



"turning data into dollars"

Tom's Ten Data Tips – December 2008

ETL

ETL stands for Extract, Transform, Load. This process merges and integrates information from source systems in the data warehouse (DWH). Authorities like Gartner, Kimball or Inmon warn us that 60-90% of DWH development budgets are typically spent on ETL. Major project risks reside here, in particular when sources prove unreliable and poorly documented.

Uniquely identifying customers is essential to arrive at a "single version of the truth" that results in a reliable 360° view of the customer (sometimes also called unified customer view, UCV). ETL programmers face intellectually challenging problems here, where tradeoffs need to be made in light of broader business objectives and architectural considerations.

1. Manual Coding Is Passé

Although it is still used frightfully often, the data warehouse (DWH) community has converged on the conclusion that doing ETL by means of manual coding is not the best way to go. It is fraught with problems like maintainability of code and (lack of) connectivity with meta data. As a result, the TCO over the lifetime of a DWH strongly suggests using "a" professional ETL tool. The last thing you want is ETL code that can only be updated by the initial programmer (*even* if he's still on your payroll). And likewise, although no common meta data standards have been set, the chance of realizing reasonable connectivity with other platforms decreases exponentially unless some kind of "structure" is enforced through an ETL application.

2. Use CDC For Large Files, Wholesale Dumps For Small Files

The two most common mechanisms for transferring data between sources and the staging area ("loading dock" at entrance of DWH) are either wholesale copying of the source, or "merely" transferring the changes since the previous load (CDC for Change Data Capture). Both have pros and cons. The advantages of wholesale copies are primarily greater (process) independence between source system(s) and the DWH, greater opportunity for quality controls, and the ability for the source to schedule processing during off-peak hours. The clear

disadvantage is that the size of files to be transferred can become prohibitive in terms of bandwidth as well as processing for (very) large dumps.

Some architects prefer “only” copying all the changes since the previous load (CDC). This clearly (significantly) decreases the volume of data to be moved. If the (RDBMS) log-files cannot be used for this, proprietary code needs to be written for this, that perforce must run exactly during peak hours of load of the source system (when the database changes occur). Also, the responsibility to “get this right”, *and* make sure it stays up to date in the face of source system changes lies with the operational system. Hence the somewhat trickier process intertwinement. Also, the mechanisms for assessing (ongoing) data quality reside mainly with the source, another drawback for the DWH. Tip# 3 & 4 are the corollary to this tip.

3. Offload Flat Files For Process Independence From Operations

This is the process of choice for the initial load of the DWH, and it has some advantages afterwards when the DWH goes in production. Although it consumes more resources than a CDC or delta-file (see tip# 2), the wholesale dump can be scheduled at any convenient time for the source to minimize the burden. Often there are efficient (sequential) utilities already available for this, requiring minimal effort and development. After the offload, there is no interference nor dependence of the source system and the DWH.

The one bit of dependence that remains in this model is the *timing* of the dump. Because data begins to age as soon as it is placed in the staging area, and no direct connection with the source may be available, time stamping of the dump is crucial to get the time dimension right in the DWH.

4. CDC (Delta-Files) Saves Time Downstream

Change Data Capture (CDC) implies that *only* the changes since the last load are moved to the DWH. There could have been multiple insertions, deletions and changes but only the delta between the last load and current status are recorded and moved. The question is: how are these identified? This is non-trivial, and there can be multiple (concurrent) approaches. Updates can have timestamps which may be selected within the appropriate time range. Sometimes a proprietary “audit file” in the source system captures changes automatically. Special purpose code is sometimes developed for the DWH, and

sometimes database logging (peripheral recording of “activities” in the database) can be turned on that will allow changes to be traced.

Although in many (most) of these variations additional burden during anticipated peak load is added, the great benefit is that only the minimally required volume of data is moved. That feature *alone* can make it the only feasible solution, in particular when bandwidth is restricted, or when the size of the entire file is prohibitive to make it part of the ongoing ETL process.

5. ETL Becomes ELT In Data Vault

Data Vault is a 2nd Generation DWH modeling architecture that has arisen as a response to ever growing changes in data feeds and increased pressure to deliver value from data faster. As the time available to run through loading cycles keeps shrinking, a robust architecture requires that “application of business rules to data transformations must be deferred and moved downstream (Linstedt et al, 2008).”

As a result of this, a lot of (very) poor quality data is bound to enter the historical data store. In part this is necessary to ensure traceability and auditability, and to let the business maintain ownership of their own data feeds. Transformations and cleaning will have to wait until the data moves from the Enterprise Data Warehouse into a data mart.

6. Include Audit Dimensions

When disparate sources are merged in the DWH, discrepancies need to be resolved. It will occur at times that information from different sources doesn’t fully “match.” By forcing these sources into a common denominator, we say we make dimensions “conform.” By this we mean establishing common dimensional attributes, which are often textual labels (like M/F) or standard units of measurement. By the same token facts are conformed by agreeing on common business metrics like key performance indicators (KPI’s) so that numbers can be compared across databases.

What is crucial here is that differences that exist in underlying systems are recorded in so-called “audit dimensions.” That way a running count of all differences between source systems is captured for monitoring and future reference. In the central DWH “one version of the truth” is established on the bases of predetermined business rules to deal with discrepancies.

7. Profile Sources *Before* The ETL stage

It is good practice to do data profiling of source systems prior to beginning the ETL phase, in fact prior to deciding on development of the DWH altogether! Data profiling consists of tallying frequencies for categorical variables, determining ranges and distributions of continuous (real number) variables, but also checking cardinality of target key variables and referential integrity. Are all anticipated key variables really unique in practice?

A system that perfectly suits the needs of an operational process (in support of customer dialogue, production planning, order taking etc.) may not be usable for the DWH. Fields that the DWH team knows are available, and were hoping to use can prove to be highly unreliable. The reason being that they might only serve peripheral needs for the operation and may therefore be too incomplete for DWH purposes. Better to find this out sooner rather than later! This way you can prepare business sponsors for the kind of effort required in the ETL project phase, or possibly the need to install some organizational (AO) change so that the sources *can* be relied on for the DWH.

8. Establish Business Rules For ETL Data Quality Failure

Much has been written about variable data quality. Despite our best efforts to resolve data quality issues as far upstream in source systems as possible, we still need to be prepared for (sudden) drops in quality. When certain thresholds for non-quality are reached, you should determine in advance how to proceed. Should the loading process be stopped and data feeds returned to the source? Or in case of "minor" deviations, can the data be loaded and the quality issues 'merely' reported to the source? Thresholds for these choices need to be considered, and decided on when a DWH goes in production mode.

9. Provide Data Lineage To Instill Trust

In most data warehouses, one and the same field in an end user report could have originated from multiple source systems. For instance, when you merge personal data across systems, is there a "master source?" A so-called system-of-record is supposed to be the most reliable originating source and its content takes precedence over other sources when they differ.

Originating sources and business rules that were applied for mapping data should be documented in the data lineage meta data. That way, end users can see which originating source was used to generate

individual fields in the DWH. They will get more sense of “ownership” of their data which instills trust, a major factor in stimulating adoption of the system.

10. Good ETL Architecture Combines “Read” Approaches

Given that data warehousing is ‘merely’ a secondary activity, it should *support*, not *hinder* primary value creating business processes (we sometimes forget this). A robust loading process decouples the DWH needs from demands that operational DBA’s might have. OLTP (On-Line Transaction Processing) systems are designed to get data *in* quickly, a DSS (Decision Support System) like a DWH is designed to get data *out*. These needs often don’t match.

Minimal disruption of primary processes comes at a premium. And although scalability requires foresight sometimes, from an investment perspective it is usually better to defer hardware purchases (and commitments) until requirements have become clearer, and value from the DWH has materialized. This can be a tricky tightrope to walk which calls for oversight, business alignment, and adaptive development (see also the December 2008 newsletter on Agile Software Development methods).