

BACK TO BUSINESS:

HOW BUSINESS MODELING RATIONALIZES DATA WAREHOUSING



HIRED BRAINS, INC.

By Neil Raden
January, 2008

ABOUT THE AUTHOR



Neil Raden is the founder of Hired Brains, Inc./Hired Brains Research. Hired Brains <http://www.hiredbrains.com> provides consulting, systems integration and implementation services in Business Intelligence, Data Warehousing, Performance Management, Advanced Analytics and Information Integration for clients worldwide. Hired Brains Research provides consulting, market research and advisory services to the technology industry. Based in Santa Barbara, CA, Neil Raden is an active consultant and widely published author and speaker and is the co-author of the recently published "[Smart \(Enough\) Systems](#)" (Prentice Hall, 2007) He welcomes your comments at nraden@hiredbrains.com.

EXECUTIVE SUMMARY

The data-centric designs of data warehousing were conceived at a time when the Business Intelligence (BI) problem was thought to be a problem of data management - finding it, cleaning it and storing it. A scan of the early literature reveals that the needs of knowledge workers were clearly subordinate to creating integrated stores. Data warehouse "architecture" simply assumed that provisioning data was sufficient. It was entirely focused on the back-end of data warehousing, a disposition that still lingers today. For example, the primary data warehouse is often off limits to most analytical processes. Stovepipe-like data marts are created when the original concept of a data warehouse was to integrate data across the stovepipes. Overly-complex designs were created, full of latency from too many layers and burdened with ongoing maintenance costs that exceed those of most operational systems.

Today's business environment demands extreme measures to find efficiencies and to exploit them. Time is precious.

Data warehouse latency is usually defined as query performance, but there is another more important latency - the duration of time between a great idea gelling in a person's mind and the time it takes to implement that great idea in technology so that it can be used. And that is where the advancements of the last decade or two hit the wall. Our existing methodologies for moving those ideas into (computer-based) production, especially when it comes to data warehousing and Business Intelligence (BI) in the broader sense, are too slow. They have too many steps. They separate the experts in the business from the experts with the machines. They are too brittle, too hard to enhance and upgrade and too expensive. They are data-centric, not business-centric.

An alternative to a data-centric approach is a model-driven approach or, in practice, a model-driven architecture. Model-driven architecture has proven to be highly effective in delivering useful results with good ROI in a short period of time. In particular, a model-driven architecture can excel at providing high levels of reusability, flexibility and maintainability. The concept behind model-driven architectures is to build declarative models, and separate the impositions of current data structures, applications and sources from the definitions of the business. This allows for business users to discuss the business in terms that they understand.

A model-driven approach to the entire process of data integration management, and information delivery including metadata, changes everything, especially the way stakeholders work with each other. A business person is able to describe the data that they need and the transformation of it into reports and analyses. The business model takes over the laborious task of managing the interaction with the different databases, structures, keys, transformations and other physical elements of the architecture. This "abstraction" layer insures that no person or process refers directly to, or has to worry about, a physical object, allowing the BI environment to be constantly tuned and optimized. In this way, it is possible to separate the specification of a "function" from the physical implementation that delivers the "function," which tends to loosen the bonds of dependencies between the elements and vastly easing the burden of maintenance and enhancement.

Data warehouses were conceived as orderly, mostly static databases, accessed in a controlled way to service business reports. With ever shortening business cycles, the externalization of business brought on by the Internet, the massive scale and relatively low cost of computing hardware and the flattening of the competitive playing field as a result, the mission of data warehouses is changing dramatically. Existing tools and methodologies cannot provide the speed and agility that is needed. A better way to model and manage these assets is needed.

THE MODEL-DRIVEN APPROACH

Model Driven Architecture, as a term, was invented and trademarked by the OMG (Object Management Group), but is so widely used now that the descriptive term model-driven architecture, or MDA for short, is almost generic. In essence, MDA refers to the creation of physical objects, such as database schema, from specifications created by those with “domain knowledge.” A domain expert can be, for example, a person or group who is or are well-versed in a particular aspect of an organization, such as finance, marketing, manufacturing, distribution, underwriting, claims or other functional aspects of a business or other type of organization.

In a MDA environment, domain experts¹ are able to directly impart their knowledge into a business modeling system using semantics and concepts that are familiar to them, that they can describe with a certain degree of facility and, most importantly of all, can be communicated and discussed with their peers. There is no requirement to understand the form or nature of the underlying physical structures that will be created to instantiate their designs. In fact, tools that use an MDA approach can produce the code to create and modify these physical structures without any intermediate steps.

The advantage of MDA is that it is an abstraction. It allows those who understand the application, not its mechanics, to translate their specifications directly into a tool that can materialize the model. It relieves developers of the need to transform written specifications into a series of logical and physical data models, which is the current practice in data warehousing today. It eliminates the need for successive development of conceptual, logical and physical models. It substitutes iterative “walk throughs” (a tedious exercise using data models to gather approval for domain experts) with a direct path from an abstraction to an application. It replaces data modeling with business modeling.

WHAT IS BUSINESS MODELING?

Business modeling is an imprecise term, but in general, it means creating descriptive replicas of a part of a business such as assets, processes or functions, for example, in terms that are consonant with the people (and processes) that use them. Usually, the goal is to render these models into computer-based applications. In general, a business model is created by someone who has certain knowledge about a process or function in the business. This can range from a single fact, such as how profit margin is calculated, to something as broad as how the manufacturing plants operate.

Spreadsheets and Business Modeling

Spreadsheets are the most common tool for business modeling today. Their popularity stems from their low individual license cost and because it is easy, intuitive and straightforward to insert whatever term, formula or relationship is needed to construct a model. Spreadsheets lack most of the features needed for collaborative, enterprise modeling efforts, however, and are not good candidates for roles beyond personal productivity and simple report sharing, at least at this point in time². They can be quite useful, however, as personal or team-based tools to capturing information needed to construct business models, such as tables of things, pro-forma rules and calculations.

¹ Other terms used more or less interchangeably with domain expert are knowledge workers, business users, users, “the business,” stakeholders and subject matter experts, to name a few. For the purposes of this discussion, they are all interchangeable.

² Microsoft, which dominates the spreadsheet market, continues to innovate with Excel and .NET and may, in time, endow Excel with many of the features that are needed for an enterprise role, but those features have not been deployed as of this writing.

Elements of Business Models

Business models usually contain numbers such as inventory values, salaries or travel expenses. The models need labels that describe or position the numbers such as a point in time or a range, as well as customer names, or locations. Business models also contain rules. These can be very simple, such as flat decision tree rules like "If the customer does not have the original receipt, only give store credit for the returned item." They can be quite a bit more complex, such as "To calculate average annual axle mileage, find the min and max dates of inspections in the calendar year, subtract the greater mileage reading from the lesser and save it as 'actual mileage.' Convert the two dates into day_of_year number and subtract the lesser from the greater and hold the difference as 'actual days.' Divide 'actual mileage' by 'actual days' and save as "actual average daily mileage. Weight all of the remaining days by either 110% for spring and summer seasons and 90% for fall and winter seasons, multiply by the 'actual average daily mileage' and calculate the average mileage for the calendar year summing all of the weighted daily mileages with the 'actual mileage' and divide by 365."

Rules such as the ones above can be applied dynamically when any request is made that requires them, or statically to generate values that remain in a database. In either case, the ability to express these rules in words that make sense to a knowledge worker, and can be reviewed, commented upon and even modified, is a central aspect of business modeling. In the best of cases, these rules can be "abstracted" from any particular application and exist as important objects in the application environment. They are, or at least should be, independent of any one application or piece of technology.

HOW IS BUSINESS MODELING DIFFERENT FROM DATA MODELING?

The end product of a data modeling effort is either a schematic or a script that leads to the formation of a physical database – tables, columns, indexes and the rest. A business model is independent of any database or any data modeling convention. It is as close to the conceptualization of the business in the mind of the knowledge worker as possible. The goal of data modeling is to resolve requirements gathered from others into a format that can be materialized as a relational database. There is often a disconnect between what the business users say and how the data modeler interprets it. It is the opposite of business modeling which captures models and business concepts directly without intermediate steps, people or practices. Data modeling is a discipline that is computer-centric, not business model centric.

THE POWER OF ABSTRACTION

Abstraction is, simply, modeling at the level of understanding and hiding the complexity of the physical implementations that provide the services. In a data warehousing environment, abstraction is useful at the back, front and in the middle.

The Front of Data Warehousing

Data management systems such as relational databases have been expanding their functionality to include areas that were formerly within the domain of data manipulation services, such as statistical tools and BI. The question of where the boundary is between data management and analytics is blurry and getting blurrier. Some databases have embedded OLAP engines, some provide native data mining algorithms and others provide complete statistical capabilities. Still others incorporate search and even text mining for unstructured data. By embedding new "datatypes" into their products, they can still be "relational" and by extending the SQL language with their own added dialect, they can provide a vast amount of functionality beyond the original concept of a relational database.

Overall, these developments are positive. Given the competitive nature of the industry, they are to be expected, and they will continue. But they do pose a problem for the technology stack of BI environment. With each new release, the distribution of processing can change drastically as core functionality migrates from one application to another. The entire ecosystem for analytics, from source systems to presentation tools, is woven together through a series of scripts, API calls and, more recently, SOAP, which mostly depend on knowing who is calling them and whom to call. By redistributing key pieces of processes, this

arrangement can become very brittle, difficult to maintain and expensive. The problem is a lack of abstraction.

Consider a B2C³ site question – “What is the right price to quote this customer?” This can be a very simple question, involving no more than matching the request to a catalog price and adding the appropriate sales taxes and shipping costs. It could be a very complex question, taking into consideration the current demand for the product, the customer’s recent behavior, buying history or even LTV⁴. In the latter case, the data to make the decision and the rules to apply to the data may reside in more than one place, possibly in neither the B2C application itself, nor the application or service that poses the question. Abstraction allows the question to be posed in the terms of the business model, but to be translated and handled by the system or systems that are currently servicing those requests. Abstraction makes it possible to shift the processing from one application to another without the need to modify the service the makes the request.

The Back of Data Warehousing

Managing the extraction and transformation of data from source systems to a data warehouse can be a straightforward process, but usually, it is not. There are problems with the data – missing data, incorrect data, timing differences, to name a few. But there are also multiple steps and at each step, an ETL⁵ tool must understand the target, or landing zone, of the data. In practice, these directions are given to the system as physical coordinates – database objects at specific places. But when examined closely, ETL development is mostly rules-driven and could easily be described in business terms, except that the target schemas are still defined physically. Using abstraction, ETL tools could be pointed to a business model instead, with the abstraction layer handling the actual mapping. In this way, persistent stores of data, whether staging areas, data warehouses or data marts, could be tuned and modified, relocated, re-partitioned or even moved to a different database platform without disturbing the ETL process. Currently, changes in data warehouse design due to new needs, acquisitions or just bug fixes can wreck havoc on ETL programs, causing long delays due to development and testing.

With a business model that can handle changes in organization, products, channels, etc, removes much of the need to embed the business logic in the ETL layer (in addition to the myriad of other places it is captured). Simplifying ETL and putting all the logic in one place is another way to shorten the development time for both new projects and continuing maintenance and enhancement of existing ones. With a business model is the single point of reference. Without it, the logic is scattered and duplicated (and often conflicted) between the ETL layer, the physical database layer and the BI tools’ metadata.

The Middle of Data Warehousing

The key to reduced latency in data warehouses is speeding up the definition of new and/or changed elements and getting those changes implemented as database objects, and populated with data. It is not a simple process. The first step is to understand what is needed, expressed in conversations, reports, memos and email. Next, the designer has to convert these concepts into specifications for the various data warehouse systems. In a mature environment, research is conducted to understand the impact the changes will have on existing systems on the front end and the back end. This process can take weeks to months.

³ B2C is an industry abbreviation for Business to Consumer and refers to websites that sell products directly to consumers as opposed to B2B, or Business to Business sites, such as a corporate procurement website.

⁴ LTV – Lifetime Value of the Customer, is a marketing calculation, whose rules vary by organization, by in essence means the net present value of all future profit in dealing with this customer

⁵ ETL – Extract, Transform and Load refers to a class of software designed specifically to manage the transfer and integration of data from operational systems to data warehouses

If the primary source of information about the data warehouse were, instead, a business model, most of this effort is avoided. Changes are made directly to the business model which searches exhaustively for dependencies that can be affected, reporting them and even rectifying them where possible. In practice, this dependency search is often the most time consuming part of the modification and enhancement effort, so a programmatic way to conduct it is a very large time saver.

When conflicts are discovered, business users work with the model until consensus is reached. Sometimes, small changes in the model can avoid any needed changes in upstream or downstream processes. Other times, the changes are noted and worked into the plan. The scripts, code or instructions --however the Platform Definition service communicates -- are issued and the data warehouse can be reconstructed immediately. The time this requires, of course, is dependent on the design and the scale of the warehouse, but it affects not only the data warehouse, but the staging areas and downstream data marts too.

SHORTCOMINGS OF CURRENT DATA WAREHOUSING PRACTICES

Today, the major hurdles to clear for effective data warehousing are, for the most part, methodology and best practices that have lagged behind both the rapid advances in technology and enormous changes in the business environment. Practitioners draw up paper designs of the architecture, based on framework principles, and arrive at a structure that makes sense from a data architecture perspective, but the complexity interferes with the smooth operation for which it was designed. The problem with these kinds of frameworks is that things tend to work fairly well at first, but as conditions change, either gradually or radically, the approach proves to have the fluidity of poured concrete - only fluid until set. These convoluted architectures are then patched with layer upon layer of new structure, creating an overall design that is too brittle and inflexible to be useful in a BI environment. Each new artifact adds latency to each query and each new patch adds latency to the delivery of needed features. The combination of artifacts creates an increasingly complex structure that impedes or even prevents enhancement or modification.

Ultimately, the expense, delay and dysfunction overwhelm the process. The environments built become too unwieldy to respond to changing business requirements or, for that matter, even the original ones they were meant to support in the first place.

Data warehousing raised the issue of metadata, to its credit, but failed to codify its use or meaning. It is widely agreed that metadata is essential, but a typical data warehouse environment will have a half dozen or more metadata schemas, one for each piece in the chain. The schemas are incompatible and, in most cases, not even actively participating in the development, operation or maintenance of its own area, taking up space as a passive place keeper. There may be metadata in place, but it is not delivering the value it should because there isn't a solid understanding, from end-to-end, of how the data and metadata fit together to form useful applications. The chasm between data warehouse methodologies and how data warehouse information is used is currently too wide for a unified approach. Unifying the business model for the entire data warehousing/BI realm is a necessity, but one that can't be resolved with the current architecture-first practices. A model-driven approach is needed.

Specifically, these architectures are under performing in the following areas:

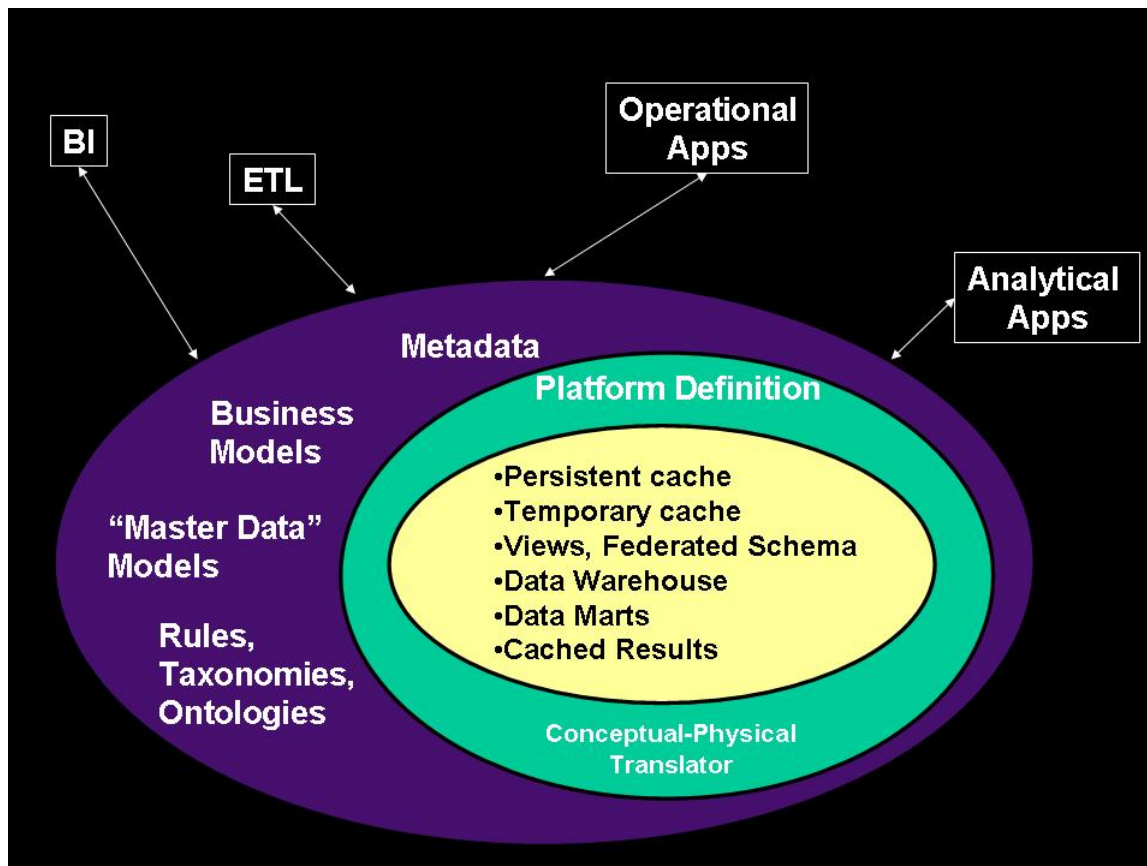
- Reusability of components: many organizations use more than one BI tool, and analytical development cannot be leveraged in another because there is no unified analytical model. Inherent in reusability is consistency of the models. Without a single point of control of the models, inconsistency can only be detected at high cost; with it, consistency can be controlled.
- Limited data abstraction/metadata not performing: the processes access data stores directly to the physical structures, such as tables and columns and views, making it impossible to rapidly

modify physical and even logical models. Each process creates its own model, in technical, physical terms

- Failure to leverage low-cost resources: Without a mechanism to virtualize or federate data stores, data is collected in single repositories
- BI tools and other pieces create a structure that impedes modification because the impact of one change cannot be predicted precisely, leading to long development and testing cycles
- High maintenance costs (software and personnel): Data warehouse environments are inordinately expensive to maintain, especially for ETL, DBA and data modeling professionals.

In addition, the inability of the application to provide the functionality that business needs leads to a significant amount of "Shadow IT," the cost of non-IT professionals building quasi-systems. This can represent a significant, hidden IT cost.

BENEFITS OF THE MODEL-DRIVEN APPROACH



In the figure, the business model is part of an abstraction layer. This layer separates the conceptual definitions from the physical implementations of them. The conceptual and physical objects are connected by the Platform Definition. When users of the warehouse (people and applications) access it, they do so at the conceptual (business model) level, and their access requests are converted to requests to the relevant physical layer in place at the time. In this way, the physical layer can be changed but the access requests from users will continue to work -- they will just generate different physical requests to get the job done. The actual designs of relational databases, multidimensional OLAP cubes or any other structure are completely invisible to the people and processes that interact with the environment. The "metamodels" are created and modified by the people that use them, with the structures adapting

dynamically. This is a radical change from the data warehousing concept, where all modeling was controlled strictly by technologists and how they interpreted "business requirements," a difficult process that has proven to be less than effective in today's business climate.

Some characteristics of this environment are:

- Optimizing the use of resources across the enterprise (and beyond) by managing the materialization and federation (and everything in between) of data and making intelligent decisions of what to materialize and what to define logically, what is persistent and managing the entire process
- Abstraction - All bi-directional data flow must happen through the abstraction layer, not to physical objects like data warehouses, data marts, multidimensional databases, flat files, SAS datasets and a host of other structures that are "read" or "written" to directly in ETL or BI applications . This allows the physical assets of the environment to be tuned, modified, upgraded, relocated or even swapped out for other applications without the need to modify every touch point. Abstraction allows for the changes to be noted in a single controlled environment, while all of the other dependent processes can continue without modification.
- BI software that aids people and processes in constructing queries, analyses and all other forms of information retrieval must no longer query physical objects, such as relational tables or their own structures (cubes, flat files, etc.). This enables all tools to utilize and contribute to the unified analytical model.

Shifting the costs and efforts from a traditional data warehouse to a truly manageable information integration intelligence architecture accelerates ROI through:

- Reduced duplication of effort
- Reduced need for specialists, power users and data-czars
- Improved change management
- Consistency of models, definitions and data
- Sharing of information and technique
- Moving beyond low-hanging fruit to tackle the difficult problems
- Reduced ramp-up time for employees in new positions
- Solving end-to-end problems
- Integrating analytics with operational systems

IMPLICATIONS FOR MDM

Master data, is, quite simply, an agreed-upon set of reference information combined with the tools and engines to manage the definition and use of the data across many applications. Currently, MDM solutions tend more toward the operational, not the analytical, but data warehouses are participating in MDM initiatives nonetheless. Prior to the recent attention given to MDM, data warehouses imposed a sort of master reference data regime by collapsing different data models into a single model, but it wasn't entirely successful. For starters, there may have been a single, central data warehouse, but there were many other models surrounding it, such as staging areas, ODSs, data marts and BI metamodels and persistent cubes. This is the reality of analytical environments.

Using business modeling, harmonizing master data in an analytical environment becomes straightforward. In exactly the same way the abstraction of business modeling orchestrates data management and data manipulation, it can resolve various contexts and dialects of master data from a central source. The implications are very important when one considers how much time and effort is expended in data warehousing researching and resolving data issues.

CONCLUSION

Abstraction is applied routinely to systems that are to some degree complex and especially when they are subject to frequent change. A 2008 model car contains more MIPS of computer processing than most computers only a decade ago. Driving the car, even under extreme conditions, is a perfect example of abstraction. Stepping on the gas doesn't really pump gas to the engine, it alerts the engine management system to increase speed by sampling and alerting dozens of circuits, relays and devices to achieve the desired effect subject to many constraints, such as limiting engine speed and watching the fuel air mixture for maximum economy or lowest emissions. If the driver needed to attend to all of these things directly, he would not get out of the driveway.

Data warehouses and BI tools still rely on at least some of the business users understanding the data models and semantics, if not the intricacies of the crafting of queries. This is a huge barrier to progress. Business people need to define their work in their own terms. A business modeling environment is needed for designing and maintaining structures, such as data warehouses and all of the other structures associated with it. It is especially important to have business modeling for the inevitable changes in those structures. It is likewise important for leveraging the latent value of those structures through analytical work that is enhanced by understandable models that are relevant and useful to business people.