KIMBALL GROUP KIMBALL UNIVERSITY

www.kimballgroup.com Number 25, July 1, 2001

Design Tip #25 Designing Dimensional Models For Parent-Child Applications

By Ralph Kimball

The parent-child data relationship is one of the basic, fundamental structures in the business world. An invoice (the parent) has many line items (the children). Other obvious examples besides invoices include orders, bills of lading, insurance policies, and retail sales tickets. Basically, any business document with an embedded repeating group qualifies as a parent-child application, especially when the embedded line items contain interesting numerical measurements like dollars or physical units.

Parent-child applications are of extreme importance to data warehousing because most of the basic control documents that transfer money and goods (or services) from place to place take the parentchild form.

But a parent-child source of data, like invoices, presents a classic design dilemma. Some of the data is only available at the parent level and some only available at the child level. Do we need two fact tables in our dimensional model or can we do it with just one? And what do we do with the data that is only available at the parent level when we want to drill down to the child level?

Let's imagine a typical product sales invoice. The overall invoice is created by the sales agent for our company and is directed at a specific customer. Each line item on the invoice represents a different product sold to the customer.

The parent level data includes

- Date of overall invoice (dimension)
- Sales agent (dimension)
- Customer (dimension)
- Payment terms (dimension)
- Invoice number (degenerate dimension, see below)
- Total net price from the line items (additive fact)
- Total invoice level discounts representing promotional allowances (additive fact)
- Total freight charges (additive fact)
- Total tax (additive fact)
- Grand total (additive fact)

The child level data includes

- Product (dimension)
- Promotion (dimension)
- Number of units of product (additive fact)
- Unit price of product (see below)
- Extended gross price (units X price) (additive fact)
- Promotional discount for this specific product (see below)
- Extended net price (units X (unit price - promotional discount)) (additive fact)

as well as the "context" from the overall invoice (four dimensions, above).

Given our dimension and fact hints, we would seem to be done. We have two nice fact tables. The parent invoice fact table has 4 dimensions and 5 facts, and the child line item fact table has 6 dimensions and 3 facts.

But our design is a failure.

We can't roll up our business by product! If we constrain by a specific product, we don't know what to do with invoice level discounts, freight charges and tax. All of our higher level views of the business by customer, sales agent, and promotions are forced to omit the product dimension.

In most businesses, this is unacceptable.

There is only one way to fix this problem. You have to take the invoice level data and allocate down to the line item level. Yes, there is some controversy in doing this allocation, and yes, you must make some arbitrary decisions, but the alternative is not being able to analyze your business in terms of your products.

We will replace the two fact tables with a single fact table, whose grain is the invoice line item. In other words, we will consistently drop to the most atomic child level when we do a parent-child dimensional design.

Remember from our discussion of Choosing the Grain (design tip #21), that we can "decorate" a measurement with everything that is known to be true at the time of the measurement. So our single line item grain fact table has the following dimensions:

- Date of overall invoice (dimension)
- Sales agent (dimension)
- Customer (dimension)
- Payment terms (dimension)
- Product (dimension)
- Promotion (dimension)

What do we do with the invoice number? It is certainly single-valued, even at the line item level, but we have already "exposed" everything we know about the invoice in our first four dimensions. We should keep the invoice number in the design but we don't need to make a dimension out of it because that dimension would turn out to be empty. We call this characteristic result a "degenerate dimension".

- Invoice number (degenerate dimension).

Now our facts for this line item child fact table include

- Number of units of product (additive fact)
- Gross extended product price (units X price) (additive fact)
- Net extended product price (units X (unit price - promotional discount)) (additive fact)
- Allocated invoice level discounts representing promotional allowances (additive fact)
- Allocated freight charges (additive fact)
- Allocated tax (additive fact)

We don't include the unit prices or discounts as physical facts because we can always divide the extended amounts by the number of units in our reporting application to get these non-additive quantities.

We can instantly recover the exact invoice level amounts by simply adding up all the line items under a specific invoice number. We don't need the separate invoice parent fact table because it is only a simple aggregation of our more granular line item child fact table. We have in no way compromised the invoice totals by performing the allocations down to the line item.

And, best of all, we can now smoothly roll up our business to the highest levels, slicing by product, and including the allocated amounts to get a true picture of our net revenues.

This technique of descending to the line item level and building a single granular fact table is at the heart of data warehouse modeling. It is our way of making good on the promise that we can "slice and dice the enterprise data every which way".