

Kimball Design Tip #10: Is Your Data Correct?

By Ralph Kimball

A common problem in the data warehouse backroom is verifying that the data is correct. Is the warehouse an accurate image of the production system? Was this morning's download complete? Could some of the numbers be corrupted?

There is no single technique for validating a data load because there is so much variability in the sources of the data. If you are downloading an unchanged image of a production source, preserving the original granularity, then you can probably run a "flash report" on the production system with up-to-the minute totals, and then you can recapitulate the same report on the data warehouse. In this case, you "know" the answer beforehand and the two results should match to the last decimal place.

But it is more common not to have a known baseline of data. Maybe you are receiving the individual sales transactions from 600 retail stores every night. You can certainly perform a gross count on the number of stores reporting, but how can you apply some additional judgment to say whether the data is "probably correct"?

Using the 600 stores as an example, let's look at the sales totals for each department in each store each morning and ask if today's new numbers are "reasonable". We will decide that today's sales total is reasonable if it falls within 3 standard deviations of the mean of the previous sales totals for that department in that store. Arrghh. Statistics. But hang in there: it's not too bad. We chose 3 standard deviations because in a "normal" distribution, 99% of the values lie within 3 standard deviations above or below the mean.

I'll describe the process in words. After that I'll include some SQL. You can just read the words if you want to get the main drift.

In order to make this process run fast, you want to avoid looking at the complete time history of old data when you are calculating the standard deviation. You can avoid looking at the complete time history by keeping three accumulating numbers for each department in each store in a special table used only in the data validity pass. You need to keep the number of days being accumulated, the accumulating sum of each day's sales (by department by store) and the accumulating sum of the SQUARE of each day's sales (by department by store). These could be kept in a little stand alone accumulating department table. The grain of this table is department by store and the three numeric fields NUMBER_DAYS, SUM_SALES and SUM_SQUARE_SALES are all Type 1 attributes that are overwritten each day. You can update all three of these fields just by adding the next day's values to the ones already there. So if you have 600 stores and 20 departments in each store, this table has 12,000 rows but does not grow over time. The table also carries the store names and department names in each row.

Now, using this accumulating department table, you look at all 12,000 department totals in this morning's data load, and kick out this morning's numbers that are more than 3 standard deviations from the mean. You can chose to examine the individual unusual numbers if there aren't too many, or you can reject the entire load if you see more than a threshold number of sales totals more than 3 standard deviations from the mean.

If this morning's load passes muster, then you release the data to your end users, and then you

update the accumulating department table to get ready for tomorrow's load.

Here's some untested SQL that may possibly work. Remember that the standard deviation is the square root of the variance. The variance is the sum of the squares of the differences between each of the historical data points and the mean of the data points, divided by N-1, where N is the number of days of data. Unfortunately, this formulation requires us to look at the entire time history of sales, which although is possible, makes the computation unattractive. But if we have been keeping track of SUM_SALES and SUM_SQUARE_SALES, we can write the variance as $(1/(N-1))*(SUM_SQUARE_SALES - (1/N)*SUM_SALES*SUM_SALES)$. Check my algebra.

So if we abbreviate our variance formula with "VAR" then our data validity check looks like

```
SELECT s.storename, p.departmentname, sum(f.sales)
FROM fact f, store s, product p, time t, accumulatingdept a WHERE
```

```
(first, joins between tables...)
f.storekey = s.storekey and
f.productkey = p.productkey and
f.timekey = t.timekey and
s.storename = a.storename and
p.departmentname = a.departmentname and
```

```
(then, constrain the time to today to get the newly loaded data...) t.full_date = #July 13,
2000# and
```

```
(finally, invoke the standard deviation constraint...)
HAVING
ABS(sum(f.sales) - (1/a.N)*a.SUM_SALES) > 3*SQRT(a.VAR)
```

where we expand VAR as in the previous explanation and use the "a." prefix on N, SUM_SALES and SUM_SQUARE_SALES. We have assumed that departments are groupings of products and hence are available as a rollup in the product dimension.

Embellishments on this scheme could include running two queries: one for the sales MORE than three standard deviations above the mean, and another for sales LESS than three standard deviations below the mean. Maybe there is a different explanation for these two situations. This would also get rid of the ABS function if your SQL doesn't like this in the HAVING clause.

If you normally have significant daily fluctuations in sales (e.g., Monday and Tuesday are very slow compared to Saturday), then you could add a DAY_OF_WEEK to the accumulating department table and constrain to the appropriate day. In this way you don't mix the normal daily fluctuations into our standard deviation test.