Data Model Overview

Modeling for the Enterprise while Serving the Individual

Debbie Smith

Data Warehouse Consultant

Teradata Global Sales Support

TERADATA
Raising Intelligence

Data Model Overview

Table of Contents

Executive Summary

The data model choice for the data warehouse is often a matter of great controversy. The desire is to offer a self-service type of environment that allows business users easy access with acceptable response times. Response time also includes the time required between the conception of a new application and the delivery of that application. What data model do you employ to provide ease of use for the business user while still being able to address current and future needs of the data warehouse in terms of updating, expansion, availability, and management? This paper will provide an overview of popular data modeling and the Teradata Corporation position regarding data modeling.

TERADATA
Raising Intelligence

Data Model Overview

Introduction

Data model choices are often a matter of great controversy when discussing building a data warehouse. When building the data warehouse, how do you build it? Do you attempt to build it for the business entity – current and future – or do you build it in a manner that satisfies current business users' needs? This paper will discuss the idea of being able to build the centralized enterprise data warehouse to sustain longevity, and use view layer to form fit individual business user needs and requirements, combining the benefits of a normalized model with the benefits of a dimensional model.

Revisit Dr. E. F. Codd's 12 Rules

In 1985, Dr. Codd published a list of 12 principles that have become the design of relational database systems guidelines (see Appendix A).[1] Dr. Codd is credited as the creator of the relational model, and within these guidelines, he outlines the structuring of information into tables, that null values are properly identified and a high-level of insert, update, and delete is maintained. Note guidelines 6, 8, and 9.

> number 6; View Updating Rule – where logical views support full range data manipulation

> number 8; Physical Data Independence – where the user is isolated from how the data is physically stored

> number 9; Logical Data Independence – where the user is not impacted should the physical data structure change

These three guidelines specifically provide a methodology to isolate users from the impact of IT activities and directly impact and lay the foundation for being able to build a logical, user-friendly data structure that is independent of the physical data structure. This foundation provisions for form fitting individual business users needs while maintaining a physical model that facilitates the needs of ETL, update frequencies, and data management requirements, enabling the enterprise data warehouse with user friendliness.

So – What is an EDW?

The foundation of this paper is built on the concept of an enterprise data warehouse (EDW). Many people use the term data warehouse to represent similar concepts; however, there can be some variances in what is meant. It is generally accepted that the data warehouse is the environment that provides for decision support within the organization. By appending the word enterprise, that environment is now thought of or expected to become reflective of the entire enterprise. Before continuing, let's discuss the environment that Teradata refers to when we say enterprise data warehouse. We define an EDW as an area where the data of the business (the enterprise) is centrally integrated, centrally stored, and accessed through common business defined methods. We believe, and our customers have shown us, that the value of centralizing the data is the synergy gained by storing the data once, managing it once and accessing it for and in many varied ways, times, methods, and reasons. (By gathering and integrating the data of the business, business users are able to have a 360-degree view of the data.) Of course, the accumulation of business data is an ongoing process and ever evolving. One of the goals is to develop the long-term strategy that provides a methodology of adding and refining business data. The higher the degree of customer success, the greater degree the infrastructure is able to answer any question, from any user (internal or external) on any data at any time, including the yet unknown queries or unknown future application needs, without the need or intervention of additional resources. This synergy allows the business to use the information of the business to quickly respond to and create changes in the market place.

Data Model Overview

Data Infrastructure

The data model is the core of the data warehouse. The decisions made when defining the data model determine the data infrastructure. This data infrastructure can impact performance, time to market for new applications, facilitate responses to changes in the market place, business users' ability to retrieve information, data latency, and the optimization of the data warehouse over the long term. The focal point of the data warehouse data modeling discussion typically covers normalization theory and dimensional theory.

Teradata® Database, as an RDBMS, is and always has been agnostic about what well defined data model is chosen. There are many customers who execute a normalized model, others using a snowflake model, others using a star schema, and many others using some variation or derivation of each/any and all. With that said, it is important to note that Teradata as a company, with its vast experience in data warehousing, does have preferences to ensure the data model employed provides the highest flexibility and responsiveness to the business for not only current needs but also future needs.

As data warehouses are evolving and taking on greater responsibility and active roles in how an enterprise conducts business, data model requirements, and expectations are changing at a faster rate to support this increased responsibility. The data warehouse is expected to reflect changes within the enterprise with increasing speed. It is also driving changes in how the enterprise conducts business by providing insight into current business practices. These increasing requirements of the data warehouse impact data modeling choices. The goal of the EDW is to address the needs of the enterprise. This requires a data model that provides an enterprise solution rather than a localized solution that only addresses specific application needs for individuals within the organization. The charge of the EDW is to enable the business at an enterprise level. To do that, hard decisions concerning data modeling must be addressed. Questions such as: should the data model reflect the needs of the enterprise or the needs of the individual? What is the impact on the data model as the needs of the individual and/or the needs of the enterprise change? Do you have to sacrifice one over the other? What if you could minimize those sacrifices? Experience has shown us that with Teradata Database, it is possible to take full advantage of Codd's guidelines of physical data independence, logical data independence, and view updating to create a data model that provides the flexibility to satisfy both current and future needs of the enterprise. And then, through the use of views, create a view model that addresses the needs of the individual. This provides a methodology to utilize both the normalized model and augment it with views that use dimensional modeling techniques, taking the best of both worlds.

With more than 35 years of experience in database technology and data warehouse design, Bill Inmon is recognized as an authority in the data warehousing industry. His lectures and papers promote a normalized model as the model of choice for the data warehouse, reserving the use of star schema models for data marts if and when necessary. He states, "It is not possible to build an effective DSS environment without an enterprise data warehouse. The design of the enterprise data warehouse is typically normalized. The classical entity relationship structuring of data accompanied by the ensuing normalization of the data structures fits very conveniently with the requirements of the usage of the enterprise data warehouse." [2]

Data Modeling Theory

Entity-Relationship Modeling

Before designing and establishing the physical data model, there's a logical data modeling process. This process includes extended discussions with the business community. During these discussions, the business requirements are identified, the data entities and elements required to meet those business requirements are established, and the relationships between the data entities are captured. These insights are later diagrammed into a representation of the business, and referred to as an Entity Relationship (ER) model. An ER model or diagram represents the data entities of the business and the relationships between those entities. At this stage, the specific functions or queries the model will be used

Data Model Overview

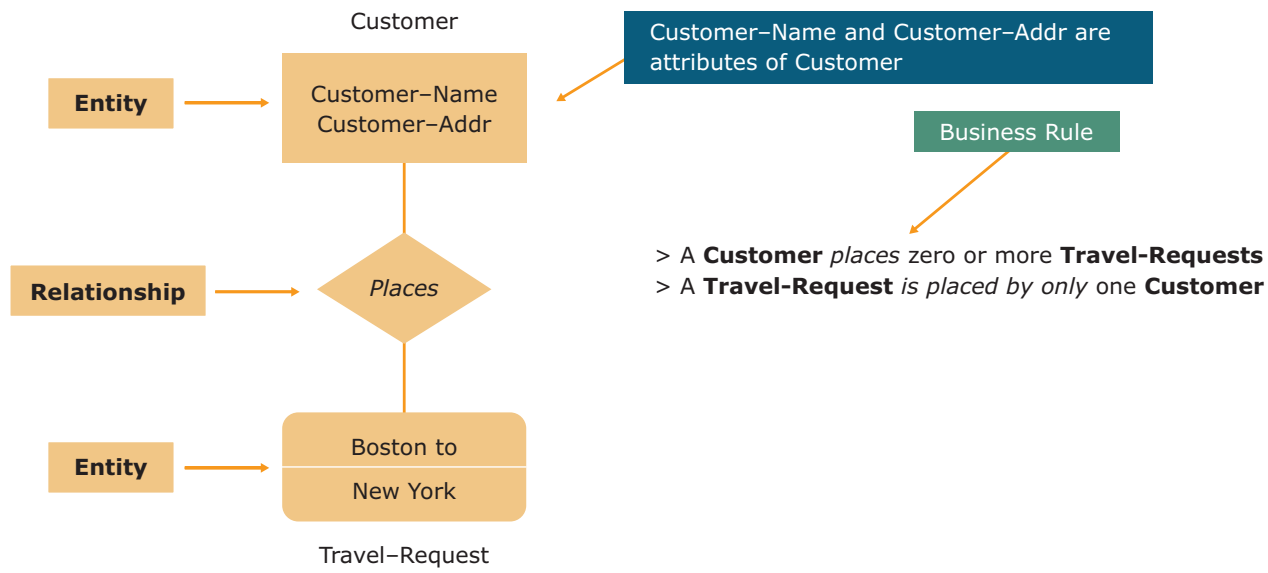

Figure 1. Example ER Diagram

to support are not included. This is a logical representation of the enterprise and is later used to develop a physical data model or schema. There are a number of tools available to map the ER diagram, ERwin®, now known as AllFusion™ Erwin Data Modeler, being the most recognized. ER modeling is guided by the rules of normalization. These are strict rules meant to ensure the essence of business relationships is captured. Examples of the components (entities and relationships) of an ER diagram are shown in Figure 1.

Dimensional Modeling

Dimensional modeling is another logical design method used to organize data for functional groups of users or business functions. It identifies the dimensions and levels within the business, separating out the facts or measures of the business. The dimensional model enforces additional rules that eliminate many to many relationships between entities, allowing only many-to-one relationships. It fuses multiple entities together into a new entity. This new entity does not directly reflect the entities and relationships that occur in the business, but is established for the convenience of storing a data point or metric that is important to the targeted group of business users. The goal of the dimensional model is to provide a presentation layer that facilitates easy navigation of the data for business users and quick access to reports.

The dimensional model is often thought of or represented as a *cube* with dimensions such as time, product, and geography. The business metric is at the intersection of these dimensions. Visualizing the dimensional model as a cube, sometimes referred to as a star schema, (See Figure 2.) makes it easy to imagine being able to slice and dice that segment of data. Creating these cubes for reporting requires understanding of what questions will be asked before designing. Each cube is designed to facilitate quick and easy access to a specific business application.

Data Model Overview

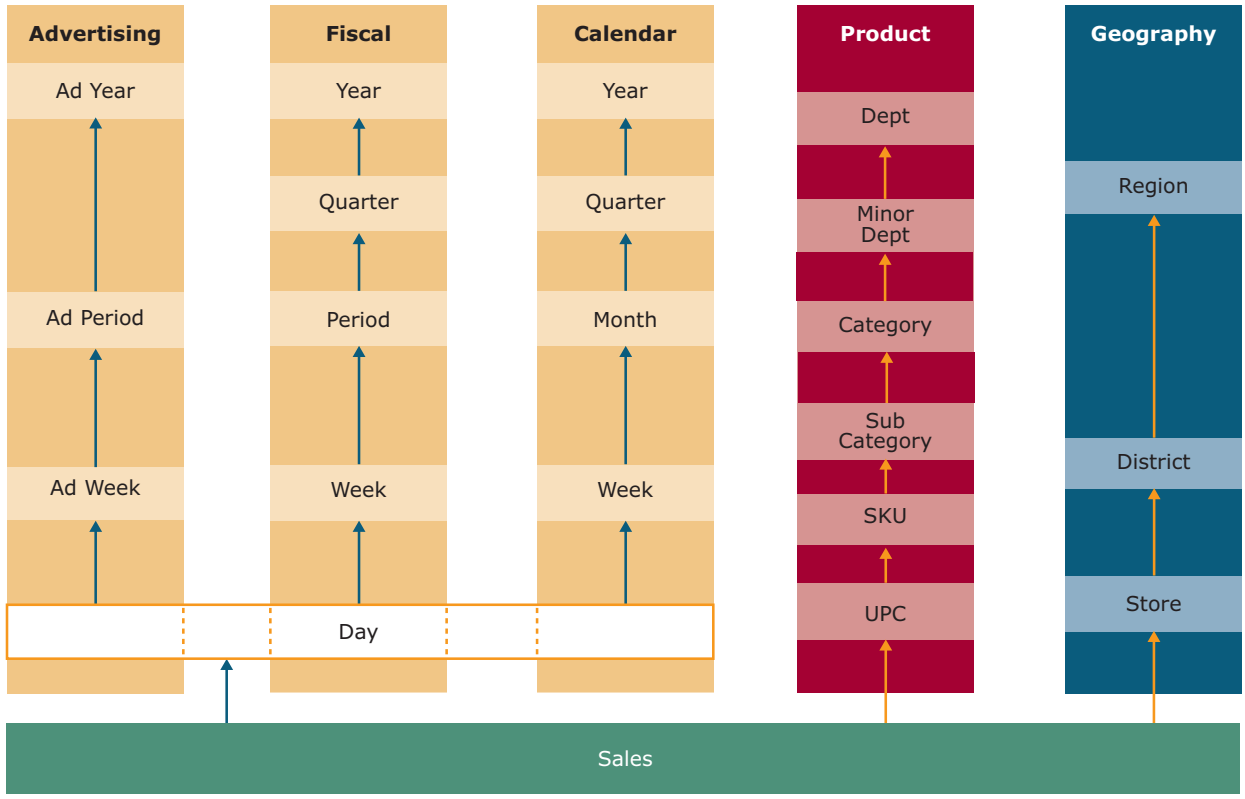

Figure 2. A Cube (Star Schema)

The dimensional model flattens the dimensions via denormalization. Business rules are used to validate and ensure additional restrictions (as in Figure 2). For example, many companies have a fiscal year that differs from the calendar year; they may even have another time dimension, such as advertising, which differs from both calendar and fiscal. If/when the advertising week does not fall within the fiscal week, business rules will be used to determine and assign the ad week to a fiscal week providing a methodology for drilling through to lower levels. In effect, the functional rules and processes are accommodated within the model. This, in turn, means that much more needs to be understood about the queries and outputs that the model is expected to support. A dimensional model can be created from an ER model, however, an ER model could not be created from a dimensional model. Once entities are fused together, separating those fused entities is difficult, if not impossible.

With a dimensional model, the central fact table is forced to a single grain, causing the initial fact table design to become brittle or inflexible. You can't incorporate new data sources without breaking the original star schema design or creating separate fact tables or data marts. To get an enterprise view, you would then have to drill across these different stars or data marts.

Data Model Overview

From Logical to Physical Modeling

Why would you create a physical model that is different from the logical model? The physical model often differs from the logical model to yield the best overall performance for specific database technologies, and not all database technologies are equal. Technologies differ so much that in an article written for *Intelligent Magazine* Neil Raden states, "As you know, the primary schema for a data warehouse is either a star schema or a "normalized" schema. The latter is a term so loosely defined that it's hard to describe, but a normalized schema typically resembles a third (or higher) normal form (3NF) schema that's not dimensional. These 3NF designs don't support query and analysis. Their sole purpose is to act as a staging area, an upstream data repository for a series of star schemas, online analytic processing (OLAP) cubes, and other structures that are directly queried by analysts. The only routine exception to this is Teradata implementations: Because of the unique characteristics of the massively parallel architecture and database optimizer, Teradata can process analytical SQL against a 3NF schema with acceptable performance."[4]

This implies the physical model is often changed from the logical model not necessarily due to business requirements but to facilitate the technology being used and ensure query speed. This transition from the logical model to physical model

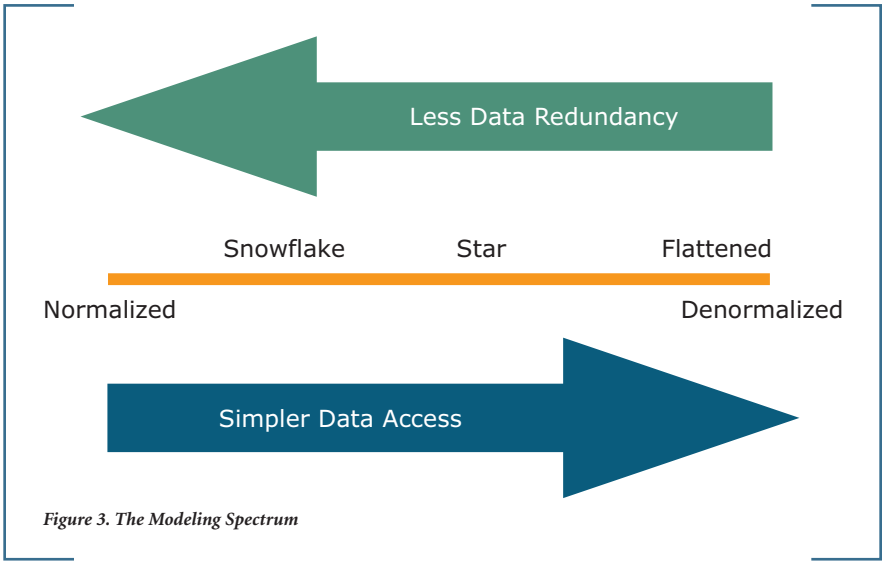

Figure 3. The Modeling Spectrum

is where the greatest variation, deviation, and difference of opinion takes shape, which of course, forms the basis for the confusion surrounding data modeling. On one end of the spectrum is the flattened denormalized model; while on the other end is the normalized model. Of course, the reality is that most customers develop a data model that falls somewhere between the opposing ends, or some combination of both. This is often the result of the data warehouse evolving to understand business demands.

In his book *Data Modelers Workbench*, Steve Hoberman discusses what he refers to as the normalization hike and the denormalized survival guide.[5] Hoberman's principles state that the logical model is completely normalized, up to fifth normal form. His analogy states that once you've accomplished this, you can see the business functions as if looking down from a skyscraper. The denormalization survival guide then provides guidelines of how to transform the normalized logical model into a physical model. *This survival guide provides some of the principles for denormalization, the basics of which are to minimize denormalization so as not to compromise the business' current and future needs.*

Physical Models

Normalized Physical Model

As you move along the modeling spectrum (see Figure 3), the further to the left your data model is, the greater the degree of normalization. A normalized model separates data into tables based on very strict rules that relate an attribute to the primary key. One of the fundamental rules of normalization is the elimination of data

Data Model Overview

redundancy within the model, keeping each attribute within each table functionally dependent upon the primary key. By following this rule, eliminating data redundancy within the model, the number of tables along with the complexity of the data model increases. As new data subjects for the data warehouse are identified, the model is extended following the same normalization rules to include the new data subjects.

[While data warehouse modelers are often embroiled in controversy over how to model the data warehouse, the world of transactional databases is in agreement that a normalized model is the optimal choice. As data warehouses evolve towards transactional decision making using very current data, the need for a more normalized physical design becomes evident.] A normalized data model provides a methodology for capturing and storing the lowest level of data, eliminates multiple updates to various tables with the same transaction, and is the model of choice for the OLTP types of transactions that are targeted and specific.

Customer experience has shown that the normalized model can answer new business questions or previously unknown questions without making changes to the structure of the database, because the relationships and entities have been represented physically and provide the greatest flexibility for the business. It eliminates data duplication and, therefore, the complexity of maintaining duplicated data. Even though there are typically more tables in a normalized model, the greater the degree of normalization, the greater the degree of storage space conservation because data is not duplicated. However, denormalized structures, such as summary tables, will and often do exist in a data warehouse that has been developed under the constructs of normalization. When the business value warrants the cost to build and manage summary table structures, they are a viable augmentation of the normalized data model.

Denormalized Model

Moving right along the modeling spectrum (see Figure 3) the degree of denormalization increases. A *flattened model* essentially resembles an Excel spreadsheet. Denormalized data models flourished out of a need to provide business users with quick and easy access to data without the requirement of understanding the underlying data model. Business users, whose job does not include knowing SQL or the logic of data models and, therefore, how to join tables together, needed easy access to information. However, those same business users did understand spreadsheet formats and, in fact, often requested information in a report format. Therefore, when data was presented in a two-dimensional form (similar to a spreadsheet); as opposed to a series of connected tables, business users were protected from data model complexity. Because the data model was generated to address specific business constituencies, new user constituencies (often called power users) were created within the organization to code and generate reports for business users. These power users became so busy that requests were often queued for weeks at a time. Business users wanted direct access to information, and they wanted it instantaneously. Denormalized models provided for simple, easy navigation of the data. The data are typically aggregated along one of the lines of dimension. The types of queries requested are well understood, and the model is established to answer those questions. These pre-defined models ensure the known query will return an answer set quickly. When new queries are identified, the model is extended to include a duplication of existing data in a form that allows the business user to navigate and quickly retrieve the answer set for the new set of queries.

The *star schema model* is typically only considered for use in data marts or the data warehouse and is built to address a specific business need, report, or application. Each star schema is built as an individual data mart. When the business needs a new report, another star schema, or data mart, is built. If the needed dimensions are the same as what has already been built, they are duplicated to this new data mart. Each of these data marts caters to the needs of the individual. There may be a temptation to think of a group of these data marts as constituting a data warehouse. However, they are distinct structures that have duplicated data in

Data Model Overview

order to store it either in a different format or on a different platform, either a different database or just different hardware. The star schema model doesn't have the goal of storing data once for multiple uses. Its goal is to facilitate end-user usage. When the business community has a new request, a new star schema is typically built to facilitate the request.

For example, when a business user wants to know the monthly sales of products, the report produced would have headings such as Item, January Sales, February Sales, March Sales, etc. The normalized Sales table would consist of columns of Item, Date, and Sales. This table would provide the greatest flexibility, allowing for the next questions. These questions might include what are sales per week for specific items, what were fiscal month sales, what items sell on Mondays, or what are weekend sales. However, because it is often difficult for users or even database software to make the conversion from the table structure to the report layout, in the dimensional model, DBAs simply store the table in the denormalized report format (see Figure 4).

In an effort to maintain conformity, the star schema is built with constrained dimensions. Ralph Kimball explains that data are placed in a staging area for transformation. These data, using what he refers to as the data warehouse bus architecture, are then propagated out to all the different data marts that require it,

Item	Date	Sales		
100012	01102001	10.00		
300012	02122001	3.00		
200012	01152001	2.50		
100012	03042001	15.00		
Item	**Jan Sales**	**Feb Sales**	**Mar Sales**	**Apr Sales**
100012	345.00	450.00	326.50	245.90
200012	456.60	376.50	210.00	390.00
300012	254.00	112.00	310.00	295.00
400012	510.00	610.00	590.00	545.00

Figure 4. Example table in denormalized format

forcing all dimensions to conform to what is established in the staging area. This bus architecture is the roadwork that connects all the different data marts and servers being used. Kimball states, "It is acceptable to create a normalized database to support the staging processes; however, this is not the end goal. The normalized structures must be off-limits to user queries because they defeat understandability and performance. As soon as a database supports query and presentation services, it must be considered part of the data warehouse presentation area. By default, normalized databases are excluded from the presentation area, which should be strictly dimensionally structured." [6]

Contrasting Ralph Kimball's belief that data warehouses are best suited for a star schema, Bill Inmon firmly believes the EDW has at its heart a normalized model, reserving the use of star schema models for data marts if and when necessary.

Inmon states, "In short, simply doing dimensional modeling as a basis for data warehouse design leads down a dark path when multiple star joins are considered. It is never apparent that there is a problem with star joins when you are looking at just one star join. But when you look at multiple star joins, the limitations of dimensional modeling become apparent. Does this mean that dimensional modeling is invalid as a database design technique for data warehousing? The answer is not at all. Dimensional modeling and star joins fit the bill very nicely for data marts. In fact, if I had to design a data mart tomorrow, I would not consider using any other approach. But, when it comes to the foundation data, it's another story. The foundation data – the data warehouse – requires a different treatment than dimensional modeling altogether. The data warehouse, which is the proper foundation for all DSS activity, including star joins, requires very granular, very

flexible data. The ideal structure for the data warehouse is normalized data. The normalized data can be bent and shaped any old way."[7]

"The fact table is connected to the dimension tables by means of foreign key relationships. The keys to the dimension tables sit in the fact table. There may be many relationships that find their way into the fact table. In addition the fact table carries with it other non key data if needed. In other words, the dimension table may share with the fact table data other than just the key. When the fact table and the dimension tables are connected, they form a star, hence the name 'star join'. The net effect of the fact table is a structure which is highly denormalized and which is very efficient to access. Once the fact table is created, it can be efficiently analysed. But there is a tradeoff. The extreme efficiency of the star join also makes it inflexible. If there is a desire to see the data in the star join in a manner other than that for which the structure is built, then the star join is very inflexible."[8]

Impact of Information Delivery Tools

Denormalized models, specifically star schemas, gained general acceptance on the basis of the ease provided for business users to directly access data. After gathering the detailed business requirements, the developers often determined that to mitigate risk and provide the business users what they requested, a denormalized model would be the easiest to implement. Since the data warehouse was updated or refreshed on a weekly or even monthly basis, the update schedule mitigated issues surrounding the complexity of updates to maintain simple data models. Business users were expected to know, understand, and code simple SQL for retrieving information, allowing IT to turn the data warehouse over to the business users and only maintain responsibility for keeping it updated. When the business began coming up with additional questions that the data model couldn't address, and joins between star schemas became complex, information delivery tools were developed to help business users retrieve the information they needed through an easy interface.

While many of these information delivery tools have grown up expecting this star data model, many are growing in sophistication and technologically developing to be able to recognize and use more normalized data models and making it easier for end users to navigate a more complex schema.

Impact of Extraction, Transformation, and Loading (ETL) Tools

While Information Delivery tools were developed to assist the business user in retrieving information from the data warehouse, ETL tools were developed to assist in extracting data from transactional systems. Since the ETL tools are meant to extract data from the transactional systems, and those are in normalized format, ETL tools work better with the normalized data model.

If the data warehouse is built with a denormalized data model, then most require additional steps to land the data in a separate area for ETL. While the ETL tools work well with a normalized model, moving those data into a denormalized model requires manipulation or transformation. The suggested way of handling this manipulation or transformation is the creation of a data staging area. This staging area is used to land the extracted data then manipulate or transform them to the format needed before loading. This staging area also provides the platform from where data, in particular dimensions, can be duplicated out to the different star schema data marts.

When loading to a normalized model, the need to transform the data for modeling purposes is significantly less, and can actually happen during the load process, eliminating the need for a staging area. In a normalized model, those using a staging area are typically using it to correct data quality issues that exist in the transactional systems. For example, ensuring character fields follow the same set of rules. If an address contains the word 'street', then all formats of 'street' are transformed to be the same, for example ST becomes Street and St. becomes Street.

Data Model Overview

Surrogate Keys

In general, a logical key comprises the data fields whose value ensures uniqueness for each row, and whose value will not change during the life of the row. Implemented physically, it is a natural key. Often, the terms key and index are used interchangeably, but they are not the same. An index is a mechanism used to optimize performance. The physical implementation differs depending on the database management system employed.

A surrogate key is an artificial key, or generated key, that is used as a primary key in substitution for natural keys. Using a surrogate key is a choice rather than a requirement. Typically, they are numerical and randomly generated to uniquely identify a row. The use of a surrogate key is often promoted on the basis of adding flexibility and data independence to the data model. In business, change is an ongoing fact of being in business. Businesses are involved in mergers and acquisitions, new products and channels are added, they are reorganized, and they enter into new markets. There are a couple of arguments for surrogate keys. One is that as business changes, so do the values, and potentially, the format of natural keys. When the natural key is used as the primary key, changing either the value or the format of that key is an intrusive effort, which could result in loss of historical information and data relationships. When a surrogate key is used as the primary key, the impact of changing the value of the natural key is the same as changing the value of any column within the table – minimal. Another argument is that when the natural key is large or consists of multiple columns, a surrogate key would be smaller and require less space as it is populated throughout the data model. To facilitate access, a process is developed that will lead to the surrogate key for joins and retrieval of information.

In a data warehouse, which brings together data, often disparate data, from the transactional environment, surrogate keys may provide a viable methodology to do so. The data warehouse is subject oriented, which contrasts the application or function orientation of transactional environments. Integrating and reorganizing multiple systems that represent the same subject, such as customer, can require the integration of data that does not occur naturally in the operations environment. A surrogate key will uniquely identify each instance of that integrated subject, each customer, without worry of duplication in the future as the business continues to change. This provides a stable data model that will withstand the test of time. However, beware not to use surrogates as replacements for good data modeling practices.

Before compromising to use surrogate keys, you must consider some things. To generate the surrogate key, a process must be put in place to ensure uniqueness and consistency. Updating and inserting new rows require first the creation of the surrogate key and also a lookup process to ensure the newly generated key hasn't been used. Since the user doesn't know the surrogate key, a process to link the values the user does know with the surrogate value must be established to retrieve information. Typically, this requires additional secondary indexes. Using surrogate keys tends to add both a column and a unique index to each table, sometimes multiple secondary indexes, and in some cases, entirely new tables. Surrogate key values also must be propagated to all the dependent tables as *foreign keys*.

While surrogate keys are frequently with star schemas, their use is not reserved for specific data models. The decision to use or not use a surrogate key should be part of the data model design considerations. That decision should be predicated on what the data modeler believes to be best for the organization? Will there be requirements for the natural key to change? Is there a need to ensure that every row is unique? Will there be a resulting impact from adding secondary indexes?

For example, if the natural key for an order table is order number, and a business user wanted to know how many units of product 'X' were ordered by company 'ABC', chances are he wouldn't know the order number. But would the business user know the product number and/or the company name? When a surrogate key is used as the primary key, order number, product, and ordering company would probably each be made secondary indexes. When the natural key of order number is used as the primary key, product and ordering company would probably each be made secondary indexes.

Data Model Overview

SCD Type One		SCD Type Two			SCD Type Three		
SKU	Catg	SKU	Catg	Updated	SKU	Catg	Prev Catg
1001	01	1001	03	20021029	1001	03	01
1002	02	1001	01	20010101	1001	01	15
1003	03	1002	02	20010101	1001	15	20
		1003	01	20020915	1002	02	02
		1003	03	20010101	1003	01	03
					1003	03	12
					1003	12	05
					1003	05	03

Figure 5. Methods to address SCD

Outside of the maintenance issues of the surrogate key, the data modeler must determine when it is wise to use either a surrogate key or a natural key. For example, while order number may make sense to use as the primary key, in the case of a customer table, it may make sense to use a surrogate key for customer id. The decision to use a surrogate key should be evaluated on a case by case occurrence and not entered into as a standard.

Changing Dimensions

A changing dimension is often referred to as a Slowly Changing Dimension (SCD). The impact of time on the data warehouse can be significant. Time has a tendency to generate many changes. Customers move and their address changes; people change their names. Area demographics change, populations shift. Products move from one category to another. These are changing dimensions. Time impacts all data models, and dimensions will change over time.

However, a changing dimension is only a problem when the model is built on dimensions. Typically, there are three methods used to address SCDs (see Figure 5).

- Type one updates in place the dimension value as it changes. This provides a dimension that is always current, eliminating the capture of history because the association of the data with the old dimension value has been overlaid.
- Type two inserts a new row into the table as the dimension changes. This method provides both current and history information of the changing dimension, and usually involves carrying an effective date to indicate which row is current.
- Type three updates the dimension in place after moving the changed dimension to an old column. This method provides for current and most recent changes.

A normalized model is typically developed as type two, developing a history as changed rows are inserted into the table. If the business application requires only the current value, a view could be used to present to the application current values only. Those applications that need history or the ability to identify values during a specific time frame would then have access to all the history data.

While the normalized model provides for the type two SCD, the construct of the dimensional model increases the challenge of providing for changing dimensions. Included in this challenge is the need to determine how to reflect the changing dimension. Which type satisfies the business need for the dimension? Does it satisfy all the business needs? Which method satisfies the needs of all the data marts in the dimensional model?

Data Model Overview

The Teradata Direction

Identifying the most appropriate choice of data model for a Teradata solution is a two-step process. First, you must distinguish between what Teradata Database lends itself to and what Teradata professionals advocate. Teradata Database is impartial to the data model. The database has the technical ability and the power to perform with any well-designed data model. Teradata's ability to parallelize every aspect of query processing eliminates technical issues and complexity of decision support processing of large volumes of data, including multi-table joins and joins within joins that are often the catalyst for denormalized data models for some technologies. Teradata engineering is always reviewing and identifying areas within the optimizer to improve performance whether the data model is normalized or the denormalized star schema. Join efficiencies improvements, such as large table/small table joins in the early 1990s, were targeted at star schemas. Advanced indexing features, join index introduced in Teradata Database V2R3 and aggregate join index introduced in Teradata Database V2R4, provide a methodology for users of Teradata Database to have a normalized model and use these features to create star schemas if and where processing merits it. Teradata Database V2R5 has made generating surrogate keys easier with the introduction of the Identity column. Optimizer learning and intelligence occurs without regard to data model.

Years of field experience have shown Teradata professionals that a normalized model of atomic data provides the greatest flexibility and, therefore, often the greatest benefit for long-term business benefit. A well integrated data warehouse is a valuable mechanism in business operations. As businesses use the data warehouse to uncover patterns in their manufacturing processes, their ordering processes, or in customer purchasing habits, business processes change. The business expects the data warehouse to enable and reflect those business changes. If business changes require an additional column (dimension) to one of the act tables or product in one category is transferred to another category, the business expects the data warehouse to reflect these changes based on a business timeline. If that timeline is negatively impacted by time requirements in addressing data model changes to enable and reflect the business changes, the data warehouse becomes an obstacle to the business's success. Customer and field experience repeatedly shows that a normalized model provides for the most flexibility and ease in delivering new products to market and for facilitating rapid business changes. So Teradata professionals often suggest starting with a normalized model because of flexibility and ease in adding new data elements. However, this does not mean that the data warehouse follows all the rules of third normal form in the strictest sense. Based on business requirements, summary tables are often found useful to augment the normalized model. The goal is to provide a model that supports the business through all of the changes that occur over time.

One of the key factors that sets the Teradata Database apart from all others is the ability to use a normalized data model that facilitates ease of management, ease of expansion, simplifies load strategies, and allows for full integration of enterprise data. This facilitates building for the enterprise. Addressing the needs of the individual becomes as easy as applying a view to the underlying data structure. This view then allows IT and the data warehouse to address the needs of the individual.

Views are a transparent layer that is on top of the underlying data structure and provide a methodology of creating a presentation layer that eases business user access. In Teradata Database, views don't require space and will transition the underlying data model into nearly any presentation with such efficiency the cost of using views is mitigated to simply the management of them.

Impact of the Data Warehousing Evolution

As data warehousing evolves to an active environment where frequent and/or continual data updates are required, where processing becomes event driven rather than report driven, where the need to address changes in the market place are expected with increasing speed, the data model becomes even more critical. The mixed workload of the shorter tactical queries with the longer strategic queries, against the same data creates additional denormalization challenges. Evolving to an

Data Model Overview

active data warehouse eliminates many of the known variables, such as what queries will be submitted, who will submit those queries, which were used in defining the data models of early data warehouses. To accommodate this evolution, some of Teradata's competitors' technologies are beginning to add functions and features that enable a more normalized model. Tools such as information delivery tools are also preparing for this evolution. As the data warehouse evolves to an active environment, the requirement for fresh data to be available and accessible erodes the time available to maintain denormalized models.

The data model chosen for the data warehouse should be chosen with a clear understanding of the benefits of each. The early days of data warehousing tended to depend on a denormalized model. Much of that decision was based on addressing specific business needs rather than addressing the breadth of the business, or on existing technology that limited choices. During those early days, when Teradata professionals recommended a normalized model, it was considered unusual. As data warehousing evolves to being active and as the need and requirement increase for businesses to change, the idea of a normalized data model for the data warehouse no longer seems so radical. In fact, many of the leading analysts and data warehouse professionals understand the value of a normalized model and are becoming more critical of denormalized models.

Summary

The goal of a data warehouse is to provide the business with a tool that facilitates and enables the business to make better business decisions and to take timely action on those decisions. The robustness of the data infrastructure determines the long-term success of the data warehouse and the ability of the business to harness the information for its own success. The robustness of the Teradata Database provides for the ability to combine the best of all worlds. The DBA can generate a DBA-friendly normalized data model that facilitates the full integration of data representing the enterprise, following the teachings of Codd, Date, and Inmon. The use of views, a presentation layer that employs the user friendliness of dimensional modeling, enables catering to the needs of the individual, interfacing with the data warehouse for business user ease of access, capitalizing on Codd's guidelines for relational databases, and following the teachings of Kimball. This allows for data to be centrally stored, eliminates redundant data easing data management, provides a methodology to support future business needs by capitalizing on existing data structures and shortening time to market.

For example, one Teradata customer had their business users request a new application from IT. After the applications group said that it would take 6 months just to find the data and once found, then they would have to determine the time required to generate the application. The business users left the meeting believing they would be unable to resolve their issues in a timely manner, time that had a direct impact on the success of the business. After determining the data was available in the data warehouse, in a normalized format, the data warehouse group was able to generate a presentation layer that resolved the business needs within a couple of hours. The ability to do this was valuable to the business bottom line.

The data model for the data warehouse should reflect requirements of the business, and it should enable the business. Taking a hard line with any specific data model type won't satisfy the business needs. However, the impact of time generates change within the business, and the data warehouse requires the flexibility to address and yield to those changes. A normalized data model is the correct place to begin.

Debbie Smith *is a Data Warehouse Consultant in Global Sales Support. She had 14 years of experience with Teradata systems at a retail customer. Her responsibilities while with the retail customer included application development/support, database administrator, business power user, and responsibility for developing, implementing, and managing an Information Delivery strategy for business end users. Since joining Teradata, Debbie has worked extensively with prospects as well as new and mature data warehouse customers to implement effective tactical and strategic data warehousing initiatives.*

Data Model Overview

Appendix A – Codd's 12 Rules

Rule 1: The Information Rule

All data should be presented to the user in table form.

Rule 2: Guaranteed Access Rule

All data should be accessible without ambiguity. This can be accomplished through a combination of the table name, primary key, and column name.

Rule 3: Systematic Treatment of Null Values

A field should be allowed to remain empty. This involves the support of a null value, which is distinct from an empty string or a number with a value of zero. Of course, this can't apply to primary keys. In addition, most database implementations support the concept of a non-null field constraint that prevents null values in a specific table column.

Rule 4: Dynamic On-Line Catalog Based on the Relational Model

A relational database must provide access to its structure through the same tools that are used to access the data. This is usually accomplished by storing the structure definition within special system tables.

Rule 5: Comprehensive Data Sublanguage Rule

The database must support at least one clearly defined language that includes functionality for data definition, data manipulation, data integrity, and database transaction control. All commercial relational databases use forms of the standard SQL (Structured Query Language) as their supported comprehensive language.

Rule 6: View Updating Rule

Data can be presented to the user in different logical combinations, called views. Each view should support the same full range of data manipulation that direct-access to a table has available. In practice, providing update and delete access to logical views is difficult and is not fully supported by any current database.

Rule 7: High-level Insert, Update, and Delete

Data can be retrieved from a relational database in sets constructed of data from multiple rows and/or multiple tables. This rule states that insert, update, and delete operations should be supported for any retrievable set rather than just for a single row in a single table.

Rule 8: Physical Data Independence

The user is isolated from the physical method of storing and retrieving information from the database. Changes can be made to the underlying architecture (hardware, disk storage methods) without affecting how the user accesses it.

Rule 9: Logical Data Independence

How a user views data should not change when the logical structure (tables structure) of the database changes. This rule is particularly difficult to satisfy. Most databases rely on strong ties between the user view of the data and the actual structure of the underlying tables.

Rule 10: Integrity Independence

The database language (like SQL) should support constraints on user input that maintain database integrity. This rule is not fully implemented by most major vendors. At a minimum, all databases do preserve two constraints through SQL.

No component of a primary key can have a null value. (See Rule 3)

If a foreign key is defined in one table, any value in it must exist as a primary key in another table.

Rule 11: Distribution Independence

A user should be totally unaware of whether or not the database is distributed (whether parts of the database exist in multiple locations). A variety of reasons make this rule difficult to implement.

Rule 12: Nonsubversion Rule

There should be no way to modify the database structure other than through the multiple row database language (like SQL). Most databases today support administrative tools that allow some direct manipulation of the data structure.

(From: ITWorld.com, *'Codd's 12 Rules – Data Management Strategies 05-07-2001'.)*

Data Model Overview

Appendix B – A Comparison

	Normalized Model	Denormalized (Star) Model
Active Data Warehousing	Supports active data warehousing initiatives.	Challenges the ability to be active.
Any Question	The flexibility of the normalized model provides for any question to be answered.	The simplicity of the denormalized model limits what questions can be answered without changes.
Complex Analysis	A normalized model supports complex analysis.	The enforced simplicity of a denormalized model is unable to support complex analysis.
Data Duplication	The rules of normalization require a single version of data.	Data is often duplicated multiple times to satisfy different groups of questions.
Data Granularity	The normalized model easily stores the lowest level of data.	To maintain the ease of navigation, the denormalized model is typically aggregated along one of the dimension lines.
Data Navigation	The normalized model is more difficult to navigate. Business users have a difficult time at best in understanding the model, the necessary joins and sometimes even the relationships between data elements.	The denormalized model is specifically designed for easy navigation by the business user.
Flexibility	The greater the degree of normalization in the data model, the greater the flexibility.	The denormalized model is meant to be simplistic and is designed for specific requirements. As requirements change, changes to the model are needed.
Maintenance	Updates to the normalized model are easier because a piece of information is in one place and requires one update. The normalized model is the standard for the transactional environment. Continuing that model throughout the data warehouse provides a closer link and reflection of transactional processes.	Updating the denormalized model requires preprocessing, aggregation and longer time frames.

Appendix C – Glossary

Attribute
Data that is attributed or describes an entity.

Cube
A way to visualize how a dimensional model is organized. It is also a type of Dimensional Model design where the measures and dimensions are stored in a format that is expected by an OLAP tool (many times proprietary format). One cube may represent one Fact table with its contributing Dimensions.

Data Mart (logical)
The data is stored in one place but Views are used to present to the end user a portion or derivation of the underlying data for their purposes. Data is not replicated.

Data Mart (physical)
A segment of data that is aggregated or preprocessed to enable specific end-user queries and requirements.

Data Warehouse
A reflection of integrated data that is organized by subject areas and is stored to address cross functional and multiple end-user requirements and queries.

Dimension
The corporate 'line' along which facts are measured (time, product, geography).

Dimensional Model
A design method that models the data based on a predefined set of business questions to facilitate ease of access and response time speed.

Drill (Across)
Movement of end-user request across different dimensions.

Drill (Up or Down)
Movement of end-user request through data along a dimension.

Entity
An identifiable object that is distinguishable from all other objects.

Entity Relationship Model
Shows the enterprise entities (or objects) and the relationships between the entities in that enterprise, without duplication.

ETL
Terminology given to a group of tools that extract data, transform it and then load it. Sometimes the sequence of processing changes to reflect ELT which is extract, load and then transform.

Fact
A measurement or value (quantity).

Flattened Model
Made up of rows that contain the data elements of all the column headings of a business report so that nothing has to be calculated to generate portions of that report.

Foreign Key
Columns in a table that reflect primary keys of other entities. Determines the relationship between entities.

Hierarchy
The arrangement or ranking of entities into a graded series.

Intelligent Key
Key is made up of or consists of values that reflect a business requirement or use.

Natural Key
The data fields whose value ensures uniqueness for each row, and the value will not change during the life of the row.

Normalized Model
Model of entities designed to remove data redundancy, the higher the 'form' (2NF, 3NF, 4NF, etc) the less redundancy.

OLTP
Online transaction processing. Consists of a set of instructions that serially, mechanically and repetitively processes many thousands of transactions through a predefined access path that ultimately updates the underlying data store. OLTP transactions are used by a transactional system that is focused on the acquisition of data. OLTP transactions normally have strict service level agreements (SLA) attached to them requiring sub-second response time. OLTP concentrates on data update and/or inserts.

Data Model Overview

Primary Key

Unique identifier for rows in a table.

Schema

Structured framework that, for purposes of this paper, provides and represents relationships between entities.

Service Level Agreement (SLA)

An agreement between IT and the business that pertains to a performance aspect of a process. The agreement could be based on availability, enforcing that data is loaded by a specific timeframe, or it could be based on retrieval processing enforcing a processing time for specific queries or types of queries.

Slowly Changing Dimensions (SCD)

The impact of time on a dimension and that dimensions changes (i.e. customer addresses).

Snowflake Schema

Made up of multiple Star Schemas without the duplication of dimension tables that can be joined together.

Star Schema

Individual fact table surrounded by dimensional tables to which it will join.

Surrogate Key

Artificial key used as a substitute for natural data keys (i.e. customer id).

Appendix D – Endnotes

1. Dr. E. F. Codd, *Codd's 12 Rules – Data Management Strategies,* ITWorld.com, 05-07-2001.
2. William Inmon, *Enterprise Data Warehouse,* http://www.billinmon.com/library/articles/introedw.asp.
3. Ralph Kimball and Margy Ross, *The Data Warehouse Toolkit, Second Edition,* Wiley Computer Publishing, New York, 2002, pp 11-12.
4. Neil Raden, edited by Ralph Kimball, *Intelligent Enterprise, Data Warehouse Designer, 'Real Time: Get Real, Part II',* www.intelligententerprise.com, June 30, 2003.
5. Steve Hoberman, *Data Modeler's Workbench, Tools and Techniques for Analysis and Design,* John Wiley & Sons, New York, 2002.
6. Ralph Kimball and Margy Ross, *The Data Warehouse Toolkit, Second Edition,* Wiley Computer Publishing, 2002, page 9.
7. William Inmon, *The Problem with Dimensional Modeling,* http://www.billinmon.com/library/articles/artdimmd.asp.
8. William Inmon, *Star Joins,* http://www.billinmon.com/library/articles/starjoin.asp.

Appendix E – References

Ralph Kimball, *The Data Warehouse Toolkit, Practical Techniques for Building Dimensional Data Warehouses, Solutions from the Expert.,* John Wiley & Sons, New York, 1996.

Daniel L .Moody, Department of Information Systems, University of Melbourne, Kortink, Mark, *From Enterprise Models to Dimensional Models: A Methodology for Data Warehouse and Data Mart Design,* Simsion Bowles & Associates, 2000.

William Inmon/Peter Meers, *The Dilemma of Change: Managing Changes over Time in the Data Warehouse/DSS Environment, White paper on http://www.billinmon.com/library/whiteprs/Dilemma of Change.pdf,* March 2001.

TERADATA
Raising Intelligence