

Kimball Design Tip #26: Adding An Audit Dimension To Track Lineage And Confidence

By Ralph Kimball

Whenever we build a fact table containing measurements of our business, we surround the fact table with "everything we know to be true". In a dimensional model, this everything-we-know is packaged in a set of dimensions. Physically we insert foreign keys, one per dimension, into our fact table, and connect these foreign keys to the corresponding primary keys of each dimension. Inside each dimension (like product or customer) is a verbose set of highly correlated text-like descriptors representing individual members of the dimension (like individual products or customers).

We can extend this everything-we-know approach to our fact table designs by including key pieces of metadata that are known to be true when an individual fact record is created. For instance, when we make a fact table record, we should know such things as

1. what source system supplied the fact data (multiple descriptors if multiple source systems).
2. what version of the extract software created the record.
3. what version of allocation logic (if any) was used to create the record.
4. whether a specific "N.A. encoded" fact field actually is unknown, impossible, corrupted, or not-available-yet.
5. whether a specific fact was altered after the initial load, and if so, why.
6. whether the record contains facts more than 2, 3, or 4 standard deviations from the mean, or equivalently, outside various bounds of confidence derived from some other statistical analysis.

The first three items describe the lineage (provenance) of the fact table record. In other words, where did the data come from? The last three items describe our confidence in the quality of data for that fact table record.

Once we start thinking this way, we can come up with a lengthy list of metadata items describing data lineage and data quality confidence. But for the purpose of this design tip, we'll stop with just these six. A more elaborate list can be found in the discussion of Audit Dimensions in the Data Warehouse Lifecycle Toolkit.

Although these six indicators could be encoded in various ways, I prefer text encoding. Ultimately we are going to constrain and report on these various audit attributes, and we want our user interfaces and our report labels to show as understandable text. So, perhaps the version of the extract software (item #2) might contain the value "Informatica release 6.4, Revenue extract v. 5.5.6". Item #5 might contain values such as "Not altered" or "Altered due to restatement".

The most efficient way to add the lineage and confidence information to a fact table is to create a single Audit foreign key in the fact table. A 4-byte integer key is more than sufficient, since the corresponding Audit dimension could have up to 4 billion records. We won't need that many!

We build the Audit dimension as a simple dimension with seven fields:

Audit Key (primary key, 4 byte integer)

Source System (text)
Extract Software (text)
Allocation Logic (text)
Value Status (text)
Altered Status (text)
Out of Bounds Status (text)

In our backroom ETL (extract-transform-load) process, we track all of these indicators and have them ready at the moment when the fact table record is being assembled in its final state. If all six of the Audit fields already exist in the Audit dimension, we fetch the proper primary key from the Audit dimension and use it in the Audit dimension foreign key slot of the fact table. If we have no existing Audit dimension record applicable to our fact table record, we add one to the maximum value of the Audit dimension primary key, and create a new Audit dimension record. This is just standard surrogate key processing. Then we proceed as in the first case. In this way, we build up the Audit dimension over a period of time.

Notice that if we are loading a large number of records each day, almost all of the records will have the same Audit foreign key, since presumably nearly all of the records will be "normal". We can modify the processing in the previous paragraph to take advantage of this by caching the Audit key of the "normal" record, and skipping the lookup for all normal records.

Now that we have built the Audit dimension, how do we use it?

The beauty of this design is that the lineage and confidence metadata has now become regular data, and can now be queried and analyzed along with the other more familiar dimensions. There are two basic approaches to decorating your queries and reports with Audit dimension indicators.

The "poor man's" approach simply adds the desired Audit attributes directly to the Select list of an SQL query. In other words in a simple sales query like

```
SELECT PRODUCT, SUM(SALES)
```

you augment the query to read

```
SELECT PRODUCT, VALUE_STATUS, SUM(SALES), COUNT(*)
```

Now your report will sprout an extra row whenever an anomalous data condition arises. You will get a count that lets you judge how bad the condition is. Notice that before you did this design, the NA (null) encoded data values just dropped silently out of your report without raising an alarm, because SUM ignores null values.

The "rich man's" approach performs full fledged drill across queries producing separate columns (not separate rows) with more sophisticated indicators of lineage or quality. For instance, our simple sales query in the above example could be embellished to produce report headings across each row like

```
PRODUCT >> SUM(SALES) >> PERCENT OF DATA FROM OLD SOURCE SYSTEM >>  
PERCENT OF DATA CORRUPTED
```

and so on. Front end tools like Cognos, Business Objects, and Microstrategy are capable of these drill across reports, using "multi-pass SQL".

I would be interested in hearing about any of your designs where you have converted metadata into data and made a dimension out of it. Write to me at ralph@kimballgroup.com.