Chapter XVI

Analyses and Evaluation of Responses to Slowly Changing Dimensions in Data Warehouses*

Lars Frank
Copenhagen Business School, Denmark

Christian Frank
Copenhagen Business School, Denmark

Abstract

A Star Schema Data Warehouse looks like a star with a central, so-called fact table, in the middle, surrounded by so-called dimension tables with one-to-many relationships to the central fact table. Dimensions are defined as dynamic or slowly changing if the attributes or relationships of a dimension can be updated. Aggregations of fact data to the level of the related dynamic dimensions might be misleading if the fact data are aggregated without considering the changes of the dimensions. In this chapter, we will first prove that the problems of SCD (Slowly Changing Dimensions) in a data warehouse may be viewed as a special case of the read skew anomaly that may occur when different transactions access and update records without concurrency control. That is, we prove that aggregating fact data to the levels of a dynamic dimension should not make sense. On the other hand, we will also illustrate, by examples, that in some situations it does make sense that fact data is aggregated to the levels of a dynamic dimen-
Analyses and Evaluation of Responses to Slowly Changing Dimensions in Data Warehouses

...tion. That is, it is the semantics of the data that determine whether historical dimension data should be preserved or destroyed. Even worse, we also illustrate that for some applications, we need a history preserving response, while for other applications at the same time need a history destroying response. Kimball et al., (2002), have described three classic solutions/responses to handling the aggregation problems caused by slowly changing dimensions. In this chapter, we will describe and evaluate four more responses of which one are new. This is important because all the responses have very different properties, and it is not possible to select a best solution without knowing the semantics of the data.

INTRODUCTION

A data warehouse is an OLAP (On Line Analytical Processing) database (Codd, 1993 and Codd et al. 1993), where the data is loaded/updated periodically. In other words, a data warehouse is not an OLTP (On Line Transaction Processing) database (e.g., Gray and Reuter, 1993). The data warehouse drill functions (e.g., Kimball et al., 1998) have been developed to accommodate the special needs for aggregating the data stored in the fact table of a data warehouse.

The traditional drill-down functions use the one-to-many relationships of the data warehouse to find more detailed information. If we take accumulated data as an example, the drill-down function will show the more detailed data elements of the accumulated data. The roll-up function can use the one-to-many relationships of the data warehouse to generate an overview of the detailed information. However, the aggregating drill functions may give misleading results as old fact data may be aggregated to dimension levels that have changed since the fact data were created. In this paper, we will evaluate the following 7 different techniques for handling the aggregation problems of slowly changing dimensions (Responses to SCD). The first three responses are the classic techniques described by Kimball et al. (1998).

- **Type 1 response**: Overwrite the dimension record with the new values by which historical information is lost.
- **Type 2 response**: Create a new additional dimension record with the current information.
- **Type 3 response**: Create a “Previous” field in the dimension record to store the immediate previous attribute value.
- **Type 4 response**: Create a new dimension using a dynamic dimension hierarchy
- **Type 5 response**: Store dynamic dimension data as fact data
- **Type 6 response**: Use finer granularity in combination with response 1 or 3
- **Type 7 response**: Store the dynamic dimension data as static facts in another data mart

Table 1 gives an overview of the most important properties of the different responses, and in section 4 the evaluation is described in more details. The most important evaluation criterion for responses to Slowly Changing Dimensions is whether the responses preserve historical information. In the next section, we will illustrate by examples that it is the semantics of the data that determine whether historical information should be preserved. That is, both history preserving and history destroying responses are important. Therefore, it is important to know that only the Type 1 and 3 responses can overwrite historical information in an effective way. The Type 3 and 6 responses do not preserve all the details of historical information but may be a good choice in special situations. Response Types 2, 4, 5, and


Analyses and Evaluation of Responses to Slowly Changing Dimensions in Data Warehouses

7 preserves historical information in an effective way but it is often application dependent to select the best solution as e.g. performance depends on what data is aggregated.

The paper is organized as follows. In this paper, we will first prove that the problems of Slowly Changing Dimensions in a datawarehouse may be viewed as a special case of the read skew anomaly that may occur when different transactions access and update records without concurrency control. The next section will describe the most important concepts in dimensional modelling used in this paper. The following section offers more detailed descriptions and an evaluation of the different responses to Slowly Changing Dimensions. Concluding remarks and suggestions for further research are presented at the end of the paper.

Related Research

Many authors working with data warehouse design have also analyzed the problems of aggregating fact data to the levels of slowly changing dimensions (e.g., Kimball et al., 1998; Anahory and Murray, 1997). The Type 4 technique of splitting a dynamic dimension into two independent dimensions to simplify the dynamicity may also be applied to groups of dimension attributes. Kimball et al. (1998) describe this as Rapidly changing monster dimensions because a group with all the dynamic attributes of a dimension may change rather frequently. However, as long as several dynamic attributes/relationships are grouped together, the technique cannot be used as a response that solves the problem of slowly...

### Table 1.

<table>
<thead>
<tr>
<th>Response type</th>
<th>Evaluation criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Is historical information preserved</td>
</tr>
<tr>
<td>Response 1 where dimension records are overwritten</td>
<td>No</td>
</tr>
<tr>
<td>Response 2 where new versions are created</td>
<td>Yes</td>
</tr>
<tr>
<td>Response 3 where only one historical version is saved</td>
<td>The current version and a single history destroying version are saved</td>
</tr>
<tr>
<td>Response 4 that use the top of a dynamic dimension hierarchy as a new static dimension</td>
<td>Yes</td>
</tr>
<tr>
<td>Response 5 with dimension data as fact data</td>
<td>Yes</td>
</tr>
<tr>
<td>Response 6 that use fine granularity in combination with response 1 or 3</td>
<td>The finer the granularity, the more historical state information is preserved</td>
</tr>
<tr>
<td>Response 7 that stores dynamic dimension data as static facts in another data mart</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Analyses and Evaluation of Responses to Slowly Changing Dimensions in Data Warehouses

changing dimensions. Anyway, if the static attributes/relationships are separated in a dimension by themselves, it may be cheaper to implement a response for the dynamic attributes/relationships. Therefore, the same technique may be used to solve different problems.

Levene and Loizou (2003) have evaluated the snowflake schema and conclude that it has several desirable properties such as new “normalization” possibilities. Therefore, it should be easy to understand as a good model of the real world, and the design should have a minimum of redundancy. We agree that a design, that is a good model of the real world, is useful but “normalization” may conflict with some of the most important responses to slowly changing dimensions, and minimizing redundancy is traditionally not the most important objective in data warehouse design. The relatively new Type 4 response has earlier been described by Frank et al., 2005 and 2006, and some of its properties have been evaluated. Frank, 2008, have described but not evaluated the Type 5 and 6 responses.

SLOWLY CHANGING DIMENSIONS MAY BE VIEWED AS AN ANOMALY

When transactions are executed without isolation, the so called isolation anomalies (Berenson et al., 1995) may occur. Isolation anomalies may totally change the real values of the data read by a transaction, and if the transaction stores its view of data, the wrong data values may be made permanent. Isolation anomalies may be dealt with by using countermeasures (Frank and Zahle, 1998) as the so called long duration transactions (e.g., Gray and Reuter, 1993) can not be managed by using traditional concurrency control. In the same way, the anomalies caused by dynamic dimensions may be dealt with by using the so called responses to SCD. As with countermeasures against isolation anomalies it is the semantics of the data that determine which response type is the best choice. Therefore, the evaluation criteria “Aggregation performance” and “Storage consumption” from Table 1 are only relevant after it is decided whether the response should preserve, destroy and/or preserve historical information in minor detail.

An isolation anomaly is defined by its history that consists of the reads, writes, commits, and aborts executed by the involved transactions. Histories can be written in shorthand notation in the following way (Berenson et al., 1995):

• “w1(x)” means a write by transaction 1 on record x.
• “r2(y)” means a read of record y by transaction 2.
• Transaction 1’s commit and abort are written “c1” and “a1”, respectively.
• Transaction 1’s reading and writing of a set of records satisfying a predicate P is denoted by “r1 (P)” and “w1 (P)”, respectively.

Using histories the classic isolation anomalies may be described as follows (Berenson et al., 1995):

• The lost update anomaly: r1(x)…w2(x)…w1(x)…c1…
• The dirty read anomaly: w1(x)…r2(x)…a1...
• The non-repeatable read anomaly or fuzzy read: r1(x)...w2(x)...c2...r1(x)...
• The phantom anomaly: r1 (P)...w2(y in P)...c2...r1 (P)...
• The read skew anomaly: r1(x)...w2(x)...w2(y)...c2...r1(y)...
• The write skew anomaly: r1(x)...r2(y)...w2(y)...w1(x)...c1...c2...

The read skew anomaly is a situation where a first transaction reads a record. Next, a second transaction updates the record and a related record. The updates are committed. Finally, the first transaction reads the related record. In this
situation, the first transaction may find that the integration of the semantic meaning of the data stored in x and y do not make sense.

The problem of Slowly Changing Dimensions in a data warehouse is not an isolation anomaly as the updating and reading transactions are executed in different time slots. However, in this section we will illustrate that the problems in data warehouses with Slowly Changing Dimensions may be viewed as a special case of the read skew anomaly.

In the following example, x is Fact data while y is Dimension data in a data warehouse. In this situation, the interpretation of the x data may depend on the y data. In the example, the evaluation of x data may change from “acceptable” to “unacceptable” depending on the value of a “size” attribute in y.

**Example 1**

Let us suppose x is a set of records with the monthly sale of a department in a store and y is the corresponding department record. And let us suppose that the department record has an attribute with the number of salesmen in the department.

As Fact data normally is not updated, the Read skew history in a data warehouse is reduced to:

\[ r_l(x) \ldots w_2(y) \ldots c_2 \ldots r_l(y) \ldots \]

From an information point of view, this Read skew history is equivalent to following history as the Fact data x is not changed.

\[ \ldots w_2(y) \ldots c_2 \ldots r_l(x) \ldots r_l(y) \ldots \]

However, this is the history of an OLAP transaction that reads fact data x created before a dimension update in a Slowly Changing Dimension y. Therefore, the Read skew anomaly may be seen as a special case of the problem of Slowly Changing Dimensions.

In the example above, it is important to preserve the relationship between record x and the historical version of record y as the monthly sale normally is very dependent on the number of salesmen in the department. However, it is semantics of the data that determine whether it is right to preserve relationships to historical dimension versions or not. This is illustrated in the following example.

**Example 2**

Let us suppose x is the sale of products in a department store and y is the department records without the attribute with the number of salesmen. In order to see how the departments are performing, it should be possible to aggregate the sale to each department periodically. However, suppose that at a point in time the departments are reorganized in such a way that some products are sold in other departments and vice versa. If the managers now want to see how the departments have performed over time the question can be answered with or without using historical dimension data. If the historical relationship between a sale x and the department y where x was sold is preserved, it may not make any sense to compare the sale of a department over time as the products sold are not the same over time. On the other hand, if historical dimension data is not used all the sales of a product are related to the department from where it is currently sold and the aggregated sales do not correspond to the sales of the real departments. Anyway, in this situation it makes sense to compare the sales of the departments over time, as the aggregated sales correspond to the real sales over time of a static group of products.

In the following example, we will illustrate that the historical versions of a dynamic dimension may only be used by some applications. That is, it is semantics of the data that determine whether historical dimension data should be preserved, destroyed, preserved without always using the
Analyses and Evaluation of Responses to Slowly Changing Dimensions in Data Warehouses

historical information, or preserved in only one historical version.

Example 3

Let us suppose x is the sale of products in a department store and y is the department records including an attribute with the monthly costs of the departments. In order to see the monthly revenue of the departments over time, it should be possible to aggregate the sales to the departments periodically. As the monthly costs per department vary with the number of employees, the department dimension is dynamic, and the historical dimension data should be preserved in order to subtract the monthly costs of a department from the corresponding aggregated monthly sale.

Again, we suppose that at a point in time the departments are reorganized in such a way that some products are sold in other departments and vice versa. In this situation, it may also be of interest to compare the performance of the departments with the performance that would have occurred if the reorganization had not taken place. However, in this situation we only need the product-department relationships before and after the reorganization and not all the history of the departments. As seen in Table 1, response 3 is tailored to deal with such a situation.

TYPES OF DYNAMICITY IN SLOWLY CHANGING DIMENSIONS

Dynamicity in slowly changing dimensions can be grouped after whether it is a dimension attribute or a dimension relationship that is dynamic. Dynamic dimension attributes are normal dimension attributes that may be updated. Aggregations to a dimension with a dynamic dimension attribute may be misleading as fact data created before the change are aggregated to a dimension level with changed properties. Dynamic dimension relationships are dimension relationships that may be updated. Aggregations to a level with a dynamic relationship towards the fact table may be problematic as fact data created before the changes may be aggregated to levels with which they had no relationship when they were created. The reason why we distinguish between dynamic dimension attributes and dynamic dimension relationships is that the aggregation problems with dynamic dimension relationships normally are more serious than the aggregation problems with dynamic dimension attributes. For example, a change in a dimension attribute value may have no or minor correlation with the aggregated fact data, while a change in a dimension relationship always will change the aggregations made through the relationship. A more technical reason for the distinction is that the dynamic attributes are stored in the dimension itself while the dynamic relationships are stored as foreign keys in the related table. Therefore, overwriting a dynamic dimension relationship must be implemented by updating the related table and not by updating the dynamic dimension itself. If the related table is the fact table, then the fact table must be updated even though fact data by definition is static.

Dynamicity in slowly changing dimensions can also be grouped after whether a history preserving response or a more or less history destroying response should be used from a semantic point of view. Normally, it is important to select responses to slowly changing dimensions that preserve historical information as data warehouses are used to analyze changes over time. However, as described in the previous section, it may be important to more or less destroy historical information. In these situations, it is important to use a proper response type that makes the aggregations over time meaningful. It is the semantics of the aggregated data that should be used to decide whether a history preserving or a more or less history destroying response should be used. Therefore, this question must be answered for all dynamic attributes/keys before the responses are chosen. As described in the previous section it
may even be necessary to implement both history preserving and history destroying responses for the same dimension because different applications may have different needs.

A *dimension hierarchy* (e.g., Kimball et al., 1998) is a hierarchy of tables connected through one-to-many relationships towards the fact table. If the tables in a dimension hierarchy are joined together to a single dimension table, we say that the joined dimension has an *internal dimension hierarchy*. In practice, internal dimension hierarchies are often used as they normally improve the speed of executing aggregation at the costs of extra space for storing redundant information. Normally, the fact table of a data warehouse has a *Time dimension hierarchy* that enables the users to aggregate fact data to the level of day, month or year. The time hierarchy may be internal or stored in separate tables as a dimension hierarchy. In the following example, the Time dimension has been designed with an internal dimension hierarchy for performance reasons. This will not produce aggregation problems as the Time dimension including its hierarchies is static. However, the example also illustrates that both internal and external dimension hierarchies with dynamic dimension relationships automatically will have serious aggregation problems.

**Example**

In Figure 1, the central fact table of the snowflake schema has three dimensions and a dimension hierarchy. Therefore, the fact table has attributes for the four corresponding dimension foreign keys. In the figure, the primary key of each table is underlined.

The Product dimension is dynamic as the Product-group of a Product may change. The Salary-group of the dimension hierarchy is dynamic as its relationship to a Salesman may change. The Salary-group dimension is stored in a hierarchy as this may save a lot of storage space if many salesmen are related to a few salary-groups with many attributes to describe the salary contracts. It may be interesting to aggregate the turnover (*Qty*·*Price*) to both the Salesman and Salary-group levels to evaluate each individual salesman as well as groups of salesmen. However, the aggregation to the Salary-group level is without meaning as some salesmen may have changed salary group. For these salesmen, the turnover that should have

*Figure 1.*
Analyses and Evaluation of Responses to Slowly Changing Dimensions in Data Warehouses

been aggregated to the previous salary group is wrongly aggregated to the new salary group. Therefore, a response type that preserves historical information should be used.

In Figure 1, the Product dimension has an internal dimension hierarchy as each Product has a relationship to one Product-group and each Product-group may have relationships to many Products. This relationship is dynamic, and, therefore, aggregation to the Product-group level in the internal Product hierarchy may be wrong. However, the Product-group is a dynamic classification criterion as one or more products may change their Product to Product-group relationship by a management decision. However, the same static Product-group definition should be used over time in aggregations even though the groups may have changed over time. Otherwise, it does not make sense from a semantic point of view to compare aggregations to the Product-group level over time. Therefore, a response type that destroys historical information should be used when data is aggregated from the Product level to the Product-group level.

The Time dimension is implemented with an internal hierarchy, too. This will give no aggregation problems as both the attributes of the dimension and the internal relationships between day, month, and year are static. However, in the real world, the Time dimension may be dynamic as some event may change a weekday to a holiday. Anyway, this may be handled as an error in the initial load of the Time dimension, and therefore the history destroying Type 1 response to slowly changing dimensions can be recommended.

RESPONSES TO SLOWLY CHANGING DIMENSIONS

In this section, we will describe and evaluate the different responses to slowly changing dimensions in more details. First, we will describe the fourth response. We describe the Type 4 response first to illustrate that it is important to examine whether it is possible to change dynamic information to static information by changing the design of the datawarehouse. The Type 7 response will also create new entities in the datawarehouse. Next, we will describe and evaluate the classic responses originally described by Kimball. Finally, we will describe and evaluate three new responses.

The Type 4 Response: Create a New Dimension Using a Dynamic Dimension Hierarchy

The aggregation problems in internal and external dynamic dimension hierarchies may disappear if the data corresponding to a dynamic entity in the hierarchy is removed and stored in a separate independent dimension directly related to the fact table. By doing so the dynamic data is either changed to static dimensions or changed to dynamic dimensions with reduced aggregation problems. Response 4 may be used recursively as different layers of dynamic entities may occur in both internal and external dimension hierarchies. The Type 4 response should only be used to transform dimensions with dynamic relationships to the fact table where it is important to preserve historical information.

Example 5

In Figure 2, the dynamic Salesmen dimension hierarchy from example 1 has been divided into two independent dimensions corresponding to the entities Salesman and Salary-group. These dimensions still have dynamic attributes, but they do not have dynamic hierarchies and therefore response 4 cannot be used any more. However, the transformation has solved all the aggregation problems as the turnover can be aggregated to the Salary-group level in a meaningful way. Anyway, the dimensions are still dynamic as historical information may be lost when the dimension attributes are updated. Therefore, it may still be
useful to use one of the other responses to slowly changing dimensions.

The implementation of a new salary-group dimension uses more storage space than the solution with a dimension hierarchy as the foreign key now is stored in the fact table and not in the Salesman dimension.

The Type 4 solution also gives a worse performance than the solution with a dimension hierarchy if data from both Salesman table and the Salary-group table are to be used in query. However, the Type 4 solution is best when only the Salary-group information is needed in a query.

The dynamic internal dimension hierarchy of the Products dimension from example 1 has also been divided into two independent dimensions corresponding to the entities Product and Product-group. However, in this case the Type 4 response is used in a wrong way from a semantic point of view, as it normally does not make sense to aggregate the turnover to Product-groups where the Products per Product-group may change over time. In order to make it possible to compare the turnover per Product-group over time, a static set of Products per Product-group should be used even though the Products per Product-group are dynamic. For example, if it turns out that some Garden products are not waterproof, management may decide to group them under Furniture products in the future. However, to make it possible to compare the turnover per Product group over time, the non-waterproof furniture should also be grouped as Furniture products in past fact records. This is not possible with the type 4 response as it preserves historical information. A correct solution to the problem is described under the Type 1 and 3 responses where historical information is not preserved, and, therefore, the turnover per Product group per time period will be corrected automatically.

The Type 1 Response: Overwrite with the New Values

Inc case of a dynamic dimension attribute, this is the simplest response to use, and the response
Analyses and Evaluation of Responses to Slowly Changing Dimensions in Data Warehouses

functions correctly if a response type that destroys historical information is needed or if the historical information of the attribute is not important from a semantic point of view. Normally, data is not important from “a semantic point of view” if the value of the attribute is not correlated to the result of the aggregations. The problem is that the fact data created before the change of the dynamic attribute are aggregated to a level where the attribute may have changed in a misleading way with respect to the correlated aggregated data. However, the Type 1 response is also the most inflexible solution, and the decision to use this response is normally irreversible as the response destroys the historical information. The following example illustrates situations where the Type 1 response may be useful.

Example 6

In Figure 2, the name attributes like Product-name, Salesman-name, and Salary-name may be changed, but normally these changes are not important from a semantic point of view. In such situations, the information can be overwritten without problems. In Figure 1, the Type 1 response may also be useful when the turnover is aggregated to the Product-group level as historical information must not be used when the turnover is aggregated to the current definition of the Product-groups.

The following examples will illustrate that the Type 1 response may give misleading aggregations if either the value of a dynamic attribute is correlated to the result of the aggregations or if a response type that preserves historical information is needed.

Example 7

In Figure 2 the Salary attribute in the Salary-Group dimension is dynamic. One may expect that Salesmen with a high salary will sell more than Salesmen with a low salary, i.e. the Salary attribute is probably correlated to the result of the aggregations. Therefore, the Type 1 response that destroys historical information, may give misleading aggregations if the user wants to analyze the aggregated turnover as a function of the Salary attribute. In Figure 1, the Type 1 response may also give misleading results when the turnover is aggregated to the Salary-group level as historical information must be used if the turnover has to be aggregated to the correct Salary-groups.

Example 8

The following star schema illustrates a data warehouse in a bank where the fact data is calculated and loaded periodically each month. For a customer that changes the relationship of an account to a branch office, half of the aggregated interest and costs for the month in question will in average be aggregated to a wrong branch office, i.e. aggregations to the branch office level may be misleading. However, if the granularity of the fact table is reduced to a day there will be no aggregation errors as banks normally calculate interest/costs on a daily basis.

The Type 2 Response: Create a New Additional Record

This response creates a new dimension record version with the changed data each time the dimension record is changed. By using the historical dimension versions, it is always possible to aggregate fact data correctly through a dynamic dimension hierarchy. However, this will increase the complexity of the datawarehouse, and the
Analyses and Evaluation of Responses to Slowly Changing Dimensions in Data Warehouses

knowledge workers making data warehouse applications must know that different versions of dimension records are stored. For example, count statistics must always control that only “living” versions are counted. The following examples illustrate that the Type 2 response is useful both as a response to dynamic dimension attributes and to dynamic dimension relationships.

**Example 9**

In Figure 2, the Salary attribute in the Salary-group dimension may change. As we believe the Salary attribute is correlated to the result of the aggregations, the Type 2 response can give a better description of the relationship between the aggregated sale and the salary of the salesmen for each time period than the Type 1 response.

**Example 10**

In Figure 3, it is possible to use the Type 2 response to changes in the accounts relationships to the branch offices. However, as the changed relationship is stored in the fact table, a new version of the fact record must be created to keep track of the changed relationship. In this situation, it is possible to divide the fact data according to the time the accounts have been related to each branch office. With this solution, special care must be taken with statistics where the number of accounts per branch office is aggregated.

The Type 3 Response: Create a “Previous” Field for the Updated Attribute

With this solution, it is not possible to keep track of all changes as only the current values and a single historical version are stored. Therefore, the Type 3 response is grouped together with the Type 1 response as a response that destroys historical information. However, as described earlier the limited historical information stored by this solution makes it an ideal choice in an analysis of a reorganization or structural change where the situation before the structural change is compared with the situation after the structural change.

**Example 11**

Dimension tables are often used to group the fact data in business-defined groups like Product-groups, Departments and Regions, where management may decide to change the groupings overnight. In contrast to the previous responses, we are now interested in solutions that will allow us to aggregate all the fact data to either the previous or the new groupings. In this situation, we will recommend using the Type 3 response to store both the new and the previous foreign keys as illustrated in the Figure 4.

In the snowflake schema above, it is possible to aggregate the fact data to both the new and the
Analyses and Evaluation of Responses to Slowly Changing Dimensions in Data Warehouses

previous product groups. Normally, it should be enough to store two groupings of foreign keys, because the only reason for storing previous groupings is to evaluate whether it was wise to make the latest change in the grouping. The two dimension hierarchies illustrated in the Figure may be joined together with the Product dimension to form a product dimension with two dynamic internal hierarchies. As we have seen in Figure 2, these may be removed by using the Type 4 response. However, the aggregations to the product group level in different time periods cannot be compared if a product is aggregated to different product groups depending on when the product group relationship was changed. Therefore, the Type 1 and 3 responses are the only correct solutions in this situation.

The Type 5 Response: Store Dynamic Dimension Data as Fact Data

Any dynamic dimension attribute may be stored as a static fact attribute. We will call this solution the Type 5 response to slowly changing dimensions. The Type 5 response is very costly in storage space compared with the Type 2 response as the data of the attribute is redundant except for the first time a new version is stored. However, if the attribute is used in aggregations to other dimensions than its own, it may improve performance considerably as illustrated in the following example.

Example 12

In Figure 1 the Price of a product should be stored as a dynamic dimension attribute in the Product dimension to ensure that there always is an up-to-date version of the Price. However, the price is stored as a static redundant fact attribute in Figure 1. This may be done for two different reasons. The first reason relates to performance as it is possible to aggregate the turnover to any aggregation level without reading a Type 2 version of the Product dimension, where the correct version of the Price can be stored. The other reason for storing the Price as a fact attribute is that the salesman who uses this solution is able to select a special price for a customer.

The Type 6 Response: Use Fine Granularity in Response 1 or 3

The idea of this response is to use a fine granularity in the fact table in combination with the Type 1 or 3 responses. This can diminish or totally eliminate aggregation errors caused by the Type 1 or 3 responses in situations where state fact data only can be aggregated to aggregation levels, which they were related to at the end of the state periods. By diminishing the granularity (i.e. the state period and the accumulated state fact data), the number of the fact data aggregated to wrong dimensions is diminished correspondingly. The example of Figure 3 illustrates how it is possible to eliminate errors in an account data warehouse by changing the granularity from monthly state data to daily state data.

The Type 7 Response: Store the Dynamic Dimension Data as Static Facts in another Mart

In response type 5 the dynamic dimension attribute was stored as a static measure in the fact table. This is always possible but it may be a short-term solution if the dynamic attribute from a business and normalization point of view origins from another business entity.

Example 13

Let us suppose a fact table stores the sale of products in a department store as described in the examples of the second section. In this example the department records are without the attribute with the number of salesmen as well as the attribute with the monthly costs of the departments. That
is, the department dimension may be static. The missing dynamic attribute values of the original department dimension of the previous examples may in this situation be calculated from new fact tables. As examples, the number of employees per department may be aggregated from an employee data mart, and the monthly cost per department may be calculated by drilling across from the employee data mart with salary information to a new procurement data mart and further to a new promotion data mart etc. This solution will be the most flexible solution as historical data may be stored in most details. However, it is normally costly and time consuming and therefore unrealistic to develop this solution for all dynamic dimension attributes.

CONCLUSION

In this paper, we have illustrated that the problems of SCD (Slowly Changing Dimensions) in a data warehouse may be viewed as a special case of the read skew anomaly that may occur when different transactions access and update records without concurrency control. That is, we have proved that the aggregating fact data to the levels of a dynamic dimension may not make any sense. In this situation, historical information about the changes in dimension data should be preserved. On the other hand, we also illustrate that in some situations, it is necessary to overwrite historical information in order to make aggregations meaningful. The reason for this is that data aggregated periodically across time periods with a structural chance, may not give any meaning, if the structural reorganization change the groupings to which data are aggregated.

However, historical information may also be overwritten if there is no correlation between the values of a dynamic attribute and the result of the aggregations. That is, it is the semantics of the data that determine whether historical information should be preserved, destroyed, preserved without always using the historical information, or preserved in minor details. Therefore, it is important to know that only the Type 1 and 3 responses can overwrite historical information in an effective way. The Type 3 response is only the best choice in situations where exactly two versions should be saved. This is illustrated with a reorganisation example where only the latest versions before and after the reorganization are saved. Response Types 2, 4, 5, and 7 all preserve historical information in an effective way but it is often application dependent to select the best solution as e.g. performance depends on what data is aggregated.

Both the classic and four new responses to Slowly Changing Dimensions have been described and evaluated in this paper. Table 1 of the paper gives an overview of the most important properties of the different solutions.

REFERENCES


Analyses and Evaluation of Responses to Slowly Changing Dimensions in Data Warehouses


ENDNOTE

This research is partly funded by the ‘Danish Foundation of Advanced Technology Research’ as part of the ‘Third Generation Enterprise Resource Planning Systems’ (3gERP) project, a collaborative project between Department of Informatics at Copenhagen Business School, Department of Computer Science at Copenhagen University, and Microsoft Business Systems.