USING THE DATA VAULT ARCHITECTURE TO ACCELERATE DELIVERY WHILE INCREMENTALLY BUILDING YOUR DW

Bruce McCartney, Database Information Systems Inc.

ABSTRACT

The purpose of this paper is to review the motivation for, structure of, and experiences with implementing an EDW using the Data Vault architecture developed by Dan Linstedt so that readers have an understanding of the benefits and risks/challenges involved in the architecture. This paper will review the rationale for selecting this architecture over federated data marts and operation data store/staging areas. The paper will present an overview of the architecture and data model and how this model best implements the main attributes of a data warehouse (subject-oriented, time-variant, non-volatile and integrated). In addition, the paper will discuss experiences with implementing the architecture and lessons learned, including a discussion of a methodology and tools for drastically reducing the time to create and populate the Data Vault.

OVERVIEW OF A DATA WAREHOUSE

One of the challenges of data warehousing is that there is no formal specification or definition of a data warehouse. The history of data warehousing in one form or another goes back to the early 1980s; my first data warehouse was a DB2 Oil and Gas Well Information system. We didn't call it a data warehouse, but it had all of the characteristics of what is now commonly called a data warehouse.

DEFINITION

Around 1991, Bill Inmon (commonly referred to as the *father of the data warehouse* www.inmoncif.com) attempted to define as follows:

"A data warehouse is a subject oriented, integrated, non volatile, time variant collection of data designed to support management decision support system needs"

A check of Bill's current website adds the following: "a collection of integrated subject oriented data bases designed to support the DSS function, where each unit of data is relevant to some moment in time. The data warehouse contains atomic data and lightly summarized data." Since that time, Bill has evolved his work to include a formal specification called DWH 2.0 and recently had the following quote: *"The Data Vault is the optimal choice for modeling the EDW in the DW 2.0 framework."* This paper will attempt to define why…

DATA WAREHOUSE ARCHITECTURES

As discussed above, with no formal definition of a data warehouse there is any number of possible architectures. Over time, two main approaches to architecture have evolved: *centralized* and *federated*. A centralized architecture involves a central data warehouse and a series of data marts for end user DSS. The central database is most often a relational DBMS and loaded periodically via Extraction, Transformation and Load (ETL) jobs. The database usually includes a true 'staging' area and is optimized for load and integration performance. A true staging area is one that is emptied after each iteration of a load. The data marts are often optimized for end user query and analysis and may contain summarized data. This architecture is also often referred to as *spoke and hub* due to its diagram's resemblance to this object (see figure 1)

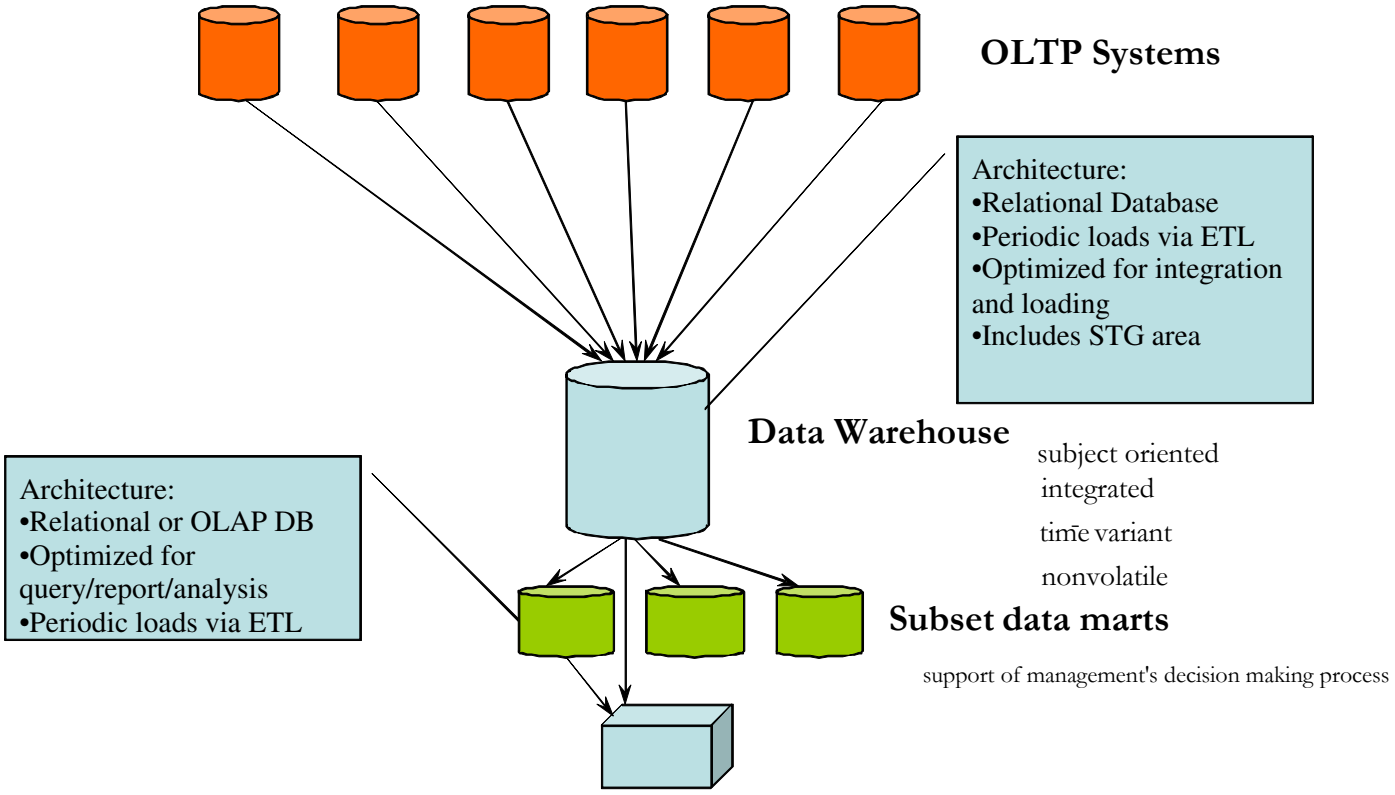


Figure 1 – Central Data Warehouse Architecture

A federated architecture involves no central database for integration and loading. The focus on this architecture is on the data marts required to support end user queries. There is usually a staging area that manages the loading and often requires some history to control data mart loading and integration. There is also a tendency to create a short term data store called an operational data store for timelier reporting on current data. The data warehouse is usually viewed as the collection of data marts used for reporting. To integrate these marts, architects often used a concept of shared or *conformed dimensions* designed to facilitate cross-mart reporting and/or consistent 'master data' reports. This architecture is described in figure 2.

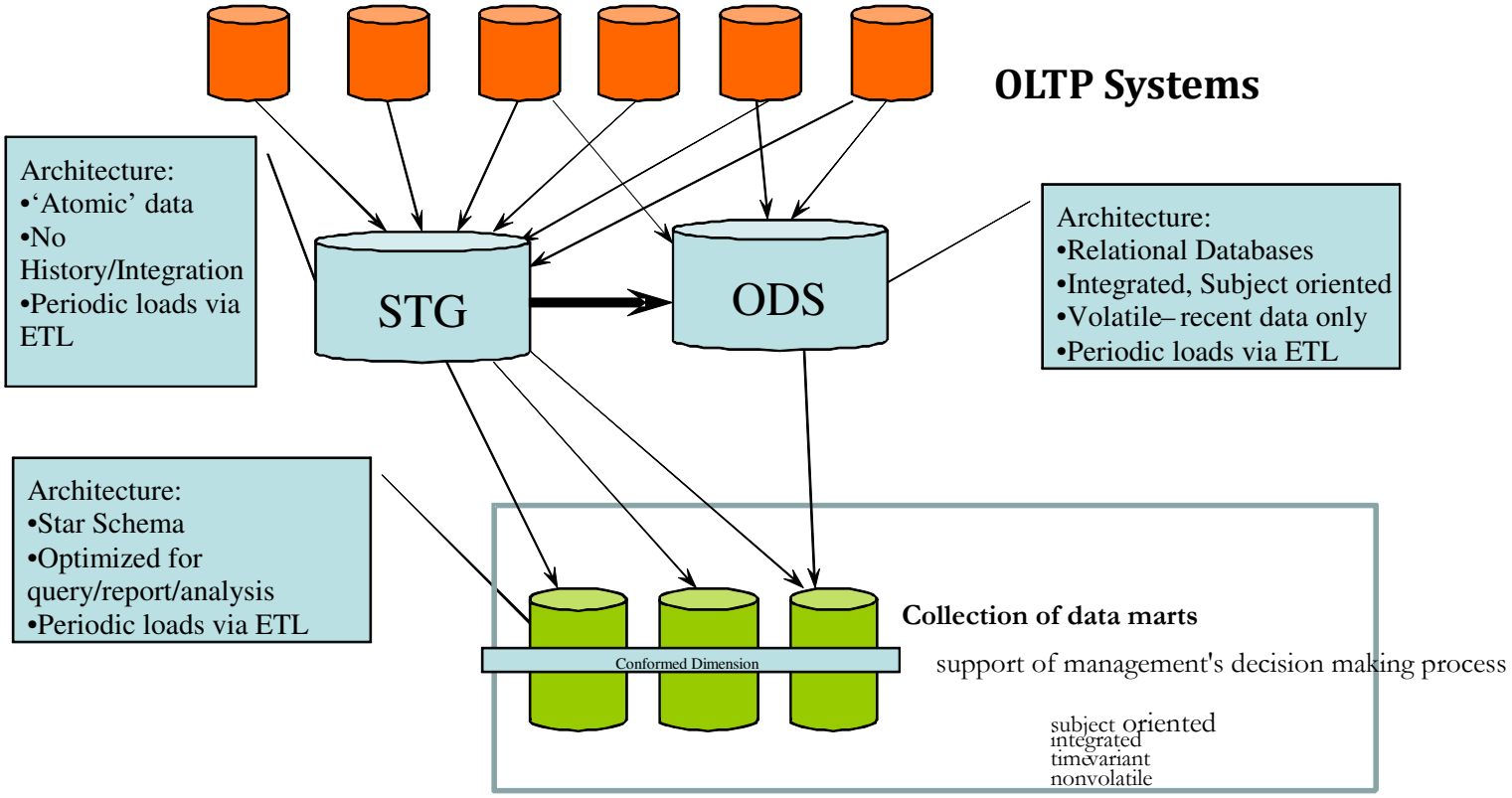


Figure 2 - Federate Data Warehouse Architecture

HOLY WARS- KIMBLEISM VS. INMONISM

Choosing between these two architectures are often the first choice a data warehouse architect must make. There are very good arguments and resources to support *both* approaches – and it can often become a war between central vs. federated. The Kimball (www.kimballgroup.com) . Dr. Kimball pioneered the *star schema* or *dimensional* model for data marts and is one of its biggest supporters. He believes that this model can support all aspects of the data warehouse by integrating the data using conformed and/or shared dimensions across marts. There has been lots of chat on the web about both approaches, and it ultimately is the choice of the architect. I have been involved with both approaches and am certain of one thing: *neither approach is easy*. After struggling with the federated approach, I ran across a posting regarding the Data Vault and have since been convinced this approach provides the best combination of scalability and flexibility and will discuss these areas in detail in the following pages. First, I think it is important to review why enterprises are moving towards a data warehouse at all.

WHY USE A DATA WAREHOUSE?

Before describing the Data Vault, it is worthwhile to discuss the objectives of a data warehouse, as well as mentioning some of the goals of a long-term enterprise data warehouse (EDW) initiative. These objectives and goals should be agreed to by your organization and can be used to help support architecture decisions.

Objectives and Goals of a Data Warehouse

There are no 'standard' objectives of an EDW or Business Intelligence (BI). The following is based upon my experience and reading on the subject – but by no means is meant to be ubiquitous.

Objectives of building an EDW:

- Integration of Data

  In my experience, Business Intelligence (BI) style reporting and analysis requires the *integration* of data kept in many operational systems. By integrating this data into information, end users can ask questions and perform analysis that was previously not possible or at least very difficult.

- Non volatile

  One of the main issues I have seen in BI projects is that of the data changing 'underneath' the user. This becomes a problem when an analysis one day reports a different result than the next day and is gone next month. Volatile data becomes suspect data.

- Time Variant

  As we will see later, time is of the essence in BI. As things change over time, many challenges exist for data warehouse designers, requiring careful consideration and diligence, particularly in light of new regulations on data governance.

- 100% of the data, 100% of the time

  One thing I have seen many times is that as requirements change, it turns out the data warehouse doesn't keep the data or has transformed the data via some business rules on the way into the EDW. This is a very expense thing to correct, and has lead me to this objective.

- Data Mining

  Often, BI projects are forced into somewhat of a 'build it and they will come' genesis. One of the biggest prizes of a successful EDW is users being enabled to see relationships in the data that they did not know existed.

Other objectives undoubtedly exist for your organization's EDW. It is a good idea to discuss them before picking an architecture. Regardless of architecture, there are a number of common goals in developing an EDW.

Goals while building the EDW:

- Scalable

  The amount of data in a successful EDW will grow, and the entire architecture must scale up as demand for data increases and tolerance for latency decreases.

- Repeatable, incremental build

  This goal arises from experiences building every new data mart using a new approach and throwing much of the old away. We as Information technology (IT) professionals need focus on an approach that is repeatable, and can incrementally grow upon previous work

- Fault tolerant

  A major issue for EDW designers is bad data. We must design processes that are not reliant on good data, but support the identification of bas source data and the seamless repair of this data *in the source system.*

- Auditable

  SOX compliance and storing *the facts as they were* allows IT to move away from the data stewardship business and towards the data service business. Any change to source system data by IT on the way into the data warehouse can result in 'ownership' of the data being shifted away from the business and that is not good for IT.

- Consistent

  A goal of consistency means that everyone involved in BI projects understands how and why things are as they are.

NOT OBJECTIVES

One thing also to consider in your organization's EDW is what are *not* some of the objectives of the EDW. These are based upon my experience and it is also a good idea to discuss them before picking an architecture.

Not Objectives of the EDW

- Integration of *applications* or *workflow*

  This is a slippery slope to address. EDW is not EAI (Enterprise Application Integration). There is a business need for more real-time data warehousing. Your architecture needs to support a so called BUS architecture as one of the consumers and even one of the providers (eventually). Use the BUS architecture to integrate your applications, use the EDW to integrate your data, treating the EDW as an application on the BUS. More on this later.

- End user reporting or analysis

  Inmon's definition says the EDW supports DSS. It is my experience that DSS against an EDW directly presents many, many challenges in the application of business rules and optimization, especially in the central architecture. This is where the dimensional model used to support a business requirement shines.

- Cleaning of data

  I believe the EDW should contain first *the data as it was at the time*. If data cleansing is required, the best place for this is in the source systems. If this is not possible or feasible, cleansed data should *augment* the source data in the EDW to support the notion of auditability and data ownership.

- Application of 'soft' Business Rules to filter, cleanse, transform, allocate or calculate data

  This builds on the previous point that applying Business Rules on the way into the EDW will make it harder to scale, change and be auditable. As we will see later, business rule change is a major challenge to EDW projects and the closer to the end use of the data we can do this stuff, the better off IT will be as we will be able more rapidly change how these soft rules are applied over time.

To build on the last point, I think we are trying to build a system of record for the enterprise over time. I often hear people refer to the desire for the EDW to provide a `single source for the truth`. I like to counter with what we are building is a *`single source for the facts as any time`*. This is because the truth is subjective; it is based on business rules that are valid at a point in time but subject to change. As we will see, a compliant EDW is one where there soft business rules are applied after the data is loaded (see figure 3).

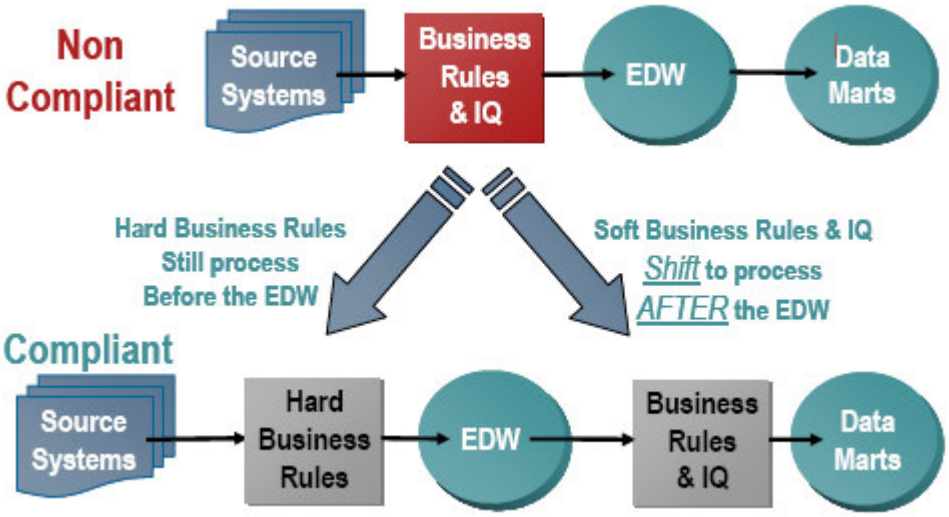


Figure 3 - Business Rules in an EDW

SUMMARY OF DATA WAREHOUSE

One thing I want to stress is that Data Warehouse is not easy. If you are looking at designing an EDW, I believe the main factors on how easy or hard this will be are latency, volume, data sources and scope. Figure 4 attempts to place these on a chart with speed and difficulty as the main axis.

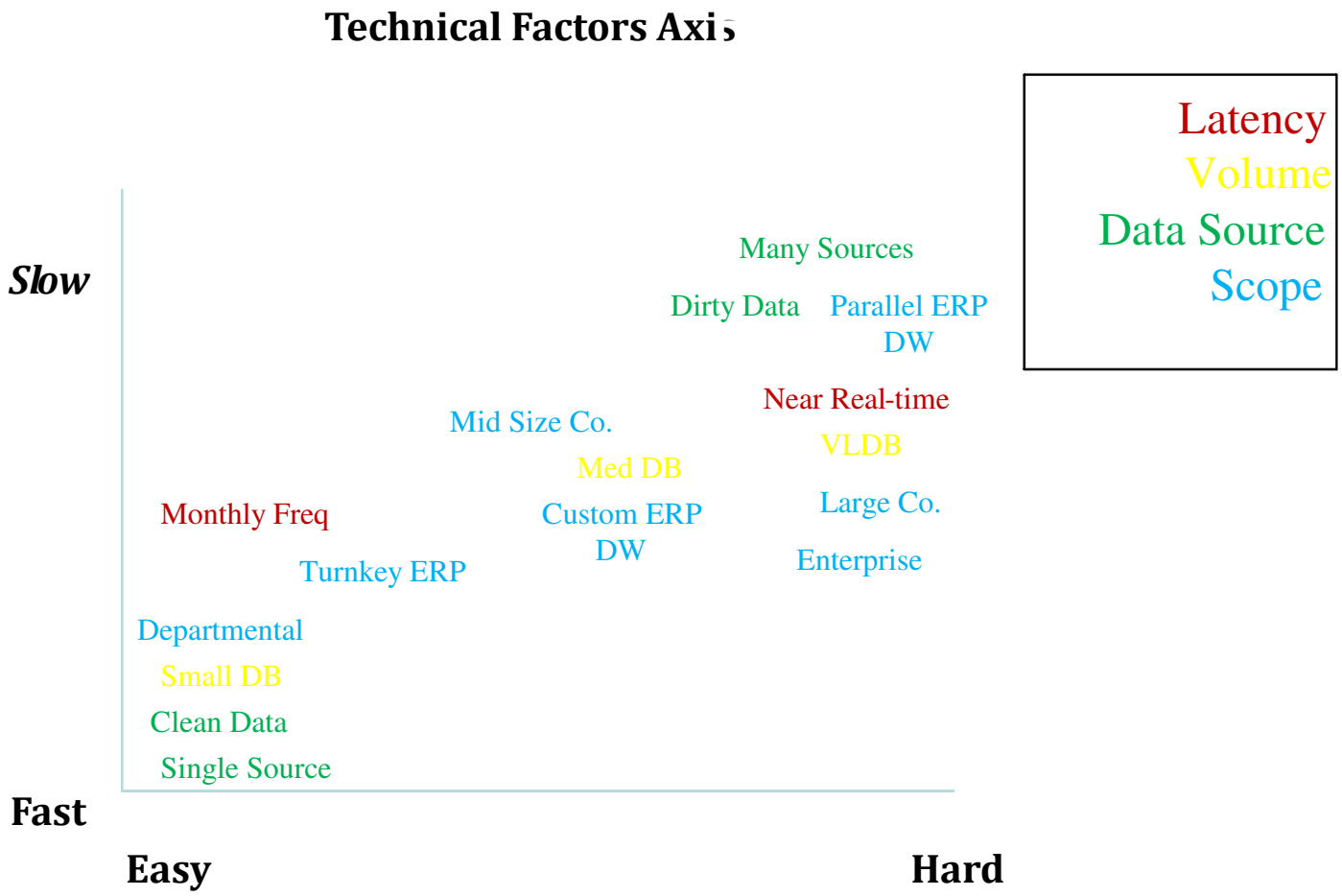


Figure 4 - Technical Factors Axis

There are many examples of large project failures in this area over the last 15 years, and according to Gartner, the number one reason for DWH or BI failure is `lack of business sponsorship, along with top-down support`. It is clear that business sponsorship and ownership of the data is a critical success factor in any EDW project. This is primarily a *management* challenge, which is not the focus of this paper or my career. From a *technical* perspective, challenges to success are as follows:

- Integration - data quality, multiple sources
- Scalability - quantity of data increases, latency demand
- Delivery - Enterprise data model vs. departmental needs
- Flexibility - response to business changes, ERP implementations
- Compliant - auditable data, repeatable process, SOX etc.

As a technical professional, I believe the Data Vault architecture is best positioned at address these challenges and the remainder of this presentation will attempt to discuss why.

OVERVIEW OF THE DATA VAULT

I would now like to turn our attention to the Data Vault. I first saw this architecture three years ago when researching a generic data model approach to an integration and business intelligence project. The architecture was developed by Dan Linstedt over the past 15+ years and is architected to address many of the challenges I have been discussing. This architecture is in the public domain and is available to all at www.danlinstedt.com. I have had the opportunity to work with Dan on a couple of customer projects and would like to publicly acknowledge and thank him for his contributions to the field of data warehousing.

DEFINITION AND KEY CONCEPTS

The Data Vault is defined as *`a detailed, historically oriented, uniquely linked set of normalized tables that support one or more functional areas of business`*. As we shall see, this architecture has been developed to address the technical challenges outline in the previous section. It contains the following key concepts:

- Everything is MANY-TO-MANY
- Time dependency on everything
- Late `BINDING` for data – the LINK
  - Simplifies load dependencies
  - Data centric view of integration
- Uses Relational DBMS
  - Tables, Columns, and Views – nothing exotic

DATA VAULT ARCHITECTURE

The best way to describe the architecture is using Figure 5.

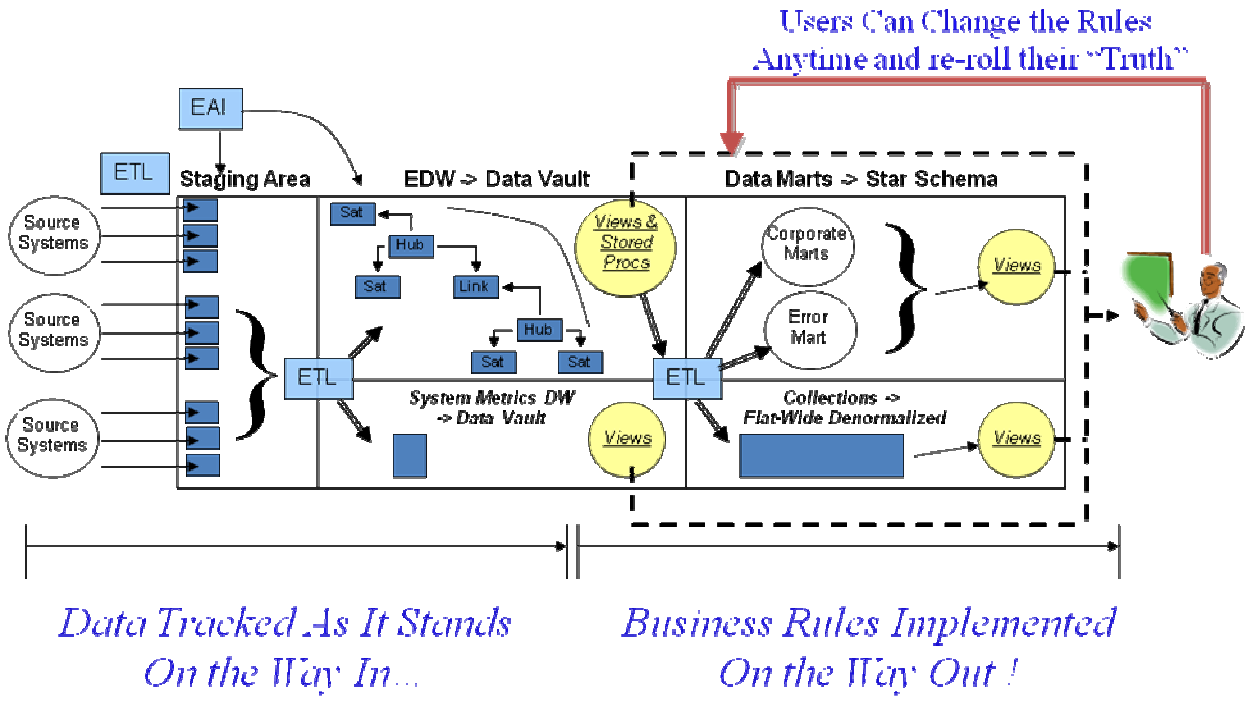


Figure 5 - Data Vault Architecture

As you can see, there is a flow of data from source system to end user through a centrally defined EDW – the Data Vault. The Data Vault is fed one of two ways:

1. Periodically via ETL loads from a staging area that is emptied each period. This is the most common approach, or
2. Event-based via and EAI bus for more real-time loading.

The model for the EDW will be discussed later. One the data is loaded, ETL is used to populate data marts based upon business requirements and rules. These data marts can be dimensional models or a flattened, denormalized model called a *collection*. Regardless of the model for the mart, the population process remains the same. Users are thus able to re-roll their truth from this auditable data source. Let's look at the components of the data model in the Data Vault.

Data Vault Components

The model for the Data Vault contains three main table types.

- HUB = list of Business Keys
- Satellite = Descriptive information
- Link = Describes relationships between keys

The HUB

Definition: a single table carrying at a minimum a unique list of business keys.

Other attributes in the Hub include:

Surrogate Key – Optional component, possibly a smart key or a sequential number.

Load Date Time Stamp – recording when the key itself first arrived in the warehouse.

Last Seen Time Stamp – recording when the key itself last arrived in the warehouse, used for data driven 'soft' delete.

Record Source – A recording of the source system utilized for data traceability.

Example:

```
CREATE TABLE HUB_AFE
(
  AFE_NO          VARCHAR2(12 BYTE)            NOT NULL,
  AFE_ID          NUMBER(22)                   NOT NULL,
  LOAD_DTTM       DATE                         DEFAULT sysdate              NOT NULL,
  LAST_SEEN_DTTM  DATE                         DEFAULT sysdate              NOT NULL,
  DATA_SOURCE     VARCHAR2(240 BYTE)           DEFAULT ACCTG'          NOT NULL
)
```

THE SATELLITE

Definition: Provide context (descriptive) information much like a Type-2 dimension, its information is subject to change over time;

The Satellite is comprised of the following attributes:

Satellite Primary Key: Hub or Link Primary Key – migrated into the Satellite from the Hub or Link.

Satellite Primary Key: Load Date Time Stamp – recording when the context information is available in the warehouse (the new row is always inserted).

Satellite Optional Primary Key: Sequence Surrogate Number – utilized for Satellites that have multiple values (such as a billing and home address), or line item numbers, used to keep the Satellites sub-grouped and in order.

Record Source – A recording of the source system utilized for data traceability.

Example:

```
CREATE TABLE SAT_AFE_STATUS
(
  AFE_ID                    NUMBER(22)          NOT NULL,
  LOAD_DTTM                 DATE                DEFAULT sysdate              NOT NULL,
  AFE_STATUS                VARCHAR2(4 BYTE)    NOT NULL,
  DATA_SOURCE               VARCHAR2(20 BYTE)   DEFAULT 'ACCTNG'      NOT NULL,
  LOAD_END_DTTM             DATE
)
```

Notes:

Many satellites possible per HUB/LINK; group be frequency of change to avoid data explosion due to rapid change

Date effective (current row is optionally identified by a NULL end date)

THE LINK

Definition: Link Entities or Links, are a physical representation of a many-to-many 3NF relationship.

The Link contains the following attributes:

Surrogate Key – Optional component, possibly a smart key or a sequential number.

Hub 1 Key to Hub N Key – Hub Keys migrated into the Link to represent the composite key or relationship between two Hubs.

Load Date Time Stamp – recording when the relationship/transaction was first created in the warehouse.

Last Seen Time Stamp – recording when the relationship/transaction was last seen in the warehouse, used for terminating relationships.

Record Source – A recording of the source system utilized for data traceability.

```
Example:

CREATE TABLE LNK_AFE_ACCOUNT
(
  L_AFE_ACC_ID     NUMBER(22)                    NOT NULL,
  ACC_ID           NUMBER(22)                    NOT NULL,
  AFE_ID           NUMBER(22)                    NOT NULL,
  LOAD_DTTM        DATE                          DEFAULT sysdate      NOT NULL,
  LAST_SEEN_DTTM   DATE                          DEFAULT sysdate      NOT NULL,
  DATA_SOURCE      VARCHAR2(20 BYTE)             DEFAULT 'DATA VAULT' NOT NULL
```

LOADING THE DATA VAULT

The normal load process for the Data Vault is very simple, repeatable and consistent. The process has been designed to scale, support near real-time as well as be fault tolerant. Figure 6 shows an overview of the load process. One main goal is to eliminate data dependencies of the loading. Additional tables loads simple fold into the process.

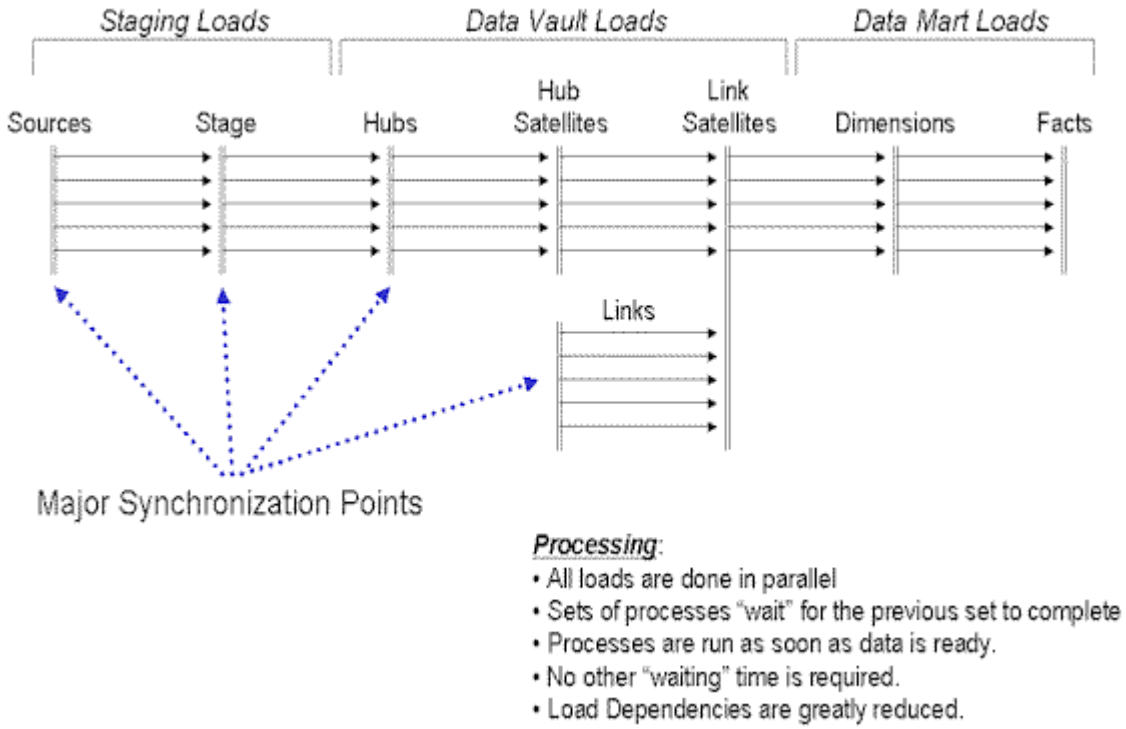


Figure 6 - Data Vault Load Process

WHY USE THE DATA VAULT

There are a number of alternative EDW architectures available to designers. You can eliminate the concept of data warehouse altogether by building interfaces or EAI connections to data marts. This is a process-centric approach and may prove effective for enterprises with well defined business process or existing EAI infrastructures. Another approach is the federated dimensional model. This model can be used to get a quick win data mart; but may limit long-term growth as the number of data marts to integrate increases. I think the Data Vault architecture provides the best approach to address the technical challenges discussed in the previous sections of this paper. The main reason for this is the *data model*. There are a number of modeling approaches to choose from, I will now discuss the most common: the 3rd normal form and the Star Schema; and describe how each presents challenges to EDW.

The 3rd normal form model or Enterprise Data Model is the Holy Grail for data architects – the mother of all models. It can take years to develop in large customers because it requires agreement amongst business units regarding the entities and relationships. This approach also has some issues:

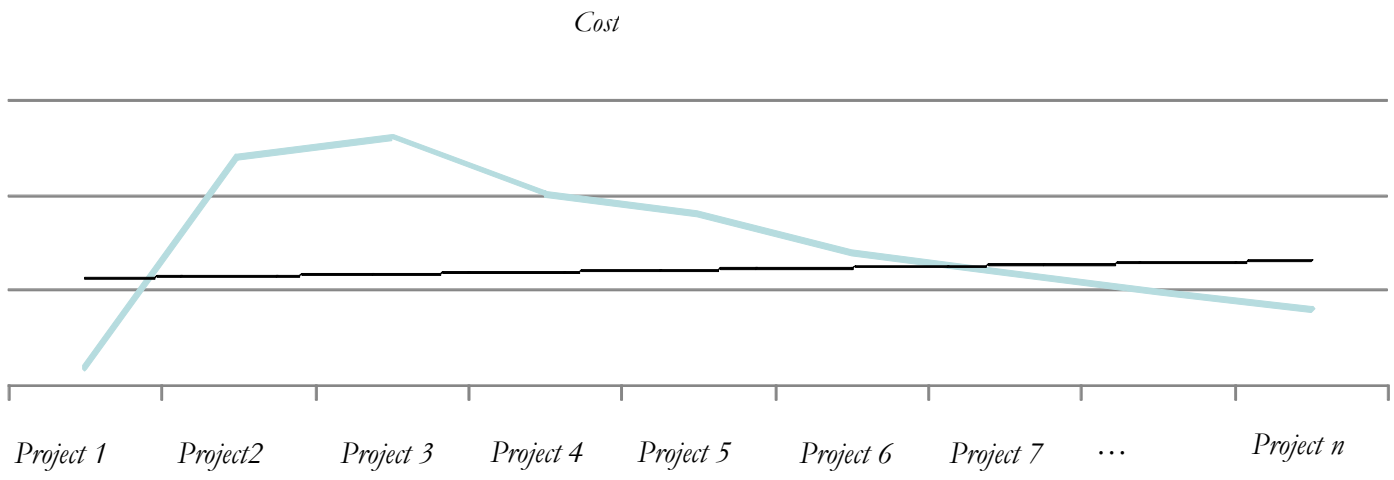
- Adaption to change is very hard
  - Growth of new relationships
  - Duplicate data sources – priority / trust layer
- Integration
  - Load dependencies for integrity
  - Data Quality != Referential Integrity
- TIME (new parent, key change)
  - Impacts ripple thru children
  - One-to-many (really!)

The Star Schema (or dimensional) model creates dimensions and facts. Facts are linked to one or more dimensions.

This approach also has some issues:

- Updates/Deletes expensive
- Dimensions over time (Type 2 and 3)
- Architecture includes 'helper', 'bridge', and 'junk' tables!
- Grain (level of detail) issues difficult to resolve
- Real-time loading impractical
- Transactions appearing before dimension data (load dependency issue)
- Loading History, changing history

The Data Vault mode is architected to be simple, layered, insert only and incrementally built over time. The cost of the Data Vaults per project therefore reduces over time as seen in Figure 7:



SUMMARY

In summary, the main benefit I have seen using the Data Vault is *agility*. Using a repeatable, consistent model and process allows you to speed deliver via templating. This has enabled us to get our data integrated and available for marts in much less time. In fact, Dan Linstedt is working on a tool called RapidACE™ which further automates the entire model and etl process and looks very promising. Our agility is also improved by the incremental build we are performing. Our second mart reused about 70% of the data of our first – saving much ETL and modeling time as we were able to reuse much of the infrastructure and Data Vault. Finally, agility is developed by adding business rules *after* the source data is loaded and integrated. This has allowed us to separate these rules and manage change much more readily.

EXPERIENCES

I want to conclude by saying that Data Warehouse is still not easy. Our experience with the Data Vault has shifted many of the technical challenges; while rapidly highlighting the management challenges of:

- Data Ownership/Stewardship
  - Business ownership of data / rules easier
  - Still need business sponsorship – Data Vault enables rapid "as is" data warehouse

From the technical side, we have seen a shift of challenges as well:

- Challenges (before Data Vault)
  - Integration
  - Scalability
  - Delivery

- Flexibility
- Compliant

- Challenges (after data Vault)
  - Data Quality
  - Define the Truth Corporately
  - Business Rules/Processes Change
  - EAI

I think these challenges are much more interesting to resolve than what we had previously.

RESOURCES

- Dan Linstedt
  - Thanks for many of the diagrams in this presentation
  - www.danlistedt.com (specification, forums)
  - www.rapidace.com (tool)
  - www.geneseeacademy.com (training DV; DWH 2.0)
- Bill Inmon
  - www.inmoncif.com (DWH 2.0)