Late Arrivals

by Sudheer Sharma - Monday, November 29, 2010

http://dwhnotes.com/data-warehousing/late-arrivals

What do you think of this post?

Awesome (0) Interesting (1) Useful (1)

Ralph Kimbal's explaination about late arrivals

http://www.rkimball.com/html/designtipsPDF/KimballDT57EarlyArriving.pdf

What do you do when you receive late-arriving data that should have been loaded into the data warehouse weeks or months ago? There are two interesting cases of this scenario that I'll discuss separately.

Late-Arriving Fact Records

Using my customer purchase scenario, suppose you receive a purchase record today that is several months old. In most operational data warehouses, you are willing to insert this late-arriving record into its correct historical position, even though your sales summary for this prior month will now change. But you must carefully choose the old contemporary dimension records that apply to this purchase. If you have been timestamping the dimension records in your Type 2 SCDs, then your processing will involve the following steps:

1. For each dimension, find the corresponding dimension record whose timestamp is the latest timestamp less than or equal to the date of the purchase.

2. Using the surrogate keys found in each of the dimension records from step 1, replace the natural keys of the late-arriving fact record with the surrogate keys.

3. Insert the late-arriving fact record into the correct physical partition of the database containing the other fact records from the time of the late-arriving purchase.

Let me make a few subtle points here.

First, I've assumed that dimension records only contain a simple timestamp that indicates when that particular detailed description became valid. You could have assumed that the dimension record contained two timestamps, indicating the beginning and end of the period of validity of the detailed description. This would seem to make the search for the correct dimension records simpler, because you might use the BETWEEN clause in SQL to constrain the purchase_date BETWEEN begin-date AND end_date. While this is possible, life turns out to be more complicated. Believe it or not, this fragment of

SQL is invalid! Standard SQL syntax requires value BETWEEN field1 AND field2, and not field BETWEEN value1 AND value2. So you must either rewrite the SQL with two open-ended comparisons, or use nonstandard SQL extensions that support this use of BETWEEN. Another objection to the twin timestamps is that updating a dimension is more complicated because you need to be careful to have an unbroken chain of non-overlapping begin- and end-dates for each customer, product, and store. You avoid these problems with the single-timestamp approach, even though it makes certain lookup queries less efficient.

A second subtle point goes back to my assumption that you have an “operational data warehouse” that is willing to insert these late-arriving records into old months. If your data warehouse has to “tie to the books,” then you can’t change an old monthly sales total, even if the old sales total was incorrect. Now you have a tricky situation in which the date dimension on the sales record is for a “booking date,” which may be today, but the other customer, store, and product dimensions should nevertheless refer to the old descriptions in the way I’ve described previously. If you are in this situation, you should have a discussion with your finance department to make sure that they understand what you are doing. An interesting compromise I have used in this situation is to carry two date dimensions on purchase records. One refers to the actual purchase date, and the other refers to the booking date.

The third subtle point is that you must insert the late-arriving purchase record into the correct physical partition of the database containing its contemporary “brothers and sisters.” This way, when you move a physical partition from one form of storage to another or you perform a backup or restore operation, you will be affecting all the purchase records from a particular span of time. In most cases this is what you want to do. You can guarantee that all fact records in a time span occupy the same physical partition if you declare the physical partitioning of the fact table to be based on the date dimension. Because you should be using surrogate keys for the date dimension, this is the one case when the surrogate keys of a dimension should be assigned in a particular logical order.

To make this point more clearly, let’s isolate the following as a design tip: You should assign the surrogate keys in a time dimension in sequence so that both the time dimension table and any associated fact table will be sorted in the correct date order when you sort them on the surrogate key.

Late-Arriving Dimension Records

A late-arriving dimension record presents an entirely different set of issues that in some ways are more complex than those in a late-arriving fact record. Suppose you have a fictitious product called Zippy Cola. In the product dimension record for Zippy Cola 12 oz. cans, there is a FORMULATION field that has always contained the value “Formula A.” You have a number of records for Zippy Cola 12 oz. cans because this is a slowly changing dimension and other attributes like the package type and the subcategory for Zippy Cola 12 oz. cans have changed over the past year or two.

Let’s say today you are notified that on July 15, 1999, the FORMULATION of Zippy Cola 12 oz. cans actually was changed to “Formula B” and has been Formula B ever since. To add this new information to the data warehouse requires the following steps:

1. Insert a fresh new record for Zippy Cola 12 oz. cans into the Product dimension with the FORMULATION field set to “Formula B” and the effective date set to July 15, 1999. You must create a

new surrogate key for this record.

2. Scan forward in the Product dimension table from July 15, 1999, finding any other records for Zippy Cola 12 oz. cans, and destructively overwrite the FORMULATION field to “Formula B” in all such records.

3. Find all fact records involving Zippy Cola 12 oz. cans from July 15, 1999, to the first next change for that product in the dimension after July 15, 1999, and destructively change the Product foreign key in those fact records to be the new surrogate key you created in step 1.

This is a fairly messy change, but you should be able to automate these steps in a good programmable ETL environment like Informatica, Sagent, Ardent, ETI, Oracle, or Microsoft DTS.

There are some subtle issues in this case, too. First, you need to check to see if some other change took place for Zippy Cola 12 oz. cans on July 15, 1999. If so, then you only need to perform step 2.

Second, you can see that using a single date stamp in the slowly changing product dimension simplifies the logic. If you are using a pair of date stamps in each product dimension record, then you need to find the closest previous-to-July-15 product record for Zippy Cola 12 oz. cans and change its end_date to July 15, 1999, and you also need to find the closest subsequent-to-July-15 product record for Zippy Cola 12 oz. cans and set the end_date for the July 15, 1999 entry to begin_date of that next record. Got it???

Finally, you can see from this example why the surrogate keys for all dimensions, except time, cannot be ordered in any way. You never know when you are going to have to assign a surrogate key for a late-arriving record.

In most of your data warehouses, these late-arriving fact and dimension records are unusual, hopefully. If nothing else, they are bothersome because they change the counts and totals for prior history. But we have taken a pledge as keepers of the data warehouse to present the history of our enterprise as accurately as possible, so we should welcome the old records because they are making our databases more complete.

Some industries, like health care, have huge numbers of late-arriving records. In those cases, these techniques, rather than being specialized techniques for the unusual case, may be the dominant mode of processing.

Courtesy :

www.intelligententerprise.com/000929/webhouse.jhtml

.

What do you think of this post?

Awesome (0) Interesting (1) Useful (1)