

Kimball Design Tip #34: You Don't Need An EDW

By Ralph Kimball

"EDW" stands for enterprise data warehouse, but more to the point, it is the name for a design approach. The EDW approach differs materially from the Data Warehouse Bus Architecture approach. The EDW embodies a number of related themes that need to be contrasted individually from the DW Bus approach. It may be helpful to separate logical issues from physical issues for a moment.

Logically, both approaches advocate a consistent set of definitions that rationalize the different data sources scattered around the organization. In the case of the DW Bus, the consistent set of definitions takes the specific form of conformed dimensions and conformed facts. With the EDW approach, the consistency seems much more amorphous. You must take it on faith that if you have a single highly normalized E/R model of all the enterprise's information, you then know how to administer hundreds or thousands of table consistently. But, overlooking this lack of precision, one might argue that the two approaches are in agreement up to this point. Both approaches strive to apply a unifying coherence to all the distributed data sources.

A side issue of the EDW enterprise data model is that frequently these models are idealized models of information rather than real models of data sources. Although the exercise of creating the idealized information model is useful, I have seen a number of these big diagrams that never get populated. I have also written a number of articles trying to shed a harsh light on the related claim that the big normalized models encapsulate an organization's "business rules". At best, the normalized models enforce SOME of the DATA rules (mostly many-to-1 relationships), and almost NONE of what a business procedures expert would call business rules. The explanatory labeling of the join paths on an E/R diagram rarely if ever is carried into the code of the backroom ETL processes or the front room query and report writing tools.

Even if we have a tenuous agreement that both approaches have the same goal of creating a consistent representation of an organization's data, as soon as you move into physical design and deployment issues, the differences between the EDW and the DW Bus become really glaring.

As most of you know, the conformed dimensions and conformed facts take on specific forms in the DW Bus architecture. Conformed dimensions are dimensions that have common fields, and the respective domains of the values in these fields are the same. That guarantees that you can perform separate queries on remote fact tables connected to these dimensions and you will be able to merge the columns into a final result. This is of course, drill across. I have written extensively on the steps required to administer conformed dimensions and conformed facts in a distributed data warehouse environment. I have never seen a comparable set of specific guidelines for the EDW approach. I find that interesting because even in a physically centralized EDW, you have to store the data in physically distinct table spaces, and that necessitates going through the same logic as the replication of conformed dimensions. But I have never seen systematic procedures described by EDW advocates for doing this. Which tables do you synchronously replicate between table spaces and when? The DW Bus procedures describe this in great detail.

The flat 2NF (second normal form) nature of the dimensions in the DW Bus design allows us to administer the natural time variance of a dimension in a predictable way (SCD types 1, 2, and 3).

Again, in the highly normalized EDW world, I have never seen a description of how to do the equivalent of slowly changing dimensions. But it would seem to require copious use of time stamps on all the entities, together with a lot more key administration than the dimensional approach requires. By the way, the surrogate key approach I have described for administering SCDs actually has nothing to do with dimensional modeling. In an EDW, the root table of a snowflaked "dimension" would have to undergo exactly the same key administration (using either a surrogate key or a natural key plus a date) with the same number of repeated records if it tracked the same slowly changing time variance as the DW Bus version.

The flat 2NF nature of dimensions in the DW Bus design allows a systematic approach to defining aggregates, the single most powerful and cost effective way to increase the performance of a large data warehouse. The science of dimensional aggregation techniques is intimately linked to the use of conformed dimensions. The "shrunken" dimensions of an aggregate fact table are perfectly conformed subsets of the base dimensions in the DW Bus architecture. The EDW approach, again, has no systematic and documented approach for handling aggregates in the normalized environment or giving guidance to query tools and report writers for how to use aggregates. This issue interacts with "drilling down" described below.

The EDW architecture is both logically and physically centralized, something like a planned economy. Maybe this is unfair, but I think this approach has the same subtle but fatal problem that a planned economy has. It sounds great up front, and the idealistic arguments are hard to refute before the project starts. But the problem is that a fully centralized approach assumes perfect information "a priori" and perfect decision making afterward. Certainly, with planned economies, that was a major reason for their downfall. The DW Bus architecture encourages a continuously evolving design with specific criteria for "graceful modification" of the data schemas so that existing applications continue to function. The symmetry of the dimensional design approach of the DW Bus allows us to pinpoint exactly where new or modified data can be added to a design to preserve this graceful character.

Most importantly, a key assumption built into most EDW architectures is that the centralized EDW "releases" data marts. These data marts are often described as "built to answer a business question". Almost always this comes from inappropriate and premature aggregation. If the data mart is only aggregated data, then of course there will be a set of business questions that cannot be answered. These questions are often not the ones asking for a single atomic record, but rather questions that ask for a precision slice of large amounts of data. A final, unworkable assumption of the EDW is that if the user wants to ask any of these precise questions, they must leave the aggregated dimensional data mart and descend into the 3NF atomic data located in the back room. EVERYTHING is wrong with this view, in my opinion. All of the leverage we have developed in the DW Bus is defeated by this hybrid architecture: drilling down through conformed dimensions to atomic data; uniform encoding of slowly changing dimensions; the use of performance enhancing aggregates; and the sanctity of keeping the back room data staging area off limits to query services.

Anyway, as you probably know, the Data Warehouse Lifecycle Toolkit book is devoted to the task of building an "enterprise" data warehouse, but in a distributed fashion based on the DW Bus architecture. If you would rather read free articles on these topics then here are some relevant links to topics raised in this design tip, in reverse chronological order. You might want to read them starting with the oldest!

- * [The Anti-Architect](#)
- * [Managing Helper Tables](#)
- * [Joint Effort](#) (administering a distributed DW)
- * [Backward in Time](#)
- * [Enforcing the \(Business\) Rules](#)
- * [There are No Guarantees \(in an E/R model\)](#)
- * [The Matrix](#) (planning a distributed DW Bus architecture)
- * [The Data Webhouse Has No Center](#)
- * [Coping With the Brave New Requirements](#)

- * [Brave New Requirements for Data Warehousing](#)
- * [Pipelining Your Surrogates](#)
- * [Surrogate Keys](#)
- * [Is Data Staging Relational?](#)
- * [Bringing Up Supermarts](#) (this article describes the DW Bus architecture before we adopted the term "Bus")
- * [Aggregate Navigation With \(Almost\) No Metadata](#)