

Design Tip #93 Transactions Create Time Spans

By Ralph Kimball

In the dimensional modeling world we talk about three kinds of fact table grains: transaction, periodic snapshot, and accumulating snapshot. If you are careful to make sure every fact table you design is based on one grain, and doesn't mix different grains, amazingly these three choices are enough to design every fact table in your data warehouse.

The transaction grain is used for measurements at a specific instant. When you make a deposit into your bank account, a transaction record is created. The record exists only if the event actually happens. We don't know when or if any more transaction events will occur. The next one could be one second later or next year.

In spite of the seemingly ephemeral nature of transactions, each transaction leaves behind a powerful legacy. At the moment of the transaction, the state of the business process is frozen and remains frozen until the next transaction occurs. As a result of your deposit, your bank account has a new balance, which remains at exactly the same value until the next transaction occurs. Thus each transaction against your bank account implicitly defines a time span until the next transaction occurs. During this time span, all aspects of your account are frozen. Wouldn't it be interesting to be able to quickly query history for bank accounts at a particular random point in time and get all the balances at that moment?

Time spans can be represented in a transaction grain fact table by adding begin and end datetime stamps to each record, and in the case of the bank account, adding a field representing the account balance that results from the transaction. These are not foreign keys to any kind of time dimension. Rather, these are full datetime fields. The begin datetime, of course, is the exact moment when the transaction takes effect. The end datetime is initially set to a maximum datetime far in the future, such as Dec 31, 9999, 11:59:59 pm. It is important to set the end datetime to a real value, not NULL, so that BETWEEN comparisons always return a value.

The end datetime remains at the artificial maximum value until the next transaction occurs. Then you must set this field to the next transaction datetime minus the smallest increment of time supported by your datetime stamp. Then the next transaction record is loaded into the fact table as described in the previous paragraph. You now have an unbroken string of time spans which can be queried using any date and time you desire.

Keep in mind the following caveats. First, your transactions must be complete. It won't work to include just customer deposits and withdrawals, while omitting other bank initiated transactions. Second, not every transaction record creates a useful unique time span. Bank account transactions fit the time span approach beautifully, but grocery store cash register transactions probably don't. So the next time you define a transaction grained fact table, give some thought to the extra flexibility that these implicit time spans can offer.