

Design Tip #79 When Do Dimensions Become Dangerous?

By Ralph Kimball

In many organizations, either the customer or product dimensions can have millions of members. Especially in the early days of data warehousing, we regarded these large dimensions as very dangerous because both loading and querying could become disastrously slow. But with the latest 2006 technology, fast processors, and gigabytes of RAM, do we have to worry any more about these large dimensions, and if so, when does a dimension become dangerous?

How big can a dimension get? Consider a typical wide customer dimension describing account holders in a large bank. Suppose there are 30 million account holders (customers) and we have done a good job of collecting 20 descriptive and demographic attributes for each customer. Assuming an average width of 10 bytes for each field, we start with $30 \text{ million} \times 20 \times 10 = 6 \text{ GB}$ of raw data as input for the ETL data loader. Of course, if we collect 100 attributes instead of 20, our dimension is five times as big, but let's avoid that one for now.

Although OLAP vendors may disagree, I think a 30 million row dimension puts us solidly in dangerous territory for OLAP deployments. Remember that usually any Type 1 or Type 3 changes to a dimension in an OLAP system forces all OLAP cubes using that dimension to be rebuilt. Be careful with Type 1 and Type 3 changes, if you can even build a cube with such a large dimension!

Relational systems do not have this sensitivity to Type 1 and Type 3 changes, but a properly supported ROLAP system needs to place an index (typically a bitmap index) on every field in the dimension. In most relational systems, this level of indexing adds as much as a FACTOR of 3 to the storage size of the dimension. Now we are up to as much as 18 GB for our customer dimension.

Most serious relational deployments should be able to support this 18 GB dimension at query time if the dimension is not also being updated. But two danger scenarios lurk close by: update frequency and slowly changing dimensions.

If your 30 million member dimension requires thousands of insertions, deletions, and updates per week, you need to plan this administration very carefully. The problem is that you can't afford to drop all the indexes every time you load and you probably can't find a way to partition the dimension in order to make the dimension updating more efficient. So you have to perform these admin actions in the presence of all your indexes. Perhaps if you are lucky, you can get your database to defer index updates until after a batch load process runs to completion. You need to study these options carefully, and consider batching certain kinds of updates together.

Perhaps the biggest threat to our dimension, however, comes from Type 2 tracking of attribute changes. Remember that this means every time any attribute in a customer record is updated, we do not overwrite; rather we issue a new record with a new surrogate key. If the average customer record undergoes two updates per year, then in three years, our 30 million rows become 180 million rows, and the 18 GB of storage becomes 108 GB. All the preceding discussions get worse by a factor of six. In this case, I would try very hard to split the customer dimension into at least two pieces, isolating the "rapidly changing" attributes (such as demographics) into an abstract demographics dimension. This takes most of the Type 2 pressure off of the original customer dimension. In a financial reporting environment this is an effective approach because our fact tables are normally periodic snapshots, where we can guarantee that the customer key and the demographics key have a

target record in the fact table every reporting period. We have described this process of splitting a dimension many times over the years under the topic “rapidly changing monster dimensions”.

Some of you readers may have successfully wrangled a dangerous dimension of this size or even larger. Please email me describing your technology and your approach and I’ll share it in a Design Tip.