Slowly Changing Dimensions

Up to this point we have pretended that each dimension is logically independent from all the other dimensions. In particular, dimensions have been assumed to be independent of time. Unfortunately, this is not the case in the real world. While dimension table attributes are relatively static, they are not fixed forever. Dimension attributes change, albeit rather slowly, over time. Dimensional designers must engage business representatives proactively to help determine the appropriate change-handling strategy. We can't simply jump to the conclusion that the business doesn't care about dimension changes just because its representatives didn't mention it during the requirements process. While we're assuming that accurate change tracking is unnecessary, business users may be assuming that the data warehouse will allow them to see the impact of each and every dimension change. Even though we may not want to hear that change tracking is a must-have because we are not looking for any additional development work, it is obviously better to receive the message sooner rather than later.

When we need to track change, it is unacceptable to put everything into the fact table or make every dimension time-dependent to deal with these changes. We would quickly talk ourselves back into a full-blown normalized structure with the consequential loss of understandability and query performance. Instead, we take advantage of the fact that most dimensions are nearly constant over time. We can preserve the independent dimensional structure with only relatively minor adjustments to contend with the changes. We refer to these nearly constant dimensions as *slowly changing dimensions.* Since Ralph Kimball first introduced the notion of slowly changing dimensions in 1994, some IT professionals—in a never-ending quest to speak in acronymese—have termed them *SCDs*.

For each attribute in our dimension tables, we must specify a strategy to handle change. In other words, when an attribute value changes in the operational world, how will we respond to the change in our dimensional models? In the following section we'll describe three basic techniques for dealing with attribute changes, along with a couple hybrid approaches. You may decide that you need to employ a combination of these techniques within a single dimension table.

Type 1: Overwrite the Value

With the type 1 response, we merely overwrite the old attribute value in the dimension row, replacing it with the current value. In so doing, the attribute always reflects the most recent assignment.

Let's assume that we work for an electronics retailer. The procurement buyers are aligned along the same departmental lines as the store, so the products being acquired roll up into departments. One of the procured products is IntelliKidz software. The existing row in the product dimension table for IntelliKidz looks like the following:

Product Key	Product Description	Department	SKU Number (Natural Key)
12345	IntelliKidz 1.0	Education	ABC922-Z

Of course, there would be numerous additional descriptive attributes in the product dimension, but we've abbreviated the column listing given our page space constraints. As we discussed earlier, a surrogate product key is the primary key of the table rather than just relying on the stock keeping unit (SKU) number. Although we have demoted the SKU number to being an ordinary product attribute, it still has a special significance because it remains the natural key. Unlike all other product attributes, the natural key must remain inviolate. Throughout the discussion of all three SCD types, we assume that the natural key of a dimension remains constant.

Suppose that a new merchandising person decides that IntelliKidz should be moved from the Education software department to the Strategy department on January 15, 2002, in an effort to boost sales. With the type 1 response, we'd simply update the existing row in the dimension table with the new department description. The updated row would look like the following:

Product Key	Product Description	Department	SKU Number (Natural Key)
12345	IntelliKidz 1.0	Strategy	ABC922-Z

In this case, no dimension or fact table keys were modified when IntelliKidz's department changed. The rows in the fact table still reference product key 12345, regardless of IntelliKidz's departmental location. When sales take off following the move to the Strategy department, we have no information to explain the performance improvement because the historical and more recently loaded data both appear as if IntelliKidz has always rolled up into Strategy.

The type 1 response is the simplest approach to dealing with dimension attribute changes. The advantage of type 1 is that it is fast and easy. In the dimension table, we merely overwrite the preexisting value with the current assignment. The fact table is left untouched. The problem with a type 1 response is that we lose all history of attribute changes. Since overwriting obliterates historical attribute values, we're left solely with the attribute values as they exist today. A type 1 response obviously is appropriate if the attribute change is a correction. It also may be appropriate if there is no value in keeping the old description. We need input from the business to determine the value of retaining the old attribute value; we shouldn't make this determination on our own in an IT vacuum. Too often project teams use a type 1 response as the default response for dealing with slowly changing dimensions and end up totally missing the mark if the business needs to track historical changes accurately.

The type 1 response is easy to implement, but it does not maintain any history of prior attribute values.

Before we leave the topic of type 1 changes, there's one more easily overlooked catch that you should be aware of. When we used a type 1 response to deal with the relocation of IntelliKidz, any preexisting aggregations based on the department value will need to be rebuilt. The aggregated data must continue to tie to the detailed atomic data, where it now appears that IntelliKidz has always rolled up into the Strategy department.

Type 2: Add a Dimension Row

We made the claim earlier in this book that one of the primary goals of the data warehouse was to represent prior history correctly. A type 2 response is the predominant technique for supporting this requirement when it comes to slowly changing dimensions.

Using the type 2 approach, when IntelliKidz's department changed, we issue a new product dimension row for IntelliKidz to reflect the new department attribute value. We then would have two product dimension rows for IntelliKidz, such as the following:

Product Key	Product Description	Department	SKU Number (Natural Key)
12345	IntelliKidz 1.0	Education	ABC922-Z
25984	IntelliKidz 1.0	Strategy	ABC922-Z

Now we see why the product dimension key can't be the SKU number natural key. We need two different product surrogate keys for the same SKU or physical barcode. Each of the separate surrogate keys identifies a unique product attribute profile that was true for a span of time. With type 2 changes, the fact table is again untouched. We don't go back to the historical fact table rows to modify the product key. In the fact table, rows for IntelliKidz prior to January 15, 2002, would reference product key 12345 when the product rolled into the Education department. After January 15, the IntelliKidz fact rows would have product key 25984 to reflect the move to the Strategy department until we are forced to make another type 2 change. This is what we mean when we say that type 2 responses perfectly partition or segment history to account for the change.

If we constrain only on the department attribute, then we very precisely differentiate between the two product profiles. If we constrain only on the product description, that is, IntelliKidz 1.0, then the query automatically will fetch both IntelliKidz product dimension rows and automatically join to the fact table for the complete product history. If we need to count the number of products correctly, then we would just use the SKU natural key attribute as the basis of the distinct count rather than the surrogate key. The natural key field becomes a kind of reliable glue that holds the separate type 2 records for a single product together. Alternatively, a most recent row indicator might be another useful dimension attribute to allow users to quickly constrain their query to only the current profiles.

The type 2 response is the primary technique for accurately tracking slowly changing dimension attributes. It is extremely powerful because the new dimension row automatically partitions history in the fact table.

It certainly would feel natural to include an effective date stamp on a dimension row with type 2 changes. The date stamp would refer to the moment when the attribute values in the row become valid or invalid in the case of expiration dates. Effective and expiration date attributes are necessary in the staging area because we'd need to know which surrogate key is valid when we're loading historical fact records. In the dimension table, these date stamps are helpful extras that are not required for the basic partitioning of history. If you use these extra date stamps, just remember that there is no need to constrain on the effective date in the dimension table in order to get the right answer. This is often a point of confusion in the design and use of type 2 slowly changing dimensions.

While including effective and expiration date attributes may feel comfortable to database designers, we should be aware that the effective date on the dimension table may have little to do with the dates in the fact table. Attempting to constrain on the dimension row effective date actually may yield an incorrect result. Perhaps version 2.0 of IntelliKidz software will be released on May 1, 2002. A new operational SKU code (and corresponding data warehouse surrogate key) would be created for the new product. This isn't a type 2 change because the product is a completely new physical entity. However, if we look at a fact table for the retailer, we don't see such an abrupt partitioning of history. The old version 1.0 of the software inevitably will continue to be sold in stores after May 1, 2002, until the existing inventory is depleted. The new version 2.0 will appear on the shelves on May 1 and gradually will supersede the old version. There will be a transition period where both versions of the software will move past the cash registers in any given store. Of course, the product overlap period will vary from store to store. The cash registers will recognize both operational SKU codes and have no difficulty handling the sale of either version. If we had an effective date on the product dimension row, we wouldn't dare constrain on this date to partition sales because the date has no relevance. Even worse, using such a constraint may even give us the wrong answer.

Nevertheless, the effective/expiration date stamps in the dimension may be useful for more advanced analysis. The dates support very precise time slicing of the dimension by itself. The row effective date is the first date the descriptive profile is valid. The row expiration date would be one day less than the row effective date for the next assignment, or the date the product was retired from the catalog. We could determine what the product catalog looked like as of December 31, 2001, by constraining a product table query to retrieve all rows where the row effective date to less than or equal to December 31, 2001, and the row expiration date to greater than or equal to December 31, 2001. We'll further discuss opportunities to leverage effective and expiration dates when we delve into the human resources schema in Chapter 8.

The type 2 response is the workhorse technique to support analysis using historically accurate attributes. This response perfectly segments fact table history because prechange fact rows use the prechange surrogate key. Another type 2 advantage is that we can gracefully track as many dimension changes as required. Unlike the type 1 approach, there is no need to revisit preexisting aggregation tables when using the type 2 approach.

Of course, the type 2 response to slowly changing dimensions requires the use of surrogate keys, but you're already using them anyhow, right? It is not sufficient to use the underlying operational key with two or three version digits because you'll be vulnerable to the entire list of potential operational key issues discussed in Chapter 2. Likewise, it is certainly inadvisable to append an effective date to the otherwise primary key of the dimension table to uniquely identify each version. With the type 2 response, we create a new dimension row with a new single-column primary key to uniquely identify the new product profile. This single-column primary key establishes the linkage between the fact and dimension tables for a given set of product characteristics. There's no need to create a confusing secondary join based on effective or expiration dates, as we have pointed out.

We recognize that some of you may be concerned about the administration of surrogate keys to support type 2 changes. In Chapter 16 we'll discuss a workflow for managing surrogate keys while accommodating type 2 changes in more detail. In the meantime, we want to put your mind somewhat at ease about the administrative burden. When we're staging dimension tables, we're often handed a complete copy of the latest, greatest source data. It would be wonderful if only the changes since the last extract, or deltas, were delivered to the staging area, but more typically, the staging application has to find the changed dimensions. A field-by-field comparison of each dimension row to identify the changes between yesterday's and today's versions would be extremely laborious, especially if we have 100 attributes in a several-million-row dimension table. Rather than checking each field to see if something has changed, we instead compute a checksum for the entire row all at once. A cyclic redundancy checksum (CRC) algorithm helps us quickly recognize that a wide, messy row has changed without looking at each of its constituent fields. In our staging area we calculate the checksum for each row in a dimension table and add it to the row as an administrative column. At the next data load, we compute the CRCs on the incoming records to compare with the prior CRCs. If the CRCs match, all the attributes on both rows are identical; there's no need to check every field. Obviously, any new rows would trigger the creation of a new product dimension row. Finally, when we encounter a changed CRC, then we'll need to deal with the change based on our dimension-change strategy. If we're using a type 2 response for all the attributes, then we'd just create another new row. If we're using a combination of techniques, then we'd have to look at the fields in more detail to determine the appropriate action.

Since the type 2 technique spawns new dimension rows, one downside of this approach is accelerated dimension table growth. Hence it may be an inappropriate technique for dimension tables that already exceed a million rows. We'll discuss an alternative approach for handling change in large, multimillion-row dimension tables when we explore the customer dimension in Chapter 6.

Type 3: Add a Dimension Column

While the type 2 response partitions history, it does not allow us to associate the new attribute value with old fact history or vice versa. With the type 2 response, when we constrain on Department = Strategy, we will not see IntelliKidz facts from before January 15, 2002. In most cases, this is exactly what we want.

However, sometimes we want the ability to see fact data as if the change never occurred. This happens most frequently with sales force reorganizations. District boundaries have been redrawn, but some users still want the ability to see today's sales in terms of yesterday's district lines just to see how they would have done under the old organizational structure. For a few transitional months, there may be a desire to track history in terms of the new district names and conversely to track new data in terms of old district names. A type 2 response won't support this requirement, but the type 3 response comes to the rescue.

In our software example, let's assume that there is a legitimate business need to track both the old and new values of the department attribute both forward and backward for a period of time around the change. With a type 3 response, we do not issue a new dimension row, but rather we add a new column to capture the attribute change. In the case of IntelliKidz, we alter the product dimension table to add a prior department attribute. We populate this new column with the existing department value (Education). We then treat the department attribute as a type 1 response, where we overwrite to reflect the current value (Strategy). All existing reports and queries switch over to the new department description immediately, but we can still report on the old department value by querying on the prior department attribute.

Product Key	Product Description	Department	Prior Department	SKU Number (Natural Key)
12345	IntelliKidz 1.0	Strategy	Education	ABC922-Z

Type 3 is appropriate when there's a strong need to support two views of the world simultaneously. Some designers call this an *alternate reality*. This often occurs when the change or redefinition is soft or when the attribute is a human-applied label rather than a physical characteristic. Although the change has occurred, it is still logically possible to act as if it has not. The type 3 response is distinguished from the type 2 response because both the current and prior descriptions can be regarded as true at the same time. In the case of a sales reorganization, management may want the ability to overlap and analyze results using either map of the sales organization for a period of time. Another common variation occurs when your users want to see the current value in addition to retaining the original attribute value rather than the prior.

The type 3 response is used rather infrequently. Don't be fooled into thinking that the higher type number associated with the type 3 response indicates that it is the preferred approach. The techniques have not been presented in good, better, and best practice sequence. There is a time and place where each of them is the most appropriate response.

The type 3 slowly changing dimension technique allows us to see new and historical fact data by either the new or prior attribute values.