Satellite Applications

#Questions

- How does sequencing works for solving the order issue in Satellite Applications > Multi-Active satellites? What are possible solutions to this issue?
- I need more examples on how the Satellite Applications > Record tracking satellites work. #developing

Overloaded satellites

It is still preferable to create one satellite per source system.

- A satellite including multiple source systems instead of separating it in multiple satellites.
- Is chosen when different source systems might have almost the same data and structure.
- This satellite still contains a lot of risks:
  - Multiple records can have the same hash key.
  - Overlapping records and no idea of which one is the most current or real.

Multi-Active satellites

- Store multiple entries per parent key. However, these records don't come from different sources, but from denormalized data source, such as COBOL copybooks or XML files.
- Example. Storing multiple phone numbers per user; I would need to create multiple records for the same user because the logical satellite design will only contain one column for the phone number (and it accepts only one value).
- One of the main issues encountered on these satellites is that if one of the records changes, the new record invalidates the original order of the records (which is not ideal).
- There are ways to overcome this issue if the order is relevant. As of right now, I cannot say that I understood the solutions (sequencing) #developing .

Status tracking satellites

- Load data from Change Data Capture (CDC) systems using CRUD or SCRUD (search, create, read, update, delete).
- If multiple source systems provide status information, only the information from the master system should be tracked.

- The satellite structure is the standard (Satellite definition) and it will include one additional attribute `operation` to indicate the status (one of SCRUD).

Effectivity satellites

- Are used to track when the link is active according to the business and provides `begin` and `end` dates for this purpose. These dates are not system generated, instead they have to be provided from a data source or any other audit trail from operational systems.
- Due to this reason they are often found as satellites owned by link entities.
- These satellites are only reasonable if the source system provides the effectivity dates, **we should avoid creating artificial effectivity dates.**
- If we include the effectivity dates, these do not override the `LoadEndDate` attribute, as we might have more than one equivalent record at a time.

Record tracking satellites

- These satellites are born in the context when it is required to track which source applications are feeding which keys and associations on what load cycle. For example, many systems provide data in full dumps every day; in some cases, not all records are included every day, instead they appear and disappear day to day without any fixed rule.
- Has no `RecordSource` and no `LoadEndDate` attributes. This is because all records are system generated (no `RecordSource`) and records are never updated (no `LoadEndDate`).
- Instead of updating the records on each load cycle, a new entry is inserted every time.
- Due to its characteristics, these satellites **are not auditable**.
- We can add an attribute `Appearance` to indicate if the business key or relationship under watch is tracked.

Computed satellites

- Usually Data Vault is used for raw data but it can also be used for system generated data.
- This satellite store the system generated data, product of aggregation, transformation, etc.
- It follows the standard structure but in the `RecordSource` will define `system–generated` or similar to indicate that it is not a raw data satellite.
- Because there is no direct raw data source for the computed data, it nos auditable by its very nature. However you can audit the procedures to generate the data.