Data Vault and ‘The Truth’ about the Enterprise Data Warehouse

Roelant Vos – 04-05-2012

Brisbane, Australia
**Introduction**

More often than not, when discussion about data modeling and information architecture move towards the Enterprise Data Warehouse (EDW) heated discussions occur and (alternative) solutions are proposed supported by claims that these alternatives are quicker and easier to develop than the cumbersome EDW while also delivering value to the business in a more direct way.

Apparently, the EDW has a bad image. An image that seems to be associated with long development times, high complexity, difficulties in maintenance and a falling short on promises in general. It is true that in the past many EDW projects have suffered from stagnation during the data integration (development) phase. These projects have stalled in a cycle of changes in ETL and data model design and the subsequent unit and acceptance testing. No usable information is presented to the business users while being stuck in this vicious cycle and as a result an image of inability is painted by the Data Warehouse team (and often at high costs).

Why are things this way? One of the main reasons is that the Data Warehouse / Business Intelligence industry defines and works with architectures that do not (can not) live up to their promises. To date there are (still) many Data Warehouse specialists who argue that an EDW is *always* expensive, complex and monolithic whereas the real message should be that an EDW is in fact driven by business cases, adaptive and quick to deliver. It is a disconcerting thought that apparently the EDW is not understood by the Data Warehouse industry.

This white paper is the first in a series to convince the industry that this statement regarding the EDW is true and that it is a concept that is effective in any situation thus negating the need for custom architectures which often turn information management into an unmanageable mess. This series of white papers will attempt to give the EDW the positive image it deserves and will lay out the approach and techniques to achieve the best results.

The key element of the solution is **the adoption of an architecture that differs fundamentally from the common approach to Data Warehousing** and from what has been used to design Data Warehouses over the past 15 years. This architecture delivers -out of the box- on flexibility in both design and implementation, true resilience, being ‘future-proof’ (changes are a fact of life – especially in the central spot a Data Warehouse operates in) and offers complete traceability of data. As a conceptual architecture the focus is not predominantly on technology although this will be addressed to some extent in the specifications of the ETL Framework.
The true Enterprise Data Warehouse

The complete EDW architecture is composed of various functions, techniques and concepts that support these ambitions. Together they define the EDW that is truly consistent, adaptable and future-proof while delivering on the traceability and, most importantly, **works the same way in every scenario.**

When looking into the details of the architecture we will find that many of the common Data Warehouse concepts such as normalization or Type-2 stacking of historical changes are still present, and for very good reasons. But it is the careful (re)positioning and application of these concepts in the (reference) architecture that makes the solution much more robust.

It is here that the introduction of the **Data Vault** modeling technique as defined by Dan Linstedt makes a big difference. By positioning Data Vault as the core EDW ‘layer’ supported by a structured ETL Framework that defines the loading templates and control processes the following results are achieved (some of it new and some of it already well-established):

- Business Rules (including managing data quality) become the responsibility of ‘the Business’. One of the ambitions of the EDW is to enable the business to change these definitions without compromising the data or the design of the Data Warehouse. This means that the EDW will initially focus on managing data and applying Data Warehouse housekeeping properly before applying the business rules. The Data Vault technique specifies ‘hard’ and ‘soft’ business rules which define when certain types of transformation logic must be implemented;
- ETL is structured, consistent and is directly related to Data Warehouse concepts;
- Data Warehouse concepts are clearly separated (decoupled). This includes key distribution and management, history management, deriving data to information and providing structure and context (i.e. hierarchies, dimensions).
- True decoupling of applications that provide data (operational systems) and the Data Warehouse. Changes are a fact of life for the Data Warehouse, but proper design can significantly reduce the impact of these changes (and thus reduce maintenance).
- A complete audit trail, always. It must be possible to trace all data back to the source and how it was processed in the Data Warehouse. The purpose ranges from data validation (how did I get this number?) to strict compliance rules (i.e. BASEL).
- Full scalability, which manifests itself in many ways. Due to the modular design the Data Vault is extremely flexible (yet easy and consistent) when it comes to adding or changing data sources. It also enables the Data Warehouse team to change the table structure when database size becomes an issue to introduce a degree of normalisation without impacting the Data Vault concepts. A properly management Data Vault is indefinitely scalable.
- Flexibility in ETL scheduling. ETL templates are designed to be able to handle the most granular of intervals (near real-time). It is also ensured that ETL processes are defined as atomic steps that serve one designated purpose in the overall architecture. By introducing these requirements to ETL design the Data Warehouse team can structure workflows to suit every need. This includes introducing parallelism in loading, changing loading intervals and even mixed workloads. This is all done without compromising the core model or losing information of flexibility.
Ultimately the goal of the EDW is to provide a meaningful reporting and analysis or data mining structure that Business Intelligence software can access. The main difference with previous EDW architectures is that this solution can accommodate all types of structures (i.e. star schemas, snowflakes, 3NF) and even change the approach without losing data or with excessive maintenance overhead.

Does this sound too much to achieve in one go? It’s not! All this can be achieved combining the Data Vault with a fresh look at the role ETL processes should play in the Data Warehouse reference architecture. As stated before, it is all about redefining where well-known concepts should be applied and letting go of some cherished ones.

**Letting go of the Truth**

One of the most persistent concepts that haunt Data Warehouses to date is the ‘single version of the truth’. As the subtitle of this paper hints at it is worthwhile to rebrand this concept into the ‘**single version of the fact**’. Deriving or calculating a truth is subjective and typically only really ‘true’ for a certain recipient of the information whereas **facts** can be used to create any ‘truth’ depending on the recipient’s point of view.

To make things worse the perception of ‘truth’ can change over time rendering existing implementations difficult to maintain or even unusable. Data Warehouses which aim to deliver on this single version of the truth will see that, at some point in time, they cannot provide the information the users really need in a timely and cost-effective manner. Additionally interpreting data early in the process effectively ruins the audit trail which is a serious concern to the Data Warehouse team.

There are many reasons that this view on Data Warehouse design is so firmly established in the industry. For a long period of time the Data Warehouse was deemed the best solution to bring together information from different ‘silos’ of data (applications). But as the business changed the (IT) Data Warehouse could not keep up with the changing requirements and it turned out that there are always valid reasons to have different views on the meaning of the information. That is, as long as you can trace the interpreted information back to the same base values.

This idea alone is one of the fundamental changes when compared to existing architectures. It essentially means that you need to maintain a ‘pure’ or ‘raw’ repository of the data as it was received and ‘reroll’ the truth (current definition) from time to time when required.

The Data Vault has the perfect qualifications to be this repository.

Letting go of the Single Version Of The Truth means that the **Data Warehouse is no longer the integrated and pure collection of information for the enterprise** but it is closer to being a large storage facility for the company **data** (as an asset!) so that the various truths (views of the business) can be created at any time and in any format (typically Data Marts). For once, the Data Warehouse can truly match the user’s requirements.
**Data Vault**

Data Vault as a modeling technique is the perfect match for the new EDW architecture and provides the modular components and built-in audit trail to meet the ambitions. Due to the way Data Vault defines the entity types the EDW becomes a fully standardised and asynchronous environment where data can be added and loaded at any time. If you want 100% of the data 100% of the time Data Vault is the matching technique.

Dan Linstedt defines the Data Vault as *‘a detailed historically oriented, uniquely linked set of normalised tables that support one or more areas of the business’*. In other words; it is a semi-normalised model where key distribution, relationships and descriptive data are separated. The historical data is denormalised to some extent, labeling it as a hybrid modeling approach for data modeling. However, it is not even the difference in data modeling that make Data Vault unique (similarities with the IBM industry models come to mind). It is the way Data Vault uses the archetypical entities to allows for a natural, almost organic, controlled expansion of the model (in a way the IBM industry models do not) that make it such a powerful technique.

The Data Vault defines three main types of entities: the Hub, Satellite and Link tables.

- **Hub**: Contains the unique list of business keys and maintains the EDW key distributions.
- **Satellite**: Historical descriptive data (about the hub or link). Much like a Type 2 dimension, the information is subject to change over time.
- **Link**: Unique list of relationships between Hub keys. A Link can be relational or transactional (with a limited number of variations). As relationships Link tables are always many-to-many. You have to be prepared for future changes.

An example of a simple Data Vault model is as follows:

![Data Vault Diagram](image)

This modeling approach has proven to be very flexible. For instance, if different information about a product (defined as a Hub entity) is added a separate Satellite table which contains new history can be created without impacting existing tables. Similarly, if an existing Satellite contains a single attribute with a high change rate the Satellite table can be split in separate smaller Satellites (and merged if the situation is the other way around). This is very easy to implement since the keys are maintained in the Hub table and not in the Satellite.

Another example of the flexibility is in the Link table. You can define the Link as a single point of integration (unique list of hub keys) and have separate Link-Satellites for maintaining transactions or relationship information. Or, you can add a new link table for a different type of relationship.
A third example focuses on moving towards an interpretation of the data. Take the example where you want to create an integrated view on the employee data (Employee Hub) following specific business rules. This results in the definition of a subset of employees with certain qualities: a 'special employees’ Hub. The relationship of which employee is a ‘special employee’ is maintained in a new Link table.

**Employee Hub:**

<table>
<thead>
<tr>
<th>ID</th>
<th>Logical Key</th>
<th>Source Row ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>201001</td>
<td>35</td>
</tr>
<tr>
<td>2</td>
<td>201004</td>
<td>36</td>
</tr>
<tr>
<td>3</td>
<td>201245</td>
<td>37</td>
</tr>
</tbody>
</table>

**Employee Satellite:** (with raw address data):

<table>
<thead>
<tr>
<th>ID</th>
<th>First Name</th>
<th>Last Name</th>
<th>Address</th>
<th>Effective Date</th>
<th>Expiry Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>John</td>
<td>Doe</td>
<td>40 George St.</td>
<td>20080101</td>
<td>99991231</td>
</tr>
<tr>
<td>2</td>
<td>John</td>
<td>Doe</td>
<td>40 George Street</td>
<td>20080101</td>
<td>99991231</td>
</tr>
<tr>
<td>3</td>
<td>Peter</td>
<td>Smith</td>
<td>70 York St.</td>
<td>20080504</td>
<td>99991231</td>
</tr>
</tbody>
</table>

After applying a ‘match’ business rule the following tables will be populated. In this case two records are merged into one using a specific match and survive algorithm. The result is a set of two employees instead of the original three.

**Special Employee Hub** (selected by business rules):

<table>
<thead>
<tr>
<th>ID</th>
<th>Logical Key</th>
<th>Source Row ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>201004</td>
<td>36</td>
</tr>
<tr>
<td>3</td>
<td>201245</td>
<td>37</td>
</tr>
</tbody>
</table>

The relationship between the original data and the cleaned data has to be maintained for the audit trail. This table shows the relationship between the original and the cleansed data:

**Employees and Special Employees Link**

<table>
<thead>
<tr>
<th>ID original employee</th>
<th>ID special employee</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

It is clear to see how these concepts deliver on the EDW promises made at the start of the document.
Positioning and misconceptions

Data Vault is often misunderstood for a complete EDW methodology and as such discarded for the wrong reasons. It does not replace existing techniques to prepare data for reporting (i.e. a star schema) or solve issues regarding data staging and interfacing. Even Dan Linstedt himself does not claim that the Data Vault is the single solution for everything; a proper EDW typically requires some form of Presentation Layer. In other words: Data Vault is a modeling technique and not a methodology.

The insights which Data Vault has delivered do render elements of existing methodologies obsolete. This is one of the reasons that Bill Inmon (the ‘father of the Data Warehouse’) has stated that the Data Vault is the ideal technique for modeling the EDW in the DW2.0 Framework. Comparison must also be made with another well-known expert; Ralph Kimball and the dimensional modeling as defined in the Lifecycle Toolkit. This extensive methodology defines concepts that are still very valuable today and these can be used in conjunction with Data Vault in a very effective way. Implementing Data Vault does however replace certain concepts to be implemented prior to creating a Star or Snowflake schema but with the advantage of making this a very easy and generic step in ETL.

When position Data Vault in the Kimball architecture the best way to think about it is to define Data Vault as the Data Staging, although it does more than that. The following diagram highlights this:

In conclusion, all techniques and methodologies still have their place and it’s the way we organise, structure and apply them which defines the truly generic EDW architecture we aim for.
Patterns

Thanks to the modular approach the Data Vault is full of deterministic patterns which can be used to shorten the time to market for the EDW. These patterns (when applied within a structured ETL Framework) cover the entire scope for ETL leaving only the specific business rules to be defined manually, and even this can be automated to an extent. In this fact lies one of the truly powerful applications of the concept: **consistent structured and efficient ETL.** The only requirement is that you have the target Data Vault model ready and work within a structured ETL Framework. This is true model-driven development for the Data Warehouse. Examples of identified (ETL) patterns using the Data Vault are:

<table>
<thead>
<tr>
<th>Concept</th>
<th>Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hub</td>
<td>Select a distinct set of only the attribute designated as business key (usually – but not always – the primary key of the source table). Do a lookup against the target Hub table, if the key does not exist for that source system, insert and generate a new Data Warehouse key.</td>
</tr>
<tr>
<td>Satellite</td>
<td>Select everything from the source table and use the designated business key for a key lookup against the Hub table. The key value is then used to lookup the history values in the Satellite. All attributes but the business keys are inserted if there is change in values.</td>
</tr>
<tr>
<td>Dimension (from Data Vault to Star Schema)</td>
<td>Join the Hub and Satellites and calculate the overlap in timelines, join this to the Link and through to the other set of Hub and Satellites. Since the keys are already distributed in the Hub table the (structure of the dimension) can be changed at any time without losing history or complex ETL.</td>
</tr>
</tbody>
</table>

These ETL patterns will be referred to in the following articles where the required steps in the ETL Framework to deliver the EDW are explained in more detail. There are many fully functioning examples, demos and implementations available to show how this can be achieved.

**The role of ETL**

While not specified as part of Data Vault, ETL in the new EDW context should be split in the smallest units of work possible. In other words; the more and smaller processes to do tasks the better. Smaller units of work also means more generic ETL and therefore more options to generate the logic instead of manually developing it. This adds to the already effective EDW deployment.

Single ETL processes should be self-sufficient and should be designed to be run at any time, for any interval without corrupting data or generally causing problems. It is also possible (and recommended) to step away from the traditional 'layer-based' loading where all ETL for a specific type is loaded first before the next batch can start. By organising workflows or batches in a more 'vertical' way true flexibility in terms of prioritising information and balancing load schedules can be achieved. This will be explained in detail in the next papers.

Scheduling wise, Data Vault as modeling technique offers tremendous flexibility in (parallel) scheduling due to the early key distribution as well as rules regarding the use of placeholder values.
(Yet) Another Framework?

No proper EDW implementation can be done without a solid ETL Framework which addresses operational / process metadata, error handling and definitions of what happens where in the ETL architecture. The implementation of an ETL Framework ties in closely with the EDW architecture. An example is regarding error handling; in the Data Vault EDW, with its focus on storing data, no error handling should be implemented on the way in to the Data Warehouse because this is essentially a business rule. How to use the error handling concepts and how this is implemented is one of the functions of the ETL Framework. Other functions include (but are not limited to):

- Specifying the ETL templates
- Defining how timelines are managed
- Integrating operational metadata
- Specify file handling

The ETL Framework itself is an implementation approach for the EDW that links in with the fundamental concepts advocated by Data Vault.

A window on the future

It seems hardly practical that ETL work as we know it today will still exist in a couple of years. More and more sensible hybrid approaches towards Data Modeling and increased insight in the necessary steps towards data and information management, as well as improvements in the options (APIs) of ETL software will lead to ‘hiding’ the complexities and shifting the focus of the work on data modeling. ETL development will move towards either incorporating the identified patterns into the way they move the data (using a semantic layer to translate the pattern into ETL logic), or ‘brute-force’ the identified patterns into ETL processes following templates (ETL generation). Either way the days of the pure ETL developer seem to come to an end.

The exciting thing in both scenarios is that there is no limitation on the type of applications you can link to the EDW and there is no locking in to specific technologies to achieve this. Right now it’s already possible to generate the ETL processes once you understand your data and have defined your EDW model in Data Vault.

Conclusion

By embracing Data Vault, ensuring it has the proper position in the EDW reference architecture and leveraging the deterministic ETL patterns using a solid ETL Framework the EDW solution has become more ‘construction’ rather than ‘architecture’. In other words; once you have designed your model the implementation is generic. Because this approach works in all situations it will contribute to the inevitable shift in skills and focus by hiding more and more of the architecture into a commonplace solution. By taking away the lengthy development times and associated distrust by the business this newly styled EDW will now both serve the business’ use and provide the future-proof, flexible and easy-to-maintain architecture to satisfy the architects and DBAs.

This is the first in a series of articles that aim to give the Enterprise Data Warehouse the solution it deserves.