IBM PRESS

Enterprise Master Data Management

An SOA Approach to Managing Core Information

Allen Dreibelbis, Eberhard Hechler, Ivan Milman,
Martin Oberhofer, Paul van Run, Dan Wolfson

Forewords by Ambuj Goyal and Aaron Zornes

6. The MDM Lifecycle Management Update Service is configured to invoke the Master Data Event Management Services.
7. Master Data Event Management Services determine that the updated master data is managed by a data governance policy. The MDM Lifecycle Management Update Service places the update into a temporary status awaiting further approval.
8. The Master Data Event Management Notification Services construct a notification message and request MDM Interface Services to send the notification to a Data Steward.
9. MDM Interface Services post the notification message to a message queue with information about the master data update.
10. Business Process Services detect the notification and initiate a data governance process. The Data Steward receives a notification message on his or her personalized portal home page, and selects a button on the home page to display the updated master data.
11. The portal verifies the user's authorization to review and approve or reject the update. The portal displays a page with the master data content for the Data Steward to approve or reject. The Data Steward approves the update. Business Process Services then send a message to an MDM Lifecycle Management Update Service to complete the processing of the master data update that was pending approval.
12. The MDM Lifecycle Management Update Service is configured to request MDM Hierarchy and Relationship Management Services to detect and create any hierarchies, groupings, or relationships that may have been defined. The update service then updates the Master Data Repository.
13. MDM Audit Logging Services update the MDM History Database with information about the transaction and the master data values that were updated.
14. The MDM Lifecycle Management Update Service requests MDM Interface Services to construct and post a message to a queue containing the updated master data. Business systems receive and process the update into their system only after each review and approval process is completed by the Data Steward for that line of business.
15. Business Process Services notify each Data Steward about the approved change. Each Data Steward receives a notification message on his or her personalized portal home page and selects a portlet that displays the updated master data. The portal verifies the user's authorization to approve or reject the update. The portal displays a page with the master data content for the Data Steward to approve or reject.
16. As each Data Steward reviews and approves the update, Business Process Services invoke a business system service to retrieve the message from the queue and process the update according to the Data Steward's decision to accept or reject the update.

3.8.7 MDM Scenario—Incremental Updates to a Data Warehouse

The analytical style of MDM supports the use of master data from the MDM System as the accurate, clean source of master data for the loading of transactional data from business systems into a Data Warehouse. We show in this scenario one approach for using master data from the MDM System as part of the incremental update process for a Data Warehouse. The

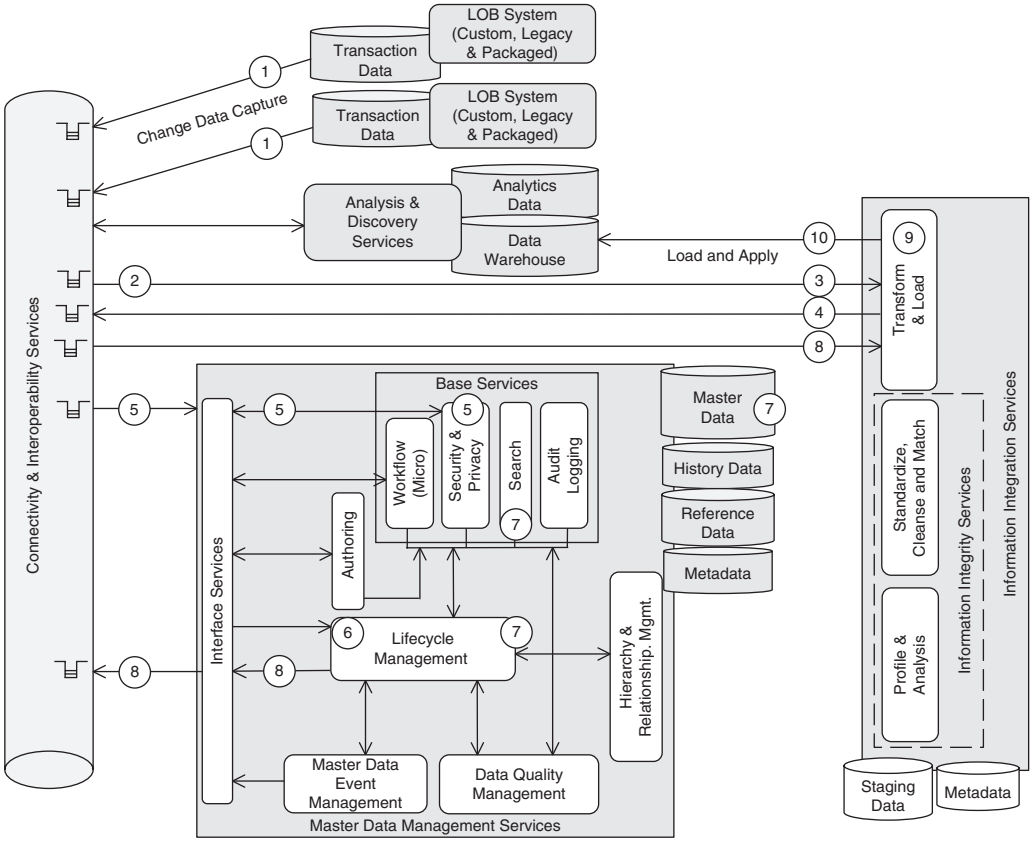

Figure 3.22 MDM Scenario–Incremental Updates to a Data Warehouse.

same scenario could be revised to support the initial loading of a Data Warehouse, but due to the amount of data being loaded, it requires additional steps for staging all of the data to merge, transform, and load into the Data Warehouse.[15]

The volume of data captured between intervals when the ETL process runs should determine if a messaging approach can be used to feed the ETL process or if a batch process should be used to request the most recent master data updates from the MDM System. We assume that the ETL process runs at a frequency that captures a small volume of changes that support the use of messaging to feed the ETL process. Figure 3.22 shows a step-by-step technical walkthrough of the scenario that is described in the following list.

15. Chapter 5 MDM Architecture Patterns contains an Enterprise Deployment BW Pattern that can be referenced for more details about the use of the MDM System to support the loading of clean dimensional data into a Data Warehouse.

1. New and changed data is captured through "Change Data Capture" techniques that use database logs, stored procedures, database triggers, and so on. The "Change Data Capture" process constructs a message with updated data and posts the message to a message queue.
2. Information Integration Data Transformation Services run on a scheduled basis to retrieve messages from the queue.
3. Information Integration Data Transformation Services retrieve the messages, sort through the updates to determine what master data information is needed from the MDM System, and transform the data into an MDM Service request message format.
4. Information Integration Data Transformation Services post a message to a message queue monitored by the MDM System.
5. MDM Interface Services detect a message on the queue, parse the message, and invoke MDM Security and Privacy Services to verify authorization for the service request.
6. MDM Interface Services confirm authorization and invoke an MDM Lifecycle Management Inquiry Service. The MDM Lifecycle Management Inquiry Service invokes an MDM Search Service.
7. The MDM Search Service queries the Master Data Repository and returns the query results to the MDM Lifecycle Management Inquiry Service. The MDM Lifecycle Management Inquiry Service requests MDM Interface Services to construct a response message with the query results.
8. MDM Interface Services create a message with the query results and post that message to a queue monitored by Information Integration Data Transformation Services.
9. Information Integration Data Transformation Services use the returned results from the MDM System as the accurate source for the transformation and enriching of the business transaction data.
10. Information Integration Load Services load the records into the Data Warehouse to complete the incremental load process.

Conclusion

The Master Data Management Reference Architecture should be referenced as input for developing an MDM Solution for the enterprise. The MDM RA is designed to support the evolution of an MDM Solution to implement one or more MDM methods of use and accommodate multiple master data domains. Designing an architecture evolves through multiple stages of elaboration and specification, taking into account system distribution; nonfunctional requirements such as performance, reliability, and high availability; the use of specific products; the choice of middleware; and other technologies. One of the key drivers for the

design of the MDM Reference Architecture is that it should be a scalable, highly available, adaptive and capable of supporting high performance. The implementation of an MDM Solution will always need to consider the existing IT environment, IT standards, enterprise architecture policies, and choice of software for the MDM System and for implementing Information Integration Services such as Information Integrity, ETL, and EII.

It is important to consider the long-term MDM strategy when selecting software for your MDM System. Designing and implementing an MDM System that will continue to deliver sustained business value to the enterprise requires the ability to support the multiple MDM methods of use for the management of master data throughout its lifecycle. In Appendix B, Software and Solution Offerings for MDM Deployments, we identify an overview of software products from relevant software vendors that can be considered for implementing master data solutions. Some of the more advanced MDM products support all of the MDM methods of use required in today's complex business and support multiple master data domains. These mature products also provide intelligent data management by recognizing critical data changes to the information that can trigger strategic enterprise processes and data governance. Finally, MDM products vary in their ability to support master data domains. Coverage may range from specializing in a single domain such as customer or product to maintaining relationships across multiple master data domains. Those that span multiple domains help to harness not only the value of the domain but also the value in the relationships between domains.

One of the desired outcomes for any MDM Solution is to support the needs of the enterprise for many years by providing the flexibility to adapt to the use of new technologies, changing business processes and requirements, legislative changes, and the development of new applications. The selected MDM software should provide the ability to easily include additional new master data domains and attributes, and to implement multiple MDM methods of use as the MDM Solution evolves over time. Criteria for selection of the MDM software should include both its ability to meet current requirements and its flexibility to easily extend to meet the needs of the future. Our experience in selecting Master Data Management software to support the concept of a Party, as defined in Chapter 1, leads to the following observations:

- Start with a robust data model to provide a solid foundation for a low-risk implementation that can drive an early win for the organization.
- A robust data model usually results in relatively few data model extensions or additions being required.
- If additions or extensions are required, it is important to select a software product that provides capabilities to reduce the level of effort to make changes to the data model and ensure consistency in the implementation of the associated services.

Because of the unique requirements of data models developed to implement Product Information Management Solutions, the software selection criteria should focus on the authoring capabilities of the MDM product to model, manage, and create product information. Product information is typically stored in a hierarchical structure. Data modeling capability should provide for reuse of data objects such as catalogs, items, hierarchies, product specifications,

attributes, views, and localization. Catalog hierarchies are used to model item taxonomies, which are tree-like arrangements of categories and subcategories. There are typically two types of hierarchies—category and organization. Items, catalogs, and hierarchies all have attributes. Inheritance provides for the reuse of the same product information across items or categories. Ideally, the product should have attribute level inheritance to maximize the reuse of attributes to define multiple catalogs, items, and hierarchies. Authoring services should also provide localization capabilities to reuse attribute information that has been converted to a particular language to support users across multiple geographies in today's global economy.

From a best-practice perspective, there can be substantial value in selecting an ETL tool that can provide reusable Information Integration Services to support both the initial loading of the MDM System and subsequent use by all systems. Using configurable software that externalizes rules and interfaces provides the flexibility needed to implement an MDM Solution that drives business value to your organization while imposing the necessary constraints that maintain data quality and integrity. The MDM System, in combination with Information Integrity Services, should provide the following:

- The ability to use code tables to normalize data and align data cleansing options with organizational needs and processes
- Externalization of data cleansing, standardization, and matching rules to ensure flexibility in the implementation
- Data validation at the attribute level performed as a common service so that reusable information services can be consistently used across the organization

Implementing high-availability techniques would be specific to the software technologies selected for implementing the MDM Solution architecture. Planning for high availability involves the identification and elimination of "single points of failure" that could prevent the MDM System or Information Integration Services from being available or impact communications between the business systems and the MDM System. When designing how to implement an MDM System within the enterprise, design points for the physical architecture should consider unplanned outages due to hardware and/or software failures and should provide the ability to perform system maintenance without requiring a scheduled system outage. Designing for high availability requires implementing redundant conceptual nodes within the architecture and using hardware and software technologies that provide high-availability features such as RAID,[16] clustering with load balancing, dynamic switching, and hot failover capabilities.

High availability can be achieved by loosely coupling application and system to system communications through the use of asynchronous messaging and by avoiding the implementation of point-to-point communications between systems or between applications. Asynchronous communications provide a mechanism to safely quiesce a system component

16. RAID refers to a redundant array of independent drives that support a storage scheme using multiple hard drives to replicate data among the drive-through techniques such as data mirroring.

without the loss of transaction data communicated between components, thus making it possible to replace or perform application and software product upgrades and maintain continuity of operations. Although all of the system functionality may not be available during the outage of any one component, critical business functionality can be designed to remain active while less critical functions are down for maintenance. Scalability is achieved through the concept of implementing well-defined components that can be individually or collectively adjusted to meet variable demands by adding additional servers (scale-out) or by increasing the processing capability of a server (scale-up). Because messaging services or middleware can also be used to queue requests until an application is ready to process them, the application can control the rate at which it processes the requests so as not to become overloaded by too many simultaneous requests.

Best practices for high availability also employ the use of load balancing techniques, workload distribution, and the clustering of multiple servers. Hardware or software is responsible for making the set of servers appear to be a single server, balancing workload requests across available servers, and directing new requests to the least-busy server. Clustering technology with load balancing also provides the ability to increase capacity by adding more servers without taking the system down. Selection of the type of load balancing technology should consider if session data for a user must be maintained to support processing subsequent requests. If the application server is processing transactions that require multiple steps, then session data about previous user interactions needs to be carried forward to future interactions. In the event that session data must be maintained, techniques such as sticky ports or the use of a central data store containing session data and accessible by each server could be used.

Software selection criteria for the MDM System should also include the ability of the software to scale to meet the long-term needs for the organization. The ability of a software product to scale is usually assessed through the implementation of a proof-of-concept and through discussions with customer references provided from the vendor. However, planning for the implementation and deployment activities of an MDM Solution should always follow good system engineering practices. The risks of not understanding how the software will perform and scale as requirements increase over time can have negative effects on the overall value the MDM Solution can provide to the business. The performance of a system usually has three aspects that must be provided for when delivering an MDM System:

- Response time, both online and batch
- Transaction throughput, for example, the number of transactions per second, or records processed per time period (minute, hour, day, month, etc.)
- Capacity, for example, processor, memory, network, storage

Every MDM project will have its own unique set of challenges and risks that need to be considered for the selection of software and the implementation strategy. The Master Data Management Reference Architecture provides the basis for developing an MDM Solution for the enterprise that is based upon architecture patterns and best practices. Selecting the right software to meet both the tactical and long-term strategic business objectives is critical for achieving both the immediate and long-term business value of an MDM Solution.