2012

6NF Conceptual Models and Data Warehousing 2.0

Curtis Knowles
Georgia Southern University, ck01693@georgiasouthern.edu

6NF CONCEPTUAL MODELS AND DATA WAREHOUSING 2.0

Curtis Knowles
Georgia Southern University
ck01693@georgiasouthern.edu

ABSTRACT

Sixth Normal Form (6NF) is a term used in relational database theory by Christopher Date to describe databases which decompose relational variables to irreducible elements. While this form may be unimportant for non-temporal data, it is certainly important when maintaining data containing temporal variables of a point-in-time or interval nature. With the advent of Data Warehousing 2.0 (DW 2.0), there is now an increased emphasis on using fully-temporalized databases in the context of data warehousing, in particular with next generation approaches such as Anchor Modeling. In this paper, we will explore the concepts of temporal data, 6NF conceptual database models, and their relationship with DW 2.0. Further, we will also evaluate Anchor Modeling as a conceptual design method in which to capture temporal data. Using these concepts, we will indicate a path forward for evaluating a possible translation of 6NF-compliant data into an eXtensible Markup Language (XML) Schema for the purpose of describing and presenting such data to disparate systems in a structured format suitable for data exchange.

Keywords

6NF, XML, sixth normal form, anchor modeling, temporal database, semantically temporal data, semantically stable data, semantic data change, valid time, transaction time, DW 2.0, Dimension Fact Model, DFM

INTRODUCTION

A temporal database is a database which contains aspects of time analysis. Specifically, such databases often deal with two concepts of time – valid time and transaction time. Valid time is defined as the time period during which a data fact may be true with respect to the real world (i.e., a time when something actually happened), while transaction time refers to the time period during which a data fact may be stored in the database (i.e., a time when something is measured and/or reported).

The concepts of valid time and transaction time were introduced and described as a part of TSQL2, a language specification developed in 1993 in response to a proposal by Richard Snodgrass for the development of temporal extensions to the Structured Query Language (SQL) (Snodgrass and Ahn, 1986). In this research, we will include the modeling of both concepts of valid and transaction time, combining them to represent a bitemporal database which is able to store not only current data, but also historical and even future data expected to occur.

TEMPORAL DATABASE DESIGN AND 6NF DATA MODELING

In his book *Temporal Data and the Relational Model*, Chris Date points out several shortcomings when attempting to model fully temporalized data using standard Fifth Normal Form (5NF) principles. To illustrate these shortcomings, first consider a non-temporalized, 5NF-compliant table schema *(see Figure 1)* in which is defined a predicate for the Suppliers relation variable (relvar) S as such: *Supplier S# is under contract, is named SNAME, has status STATUS, and is located in city CITY*:

S

S#	SNAME	STATUS	CITY
S1	Smith	20	London

Figure 1. Non-temporalized Table Schema

To introduce a temporal element into table S, we simply add a temporal attribute SINCE to the table which accounts for valid time *(see Figure 2)*. The resulting table S_SINCE now represents that supplier S# has been under contract since day SINCE:

S_SINCE

S#	SNAME	STATUS	CITY	SINCE
S1	Smith	20	London	D04

Figure 2. Semi-temporalized Table Schema using SINCE

Alternatively, we may want to track an interval period during which a relvar is true *(see Figure 3)*. The resulting table S_DURING would represent that supplier S# was under contract during time period DURING:

S_DURING

S#	SNAME	STATUS	CITY	DURING
S1	Smith	20	London	[D04:D06]

Figure 3. Semi-temporalized Table Schema using DURING

In both of these cases, even though a temporal element has been introduced into the database, the relvars are badly designed for two reasons:

1. One temporal attribute alone is insufficient to model conditions where each of the other individual attributes may vary independently of one another over time, and

2. Data can be lost when single temporal attributes are updated. Historical data thus cannot be tracked in this scenario.

Due to these shortcomings, Date refers to this type of modeling as "semi-temporalizing" (Date, Darwen and Lorentzos, 2002).

To overcome the first issue, a SINCE attribute may be added for each descriptive attribute in the table *(see Figure 4)*. The result is a horizontal explosion of table S_SINCE which now can be defined as follows - *Supplier S# was under contract since day S#_SINCE, has been named SNAME since day SNAME_SINCE, has had status STATUS since day STATUS_SINCE, and has been located in CITY since day CITY_SINCE*:

S_SINCE

S#	S#_SINCE	SNAME	SNAME_SINCE		...
S1	D04	Smith	D04		...

...		STATUS	STATUS_SINCE	CITY	CITY_SINCE
...		20	D10	London	D04

Figure 4. Semi-temporalized Table Schema using SINCE

It should be pointed out that S_SINCE is still considered to be "semi-temporalized" since it contains no concept of historical data. For example, if supplier S1's status changes on D12 to 30, the STATUS value of 20 is replaced with 30 and the STATUS_SINCE value of D10 is replaced with D12. The data fact that supplier S1 ever had a status of 20 is now lost.

To overcome the issues of tracking historical data, consider the S_DURING relvar. Given the tuple in *Figure 3*, consider a scenario where supplier S1's status changes to 30 on day 7. The obvious way to account for this change in S_DURING is to delete the existing tuple and insert two new ones as follows:

S_DURING

S#	SNAME	STATUS	CITY	DURING
S1	Smith	20	London	[D04:D06]
S1	Smith	30	London	[D07:D07]

Figure 5. Semi-temporalized Table Schema using DURING

The problem that exists with this solution is that relvar S_DURING timestamps too much information (supplier is under contract, supplier has a name, supplier has a status, and supplier has a city). To resolve this problem, we must vertically decompose the relvar into a series of four separate relvars, each with its own timestamp attribute *(see Figure 6)*.

S_DURING

S#	DURING
S1	[D04:D07]

S_NAME_DURING

S#	SNAME	DURING
S1	Smith	[D04:D07]

S_STATUS_DURING

S#	STATUS	DURING
S1	20	[D04:D06]
S1	30	[D07:D07]

S_CITY_DURING

S#	CITY	DURING
S1	London	[D04:D07]

Figure 6. Fully-temporalized 6NF-compliant Table Schema

Adding transaction time columns CREATED and UPDATED to indicate when a tuple was first created and last updated, respectively, results in a fully-temporalized set of 6NF-compliant relvars capable of tracking historical data *(see Figure 7)*.

S_DURING

S#	DURING	CREATED	UPDATED
S1	[D04:D07]	D04	D07

S_NAME_DURING

S#	SNAME	DURING	CREATED	UPDATED
S1	Smith	[D04:D07]	D04	D07

S_STATUS_DURING

S#	STATUS	DURING	CREATED	UPDATED
S1	20	[D04:D06]	D04	D06
S1	30	[D07:D07]	D07	D07

S_CITY_DURING

S#	CITY	DURING	CREATED	UPDATED
S1	London	[D04:D07]	D04	D04

Figure 7. Fully-temporalized 6NF-compliant Table Schema using Valid Time and Transaction Time

This vertical decomposition of relvar S_DURING is similar to a classic normalization process and leads us to the introduction by Date of the Sixth Normal Form (6NF) with this formal definition:

> "A relvar (table) *R* is in 6NF if and only if *R* satisfies no nontrivial join dependencies at all, in which case *R* is said to be irreducible." (Date et al., 2002)

Before adding historical data, relvar S_DURING (as seen in *Figure 3*) is 5NF-compliant since the only nontrivial join dependencies it satisfies are ones that are implied by its sole candidate key [S#, DURING]. After adding historical data, however, S_DURING is no longer in 5NF form since the tuple [S#, SNAME, CITY] is redundant over candidate key [S#, DURING]. Therefore, S_DURING can benefit from further decomposition into the set of 6NF projections as shown in *Figure 6*. The result is a set of relvars with irreducible elements, each containing its own interval temporal attribute and each tied to the surrogate key S#.

By using conceptual models, 6NF datasets such as those shown in *Figure 7* can be easily visualized and integrity constraints can be clearly defined. Furthermore, XML documents and schemas can be created by database analysts and designers to produce a method for exchanging complex 6NF-compliant data among disparate systems in a structured format.

RELATIONSHIP TO DATA WAREHOUSING 2.0

Data Warehousing 2.0 (DW 2.0) is a second-generation attempt to define a standard Data Warehouse architecture. One of the advantages introduced in DW 2.0 is its ability to support changes of data over time. The authors of DW 2.0 address this issue by saying, "The answer is that semantically static and semantically temporal data should be physically separate in all database designs." (Inmon, Strauss and Neuschloss, 2008)

Semantic data change occurs when the definition of a data entity changes (i.e., new columns are added to an existing entity). This is different from content data change, where the value of an entity changes (i.e., an amount increases from $100 to $200). Semantically temporal data is likely to undergo semantic data change whereas semantically stable data is unlikely to undergo such change. An example of semantically temporal data is a customer profile, where you may want to track new types of information about a customer over time and therefore add new columns of data to the customer entity. For instance, if a company decides to send out a monthly email newsletter, it may add an email address to its customer profiles. An example of semantically stable data is the details of a customer sale. Once a date of sale, amount of sale, item sold, etc. has been determined, this data is unlikely to change over time. If the data is ever updated, the change is most likely the result of correcting an error in the original recording of data and not due to an actual change in the transaction itself.

The designers of DW 2.0 point out that, in an environment where temporal and stable data are grouped together, a simple change in business requirements usually causes the technology infrastructure to go haywire (Inmon et al., 2008). However, separating temporal and stable data can mitigate such an effect. At the point where business change occurs, a new snapshot of the semantics can simply be created and delimited by a time factor using *to* and *from* dates. Date's temporalization approach comes into play here because historical snapshots must be created in such a way as to decompose data into a series of irreducible 6NF-compliant tables.

By using this approach, semantically stable data is not affected at all and previously existing semantically temporal data also remains unchanged. Capturing these snapshots generates a historical record of data by which an analyst or end-user can easily locate and retrieve information while also providing a technological infrastructure that can withstand change over time (Inmon et al., 2008).

ANCHOR MODELING AS A CONCEPTUAL METHOD

Anchor Modeling is a database modeling technique built on the premise that the environment surrounding a data warehouse is in a constant state of change, and furthermore that a large change on the outside of the model should result in only a small change on the inside of the model (Rönnbäck, Regardt, Bergholtz, Johannesson and Wohed, 2010). The goal of using the anchor modeling technique is to "achieve a highly decomposed implementation that can efficiently handle growth of the data warehouse without having to redo any previous work", mimicking the ideas of isolated semantic temporal data put forth in DW 2.0 architecture (Rönnbäck et al., 2010).

When anchor models are translated to physical database designs, content changes are emulated using standard 6NF principles and the tables in the resulting relational database will be 6NF-compliant. Pieces of data are tied to points in time or intervals of time. Time points are modeled as attributes (i.e., the time a part is shipped), and intervals of time are modeled as a historization of attributes or ties (i.e., times during which a part was in stock). Transaction time elements, such as the time information entered or was updated in the database, are handled by metadata and are not included in the standard anchor modeling constructs.

Anchor Modeling provides a graphical notation for conceptual database modeling similar to Entity-Relationship (ER) modeling, with extensions for temporal data. The model is based on 4 constructs: the Anchor, Attribute, Tie, and Knot *(see Figure 8)*. Anchors model entities and events, Attributes model properties of Anchors, Ties model relationships between Anchors, and Knots model shared properties.

By using a double line, an Attribute or a Tie can show a history of changes for the information they model. In *Figure 8*, the double line around attributes SUADR and COTRM and tie SUPA indicates that a history of changes is kept for supplier addresses, contract terms, and supplier-part relationships. Each of these entities is likely to change in content over time – suppliers may change addresses, contract terms may be updated or renewed, and the company may change which suppliers they order certain parts from.

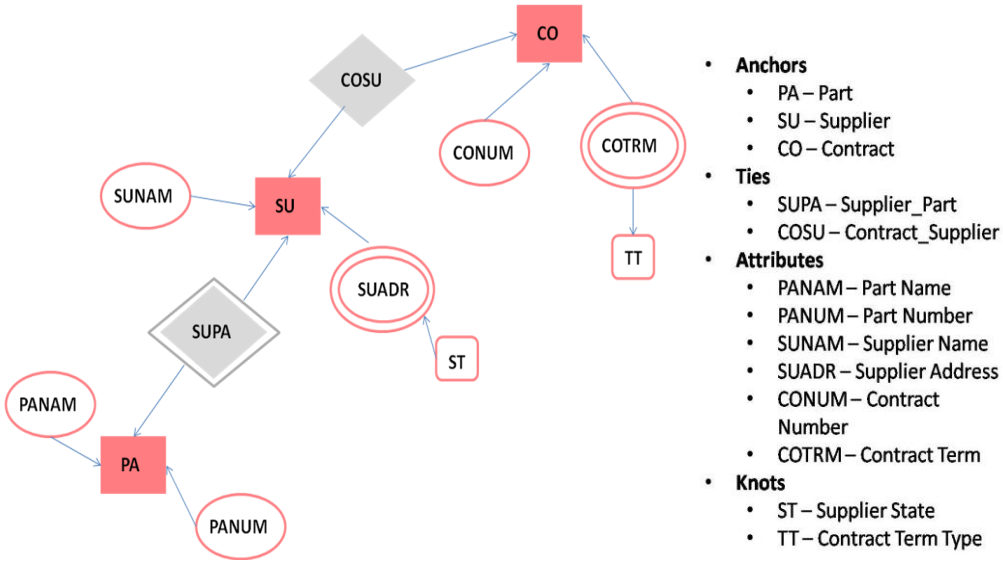

Figure 8. Anchor Modeling – Constructs and a Basic Example

The resulting anchor model can further be translated into a model useful for data warehousing. Using the Dimension Fact Model (DFM) technique put forth by Golfarelli and Rizzi (2009), we can translate the anchor model relationships and entities into dimensions and facts (*see Figure 9*). Fact DELIVERY can be generated which indicates the delivery of a certain part to a customer. Pertinent information can then be built around this fact for dimensions SUPPLIER, PART, and CONTRACT. To account for temporal aspects of the data, the DATE dimension is added. Note that although the DFM can be translated from an anchor model, it is not strictly 6NF compliant. It is simply used here as a traditional conceptual DW modeling technique.

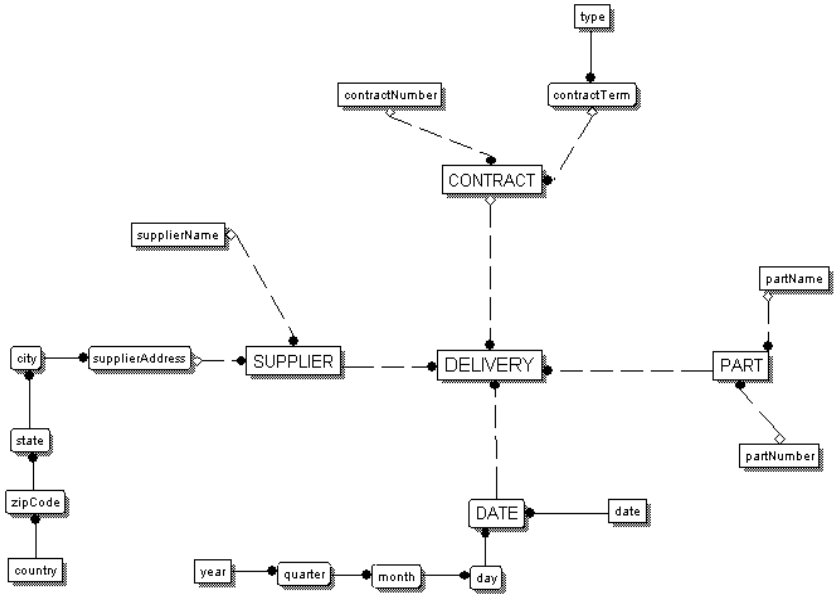

Figure 9. Anchor Data Modeled as Conceptual Dimension Fact Model

CONCLUSION

With the emergence of data warehousing and the emphasis placed on the advantages of using temporal modeling in DW 2.0, the issue of how to store and model changes to data over time has become one of practical importance to business and industry. In this paper, we have explored a generalization of temporal aspects in database and data warehouse design using Sixth Normal Form (6NF) and Anchor Modeling techniques. By using such methods, snapshots of data may be captured which generate a historical record of data by which an analyst or end-user can easily locate and retrieve information while also providing a business with a technological infrastructure that can withstand change over time.

There is now a need to describe and present temporal data to disparate data warehouse systems in a structured format suitable for data exchange across the internet and other networked resources. Since XML has become the preferred means of data exchange, an adequate definition of a standard XML Schema document definition template that describes 6NF data is desired. It is our intention to further this research with the result of producing an original standardized XML Schema mechanism for exchanging such information. We also hope to indicate advantages in application areas benefiting from the use of the proposed formats, as well as limitations.

Researchers have previously described an XML anchor schema definition in the paper *"From Anchor to XML"* (Rönnbäck, Regardt, Bergholtz, Johannesson and Wohed, 2010). We hope to expand upon this work by using metamodels to address a logical level of design in defining a XML schema profile similar to one previously employed to describe Unified Modeling Language (UML) class diagrams (Routlige, Bird, and Goodchild, 2002). Ultimately, the outcome should define a method for using XML schema to communicate with DW 2.0-compliant temporal data warehouses.

REFERENCES

1. Date, C.J., Darwen, H. and Lorentzos, N. (2002) Temporal data and the relational model: A detailed investigation into the application of interval and relation theory to the problem of temporal database management, Morgan Kaufmann Publishers, Amsterdam.
2. Golfarelli, M. and Rizzi, S. (2009) Data warehouse design: Modern principles and methodologies, McGraw Hill, New York.
3. Inmon, W.H., Strauss, D. and Neushloss, G. (2008) DW 2.0: The architecture for the next generation of data warehousing, Morgan Kaufmann Publishers, Amsterdam.
4. Rönnbäck, L., Regardt, O., Bergholtz, M., Johannesson, P. and Wohed, P. (2010) Anchor modeling - agile information modeling in evolving data environments, *Data & Knowledge Engineering*, 1229–1253.
5. Rönnbäck, L., Regardt, O., Bergholtz, M., Johannesson, P. and Wohed, P. (2010) From anchor model to XML, http://www.anchormodeling.com/wp-content/uploads/2010/09/AM-XML.pdf.
6. Routlige, N., Bird, L. and Goodchild, A. (2002) UML and XML Schema, *13th Australian Database Conference*, Melbourne, Australia, 1-10.
7. Snodgrass, R. and Ahn, I. (1986) Temporal databases, *IEEE Computer, 19(9)*, 35-42.