

WHITE PAPER

DATA MODELING AND MDM: THE FACTS AND THE FICTION

By Evan Levy
PARTNER AND CO-FOUNDER
BASELINE CONSULTING

Table of Contents

In the Beginning: The Data Model Gets Traction...	2
Make Sure the Cart Follows the Horse.....	3
The Data Model Evolves	4
The Features of an MDM Data Model	5
The Five Pillars of MDM.....	6
MDM Data Model Support	7
Data Standards	7
Data Management.....	8
MDM Processing.....	8
MDM Data Models: Build or Buy?.....	8
Getting Ready for MDM.....	10

As Fortune 500 companies investigate MDM efforts within their companies, the concept of data models frequently appears. From business leaders to CIOs to database administrators, the question arises – what’s the relationship of one to the other? Do I need a data model before I start my initiative? Where can I find one? Should I build it myself? If I get it wrong, is my project doomed? Fortunately, data models aren’t a new concept to the IT world. Understanding the history, different definitions and goals of a data model can help answer many of these questions.

In the early 1990s, a practice called data modeling was all the rage. Numerous companies embraced data model development. There was even an entire software category called CASE (Computer Aided Software Engineering) tools, with its own conventions and subcategories. The concept of business requirements, data analysis and database design had been around for more than 20 years; the documentation created by CASE tools brought visibility to the value of the data model as a tool for illustrating data requirements, data details and data relationships. As many learned, the data model itself would turn out to eclipse the automation used to document it.

In the Beginning: The Data Model Gets Traction

Data modeling was a natural outgrowth of the emergence of relational databases. The whole idea was to provide a means of documenting business information in a manner that reflected how a company operates—and reflecting its business practices in its data.

For most of computer history, systems identified and stored data in a structure to enable applications to function quickly and efficiently. The data was created, named and stored in a manner that was intuitive to application developers. Unfortunately, the data structures and the associated details weren’t easily understandable to business users. Elements had names like “Acct_ID,” “X3RR01” and “tSbar05.” Before data modeling, most business and IT professionals considered data to be a by-product of the systems that generated it.

The issue of data standards and structures didn’t really become visible to business users until the value of data analysis and reporting rose in visibility. While every application system had some amount of built-in reporting, the analytical needs of many business users exceeded the data content of an individual application system. So, the need for integrating data from multiple systems became necessary to support the analytical needs. Once the data moved out of its native environment, naming conventions, value standardization, and data relationships suddenly became important. Without these details, there was no way to understand the data content without contacting the original developer. And as systems were enhanced, combined, or even retired, managing all this data became more difficult. The lack of data standards became an obstacle to analysis; consequently, sabotaging key business decisions.

Data modeling became popular as a means of identifying the data requirements and ensuring data reuse. It provided the convention for the identification and definition of every data element contained within the application, and offered a means of documenting business relationships and definitions. The data model often became a touchstone for understanding the meaning and context of corporate information.

There are a few important points to realize about data modeling:

- A data model identifies data according to its business usage and meaning.
- Because applications are unique in their business purpose, data models and their content may differ across different systems, organizations and lines of business.
- Data modeling is a logical construct, and arguably a business one. However, the term is increasingly being used synonymously with physical database design, which represents the design of the physical data structures usually as an outgrowth of the data model.

DM Review defines a logical data model as “a representation of business concepts laid out in a visual format that clearly shows these concepts and their various relationships.”¹ A data model is a documentation artifact that identifies and describes data using a sanctioned convention—dimensional modeling and third-normal form are the two prevailing ones—that ensures that data elements, their definition and their relationships are clearly identified and distinguished.

Data modeling has also proved to be valuable as a means of documenting data requirements. All too often systems are built based on processing needs with little attention paid to the specific data requirements. Data modeling actually supports the design process by ensuring that all data elements are clearly identified (using business terminology), valued and described in business terms.

For instance, a data model might illustrate that an address will include a value called state or province, denoted by a 2-character standard abbreviation. It would further identify accepted values and if nulls were accepted. The benefit of this approach is simple; the data content and rules are well-defined and documented. Irrespective of the convention used—there are several different methods for developing and documenting a data model—the value of creating a data model was that it represented the real-life business relationships and definitions of data elements.

Make Sure the Cart Follows the Horse

Even in its heyday, data modeling still caused a certain amount of confusion. People often gave data models undue influence, allowing a level of visibility that didn't match its relative importance and using it as a substitute for business-requirements gathering or end-user interviews. The time and effort associated with data model construction often reflected the domain knowledge of the business users and data analysts involved, instead of being confined to the functional boundaries of the targeted system. In her book *e-Data: Turning Data Into Information with Data Warehousing*, author Jill Dyché revealed the so-called “dirty little secrets” of data warehousing. Her Dirty Little Secret Number 2? “Data modeling gets a disproportionate amount of attention.”²

¹ Sreedhar Srikant, “Logical Data Modeling: A Key to Successful Enterprise Data Warehouse Implementations,” *DM Review*, September 2006.

² Jill Dyché, *e-Data: Turning Data Into Information With Data Warehousing*, (Addison Wesley, 2000), page 269.

Dyché went on to advise her readers that:

“When requirements are gathered first and then published in a document that a cross-section of managers, end-users, and technical staff can review and approve, data modeling becomes what it was always meant to be: a means of representing business requirements through documenting data relationships, serving as input to the subsequent implementation of the database tables.”

More than a dozen years later, the high-tech industry is still in the midst of data modeling debates. MDM (master data management) has forced the data model discussion back into the limelight as practitioners and business users alike inquire, “Can we use our existing data model for MDM?” “Is the data model for MDM any different anyway?” “Can we re-do our outdated data model, or do we have to create a new one from scratch?” or “Can we just purchase an off-the-shelf model?”. The answer lies somewhere in between. Although the exercise of data modeling does indeed force many companies to deconstruct and understand their business data—a valuable exercise for a variety of reasons—for MDM the value of the data model is different.

The Data Model Evolves

The truth is a data model’s value varies widely depending on its ultimate purpose. For example, a retailer’s POS (point-of-sale) data model is designed to support fast price lookup and data collection, and is relatively narrow in scope to enable the lightning-fast response times essential when a customer is standing in at the cash register, money in hand. On the other end of the spectrum, the same retailer may have a purchase analytics data model that is much larger and more complex to allow for the storage of history, in-depth analytics and reporting on trends. Since this latter model must support heavy loads and complex queries, it compromises a bit on speed and response time to provide for more comprehensive business intelligence requirements.

Specifically, data models differ as their information content varies. Systems with a focused function typically process and collect low-level, event-based detail. Cross-functional systems frequently focus on storing the final result or aggregate level detail. For example, an ATM relies on a lean data model, since the goal is speedy processing of a limited set of transactions. The moment you insert your card and enter your PIN, the ATM has to determine who you are and your available accounts, and then process the requested transactions. The model is based on the data contained within the system. With the limited amount of data available at point of transaction, you can get cash, make deposits or transfer funds at ATMs, but are restricted from transactions requiring significantly more data. For example, you can’t apply for a loan or refinance your mortgage.

That same bank, however, also has back-end systems that contain large, complex data models to perform ad-hoc analysis on transactions, processing and customer behavior. In those data warehouses, there are dozens of tables containing the data necessary to support the complex analytical processing. These tables house essential information to support data staging areas, data cleansing, transformation, processing and other functionalities. A data model doesn’t just contain the data accessible to end users;

it contains all the data details necessary to support the processing occurring within a system or application. In fact, many data warehouses provide end-users access to less than a quarter of the total number of actual tables.

	BANKING ATM	BANK ACCOUNT ANALYSIS	RETAIL POS	PURCHASE ANALYTICS
BUSINESS PROCESS	Processing account transactions and inquiries	Historical, trending analysis of business assets (customers accts, products, etc.)	Item price lookup and payment processing for purchase	Current and historical analysis of product purchase details
DATA PROCESS	Identity, search, transaction capture, rule check	Complex query and retrieval, large data volume manipulation	Identity, search, transaction capture, rule check	Complex query and retrieval, large data volume manipulation
LOGICAL DATA MODEL	Contains low-level functional events and reference data. Focused on accuracy	Multiple subjects (dozens of entities) for cross-functional analysis	Small number of entities focused on lookup and transaction capture	Multiple subjects (dozens of entities) for cross-functional analysis
PHYSICAL SCHEMA	Small number of tables. Typically denormalized for speed	Must support large volume loads & complex analysis (many tables, many joins)	Small number of tables typically denormalized for speed	Supports heavy loads, maintain history, and support complex queries
SCHEMA COMPLEXITY	Large volume, high speed transactions requires special design	Typically large but simple in structure to support access	High speed lookup processing requires denormalized schema	Typically large but simple in structure to support access

This figure illustrates why data models differ based on the different needs of the various applications they support. ▶

A POS system collects and creates data, reflecting a very narrow set of business operations, so the data model should be designed to support these functions. The POS system must also deal with transactional integrity. This invites additional content into the data model, such as the time the transaction took place. A data mart supporting purchase analytics, on the other hand, doesn't create data—its data model will cover multiple subject areas reflecting how the business functions. The POS system was designed for streamlined data collection and processing; the purchase analytics system for understanding after-the-fact business behaviors.

The actual construction of a data model thus requires an understanding of the content the system needs to do its job. Stakeholders pose both the biggest impetus and the largest roadblock to data model success, as they must agree on the terminology and definitions that drive their business. This is easier said than done.

The Features of an MDM Data Model

More and more companies are turning to MDM in their quest to better manage the mounds of disjointed data. MDM is about consolidating the methods, processes, controls and automation necessary to standardize and integrate data originating from different sources. Depending on your business, this can include customers, stores, locations, suppliers, employees, assets, products, partners and more. The challenge of MDM is managing – and truly mastering – all your data while delivering a clean, consistent, complete image.

Because corporate information is always changing, mastering data also involves supporting a robust change control component within MDM. Business data changes in real-time, and an MDM environment must recognize which changes are acceptable. A mature MDM hub understands when a data element change can be supported in an automated fashion, or when it requires intervention from an external process (e.g. another system, a data steward, etc.). Unlike traditional IT-based change control methods that focus on application changes or business process changes, MDM focuses on data value changes.

MDM also involves the actual processing of data, from basic matching and identification to data correction and CRUD (create, read, update, delete) processing. Processing rules contain the details for determining whether two records are the same. For example, are customers Robert Smith and Bob Smyth the same? If so, should one of the values be updated? If Bob calls and updates his address, should that change be propagated throughout the other databases in the system, including the master record? Does it matter if the address change comes from a third-party data vendor instead? Which update should the system trust? And if it turns out that the data was incorrectly entered, the data model should support unwinding the match to set the record straight.

This discussion of the five pillars is helpful in understanding the differences between MDM and other strategic technology solutions, and will help to put the data model in better perspective.

MDM Data Model Support

Given the specific needs discussed above, MDM requires a different type of data structure, and hence a different type of model altogether. To make all these moving pieces work together, a well thought-out data model is essential to a successful MDM implementation. But that model is likely to look very different from the industry model you purchased for your data warehouse, or from the enterprise data model your data architects painstakingly developed from scratch.

Why? Because the MDM data model should reflect information about the data sources in order to support the processing of the MDM hub itself. Looking back at Figure 2, notice the three categories at the bottom that encapsulate the various MDM functions. These categories, Data Standards, Data Management and MDM Processing, are what the MDM data model needs to support.

Data Standards

The MDM data model must support the rules and details necessary to track the various values associated with a subject entry contained within the individual attached systems in order to support the identification and match processing. Additionally, the MDM data model must retain the relationship details to ensure that any data changes are processed in a manner reflecting all the details associated with an individual value (e.g. the new value is an acceptable value, it's formatted accurately, represented correctly, etc.) Additionally, the effect of a value change must also be propagated through any relationship detail to ensure the integrity of any groupings or hierarchy remain accurate.

The MDM data model must store not only the golden record value, but all of the details associated with identification, matching and relationships.

Data Management

For MDM, tracking data lineage is essential. To maintain the integrity of the data and any updates, it's important to ensure that all lineage and transaction history are retained. We need to know where the data came from, details about its original format and its creation date. Did the Robert Smith customer record change based on a request from the customer support system? Or did it occur through a third-party data source (such as the Department of Motor Vehicles)? Because MDM supports operational data identification, matching and integration, the data model must be able to store the details associated with any platform associated with data access.

In this age of growing privacy concerns, a data model must have strict rules for security and access. Unlike traditional DBMS systems where security is focused on database objects (tables, views, etc.), MDM supports security at the data element level. This means detailed security rules must be retained to provide individual data element access control. Who in the organization should be able to create or update records? Should some users have read-only access on certain, vital fields? Perhaps some attributes must be hidden entirely. Who can delete entire records? To support the breadth of data management functionality, the MDM hub must support storing data lineage and security and access details for each master data element

MDM Processing

In addition to supporting data management functionality, an MDM data model needs to support retaining the operational (or data transactions) activities performed by the hub. Retaining a limited amount of data transaction history (often referred to as "transaction logs") is important in order to support any audit requirements or address any potential error situations. As the number of systems accessing the hub grows, the likelihood of errors grows and the need for transaction corrections becomes necessary. If Robert Smith's customer record was inadvertently deleted due to an error, it will be necessary to recreate his details. Having attribute history available within a data model enables the organization to undo the erroneous change across the enterprise. A successful data model should support attribute history and use that history when unforeseen situations occur and are distributed to downstream systems.

MDM Data Models: Build or Buy?

All MDM data models require some amount of configuration to customize the model to meet your business' unique needs, systems and definitions. There are two common alternatives to building a data model from scratch to support an MDM initiative: acquiring an industry standard model or utilizing an existing data model.

Even within industries, it is difficult to develop a single data model to fit an entire industry's needs. An industry data model that is focused on a generic enterprise business view of data still requires a degree of customization to take into account data unique to your business.

While an industry data model can support your industry’s standard components, effort will be necessary to ensure that the data standards and management details align to your individual company (e.g. Do you work with suppliers or partners? Do you sell to customers or clients?) And, it’s important to remember that you still need to take into account your specific systems and data. The typical industry data model—usually designed for use by a data warehouse or BI application—is not designed to support the three categories described above: data standards, data management, and MDM processing. An industry data model will focus on the subject areas and descriptive details associated with a particular industry and how they relate. An MDM hub focuses on a single subject area. An industry data model contains far more detail than the MDM hub needs, and at the same time typically doesn’t reflect MDM-specific functionality like value standardization and data lineage.

Some companies assume they can leverage their pre-existing data model—again, one that was originally built to support an enterprise data warehouse or analytical application. There is some merit to leveraging an existing enterprise data model, namely the ability to identify the discrete attributes associated with a particular subject area (e.g. product, customer). However, since MDM is usually focused on a single subject area, a multi-subject, cross-functional data model can be overkill. If your MDM project’s aim is to better understand your customer, you’re only trying to master the customer data; product details, deposit history and branch location information is irrelevant. It’s also important to remember, your existing data model will have to be enhanced to support the data standards, data management and processing details associated with master data management.

In either of these situations, delivering data to other systems will require integration – the ongoing connection of new systems to the MDM system means translation from one MDM model to another. For the rest listed above, it’s unlikely the data model you need to support your MDM efforts will match exactly to other business systems. The use of existing data model artifacts can prove too valuable to ignore in MDM implementations. Nevertheless, it’s important to realize that a data warehouse data model is different than an MDM data model.



Figure 3: Data Model Differences ▶

The fact is, many of today's businesses have already expended tremendous time and effort defining their customers and products, and some also have well-defined data models. This work can be leveraged successfully into a data model as a preparation for MDM. However, it is essential to understand that even these definitions may change as you simplify and streamline your data tables to prepare for the speed of MDM processing. The physical schema will inevitably require some redesign to address these performance issues.

Getting Ready for MDM

No matter the stage of your MDM journey, there are questions to ask potential MDM vendors to ensure that you can make the most of your data and achieve the necessary results. Some of the questions to ask when considering whether to build a data model from scratch, reuse existing data model, or purchase an industry data model include:

- Does the data model support the necessary data standard and data management detail?
- Can the data model support modification and expansion of content to address the inevitable business changes? Will those changes affect the underlying processing of the MDM environment?
- Does the MDM model reflect the specific details about your company, or will modification be necessary to convert generic industry terms to specific business terms?
- How are the MDM processing details supported within the data model (logging, audit, security, etc.)?
- How is change control to the model and underlying data handled? How much time (and development) is necessary to support your unique content requirements?
- Is an MDM-specific data model even necessary? Do you have an existing data model that can be enhanced to support your MDM requirements?

It is clear that a robust data model is crucial to MDM success; however, it's important to keep the data model question in perspective. Whether you choose to build your own MDM hub or acquire an MDM product off-the-shelf, you will likely spend time reviewing and planning for data model development. The real challenge will be to ensure that any data model activities are focused solely on the specifics of your company. Given that MDM hubs are focused on supporting a single subject area, we often find that the data model discussion is typically blown out of proportion. Defining and developing an MDM data model isn't rocket science, but it should be custom to the specifics of your business. And, with the value proposition of master data being increasingly embraced in the C-suite, the sooner you begin, the better for your business.

For more information on how software from Initiate Systems can help with your master data management efforts, visit www.Initiate.com.



Facebook



Twitter

United States & Canada
+1 312 395 1350

Initiate Systems Government Operations
+1 301 571 2490

Asia Pacific - Australia
+61 (0) 2 8061 3800

Europe, Middle East and Africa - UK
+44 (0) 118 925 3322

Evan Levy is a partner and co-founder of Baseline Consulting (www.baseline-consulting.com), a technology and management consulting firm specializing in data management, data integration and data warehousing. In addition to his executive management responsibilities at Baseline, Evan advises vendors and VC firms on new and emerging product strategies. Considered an industry leader on the topic of data integration and management, Evan is a faculty member of The Data Warehousing Institute and has been a featured speaker at TechTarget and SourceMedia events. He is also co-author of the book *Customer Data Integration: Reaching a Single Version of the Truth* (John Wiley and Sons).