An Introduction to Data Vault

The Data Vault is defined as a detail oriented, historical tracking and uniquely linked set of normalised tables that support one or more functional areas of business. It is a hybrid data modelling approach encompassing the best of breed between 3rd normal form (3NF) and star schema (Dimensional), both of which are traditional techniques to design Data Warehouse platforms. The technique is officially titled as the 'Common Foundational Integration Modelling Architecture', but is far better known under the name Data Vault.

The keywords 'detail oriented', 'historical tracking', 'uniquely linked' and 'normalised' are examples of the technique being geared towards Data Warehouse use-cases.

- Detail oriented aims at the principle of storing data at the lowest available level of detail (grain)
- Historical tracking supports the concept that all changes that occur should be captured (tracked) in the Data Warehouse
- Uniquely linked refers to the modelling of relationships as they (logically) are between business concepts
- Normalised aims at the separation between core business concepts and keys (Hubs) and contextual data (Satellites)

These are very similar to the well-known Data Warehouse definitions of 'subject-oriented', 'integrated', time-variant' and 'non-volatile' as defined by Bill Inmon. Data Vault is a Data Warehouse approach, and the technique fully adheres to this more generic definition. In fact, Bill Inmon prefers Data Vault as the approach to model the core Data Warehouse components in the Data Warehouse 2.0 methodology.

Data Vault was first published in 2002 by Dan Linstedt in The Data Warehouse Newsletter (TDAN) with of five articles covering the data modelling fundamentals. These can be found here: http://tdan.com/data-vault-series-1-data-vault-overview/5054. During its lifecycle, the technique has evolved and updated standards were released in 2013 as part of Data Vault 2.0. (DV2.0).

These days Data Vault, and especially DV2.0, has evolved into a broader methodology centered around the original hub-and-spoke Data Vault modelling technique but also covering integration with concepts such as Six Sigma, CMM and Agile. The move from original Data Vault to DV2.0 has also defined better data integration (Extract, Transform and Load or ETL) patterns and standards which allow for a more straightforward and better defined implementation. Especially the relationship with Agile is often mentioned; hub-and-spoke type approaches allow for incremental extension of the solution which is an excellent fit with Agile frameworks such as Disciplined Agile Delivery (DAD).

As indicated in this section, Data Vault belongs to the family of hybrid modelling approaches which also include Head and Version modelling, Anchor modelling, Focal Point modelling and various related ways of thinking. From this family of similar approaches the Data Vault is by far the most popular and adopted globally. The shared characteristic of these techniques is to focus the information (data) modelling around the definition of core business concepts, and to 'separate concerns'.

Separating concerns

Data Vault focuses on separation of concerns as a way to enable flexibility when handling business requirements. Separation of concerns means that dedicated and specific tasks that a Data Warehouse must perform are broken up into smaller atomic processes with defined places in the overall architecture. This is almost opposite to the traditional approaches which invariably feature complex ETL processes which 'have to do a lot' in a single operation.

Traditional 2-tiered solution designs typically load data into a staging area that resembles the destined (Dimensional) model (stage 1) and then merge this with the already existing information (stage 2). ETL that executes this logic has to support a wide variety of functions to deliver this, such as for example:

- Implementing a variety of business rules
- Integrating multiple sources
- Managing changes over time
- Key distribution
- Defining structure, hierarchies
- Balancing performance
- Granularity / aggregation
- Delta detection / change data capture

As most of these functions need to be implemented into a single ETL process, this results in Data Warehouse solutions that can be difficult to extend and suffer from increasingly complex interdependencies. The additional effort required pushes out load times and creates gaps with (changing) business requirements which can cause critical issues. This is especially true when requirements change over time as it is increasingly difficult to make the required changes, at least in an acceptable timeframe. The bigger the system gets, the more the gap with the (evolving) intent of the solution is exacerbated.

Data Vault addresses these issues by recognising that Data Warehouse management and business logic should be separated to achieve maximum flexibility and ease of maintenance. While a typical Dimensional Modelling ETL process must handle almost all of the previously mentioned functionality, Data Vault breaks up these operations into atomic tasks. For example, key distribution is handled by the Hub entity, which embodies the central business concept or business key. Managing change over time is separated and embedded into the Satellite entity whereas handling relationships between business concepts (Hubs) is handled in Link entities.

Relationships in Data Vault (Link entities) are *always* many-to-many, regardless of how the source systems are modelled. This provides future proofing when the underlying systems change. If (when) a one-to-many relationship between Customer and Policy changes into many-to-many this has little impact on a Data Vault model whereas impacts on traditional 3NF or 2-tiered Dimensional Model solutions can be severe.

Business rules, or interpretation of data, is layered on top of the core Data Vault model, which effectively separates the Data Warehouse housekeeping from the business rules. One of the key principles of Data Vault is that requirements change, especially during the early phases a project. Separating the delivery of information (through business rules) from storing and managing underlying data enables easier managing of different perspectives. Data Vault recognises that the truth is subjective, and may even be different for separate consumers of information within the same company. As opposed to the 'single version of the truth', Data Vault approaches favour a single version of the facts: transactions. It is for this reason that Data Vault is sometimes referred to as the Corporate Memory.

While DV2.0 provides more guidelines as to how to deliver information using Information (Data) Marts, the core focus on Data Vault is on the back-room: collecting, integrating and recording data from online transactional processing (OLTP / source) systems. By having a solid and robust back-room process that 'just works', more emphasis can be placed on iteratively liaising with the business to get the best possible outcomes. For this reason, the Data Vault slogan is '100% of the data 100% of the time'.

In the ideal world, it contains any transaction that ever occurred even if the original source system does not contain this information anymore.

Core entities of Data Vault

Data Vault divides the world into three types of information: Hubs, Links and Satellites.

Hubs are business entities; these are 'things' that drive the business at a specific level of granularity. They are the concepts around which your model revolves; something that the business uses to track, locate, and identify information. Each of these entities will be defined around a business key which is how the 'thing' in question can be identified by the business. This does not necessarily reflect how systems capture this information though. Business keys, stored in business entities, are the foundation of the Data Vault around which the rest of the model is built up.

Examples of business entities are Customer, Policy, Claim, State, Intermediary and Risk. The Hub entity contains the distinct (unique) list of business keys for a determined type, for example the distinct list of customers would be available in the Customer Hub. What constitutes a business key is specific for each individual organisation; what is important for one organisation is not necessarily important for the other. For instance, some organisations require to model our types of employees as individual Hubs, whereas for other organisations simply an 'Employee' Hub would be sufficient. Getting the granularity right and selecting the correct business key is a pure information modelling task, and requires understanding and liaising with subject matter experts.

Whatever the case, there is strictly no super-typing allowed! This means for instance it considered 'not Data Vault' to model a Party entity with circular references defining types (e.g. Employee, Company).

Links cover relationships; they are a representation of how the business entities interact with each other. The most common example of a relationship is a foreign key from table to the next, but relationships can also be modelled because it makes sense from a logical model irrespective of how this is available in source systems. Following the conventions of pure Data Vault modelling transactions are also relationships, although this is often modelled as a separate Hub / business entity.

Examples of relationships:

- Intermediary selling a Policy
- Risk covered within a Policy Term
- Claim against a Policy Risk Cover
- Risk Cover Payment Transaction for a Risk Cover within a Policy (header) made by a Customer

Satellites provide context, the descriptive attributes; any information about a business entity or about a relationship between entities. It is the descriptive data that provides the 'meat' of the model. Contextual information is also anything that changes over time, so it is what provides the EDW model with a time variant aspect. When the context pertains to a business entity, it is descriptive information specifically about that entity and how it changes over time. When the context pertains to a relationship ('Link-Satellite'), it describes the interaction between entities and how it changes over time.

Examples of context:

- The name, date of birth of a policy holder
- The description of a product or service
- The date and time when an incident was reported

Data Vault 2.0

DV 2.0 takes the established entity types and places these into a broader and more specified solution design / architecture. This includes better specification of how information can be delivered into Information Marts (it's information, not data at this stage!) with a strong message on how to iteratively work from data to information together with business counterparts. As part of the launch of DV2.0 the concept of Hash Keys was also introduced. Hash Keys are broadly accepted as a better way to manage key distribution as many of the loading dependencies are removed, allowing for a far greater degree of parallelism and the ability to create cross-platform relationship.

Hash Keys use standard cryptographic hash algorithms such as MD5, MD6 or SHA to apply one-way encryption on the business key. Because this function is deterministic it can be applied to completely separate loading processes to achieve the same outcome. It enables for instance to load contextual data (Satellites) before Hub processes, which is in a way comparable to loading Fact tables before Dimensions in the (Kimball) world of Star Schemas.

Another key component of DV2.0 is clearer support on integrating data across different platforms, including Massively Parallel Processing (MPP) support for concepts such as easier key distribution across nodes and co-location. Since the output of a given hash algorithm in C#, Oracle or Python produces the same result it becomes possible to span Data Warehouse solutions across multiple environments.

Data Vault into the future

Data Vault provides an easy-to-explain and flexible way to tackle complex issues for bringing data together, and technology is getting to a point where we can move towards a more logical way of working with a core Data Warehouse model (hence the term 'Logical Data Warehouse'). An approach such as Data Vault provides a meaningful way to apply schema to the raw data in a way that is easily communicated and abstracted to subject matter experts and designers.

This metadata is the real Intellectual Property. With good metadata management and adoption of concepts such as ETL generation you can easily change between platforms, technologies and approaches. These days it is more a question on how best to deploy things for a certain platform than defining a generic design pattern: how to make things work as opposed to focusing just on scalability and performance for a single type of environment. Different environments have different characteristics that you need to address but the core principles described in this paper remain the same.

The best way to look at Data Vault holistically is to support the journey to perfecting data management with flexibility. It is perfectly doable to start developing on a relational database management system such as SQL Server or Oracle and then move components of the architecture to Hadoop. Some organisations choose to virtualise some areas either in the core model or the source layer. This is all possible as long as each element of work improves and contributes to your design and metadata.

From this perspective, discussions on what technology or idea works best or doesn't are less relevant. It is OK to start small one way and then evolve into another direction, which may even be cross-platform. For example, there are real-life examples of projects starting in C# while waiting for database access, moving to SSIS to T-SQL and settle using distributed processing – it is all more of the same, a generation template, but the metadata fundamentally remains the same.

This also is true for deciding where and how data needs to be persisted. An example is implementing Data Vault by deploying physical Hubs supported by virtual Satellites using the Persistent Staging Area concept. This can work well up to a certain point when performance is impacted, in which case some tables can be persisted or moved elsewhere. Data Marts are typically persisted, but refreshed on certain intervals depending on requirement although this itself can be toggled between virtual and physical at any time. Using the above mentioned concepts, instantiation should be treated as a checkbox, an option.

By managing your metadata properly, you can achieve great flexibility. The key uniting factor is keeping a copy of raw data somewhere, which is the underlying idea behind the Persistent Staging Area. This flexibility also relates to model output itself, down to certain design decisions in your Data Vault implementation – as long as you keep improving with the occasional step back you'll get there in the end. This is why even in the world of data virtualisation, and the Logical Data Warehouse the Data Vault model still makes sense: applying and defining data and context around you business' core concepts as embodied in the Hubs enables this.

In short, it is all flexible but the key ingredients that are always required are:

- Archive of raw data (again either delta or full file)
- Core Data Warehouse model (Data Vault)
- Interpretation / marts / business logic
- ETL metadata

The platform where you deploy things to requires certain decision to be made on how to implement these things relative to the environment, but this is where our expertise as specialists is relevant. Scalability is really about deciding the best implementation approach; the pattern for a specific technology and scenario. We still need to make it work, but technologies come and go, and database vendors will also find ways to improve and incorporate the successful use-cases of for instance Hadoop into their database management platforms.

So, is a Data Vault model still needed at this stage? My personal view is it would be, as ultimately the step from raw data to interpretation (marts) is usually too big, and Data Vault creates a clear separation of concerns and understandable layers of abstraction. With this, it supports a gradual increase in understanding about data and its interpretation in the wider business. This process allows teams to gradually tweak and perfect the true intellectual property: the model and ETL metadata.

This flexibility also manifests itself in that some groups of users do not necessarily require all data to be integrated. An example is the advanced analytics (data science) work that may require one-off external data sources. These activities can sometimes highlight parts of data what you would want or need to incorporate into a more structured design in due time. The recommended approach therefore is to capturing things into a Persistent Staging area and gradually go through the motions of information modelling into a Data Vault and beyond.

Even with the ever-increasing capabilities of technology, the core principles of Data Vault hold up as there is value in understanding the core business concepts and grouping context around these business concepts as an abstraction layer, a logical representation. The more diverse the implementation becomes, the more refined the model will be, and everyone's input will contribute to understanding how data relates together.

In this way, Data Vault is the ultimate cross-platform schema-on-read.