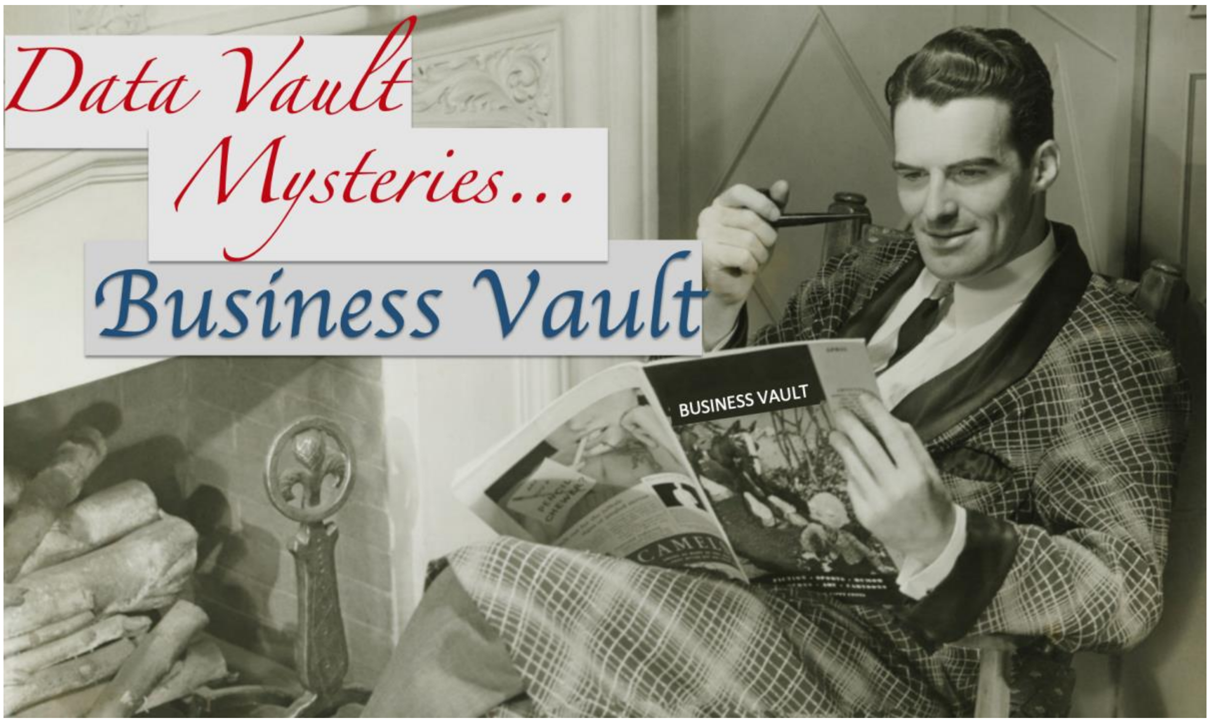


In today’s episode of Data Vault Mysteries we open the book on Business Vault.

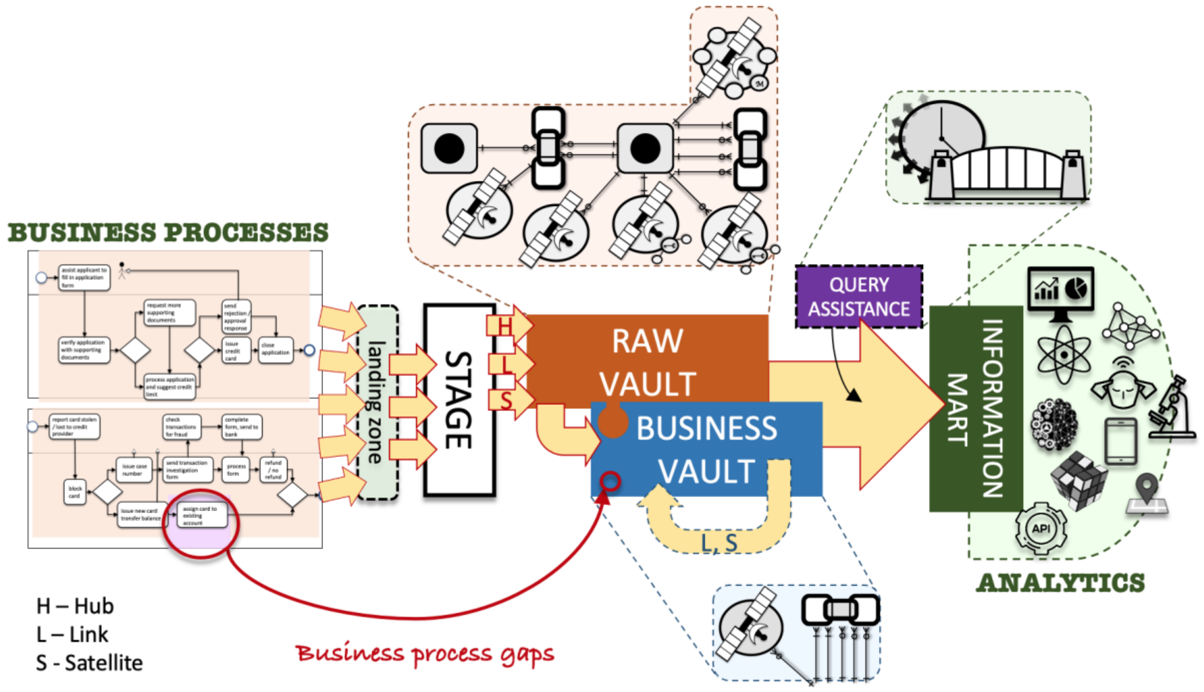
What is it?

To understand Business Vault we need to understand **Raw Vault** first. A data vault model is made up of **hubs, links and satellites** as it is well documented. Each of these artefacts has a special purpose that is defined as the following:

- A **hub** table is the ***unique list of business keys***, and a raw vault model can have many; each representing a domain, subject area, capability or part of a business process of the organization. For a unique list of account numbers you will typically find in a table called hub_account, customer ids would have been loaded to hub_customer and so on.
- A **link** table not only contains the relationships between business keys but also represents the ***unit of work***. Where does this unit of work come from, well they come from the automated data output of ***business processes***.
- A **satellite** table is used to track the ***change record*** for either a hub table (business key) or a link table (the unit of work).

Why then are these defined as ***RAW*** vault artefacts? Because the data is produced by source applications that ultimately are ***automation engines*** for business processes. And they continue to produce data, collected and modelled into raw vault.
The business vault has many uses, in the context of raw vault the business vault fills the ***gaps in business processes***. Imagine the viewpoint from the source applications, often these are third party software catering for their own bottom line and often servicing hundreds or thousands of their own clients, sometimes *globally*. To the best of their knowledge and generalization of the industry they are designed to serve, the business processes automated will be of a more *generic* nature. Sure they take suggestions on how to improve/adapt their software, and through version and maintenance releases they update their data model but

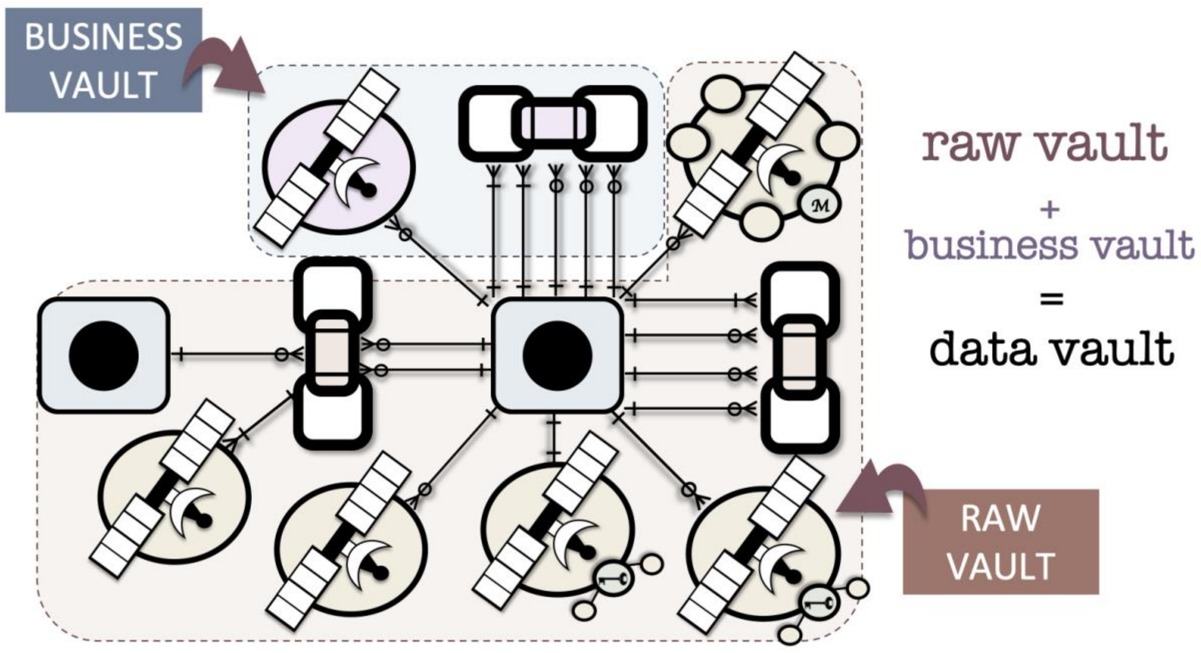
their clients usually cannot wait for those change requests to be actioned in order to proceed with their businesses. Some change requests may in fact *never* get done!
That is what business vault is designed to bridge! Fill in the ***business process gaps*** that would have ideally been solved by source systems! It would probably have been easier if the application was built in-house but even in those circumstances you cannot *always* expect the application to respond exactly to the business analytics' timelines.



If these gaps can be filled by the source application then that is the best place to solve it! Model the change when it is solved and migrate the business vault artefact history to a new raw vault artefact *where necessary*.
The second purpose of business vault is to centralise derived content by reusing the same loading patterns for raw vault! That means the same ***agility***, ***auditability*** and ***automation*** you expect in raw vault is a given in business vault!

Where does it live?

Derived content that features the auditability mentioned above should contain the lineage of how the data was produced; that means that the business vault is ***based*** on raw vault and extends the overall raw vault model where it is needed. If a business vault is delivered as views instead of physical tables you do gain in performance (initially) but then through time as the raw vault grows your business vault ***query performance*** starts to ***suffer***. A view is unexecuted code logic, that means that by defining a business vault as a view introduces ***vendor lock-in***, rather if the derived business rules outcome is persisted instead then there is no vendor lock-in and you are free to deliver derived results on ***any tool*** and persist the outcome against the unit of work or business entity.

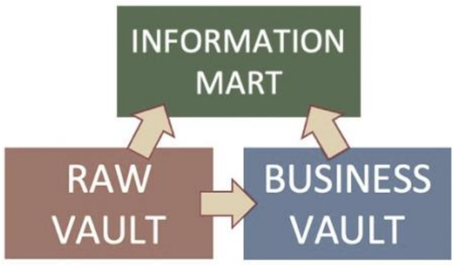


Business vault physicalised also means you can better manage business rules as ***versions***, that is if a business rule is updated it does not update history, but rather is applicable from a point-in-time onwards. It also means that Information marts are not deployed as views on top of views! A recipe for a slowly-*degrading*-information mart!

What does it look like?

Although you re-use raw vault artefacts to load a business vault it is ***never a replication*** of raw vault. You design derived business rule outcomes that are in itself ***autonomous***, that is, it should **not** be reliant on a previously calculated derived value to get to a new derived value. Why? If any correction is needed in the previous value the potential snowball effect of fixing downstream values could have a knock-on effect on every dashboard and report based on that value. That's not to say that rolling aggregates are banned! No, just consider the implications of designing such a business vault and the potential for ***corrections***! Business vault is not where you ***conform*** column names to how business would like to see it! Don't forget although it is not raw vault, business vault still requires many join conditions to be executed in order to return the data from the data vault! Business vault serves as the ***decoupled*** derived intelligence based on raw vault.
Business vault will have the same data vault ***metadata tags*** as raw vault, and here is where the integration comes in. A business vault ***record source*** should be the name of the derived ***business rule name*** and its ***version number***. Yes, this ensures ***business rule lineage tracking*** through your business rule version tool of choice. That means a change record can occur in business vault when,

a) a change in raw data the business vault artefact is based on, and/or
b) a change in other business vault the business vault is based on, and/or
c) a change in a combination of business or raw vault the business vault is based on, and/or

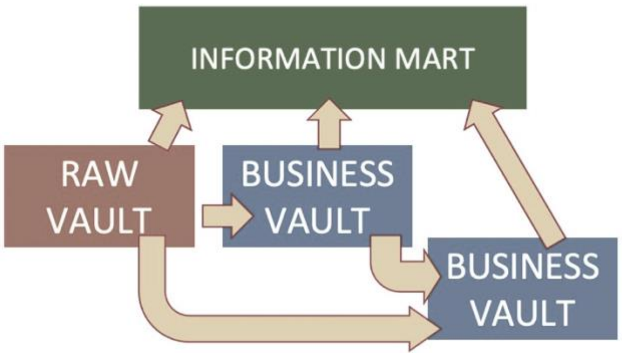


d) an update to the business rule (derived code) the business vault is based on

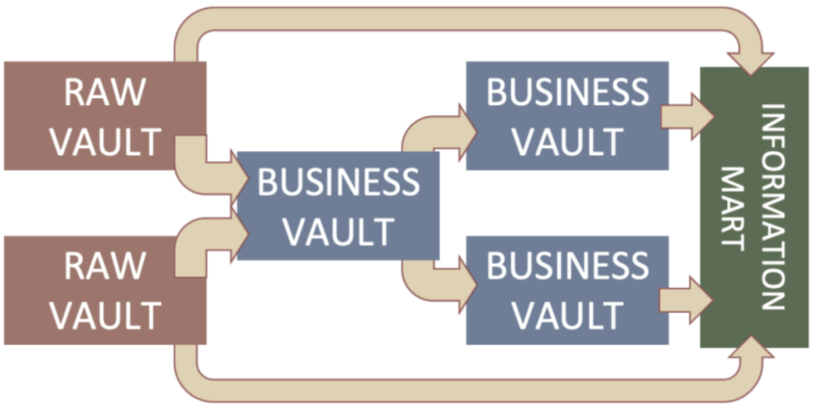
As we discussed business vault output can be delivered by reusing raw vault artefacts, that means we can build

- a ***business vault link*** - often used to represent the business view of the unit of work (which may differ to how a 3rd party application defines the unit of work), resolve a relationship based on derived business rules, testing business rules before persisting them; and all may include ***dependent child keys***;
- a ***business vault satellite*** - derived descriptive output stored against a business entity or relationship from simple case logic to statistical outcomes; this may include dependent child keys as well;
- a ***business vault multi-active satellite*** - if your output needs a derived ***SET*** of outcomes against a business entity or relationship;
- a ***business vault effectivity satellite*** - tracking driver to non-driver relationships not available in the unit of work coming from raw data;
- a ***business vault status tracking*** - deriving a status on a business entity or relationship by a derived business rule like entity or relationship ***aging***

How do I build one?



Define the derived rules that need ***auditability***, other ***soft rules*** can be integrated into the ***information mart layer***. Deployed as physical tables you can start to see how this method of business vault delivery has an eye on performance! As the data vault model is built out with more raw and business vault artefacts the same table structures will contain the same data vault metadata tags and the same ***key and index*** structures required for optimal performance. When performance lags additional table structures exist that in essence are ***not*** data vault tables (because they do not offer auditability like hubs, links and satellites do). They are the fabled ***point-in-time (PIT)*** and ***bridge*** tables! In the first diagram above they are the ***query assistance*** tables designed to take advantage of index-on-index joins, in Teradata terminology they act like ***index-joins***.



In the absence of indexes alternate information mart structures may be sought or potentially you may be looking at ***physicalizing*** and processing deltas to the information mart layer. Are we creating more layers in the data warehouse? Yes and no, the data vault looks to build the data warehouse through agility (it is ***non-destructive to change*** a data vault model with ***little*** or ***no refactoring***) and guarantee ***auditability***, the information mart in the data vault world is ***disposable***, if it breaks we have the full lineage and audit history in the data vault and we can easily recreate the information mart whenever it is needed!

In conclusion

Building your Business Vault extends your business capabilities with ***auditability***, ***agility*** and ***automation***, the same that is expected from Raw Vault. The only difference is where the data came from! That means your automation patterns are already there and your task as a data modeller is to decide which business entity or relationship the derived content will describe and how to generate that derived outcome, and of course ***don't forget about the data governance!***