

Publication: Data Management Review

Column Title: Data Warehouse Delivery

Author: Douglas Hackney

Headline: De-Coddify Your Brains

Issue: September 1998

[begin copy]

The DAs (Data Architects) and DBAs (Data Base Administrators) that are employed on data warehousing projects have spent the last 20 years learning and implementing the principles of normalized data design laid down by the inventor of the relational data base, Dr. Codd. These normalized database design principles mandate that no data point should ever be duplicated and that a database table should never contain a field that is not a unique attribute of the key of that table. These principles yield a database design that is optimized for transactional systems and for ease of maintenance and update.

The principles of successful data warehouse *end user access* database design, however, are the antithesis of Dr. Codd's normalized database design principles. When designing a data warehouse or data mart that users are going to "touch", or use for direct access for query and analysis purposes, there is widespread replication of data and rampant denormalization. When designing a database for end user utilization, there is only one litmus test: **ease of use**.

If the addition of a replicated data point or a derived metric will make the system easier to use, then it is added. For example, when designing an aggregation data set, all members of a hierarchy are included, along with all descriptive elements for all key values. For instance, an aggregation for sales by day would also include columns for week, month, quarter and year. This allows users to easily and simply roll-up to a higher level of the hierarchy. In this way, one aggregation table can serve the needs of users who need daily, monthly, quarterly or annual totals. By including additional hierarchies, such as sales geography, the table becomes even

more powerful, allowing roll ups to any level of multiple hierarchies. (see figure 1) This is a key strategy to minimize the number of aggregation tables required to meet the needs of the users.

Sales by Cust by Territory

- CUST_ID
- CUST_DESC
- CUSTYP_ID
- CUSTYP_DESC
- TER_ID
- TER_DESC
- ZN_ID
- ZN_DESC
- RG_ID
- REG_DESC
- CCYYDDD
- CCYYWW
- CCYYMM
- CCYYQ
- CCYY
- QUAN
- COST
- GR_SLS
- NET_SLS

Figure 1 – Denormalized aggregation with multiple hierarchies.

When designing a dimension for a star schema, derived metrics are included to aid ease of use, such as Total Sales Prior Year to Date, Total Sales Year to Date, Total Sales PYTD/CYTD Delta, Total Sales PYTD/CYTD Delta %, Top Ten in Total Sales YTD Y/N, Bottom Ten in Total Sales YTD Y/N, Ranking in Sales YTD, etc. These metrics are often standardized and utilized in all appropriate dimensions, such as customer, product, geography, sales geography, etc.

The Data Sandwich

The design principles for data warehousing can perhaps best be illustrated by the concept of the “data sandwich.” Normalized database design is very similar to how a grocery warehouse and a typical American supermarket grocery store are organized. In order to purchase a sandwich at a supermarket (we will ignore the deli counter for the purposes of this discussion) you must first discover where everything is located. Anyone who has moved to a new neighborhood or city can provide ample testimony to how much longer it takes to shop the first few times at a new store than it did at your old one. It is not uncommon to spend a couple of hours on each of the first few visits to the new store while you learn the location of all the items you need. In a normalized data model, the users must also know where everything is located. Due to turnover and attrition, you can be assured that 15-25% of the users will be new to the system at any given point in time.

Once we know where everything is in the grocery store, we must now visit each department in order to obtain the elements of our sandwich. We must visit the produce section for the lettuce, the dairy section for the cheese, the meat department for the meat, the bakery section for the bread and the condiment aisle for the mayonnaise. We must then proceed to the checkout, purchase all of our elements, and then proceed to the parking lot where we can assemble our sandwich and finally eat it. A normalized data model forces the users to make this same journey, venturing to each individual table to select the data they need, then assembling it, before they can use it. This experience is not optimum for the consumers, as they must intimately know the structure and use of the data and do all the assembly. It is, however, optimized for the operations staff that needs to maintain the grocery store and the normalized data model.

In the grocery store, when the mayonnaise needs to be restocked, the night staff pulls a pallet of mayonnaise into the condiment aisle and restocks the shelves. All the different mayonnaise is centrally located in this aisle, so it is easy to restock, add a new type, re-price or re-arrange the mayonnaise. It is the same for a normalized data model. If a new product type is created, only the product type table needs to be modified by adding a new row to reflect the new product type. This type of design is optimized for operations, and is a favorite of technical staffs.

Contrast this to the world of convenience stores, such as a 7-11. If a consumer wants a sandwich in a convenience store, they simply walk back to the sandwich section, select the type of sandwich they want, purchase it and eat it. This experience is optimized for consumers. The same is true of a denormalized data model. The user does not need to know the detailed contents and proper utilization of dozens, scores, hundreds or thousands of tables. Again, the system is optimized for consumers.

The key issue here is that **somebody has to do the work**. If you expect users to do the work on a sustained basis, you will not succeed. Any system that puts the onus on the user community is not politically sustainable. In the data sandwich example, somebody has to make the sandwiches. If you expect your users to make the data sandwiches, you will be bitterly disappointed. You need to resign yourself to the fact that in systems designed for end user access, the IT team must do the work up-front. It is true that database designs that incorporate extensive denormalization and replication are more difficult to maintain and modify than normalized designs. It is also true that practically no one will use a system that forces them to navigate hundreds of tables and assemble all their own data sandwiches. So, you have a choice: you can either build a system that is easy to build and maintain, and no one will use; or you can build a system that is difficult to build and maintain, and many will use. You cannot escape the simple law of **somebody has to do the work**.

The transition by DAs and DBAs from normalized database design principles to data warehouse end user access database design principles is one of the most challenging cultural aspects to

data warehouse project management. The necessity of this transition cannot be overemphasized, nor the requirement for one simple test for all design decisions: ease of use.

[end copy]

[add the following to the usual tag]

[Excerpted from "The Seven Deadly Sins of Data Warehousing", a forthcoming book from Addison Wesley Longman.]