Introduction to Data Vault Modeling

Compiled and Edited by Kent Graziano, Senior BI/DW Consultant

1.0 Abstract

The purpose of this article is to present an introduction to the technical components of the Data Vault Data Model. The examples give you the basics for how to build, and design structures when using the Data Vault modeling technique (there are many more details in the book..wink, wink). The target audience is anyone wishing to explore implementing a Data Vault for integration purposes whether it be an Enterprise Data Warehouse, Operational Data Warehouse, or Dynamic Data Integration Store.

2.0 Introduction

Data Vault Modeling is a specific data modeling technique for designing highly flexible, scalable, and adaptable data structures for enterprise data warehouse repositories. The formal definition is as follows:

The Data Vault is a detail oriented, historical tracking and uniquely linked set of normalized tables that support one or more functional areas of business. It is a hybrid approach encompassing the best of breed between 3rd normal form (3NF) and star schema. The design is flexible, scalable, consistent, and adaptable to the needs of the enterprise. It is a data model that is architected specifically to meet the needs of today's enterprise data warehouses.

The Data Vault model follows all definitions of the Data Warehouse (as defined by Bill Inmon) except one: the Data Vault is functionally based, not subject oriented – meaning that the business keys are horizontal in nature and provide visibility *across* lines of business.

2.1 Data Model Evolution

Data Vault is a hybrid, best of breed solution. The Data Vault is architected and designed to meet the needs of enterprise data warehousing. It is ***NOT*** an adaptation. The research to develop the Data Vault approach began in the early 1990s, and completed around 1999 (see figure 2-1). Throughout 1999, 2000, and 2001, the Data Vault design was tested, refined, and deployed into specific early adopter sites. In 2002, the industry thought leaders were asked to review the architecture. In 2003, the first public classes teaching the modeling techniques to the mass public were offered (in Denver, Colorado).

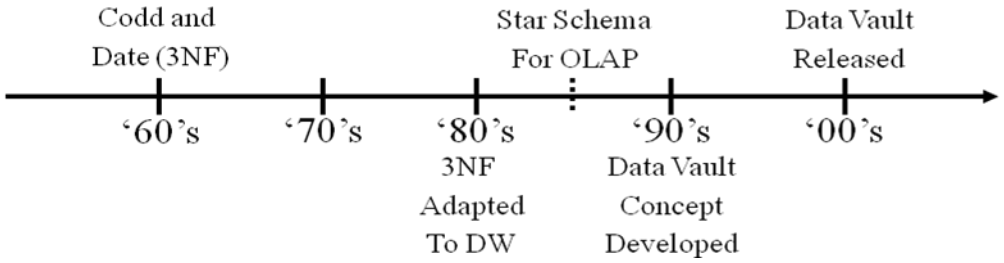


Figure 2-1 - Data Modeling Evolution

When we look at the evolution of the data modeling architectures, we notice that there had not been an architecture specifically designed to meet the needs of Enterprise Data Warehousing. The Data Vault is just such an architecture. It's an evolutionary approach because it combines best of breed (from 3rd Normal Form, and Star Schema), and re-architects from the ground up – specifically to meet the needs of the Enterprise Data Warehouse.

An analogy we have used is the Ferrari and the Big Rig. The Big Rig is the 3rd Normal Form (3NF), architected and designed to meet the needs of a specific set of requirements (i.e., OLTP systems). It was *adapted* to meet the needs of data warehousing. The Ferrari (Star Schema) was architected and designed to meet the goal of query speed to make data accessible to business users for analysis. It was *adapted* to meet the needs of enterprise-level data warehousing. Neither the Ferrari, nor the Big Rig was designed for the off-road/on-road requirements that the SUV space fills. The SUV could be considered a best-of-breed hybrid solution of truck and car together. The Data Vault is the SUV of data modeling as it was specifically architected to meet the needs of Enterprise Data Warehousing.

2.2 Architectural Definitions

So where does a "Data Vault" fit in the big picture? The architecture is really compliant with the traditional Corporate Information Factory (CIF) approach as put forth by Bill Inmon (father of data warehousing). In that framework, the Data Vault fulfills the role of a centralized enterprise data warehouse (EDW) which in turn provides data to star schema data marts as well as flat (denormalized) report tables and/or exploration marts.

The Data Vault methodology includes each of these components. The architectural component discussed in this article the central EDW/Data Vault.

2.3 EDW – Data Vault

The EDW (enterprise data warehouse), or core historical data repository, consists of the Data Vault modeled tables. The EDW holds data over time at a granular level (raw data sets). The Data Vault is comprised of Hubs, Links, and Satellites. The Enterprise Data Warehousing Layer is comprised of a Data Vault Model where all raw granular history is stored. ***Unlike many existing data warehouses today, referential integrity is complete across the model and is enforced at all times.*** The Data Vault model is a highly normalized architecture. Some Satellites in the Data Vault may be denormalized to a degree under specific circumstances.

The Data Vault modeling architecture has been likened to 3½ normal form. The business keys in the Hub appear to be 6th normal form, while the load date and record source are 3rd normal form. The Data Vault model should represent the lowest possible grain of source data. The Hubs and Links in the Data Vault model provide the back-bone structure to which context (the Satellites) are applied.

3.0 Hubs, Links, and Satellites

The Data Vault model consists of three basic entity types: Hubs, Links, and Satellites (see Figure 3-1). The Hubs are comprised of unique lists of business keys. The Links are comprised of unique lists of associations (commonly referred to as transactions, or intersections of two or more business keys). The Satellites are comprised of descriptive data about the business key OR about the association. The flexibility of the Data Vault model is based in the normalization (or separation of) data fields in to the corresponding tables.

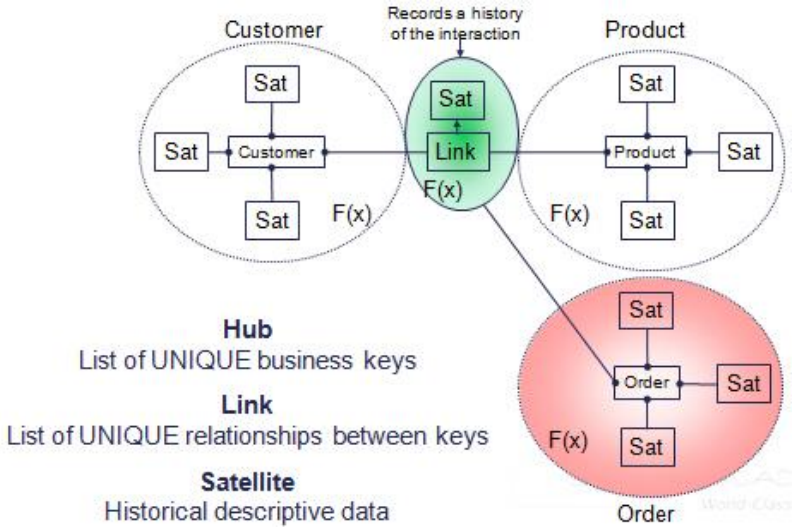


Figure 3-1: Example Data Vault

Data Vault models are representative of business processes and are tied to the business through the business keys. Business keys indicate how the businesses integrate, connect, and access information in their systems. Data Vault models are built based on the conceptual understanding of the business.

> Concepts such as customer, product, order, email, sale, inventory, part, service, account, and portfolio are used to represent ideas that cross lines of business. Examples of lines of business may include: sales, finance, marketing, contracting, manufacturing, planning, production, and delivery. These concepts can be represented with business keys that should cross lines of business.

The Links represent association across the business keys. The associations can change over time, some have direction (akin to mathematical vectors), others are directionless. Links are physical representations of foreign keys, or in data modeling terms: an associative entity.

> Hubs and Links do not contain context. Satellites provide the *context* defining the keys and associations *for a specific point in time*. The Satellites contain the descriptive data attributes about a Hub or Link that can change over time. Satellites are the *data warehousing* portion of the Data Vault model.

Hubs and Links are like the skeleton and ligaments of the human body – without them we have no structure. Without them, our Data Warehouses are blobs of data loosely coupled with each other. But WITH them we have definition, structure, height, depth, and specific features. We as humans couldn't survive without a skeleton. The Data Warehouse cannot survive without Hubs and Links. They form the foundations of how we hook the data together.

Finally, the Satellites are added. Satellites are like the skin, muscle, and organs. They add color, hair, eyes, and all the other components we need to be described.

By separating the concepts of descriptive data from structural data, and structural data from Linkage data, we can easily begin to assemble a picture or an image of what our companies look like. The Hubs provide the working constructs to which everything is anchored. The Links provide the assembly structure to how these anchors interact, and the Satellites define the context of all of these components.

Remember this: the Data Vault is targeted to be an Enterprise Data Warehouse. Its job is to ***integrate*** disparate data from many different sources, and to ***Link*** it all together while ***maintaining*** source system context.

The Data Vault sits between the source systems and the data marts, as mentioned in Section 2.2. Why? Because the data marts are the ***interpretation*** layer of the integrated data warehouse data. In human terms think about it this way: think about a certain event that occurred in your life that you shared with another person. Do you both remember it the same way? Do you both remember the exact details? Or is your interpretation of the event slightly different than that of your friend? This is why it's so important to separate ***interpretation*** from the ***facts.*** Let your Data Vault house the facts, and build your data marts to house the interpretation.

4.0 Hub Entity Details

Hubs are defined by a unique list of business keys. They are surrounded with additional technical metadata elements such as load date time stamp, record source and sequence number (i.e., surrogate key). Business keys may be composite (made up of more than one field), intelligent (smart-keys) – contains meaning across parts of the keys, or sequential in nature.

Unfortunately in the real-world, what appear to be "business" keys change depending on the system being used. The keys change from one state to another as the customer information passes from one system to another. These changes are typically a manual process resulting in little to no visibility at the corporate level for where a customer is in the life-cycle of business.

One of the "jobs" that a good data warehouse should perform is: gap analysis - that is: provide the business with a view of the GAP between the way the business believes they are operating their business, and the way the systems are collecting the data. By examining this gap, the business can quickly locate where they are hemorrhaging money.

4.1 Hub Definition and Purpose

The job of a Hub is to track the first time the Data Vault sees a business key arrive in the warehousing load, and where it came from. ***The Hub is a business key recording device***. The business keys in a Hub should be defined at the same semantic granularity. Hubs have several standard fields including sequence number (or id), Load Date , and Record Source as shown in figure 4-1.

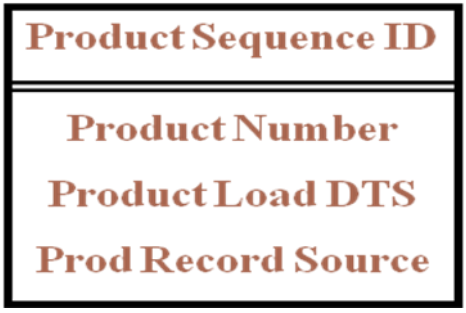


Figure 4-1: Hub Example

The purpose of the Hub is to provide a soft-integration point of raw data that is not altered from the source system, but is supposed to have the same semantic meaning. The resulting singular list of keys assists in the discovery of patterns across systems. The Hub key also allows corporate business to track their information across lines of business; this provides a consistent view of the current state of application systems. These systems are *supposed* to synchronize, but often don't – when they don't synchronize, business keys begin to be replicated and worse yet, are then applied to different contextual data sets. An example of Product Hub data is shown in Figure 4-2. Do you see and patterns in the Product # that may indicate data quality issues or other disconnects between systems?

ID	PRODUCT #	LOAD DTS	RCRD SRC
1	MFG-PRD123456	6-1-2000	MANUFACT
2	P1235	6-2-2000	CONTRACTS
3	*P1235	2-15-2001	CONTRACTS
4	MFG-1235	5-17-2001	MANUFACT
5	1235-MFG	7-14-2001	FINANCE
6	1235	10-13-2001	FINANCE
7	PRD128582	4-12-2002	MANUFACT
8	PRD125826	4-12-2002	MANUFACT
9	PRD128256	4-12-2002	MANUFACT
10	PRD929929-*	4-12-2002	MANUFACT

Figure 4-2: Hub Example Data

4.2 Hub Entity Structure

The Hub entity structure consists of these required elements: a surrogate sequence id, a business key, a load date stamp, and a record source.

> The Hub entity must NEVER contain foreign keys. If the Hub structure is compromised (i.e., the modeling standards are not adhered to), then the integrity of the data; and the flexibility of the model are immediately compromised.

Hubs must stand alone (be a parent to all other tables), they must never be children. Figure 4-3 shows the basic Hub Entity Structure.

Figure 4-3: Hub Entity Structure

4.3 What is a Business Key?

A business key is something that the business uses to track, locate, and identify information. It is the value you see in a data entry screen or may be used as a parameter filter on a report. It is a best practice to have unique business keys. The business key will also serve as a secondary unique constraint in the Hub table. Business keys are also known as natural keys. Business keys may actually be source system sequence (surrogate) ID's that have been released to business users and now are embedded in business processes. Business keys are supposed to have meaning to the business *independent from the operational system*.

5.0 Link Entity Details

Link entities act as the flexibility component of the Data Vault model. They are the glue that pulls together any related association of two or more business keys. Where business keys interact, Links are created. Link entities are generated as a result of a transaction, discovery, relationship, or interaction between business units, business processes, or business keys themselves.

Links provide flexibility to the Data Vault model by allowing change to the structure over time. Mutability of the model ***without*** loss of history is critical to the success and long-term viability of the enterprise data warehouse. In other words, the model itself can now be adapted, morphed, and changed at the speed of business without loss of auditability, and compliance. The Data Vault model also gains flexibility from this technique because of the Link entity. The Link entity (in data modeling terms) is commonly referred to as an *associative entity.*

5.1 Link Definition and Purpose

A Link Entity is an intersection of business keys. It contains the surrogate ID's that represent the Hub's and Link's parent business keys. A Link ***must*** have more than one parent table. A Link table's grain is defined by the number of parent keys it contains. Each Link represents a unit-of-work (UOW) based on source system analysis and business analysis.

The purpose of the Link is to capture and record the past, present, and future relationship (intersection) of data elements at the lowest possible grain. The Link Entity also provides flexibility and scalability to the Data Vault modeling technique. Typical examples of Links include: transactions, associations, hierarchies, and re-definition of business terms

5.2 Reasons for Many To Many Relationships

Within the Data Vault modeling constructs a Link is formed any time there is a 1 to 1, 1 to many, many to 1, or many to many relationship between data elements (business keys). The resulting physical Data Vault can capture "what the relationship was", while it captures "what the relationship is", and can adapt to "what the relationship will be in the future."

Many-to-Many relationships provide the following benefits:

1. Flexibility
2. Granularity
3. Dynamic adaptability
4. Scalability

Many-to-many relationships allow the physical model to absorb data changes and business rule changes with little to no impact to *both* existing data sets (history) and existing processes (load and query). Businesses ***must*** change at the speed of business, and IT ***must*** become more agile and responsive to handling those changes. More and more business rules are changing, faster and faster.

Through the Link entity the Data Vault mitigates the need to restructure/redesign the EDW model because the relationship changes. For example: today the business states "1 portfolio can handle many customers, but each customer must be handled by 1 and only 1 portfolio." If the model is designed in a *rigid* fashion (that is to say with parent-child dependencies) then it represents the current business rules quite well. All is well until the business (tomorrow, next year, or 2 years ago) decides to change their business rule: "now, a customer may be handled by 3 or 4 different portfolios."

One of the problems of modeling today's relationship in ***any*** data warehouse is that it makes the structures static. It forces the structures to represent today's relationship rules. These relationships have changed in the past, and will change again in the future. This is the dynamic changing nature of the business: grow, change, or die. The problem with introducing static relationships in to the model is that it also re-introduces business rules to the loading processes. It also introduces static relationship enforcement in to the loading routines. When the relationship does change, IT is forced to re-engineer the loading routines, the modeling architecture, and the queries to get the data set in to the Data Warehouse. This is an unacceptable and un-maintainable cost going forward.

The Data Vault must remain flexible, and not introduce the need for re-engineering as the model grows. By modeling the Links as a many-to-many relationship, we can easily accomplish this goal. The Link table functions to future-proof the model and provide maximum flexibility. Figure 5-1 demonstrates the reason for using a Link table:

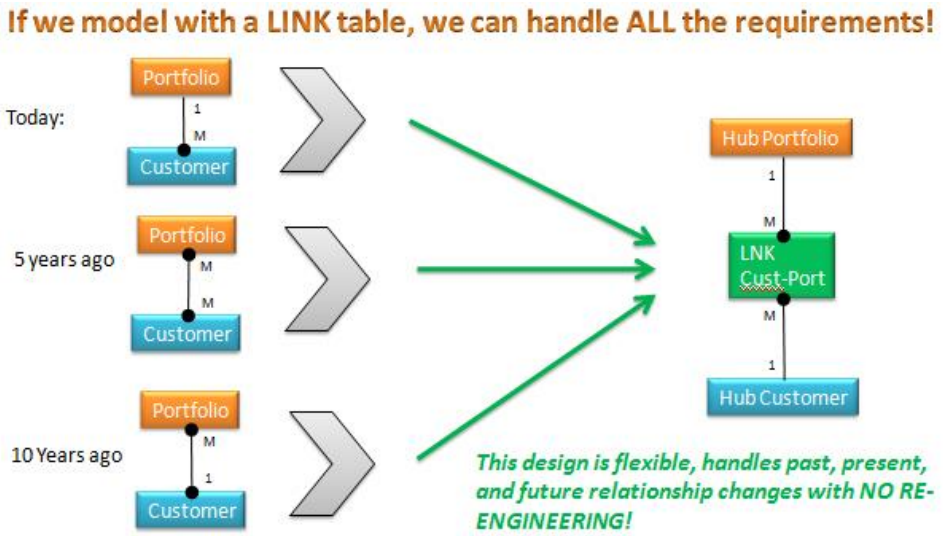


Figure 5-1: Link Table Structure Housing Multiple Relationships

5.3 Link Entity Structure

The Link entity structure consists of basic required elements: surrogate sequence id, multiple business sequence keys, load date stamp, and record source. Items such as last seen date, confidence rating, strength rating, encryption key, and possibly other metadata elements may be added for query purposes, performance purposes, and discovery purposes as business requires.

The Link entity must ***NEVER*** contain natural business keys, or begin and end dates. ***If a Link structure is compromised, then the flexibility of the data vault model is immediately compromised.*** If the structure of the Link is compromised then you are sure to need reengineering in the future. Adding business keys to a Linked table insures that it depends on the business logic for loading; this raises the complexity of the loading routines. Links must contain two or more key sequence fields (from either Hubs or Links) in order to be considered valid; a Link with a single Hub sequence key is considered a peg leg Link and is invalid. Figure 5-2 shows the basic Link Entity Structure and an example table.

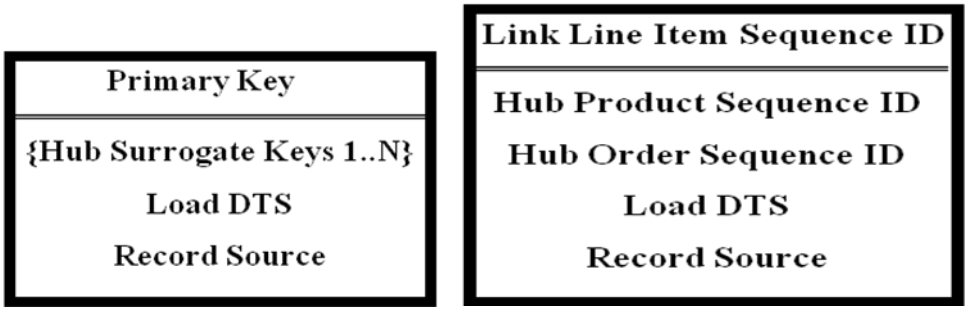


Figure 5-2: Link Entity Structure & Example

6.0 Satellite Entities

Satellite entities are the warehousing portion of the Data Vault. Satellites store data over time. Satellites are comprised of descriptive data that provide context to the keys and associations ***at a point in time*** or ***over a time period.*** Descriptive data in warehouses often changes; the purpose of the Satellite is to capture all deltas (all changes) to any of the descriptive data which occurs.

Satellites are typically arranged by type or classification of data, and rate of change. There are many different manners in which to setup classifications of data within a Satellite. For example, the attributes could be classified by data type, or by content, or by context – each of which will yield the same result physically – but a different result in the understanding or interpretation of the model.

Rate of change is yet another classification of Satellite data. Rate of change allows the Satellite to split away groups of fields that change more quickly than others. This prevents or removes the need for column data replication (of the slower changing attributes). By splitting the Satellites by rate of change, the rows are also reduced in size – allowing the data to insert more quickly, and be more responsive to real-time feeds. The lower the latency of arrival, the faster the database must respond with insert speed, the nature of these mechanics will be covered in the Data Vault implementation book.

6.1 Satellite Definition and Purpose

A Satellite is a time-dimensional table housing detailed information about the Hub's or Link's business keys. The purpose of the Satellite is to provide context to the business keys. Satellites ***are*** the data warehouse portion of the Data Vault. The Satellite tracks data by delta, and only allows data to be loaded if there is at least one change to the record (other than the system fields: sequence, load-date, load-end-date, and record source). A Satellite can have one and only one parent table.

Satellites provide the descriptive data about the business key, or about the relationship of the keys. They ***describe*** the relationship changes over time. Their job is to ***record*** the information as it is loaded from the source system. They use load dates and load-end-dates to indicate record life-cycles because most database systems today are not capable of internally representing time-series properly.

6.2 Satellite Entity Structure

The Satellite entity structure consists of basic required elements: surrogate sequence id (from the parent table), load date stamp, load end date stamp, and record source. Database engines today do not currently support (natively) time-series based table structures. Due to this limitation, the architecture is forced to compensate with Load Date Stamps and Load End Date Stamps. (**Note:** These date stamps have been described in detail Chapter 3 of the book.)

The Satellite entity must ***NEVER*** contain foreign keys (except for the single parent on which it relies). If a Satellite structure is compromised, then the flexibility of the model is immediately compromised, in other words: all possible hope of future proofing the data model is immediately lost. You are then forced to reengineer the data model in the near future when the business changes the way relationships are structured.

Satellites ***must*** have one and only one parent table, none others are allowed. Figure 6-1 below shows a standard structure of a Satellite Entity.

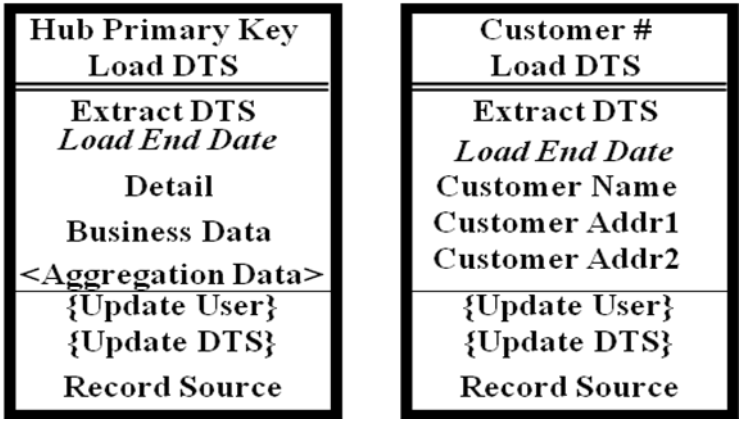
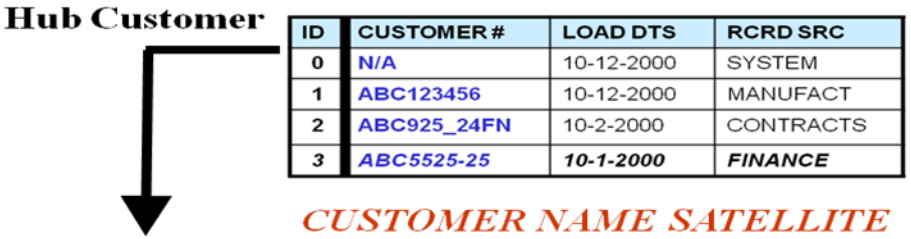


Figure 6-1: Satellite Entity Structure & Example

The primary key of the Satellite is a two-part key consisting of the PK Sequence of the parent Hub/Link combined with the Load Date time stamp. By adding millisecond timer, real-time data can easily flow directly in to the Satellite without creating duplicate primary keys (as a result of load date collisions).

6.3 Satellite Examples

Figure 6-2 shows an example of Hub and Satellite related to Customer and changes over time to the Name. Note that most real Satellites will have more than one attribute.



ID	CUSTOMER #	LOAD DTS	RCRD SRC
0	N/A	10-12-2000	SYSTEM
1	ABC123456	10-12-2000	MANUFACT
2	ABC925_24FN	10-2-2000	CONTRACTS
3	*ABC5525-25*	***10-1-2000***	***FINANCE***

CUSTOMER NAME SATELLITE

CSID	LOAD DTS	NAME	RCRD SRC
0	10-12-2000	N/A	SYSTEM
1	10-12-2000	ABC Suppliers	MANUFACT
1	10-14-2000	ABC Suppliers, Inc	MANUFACT
1	10-31-2000	ABC Worldwide Suppliers, Inc	MANUFACT
1	12-2-2000	ABC DEF Incorporated	CONTRACTS
2	10-2-2000	WorldPart	CONTRACTS
2	10-14-2000	Worldwide Suppliers Inc	CONTRACTS

Figure 6-2: Example Satellite Entities

6.4 Importance of Keeping History

History is partly what a data warehouse is all about. The Data Vault is no different, except that in the Data Vault, history is raw operational data (i.e., un-transformed). Satellite structures being what they are, can be changed, altered, and re-designed (as is documented in detail in the book).

7.0 Conclusion

This article has hopefully opened your eyes to a more flexible and agile (or at least novel) way to model a central EDW repository. While Data Vaults are not intended for direct end user query access with (or without) a BI tool, they are ideal for storing all the data and changes to that data from all of your source systems in a very scalable structure. They can then be leveraged for data mining as well as feeding data marts.

Using this modeling approach will allow you to adopt a more agile methodology as you can grow the model incrementally. You can start will just a small set of Hubs as you work with the business users to clearly identify the true business keys across the enterprise (and identify issues with those systems). You can then add Satellites to those Hubs (one at a time if needed) to capture the historical descriptive data. With Hubs and Satellites in place you can then add in the Link to capture all the important business key relationships. All along, with user invomelent, you can be doing data quality checks and show them their data (integrated across source systems and functions) in a way that they may have never seen. Who knows what insights this may lead to…

About the Authors:

Kent Graziano is a Senior BI/DW Consultant for TrueBridge Resources in Houston, Texas and a lifetime member of RMOUG. He is a certified Data Vault Master, an expert data modeler and architect with nearly 30 years of experience, including over 20 years working using Oracle (since version 5), Oracle Designer, and doing data warehousing. Kent has written numerous articles and done over 40 presentations (both nationally and internationally). He was the recipient of the 1999 Chris Wooldridge Award (from IOUG) for outstanding contributions to the Oracle user community. In 2003 he was presented with The Doug Faughnan Award for his dedicated service and outstanding contributions to RMOUG. In 2007, he was the recipient of the ODTUG Volunteer Award. In 2005 he was named one of the first Oracle ACE's by Oracle Corporation. He is a co-author of *The Data Model Resource Book*, *Oracle Designer: A Template for Developing an Enterprise Standards Document*, and *The Business of Data Vault Modeling*. Kent can be contacted at kent.graziano@att.net .

Dan Linstedt is the inventor of the Data Vault Data Model and Methodology. He has been in the IT industry and DW/BI for the past 20 years, as a consultant and systems architect, building both OLTP and EDW class systems for federal government and commercial enterprises. Dan is trained in SEI / CMMI Level 5 compliance and governance and is an expert in VLDB/VLDW and large scale (petabyte) sized system design and architecture. He is now an independent consultant helping organizations around the world build successful, scalable, and repeatable enterprise data warehouses. He has written numerous articles and white paper and has presented internationally including TDWI, DAMA, and ODTUG. He is the primary author of *The Business of Data Vault Modeling* (available at lulu.com) and the new book *Super Charge Your Data Warehouse: Invaluable Data Modeling Rules to Implement Your Data Vault* (available at LearnDataVault.com). Dan can be reached at danl@danlinstedt.com .