

Kimball Design Tip #37: Modeling A Pipeline With An Accumulating Snapshot

By Ralph Kimball

In the Intelligent Enterprise article *Fundamental Grains* (www.intelligententerprise.com/db_area/archives/1999/993003/warehouse.shtml) I described the three different types of grains that all fact tables seem to fall into. Remember that a fact table is a place where we store measurements emanating from a particular business process. Dimension tables surround and describe these measurements.

Remember also that the grain of a fact table is the definition of exactly what does a fact record represent. It is certainly true that the KEY of a fact table implements the grain, but frequently the clearest declaration of the grain is a business concept rather than a technical list of foreign keys. For instance, the grain of a fact table representing an order taking process may be "line item on the order" whereas the technical definition of the key to that table may turn out to be "invoice number BY product BY promotion".

In all the thousands of fact table designs I have seen and looked at, they all have sorted themselves into three fundamental grains:

1. The TRANSACTION grain, that represents a point in space and time;
2. The PERIODIC SNAPSHOT grain, that represents a regular span of time repeated over and over; and
3. The ACCUMULATING SNAPSHOT grain, that represents the entire life of an entity.

The accumulating snapshot fact table is unusual in a number of ways. Unlike the other grains, the accumulating snapshot usually has a number of Time dimensions, representing when specific steps in the life of the "accumulating entity" take place. For example, an order is

- 1) created,
- 2) committed,
- 3) shipped,
- 4) delivered,
- 5) paid for, and maybe
- 6) returned.

So the design for an orders accumulating snapshot fact table could start off with six time keys, all being foreign keys to views on a single date-valued dimension table. These six views of the date table are called "roles" played by the date table and they are semantically independent as if they were separate physical tables, because we have defined them as separate views.

The other unusual aspect of the accumulating snapshot fact table is that we revisit the same records over and over, physically changing both foreign keys and measured facts, as the (usually short) life of the entity unfolds. The orders process is a classic example.

Now that we have reminded ourselves of the salient design issues for accumulating snapshot fact tables, let's apply this design technique to a pipeline process. We'll use the student admissions pipeline, but those of you interested in sales pipelines should be able to apply this design to your situation easily.

In the case of admissions tracking, prospective students progress through a standard set of admissions hurdles or milestones. We're interested in tracking activities around no less than 15 key steps in the process, including 1) receipt of preliminary admissions test scores, 2) information requested (via web or otherwise), 3) information sent, 4) interview conducted, 5) on-site campus visit, 6) application received, 7) transcript received, 8) test scores received, 9) recommendations received, 10) first pass review by admissions, 11) application reviewed for financial aid, 12) final decision from admissions, 13) student accepted, 14) student admitted and 15) student enrolled. At any point in time, managers in the admissions and enrollment departments are interested in how many applicants are at each stage in the pipeline. It's much like a funnel where many applicants enter the pipeline, but far fewer progress through to the final stage. Managers also want to analyze the applicant pool by a variety of characteristics. In this admissions example, we can be confident that there is a very rich Applicant dimension filled with interesting demographic information.

The grain of the accumulating snapshot is one row per applicant. Because this is an accumulating snapshot, we revise and update each applicant's unique record in the fact table whenever one of the steps is completed.

A key component of the design is a set of 15 numeric "facts", each a 0 or 1 corresponding to whether the applicant has completed one of the 15 steps listed above. Although technically these 15 1/0 facts could be deduced from the 15 date keys, the additive numeric facts make the application elegant and easy to use with almost any query or reporting tool.

As an extra goodie, we add four more numeric additive facts representing "lags" or time gaps between particularly important steps in the process. These include

Information Requested	==>	Sent lag
Application Submitted	==>	Complete lag
Application Submitted	==>	Final Decision lag
Final Decision	==>	Accept or Decline lag

These lag facts are both good diagnostics for picking out stalled applications, but they help the managers tune the process by identifying bottlenecks.

Our final fact table design looks like

- Preliminary Test Score Receipt Date Key (FK)
- Information Requested Date Key (FK)
- Information Sent Date Key (FK)
- Interview Conducted Date Key (FK)
- On-Site Campus Visit Date Key (FK)
- Application Submitted Date Key (FK)
- Transcript Received Date Key (FK)
- Test Scores Received Date Key (FK)
- Recommendations Received Date Key (FK)
- Admissions First Pass Review Date Key (FK)
- Reviewed for Financial Aid Date Key (FK)
- Admissions Final Decision Date Key (FK)
- Applicant Decision Received Date Key (FK)
- Admitted Date Key (FK)
- Enrolled Date Key (FK)
- Applicant Key (FK)
- Admissions Decision Key (FK)
- Preliminary Test Score Receipt Quantity
- Information Requested Quantity
- Information Sent Quantity
- Information Requested-Sent Lag

Interview Conducted Quantity
On-Site Campus Visit Quantity
Application Submitted Quantity
Transcript Received Quantity
Test Scores Received Quantity
Recommendations Received Quantity
Application Complete Quantity
Application Submitted-Complete Lag
Admissions First Pass Review Quantity
Reviewed for Financial Aid Quantity
Admissions Final Decision Quantity
Application Submitted-Final Decision Lag
Accepted Quantity
Decline Quantity
Final Decision-Accepted/Decline Lag
Admitted Quantity
Enrolled Quantity

Interesting design! Imagine how easy it would be to summarize the state of the pipeline at any point in time. Although the records are obviously wide, this is not an especially big table. If you are a big state university with 100,000 applicants per year, you would only have 100,000 records per year. Assume the 17 foreign keys are all 4 byte integers (nice surrogate keys), and the 21 quantities and lags are 2 byte tiny integers. Our fact table records are then $17 \times 4 + 21 \times 2 = 110$ bytes wide. This makes about 11 MB of data per year in this fact table. Check my math. Actually this is a common outcome for accumulating snapshot fact tables. They are the smallest of the three types, by far.

I would like to hear from you about other pipeline processes where you have either already applied this modeling technique or you realize that it would fit your situation well. Write to me in the next week or two and I'll talk about good examples I receive in the next design tip.