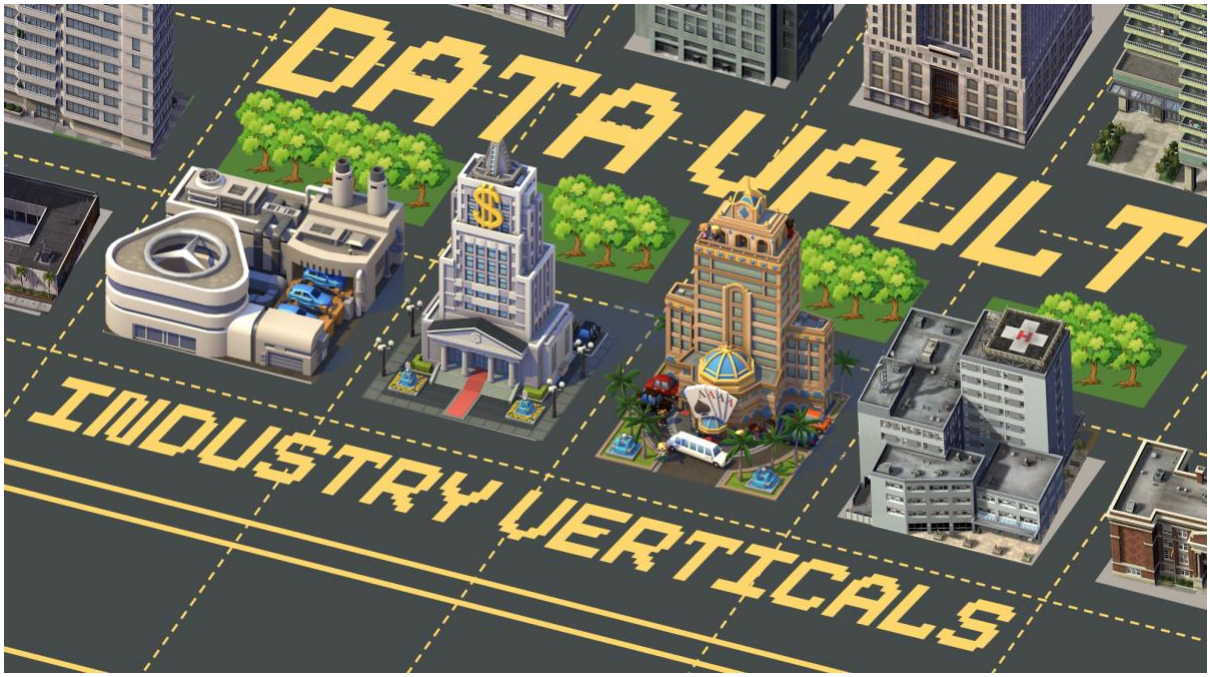
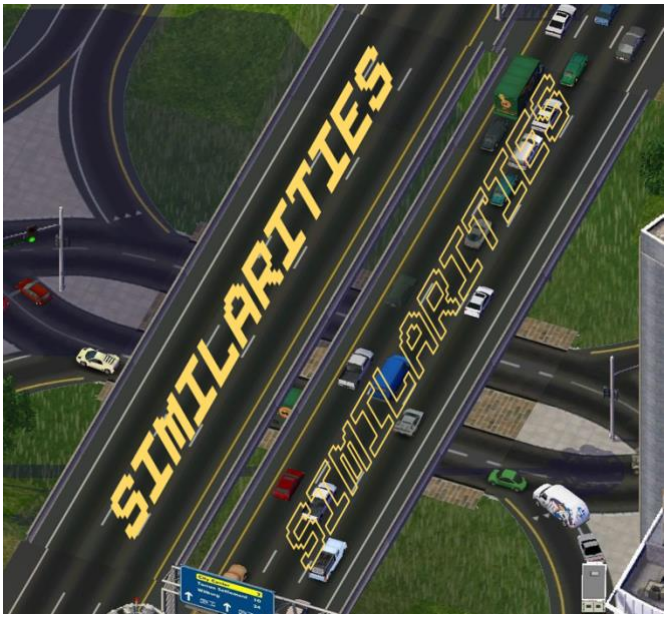


Data Vault Industry Verticals

In the last year I was asked to review a Data Vault interpretation of an industry data model. When in the field you know that there isn't really a thing called a *Data Vault industry model* because the Data Vault model itself lends its flexibility to suite *any* industry or data modelling layer pattern (David Hay's Enterprise Modelling Patterns). Upon review however I did see a lot of similarities to Data Vault calling out things like Agreements (aka contracts), Persons and Accounts amongst other **Business Objects**, it really resembled a Level 2 model… but there were some *oddities*. I list them both out here.

Follows the standard

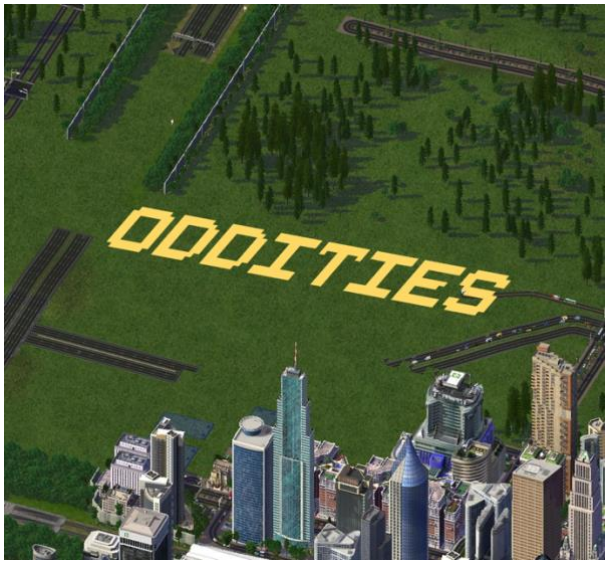


- The usual hubs, links and satellites and reference tables are separate and loaded as *slowly changing dimension* (SCD) Type 2 tables without additional links to those reference tables. Reference codes are treated as ***dependent-child keys*** in some of the link tables. Some of the reference tables are in fact *role-playing dimensions* as its code-value is reused in *multiple* links.
- ***Tenant id*** – based on a code tied to a department or organization the same data model *structures* could be used to host multiple tenants, but the content remains

only available to *certain* tenants. The extension of this leads to views and query assistance table structures that *filter by the tenant id*, and restricting the joins between hubs and links, hubs and satellites and links and its satellites ensures that you only return the records applicable to that tenant. *Why equi-joins matter?*

- **Delete_ind** – yes, the source notifies of record deletion, this is excellent and since the source supplies this it should remain with the rest of the satellite content. It doesn't mean we delete the data in the data vault, we simply record that the record was deleted in the source and by virtue of the record being the current active record in data vault we know it is deleted.

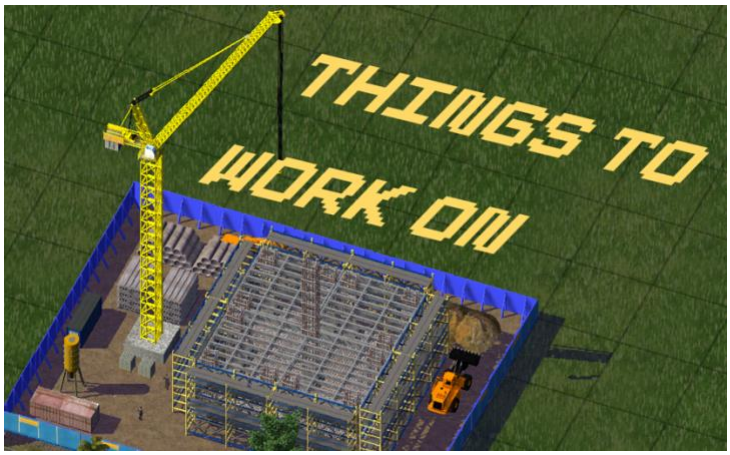
Deviates from the standard



- The data model understood the potential for business object *collisions* between integrated sources but chose to use the source-system-code as a *tie breaker*. This was *close* to being a business key collision code except that the model had it tied to the source that supplied the key ***by default***. *Folks!* This is a **problem** and an improper use of the *passive integration* concept. You see if source systems *do share* business keys, then you have illogically separated them in the loaded hub, meaning you need some *integration debt*… ahem… *additional logic* or tables to decipher this integration debt your data model introduced! (See: bit.ly/3xlFK0s)
- The data model understood the concept of a dependent-child key but decided to put that dep-key into the **hub table itself!** Now this is not a composite key but a sub-category key of the business object itself. Yes, it is a type-column, but it means nothing without its parent key. It obscures the meaning of the hub definition and should **not** be loaded into the hub!
- Raw satellite tables weren't raw… well the data model is an *industry data model*, and it deliberately chooses to mould the various sources into a single satellite table per information-type. The danger here of course is that should one of the sources experience ***schema drift*** that the mappings could need an update, or worse the grain no longer matches the rest of the other satellite table source. And what if both sources provide the same column name to be loaded to the same satellite but in fact have different interpretations of what gets populated in there? What if one of the sources does not supply a value for delete_ind? (See: bit.ly/3iEiHZB). Data Vault is sold as not having to refactor modelled data, this pattern inevitably would.

Further advice,

The modeller did seek advice about testing and querying their data vault-styled data model and here is a summary of the advice given:



- **Testing** should (like DV) be based on a *repeatable template* and it should be simple as well as *insert only*. To my mind there are two kinds of automated testing that serve a slightly different purpose.
    - **Pro-active testing** based on a *warranty* – test what you will load won't break the model, these are:
        - **Vertical testing** - does the act of hashing within the staged table create *integrity* issues between surrogate and business key? Simple test *within staging*, does the same tenant-id + business key collision code + business key(s) generate more than one surrogate hash key value. And does a single hash key value relate to more than one set of tenant-id + business key collision code + business key(s). The same check can be applied to the link-hash-key and the columns used to create the surrogate link-hash-key.
        - **Horizontal testing** – check that for the *same* tenant-id + business key collision code + business-key(s) combination does not generate a surrogate hash key *different* to that of the target data vault artefact, hub-hash keys and link-hash keys.
    - **Reactive testing** should run *forever* and ensures the *trust and integrity* of what is in the data model matches up, against the three data vault artefact types there are three such tests
        - Have I now got duplicates after the load? (needed if your db-platform does not enforce primary keys)
            - Hub duplicates by surrogate hash key.
            - Link duplicates by surrogate link-hash key.
            - Satellite by parent surrogate hash key and *current* load date.
        - Does the staged content reconcile to the target data model artefacts?
            - Are my keys loaded to the hub(s), one or many?
            - Are my keys loaded to the link(s), zero or many?
            - Are my attributes loaded to the satellite(s), zero or many?
        - Do my links and satellites reconcile by parent key to their parent entities?
            - Hub satellite to hub table.
            - Link satellite to link table.
            - Link table to one or more hubs.

    - **Periodic-integrity testing**, can I recreate my source file at any point in time? Don't run-this everyday but do run it on a schedule, say once a week or month and so on. Special consideration is needed for relationships within a link that can revert to a previously captured relationship and no business event date is supplied for that change. If this is true in your data model than the driver-key + effectivity satellite (or record tracking, or status tracking) is applicable (see: bit.ly/3oS4k70). By the way, *anyone who claims you can add an effectivity satellite to a hub table does **not** understand how to build one.*
    - **Data Quality**, there are two such DQ tests for this and they are
        - **Technical**, schema tests, checking for domain values and other expected value tests on an individual table, likely satellite attributes.
        - **Business process** tests, checking that the source application that map and automate business processes are meeting the required service level agreement (SLA).

      Both such tests can and should be persisted to **business vault satellites** and show as a measure of the source application health!
- **Querying**, of course Data Vault has many tables to join to, the two prominent query assistance (disposable) tables are:
    - Point-in-time (**PIT**) tables – used to simplify and improving query join performance around a hub or link.
    - **Bridge** tables – used to shorten the distance travelled between a hub at one end of the data model and a hub at the other end of the data model.

  Both are information mart focused and should be short and thin and filtered to suite the use case.

Conclusions

Begs the question, aside from the oddities that fails DV standards, could there be a place for industry hub tables…? Industry reference models are worked on and managed by Business Architecture and **Business Architecture** professionals, as a business recognizes Business

Objects these Business Objects are mapped as **Information Concepts** that fit and map the organization's business processes and **value streams**. So, come to think of it, it is not the data vault model that could be industry related but the hub table naming standards themselves, because what is it that uniquely identifies a business object? **The Business Key**.

More to come soon!

The views expressed in this article are that of my own, you should test implementation performance before committing to this implementation. The author provides no guarantees in this regard.