Data Vault
Data Modeling Specification v 2.0.2

Focused on the Data Model Components

Abstract

New specifications for Data Vault 2.0 Methodology, Architecture, and Implementation are coming soon... For now, I've updated the ***modeling specification*** *only* to meet the needs of Data Vault 2.0. This document is a ***definitional*** document, and does not cover the ***implementation details*** or "how-to" best practices – for those, please refer to Data Vault Implementation Standards.

Please note: ALL of these definitions are taught in our Certified Data Vault 2.0 Practitioner course. They are also defined in the book: ***Building a Scalable Data Warehouse with Data Vault 2.0*** available on Amazon.com

These standards are ***FREE*** to the general public, and these standards are up-to-date, and current. All standards published here should be considered the ***correct and current*** standards for ***Data Vault Data Modeling***.

NOTE: tooling vendors: if you *CLAIM* to support Data Vault 2.0, then you ***must*** support ***all*** *standards as defined here.* Otherwise, you are not allowed to claim support of Data Vault 2.0 in any way. In order to make the public statement "certified/Authorized to meet Data Vault 2.0 Standards" or "endorsed by Dan Linstedt" you must make prior arrangements directly with me.

Table of Contents

1.0 Entity Type Definitions

1.1 Hub Entity

A UNIQUE List of business Keys.

Please NOTE: This is the original and authentic definition of a Hub. Nowhere in this definition have I ever stated that surrogates (either hashes or sequences) are required for a Hub to work or to be in existence, yet people insist on making this part of the base definition. It is NOT part of the base definition, never has been, and never will be.

*Below is a Particularly Labeled Hub Type. There are **no** sub-classes of Hub Entities.*

1.1.1 Stub Hub

A Stub Hub is simply a Hub with a unique list of Business Keys. It is applied as a place-holder for future expansion. It is a known business key, however the source system(s) may be out of scope for the sprint or iteration, or the Satellite details / descriptive data may be out of scope for the current build.

A Stub Hub is not a class of Hub, as it is an ***application of a label*** for a standard Hub Class.

1.2 Link Entity

Unique List of relationships (intersections) between two or more business keys.

A link is also known as: Transaction, Granularity Change, Hierarchical Relationship, Recursive Relationship. ***Links are NEVER temporally based, they never contain begin or end date attributes***, the ***one exception***: a non-historized link may contain a transaction date.

NOTE: Links ***can never*** be Hubs. Links are ***always*** associative in nature and should forever remain this way. This has to do with the basics of set logic mathematics.

Below is a list of Link Subclasses.

1.2.1 Hierarchical Link

A Hierarchical Link is a many-to-many recursive relationship structure, housing a unique list of associations across two or more business keys at different levels of data.

1.2.2 Same-As Link

A Same-As Link is a one (1) level deep hierarchy that maps similar terms (synonyms) of business keys to a single ***master*** term or business key.

1.2.3 Non-Historized Link

A Non-Historized Link is a flattened denormalized structure containing immutable facts. It can include the association keys, along with the immutable descriptive data. For example: an immutable transaction, immutable event, or immutable fact based data set that is never updated or altered.

1.2.4 Exploration Link (Deep Learning Link)

An exploration Link or a ***deep learning link*** is a standard link (see definition of a Link entity above) built for a specific business purpose. The data set it houses generally is created by business rules. Therefore, the exploration link is a ***throw-away*** link. It can be built and discarded at will without affecting auditability.

Its' typical use is to contain two specially computed columns: strength and confidence. Each instance (row) of data is measured by its' neural network correlation. Strength of the relationship should tell you (in percentage from zero to 100%) how ***strong*** the correlation is between the associated data sets. Confidence rating (from zero to 100%) should describe how ***confident*** the decision-making engine is in its' strength prediction. Additional attributes may be added or attached to the Exploration Link in order to meet the needs of the neural network or deep learning algorithms.

1.3 Satellites

Delta Driven Descriptive Information (data that changes over time).

Satellites are driven by CDC (change data capture) and split by ***rate of change or type (classification) of data.*** Satellites contain the "data over time".

Below is a list of Satellite Sub Classes

1.3.1 Multi-Active Satellite

A Multi-Active Satellite is a structure built to house multiple active child records, where these child records have ***no stand-alone business key*** of their own. A prime example would be an employee with multiple active addresses at the same time.

Conditions that force a Multi-Active Satellite to be built include: bad quality source data, lack of metadata on the source feed, lack of stand-alone business key for the child records. In the example above, the addresses for each employee do not have a properly formed unique business key.

1.3.2 Effectivity Satellite

An Effectivity Satellite is a standard Satellite housing temporal views of a Link or Hub record. In other words, one or more sets of ***begin*** and ***end*** dates that may dictate the ***effectivity*** of the relationship (Link record), or of the business key (Hub Record).

Effectivity Satellites may also house ***deleted dates***, or other metadata markers that describe the parent records within or across timelines. NOTE: Physical ***updates*** are ***never*** allowed on this data. See the ***Data Vault Implementation Standards*** for more information.

1.3.3 System Driven Satellite

A system driven Satellite is a structure that is populated by hard business rules and is driven by systematic or system regulated processes. A Point-in-Time and a Bridge table are examples of system driven Satellites, they are built strictly for performance reasons, and generally are populated with primary keys and business key values from other Satellite structures. Another example of a System Driven Satellite is a ***record source tracking*** Satelite.

System driven Satellites can be dropped and re-created for specific business purposes without harming or affecting the underlying auditability of the data sets.

2.0 Other Table Classifications

2.1 Stand-Alone Tables

Calendar and time are prime examples of true reference tables. A stand-alone table may be referred to or grouped with Reference Tables in the architecture, however, they may also simply exist in the model. Stand-alone tables do not exhibit any delta's ever, nor do they house history, and typically their data is immutable, and globally agreed upon.

2.2 Reporting Tables

A Reporting Table is a flat-wide denormalized structure, and generally the data has been modified by ***soft*** business rules before loading. These reporting tables generally live in the ***information mart*** layer of the architecture and are leveraged for maximum performance. They can be imagined as a flattened cube, or an excel spreadsheet represented in one large table structure.

2.3 Point in Time and Bridge Tables

Point in time table is a System Driven Satellite. It is comprised of primary key values and business key values from a single Hub, and that Hubs' surrounding Satellites. It may also be comprised of a single Link and that Links' surrounding Satellites. It is a snapshot table populated with snapshot-based records of keys and key structures. It provides equal-join capacities to view based Dimensions and view based Facts. It is built for performance of the queries in getting data out of the Raw Data Vault. They are based on the principles of Join Indexes as written by Teradata, only the point-in-time structures can be implemented on ANY platform. Because Point-in-Time tables live within the Information Mart logical construct they can also house computed fields, and / or additional temporality (such as begin and end dates that have been computed for business purposes).

The Bridge Table is a combination of primary keys and business keys spread across multiple Hubs and Links. They can be thought of as "base level Fact Tables". They provide a snapshot of key structures and are generally not temporal in nature. That said, because Bridge Tables live within the Information Mart logical construct, they can also house computed fields, and / or temporality.

2.4 Reference Tables

Reference tables are a logical collection of code and description lookup structures. They utilize ***natural*** business keys, and are constructed from standard Hub, Link, and Satellite entities. Resolution occurs through queries at run time. They do not house nor require foreign keys. In general, the codes (natural keys) are found housed in the Satellites as they typically describe other keys or other relationships.

2.5 Staging Tables

Staging tables are a logical collection of structured data sets. They are generally applied to house data that is loaded to a particular database platform. Staging tables have two main purposes: a) to provide performance with bulk loaders, b) to provide parallelism on loads to the Data Vault (raw data warehouse) downstream.

On occasions where bulk loaders cannot compute the system fields on load, a ***second level staging*** table may be built. The second level staging table then houses the primary key of the staging record, followed by the computed system fields (such as Hash Keys, Hash Differences, Load Dates, and record sources).

Note: for details on the loading process see the *Data Vault Implementation Standards* guidelines.

2.6 Landing Zone Files (in NoSQL environment)

Landing Zone Files are simply files that have been loaded or dumped in to a Hadoop base environment, or other alternative ***metadata managed*** storage environment. A metadata managed storage environment has the following characteristics: parallel access, partitioning, sharding, and possibly sub-partitioning of the data, where the metadata is registered with an access engine (similar to Hadoop).

3.0 Common field elements

Common field elements are SYSTEM DRIVEN, and SYSTEM Managed – they do *NOT* fall under the scrutiny of an audit. They are generated fields on the way IN to the target (stage, data vault, or star schema) and are necessary for assisting in the traceability of individual fields.

3.1 Hash Key (Optional ** see note below)

A Hash Key is a ***deterministic value*** produced by a given hashing function. The requirement is to leverage the business key attribute(s) as input and produce a deterministic output. Due to the nature of deterministic calculation data can be loaded in parallel without any foreign key lookups (which is what Sequence identifiers require in order to resolve parent child relationships).

The purpose of the Hash Key ***is not*** to be encrypted and ***not*** to protect the data. However there are a series of encryption functions which produce hash-like values that ***may*** be utilized. This is particularly helpful when the clear-text data cannot leave physical boundaries or data stores (either in country, or on premise) due to security and privacy regulations.

Due to the deterministic nature of Hashing, most hashing algorithms can be applied for average equal random distribution across multiple partitions or shards. In other words, load balancing the data sets.

NOTE: Hash Keys are ***OPTIONAL*** on ***platforms like Teradata*** that support internal hashing of natural business keys. If a platform can function and hashes the natural keys (including potentially long text fields) internally, then an externally declared Hash Key is no longer required. *** Remember, the core definition of a Hub is: Unique List Of Business Keys ***

NOTE 2: Some Business keys (not necessarily natural keys) are sequence numbers, for example: Invoice Number, and will never change regardless of the business position. It is possible in these cases to ***avoid*** using Hash Keys, and simply leverage the source business key in its native format. That said, even in these cases – a ***sequence number surrogate key*** is ***not recommended.*** In fact, sequence number surrogate keys have been deprecated due to their inability to scale. ** See the notes below for further information **

NOTE 3: Hash results should be stored in ***binary format*** in platforms that support it, and in ***big number*** format in platforms that support it. Java, and platforms like SnowflakeDB support number(32) unsigned. Which means converting MD5 output to a number(32) unsigned is allowed, and encouraged. In SnowflakeDB use MD5_Number(x) to convert.

NOTE 4: IF a hashing value will be utilized then it is required to design and document a hash collision strategy (see Wikipedia for birthday problem, and hash collision discussions).

NOTE 5: IF Business Keys are chosen, then a strategy for dealing with multi-field joins must be devised, particularly to overcome multi-field join performance issues. Generally this means concatenating multiple business key pieces (composite pieces) together to arrive at a single field join. *See Data Vault Implementation Standards for more details.*

3.2 Business Key (Required)

A Business Key is defined to be the same semantic grain and same semantic meaning across multiple lines of business. If the grain is different, or the semantic meaning changes (regardless of the value), then separate Hub structures are created.

A Business Key ***may be defined*** as a source system surrogate identifier (sequence number from a source system). This happens when the value is shown to the business users, it is at that point that the business begins to leverage the value as THE unique identifier for the records.

NOTE: in a *perfect* world, we would construct Hub structures with JUST the business key which in effect would turn the Hubs in to indexing structures. Again, see Data Vault Implementation Standards for best practices as to how, when, and why to utilize business keys (instead of hash keys or EDW sequences) in different environments.

3.3 Hash Difference (Optional)

A Hash Difference is a computed field value based upon concatenation of descriptive attributes applied to a satellite and pushed through a hashing function. Instead of comparing each column (column by column) to determine a delta, the hash difference attribute can be compared. If they differ – a delta for the Satellite has been found.

This particular field is not necessary in database engines such as Teradata – due to the massive block size and high parallelism of the query engine, Teradata can compare many columns just as quickly as a predetermined Hash Difference column. That said, most other platforms benefit (performance wise) from utilizing a Hash Difference for comparison and delta checking purposes.

3.4 Load Date Time Stamp (Optional)

The Load Date Time Stamp is the ***date and time*** of insert for the ***record.*** Obviously in data management platforms like Hadoop HDFS, no such concept of ***record*** exists. Therefore, this field is now optional. (see the note below)

An attribute in the Hub, and Link – part of the ***primary key*** in a ***relational Satellite***. This is the date stamp of the date/time that the data was loaded into the database. This is stamped this way for consistency of information across the ***relational*** database. The Load Date Time Stamp ***is never*** a part of the primary key of a Hub or a Link.

In a ***real-time*** solution the load-date may ***not*** be dependable unless the transactions can be guaranteed to be inserted in the order in which they were created. ** See APPLIED DATE below **

NOTE: It is REQUIRED for Relational Database Engines, however it is not possible to apply to Hadoop Files. In reality, Hadoop ***stamps*** the file at the file level in HCatalog with the date and time the file was "ingested". This is as close as the data can get to a Load-Date. Hence it is now ***optional***.

2.3 Record Source (Required)

This is the source system that generated the information, it is mechanically stamped when the information is loaded to the database. Used when there is no meta-data project in place to trace information back to the source. It provides tracability of every record at a granular level back to the source system. While optional, it is implemented by 98% of the customers today.

2.4 Update User (optional)

This is there to track DBA level modifications to the data. It is optional and should be in another metrics tracking area. Typically found in the Metrics Vault component. Update User should never appear in any of the data sets inside the Raw Data Vault, as it requires a physical hard update against any records that are being modified. Physical raw updates ***do not scale (in performance numbers)*** to massive data set sizes.

2.5 Update Time Stamp (optional)

This is another DBA tracking field. It also is optional and should be in another metrics tracking area. Typically found in the Metrics Vault component. Update Time Stamp should never appear in any of the data sets inside the Raw Data Vault, as it requires a physical hard update against any records that are being modified. Physical raw updates ***do not scale (in performance numbers)*** to massive data set sizes.

2.8 Extract Date (Optional)

This has proven to be beneficial if included in the module. There are times at which knowing what the extract date is, helps. Extract Dates should always be additional attributes in Hubs, Links, and Satellites. They should ***never*** be applied as load dates.

2.9 Applied Date Time Stamp (OPTIONAL)

Applied Dates are date time stamps which can be manipulated to indicate ***when*** in the business timeline the data ***applies.*** Applied Date Time Stamps can appear in Hubs, Links, and Satellites. In other words, instead of setting or changing load dates, instead of leveraging extract dates, the Applied Date can be set according to business needs. In the case of real-time feeds, an Applied Date may be the ***transaction creation date***. In the case of historical batch feeds, it may be set to the "back-dated date" that represents when in history the data applies.

Applied dates ***are not*** the same as load dates. Applied Dates are to be leveraged by SQL queries, aggregates, and business applications.

4.0 Hub Rules

4.1 Business Key

A Hub must have at least one (1) business key driven field. (See definition of Business Key Above)

4.2 Business Key to Surrogate

A Hub's business key must be one to one with the surrogate *IF a surrogate (hash key or sequence) is utilized

4.3 Multiple Distinct Business Keys

A Hub cannot contain a composite set of ***multiple*** business keys that are stand-alone. For instance: Account Number and Invoice Number, these two business keys are clearly stand-alone keys, and must be modeled each in their own Hub with a Link association between them.

4.4 Hub Satellite (Optional)

A Hub ***should*** support at least one satellite to be in existence, Hubs without Satellites usually indicate "bad source data", or poorly defined source data, or business keys that are missing valuable metadata. Hubs without any Satellites have ***no context***.

4.5 Hub Composite Key Attributes

A ***single*** Hub ***business Key*** can be composite when: two of the same operational systems are using the same keys to mean different things AND these keys collide when integrated back together again.

Please be aware: BAD DATA CAUSES BREAKS IN THESE RULES – THESE ARE GUIDING PRINCIPLES. Exceptions to this rule should not happen (but do), also be aware, bad architecture in source systems causes breaks in these rules too.

4.6 Hub Business Key Stands Alone

A Hub's business key must stand-alone in the environment – either be a system created key, or a true business key that is the single basis for "finding" information in the source system. A True business key is often referred to as a NATURAL KEY.

Alternative source business key ***may*** be a sequence id. The minute a ***sequence*** on the source system is shown to the business user, is the minute it becomes a ***business key*** on the source system. It is at this point it qualifies as a business key in the Hub.

4.7 Hub Load Date Time Stamp

A Hub's load-date-time stamp or observation start date must be an attribute in the hub, and ***never*** part of the hub's primary key structure. A Hubs' Load Date represents the ***first time*** the data was inserted to the Hub.

4.6 Hub Record Source

A Hub's PRIMARY KEY cannot contain a record source. Hub Record source indicates the ***first system*** to insert the data.

5.0 Link Rules

5.1 A Link is an Intersection of Hubs in a Venn Diagram

A Link must contain ***two (2) or more*** imported Hub primary keys. A Link ***can never*** depend on another link.

5.2 Hierarchical Link Relationship

A Link can contain two keys imported from the same hub for a hierarchical relationship or rolled up relationship. A HIERARCHICAL representation of a relationship or aggregation across a single hub's key, migrated in exactly two times.

Any further hierarchy is broken down into two migrations, this way no limitation is placed upon the hierarchy, and the Link is NOT playing a role-game which is dangerous. Also, a Hierarchical Link must contain at least one Satellite to indicate effectivity of the relationship (start and end dating of the hierarchical relationship.

5.3 Link Load Date (Required)

A Link's load-date-time stamp or observation start date must be an attribute in the link, and ***never*** a part of the links' primary key structure.

5.4 Link Composite Key

A Links composite key must be unique (A unique business key). The Links' composite key is comprised of all business keys or hashes from parent Hub Structures.

5.5 Link Surrogate Key (Optional)

A Link may contain a surrogate ***hash*** key (if the composite is too large, or the database doesn't work well with natural keys). If a Surrogate Hash Key is built, it will be ***the only*** attribute in the primary key of the Link. Note: Non-Historized Links (transactional links) do not require a Link Surrogate Key since they never have any child Satellites built.

5.6 Link Granularity

A Links' granularity is determined by the number of imported Hub parent keys.

5.7 Link May be Defined as

A Link is a transaction, or a hierarchy, or a relationship.

5.8 Link Satellites are Optional

A Link may have zero or more Satellites attached.

5.9 Links and Temporality

Links ***never*** contain begin or end dates. These are always assigned to exist in Satellite Effectivity. The sole exception: ***non-historized Link.*** (See the definition of Link Section 1.2 above)

5.10 Link Granularity

A Link must be at the lowest level of granularity for tracking purposes.

5.11 Link Data Representation

A Link must represent at most, 1 instance of a relationship component for ***ALL*** time.

5.13 Link Driving Key Rule

In a Link, the ***consistent part of the key*** is the primary driver for the relationship.

6.0 Satellite Rules

6.1 Satellite Parent Key

A Satellite ***must =*** have at least one Hub or Link primary key imported. A Satellite can ***never*** generate its' own primary key. A Satellite can ***never*** maintain or create its' own surrogate sequence key or its' own hash key.

5.2 Satellite Single Parent ONLY

A Satellite cannot be attached to more than one Hub or Link parent structure. A Satellite ***can NEVER EVER be snow flaked*** (in accordance with snowflake definitions provided by Kimball Star Schemas)

5.3 Satellite Load Date Time Stamp (Required)

A Satellite MUST have a load-date-time stamp (observation start date) as a part of its primary key.

5.4 Satellite Sub-Sequence (Optional)

A Satellite may contain a sub-sequence identifier or and ordering identifier as part of the primary key for uniqueness. This additional sub-sequence can be a microsecond counter, or an ordering attribute (as in the case of a multi-active Satellite record)

5.5 Satellite Record Source (Required)

A Satellite must contain a record source attribute for data traceability.

5.6 Satellite Descriptive Elements (Required)

A Satellite must contain at least one descriptive element about the Hub or Link to which it's attached in order to be valid. This rule includes effectivity and system driven Satellites.

5.7 Business Satellite Data

A Business Vault Satellite may contain system generated or aggregated attributes as a result of ***soft rule*** calculations.

5.8 Satellite Purpose

A Satellite's purpose is to store data over time.

5.9 Satellite Reference Table Access

A Satellite may contain a natural key (i.e. code) to a stand-alone code/description table. FK's to reference tables are ***implied*** – ***never physically implemented***. Indirect references to date time calendar table, or geography is acceptable. These foreign keys are NOT to be represented within the physical data model.

5.10 Satellite Split or Merge Actions

A Satellite may be split or merged at any time, as long as NO HISTORICAL VALUE is lost, and NO HISTORICAL AUDIT TRAIL is lost. See the ***Data Vault Implementation Standards*** for rules and processes around how-to execute a split or merge.

7.0 Naming Conventions

Naming conventions are enforced in order to meet the needs automation and standardization. Without naming conventions, the models will get out of hand and become unmanageable. Data Vault models which do not follow documented naming conventions cannot be automated.

The conventions proposed below are recommendations, not requirements. What is required is to create and maintain a naming convention document for every Data Vault Model that is designed.

6.1 Entity Naming Conventions

Entity Type	Prefix or Suffix
Hub	**H, HUB, HB**
Link	**L, LINK, LNK**
Satellite	**S, SAT**
Stage	**STG**
Hierarchical Links	**HL, HLNK, HLINK**
Same-As Links	**SAL, SALNK, SLNK**
Point-in-Time	**PIT, PT**
Bridge	**B, BRDG, BRG**
Business Hub	**BH, BHUB**
Business Link	**BL, BLNK, BLINK**
Business Satellite	**BS, BSAT, BST**
View	**V**
View Dimension	**VDIM, VD**
View Fact	**VF, VFCT**
Fact	**FCT, FACT, F**
Dimension	**D, DIM**
Report Collection	**RPT, RC**

6.2 Field Naming Conventions

Attribute Type	Prefix or Suffix
Record Source	**R, RSRC, RS**
Sequence ID	**SQN, SEQ**
Date Time Stamps	**DTS, DTM**
Date Stamps	**DS, DT**
Time Stamps	**TMS, TS, TM**
Load Date Time Stamps	**LDTS, LDDTS, LDTM**
User Watch Fields	**USR, U**
Occurrence Numbers	**OCC, OCNUM, ONUM**
Load End Date Time Stamps	**LEDTS**
Sub Sequence	**SSQN, SSQ, SUBSQN**
Applied Dates	**APPDT, ADT, APDT**
Hash Keys	**HK, HashKey, HKEY**
Hash Differences	**HD, HashDiff, HDIFF, HDF**

10.0 DEPRICATED RULES

These rules have lost their ability to be effective in the Data Vault 2.0 Landscape due to big data (volume), platform issues, velocity of data arrival, or variety of data sets. Therefore, these rules are deprecated / removed from the standard, and should ***never*** be utilized going forward.

10.1 Hub and Satellite Sequence ID (DEPRECATED)

WARNING SURROGATE SEQUENCE BASED IDENTIFIERS SHOULD NO LONGER BE UTILIZED. Why has it been deprecated? For the following reasons:

- It requires lookups on a row by row basis for resolution. These lookups cannot scale in to the hundreds of terabytes or petabyte ranges of data movement and integration. These lookups are ***sequential processes***
- ***Sequence ID's are non deterministic.*** Requiring "hybrid" or geographically split platforms to perform lookup operations across multiple database instances. Lookups across geographic splits, or hybrid (in-cloud / on-premise) solutions are not fully scalable, and do not perform to the expectations set by the business today.
- Lookups across multiple country boundaries by clear-text business keys are non-compliant with data protection and privacy regulations, which make the lookups infeasible to utilize.
- Being non-deterministic, means they are not good at: uniquely identifying (consistently and deterministic identifying): text, documents, video, audio, or image files ***in parallel environments*** at the same time.

10.2 Satellite Load End-Dates (DEPRECATED)

Load End dates are no longer allowed in the RAW data Vault, as they require physical updates which a) are not supported by all platforms, b) do not scale properly over hundreds of terabytes of data. Also, load end dates are ***not consistent*** in real-time inflow of data sets because data may arrive ***out of order*** from when it was created – causing multiple problems with load end dates.

Note: If End-Dates are truly necessary, use Lead/Lag functions to compute them, and instantiate their values in the Point in Time and Bridge Table structures. See Data Vault Implementation Standards for more information.

10.3 Hub and Link Last Seen Dates (DEPRECATED)

Last seen dates are no longer allowed, as they require physical updates which a) are not supported by all platforms, b) do not scale properly over hundreds of terabytes of data.