Introduction

The pain that organizations are experiencing around consistent reporting, regulatory compliance, strong interest in Service-Oriented Architecture (SOA), and Software as a Service (SaaS) has prompted a great deal of interest in Master Data Management (MDM). This paper explains what MDM is, why it is important, and how to manage it, while identifying some of the key MDM management patterns and best practices that are emerging. This paper is a high-level treatment of the problem space. In subsequent papers, we will drill down into the technical and procedural issues involved in Master Data Management.

What Is Master Data?

Most software systems have lists of data that are shared and used by several of the applications that make up the system. For example, a typical ERP system as a minimum will have a Customer Master, an Item Master, and an Account Master. This master data is often one of the key assets of a company. It's not unusual for a company to be acquired primarily for access to its Customer Master data.

Rudimentary Definitions

There are some very well-understood and easily identified master-data items, such as "customer" and "product." In fact, many define master data by simply reciting a commonly agreed upon master-data item list, such as: customer, product, location, employee, and asset. But how you identify elements of data that should be managed by a master-data management system is much more complex and defies such rudimentary definitions. In fact, there is a lot of confusion around what master data is and how it is qualified, necessitating a more comprehensive treatment.

There are essentially five types of data in corporations:

- **Unstructured**—This is data found in e-mail, white papers like this, magazine articles, corporate intranet portals, product specifications, marketing collateral, and PDF files.
- **Transactional**—This is data related to sales, deliveries, invoices, trouble tickets, claims, and other monetary and non-monetary interactions.
- **Metadata**—This is data about other data and may reside in a formal repository or in various other forms such as XML documents, report definitions, column descriptions in a database, log files, connections, and configuration files.
- **Hierarchical**—Hierarchical data stores the relationships between other data. It may be stored as part of an accounting system or separately as descriptions of real-world relationships, such as company organizational structures or product lines. Hierarchical data is sometimes considered a super MDM domain, because it is critical to understanding and sometimes discovering the relationships between master data.

- **Master**—Master data are the critical nouns of a business and fall generally into four groupings: people, things, places, and concepts. Further categorizations within those groupings are called subject areas, domain areas, or entity types. For example, within people, there are customer, employee, and salesperson. Within things, there are product, part, store, and asset. Within concepts, there are things like contract, warrantee, and licenses. Finally, within places, there are office locations and geographic divisions. Some of these domain areas may be further divided. Customer may be further segmented, based on incentives and history. A company may have normal customers, as well as premiere and executive customers. Product may be further segmented by sector and industry. The requirements, life cycle, and CRUD cycle for a product in the Consumer Packaged Goods (CPG) sector is likely very different from those of the clothing industry. The granularity of domains is essentially determined by the magnitude of differences between the attributes of the entities within them.

Deciding What to Manage

While identifying master data entities is pretty straightforward, not all data that fits the definition for master data should necessarily be managed as such. This paper narrows the definition of master data to the following criteria, all of which should be considered together when deciding if a given entity should be treated as master data.

Behavior

Master data can be described by the way that it interacts with other data. For example, in transaction systems, master data is almost always involved with transactional data. A *customer* buys a *product*. A *vendor* sells a *part*, and a *partner* delivers a crate of materials to a *location*. An *employee* is hierarchically related to their manager, who reports up through a *manager* (another *employee*). A *product* may be a part of multiple hierarchies describing their placement within a *store*. This relationship between *master data* and *transactional* data may be fundamentally viewed as a noun/verb relationship. Transactional data capture the verbs, such as sale, delivery, purchase, email, and revocation; master data are the nouns. This is the same relationship data-warehouse facts and dimensions share.

Life Cycle

Master data can be described by the way that it is created, read, updated, deleted, and searched. This life cycle is called the CRUD cycle and is different for different master-data element types and companies. For example, how a customer is created depends largely upon a company's business rules, industry segment, and data systems. One company may have multiple customer-creation vectors, such as through the Internet, directly through account representatives, or through outlet stores. Another company may only allow customers to be created through direct contact over the phone with its call center. Further, how a customer element is created is certainly different from how a

vendor element is created. The following table illustrates the differing CRUD cycles for four common master-data subject areas.

Sample CRUD cycle

	Customer	**Product**	**Asset**	**Employee**
Create	Customer visit, such as to Web site or facility; account created	Product purchased or manufactured; SCM involvement	Unit acquired by opening a PO; approval process necessary	HR hires, numerous forms, orientation, benefits selection, asset allocations, office assignments
Read	Contextualized views based on credentials of viewer	Periodic inventory catalogues	Periodic reporting purposes, figuring depreciation, verification	Office access, reviews, insurance-claims, immigration
Update	Address, discounts, phone number, preferences, credit accounts	Packaging changes, raw materials changes	Transfers, maintenance, accident reports	Immigration status, marriage status, level increase, raises, transfers
Destroy	Death, bankruptcy, liquidation, do-not-call.	Canceled, replaced, no longer available	Obsolete, sold, destroyed, stolen, scrapped	Termination, death
Search	CRM system, call-center system, contact-management system	ERP system, orders-processing system	GL tracking, asset DB management	HR LOB system

Cardinality

As cardinality (the number of elements in a set) decreases, the likelihood of an element being treated as a master-data element—even a commonly accepted subject area, such as customer—decreases. For example, if a company has only three customers, most likely they would not consider those customers master data—at least, not in the context of supporting them with a master-data management solution, simply because there is no benefit to managing those customers with a master-data infrastructure. Yet, a company with thousands of customers would consider Customer an important subject area, because of the concomitant issues and benefits around managing such a large set of entities. The customer value to each of these companies is the same. Both rely upon their customers for business. One needs a customer master-data solution; the other does not. Cardinality does not change the classification of a given entity type; however, the importance of having a solution for managing an entity type increases as the cardinality of the entity type increases.

Lifetime

Master data tends to be less volatile than transactional data. As it becomes more volatile, it typically is considered more transactional. For example, some might consider "contract" a master-data element. Others might consider it a transaction. Depending on the lifespan of a contract, it can go either way. An agency promoting professional athletes might consider their contracts as master data. Each is different from the other and typically has a lifetime of greater than a year. It may be tempting to simply have one master-data item called "athlete." However, athletes tend to have more than one contract at any given time: one with their teams and others with companies for endorsing products. The agency would need to manage all those contracts over time, as elements of the contract are renegotiated or athletes traded. Other contracts—for example, contracts for detailing cars or painting a house—are more like a transaction. They are one-time, short-lived agreements to provide services for payment and are typically fulfilled and destroyed within hours.

Complexity

Simple entities, even valuable entities, are rarely a challenge to manage and are rarely considered master-data elements. The less complex an element, the less likely the need to manage change for that element. Typically, such assets are simply collected and tallied. For example, Fort Knox likely would not track information on each individual gold bar stored there, but rather only keep a count of them. The value of each gold bar is substantial, the cardinality high, and the lifespan long; yet, the complexity is low.

Value

The more valuable the data element is to the company, the more likely it will be considered a master data element. Value and complexity work together.

Volatility

While master data is typically less volatile than transactional data, entities with attributes that do not change at all typically do not require a master-data solution. For example, rare coins would seem to meet many of the criteria for a master-data treatment. A rare-coin collector would likely have many rare coins. So, cardinality is high. They are valuable. They are also complex. For example, rare coins have a history and description. There are attributes, such as condition of obverse, reverse, legend, inscription, rim, and field. There are other attributes, such as designer initials, edge design, layers, and portrait.

Yet, rare coins do not need to be managed as a master-data item, because they don't change over time—or, at least, they don't change enough. There may need to be more information added, as the history of a particular coin is revealed or if certain attributes must be corrected. But, generally speaking, rare coins would not be managed through a master-data management system, because they are not volatile enough to warrant a solution.

Reuse

One of the primary drivers of master-data management is reuse. For example, in a simple world, the CRM system would manage everything about a customer and never need to share any information about the customer with other systems. However, in today's complex environments, customer information needs to be shared across multiple applications. That's where the trouble begins. Because—for a number of reasons—access to a master datum is not always available, people start storing master data in various locations, such as spreadsheets and application private stores. There are still reasons, such as data-quality degradation and decay, to manage master data that is not reused across the enterprise. However, if a master-data entity is reused in multiple systems, it's a sure bet that it should be managed with a master-data management system.

To summarize, while it is simple to enumerate the various master-data entity types, it is sometimes more challenging to decide which data items in a company should be treated as master data. Often, data that does not normally comply with the definition for master data may need to be managed as such, and data that does comply with the definition may not. Ultimately, when deciding on what entity types should be treated as master data, it is better to categorize them in terms of their behavior and attributes within the context of the business needs than to rely on simple lists of entity types.

Why Should I Manage Master Data?

Because it is used by multiple applications, an error in master data can cause errors in all the applications that use it. For example, an incorrect address in the customer master might mean orders, bills, and marketing literature are all sent to the wrong address. Similarly, an incorrect price on an item master can be a marketing disaster, and an incorrect account number in an Account Master can lead to huge fines or even jail time for the CEO—a career-limiting move for the person who made the mistake!

Here is a typical master-data horror story: A credit-card customer moves from 2847 North 9th St. to 1001 11th St. North. The customer changed his billing address immediately, but did not receive a bill for several months. One day, the customer received a threatening phone call from the credit-card billing department, asking why the bill has not been paid. The customer verifies that they have the new address, and the billing department verifies that the address on file is 1001 11th St. N. The customer asks for a copy of the bill, to settle the account. After two more weeks without a bill, the customer calls back and finds the account has been turned over to a collection agency. This time, they find out that even though the address in the file was 1001 11th St. N, the billing address is 101 11th St. N. After a bunch of phone calls and letters between lawyers, the bill finally gets resolved and the credit-card company has lost a customer for life. In this case, the master copy of the data was accurate, but another copy of it was flawed. Master data must be both correct *and* consistent.

Even if the master data has no errors, few organizations have just one set of master data. Many companies grow through mergers and acquisitions. Each company you acquire

comes with its own customer master, item master, and so forth. This would not be bad if you could just Union the new master data with your current master data, but unless the company you acquire is in a completely different business in a faraway country, there's a very good chance that some customers and products will appear in both sets of master data—usually, with different formats and different database keys. If both companies use the Dun & Bradstreet number or Social Security number as the customer identifier, discovering which customer records are for the same customer is a straightforward issue; but that seldom happens. In most cases, customer numbers and part numbers are assigned by the software that creates the master records, so the chances of the same customer or the same product having the same identifier in both databases is pretty remote. Item masters can be even harder to reconcile, if equivalent parts are purchased from different vendors with different vendor numbers.

Merging master lists together can be very difficult. The same customer may have different names, customer numbers, addresses, and phone numbers in different databases. For example, William Smith might appear as Bill Smith, Wm. Smith, and William Smithe. Normal database joins and searches will not be able to resolve these differences. A very sophisticated tool that understands nicknames, alternate spellings, and typing errors will be required. The tool will probably also have to recognize that different name variations can be resolved, if they all live at the same address or have the same phone number. While creating a clean master list can be a daunting challenge, there are many positive benefits to your bottom line from a common master list:

- A single, consolidated bill saves money and improves customer satisfaction.
- Sending the same marketing literature to a customer from multiple customer lists wastes money and irritates the customer.
- Before you turn a customer account over to a collection agency, it would be good to know if they owe other parts of your company money or, more importantly, that they are another division's biggest customer.
- Stocking the same item under different part numbers is not only a waste of money and shelf space, but can potentially lead to artificial shortages.

The recent movements toward SOA and SaaS make Master Data Management a critical issue. For example, if you create a single customer service that communicates through well-defined XML messages, you may think you have defined a single view of your customers. But if the same customer is stored in five databases with three different addresses and four different phone numbers, what will your customer service return? Similarly, if you decide to subscribe to a CRM service provided through SaaS, the service provider will need a list of customers for their database. Which one will you send them?

For all these reasons, maintaining a high-quality, consistent set of master data for your organization is rapidly becoming a necessity. The systems and processes required to maintain this data are known as *Master Data Management*.

What Is Master Data Management?

For purposes of this article, we define Master Data Management (MDM) as the technology, tools, and processes required to create and maintain consistent and accurate lists of master data. There are a couple things worth noting in this definition. One is that MDM is not just a technological problem. In many cases, fundamental changes to business process will be required to maintain clean master data, and some of the most difficult MDM issues are more political than technical. The second thing to note is that MDM includes both creating and maintaining master data. Investing a lot of time, money, and effort in creating a clean, consistent set of master data is a wasted effort unless the solution includes tools and processes to keep the master data clean and consistent as it is updated and expanded.

While MDM is most effective when applied to all the master data in an organization, in many cases the risk and expense of an enterprise-wide effort are difficult to justify. It may be easier to start with a few key sources of Master Data and expand the effort, once success has been demonstrated and lessons have been learned. If you do start small, you should include an analysis of all the master data that you might eventually want to include, so you do not make design decisions or tool choices that will force you to start over when you try to incorporate a new data source. For example, if your initial Customer master implementation only includes the 10,000 customers your direct-sales force deals with, you don't want to make design decisions that will preclude adding your 10,000,000 Web customers later.

An MDM project plan will be influenced by requirements, priorities, resource availability, time frame, and the size of the problem. Most MDM projects include at least these phases:

1. **Identify sources of master data.** This step is usually a very revealing exercise. Some companies find they have dozens of databases containing customer data that the IT department did not know existed.
2. **Identify the producers and consumers of the master data.** Which applications produce the master data identified in the first step, and—generally more difficult to determine—which applications use the master data. Depending on the approach you use for maintaining the master data, this step might not be necessary. For example, if all changes are detected and handled at the database level, it probably does not matter where the changes come from.
3. **Collect and analyze metadata about for your master data.** For all the sources identified in step one, what are the entities and attributes of the data, and what do they mean? This should include attribute name, datatype, allowed values, constraints, default values, dependencies, and who owns the definition and maintenance of the data. The owner is the most important and often the hardest to determine. If you have a repository loaded with all your metadata, this step is an easy one. If you have to start from database tables and source code, this could be a significant effort.
4. **Appoint data stewards.** These should be the people with the knowledge of the current source data and the ability to determine how to transform the source into the master-data format. In general, stewards should be appointed from the owners

of each master-data source, the architects responsible for the MDM systems, and representatives from the business users of the master data.

5. **Implement a data-governance program and data-governance council.** This group must have the knowledge and authority to make decisions on how the master data is maintained, what it contains, how long it is kept, and how changes are authorized and audited. Hundreds of decisions must be made in the course of a master-data project, and if there is not a well-defined decision-making body and process, the project can fail, because the politics prevent effective decision making.

6. **Develop the master-data model.** Decide what the master records look like: what attributes are included, what size and datatype they are, what values are allowed, and so forth. This step should also include the mapping between the master-data model and the current data sources. This is normally both the most important and most difficult step in the process. If you try to make everybody happy by including all the source attributes in the master entity, you often end up with master data that is too complex and cumbersome to be useful. For example, if you cannot decide whether weight should be in pounds or kilograms, one approach would be to include both (WeightLb and WeightKg). While this might make people happy, you are wasting megabytes of storage for numbers that can be calculated in microseconds, as well as running the risk of creating inconsistent data (WeightLb = 5 and WeightKg = 5). While this is a pretty trivial example, a bigger issue would be maintaining multiple part numbers for the same part. As in any committee effort, there will be fights and deals resulting in sub-optimal decisions. It's important to work out the decision process, priorities, and final decision maker in advance, to make sure things run smoothly.

7. **Choose a toolset.** You will need to buy or build tools to create the master lists by cleaning, transforming, and merging the source data. You will also need an infrastructure to use and maintain the master list. These functions are covered in detail later in the paper.

You can use a single toolset from a single vendor for all of these functions, or you might want to take a best-of-breed approach. In general, the techniques to clean and merge data are different for different types of data, so there are not a lot of tools that span the whole range of master data.

The two main categories of tools are Customer Data Integration (CDI) tools for creating the customer master and Product Information Management (PIM) tools for creating the product master. Some tools will do both, but generally they are better at one or the other.

The toolset should also have support for finding and fixing data-quality issues and maintaining versions and hierarchies. Versioning is a critical feature, because understanding the history of a master-data record is vital to maintaining its quality and accuracy over time. For example, if a merge tool combines two records for John Smith in Boston, and you decide there really are two different John Smiths

in Boston, you need to know what the records looked like before they were merged, in order to "unmerge" them.

8. **Design the infrastructure.** Once you have clean, consistent master data, you will need to expose it to your applications and provide processes to manage and maintain it. This step is a big-enough issue, I devote a section to it later in the document. When this infrastructure is implemented, you will have a number of applications that will depend on it being available, so reliability and scalability are important considerations to include in your design. In most cases, you will have to implement significant parts of the infrastructure yourself, because it will be designed to fit into your current infrastructure, platforms, and applications.

9. **Generate and test the master data.** This step is where you use the tools you have developed or purchased to merge your source data into your master-data list. This is often an iterative process requiring tinkering with rules and settings to get the matching right. This process also requires a lot of manual inspection to ensure that the results are correct and meet the requirements established for the project. No tool will get the matching done correctly 100 percent of the time, so you will have to weigh the consequences of false matches versus missed matches to determine how to configure the matching tools. False matches can lead to customer dissatisfaction, if bills are inaccurate or the wrong person is arrested. Too many missed matches make the master data less useful, because you are not getting the benefits you invested in MDM to get.

10. **Modify the producing and consuming systems.** Depending on how your MDM implementation is designed, you might have to change the systems that produce, maintain, or consume master data to work with the new source of master data. If the master data is used in a system separate from the source systems—a data warehouse, for example—the source systems might not have to change. If the source systems are going to use the master data, however, there will likely be changes required. Either the source systems will have to access the new master data or the master data will have to be synchronized with the source systems, so that the source systems have a copy of the cleaned-up master data to use. If it's not possible to change one or more of the source systems, either that source system might not be able to use the master data or the master data will have to be integrated with the source system's database through external processes, such as triggers and SQL commands.

The source systems generating new records should be changed to look up existing master record sets before creating new records or updating existing master records. This ensures that the quality of data being generated upstream is good, so that the MDM can function more efficiently and the application itself manages data quality. MDM should be leveraged not only as a system of record, but also as an application that promotes cleaner and more efficient handling of data across all applications in the enterprise. As part of MDM strategy, all three pillars of data management need to be looked into: data origination, data management, and data consumption. It is not possible to have a robust enterprise-level MDM strategy if any one of these aspects is ignored.

11. **Implement the maintenance processes.** As we stated earlier, any MDM implementation must incorporate tools, processes, and people to maintain the quality of the data. All data must have a data steward who is responsible for ensuring the quality of the master data. The data steward is normally a business person who has knowledge of the data, can recognize incorrect data, and has the knowledge and authority to correct the issues. The MDM infrastructure should include tools that help the data steward recognize issues and simplify corrections. A good data-stewardship tool should point out questionable matches that were made—customers with different names and customer numbers that live at the same address, for example. The steward might also want to review items that were added as new, because the match criteria were close but below the threshold. It is important for the data steward to see the history of changes made to the data by the MDM systems, to isolate the source of errors and undo incorrect changes. Maintenance also includes the processes to pull changes and additions into the MDM system, and to distribute the cleansed data to the required places.

As you can see, MDM is a complex process that can go on for a long time. Like most things in software, the key to success is to implement MDM incrementally, so that the business realizes a series of short-term benefits while the complete project is a long-term process. No MDM project can be successful without the support and participation of the business users. IT professionals do not have the domain knowledge to create and maintain high-quality master data. Any MDM project that does not include changes to the processes that create, maintain, and validate master data is likely to fail. The rest of this paper will cover the details of the technology and processes for creating and maintaining master data.

How Do I Create a Master List?

Whether you buy a tool or decide to roll your own, there are two basic steps to creating master data: clean and standardize the data, and match data from all the sources to consolidate duplicates. Before you can start cleaning and normalizing your data, you must understand the data model for the master data. As part of the modeling process, the contents of each attribute were defined, and a mapping was defined from each source system to the master-data model. This information is used to define the transformations necessary to clean your source data.

Cleaning the data and transforming it into the master data model is very similar to the Extract, Transform, and Load (ETL) processes used to populate a data warehouse. If you already have ETL tools and transformation defined, it might be easier just to modify these as required for the master data, instead of learning a new tool. Here are some typical data-cleansing functions:

- **Normalize data formats.** Make all the phone numbers look the same, transform addresses (and so on) to a common format.
- **Replace missing values.** Insert defaults, look up ZIP codes from the address, look up the Dun & Bradstreet number.

- **Standardize values.** Convert all measurements to metric, convert prices to a common currency, change part numbers to an industry standard.
- **Map attributes.** Parse the first name and last name out of a contact-name field, move Part# and partno to the PartNumber field.

Most tools will cleanse the data that they can, and put the rest into an error table for hand processing. Depending on how the matching tool works, the cleansed data will be put into a master table or a series of staging tables. As each source is cleansed, the output should be examined to ensure the cleansing process is working correctly.

Matching master-data records to eliminate duplicates is both the hardest and most important step in creating master data. False matches can actually lose data (two Acme Corporations become one, for example) and missed matches reduce the value of maintaining a common list. The matching accuracy of MDM tools is one of the most important purchase criteria. Some matches are pretty trivial to do. If you have Social Security numbers for all your customers, or if all your products use a common numbering scheme, a database JOIN will find most of the matches. This hardly ever happens in the real world, however, so matching algorithms are normally very complex and sophisticated. Customers can be matched on name, maiden name, nickname, address, phone number, credit-card number, and so on, while products are matched on name, description, part number, specifications, and price. The more attribute matches and the closer the match, the higher degree of confidence the MDM system has in the match. This confidence factor is computed for each match, and if it surpasses a threshold, the records match. The threshold is normally adjusted depending on the consequences of a false match. For example, you might specify that if the confidence level is over 95 percent, the records are merged automatically, and if the confidence is between 80 percent and 95 percent, a data steward should approve the match before they are merged.

Most merge tools merge one set of input into the master list, so the best procedure is to start the list with the data in which you have the most confidence, and then merge the other sources in one at a time. If you have a lot of data and a lot of problems with it, this process can take a long time. You might want to start with the data from which you expect to get the most benefit having consolidated; run a pilot project with that data, to ensure your processes work and you are seeing the business benefits you expect; and then start adding other sources, as time and resources permit. This approach means your project will take longer and possibly cost more, but the risk is lower. This approach also lets you start with a few organizations and add more as the project demonstrates success, instead of trying to get everybody on board from the start.

Another factor to consider when merging your source data into the master list is privacy. When customers become part of the customer master, their information might be visible to any of the applications that have access to the customer master. If the customer data was obtained under a privacy policy that limited its use to a particular application, you might not be able to merge it into the customer master. You might want to add a lawyer to your MDM planning team.

At this point, if your goal was to produce a list of master data, you are done. Print it out or burn it to a CD, and move on. If you want your master data to stay current as data is added and changed, you will have to develop infrastructure and processes to manage the master data over time. The next section provides some options on how to do just that.

How Do I Maintain a Master List?

There are many different tools and techniques for managing and using master data. We will cover three of the more common scenarios here:

- **Single-copy approach**—In this approach, there is only one master copy of the master data. All additions and changes are made directly to the master data. All applications that use master data are rewritten to use the new data instead of their current data. This approach guarantees consistency of the master data, but in most cases it's not practical. Modifying all your applications to use a new data source with a different schema and different data is, at least, very expensive; if some of your applications are purchased, it might even be impossible.
- **Multiple copies, single maintenance**—In this approach, master data is added or changed in the single master copy of the data, but changes are sent out to the source systems in which copies are stored locally. Each application can update the parts of the data that are not part of the master data, but they cannot change or add master data. For example, the inventory system might be able to change quantities and locations of parts, but new parts cannot be added, and the attributes that are included in the product master cannot be changed. This reduces the number of application changes that will be required, but the applications will minimally have to disable functions that add or update master data. Users will have to learn new applications to add or modify master data, and some of the things they normally do will not work anymore.
- **Continuous merge**—In this approach, applications are allowed to change their copy of the master data. Changes made to the source data are sent to the master, where they are merged into the master list. The changes to the master are then sent to the source systems and applied to the local copies. This approach requires few changes to the source systems; if necessary, the change propagation can be handled in the database, so no application code is changed. On the surface, this seems like the ideal solution. Application changes are minimized, and no retraining is required. Everybody keeps doing what they are doing, but with higher-quality, more complete data. This approach does have several issues:
    - **Update conflicts are possible and difficult to reconcile.** What happens if two of the source systems change a customer's address to different values? There's no way for the MDM system to decide which one to keep, so intervention by the data steward is required; in the meantime, the customer has two different addresses. This must be addressed by creating data-governance rules and standard operating procedures, to ensure that update conflicts are reduced or eliminated.
    - **Additions must be remerged.** When a customer is added, there is a chance that another system has already added the customer. To deal with

this situation, all data additions must go through the matching process again to prevent new duplicates in the master.

- **Maintaining consistent values is more difficult.** If the weight of a product is converted from pounds to kilograms and then back to pounds, rounding can change the original weight. This can be disconcerting to a user who enters a value and then sees it change a few seconds later.

In general, all these things can be planned for and dealt with, making the user's life a little easier, at the expense of a more complicated infrastructure to maintain and more work for the data stewards. This might be an acceptable trade-off, but it's one that should be made consciously.

Versioning and Auditing

No matter how you manage your master data, it's important to be able to understand how the data got to the current state. For example, if a customer record was consolidated from two different merged records, you might need to know what the original records looked like, in case a data steward determines that the records were merged by mistake and really should be two different customers. The version management should include a simple interface for displaying versions and reverting all or part of a change to a previous version. The normal branching of versions and grouping of changes that source-control systems use can also be very useful for maintaining different derivation changes and reverting groups of changes to a previous branch.

Data stewardship and compliance requirements will often include a way to determine who made each change and when it was made. To support these requirements, an MDM system should include a facility for auditing changes to the master data. In addition to keeping an audit log, the MDM system should include a simple way to find the particular change you are looking for. An MDM system can audit thousands of changes a day, so search and reporting facilities for the audit log are important.

Hierarchy Management

In addition to the master data itself, the MDM system must maintain data hierarchies— for example, bill of materials for products, sales territory structure, organization structure for customers, and so forth. It's important for the MDM system to capture these hierarchies, but it's also useful for an MDM system to be able to modify the hierarchies independently of the underlying systems. For example, when an employee moves to a different cost center, there might be impacts to the Travel and Expense system, payroll, time reporting, reporting structures, and performance management. If the MDM system manages hierarchies, a change to the hierarchy in a single place can propagate the change to all the underlying systems. There might also be reasons to maintain hierarchies in the MDM system that do not exist in the source systems. For example, revenue and expenses might need to be rolled up into territory or organizational structures that do not exist in any single source system. Planning and forecasting might also require temporary hierarchies to calculate "what if" numbers for proposed organizational changes. Historical

hierarchies are also required in many cases to roll up financial information into structures that existed in the past, but not in the current structure. For these reasons, a powerful, flexible hierarchy-management feature is an important part of an MDM system.

Conclusion

The recent emphasis on regulatory compliance, SOA, and mergers and acquisitions has made the creating and maintaining of accurate and complete master data a business imperative. Both large and small businesses must develop data-maintenance and governance processes and procedures, to obtain and maintain accurate master data. While it's easy to think of master-data management as a technological issue, a purely technological solution without corresponding changes to business processes and controls will likely fail to produce satisfactory results. This paper has covered the reasons for adopting master-data management, the process of developing a solution, and several options for the technological implementation of the solution. Future papers in this series will explain the technological and procedural issues that must be resolved to implement an MDM system.