

Strategies for Scoping a Data Warehouse Engagement

John Shantz
President
Data Warehouse Consultants LLC
August 2002

The purpose of this paper is to define some key considerations for developing the scope requirements of a data warehouse engagement. While setting scope boundaries is very important for any project, data warehousing assignments are particularly susceptible to “scope creep” because of the difficulty in creating definable measures of success at the beginning of the project. In other words, it is difficult to maintain scope and still tell the client what they will get. Below, I propose some strategies for achieving this goal.

Business Questions

For many data warehouses in the Health & Human Services (HHS) field we have set the scope for data warehouses (DW) by defining the number of business questions that will be answered by the final product. This quantity approach is a useful tool for helping the client to understand what they are buying, but less useful in managing scope once the project is underway.

The problem is that a business question is not a very stable baseline. Anyone who has ever been to an all-you-can-eat buffet knows that “all-you-can-eat” is a relative term. Likewise, a business question can be either straightforward (like a two-year-old at the buffet) or loaded to answer many questions at once (a sumo wrestler). Consider the following business question “evolution”:

Client 1 (December 1999)

5. How many children are known to the [child welfare] system by geography?

Client 2 (December 2001)

48. What is the unduplicated count of active clients who are referred for mental health, public health, child abuse, or substance abuse service in a month, quarter, year by number of those referred who are active to Department of Mental Health, Department of Public Health, Department of Family and Children or Department of Alcohol and Drug services (respectively) during that same time period by program dimensions, client dimensions, and department combination?

Keep in mind that “program dimensions, client dimensions and department combination” all contain multi-level hierarchies within themselves in addition to the other dimensions. Hopefully, this example makes the scope management problems of the “number of business questions” approach self-evident.

We cannot forget that business questions do serve an important purpose, however. They tell the client what they are getting. For that reason, they must remain as a tool in defining the parameters of a data warehouse engagement. They should not be used to define the technical aspects of what will be included in the data warehouse, however. For that purpose they are wholly inadequate and other forms of scope management must be considered.

To manage the technical aspects of what will be included in the data warehouse, you need technical measures. This is somewhat difficult, however, because technical measures are not

always available at the onset of the engagement. To handle this issue, multiple levels of quantity-defined goals must be set. I refer to this as the “Tiered Approach”.

Number of Source Systems

It is true that a good data warehouse begins from evaluating the needs of management and designing a structure that will support those needs. Once the design of the system is complete, however, the data warehouse is *built* from the ground-up like any other system.

Defining the number of source systems that will contribute to the warehouse naturally becomes a starting point to define scope. Limiting the number of source systems that will contribute data to the warehouse will limit the amount of information that will later be merged together to create the end product. Combining clients and data from across systems is always one of the more challenging aspects of a data warehouse engagement (especially in HHS engagements). To understand the scope of a project, it is absolutely necessary to define which systems will be included in the warehouse at the beginning of the effort. If this is not possible, the *maximum number* of systems must at least be defined.

In one previous engagement this was a point that did not become a major issue, but it was important nonetheless. During a “Cross Systems” phase of the engagement, there was a desire to match address information from each of the source systems with the Registrar of Voters (ROV) database. What was not recognized early-on in the discussions with the client was that the ROV database really involved mapping another source system and its associated processes into the warehouse. This became a significant effort. Had the specific source systems been defined ahead of time (or at least the number), this system would surely have not been a high priority to put into the warehouse and would have been excluded. In the end significant effort was spent integrating this system, but due to problems with matching addresses the usefulness of its inclusion was questionable at best.

Number of Extracts

History has shown that it really is not enough to limit the number of extracts contributing data to the warehouse. In addition, you must limit the “subject areas” of information that are provided. The means to accomplishing this goal is to limit the number of extracts from one or more systems.

A strong tendency exists on the part of the business units to essentially do “data dumps” into the warehouse so that everything can become available. This approach is not good for the construction of a well-focused warehouse, and even worse for the data warehouse development team.

Business units must carefully examine their data and determine the logic for combining disparate elements in their systems and deciding how to bring the data together in general. *The best time to do this is before the extracts are provided.* The business units know the data better than anyone, and they know the rules to join various files or tables together. By allowing the business units to dump un-integrated files or tables into the warehouse, you excuse them from deciding how to resolve the business-level discrepancies in the data integration.

The result of these data dumps severely hurts scope management. The problem is that the “combination logic” is completely unknown. Source data entities might fit together relatively easily, or complex logic may have to be performed to join the information correctly. Moreover, the data warehouse team defines the logic to bring the data together, and it may not be correct.

An iterative process develops where the warehouse team makes small logic changes that are subsequently reviewed by the business units. Errors are found, the logic must be corrected, the warehouse must be re-loaded and the cubes must be rebuilt each time. *This is a scope monster.*

Unfortunately, this is also a difficult problem to address. Previous attempts to limit this problem have focused on limiting the number of “data elements”¹ that will be brought into the warehouse. This approach does not work in practice, however.

The problem with limiting data elements is that it really isn't very hard to pull multiple fields from a file or table once you have decided to pull one field. For example, it is negligibly more difficult to get the last name once you've retrieved the first name (assuming these are stored logically). Likewise, why not get the date of birth if it is there? You just might need it later. **In reality, it is actually a good idea to pull all of the related information into the warehouse once you have decided to visit a particular table or file.** It saves you work when you discover later that you need it to answer a business question. The only trade-off is a need for increased storage, which is essentially a non-issue in the world of 500 GB hard drives.

The difficulty comes into play once you decide to access a new entity, however. Now you must define how the new entity you are adding relates to the first. You must define the relationships, commonize the grain of the data and perform any number of other tasks related to linking the entities. This is the “combination logic” that kills scope.

This section began by mentioning “subject areas” of information, which will now be explored. A subject area is an integrated set of source tables or files that the client defines before providing the extracts. It can also be thought of as a “grain-level” of information that will be provided in the warehouse. Admittedly, this is an abstract concept.

Essentially, we must set scope around the different types of questions that will be addressed in the business questions. Questions about clients fall into one subject area. Questions about the cost of service are included in another. Questions about performance of outcomes based on form data are yet another. Answering three business questions when each one comes from a separate subject area is much more difficult than answering three questions about the cost of service.

The tool to technically limiting the abstract concept of subject areas in the warehouse is limiting the number of extracts that will be provided from each source system. Ideally, you want to have an *equal* number of extracts and subject areas in the warehouse. If the business unit wants to know how many clients they have by twenty different dimensions, they should provide all the data to support those questions in one extract. If they want to measure outcomes based on scores from several standardized forms, provide the data from all the forms in one extract.

Of course this is an ideal situation. Ultimately, I believe limiting the number of extracts to no more than one to two extracts per expected subject area is a good compromise. For example, if you foresee supporting three subject areas or star-schemas for a particular source, limit the number of extracts to four or five from that particular source system. If more than one source is being used from a single business unit, limit the total number of extracts that can be delivered from the unit. This is a major enhancement that can be added to any scoping process.

¹ A “Data Element” can be thought of as a data field in a table or a field in a hierarchical data file.

The benefits of structuring the project in this manner are immense. To construct an extract that will support client data across 20 different dimensions, the business unit **MUST** analyze the data and define how it all fits together. There is no other way to do it. This would cut the work of the data warehouse team considerably and create a much better product. The data warehouse team could focus primarily on integrating data from *across* source systems instead of focusing on integrating data *within* source systems.

Other benefits exist as well. It greatly enhances the maintainability of the warehouse in the long run. At the aforementioned client, the Cross Systems phase developed over 200 mappings to build the data warehouse. Many of these were combining “data dump” type extracts that should have been combined in the “pre-extract” phase. The result is a spider-web (although a carefully documented one) of logic that runs well, but will be difficult to change in the future. Had the business units developed the combination logic on their end, they could maintain the logic much more efficiently as the needs evolved.

Of course this is not without cost. The business units may not have the resources to pull the information together. Historically, the data warehouse team has better software and more technical expertise on integrating data than the business units can muster. The units will therefore resist performing more of the logic on their end. The fact remains, however, that the business unit *is* where the data should be combined.

Because more of the onus will be on the business units to provide combined information, they will require more lead-time for what is expected of them. Hence, timelines that are dependent on the client providing the extracts will have to be managed more aggressively. Detailed extract-related milestones and deadlines must be established early in the project cycle, as well as specific consequences if those client-dependent milestones are not met.

Number of Derived Fields

One final tool to limiting scope on a data warehouse project is reducing the number of derived fields that will be provided in the warehouse. The primary purpose for this limitation is similar to the problem of integrating multiple source entities mentioned in the section above. Ultimately it is in the best interests of everyone involved for the business units to determine the derived fields pre-extract rather than relying on the data warehouse team to derive them instead.

If you rely on the DW team to derive the field, you again develop an iterative process where the data must be checked by the business unit, corrected, reloaded and rechecked in infinitum. It is understood that some fields must be derived in the warehouse, especially the ones that rely on data from across systems. But concepts and fields that can be generated using data from only one business unit should generally be created there and provided to the warehouse team. The scope of the DW project should contain language to this effect.

Again we saw examples of this at previous clients. One department wanted to identify “Active Clients” which was not a field that they kept anywhere in their systems. In the operational systems, a complex program was used to traverse the hierarchical database and determine if a client was “Active” at any given time.

Against the advice of the data warehouse team, the probation department pushed for the warehouse as the place to implement this logic. After investing weeks of effort to implement this logic to no avail, the DW team and the department came to the conclusion that it may not be possible to precisely replicate the logic that the business unit wanted with the data that was

available. The final result was that the business unit created the “Active Client” logic and provided the DW team an additional extract file. Many person-hours were wasted on the data warehouse end developing defunct logic, and ultimately more hours were wasted integrating an additional extract (the problem addressed in the last section). This example highlights the concern that derived fields can significantly impact the scope of an engagement.

Project Checkpoints

It goes without saying that scope management is an ongoing process. The above “limits” should be used as a baseline for defining what will be included in the data warehouse at a technical level. This must be combined with the current practice of establishing project checkpoints at key milestones in the project as well. After requirements, for example, the process of establishing a “scope understanding” with the client must be continued. The reasons that this is necessary are common to the management of any project.

Summary

Defining the scope of a data warehouse at the very beginning of a project is a difficult task. It is important to establish different levels of requirements to accurately manage scope during the life of the engagement. The following levels of scope management are recommended:

1. Establish a set number of business questions that will be answered by the warehouse. This is crucial to help the client understand what they are buying and instrumental in the design of the warehouse itself.
2. Establish the specific source systems that will be included in the warehouse. If this is not known at the onset of the project, establish a maximum number of source systems that can be included.
3. Establish a set number of extracts that will be provided from each source system. Ideally, you want only one extract per anticipated “subject area” from each system. Determining the optimal number of extracts is the most important tool for maintaining the scope of the engagement after it is underway.
 - a. (Optional) Establish a set number of star-schemas that will be provided in the warehouse. This is a more technical (and less effective) solution to the “subject area” issue, but it does provide a negotiation tool to revamp some of the business questions to answer a similar question that may be much less problematic.
4. Require all intra-department derived fields to be provided in the extracts. Business units know their own data much better than the data warehouse team. They should do the logic to define those fields for the warehouse.