Six Key Decisions for ETL Architectures

kimballgroup.com/2009/10/six-key-decisions-for-etl-architectures/

By Bob Becker 10/9/2009

This article describes six key decisions that must be made while crafting the ETL architecture for a dimensional data warehouse. These decisions have significant impacts on the upfront and ongoing cost and complexity of the ETL solution and, ultimately, on the success of the overall BI/DW solution. Read on for Kimball Group's advice on making the right choices.

1. Should We Use an ETL Tool?

One of the earliest and most fundamental decisions you must make is whether to hand code your ETL system or use a vendor-supplied package. Technical issues and license costs aside, you shouldn't go off in a direction that your employees and managers find unfamiliar without seriously considering the decision's long-term implications. This decision will have a major impact on the ETL environment, driving staffing decisions, design approaches, metadata strategies, and implementation timelines for a long time.

In today's environment, most organizations should use a vendor-supplied ETL tool as a general rule. However, this decision must be made on the basis of available resources to build and manage the system. ETL tools are really system building environments that use icons, arrows and properties to build the ETL solution rather than writing code. Be careful; if your proposed ETL development team is comprised of a number of old-school hand coders, they might not adapt well to an ETL tool. For this reason alone, some organizations find that custom ETL development is still a reasonable solution.

If you decide to use an ETL tool, don't expect a huge payback in your first iteration. The advantages will become more apparent as you traverse additional iterations and begin to leverage the development advantages of using a tool during subsequent implementations. You'll also experience the benefits of enhanced maintenance capabilities, more complete documentation and improved metadata support over time.

2. Where and how should data integration take place?

Data integration is a huge topic for IT because, ultimately, it aims to make all systems work together seamlessly. The "360 degree view of the enterprise" is a commonly discussed goal that really means data integration. In many cases, serious data integration should take place among an organization's primary transaction systems before data arrives at the data warehouse. However, rather than dealing with integration in the operational environment, these requirements are often pushed back to the data warehouse and the ETL system.

Most of us understand the core concept that integration means making disparate databases function together in a useful way. We all know we need it; we just don't have a clear idea of how to break it down into manageable pieces. Does integration mean that all parties across large organizations agree on every data element or only on some data elements? This is the crux of the decision that must be made. At what level in the data does business management agree/insist integration will occur? Are they willing to establish common definitions across organizations and to abide by those definitions?

Fundamentally, integration means reaching agreement on the meaning of data from the perspective of two or more databases. With integration, the results of two databases can be combined into a single data warehouse analysis.

Without such an accord, the databases will remain isolated stovepipes that can't be linked in an application. For the context of our ETL environment, data integration takes the form of conforming dimensions and conforming facts in the data warehouse. Conforming dimensions means establishing common dimensional attributes across separated fact tables so that "drill across" reports can be generated using these attributes. Conforming facts means making agreements on common business metrics such as key performance indicators (KPIs) across separated databases so that these numbers can be compared mathematically for calculating differences and ratios.

3. Which change data capture mechanisms should we choose?

During the data warehouse's initial historic load, capturing source data content changes is not important since you are loading all data from a point in time forward. However, most data warehouse tables are so large that they cannot be refreshed during every ETL cycle. You must have a capability to transfer only the relevant changes to the source data since the last update. Isolating the latest source data is called change data capture (CDC). The idea behind change data capture is simple enough: just transfer the data that has been changed since the last load.

But building a good change data capture system is not as easy as it sounds. Realize that the mechanism selected must be absolutely fool proof — all changed data must be identified. Finding the most comprehensive strategy can be elusive; many times updates to source system tables can occur outside the application itself. A mistake here will result in inconsistent results that can't be easily explained; often it takes significant data reconciliation to identify the culprit. Problems can be a very costly in terms of rework — not to mention embarrassing. In short, capturing data changes is far from a trivial task, and you must clearly understand the source data systems. This knowledge will help the ETL team evaluate data sources, identify change data capture problems and determine the most appropriate strategy.

4. When should we stage the data?

In today's data warehousing environment, it's quite possible for ETL tools to establish a direct connection to the source database, extract and stream the data through the ETL tool to apply any required transformation in memory, and finally write it, only once, into the target data warehouse table. From a performance viewpoint, this is a great capability as writes, especially logged writes into the RDBMS, are very expensive; it's a good design goal to minimize them. However, despite the performance advantages, this may not be the best approach. There are several reasons an organization might decide to physically stage the data (i.e., write it to disk) during the ETL process:

- The most appropriate CDC method requires a compare of the current copy of the source table to the prior copy of the same table.
- The organization has elected to stage the data immediately after extract for archival purposes — possibly to meet compliance and audit requirements.
- A recovery/restart point is desired in the event the ETL job fails in midstream — potentially due to a break in the connection between the source and ETL environment.
- Long running ETL processes may open a connection to the source system that create problems with database locks and that stresses the transaction system.

5. Where should we correct data?

Business users are aware that data quality is a serious and expensive problem. Thus most organizations are likely to support initiatives to improve data quality. But most users probably have no idea where data quality problems originate or what should be done to improve data quality. They may think that data quality is a simple execution problem for the ETL team. In this environment, the ETL team needs to be agile and proactive: data quality cannot be improved by ETL alone. The key is for the ETL team to partner with the business and the IT teams that support the

source systems.

The key decision is where to correct the data. Clearly, the best solution is to have the data captured accurately in the first place. Of course, this isn't always the case, but nonetheless, in most cases, the data should be corrected back in the source system. Unfortunately, it is inevitable that poor quality data will reach the ETL system. In this event, there are three choices:

1) halting the entire load process

2) sending the offending record(s) to a suspense file for later processing

3) merely tagging the data and passing it through

The third choice is by far the best choice, whenever possible. Halting the process is obviously a pain because it requires manual intervention to diagnose the problem, restart or resume the job, or abort completely. Sending records to a suspense file is often a poor solution because it is not clear when or if these records will be fixed and reintroduced to the pipeline. Until the records are restored to the data flow, the overall integrity of the database is questionable because records are missing. I recommend not using the suspense file for minor data transgressions. The third option of tagging the data with the error condition often works well. Bad fact table data can be tagged with an audit dimension that describes the overall data quality condition of the offending fact row. Bad dimension data can also be tagged using the audit dimension or, in the case of missing or garbage data, can be tagged with unique error values in the field itself. From this point, data quality reporting capabilities can flag the offending fact and dimension rows, indicating a need for resolution and ideally repair of the date in the source system.

6. How quickly must source data be available via the DW/BI system?

Data latency describes how quickly source system data must be delivered to the business users via the DW/BI system. Data latency obviously has a huge effect on the costs and complexity of your ETL environment. Clever processing algorithms, parallelization, and potent hardware can speed up traditional batch-oriented data flows. But at some point, if the data latency requirement is sufficiently urgent, the ETL system's architecture must convert from batch to streaming oriented. This switch isn't a gradual or evolutionary change; it's a major paradigm shift in which almost every step of the data delivery pipeline must be re-implemented.

Typically, ETL streams for most organizations require a data latency which matches the natural rhythm of the business. We find in most organizations this typically results in daily updates for most ETL streams and weekly or monthly updates for other ETL streams. However, in some circumstances, more frequent updates or even real time updates best suit the rhythm of the business. The key is to recognize that only a handful of business processes within any organization are appropriate for real time updating. There's no compelling reason to convert all ETL processing to real time. The rhythm of most business processes simply doesn't demand real time treatment.

Be careful: asking business users if they want "real time" delivery of data is an open invitation for trouble. Of course, most business users will respond positively to having data updated more frequently, regardless of whether they understand the impact of their request. Clearly this kind of data latency requirement could be dangerous. We recommend dividing the real time challenge into three categories: daily, frequently, and instantaneous. You need to get your end users to describe their data latency requirements in similar terms and then design your ETL solution appropriately to support each set of requirements:

- Instantaneous means that the data visible on the end user's screen represents the true state of the source transaction system at every instant. When the source system status changes, the online screen must also respond instantly.
- Frequently means that the data visible to the end user is updated many times per day but is not guaranteed to be the absolute current truth as of this instant.

- Daily means that the data visible on the screen is valid as of a batch file download or reconciliation from the source system at the end of the previous working day.

In this article we have discussed a number of key decisions that must be evaluated during the creation of an ETL architecture to support a dimensional data warehouse. There is seldom a single correct choice for any of these decisions, as always, the correct choice should be driven by the unique requirements and characteristics of your organization.