An Architecture for Data Quality

A Kimball Group White Paper

By Ralph Kimball

KIMBALL GROUP
Consulting | Kimball University

Table of Contents

Executive Summary

In this white paper, we propose a comprehensive architecture for capturing data quality events, as well as measuring and ultimately controlling data quality in the data warehouse. This scalable architecture can be added to existing data warehouse and data integration environments with minimal impact and relatively little upfront investment. Using this architecture, it is even possible to progress systematically toward a Six Sigma level of quality management. This design is in response to the current lack of a published, coherent architecture for addressing data quality issues.

About the Author

Ralph Kimball founded the Kimball Group. Since the mid 1980s, he has been the data warehouse/business intelligence (DW/BI) industry's thought leader on the dimensional approach and trained more than 10,000 IT professionals. Prior to working at Metaphor and founding Red Brick Systems, Ralph co-invented the Star workstation at Xerox's Palo Alto Research Center (PARC). Ralph has his Ph.D. in Electrical Engineering from Stanford University.

The Kimball Group is the source for dimensional DW/BI consulting and education, consistent with our best-selling *Toolkit* book series, Design Tips, and award-winning articles. Visit www.kimballgroup.com for more information.

A Growing Awareness for Data Quality

Three powerful forces have converged to put data quality concerns near the top of the list for organization executives. First, the long term cultural trend that says "if only I could see the data then I could manage my business better" continues to grow, until today in 2007, most knowledge workers believe instinctively that data is a crucial requirement to perform their jobs. Second, most organizations understand that they are profoundly distributed, often around the world, and that effectively integrating myriad disparate data sources is required. And third, the sharply increased demands for compliance mean that careless handling of data is not going to be overlooked or excused.

These powerful converging forces illuminate data quality problems in a harsh light. Fortunately, the big pressures are coming from the business users, not just from IT. The business users have become aware that data quality is a serious and expensive problem. Thus the organization is more likely to support initiatives to improve data quality. But most business users probably have no idea where data quality problems originate or what an organization can do to improve data quality. They may think that data quality is a simple execution problem in IT. In this environment IT needs to be agile and proactive: data quality cannot be improved by IT alone. An even more extreme view says that data quality has almost nothing to do with IT.

Where Do Data Quality Problems Originate?

It is tempting to blame the original source of data for any and all errors that appear downstream. If only the data entry clerk were more careful, and REALLY cared! We are only slightly more forgiving of typing-challenged salespeople who enter customer and product information into their order forms. Perhaps we can fix data quality problems by imposing better constraints on the data entry user interfaces. This approach provides a hint of how to think about fixing data quality, but we must take a much larger view before pouncing on technical solutions. At a large retail bank we worked with, the social security number fields for customers were often blank or filled with garbage. Someone came up with the brilliant idea to require input in the 999-99-9999 format, and to cleverly disallow nonsensical entries such as all 9's. What happened? The data entry clerks were forced to supply valid social security numbers in order to progress to the next screen, so when they didn't have the customer's number, they typed in their own!

Michael Hammer, in his revolutionary book, *Reengineering the Corporation* (HarperBusiness 1994), struck to the heart of the data quality problem with a brilliant insight that I have carried with me throughout my career. Paraphrasing Hammer: "Seemingly small data quality issues are, in reality, important indications of broken business processes." Not only does this insight correctly focus our attention on the source of data quality problems, but it shows us the way to the solution.

Establish a Quality Culture and Reengineer the Processes

Technical attempts to address data quality will not function unless they are part of an overall quality culture that must come from the very top of an organization. The famous Japanese car manufacturing quality attitude permeates every level of those organizations; quality is embraced enthusiastically by all levels, from the CEO down to the assembly line worker. To cast this in a data context, imagine a company like a large drug store chain, where a team of buyers contracts with thousands of suppliers to provide the drug store inventory. The buyers have assistants, whose job it is to enter the detailed descriptions of everything purchased by the buyers. These descriptions contain dozens of attributes. But the problem is that the assistants have a deadly job. They are judged on how many items they enter per hour. The assistants have almost no visibility of who uses their data. Occasionally the assistants are scolded for obvious errors. But more insidiously, the data given to the assistants is itself incomplete and unreliable. For example, there are no formal standards for toxicity ratings, so there is significant variation over time and across product categories for this attribute. How does the drug store improve data quality? Here is a nine step template, not only for the drug store, but for any organization addressing data quality:

1. Declare a high level commitment to a data quality culture.
2. Drive process reengineering at executive levels.
3. Spend money to improve the data entry environment.
4. Spend money to improve application integration.
5. Spend money to change how processes work.
6. Promote end-to-end team awareness.
7. Promote interdepartmental cooperation.
8. Publicly celebrate data quality excellence.
9. Continuously measure and improve data quality.

At the drug store, money needs to be spent to improve the data entry system so that it provides the content and choices needed by the buyers' assistants. The company's executives need to assure the assistants that their work is very important and their efforts affect many decision makers in a positive way. Diligent efforts by the assistants should be publicly praised and rewarded. And end-to-end team awareness and appreciation of the value of data quality is the final goal.

The Role of the Data Steward

The data steward is responsible for driving organizational agreement on definitions, business rules, and permissible domain values for the data warehouse data, and then publishing and reinforcing these definitions and rules. Historically, this role was referred to as data administration, a function within the IT organization. However, it's much better if the data steward role is staffed by the subject matter experts from the business community.

Clearly, this is a politically challenging role. Stewards must be well respected leaders, committed to working through the inevitable cross-functional issues, and supported by senior management, especially when organizational compromise is required.

Sometimes the data stewards are supported by quality assurance (QA) analysts who ensure that the data loaded into the warehouse is accurate and complete. They identify potential data errors and drive them to resolution. The QA analyst is sometimes also responsible for verifying the business integrity of the business intelligence (BI) applications. This role is typically staffed from within the business community, often with resources who straddle the business and IT organizations. Once a data error has been identified by the QA analyst, then the errors must be corrected at the source, fixed in the extract, transform, and load (ETL) steps, or tagged and passed through the ETL steps. Remember, data quality errors are indicators of broken business processes and often require executive support to correct. Relatively few data errors can be fixed in the data warehouse.

The QA analyst has a significant workload during the initial data load to ensure that the ETL system is working properly. And given the need for ongoing data verification, the QA analyst role does not end once the warehouse is put into production.

How Can Technology Address Data Quality?

Once the executive support and the organizational framework are ready, then specific technical solutions are appropriate. The rest of this article describes how to marshal technology to support data quality. Our goals for the technology include:

- Early diagnosis and triage of data quality issues.
- Specific demands on source systems and integration efforts to supply better data.
- Specific descriptions of data errors expected to be encountered in ETL.
- Framework for capturing all data quality errors.
- Framework for precisely measuring data quality metrics over time.
- Quality confidence metrics attached to final data.

Importance of Data Profiling

Data profiling is the systematic analysis of data to describe its content, consistency, and structure. In some sense, any time you perform a SELECT DISTINCT investigative query on a database field you are doing data profiling. Today there are a variety of tools purpose-built to do powerful data profiling. It probably pays to invest in a tool rather than build your own, because the tools allow many data relationships to be explored easily with simple user interface gestures. You will be much more productive in the data profiling stages of your project using a tool rather than hand coding all your data content questions.

Data profiling plays distinct strategic and tactical roles. At the beginning of a data warehouse project, as soon as a candidate data source is identified, a quick data profiling assessment should be made to provide a ”go/no go” decision about proceeding with the project. Ideally this strategic assessment should occur with a day or two of identifying a candidate data source! Early disqualification of a data source is a responsible step that will earn you respect from the rest of the team, even if it is bad news. A very late revelation that the data source cannot support the mission can be a fatal career outcome for you if this realization occurs many months into a project.

Negotiate Data Quality with the Source Systems

Data profiling is a process of on-going discovery. As far as the data warehouse is concerned, the ideal place to address data quality issues is at the source. However, the data warehouse team needs to proceed cautiously in order to get the best possible response from the owners of the source systems. Before even starting, a culture needs to be established by senior management that encourages finding data quality issues and fixing them all as part of an enterprise-wide team effort. We can take a page from the experience of car manufacturers who have found it very successful to give rewards to workers who find quality problems and propose solutions.

The data warehouse team needs to be sensitive to the impact that process re-engineering may have on the source systems, both at the technical implementation level and the operational level. Ideally, a single list of showstopper quality problems can be developed in the initial data profiling effort and dealt with by a joint team consisting of DW/BI and source system personnel and chaired by a senior executive. Again, ideally the DW/BI team does not return repeatedly with more demands for significant changes in the source systems. But we rarely live in an ideal world.

Establish a Feedback Path to the Source

If the enterprise has a master data management (MDM) system that assembles master copies of important entities such as customer, product, and location, then presumably source systems around the enterprise are committed to using “gold” copies of these entities. Perhaps updated customer lists are downloaded as batch files by a source system from the MDM system, or individual customer profiles are fetched on request from the MDM system through a service oriented architecture (SOA) interface. Either of these architectures is a great outcome for the data warehouse because it means not only that data quality is recognized and valued, but all those tasks are being already handled by someone else!

But it is reasonably likely that the data warehouse itself will gradually morph into an MDM system, perhaps without the enterprise being aware of the implications. In *The Data Warehouse ETL Toolkit* (Wiley 2004), we describe the steps of cleaning, de-duplicating, surviving and conforming multiple data feeds from different source systems to develop the same kind of gold masters of entities that MDM systems produce. If that has happened, the data warehouse team should get senior management’s commitment to take the final step of feeding the cleaned data back to the source system, to avoid the Sisyphean fate of cleaning the same dirty data over and over.

Quality Screens: The Heart of the Architecture

The heart of the data warehouse quality architecture is a set of *quality screens* that act as diagnostic filters in the back room ETL data flow pipelines. A quality screen is simply a test, implemented at any point in the ETL or data migration processes. If the test is successful, nothing happens and the screen has no side effects. But if the test fails, then every screen has two simple responsibilities:

- Drop an error event record into the *error event schema*, and
- Choose to halt the process, send the offending data into suspension, or merely tag the data.

Although all quality screens are architecturally similar, it is convenient to divide them into three types, in ascending order of scope. Here we follow the categorizations of data quality as defined by Jack Olson, in his seminal book, *Data Quality: The Accuracy Dimension* (Morgan Kaufman 2002):

- Column screens
- Structure screens
- Business rule screens

Column screens test the data within a single column. These are usually simple, somewhat obvious tests. Jack Olsen gives a number of useful examples of each kind of screen, which he calls rules, and which we repeat here with kind permission. His column screen examples include:

- Value must be not null.
- Value must be one character and from a finite fixed list.
- Value must be within a range.
- Value must fit a specific field pattern such as Z9999.
- Value must either be null or greater than 5 characters of free text.
- Value must not be in a specific exclusion list.
- Value must not fail spell checker.

Structure screens test the relationship of data across columns. Two or more fields may be tested to verify that they implement a hierarchy (e.g., a series of many-to-one relationships). Structure screens include testing foreign key/primary key relationships between fields in two tables, and also include testing whole blocks of fields to verify that they implement valid postal addresses.

Jack Olsen's examples include:

- A combination of fields must implement a primary key for the surrounding table.
- An inventory history part number must appear in the inventory master.
- All inventory parts must have at least one source.
- All suppliers must supply at least one part.
- A supplier may have no orders.

Business rule screens implement more complex tests that do not fit the simpler column or structure screen categories. For example, a customer profile may be tested for a complex time dependent business rule, such as requiring that a Lifetime Platinum frequent flyer has been a member for at least 5 years and has more than two million frequent flyer miles. Business rule screens also include *aggregate threshold* data quality checks, such as checking to see if a statistically improbable number of MRI examinations have been ordered for minor diagnoses like sprained elbow. In this case, the screen only throws an error after a threshold of such MRI exams is reached.

Jack Olsen further subdivides business rule screens into three subcategories, including simple data rules, complex data rules, and value rules.

Simple data rules:

- The quantity on order cannot be less than the minimum order quantity.
- If the number of orders placed is zero, the quantity ordered must also be zero.
- The date a supplier is established must not be later than the last order placed with this supplier. Both dates must not be later than the current date.

Complex data rules:

- Two sources for the same part number cannot have the same priority.
- There should be no outstanding orders for parts that are marked as *Do not order*.

Value rules:

- Compute the number of orders for the same part in each month period and verify that no part has more than two orders in a single month, because the inventory re-ordering algorithm is supposed to prohibit this from happening.
- Verify that the total number of orders for each major category of inventory does not vary month-to-month by more than 10%.

Error Event Schema

The error event schema is a centralized dimensional schema whose purpose is to record every error event thrown by a quality screen anywhere in the data warehouse ETL pipeline. Although we are focusing on data warehouse processing, this approach obviously can be used in general data integration (DI) applications, where data is being transferred between legacy applications. The error event schema is shown in Figure 1.

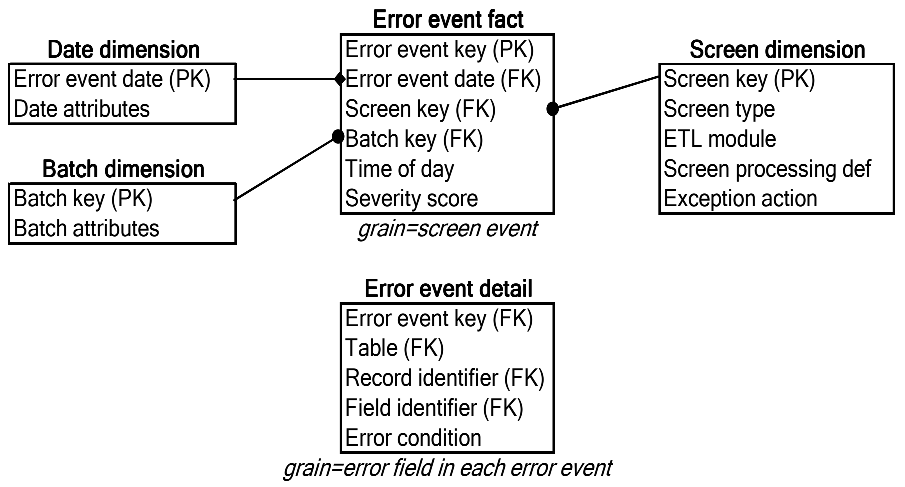


Figure 1. The Error Event Schema.

The main table is the error event fact table. Its grain is every screen event: an error thrown (produced) by a quality screen anywhere in the ETL or data migration system. Remember that the grain of a fact table is the physical description of why a fact table record exists. Thus every quality screen error produces exactly one record in this table, and every record in the table corresponds to an observed error.

The dimensions of the error event fact table include the calendar date of the error, the batch job in which the error occurred, and the screen which produced the error. The calendar date is not a minute and second time stamp of the error, but rather provides a way to constrain and summarize error events by the usual attributes of the calendar, such as weekday or last day of a fiscal period. The time-of-day fact is a full relational date-time stamp that specifies precisely when the error occurred. This format is useful for calculating the time interval between error events because you can take the difference between two date-time stamps to get the number of seconds separating events.

The batch dimension is an interesting dimension that contains as much data as possible describing the specific batch job in which the error occurred. The batch dimension can be generalized to be an individual processing step in cases where data is streamed, rather than batched. The information in the batch dimension comes from the ETL system workflow monitor, and could include:

- Batch schedule date-time stamp
- Actual batch starting and ending date-time stamps
- Total number of records processed in batch run
- Total number of screen tests performed in batch run
- Total number of errors encountered in batch run
- Database, processor, memory, and disk contention
- Maximum error severity score in batch run

The attributes in the batch dimension behave alternatively like dimensional attributes subject to constraining and grouping, or like numeric facts subject to summing and calculating. If this is the case, a separate batch measurement fact table can also be built.

The screen dimension identifies precisely what the screen criterion is and where the code for the screen resides. It also defines what to do when the screen throws an error. Screen dimension attributes include:

- Screen type: column, structure, or business rule. This could be augmented with a simple diagnostic assessment of the error, such as Incorrect, Ambiguous, Inconsistent, or Incomplete. This would allow the analyst to aggregate error events into interesting classifications.
- ETL module: one or more fields, probably describing a hierarchical location in the overall ETL system, indicating where the quality screen is embedded. The exact nature of these fields will depend on the architecture of your ETL system and if you are using a vendor supplied ETL tool.
- Screen processing definition: A simple and straightforward design would include a terse, free text description of the screen test as well as a field containing the exact file location of the screen code. In some systems, the screen code could actually be fetched at run time from this location.
- Exception action: halt the process, send the record to a suspense file, or merely tag the data. We discuss the impact of these actions in the next section. The exception action can be generalized to describe what notifications are sent when a process is halted; where the suspense file is located; and how the data is tagged if it is passed through the screen. We describe exactly how to tag data in the section on the audit dimension.
- Optional default severity score not shown in figure: the value of the severity score (assigned within a range from 0.0 to 1.0) normally used in the fact table when this error event occurs. Special business rules could change the severity score in the fact table at certain points in time, or if many such errors have been reported.

The main error event fact table contains a single part primary key shown as the error event key, a time of day field and a severity score, besides the foreign keys to the dimensions. The error event key, like the primary key of the dimension tables, is a surrogate key consisting of a simple integer assigned sequentially as records are added to the fact table. This key field is necessary in those situations where an enormous burst of error records is added to the error event fact table all at once. Hopefully this won't happen to you. This single field primary key is also helpful for the DBA who may need to call attention to a single record in this large fact table.

Near the end of this paper we discuss adapting the Six Sigma quality measuring methodology to data warehouse data. The fundamental goal of reaching the Six Sigma level is to achieve less than 3.4 errors per million opportunities. The batch dimension described above contains enough information to test for Six Sigma. In addition, the severity score in the error event fact table allows a more thoughtful weighting of the severity of the errors if the team is uncomfortable "being penalized" for all data quality errors in the same way. Since the severity score ranges from 0.0 to 1.0, these numbers can simply be added to get the weighted, as opposed to absolute, error total in a given batch run. You can consult with local Six Sigma methodologists to see if this weighted error reporting is controversial!

The error event schema includes a second *error event detail* fact table at a lower grain. Each record in this table identifies an individual field in a specific data record that participated in an error. Thus a complex structure or business rule error that triggers a single error event record in the higher level error event fact table may generate many records in this error event detail fact table. The two tables are tied together by the error event key, which is a foreign key in this lower grain table. In other words, there is a strict 1-to-many relationship between records in the parent error event fact table and the child error event detail fact table. The error event detail table identifies the table, record, field, and precise error condition, and likewise could optionally inherit the date, screen, and batch dimensions from the higher grain error event fact table. Thus a complete description of complex multi-field, multi-record errors is preserved by these tables The error event detail table could also contain a precise date-time stamp, to provide a full description of aggregate threshold error events where many records generate an error condition over a period of time.

We now appreciate that each quality screen has the responsibility for populating these tables at the time of an error.

The error event schema described in this section is at the heart of the ETL pipeline, and may be subject to intense bursts of record insertions when the pipeline is processing data. In many ways, these tables are like production tables in a transaction processing system. Complex queries on these tables should be prohibited during times when that activity would slow the ETL pipeline. Or a hybrid architecture might be used where individual screens simply write to log files, and the error event schema updating is taken off the critical path of ETL processing. Of course, that architecture might conflict with ETL managers' desires for real time reporting of errors!

In a large enterprise spanning many machines, many database instances, and multiple independent ETL pipelines, you should not try to build a single centralized error event set of tables. Rather, each ETL environment should have its own error event schema. This certainly simplifies the immediate collection and management of errors at a local level, but it adds an acceptable increased complexity of drilling across all the error event schema instances to see the enterprise view of data quality.

Responding to Quality Events

We have already remarked that each quality screen has to decide what happens when an error is thrown. The choices are: 1) halting the process; 2) sending the offending record(s) to a suspense file for later processing; and 3) merely tagging the data and passing it through to the next step in the pipeline. 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. We 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 the audit dimension described in the next section. 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.

Missing or obviously corrupt fact data can be handled with a least biased estimator: an artificially generated value that is a good estimate of what the missing data should have been. The audit dimension can adequately describe this condition so that business users are not misled. Most data warehouse people are uncomfortable with this approach because understandably they don't like making up the data. But consider the alternatives of dropping the activity record in question or replacing the corrupt value with NULL, or even worse, zero. In any of these cases, the total activity measure will be misleading and wrong. Using a least biased estimator brings the total activity counts and measures much closer to reality.

The Audit Dimension

The audit dimension is a normal dimension that is assembled in the back room by the ETL process for each fact table. A sample audit dimension attached to a shipments fact table is shown in Figure 2.

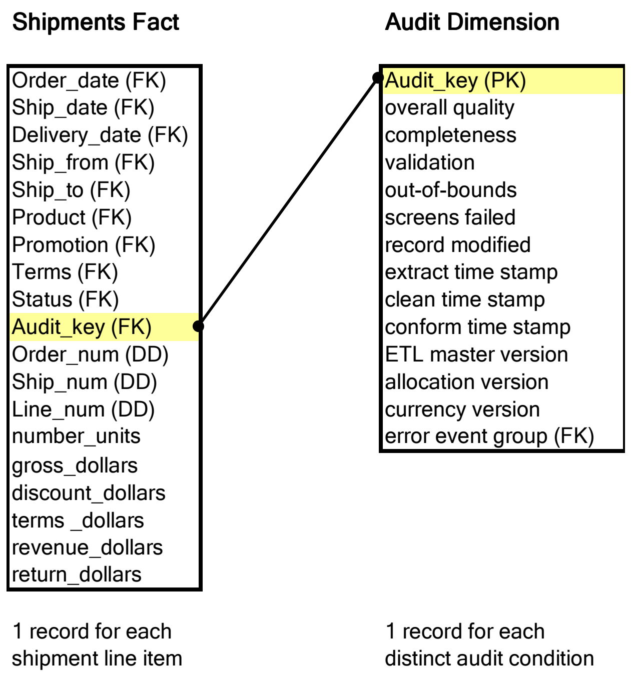


Figure 2. A Sample Audit Dimension Attached to a Shipments Fact Table.

In Figure 2, the shipments fact table contains a typically long list of dimensional foreign keys each labeled with FK, three degenerate dimensions labeled with DD, and six additive numeric facts. This style of fact table with its keys and other fields has been described many times in Kimball Group articles and books.

The audit dimension in Figure 2 contains typical metadata context recorded at the moment when a specific fact table record is created. One might say that we have elevated metadata to real data! The designer of the data quality system can include as much or as little metadata as is convenient to record at the time of an error. To visualize how audit dimension records are created, imagine that this shipments fact table is updated once per day from a batch file. Suppose that today we have a perfect run with no errors flagged. In this case we would generate only one audit dimension record and it would be attached to every fact record loaded today. All of the error conditions and version numbers would be the same for every record in this morning's load. Hence only one dimension record would be generated. This audit dimension record would be created during the final delivery step of the shipment line item fact table by looking up the required information in the error event schema.

Now let us relax the strong assumption of a perfect run. If we had some fact records whose discount dollars triggered an out-of-bounds error, then one more audit dimension record would be needed to handle this condition. In the next few paragraphs we discuss each of the fields in the audit dimension.

The overall quality attribute could be a text field with a small number of discrete values, such as Zero Defects, Minor, or Major. Additionally, the quality metric could include a computed numeric field equal to 1 minus the fraction of the number of defects found in this record compared to the number of screen tests performed. Remember that all of this information is available from the tables in the error event schema.

The completeness, validation, and out of bounds attributes in the audit dimension allow more precise descriptions of the confidence of the particular shipment line item fact table record. These kinds of measures are particularly appealing to business users because they can develop confidence in the validity of the data. Completeness refers to whether all of the facts in the record have been delivered. Validation refers to whether any facts in the record have triggered business rule violations (as opposed to column or structure rule violations). The out of bounds attribute describes whether any of the values in the record exceed the out of bounds threshold defined by the data quality screen. A more sophisticated version of this could include the maximum variance from the expected mean of any of the fact values in the record.

The screens failed attribute is a simple count of all screens that indicated an error for this fact record. The record modified attribute shows whether this record has ever been updated since its first creation. The fact table used in this example is an accumulating snapshot grain table, which is expected to undergo revisions to each record. If the fact table were of the transaction grain or periodic snapshot grain, then the record modified attribute would potentially indicate a much more serious Type 1 overwrite. In any of these cases, the record modified attribute could be augmented with a record updated time stamp that would need to have a default value for the case where no update had occurred. The three types of fact tables and the various slowly changing dimension (SCD) types have been described many times in the *Toolkit* series of books and Kimball Group articles.

The extract, clean and conform time stamps refer to when the overall batch processing finished that contained the fact record in question.

Finally, the three version numbers are examples of master metadata version numbers that hopefully you maintain to describe exactly which version of the processing logic you used when this fact record was created. These version numbers are enormously useful in financial auditing or compliance tracking situations. Suppose that the allocation logic that assigned cost data to various product categories was changed at some point during the fiscal year. The allocation version number could be used as a row header in a profitability report to show how much profit was reported under the old and new allocation schemes. See the next section where we show how to use the audit dimension.

The Audit Dimension Payoff

The power of the audit dimension becomes most apparent when the dimension attributes are used in a business user report as shown in Figure 3.

Normal Report

Product	Ship From	Qty Shipped	Revenue
Axon	East	1438	$235,000
Axon	West	2249	$480,000

Instrumented Report (add Out of Bounds Indicator to SELECT)

Product	Ship From	Out of Bounds Indicator	Qty Shipped	Revenue
Axon	East	Abnormal	14	$ 2,350
Axon	East	OK	1424	$ 232,650
Axon	West	Abnormal	675	$ 144,000
Axon	West	OK	1574	$ 336,000

Figure 3. Normal and Instrumented Reports Using the Audit Dimension.

The top report is a normal report showing sales of the Axon product in two geographic regions. The lower report is the same report with the out-of-bounds indicator added to the set of row headers with a simple user interface command. This produces an instant data quality assessment of the original report, and shows that the Axon West sales are suspect.

Advanced Design Issues

In this section, we describe further more technical extensions that leverage the audit dimension and error event data.

Using the Error Event Group

Figure 2 shows a foreign key in the audit dimension referring to an error event group. Figure 4 illustrates the error event group table for the fact table in our example.

Error Event Group (FK)
Fact Name
Quality Rating

Figure 4. The Error Event Group Table Corresponding to Figure 2.

The fact table we have been using as an example in this paper has six fact fields shown at the bottom of the table. In order to separately describe the data quality status of each field, we need a record in the error event group table for each fact in each distinct group of quality ratings. Thus in our example where we had a perfect run, we would need exactly one error event group that would contain six records. The six records would correspond to each field and a Normal rating. See the top table in Figure 5 where the surrogate error event group key equals 2973.

Key	Fact Name	Rating
2973	number_units	Normal
2973	gross_dollars	Normal
2973	discount_dollars	Normal
2973	terms_dollars	Normal
2973	revenue_dollars	Normal
2973	return_dollars	Normal

Key	Fact Name	Rating
3575	number_units	Normal
3575	gross_dollars	Normal
3575	discount_dollars	Out of Bounds
3575	terms_dollars	Normal
3575	revenue_dollars	Normal
3575	return_dollars	Normal

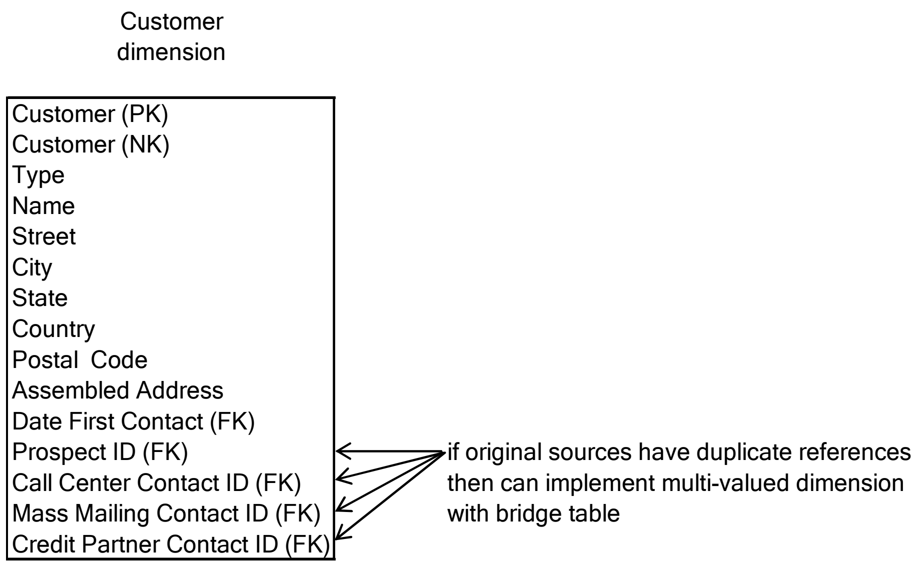
Figure 5. Error Event Group Table Records for Two Error Conditions.

If some other shipment fact records contained values where the discount_dollars triggered an out of bounds data quality warning, then the audit dimension for these fact records would contain the foreign key 3575 pointing to the records shown in the lower table in Figure 5.

The power of this event group table becomes apparent if the business user drags the Fact Name and the Rating into a report as row headers, much like our example in Figure 3. This would break out separate rows for each quality rating for that fact.

Propagating Error Tracking Back Through a De-Duplicating Step

A good ETL system will detect duplicate entries in major dimension, such as customer, and will combine the activity records for those duplicated entries under a single key, such as the customer key. This may be supported by the master data management (MDM) system. This de-duplicating step raises a problem because information is potentially lost when separate original natural keys are combined into one “super-natural” key. The solution, in this case, is to maintain back pointers to the original natural keys from the final customer dimension, for instance, as shown in Figure 6.



Customer dimension
Customer (PK)
Customer (NK)
Type
Name
Street
City
State
Country
Postal Code
Assembled Address
Date First Contact (FK)
Prospect ID (FK)
Call Center Contact ID (FK)
Mass Mailing Contact ID (FK)
Credit Partner Contact ID (FK)
if original sources have duplicate references then can implement multi-valued dimension with bridge table

Figure 6. Example Customer Dimension Showing Back Pointers to Original Natural Keys.

In Figure 6, the Customer primary key shown as the first field is the clean de-duplicated surrogate key that is the end result of the cleaning process. The last four fields in the table are literal natural keys coming from the original source systems that interact with the customer. In cases where an error is thrown by a quality screen testing a final fact table value, the analyst has the option of tracing back to the original raw input data through one of the back pointers, or conversely if the error refers to the original source data, the analyst can trace the impact of this error downstream past the de-duplication step.

Estimating Correct Measured Values from Prior History

Suppose we have a fact table that tracks daily sales in 600 stores, each of which has 30 departments. We therefore receive 18,000 sales numbers each day. This section describes a quick statistical check, based on calculating standard deviations, that allows us to judge each of the 18,000 incoming numbers for reasonableness and assign data confidence metrics to all of them in a single pass. We assume that this test is conducted by a data quality screen.

The technique described here lets us quickly update the statistical base of numbers to get ready for tomorrow's data load. Going back to the statistics course you took in college, remember that the standard deviation is the square root of the variance. The variance is the sum of the squares of the differences between each of the historical data points and the mean of the data points, divided by N-1, where N is the number of days of data. Unfortunately, this formulation could require us to look at the entire time history of sales, which, although possible, makes the computation unattractive in a fast-moving ETL environment. But if we have been keeping track of SUM SALES and SUM SQUARE SALES, we can write the variance as (1/(N-1))*(SUM SQUARE SALES - (1/N)*SUM SALES*SUM SALES). Check the algebra!

So if we abbreviate the above variance formula with VAR, our data validity check for all anomalous values more than 3 standard deviations from the expected mean looks like:

```
SELECT s.storename, p.departmentname, sum(f.sales)
FROM fact f, store s, product p, time t, accumulatingdept a

WHERE
(first, joins between tables... )
f.storekey = s.storekey and f.productkey = p.productkey and
f.timekey = t.timekey and s.storename = a.storename and
p.departmentname = a.departmentname and
```

(then, constrain the time to today to get the newly loaded data...)

```
t.full_date = #October 13, 2007# and
```

(finally, invoke the standard deviation constraint...)

```
HAVING ABS(sum(f.sales) - (1/a.N)*a.SUM_SALES) > 3*SQRT(a.VAR)
```

We expand VAR as in the previous explanation and use the a. prefix on N, SUM SALES and SUM SQUARE SALES. We have assumed that departments are groupings of products and hence are available as a rollup in the product dimension.

Every row returned from this query would provoke an entry in the error event fact table, where the diagnosis was "Abnormal Value."

Embellishments on this scheme could include running two queries: one for the sales MORE than three standard deviations above the mean and another for sales LESS than three standard deviations below the mean. Maybe there is a different explanation for these two situations. This would also get rid of the ABS function if your SQL doesn't like this in the HAVING clause. If you normally have significant daily fluctuations in sales (for example, Monday and Tuesday are very slow compared to Saturday), you could add a DAY OF WEEK to the accumulating department table and constrain to the appropriate day. In this way, you don't mix the normal daily fluctuations into our standard deviation test.

When you are done checking the input data with the preceding SELECT statement, you can update the existing SUM SALES and SUM SQUARE SALES just by adding today's sales and today's square of the sales, respectively, to these numbers in the accumulating department table. Then you are ready for tomorrow's data.

Conforming Audit Dimensions across Multiple ETL Environments

In a distributed enterprise data warehouse environment, with many separate ETL environments, there will be many versions of audit dimensions, with potentially different quality metrics. This is quite understandable since the fact tables across the enterprise will be very different and have legitimately different criteria for data quality. However, like any dimension appearing across the enterprise, a significant effort should be made by the joint ETL teams to define a core set of quality metrics that have the same meaning for every instance of the audit dimension. This core set of metrics then allows drilling across fact tables to build enterprise wide reports labeled by key quality metrics. From our example audit dimension shown in Figure 2, we would propose the overall quality, completeness, validity, and out of bounds attributes as subject to the conformed standards. This means that every audit dimension assign values to these attributes from the same standard set and with the same business rule meanings.

Six Sigma Data Quality

The data warehouse community can borrow some useful experience from the manufacturing community by adopting parts of their quality culture. In the manufacturing world, the Six Sigma level of quality is achieved when the number of defects falls below 3.4 defects per million opportunities. The error event fact table is the perfect foundation for making the same Six Sigma measurement for data quality. The defects are recorded in the error event schema and the opportunities are recorded in the organization’s workflow monitor tool that records total number of records processed in each job stream.

Building a Data Quality Architecture

The data quality architecture described in this article can be incrementally added to an existing data warehouse or data integration environment with very little disruption. Once the error event schema is established, the library of quality screens can grow indefinitely from a modest start. The screens need only obey the two simple requirements outlined earlier: logging each error into the error event schema, and determining the system response to the error condition. Error screens can be implemented in multiple technologies throughout the ETL pipeline, including standalone batch jobs as well as data flow modules embedded in professional ETL tools.

Even though this paper describes a comprehensive and complex architecture for data quality, it is possible to start simply and grow the capabilities of the system as the ETL team gains confidence. The types of errors thrown by the screens can be very simple, perhaps using only two or three levels of severity. Screens can be added incrementally, whenever the ETL team decides to expand the library of screens.

Of course, the error event schema provides a quantitative basis for managing data quality initiatives over time, since it is a time series by definition. The dimensionality of the error event data allows studying the evolution of data quality by source, software module, key performance indicator (KPI), and type of error.

Conclusion

The industry has talked about data quality endlessly, but there have been few unifying architectural principles. This paper describes an easily implemented, non-disruptive, scalable, and comprehensive foundation for capturing data quality events, as well as measuring and ultimately controlling data quality in the data warehouse.