

Monitoring Data Quality in Batch Feeds

In the old days, when Roman legions wanted to sack a fortified city, they hurled heavy stones at its walls, day after day. Not many walls could withstand such an assault. In the modern world, the databases suffer the same unrelenting onslaught of batch feeds. While real-time data integration has become more common in recent years, the majority of the interfaces will always utilize batch feeds. Each batch carries large volumes of data, and any problem in it causes great havoc further magnified by future feeds.

The batch feeds are especially dangerous because newly arrived records do not sit quietly. The incoming transactions usually trigger immediate processing in the target database, creating more and more errors in an avalanche of bad data. The cost of a single bad record can run into thousands of dollars. It is hard to even visualize the destructive power of a batch feed full of erroneous data.

So why do the well-tested batch feed programs falter? The source system that originates the batch feed is subject to frequent structural changes, updates, and upgrades. Testing the impact of these changes on the data feeds to multiple independent downstream databases is a difficult and often impractical step. Lack of regression testing and quality assurance inevitably leads to numerous data problems with batch feeds any time the source system is modified – which is all of the time!

The solution to the problem is to design programs operating between the source and target databases, which are entrusted with the task of analyzing the data before it is loaded and processed. The objective of interface monitoring is to prevent the errors from getting into the target database, or at least to identify problems as early as possible to minimize the damage they inflict.

In general, these monitors fall into two categories: individual data monitors and aggregate monitors. Individual data monitors use data quality rules to test data accuracy and integrity. Simple screens can validate the data inside a single batch and are trivial to implement, but they find few errors. Advanced monitors that compare data across batches or against target database identify more problems, but they require extensive development and processing.

Aggregate monitors search for unexpected changes in batch interfaces. They compare various aggregate attribute characteristics (such as counts of attribute values) from batch to batch, relying on the fact that such aggregate characteristics change little from batch to batch or follow predictable patterns. A value outside of the reasonably expected range indicates a potential problem.

Consider, for example, a batch feed with new HR transactions. Among other items, it contains basic indicative data about all new hires. We can calculate the number of all new hires shown in each batch as well as the percentage of those among them whose date of birth falls in January. Since there is no reason for a new hired person to have date of birth in January more or less often than in any other month, the proportion of new hires born in January is expected to stay around $8\frac{1}{3}$ percent. A large deviation from this value (say below 6% or above 10%) is indicative

of a problem. For instance, such deviation might occur if a number of new hires are missing date of birth, and a default value of 1/1/1930 is entered instead.

This example offers a blueprint for monitoring any unexpected changes in batch feeds. For many aggregate metrics, the sequence of values from different batches can be viewed as a time series. More importantly, many such metrics either change little over time or follow simple predictable patterns. ***Time series monitor*** is a filter that predicts reasonable range of values for an aggregated metrics in the batch and tests actual value against it. The monitor raises a “red flag” if actual value falls outside predicted range.

First step toward implementation of a time series monitor is to select aggregate characteristics for monitoring. The most basic metrics is the count (or percentage) of particular values of an attribute. For any attribute with discrete set of values, a monitor can be implemented for each value. The list of such values can be obtained through data profiling. Also, for all attributes the count of missing values should be monitored. For attributes with non-discrete values, such as amounts and dates, we cannot monitor all distinct values. Instead, we can use various aggregate or sample statistics. Count of values falling into a specific range can be easily monitored. For instance, an effective technique for date/time attributes is to use count of records with values falling in a particular calendar month or another recurring (but meaningless!) time period. Mean value and standard deviation should also be monitored. These are very easy to calculate and typically follow well-defined patterns of change (or simply remain static). Finally, more sophisticated statistics can be monitored to ensure that the overall distribution of values of an attribute does not change much from batch to batch.

The second step is to choose the model for predicting the reasonable range of values in the future batches. ***Heuristic monitors*** use simple “common sense” methods to set reasonable range boundaries. For instance, we can assume that a metrics would have the same constant value in all batches, save for the slight deviations due to random unexpected factors. We can further assume that a deviation of more than 20% is suspicious. The greatest advantage of heuristic monitors is their simplicity and ease of implementation.

Statistical time series monitors are the more sophisticated alternative to heuristic monitors. The idea behind them is to assume a strict mathematical model for the dynamics of the aggregate characteristics between batches. The advantage of statistical monitors is that the reasonable ranges of values are built based on the actual data rather than simplifying subjective assumptions. As a result, the monitors are more precise and give a better chance of identifying errors without raising too many false alarms. The main drawback of statistical monitors is that they require a history of at least 8-12 feeds for which the metrics had been calculated. Also, these monitors are more difficult to implement and require extensive knowledge of statistical algorithms or access to statistical software.

The next step is to implement the monitor in an automated computer program. It involves setting up a program to calculate the reasonable range for the metrics value in the next batch; setting up a program to calculate the actual metrics value in the next batch; defining actions that will be taken if actual value falls out of range; and designing the data storage and desired monitor performance reports.

Once the monitor is in place, it will be used every time the new batch arrives. From time to time, the monitor will raise a “red flag” indicating an abnormality in the data. Research will show whether this is a true problem or a “false alarm”. Based on the experience we can fine-tune and improve the monitor.

Aggregate monitors are easy to implement, though they require some special knowledge in data profiling and statistical time series analysis. They can be largely automated and will catch all unexpected changes in the batch interfaces. When implemented systematically, aggregate monitors form impenetrable walls that offer invaluable data quality protection against even the most destructive batch feeds.