CHAPTER 3

Data Quality

"Virtually everything in business today is an undifferentiated commodity, except how a company manages its information. How you manage information determines whether you win or lose."

—Bill Gates

Everybody wants better quality of data. Some organizations hope to improve data quality by moving data from legacy systems to enterprise resource planning (ERP) and customer relationship management (CRM) packages. Other organizations use data profiling or data cleansing tools to unearth dirty data, and then cleanse it with an extract/transform/load (ETL) tool for data warehouse (DW) applications. All of these technology-oriented data quality improvement efforts are commendable—and definitely a step in the right direction. However, technology solutions alone cannot eradicate the root causes of poor quality data because poor quality data is not as much an IT problem as it is a business problem.

Other enterprise-wide disciplines must be developed, taught, implemented, and enforced to improve data quality in a holistic, cross-organizational way. Because data quality improvement is a process and not an event, the following enterprise-wide disciplines should be phased in and improved upon over time:

- A stronger personal involvement by management
- High-level leadership for data quality
- New incentives

- New performance evaluation measures
- Data quality enforcement policies
- Data quality audits
- Additional training for data owners and data stewards about their responsibilities
- Data standardization rules
- Metadata and data inventory management techniques
- A common data-driven methodology

CURRENT STATE OF DATA QUALITY

We repeatedly run into a common example of data quality problems when trying to speak with a customer service representative (CSR) of a bank, credit card company, or telephone company. An automated voice response system prompts you to key in your account number before passing your call to a CSR. When a person finally answers the call, you are asked to repeat your account number because the system did not pass it along. Where did the keyed-in data go?

Another more serious data quality problem involves a report in 2003 about the federal General Accounting Office (GAO) not being able to tell how many H-1B visa holders worked in the U.S. The GAO was missing key data and its systems were not integrated. This presented a major challenge to the Department of Homeland Security, which tried to track all visa holders in the U.S.

According to Gartner, Inc., Fortune 1000 enterprises may lose more money in operational inefficiency due to data quality issues than they spend on data warehouse and CRM initiatives. In 2003, the Data Warehouse Institute (TDWI) estimated that data quality problems cost U.S. businesses $600 billion each year.

At an Information Quality Conference in 2002, a telecom company revealed that it recovered over $100 million in “scrap and rework” costs, a bank claimed to have recovered $60 million, and a government agency recovered $28.8 million on an initial investment of $3.75 million. Clearly, organizations and government are slowly realizing that data quality is not optional.

Many companies realize that they did not pay sufficient attention to data while developing systems during the last few decades. While delivery schedules have been shrinking, project scopes have been increasing, and companies have been struggling to implement applications in a timeframe that is acceptable to their business community. Because a day has only 24 hours, something has to give, and what usually gives is quality, especially data quality.

RECOGNIZING DIRTY DATA

When asked to define "data quality," people usually think of error-free data entry. It is true that sloppy data entry habits are often the culprit, but data quality is also affected by the way we store and manage data. For example, old file structures, such as flat files, did not have strong data typing rules, and it was common practice to use REDEFINE and OCCURS clauses with those structures. A REDEFINE clause allows you to change the data type of a data element or a group of data elements. For example, a character name field can be redefined and reused as a numeric amount field or a date field. An OCCURS clause allows you to define an array of repeating data elements. For example, an amount field can occur 1–12 times, if you were capturing monthly totals for January through December. Relational database management systems and the new generation of object-oriented programming practices no longer encourage such untidy data typing habits, but they do not provide any deterrence for other types of data abuse, such as some extensible markup language (XML) document type definition (DTD) usage that propagates into the relational databases. Many of the dirty data examples described in the following list can be found in relational databases as often as they can be found in flat files:

- **Incorrect data—**For data to be correct (valid), its values must adhere to its domain (valid values). For example, a month must be in the range of 1–12, or a person's age must be less than 130. Correctness of data values can usually be programmatically enforced with edit checks and by using lookup tables.

- **Inaccurate data—**A data value can be correct without being accurate. For example, the state code "CA" and the city name "Boston" are both correct, but when used together (such as Boston, CA), the state code is wrong because the city of Boston is in the state of Massachusetts, and the accurate state code for Massachusetts is "MA." Accuracy of dependent data values is difficult to programmatically enforce with simple edit checks or lookup tables. Sometimes it is possible to check against other fields or other files to determine if a data value is accurate in the context in which it is used. However, many times accuracy can be validated only by manually spot-checking against paper files or asking a person (for instance, a customer, vendor, or employee) to verify the data.

- **Business rule violations—**Another type of inaccurate data value is one that violates business rules. For example, an effective date should always precede an expiration date. Another example of a business rule violation might be a Medicare claim for a patient who is not yet of retirement age and does not qualify for Medicare.

- **Inconsistent data—**Uncontrolled data redundancy results in inconsistencies. Every organization is plagued with redundant and inconsistent data. This is especially prevalent with customer data. For example, a customer name on the order database might be "Mary Karlinsky," the same name on the customer database might be "Maria Louise Karlinsky," and on a down-stream customer-relationship, decision-support system the same name might be spelled "Mary L. Karlynski."

- **Incomplete data—**During system requirements definition, we rarely bother to gather the data requirements from down-stream information consumers, such as the marketing department. For example, if we build a system for the lending department of a financial institution, the users of that department will most likely list Initial Loan Amount, Monthly Payment Amount, and Loan Interest Rate as some of the most critical data elements. However, the most important data elements for users of the marketing department are probably Gender Code, Customer Age, or Zip Code of the borrower. Thus, in a system built for the lending department, data elements, such as Gender Code, Customer Age, and Zip Code might not be captured at all, or only haphazardly. This often is the reason why so many data elements in operational systems have missing values or default values.

- **Nonintegrated data—**Most organizations store data redundantly and inconsistently across many systems, which were never designed with integration in mind. Primary keys often don't match or are not unique, and in some cases, they don't even exist. More and more frequently, the development or maintenance of systems is outsourced and even off-shored, which puts data consistency and data quality at risk. For example, customer data can exist on two or more outsourced systems under different customer numbers with different spellings of the customer name and even different phone numbers or addresses. Integrating data from such systems is a challenge.

DATA QUALITY RULES

There are four categories of data quality rules. The first category contains rules about business objects or business entities. The second category contains rules about data elements or business attributes. The third category of rules pertains to various types of dependencies between business entities or business attributes, and the fourth category relates to data validity rules.

Business Entity Rules

Business entities are subject to three data quality rules: uniqueness, cardinality, and optionality. These rules have the following properties:

- **Uniqueness—**There are four basic rules to business entity uniqueness:
  - Every instance of a business entity has its own unique identifier. This is equivalent to saying that every record must have a unique primary key.
  - In addition to being unique, the identifier must always be known. This is equivalent to saying that a primary key can never be NULL.
  - Rule number three applies only to composite or concatenated keys. A *composite key* is a unique identifier that consists of more than one business attribute. This is equivalent to saying that a primary key is made up of several columns. The rule states that a unique identifier must be minimal. This means the identifier can consist only of the minimum number of columns it takes to make each value unique—no more, no less.
  - The fourth rule also applies to composite keys only. It declares that one, many, or all business attributes comprising the unique identifier can be a data relationship between two business entities. This is equivalent to saying that a composite primary key can contain one or more foreign keys.

- **Cardinality—**Cardinality refers to the degree of a relationship, that is, the number of times one business entity can be related to another. There are only three types of cardinality possible. The "correct" cardinality in every situation depends completely on the definition of your business entities and the business rules governing those entities. You have three choices for cardinality:
  - One-to-one cardinality means that a business entity can be related to another business entity once and only once in both directions. For example, a man is married to one and only one woman at one time, and in reverse, a woman is married to one and only one man at one time, at least in most parts of the world.
  - One-to-many (or many-to-one) cardinality means that a business entity can be related to another business entity many times, but the second business entity can be related to the first only once. For example, a school is attended by many children, but each child attends one and only one school.
  - Many-to-many cardinality means that a business entity can be related to another business entity many times in both directions. For example, an adult supports many children, and each child is supported by many adults (in the case of a mother and father supporting a son and a daughter).
- **Optionality—**Optionality is a type of cardinality, but instead of specifying the maximum number of times two business entities can be related, it identifies the minimum number of times they can be related. There are only two options: either two business entities must be related at least once (mandatory relationship) or they don't have to be related (optional relationship). Optionality rules are sometimes called reference rules because they are implemented in relational databases as the referential integrity rules: cascade, restrict, and nullify. Optionality has a total of five rules; the first three apply to the degree of the relationship:
  - One-to-one optionality means that two business entities are tightly coupled. If an instance of one entity exists, then it must be related to at least one instance of the second entity. Conversely, if an instance of the second entity exists, it must be related to at least one instance of the first. For example, a store must offer at least one product, and in reverse, if a product exists, it must be offered through at least one store.

- One-to-zero (or zero-to-one) optionality means that one business entity has a mandatory relationship to another business entity, but the second entity does not require a relationship back to the first. For example, a customer has purchased at least one product (or he wouldn't be a customer on the database), but conversely, a product may exist that has not yet been purchased by any customer.
- Zero-to-zero optionality indicates a completely optional relationship between two business entities in both directions. For example, the department of motor vehicles issues drivers licenses and car licenses. A recently licensed driver may be related to a recently licensed car and vice versa, but this relationship is not mandatory in either direction.
- Every instance of an entity that is being referenced by another entity in the relationship must exist. This is equivalent to saying that when a relationship is instantiated through a foreign key, the referenced row with the same primary key must exist in the other table. For example, if a child attends a school and the school number is the foreign key on the CHILD table, then the same school number must exist as the primary key on the SCHOOL table.
- The reference attribute does not have to be known when an optional relationship is not instantiated. This is equivalent to saying that the foreign key can be NULL on an optional relationship.

Business Attribute Rules

Business attributes are subject to two data quality rules, not counting dependency and validity rules. The two rules are data inheritance and data domains:

- **Data inheritance—**The inheritance rule applies only to supertypes and subtypes. Business entities can be of a generalized type called a supertype, or they can be of a specialized type called a subtype. For example, ACCOUNT is a supertype entity, whereas CHECKING ACCOUNT and SAVINGS ACCOUNT are two subtype entities of ACCOUNT. There are three data inheritance rules:
    - All generalized business attributes of the supertype are inherited by all subtypes. In other words, data elements that apply to all subtypes are stored in the supertype and are automatically applicable to all subtypes. For example, the data element Account Open Date applies to all types of accounts. It is therefore an attribute of the supertype

ACCOUNT and automatically applies to the subtypes CHECKING ACCOUNT and SAVINGS ACCOUNT.

- The unique identifier of the supertype is the same unique identifier of its subtypes. This is equivalent to saying that the primary key is the same for the supertype and its subtypes. For example, the account number of a person's checking account is the same account number, regardless of whether it identifies the supertype ACCOUNT or the subtype CHECKING ACCOUNT.
- All business attributes of a subtype must be unique to that subtype only. For example, the data element Interest Rate is applicable to savings accounts, but not checking accounts, and must therefore reside on the subtype SAVINGS ACCOUNT. If the checking accounts were interest bearing, then a new layer of generalization would have to be introduced to separate interest-bearing from noninterest-bearing accounts.

- **Data domains—**Domains refer to a set of allowable values. For structured data, this can be any of the following:
    - A list of values, such as the 50 U.S. state codes (AL … WY)
    - A range of values (between 1 and 100)
    - A constraint on values (less than 130)
    - A set of allowable characters (a … z, 0 … 9, $, &, =)
    - A pattern, such as a date (CCYY/MM/DD)

  Data domain rules for unstructured data are much more difficult to determine and have to include meta tags to be properly associated with any corresponding structured data. Unstructured data refers to free-form text (such as web pages or e-mails), images (such as videos or photos), sound (such as music or voice messages), and so on. We describe unstructured data in more detail in Chapter 11, "Strategies for Managing Unstructured Data."

Data Dependency Rules

The data dependency rules apply to data relationships between two or more business entities as well as to business attributes. There are seven data dependency rules: three for entity relationships and four for attributes:

- **Entity-relationship dependency—**The three entity-relationship dependency rules are:
    - The existence of a data relationship depends on the state (condition) of another entity that participates in the relationship. For example, orders cannot be placed for a customer whose status is "delinquent."
    - The existence of one data relationship mandates that another data relationship also exists. For example, when an order is placed by a customer, then a salesperson also must be associated with that order.
    - The existence of one data relationship prohibits the existence of another data relationship. For example, an employee who is assigned to a project cannot be enrolled in a training program.
- **Attribute dependency—**The four attribute dependency rules are:
    - The value of one business attribute depends on the state (condition) of the entity in which the attributes exist. For example, when the status of a loan is "funded," the value of Loan Amount must be greater than ZERO and the value of Funding Date must not be NULL. The correct value of one attribute depends on, or is derived from, the values of two or more other attributes. For example, the value of Pay Amount must equal Hours Worked multiplied by Hourly Pay Rate.
    - The allowable value of one attribute is constrained by the value of one or more other attributes in the same business entity or in a different but related business entity. For example, when Loan Type Code is "ARM4" and the Funding Date is prior to 20010101, then the Ceiling Interest Rate cannot exceed the Floor Interest Rate by more than 6 percent.
    - The existence of one attribute value prohibits the existence of another attribute value in the same business entity or in a different but related business entity. For example, when the Monthly Salary Amount is greater than ZERO, then the Commission Rate must be NULL.

Data Validity Rules

Data validity rules govern the quality of data values, also known as data domains. There are six validity rules to consider:

- **Data completeness—**The data completeness rule comes in four flavors:
    - *Entity* completeness requires that all instances exist for all business entities. In other words, all records or rows are present.

  - *Relationship* completeness refers to the condition that referential integrity exists among all referenced business entities.
  - *Attribute* completeness states that all business attributes for each business entity exist. In other words, all columns are present.
  - *Domain* completeness demands that all business attributes contain allowable values and that NULL values can be differentiated from missing values.

- **Data correctness—**This rule requires that all data values for a business attribute must be correct and representative of the attribute's:
  - Definition (the values must reflect the intended meaning of the attribute)
  - Specific individual domains (list of valid values)
  - Applicable business rules
  - Supertype inheritance (if applicable)
  - Identity rule (primary keys)
- **Data accuracy—**This rule states that all data values for a business attribute must be accurate in terms of the attribute's dependency rules and its state in the real world.
- **Data precision—**This rule specifies that all data values for a business attribute must be as precise as required by the attribute's:
  - Business requirements
  - Business rules
  - Intended meaning
  - Intended usage
  - Precision in the real world
- **Data uniqueness—**There are five aspects to the data uniqueness rule:
  - Every business entity instance must be unique, which means no duplicate records or rows.
  - Every business entity must have only one unique identifier, which means no duplicate primary keys.

  - Every business attribute must have only one unique definition, which means there are no homonyms.
  - Every business attribute must have only one unique name, which means there are no synonyms.
  - Every business attribute must have only one unique domain, which means there are no overloaded columns. An overloaded column is a column that is used for more than one purpose. For example, a Customer Type Code has the values A, B, C, D, E, F, where A, B, and C describe a type of customer (for example, a corporation, partnership, or individual), but D, E, and F describe a type of shipping method (for example, USPS, FedEx, or UPS). In this case, the attribute Customer Type Code is overloaded because it is used for two different purposes.

- **Data consistency—**Use the following two rules to enforce data consistency:
  - The data values for a business attribute must be consistent when the attribute is duplicated for performance reasons or when it is stored redundantly for any other reason, such as special timeliness requirements or data distribution issues. Data should never be stored redundantly because of departmental politics, or because you don't trust the data from another user, or because you have some other control issues.
  - The duplicated data values of a business attribute must be based on the same domain (allowable values) and on the same data quality rules.

DATA QUALITY IMPROVEMENT PRACTICES

Many organizations still sidestep long-term data quality improvement practices in favor of achieving short-term goals. However, an increasing number of organizations realize that the consequences of not addressing the poor quality of data may result in adverse effects, such as customer attrition or severe loss in market share. Analyst firms, such as the Gartner Group, have warned of consequences as grave as total business failures.

Data Profiling

The first step in improving data quality is to uncover your data defects through data profiling, sometimes called data archeology, which is the process of analyzing the data for correctness, completeness, uniqueness, consistency, and reasonability. Once a difficult and tedious task requiring dozens of SQL and 4GL/5GL programs searching through every record on every file or database to find data anomalies, data profiling, data cleansing tools now have the capability to profile the data for you.

Similarly, you may be able to leverage some functions of your data mining tool to assess your data quality. For example, Teradata's data mining tool Warehouse Miner has two functions that can be used for source data analysis. Their "values analysis" function identifies characteristics of the data values, such as ZEROs, NULLs, and number of unique values, whereas their "overlap analysis" function identifies the number of overlapping keys that the tables share, which is helpful for data mart consolidation. Histograms and scatter plots allow you to visually detect outliers. In addition, the SQL generated by the tool can be run against the entire database to quickly differentiate the aberrant value deviations from the norm.

Data Cleansing

After the extent of "dirty data" is known, the easiest place to start the data quality improvement process is by cleansing operational data at the time it is moved into DW databases where it is used for cross-organizational reporting. However, data cleansing is a labor-intensive, time-consuming, and expensive process, and cleansing all the data is usually neither cost-justified nor practical. On the other hand, cleansing none of the data is equally unacceptable. It is therefore important to carefully analyze the source data and to classify the data elements as critical, important, or insignificant to the business. Then, concentrate on cleansing all the critical data elements, and as time permits, cleanse as many of the important data elements as practical, leaving the insignificant data elements unchanged. In other words, you do not need to cleanse all the data, and you do not need to do it all at once.

Another factor that will influence your ability to cleanse the data is whether the correct data still exists or whether it can be recreated with a minimal amount of manual or automated effort. There are situations where values are so convoluted or disparate—even with different and opposing meanings to the same fact—that any attempt to decipher such data might produce even worse results. In that case, it might be best to just leave the data alone.

Another decision to make is *how* to cleanse what can reasonably be cleansed. Can the data cleansing products on the market today handle most of the common data quality problems? The answer is yes. Are the data cleansing and extract/transform/load (ETL) products on the market capable of resolving all of the complicated and unique "dirty data" situations on all of your platforms, and will they ever be? The answer is probably no. Therefore, if you are truly serious about creating value-added information out of the dirty data, then you will probably have to invest in writing some procedural code to supplement the capabilities of your tools.

Data Defect Prevention

The next decision to make is how to prevent future "dirty data" from being entered. That begins by identifying the root causes for the data defects, which can be a combination of the following:

- Defective program logic
- Not enough program edits
- Not understanding the meaning of a data element
- No common metadata
- No domain definitions
- No reconciliation process
- No data verification process
- Poor data entry training
- Inadequate time for data entry
- No incentive for quality data entry

The owners of the operational systems should plan to improve their programs and edit checks, unless the effort is unreasonably high. For example, if the corrective action requires changing the file structure, which means modifying (if not rewriting) most of the programs that access that file, then the cost for such an invasive corrective action on the operational system is probably not justifiable—especially if the bad data does not interfere with the operational needs of that system. This type of decision cannot—and should not—be made by IT alone. Downstream information consumers must negotiate with the data originators about justifying and prioritizing the data quality improvement steps.

A data governance group should be established at the enterprise level, which should be staffed with data administrators, metadata administrators, and data quality stewards:

- **Data administrators—**These people are responsible for the enterprise logical data model, for establishing and maintaining naming standards, and for capturing data-related business rules.

- **Metadata administrators—**These people are responsible for loading, linking, managing, and disseminating metadata to facilitate the common understanding of data and to encourage data reuse. Metadata is the contextual information about the data. Metadata components include data names, data definitions, business rules, data content (domains), data type, data length, data owner, data transformations, degree of cleanliness, and so on.

- **Data quality stewards—**These people are charged with preventing the propagation of inferior quality data throughout the enterprise, and thus, the decision-making processes. Therefore, it is their responsibility to perform regular data audits on business data, metadata, and data models, and to be involved in data reconciliation efforts by helping to identify and resolve the root causes of data quality issues. The findings of the audits and reconciliation efforts should feed back into a continuous data quality improvement cycle.

Data quality training should be instituted to address poor data entry habits. Not all data rules can be enforced through edit checks or by the features of relational databases, such as strong data typing, referential integrity, use of look-up tables, and the use of stored edit procedures. Many data violations can still occur because of human error, negligence, or intentionally introduced errors. For example, if an end user needs a new data element but must wait six months for IT to change the database, then the end user might simply decide to *overload* an existing column and use it for dual (or triple) purposes, such as putting the date of the last promotion into the Account Closed Date column.

ENTERPRISE-WIDE DATA QUALITY DISCIPLINES

Organizations have a number of data quality disciplines at their disposal, but rarely will they implement all disciplines at once because improving data quality is a process and not an event. This process is measured on a data quality maturity scale of 1–5. Depending on how fast an organization advances through the data quality maturity levels, it will either institute stringent, light, or no disciplines.

Data Quality Maturity Levels

An easy way to determine your organization's level of data quality maturity is to look at your current data quality improvement activities. Figure 3.1 shows the common data quality improvement activities in each of the five data quality maturity levels based on Larry English's adaptation of the capability maturity model (CMM) to data quality. The five levels are:

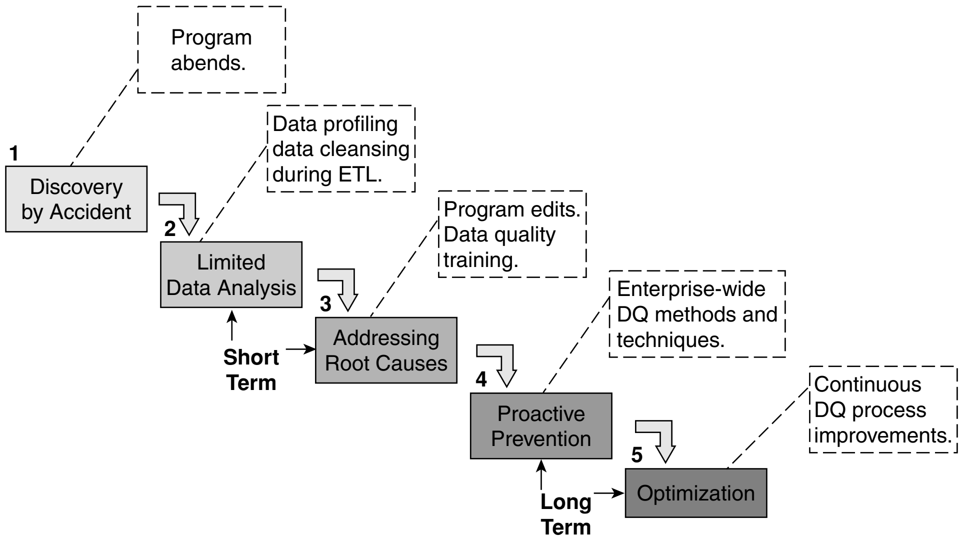
Program abends.
1
Discovery by Accident
Data profiling data cleansing during ETL.
2
Limited Data Analysis
Program edits. Data quality training.
3
Addressing Root Causes
Short Term
Enterprise-wide DQ methods and techniques.
4
Proactive Prevention
Continuous DQ process improvements.
5
Optimization
Long Term

Figure 3.1: Data Quality Improvement Activities

Level 1: Uncertainty—At Level 1, the organization is stumbling over data defects as its programs abend (crash) or its information consumers complain. There is no proactive data quality improvement process, no data quality group, and no funding. The organization denies any serious data quality problems and considers data analysis a waste of time. Or the CIO is ready to retire and doesn't want anything to disrupt it. Basically, the organization is asleep and doesn't want to be awakened.

Level 2: Awakening—At Level 2, the organization performs some limited data analysis and data correction activities, such as data profiling and data cleansing. There still is no enterprise-wide support for data quality improvement, no data quality group, and no funding. However, a few isolated individuals acknowledge their dirty data and want to incorporate data quality disciplines in their projects. These individuals can be data administrators, database administrators, developers, or business people.

Level 3: Enlightenment—At Level 3, the organization starts to address the root causes of its dirty data through program edits and data quality training. A data quality group is created and funding for data quality improvement projects is available. The data quality group immediately performs an enterprise-wide data quality assessment of their critical files and databases, and prioritizes the data quality improvement activities. This group also institutes several data quality disciplines and launches a comprehensive data quality training program across the organization.

Level 4: Wisdom—At Level 4, the organization proactively works on preventing future data defects by adding more data quality disciplines to its data quality improvement program. Managers across the organization accept personal responsibility for data quality. The data quality group has been moved under a chief officer—either the CIO, COO, CFO, or a new position, such as a chief knowledge officer (CKO). Metrics are in place to measure the number of data defects produced by staff, and these metrics are considered in the staff's job performance appraisals. Incentives for improving data quality have replaced incentives for cranking out systems at the speed of light.

Level 5: Certainty—At Level 5, the organization is in an optimization cycle by continuously monitoring and improving its data defect prevention processes. Data quality is an integral part of all business processes. Every job description requires attention to data quality, reporting of data defects, determining the root causes, improving the affected data quality processes to eliminate the root causes, and monitoring the effects of the improvement. Basically, the culture of the organization has changed.

Standards and Guidelines

Data quality does not happen by accident. Organizations must establish standards and guidelines for all personnel to follow to ensure that data quality is addressed during the entire lifecycle of a system. For example, standards should be established for defining the data, naming the data, establishing domains and business rules, and modeling the data. Guidelines should be in place for data entry, edit checking, validating and auditing of data, correcting data errors, and removing the root causes of data contamination. Training and familiarization with the standards and guidelines should be required of all data entry staff, developers, data stewards, and information consumers.

Standards and guidelines should also include policies and procedures, such as operating procedures, change-control procedures, issue management procedures, and data dispute resolution procedures. Additional policies and procedures should be considered for the communication processes, estimating guidelines, roles and responsibilities, and standard documentation formats.

Development Methodology

A development methodology is a common roadmap that provides a complete list of all the major activities and tasks to be performed on projects. The trouble with traditional methodologies is that they do not support cross-organizational data integration activities because operational systems were rarely designed with integration in mind. But increasing demand for integrated systems (including ERP, CRM, and DW) requires a new type of data-driven methodology that includes the appropriate data quality improvement tasks. For example, the methodology must have a separate development step for incrementally building the enterprise logical data model and enforcing data standardization across all projects.

Data Naming and Abbreviations

Data naming and abbreviation standards provide consistency and a common look and feel that are useful for both developers and business people. Proven standards can be applied, such as the convention of name compositions using prime words, qualifiers or modifiers, and class words. Data administrators are usually trained in the various industry-standard naming conventions.

Abbreviations are part of naming standards, but they apply only to physical names, such as column names, table names, or program names. Business names should always be spelled out for clarity and understanding regardless of how long they are. You should publish a standard enterprise-wide abbreviations list that includes industry-specific and organization-specific acronyms. Every project team should use these abbreviations and acronyms.

Metadata

Metadata is descriptive contextual information about architectural components. Metadata can be business metadata, technical metadata, process metadata, and usage metadata. Large amounts of business metadata can be collected about business functions, business processes, business entities, business attributes (data elements), business rules, and data quality. Technical metadata represents the physical architectural components, such as programs, scripts, databases, tables, columns, keys, and indices. Process metadata describes any type of program logic that manipulates data during data capture, data movement, or data retrieval. Usage metadata is statistical information about how systems are used by the business people. For example, what type of data is accessed, by whom, how often, and for what purpose.

You should set up standards or guidelines that govern who captures which metadata components and how, when, and where to capture them. The metadata repository should be set up in such a way that it supports the standards for metadata capture and usage. Metadata is discussed in more detail in Chapter 4, "Metadata."

Data Modeling

There is a difference between logical data modeling and physical data modeling. A logical data model is a normalized business model and a physical data model is a denormalized database design model, also known as a logical database design. These two different types of data models are described in Chapter 5, "Data Modeling." Data quality must be addressed in both sets of models. In addition, the data models themselves must meet data-modeling quality standards with respect to data policies and modeling rules, such as compliance with naming conventions, consistent use of data types and data domains for semantically equivalent attributes, and so on.

For the purpose of finding redundant and inconsistent data, logical entity-relationship modeling with complete data normalization is still the most effective technique because it is a business analysis technique that includes identification, rationalization, and standardization of data through business metadata. Because every business activity or business function uses or manipulates data in some fashion, a logical data model documents those logical data relationships and the business rules, regardless of how the data or the functions are implemented in the physical databases and applications.

Logical data models created for individual applications should be merged into one cohesive, integrated enterprise logical data model. This activity is usually performed by the data administration department, which might be part of the data quality group. The enterprise logical data model is the baseline business information architecture into which physical files and databases are mapped. You should establish standards for creating logical data models as part of system development activities and for merging the models into the enterprise logical data model.

Data Quality

Because most organizations have a lot of dirty data—too much to cleanse it all—they must establish guidelines about triaging (categorizing and prioritizing) dirty data for cleansing. Some data is critical to the organization, some is important but not critical, and some is nice to have but relatively insignificant to the business people. You should create standards that define acceptable data quality thresholds for each of these categories and specify how to measure data quality during and after database updates. Processing rules for error handling and suspending dirty data records for subsequent correction also should be part of the standards.

Testing

You should specify what types of testing should be performed during system development and who should participate in the various types of testing. Specific types of testing include unit testing, integration or regression testing, performance testing, quality assurance testing, and user acceptance testing. Guidelines should be established that describe the types of test cases required, how much regression testing to perform, and under what circumstances to regression test. Testing guidelines should include a brief description of a test plan, perhaps even a template, as well as instructions for how to organize and manage the various testing activities.

Reconciliation

Similar to testing, yet in a separate category, is reconciling the results of any data manipulation, which is the process of capturing, storing, extracting, merging, separating, copying, moving, changing, or deleting data. This is especially true for DW applications that extract data from multiple operational source files and merge the data into one target database. If your organization has adopted an architected data mart strategy, then the various data marts also have to be reconciled to each other to guarantee consistency. This includes having one central staging area with extensive reconciliation programming for every input-process-output module.

Security

Security guidelines apply to operational systems as well as decision-support systems. The only time data security can be slightly relaxed is in data marts where data is highly summarized and the ability to drill down to the details is not enabled. You should establish security standards to guide the project teams on what types of security measures are mandatory for what types of data exposure. The security standards should have guidelines for categorizing data sensitivity and risks of exposure for the organization. Security standards should cover application security, network security, database security, and Web security against intrusions, hackers, and viruses.

Data Quality Metrics

Data quality metrics ordinarily reflect the explicit as well as the implicit business principles of an organization. Business principles are explicit if stated in mission or vision statements, implicit if they are just "understood" by the staff. For example, if an organization rewards project managers for meeting deadlines even though their applications are full of errors, while it punishes project managers for missing deadlines even though their applications are flawless, then the implicit principle is "speed before quality." Therefore, when creating data quality metrics, the explicit as well as implicit business principles must be reviewed and changed, if necessary, to support the metrics.

Another important aspect to measuring data quality is setting goals. Organizations need to be clear on where they are today and what they're trying to achieve in the short term, medium term, and long term. What are the priorities in the organization? Should operational data be addressed or only analytical data? Should financial data be cleansed first or a specific subject area for an application, such as CRM? What is the plan for incrementally managing data quality improvements? What are the staffing requirements and what are the roles and responsibilities for a data quality improvement initiative? These questions must be answered to develop meaningful and actionable data quality metrics.

Enterprise Architecture

Creating and maintaining an enterprise architecture (EA) is a popular method for controlling data redundancies as well as process redundancies, and thereby reducing the anomalies and inconsistencies that are inherently produced by uncontrolled redundancies. EA is comprised of models that describe an organization in terms of its business architecture (business functions, business processes, business data, and so on) and technical architecture (applications, databases, and so on). The purpose of these models is to describe the actual business in which the organization engages. EA is applicable to all organizations, large and small. Because EA models are best built incrementally, one project at a time, it is appropriate to develop EA models on DW and BI projects, as well as on projects that simply solve departmental challenges.

EA includes at least five models, with the business data model and metadata repository being the two most important components for data quality.

- **Business Function model—**This model shows the hierarchy of business functions of an organization. In other words, it shows *what* the organization does. This model is used for organizing or reorganizing the company into its lines of business.

- **Business Process model—**This model shows the business processes being performed for the business functions. In other words, it shows *how* the organization performs its business functions. This model is used for business process reengineering and business process improvement initiatives.

- **Business Data model—**This model is the enterprise logical data model, also known as enterprise information architecture, that shows what *data* supports the business functions and business processes. This model contains:

    - Business objects (data entities)
    - Business activities involving these entities (data relationships)
    - Data stored about these entities (attributes)
    - Rules governing these entities and their attributes (metadata)

  In the real world, business objects and data about those objects are intrinsically unique. Therefore, they appear as entities and attributes once and only once on a business data model, regardless of how many times they are redundantly stored in physical files and databases. There should be only *one* business data model for an organization showing the "single version of the truth" or the "360-degree view" of the organization.

- **Application inventory—**The application inventory is a description of the physical implementation objects that support the organization such as applications (programs and scripts), databases, and other technical components. It shows *where* the architectural pieces reside in the technical architecture. You should always catalog and document your systems because such inventories are crucial for performing impact analysis.

- **Metadata repository—**Models have to be supported by descriptive information, which is called metadata. Metadata is an essential tool for standardizing data, for managing and enforcing the data standards, and for reducing the amount of rework performed by developers or users who are not aware of what already exists and therefore do not reuse any architectural components.

Data Quality Improvement Process

In addition to applying enterprise-wide data quality disciplines, creating an enterprise data model, and documenting metadata, the data quality group should develop their own data quality improvement process. At the highest level, this process must address the six major components shown in Figure 3.2. These components are:

- **Assess—**Every improvement cycle starts with an assessment. This can either be an initial enterprise-wide data quality assessment, a system-by-system data quality assessment, or a department-by-department data quality assessment. When performing the assessment, do not limit your efforts to profiling the data and collecting statistics on data defects. Analyze the entire data entry or data manipulation process to find the root causes of errors and to find process improvement opportunities.

  Another type of assessment is a periodic data audit. This type of assessment is usually limited to one file or one database at a time. It involves data profiling as well as manual validation of data values against the documented data domains (valid data values). These domains should have already been documented as metadata, but if not, they can be found in programs, code translation books, online help screens, spreadsheets, and other documents. In the worst case, they be discovered by asking subject matter experts.

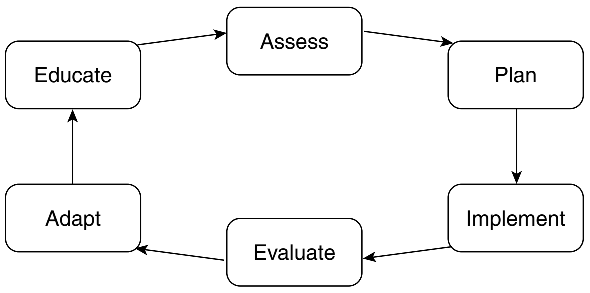

Figure 3.2: Data Quality Improvement Cycle

- **Plan—**After opportunities for improvement have been defined, the improvements should be analyzed, prioritized, approved, funded, staffed, and scheduled. Not all improvements have the same payback and not all improvements are practical or even feasible. An impact analysis should determine which improvements have the most far-reaching benefits. After improvement projects have been prioritized, approved, and funded, they should be staffed and scheduled.

- **Implement—**In some cases, the data quality group can implement the approved improvements, but in many cases, other staff members from both the business side and IT will be required. For example, a decision might have been made that an overloaded column (a column containing data values describing multiple attributes) should be separated in a database. That would involve the business people who are currently accessing the database, the database administrators who are maintaining it, and the developers whose programs are accessing it.

- **Evaluate—**The best ideas sometimes backfire. Although some impact analysis will have been performed during planning, occasionally an adverse impact will be overlooked. Or worse, the implemented improvement might have inadvertently created a new problem. It is therefore advisable to monitor the implemented improvements and evaluate their effectiveness. If deemed necessary, an improvement can be reversed.

- **Adapt—**Hopefully, most improvements do not have to be reversed, but some may have to be modified before announcing them to the entire organization or before turning them into new standards, guidelines, or procedures.

- **Educate—**The final step is to disseminate information about the new improvement process just implemented. Depending on the scope of the change, education can be accomplished through classroom training, computer-based training, an announcement on the organization's intranet website, an internal newsletter, or simple e-mail notification.

BUSINESS SPONSORSHIP

Without executive sponsorship from the business side, the data quality policies of the organization and the work habits of the staff will not change. The best data quality disciplines will have little effect if senior executives continue to reward their staff for speed rather than quality. Senior business executives must institute an incentive program for employees to follow the new data quality policies. The incentive program should be composed of two main parts. One should be public recognition of employees who make major contributions toward the data quality improvement process, and the other should be a monetary bonus. Only through strong business sponsorship and commitment can incentives be changed and a quality improvement process be enforced.

Business Responsibility for Data Quality

Data archeology (finding bad data), data cleansing (correcting bad data), and data quality enforcement (preventing data defects at the source) should be *business* objectives. Therefore, data quality initiatives are *business* initiatives and require the involvement of *business* people, such as information consumers and data originators.

Because data originators create the data and establish business rules and policies over the data, they are directly responsible to the downstream information consumers (knowledge workers, business analysts, and business managers) who need to use that data. If downstream information consumers base their business decisions on poor-quality data and suffer financial losses because of it, then the data originators must be held accountable. Data quality accountability is neither temporary nor application-specific. Thus, the business people must make the commitment to permanently accept these responsibilities.

Data originators, also known as information producers and data owners, are key players in data quality. They are usually business managers and staff responsible for a distinct function or operation of the business. Most operational systems are developed for them, thus, they are the ones who provide the original data requirements, data definitions, data domains, business rules, and process rules. During the requirements definition phase of a new system or during a conversion, data originators should involve downstream information consumers to collect and include the data requirements from these constituents. Information consumers are typically marketing people, the sales force, customer service representatives, or financial analysts.

Data originators are also responsible for participating in all testing activities as well as in retroactive data profiling and data assessment activities. If data defects are discovered, then the data originators should plan to address the root causes that reside in their systems or that resulted from their poor data-entry habits. Information consumers should know who the data originators are, so that they can take their data questions or data disputes directly to them for resolution.

Information consumers are the internal customers who need to consume business data for operational, tactical, or strategic decision-making purposes. They are usually business managers and staff who are responsible for resolving customer inquiries or disputes on the operational level, or for providing executive management with reports for strategic planning. Their data requirements are not the same as those of the data originators, but must be considered when a new system is developed.

Information consumers should participate during the requirements gathering activities for all systems from which they will eventually extract data for their own analytical use. They must participate in the data quality improvement process because they are frequently the first to discover data discrepancies that are not obvious to an operationally-oriented business person.

CONCLUSION

The time has come to acknowledge that an organization can no longer treat data as a byproduct of their systems. In the intelligent enterprise, information is the product and data is its raw material. Because the quality of the product can only be as good as the quality of its raw materials, organizations must bite the bullet and invest in data quality improvement practices. Although you can start small with limited data profiling and data cleansing activities, you must rapidly evolve into a robust data quality improvement program with focus on restoring the cross-organizational, 360-degree view of your business.

REFERENCES

Adelman, Sid and Larissa Terpeluk Moss. *Data Warehouse Project Management*. Boston, MA: Addison-Wesley, 2000.

Duncan, Karolyn and David L. Wells, “Rule-Based Data Cleansing.” *The Journal of Data Warehousing*, Fall 1999.

English, Larry P. *Improving Data Warehouse and Business Information Quality*. New York: John Wiley & Sons, Inc., 1999.

English, Larry. “New Year; New Name; New Resolve for High IQ.” *DM Review*, Volume 13, Number 1, January 2003.

Eckerson, Wayne W. “Data Quality and the Bottom Line.” *TDWI Report Series*, 2003.

Loshin, David. "Customer Care, Consistency and Policy Management." *DM Review*, Volume 13, Number 8, August 2003.

Loshin, David. *Enterprise Knowledge Management—The Data Quality Approach*. San Francisco, CA: Morgan Kaufmann, 2001.

Moss, Larissa and Shaku Atre. *Business Intelligence Roadmap, The Complete Project Lifecycle for Decision-Support Applications.* Boston, MA: Addison-Wesley, 2003.

"TDWI Data Cleansing: Delivering High-Quality Warehouse Data." seminar, TDWI World Conference, San Diego, California: 2004.

Thibodeau, Patrick. "Data Problems Thwart Effort to Count H-1Bs." *Computerworld*: 6 October 2003.