5	002272C8BE18D58B	2024-09-24T14:20:08.138	2024-01-01	2099-12-31	2023-12-11T14:41:16.0...	null
6	002272C8BE18D58B	2024-10-07T03:10:46.345	2024-01-01	2099-12-31	2023-12-11T14:41:16.0...	2024-10-06T00:25:10.8...
7	002272C8BE18D58B	2024-10-07T03:10:46.345	2024-01-01	2024-12-31	2024-10-06T00:25:10.8...	null
8	002272C8BE18D58B	2024-10-07T03:10:46.345	2025-01-01	2099-12-31	2024-10-06T00:25:10.8...	null



Callista

Pro Hash Key und Beginndatum:

- Letzte geladene Version der letzten Änderung
- Nur technisch gültige Versionen (REPLACED_AT is null)

Current View für Hub mit einem Satelliten

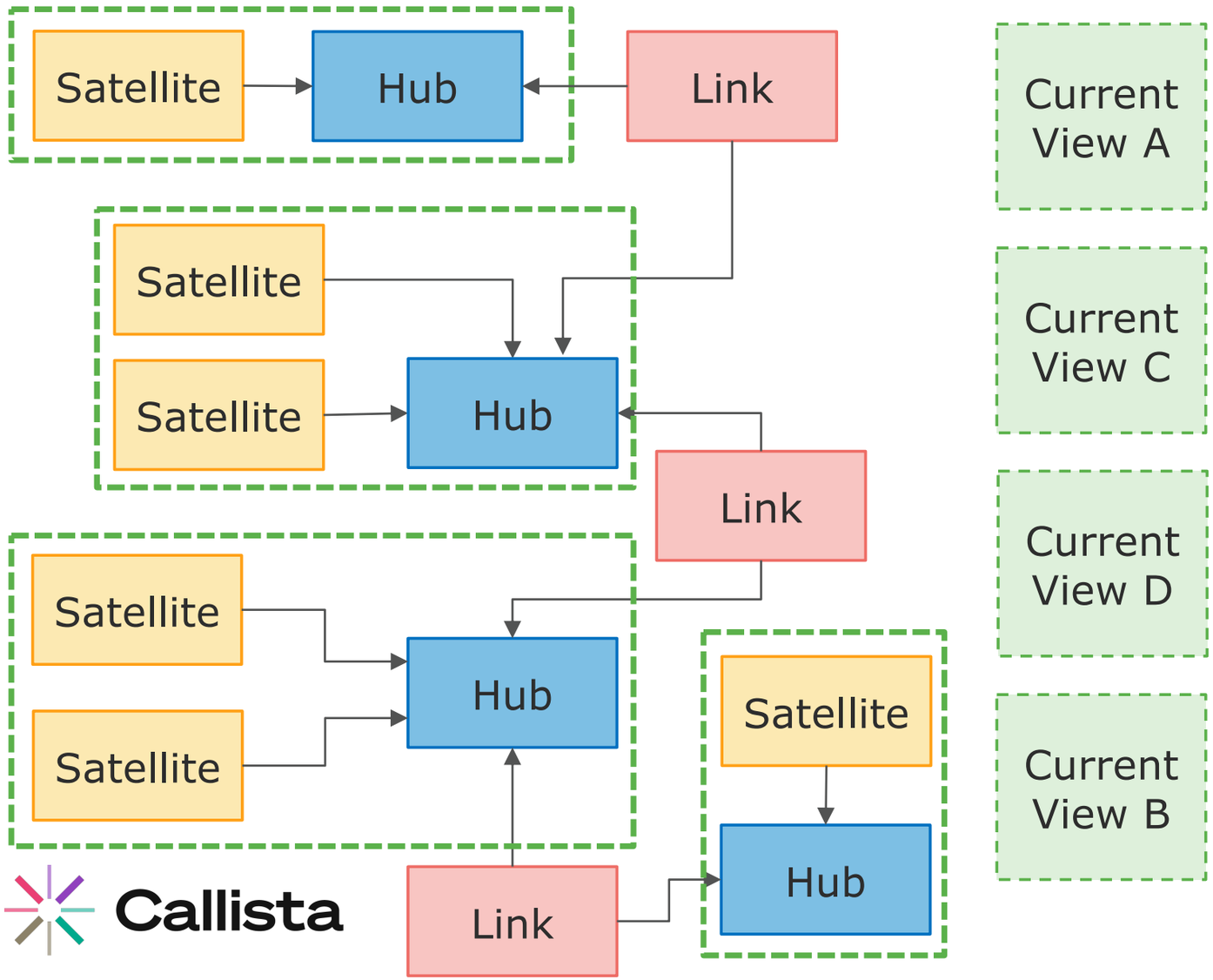
C

```
CREATE OR REPLACE VIEW example3_curr AS
WITH last_version AS (
   SELECT sat.*
        , RANK() OVER (PARTITION BY hash_key, begin_date
                       ORDER BY created_at DESC, load_timestamp DESC) rnk
     FROM example3_sat sat
)
SELECT h.hub_key
     , s.begin_date
     , s.end_date
     , ...
FROM example3_hub h
  LEFT JOIN last_version s
    ON s.hash_key = h.hash_key
   AND s.rnk = 1 AND s.replaced_at IS NULL
```

HASH_KEY	LOAD_TIMESTAMP	BEGIN_DATE	END_DATE	CREATED_AT	REPLACED_AT
002272C8BE18D58B	2024-09-24T14:20:08.138	2023-01-01	2099-12-31	2022-09-18T08:43:22.9...	2022-10-06T02:39:39.1...
002272C8BE18D58B	2024-09-24T14:20:08.138	2023-01-01	2099-12-31	2022-10-06T02:39:39.1...	2023-10-07T10:02:24.5...
002272C8BE18D58B	2024-09-24T14:20:08.138	2023-01-01	2023-12-31	2023-10-07T10:02:24.5...	null
002272C8BE18D58B	2024-09-24T14:20:08.138	2024-01-01	2099-12-31	2023-10-07T10:02:24.5...	2023-12-11T14:41:16.0...
002272C8BE18D58B	2024-09-24T14:20:08.138	2024-01-01	2099-12-31	2023-12-11T14:41:16.0...	null
002272C8BE18D58B	2024-10-07T03:10:46.345	2024-01-01	2099-12-31	2023-12-11T14:41:16.0...	2024-10-06T00:25:10.8...
002272C8BE18D58B	2024-10-07T03:10:46.345	2024-01-01	2024-12-31	2024-10-06T00:25:10.8...	null
002272C8BE18D58B	2024-10-07T03:10:46.345	2025-01-01	2099-12-31	2024-10-06T00:25:10.8...	null

Callista

Data Vault + Current Views



Current View A

Current View C

Current View D

Current View B

Current View für Hub mit mehreren Satelliten

```
CREATE OR REPLACE VIEW example4_curr AS
WITH sat1_curr AS (...)
,    sat2_curr AS (...)
SELECT h.hash_key
     , h.business_key
     , s1....
     , s2....
  FROM pit_intervals pit
  JOIN hub h
    ON h.hash_key = pit.hash_key
  LEFT JOIN sat1_curr s1
    ON s1.hash_key = h.hash_key
  LEFT JOIN sat2_curr s2
    ON s2.hash_key = h.hash_key
```

A

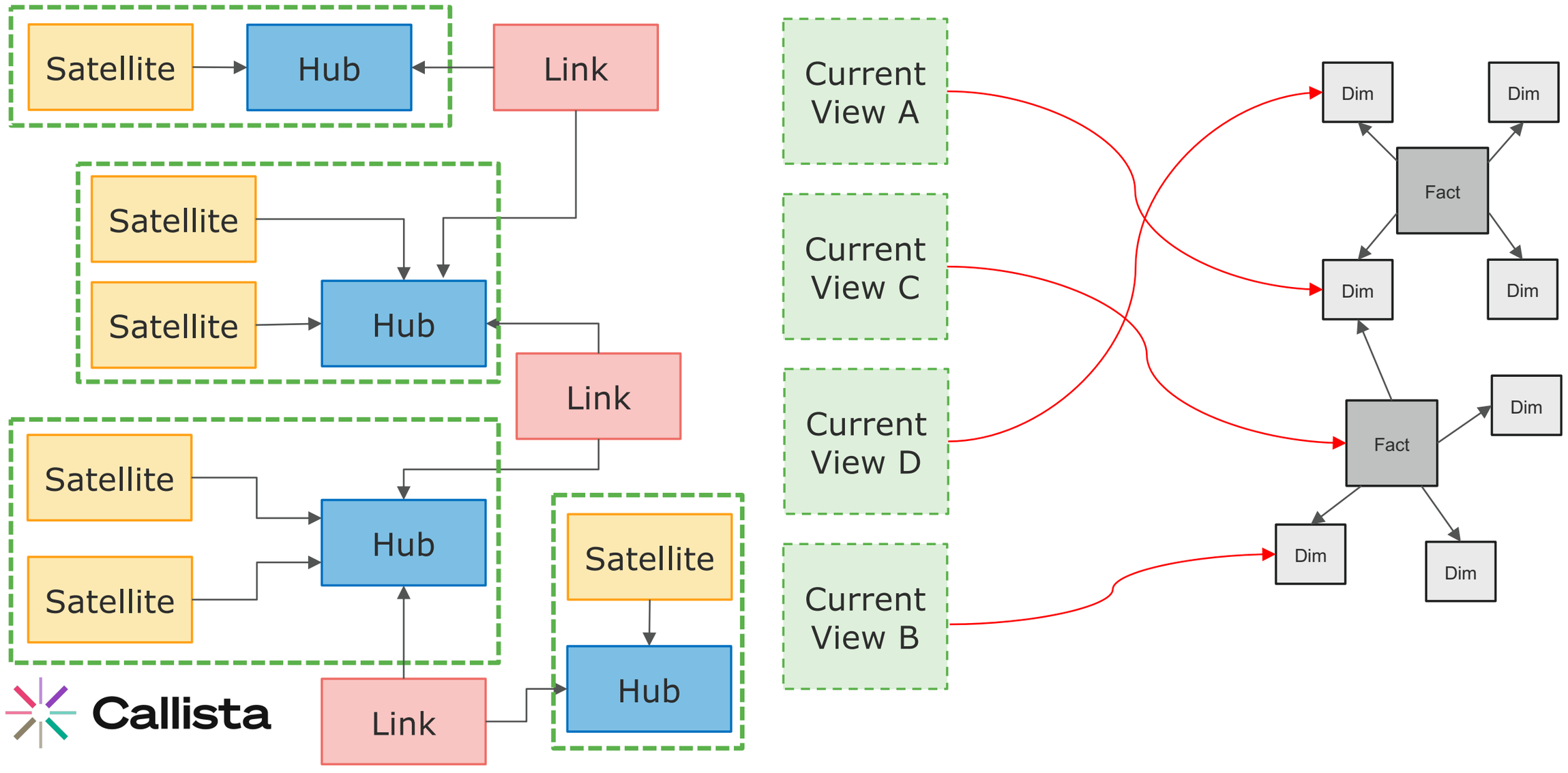
B

C

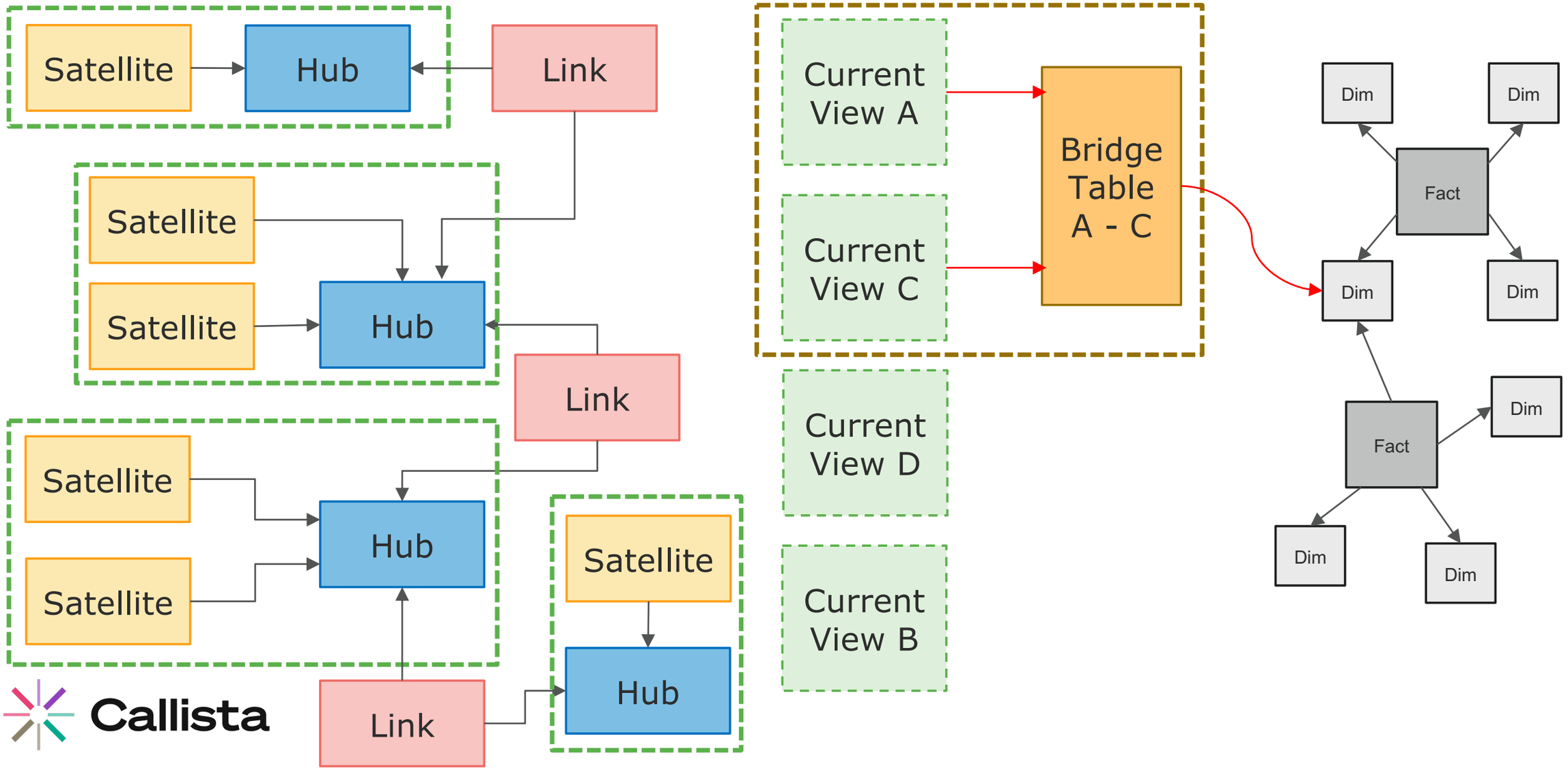
```
CREATE OR REPLACE VIEW example5_curr AS
WITH sat1_curr AS (...)
,    sat2_curr AS (...)
, valid_dates AS (
   SELECT hash_key, begin_date AS valid_date
     FROM sat1_curr
   UNION
   SELECT hash_key, end_date+1 AS valid_date
     FROM sat1_curr
   UNION
   SELECT hash_key, begin_date AS valid_date
     FROM sat2_curr
   UNION
   SELECT hash_key, end_date+1 AS valid_date
     FROM sat2_curr
)
, pit_intervals AS (
   SELECT hash_key
        , valid_date AS begin_date
        , LEAD(valid_date-1, 1) OVER (PARTITION BY hash_key
                                      ORDER BY valid_date) AS end_date
     FROM valid_dates
)
SELECT pit.hash_key
     , pit.begin_date
     , pit.end_date
     , h.business_key
     , s1....
     , s2....
  FROM pit_intervals pit
  JOIN hub h
    ON h.hash_key = pit.hash_key
  LEFT JOIN sat1_curr s1
    ON s1.hash_key = h.hash_key
   AND pit.begin_date BETWEEN s1.begin_date AND s1.end_date
  LEFT JOIN sat2_curr s2
    ON s2.hash_key = h.hash_key
   AND pit.begin_date BETWEEN s2.begin_date AND s2.end_date
 WHERE pit.end_date IS NOT NULL
```

Callista

Data Vault + Current Views + Data Marts



Data Vault + Current Views + Bridge Tables



Bridge Table

B C

BEGIN_DATE	END_DATE	CUSTOMER_KEY	CUSTOMER_BEGIN_DATE	ADDRESS_KEY	ADDRESS_BEGIN_DATE
1900-01-01	2010-01-14	3AB7651AF0093F0A	1900-01-01	736B17EC37DE422A	1900-01-01
2010-01-15	2015-09-11	3AB7651AF0093F0A	2010-01-15	736B17EC37DE422A	1900-01-01
2015-09-12	2019-07-30	3AB7651AF0093F0A	2010-01-15	736B17EC37DE422A	1900-01-01
2019-07-31	2019-07-31	3AB7651AF0093F0A	2010-01-15	736B17EC37DE422A	1900-01-01
2019-08-01	2024-08-12	3AB7651AF0093F0A	2010-01-15	736B17EC37DE422A	2019-08-01
2024-08-13	2099-12-31	3AB7651AF0093F0A	2024-08-13	736B17EC37DE422A	2019-08-01

Source: Golden gate bridge icons created by Freepik - Flaticon

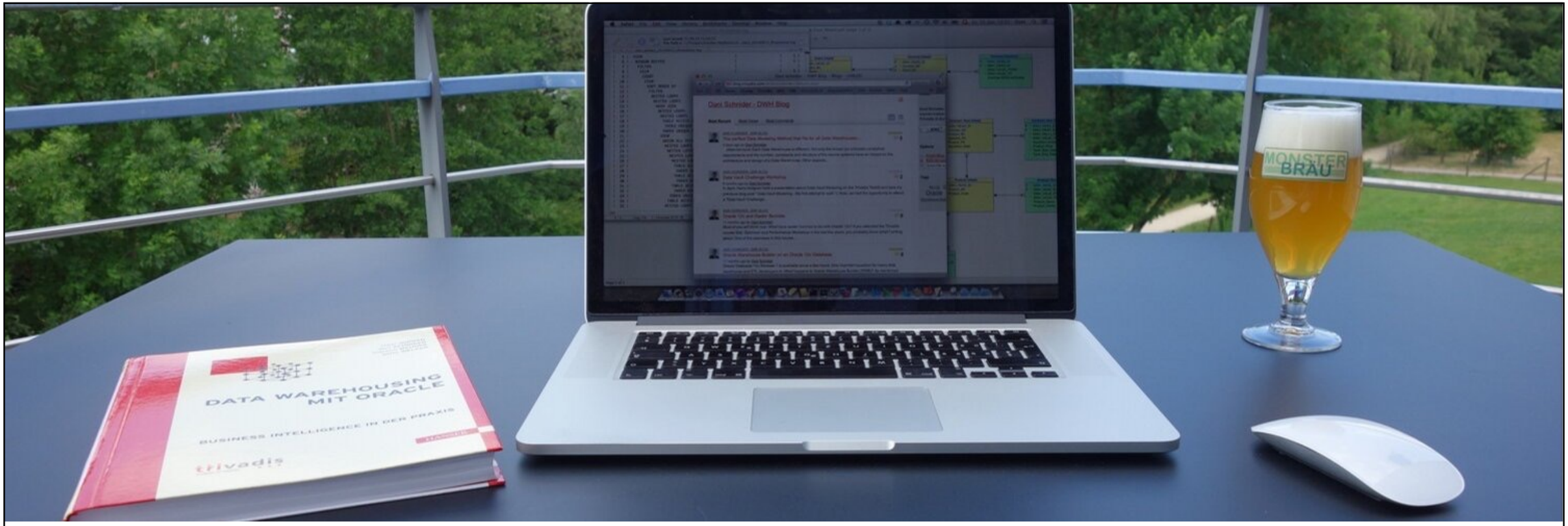
```
CREATE TABLE customer_address_bridge AS
WITH cust_adr_lnk AS (
    SELECT lnk.customer_key
         , cal.address_key
         , cal.begin_date
         , cal.end_date
      FROM address_curr adr
      JOIN customer_address_lnk lnk
        ON cal.address_key = lnk.address_key
)
, valid_dates AS (
    SELECT cus.customer_key AS customer_key
         , cus.begin_date AS valid_date
      FROM customer_curr par
    UNION
    SELECT cus.customer_key AS customer_key
         , cus.end_date+1 AS valid_date
    FROM customer_curr par
    UNION
    SELECT cal.customer_key AS customer_key
         , cal.begin_date AS valid_date
    FROM cust_adr_lnk cal
    UNION
    SELECT cal.customer_key AS customer_key
    , cal.end_date+1 AS valid_date
    FROM cust_adr_lnk cal
)
, pit_intervals AS (
    SELECT customer_key
         , valid_date AS begin_date
         , LEAD(valid_date-1, 1)
                OVER (PARTITION BY customer_key
                      ORDER BY valid_date) AS end_date
      FROM valid_dates
)
SELECT pit.begin_date   AS begin_date
     , pit.end_date     AS end_date
     , cus.customer_key AS customer_key
     , cus.begin_date   AS customer_begin_date
     , cal.address_key  AS address_key
     , cal.begin_date   AS address_begin_date
  FROM pit_intervals pit
  LEFT JOIN customer_curr cus
    ON pit.customer_key = cus.customer_key
   AND pit.begin_date BETWEEN cus.begin_date AND cus.end_date
  LEFT JOIN cust_adr_lnk cal
    ON pit.customer_key = cal.customer_key
   AND pit.begin_date BETWEEN cal.begin_date AND cal.end_date
 WHERE pit.end_date IS NOT NULL
```

Callista

Lessons Learned

- Multi-temporale Historisierung ist komplex
- Data Vault für Anforderung gut geeignet
  - Laden ist einfach ☺
  - Lesen komplexer, aber mit Standard-Patterns möglich
- Beschränkung auf eine Zeitachse für Auswertungen
  - Für die meisten BI-User nur fachliche Historisierung relevant
  - Für Spezialanwendungen Zugriff auf Raw Data Vault möglich
- Knowhow-Transfer wichtig!
- Historisierung in Quellsystemen muss stimmen!!!

Callista



https://danischnider.wordpress.com

https://bsky.app/profile/danischnider.bsky.social

https://www.linkedin.com/in/danischnider/

Q&A

Feedback

Callista

Mi 9.4


DRIVEN BY EXCELLENCE
Callista