Considerations for Multi-Active Satellites

Right or wrong?

When it comes to data management there are almost always various alternatives for implementation and none of them are necessarily right or wrong. They represent the various options and consequences to consider, and the right solution usually is the one which is made with full understanding of these consequences, with 'eyes wide open'.

Supporting multi-active, sometimes referred to as 'multi-variant' or 'multi-valued', behaviour of Satellites is one of these areas where opinions vary.

This paper intends to outline the design decisions related to the multi-active concept, and to provide some background to assist with locking down your own fit-for-purpose approach.

What is multi-active?

The multi-active approach provides a way to handle situations where the data doesn't fit the ideal world of your target (Data Vault) model.

We refer to multi-active when you want to associate context (Satellite) to a specific business concept (Hub) but it just doesn't quite fit the granularity. Or to be more precise: a Satellite that houses multiple active records (valid values) for a given business key for a specific point in time.

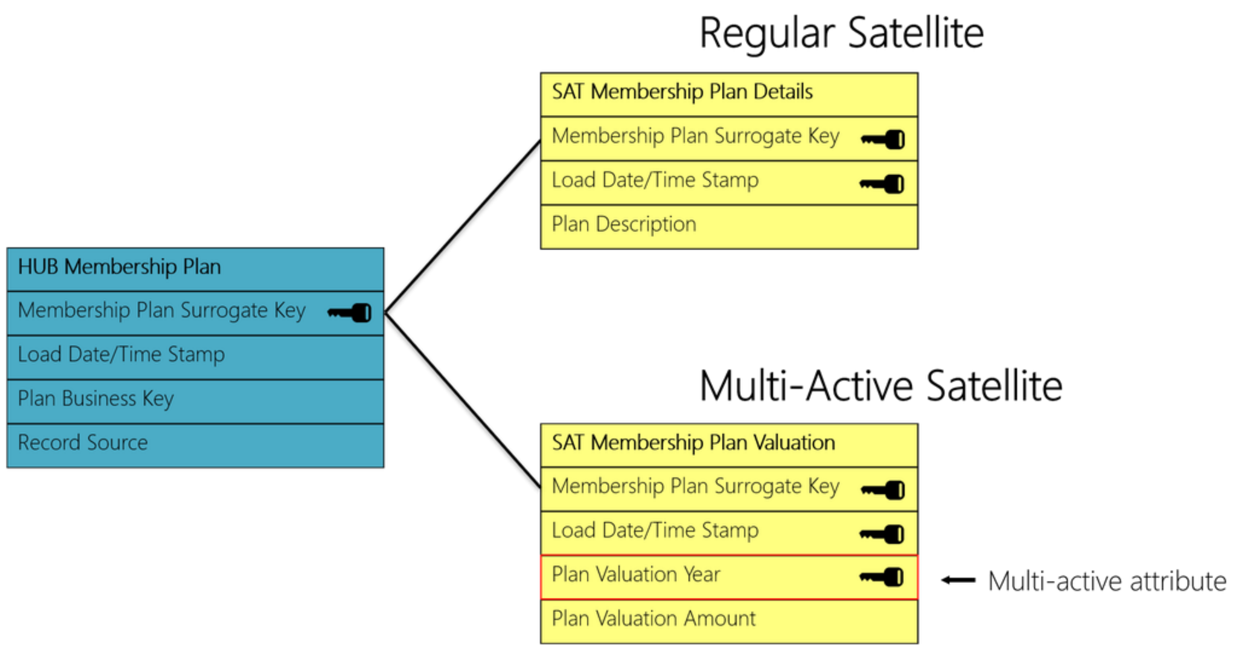
In these cases, you may want to associate this context to the business concept *per something else*: another attribute that shifts the granularity to the desired level while still referring to the intended business concept.

In other words: multi-active attributes are a non-Hub key field in the Primary Key of the Satellite. They change the granularity, but *still* directly describe the Business Concept.

The default example and implementation of a multi-active Satellite is that this 'other' attribute is added to the Primary Key of the Satellite, in addition to the Hub key that is inherited by the Business Concept.

This is opposed to the 'regular' or base Satellite pattern where the Primary Key consists of only the Surrogate Key (Hash Key) and the Load Date/Time Stamp (LDTS). For demonstration purposes both versions are shown in the example.

An example of this is displayed below.



In this scenario you have a hypothetical Membership Plan ('Plan') business concept that has a yearly valuation of the funds associated with the Plan. These funds, the 'plan valuation amount' in the example, do not directly describe the Plan: they belong to a Plan *for a given year*.

A better definition is to say that if you have *multiple* active plan values for a specific plan at a *specific* point in time – then that makes the plan valuation amount multi-active.

In contrast, the regular Satellite has the context attribute 'plan description', which directly describes the Plan because the Plan business key (surrogate key) and the date/time stamp are the only key elements.

This can be applied to both Satellites as well as Link-Satellites, if you choose to model your Data Vault this way.

The pattern(s)

Some would argue that multi-active breaks the Satellite pattern. After all, the context is not directly describing the business key anymore. Also, to 'support multi-active' you need to cater for this scenario in your ETL pattern and generation logic. It can be considered a separate pattern.

You basically have three options for implementing this use case:

Scenario 1: incorporate a multi-active attribute in the Satellite

This is the base (traditional) solution, which is explained in the previous section. The multi-active attribute acts as a value / type classification in the Satellite.

The advantage of this is that you limit the number of tables (not a massive consideration in the context of automation and abstraction I know – but still), but also keep a greater level of flexibility in handling data changes. If there are new types to support, these would natively be handled in the same pattern. This is both good and bad, and will be explained in more detail in the 'what to look out for' section further in this post.

The disadvantage is that you have to support this additional template / behaviour in your ETL pattern and generation code. You have to detect and cater for the multi-active attribute as part of the key, for example when applying row condensing, looking for the 'most recent row' and things like end-dating if this is part of your approach.

This makes the development but also the interpretation of your Data Warehouse a bit more complex. After all, the context is stored at a slightly different level and requires better understanding to be retrieved.

Consider the example below which contains the Valuation Amount information for a given plan.

SAT Membership Plan Valuation	
Membership Plan Surrogate Key	🔑
Load Date/Time Stamp	🔑
Plan Valuation Year	🔑
Plan Valuation Amount	

Membership Plan Surrogate Key 🔑	Load Date/Time Stamp 🔑	Plan Valuation Year 🔑	Plan Valuation Amount
b3a91fdeeff141eede04fb67686bac76	2016-04-03	2016	100000
b3a91fdeeff141eede04fb67686bac76	2018-07-02	2016	110000
b3a91fdeeff141eede04fb67686bac76	2017-12-31	2017	50000
b3a91fdeeff141eede04fb67686bac76	2018-02-06	2018	1250

If you are looking to retrieve the amount values for a given plan a direct join between the Hub and the Satellite will give you the four records.

In the simplest example of having this information made available in a Type 1 Dimension you may want to exclude the first record (LDTS of 2016-04-03) by applying the 'most recently received' view. In this case you have to select the 'most recent' row for each Surrogate Key *and* Plan Valuation Year to reflect the multi-active nature.

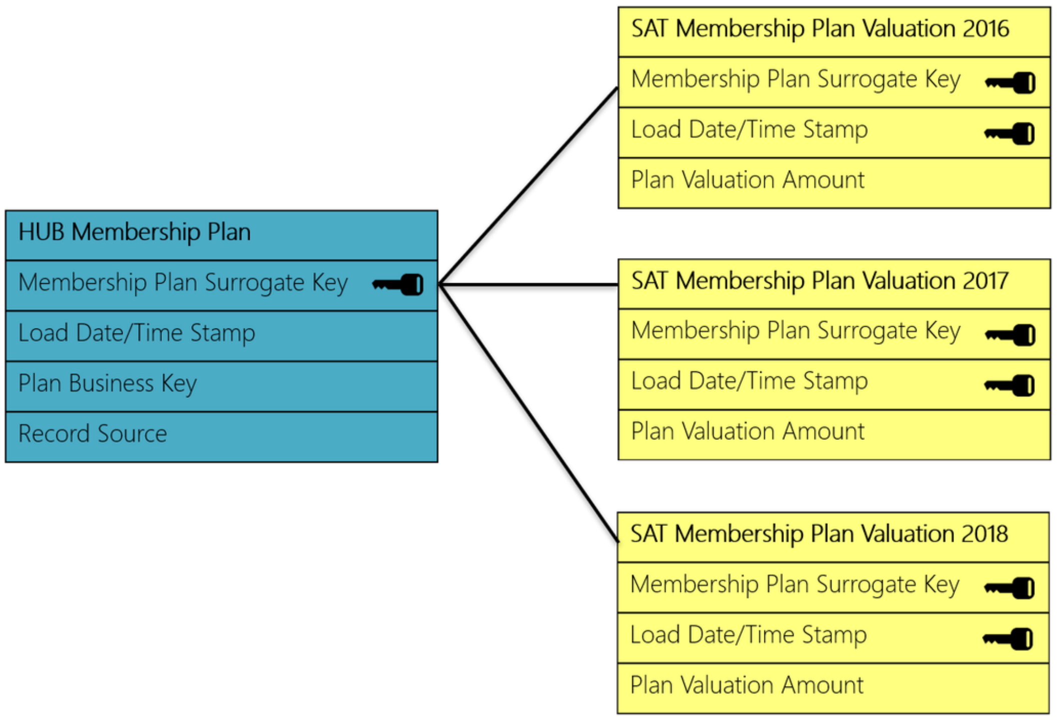
In the Type 1 Dimension example, the results could look like this after transposing the result set from the query on the Data Vault:

Dimension Key 🔑	Membership Plan Surrogate Key	2016 Valuation	2017 Valuation	2018 Valuation
1	b3a91fdeeff141eede04fb67686bac76	110000	50000	1250

Scenario 2: implement a separate table for each value / type / classification

Instead of having an attribute that specifies what the split or type should be in a table, you can also create separate tables by type.

An example of this is provided below. Each 'year' (as type / classification) is modelled as a dedicated table.



SAT Membership Plan Valuation 2016
Membership Plan Surrogate Key
Load Date/Time Stamp
Plan Valuation Amount
HUB Membership Plan
Membership Plan Surrogate Key
Load Date/Time Stamp
Plan Business Key
Record Source
SAT Membership Plan Valuation 2017
Membership Plan Surrogate Key
Load Date/Time Stamp
Plan Valuation Amount
SAT Membership Plan Valuation 2018
Membership Plan Surrogate Key
Load Date/Time Stamp
Plan Valuation Amount

The advantage is that you can reuse the existing 'base' Satellite pattern, after all there is no difference in table structure, ETL pattern and generation anymore. It's like a normal Satellite, only you have more of them.

You have to make sure is that you route data by using a filter criterion in your ETL, which should be part of the fundament generation logic and ETL pattern anyway.

However, you also need to be able to rely much more on your data consistency and monitor that the filter criteria are both mutually exclusive and don't have gaps. You don't want to have data 'slip through' and consequently not be available in the Data Warehouse.

In other words, this doesn't work as well when the multi-active attribute (now rebranded as filter criterion) is not as 'immutable' as believed, and values change unexpectedly as a result. This approach is often applied for things like address types (i.e. 'Home', 'Work').

Of course, this requires model changes when type changes are applied. In the example used here, a new table would have to be created for the 2019 results.

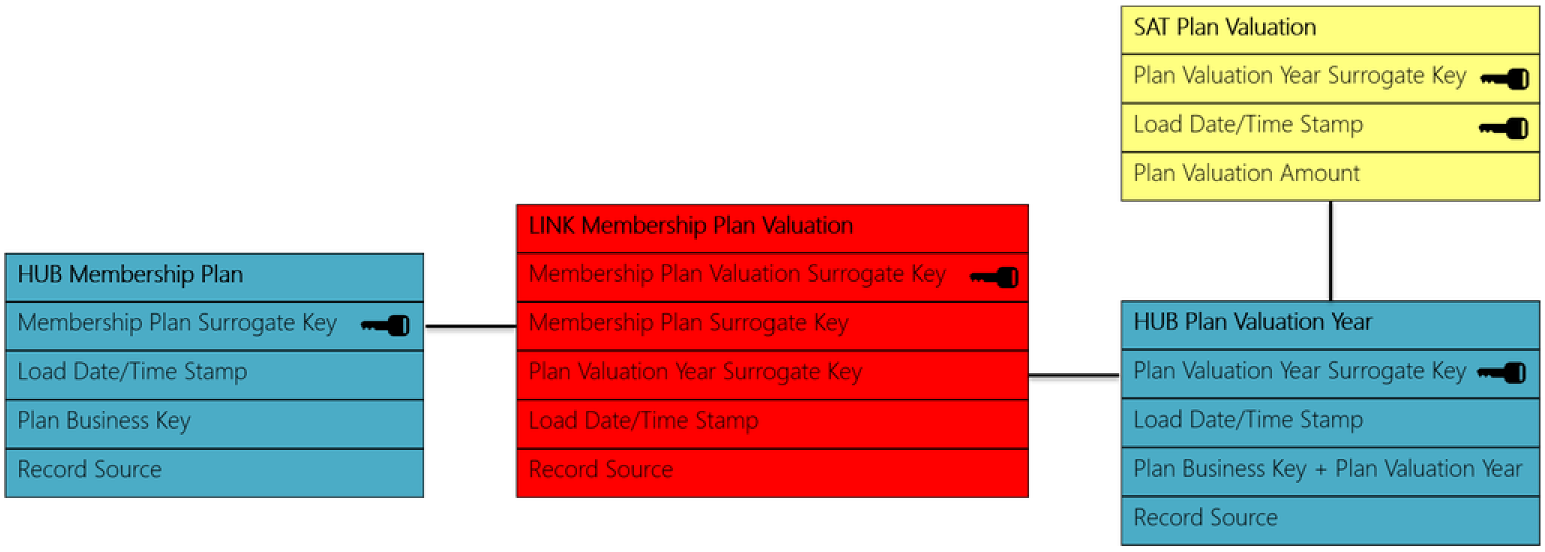
One thing to consider is that there may be cases where the source system does not limit the number of types, in which case splitting types per table is not a good solution. If you have a Person business key with contact details as context, there may not be a limit to the amount of phone numbers and email addresses.

It is relatively easy to move from this model to the Type 1 Dimension as used before. This can be achieved by simply joining the three Satellite tables to the Hub using the 'most recently received' view of selecting the highest Load Date/Time Stamp. The information is already in the correct structure to be applied to the example Dimension (no transposing required).

Scenario 3: creating a 'weak' Hub that incorporates the type / classification

Instead of using the Satellite approach to begin with, you can also create a dedicated Hub that 'role-plays' the multi-active behaviour. This Hub can be added to a Link together with the other (main) business concept.

This is sometimes considered a 'weak' Hub as it may not be an agreed business concept. An example is below (Hub Plan Valuation Year being the 'weak' Hub):


SAT Plan Valuation
Plan Valuation Year Surrogate Key
Load Date/Time Stamp
Plan Valuation Amount
LINK Membership Plan Valuation
Membership Plan Valuation Surrogate Key
Membership Plan Surrogate Key
Plan Valuation Year Surrogate Key
Load Date/Time Stamp
Record Source
HUB Membership Plan
Membership Plan Surrogate Key
Load Date/Time Stamp
Plan Business Key
Record Source
HUB Plan Valuation Year
Plan Valuation Year Surrogate Key
Load Date/Time Stamp
Plan Business Key + Plan Valuation Year
Record Source

The advantage of this approach is that you achieve the same outcomes as creating a multi-active Satellite as per scenario 1, without the need to have separate patterns. Indeed, the regular arguably simpler patterns and generation code can be used as there is no need to handle the additional key part as would be the case for scenario 1.

The disadvantage is that you sacrifice a part of the model to be 'more technical' as opposed to 'closer to the business'. There is nothing specifically wrong with this, it's simply a consideration as to what each part of the architecture and model is supposed to do.

I have seen variations of this being implemented under the banner of 'raw Data Vault' with a separate set of derived tables having a more business view in the 'business Data Vault'.

What to look out for

When adopting multi-active the key thing to understand is the *immutable* property of the multi-active attribute. This simply means there is (or should be) little or no reason to ever change the value once it has been set.

There are different ways to achieve this, including governance and adoption of technical techniques such as using (deterministic / consistent) sequences.

The bottom line is that you have to rely on the information you are receiving into your Data Vault, and this exposes a risk that the Data Warehouse team should be aware of (and accept). You are, to some extent, at the mercy of the information you receive when you implement multi-active.

This ties in closely to the interpretation and detection of deltas, including logical deletes. If an existing multi-active attribute is changed (i.e. it is in fact mutable) this means the records (the 'key') associated with the *old* value need to be end-dated because the new value will start its own new timeline in the Satellite.

On some level this is similar to a change on a composite key on a normal table in a database: an update on one of the key fields would be interpreted as an insert (of the new record) and delete (of the new records): as two events as opposed to an update on a single key.

This is where issues are likely to occur. From a Data Warehouse perspective, the feeding (source) systems can't always be trusted and some attributes have a tendency not to be as immutable as intended.

I have seen things go 'pear-shaped' because values that were used as multi-active were later updated (sometimes bypassing the application at database level by support teams).

A common issue is encountered when business effective dates are modelled as multi-active attributes. The fact that some feeding systems uses a form of effective dating is *not* a good reason to implement this as multi-active in the Data Warehouse. Imagine the multi-active attribute is an effective date, which is later updated to something else – this can make interpretation very difficult.

As is always the case it's important to be aware of these dynamics. Hopefully this sheds some additional light on the multi-active optic and helps to select the modelling views and patterns that are compatible for your solution architecture.

Which approach should I pick?

I hope it is clear that the approach to follow largely depends on your personal preferences and (almost philosophical) views on what the role of the model is when it comes to representing the data. All approaches can be made to work.

For me personally though, I do like the traditional approach (scenario 1) for the way the model looks. You can easily tell what context is available for a certain Business Concept.

At the same time however, when you consider abstracting out the design to higher levels of modelling following the engine and graph model (collapsing, expanding) line of thinking, this becomes less relevant as the physical model (conceptually) almost becomes a run-time execution which most of the time you may not be looking at.

It does appear that the traditional multi-active concept comes with certain challenges for many people. It's a recurring topic in the trainings I host, and I have seen it gone wrong in various projects. The reason seems to be that in these cases the selected multi-active key wasn't as immutable as first thought, which can lead to overloading complexity when querying data from the Data Vault.

It is for reasons like these that, for example, Peter Avenant from Varigence mentioned that the traditional multi-active functionality (i.e. the 'scenario 1') will be hidden by default in the new release of BimlFlex (still supported however).