Agile Data Modeling

Not an Option, but Essential

A Technical Whitepaper

Rick F. van der Lans
Independent Business Intelligence Analyst
R20/Consultancy

Sponsored by
IDERA
NEVER. SLOW. DOWN.

Table of Contents

1 Management Summary

The time has passed when data models could be tucked away in an ivory tower (IT) and locked behind bars only for the IT in-crowd to be consulted. It's their data, so, in a way, the data models are theirs as well. For business users a data model is the Rosetta Stone to understand the data correctly. This new situation changes how we develop, maintain, and manage data models. It means that agile data modeling *is not an option anymore, it's essential.*

Everything about data has changed. For example, we are living in the big data era now; new data storage technologies have been introduced, such as Hadoop and NoSQL; and self-service business intelligence has become the preferred approach to analyze data. Data has turned from a simple reporting source for administrative tasks to a critical asset for many lines of businesses. With the right data organizations can optimize their business processes, improve their customer relationships, and differentiate themselves from the competition. With that the dependency of organizations on data has intensified. Data has changed and it has changed the organizations.

The dependency of organizations on data has intensified.

But without a *data model*, data is not very valuable to an organization. A data model describes what data means, what the relationships are, and what the characteristics of data are. With the increasing business value of data, data models are on the radar of business users, while there was a time when data models were only touched by experts in white coats wearing soft cotton gloves while carrying them around. That time is long gone. Business users are accessing and integrating data themselves and don't wait for a Business Intelligence Competence Center[1] anymore, and so they need access to the data models. They need to know what the data they're accessing means, and what all the rules are that apply to the data.

Business users have become involved in the development, maintenance, and management of data models. With this, data models are moving to the highly dynamic business world. Data models should no longer be seen as frozen documents, or as models that are cast in concrete, which can only be changed and extended by specialists. Today, data models have become dynamic sources of information to understand data and this requires a dynamic approach to data modeling. This means that organizations have to adopt *agile data modeling*, which, as indicated, is not an option, but essential.

Adopting agile data modeling is not an option, but essential.

This whitepaper describes the following key requirements for agile data modeling:

- Data storage agnostic
- Collaboration
- Business glossary
- Flexible data model

The whitepaper closes with a section on how *IDERA's ER/Studio* supports agile data modeling.

[1] M. Rouse, *Business Intelligence Competence Center*, July 2010; see http://searchbusinessanalytics.techtarget.com/definition/business-intelligence-competency-center-BICC

2 Everything About Data Has Changed

Data is available in abundance.

Over the last few years, everything about data has changed. The sheer amount of data processed and stored by organizations has sky-rocketed, the value of data to organizations has dramatically increased, the dependency of organizations on data has intensified, the technology for storing data has revolutionized, the self-service style of working has changed from being special to commonplace, the required accuracy of data has changed from a few days to a few seconds, and so on. Data is available in abundance and is playing a pivotal role in business operations. However, not every organization is aware of this change, or is in some form of denial.

We live in the big data era.

From Small Data to Big Data – No one was aware of it at the time, but years ago we lived in the *small data era*. Of course, some organizations were storing terabytes of data, but they were the exception to the rule. The amount of data stored by most organizations did not exceed 1 terabyte. Especially in the last few years, this has changed enormously. For example, new production systems store much more detailed data than before, websites generate massive amounts of weblog data, social media networks and public data sources contain millions and millions of records, and so on.

Today, we clearly live in the *big data era*. For example, Ford's modern hybrid Fusion model generates up to 25 gigabytes of data per hour[2]. The video game industry is tracking and analyzing over 4 terabytes of operational logs each day[3]. These data sizes were unthinkable ten years ago. Now, they are considered the norm.

Besides offering many new analytical opportunities, big data also introduces new challenges. For example, what's the best and most efficient way to backup big data, is storage of big data costly, how do applications handle variable data structures in big data, and how can unstructured data be analyzed? Organizations must find new solutions for all these typical big data issues.

New Data Storage Technologies – Lately, several new data storage technologies have become available, including Hadoop, and NoSQL products, such as MongoDB, Cassandra, and RavenDB. Most of them have been designed to support big data systems. They excel at ingesting massive amounts of data quickly, at processing large numbers of concurrent transactions, and at analyzing large amounts of data.

New storage technologies don't support SQL and relational concepts.

There's only one caveat. To be able to get to these high levels of performance and throughput for a reasonable price, these products had to let go of the familiar SQL and relational concepts. Most of them support their own proprietary database language and their own database concepts, which are not always easy to translate to relational or SQL concepts. This makes it harder to exploit all that big data using most of the traditional tools for reporting and analytics, ETL, and also data modeling.

[2] http://www.datanami.com/2013/03/16/how_ford_is_putting_hadoop_pedal_to_the_metal/
[3] http://news.dice.com/2012/12/07/for-riot-games-big-data-is-serious-business/

Applications Come And Go – Applications come and go, but data is here to stay! Over the years, this has become increasingly clear. Look at some of the databases of financial organizations. The initial versions were developed over thirty years ago and they are still operational. These companies heavily depend on the data collected a long, long time ago. Since then, more data has been added, more attributes have been added, but in principle it's still the same data.

Applications come and go, but data is here to stay.

In the meantime, new applications have been developed to make use of the same data. For example, new applications have been developed to allow users to access the data via their smartphones and tablets, and over the internet, so that they wouldn't be limited to access the data on their desktops in their office. The same applies for customers and consumers, who needed applications on their smartphones and on the web to access data.

New applications have been developed, old ones have been phased-out, and existing ones have been changed, changed, and changed again. But at the core of every IT environment stands the data. Data has become the lifeblood of many organizations, and is here to stay, whereas applications are like passers-by.

Data in the Hands of the Users – In 1990, Bill Gates gave his famous keynote address[4] at the Comdex Fall event titled "Information at your fingertips." These four words summarized Microsoft's long-term goal: make any piece of data easily and readily available to users. This stage has been reached. With all the new *self-service BI tools*, such as Qlikview, Tableau, and Spotfire, users can access any kind of data source, they can integrate them (sometimes called *data blending*), they can even transform and analyze the data themselves without any involvement of IT specialists.

Compare this to years ago, when data was guarded by IT, and IT would determine how and when users could access the data. In the BI and data warehousing world, users don't wait for the *BI Competence Center* (BICC) anymore. They prefer the *do-it-yourself approach*. They don't want to be limited to having access to a data mart or the data warehouse, they want to dig into operational databases directly.

Data modeling is heavily influenced by the changing world of data.

Summary – The world of data has drastically changed. Today data is available in abundance, new database storage technologies are deployed, and data is in the hands of the users. All these changes have numerous consequences for several related topics, and one is *data modeling*.

3 The Business Value of Data Models

Data modeling is not a new topic. The first articles on data modeling were published more than 50 years ago in the second half of the 1960s[5]. Over the years, data modeling techniques and notation techniques have evolved and matured, although some modeling principles remain unchanged.

[4] B. Gates, *Information at Your Fingertips*, November 1990; see https://www.youtube.com/watch?v=kL8zoQVJaD8
[5] C.W. Bachman, *Data Structure Diagrams*, in Database: A Quarterly Newsletter of SIGBDP, vol. 1, no. 2, Summer 1969.

Why Data Models? – The result of data modeling is a *data model*, which is a description of the data. It describes what data means, what the relationships are, what the characteristics of data are, and so on. Without a data model, data has no real value. It's like having a vault loaded with gold, but you've lost the key. For example, the following data <John, December 15, 1999> can mean anything. Was John born or did he get married on that date, or is it purely a date indicating when John was entered in the system? By the way, is John a person or maybe an animal? It's only when a description is added that it starts to make sense. When it's explicitly indicated that John is the first name of an employee, and that the date December 15, 1999 indicates when he joined the company, then data makes sense. And that's exactly what a data model does. Data without a data model is like words without a dictionary. Maybe we can guess what the words mean, but it's gambling with data.

Data modeling is about understanding data.

Data Modeling – If a data model is about describing what data means, then *data modeling* is the process that leads to data models; data modeling is about understanding the data and formalizing that understanding in the form of data model diagrams, definitions, descriptions, pictures, and so on. Wikipedia[6] defines data modeling as follows: "Data modeling in software engineering is the process of creating a data model for an information system by applying formal data modeling techniques."

Does NoSQL Make Data Modeling Superfluous? – Some of the new data storage technologies are much more flexible with respect to data structures than the more classical SQL database servers. For example, each record in a table can have a different structure. This means that a record can be inserted in an existing table even if that new record contains more columns than the other records in the table, or if it has a slightly different structure. Doing something similar with more classical technologies is a time-consuming exercise, that may involve an unload of the entire table, a redefinition of the table (adding a column), and a full reload. This can easily take hours when dealing with large data volumes, and in the meantime data is inaccessible for users. In other words, NoSQL products support *variable data structures* for their tables, while the older products support *static data structures*.

Also, some of the newer products support *schema-on-read*. Schema-on-read means that data is stored without the data storage technology knowing the structure of the data. It stores all the data as one long byte string. The applications, on the other hand, do understand the data and they assign a data structure when the data is retrieved. On the database side this is more flexible. Almost never is an unload/reload of data required, and different applications can view the same data with different structures.

NoSQL supports variable data structures and schema-on-read.

Some may think that concepts, such as variable data structures and schema-on-read, make data modeling superfluous. Utterly false, data modeling is needed even more. Regardless of how data is stored, applications need a data model to understand the retrieved data. The new data storage technologies make the data model even more important than it already was. Data storage flexibility does not imply that no data model is needed.

NoSQL makes the data model even more important.

A Data Model is More Than a Diagram – Each IT department still has at least one room with a wall that is decorated with a large diagram on yellowed paper. The diagram is full of faded boxes and countless connecting and dissecting lines. At the top it says that the diagram represents the data model of some

[6] Wikipedia, *Data modeling*, March 2013; see http://en.wikipedia.org/wiki/Data_modeling

past IT system. It's still on the wall, because it covers some nasty stains. It's only when the room must be redecorated or if the wall has to be torn down that the diagram is removed.

Many still associate a data model with such a diagram. It's important to understand that a data model is much more than a diagram. A diagram cannot fully describe and define all the aspects of the data. In addition to the diagram, a data model contains:

- Textual definitions
- List of synonyms
- Textual descriptions
- Integrity rules related to data quality
- Business glossary
- Security rules

Data Models for Business Users – There was a time when data models were only touched by experts in white coats wearing soft cotton gloves while carrying them around. The data models were kept in a vault that Fort Knox would have been jealous of. Users would never ever get to see these models. That time is over. Because users access and integrate data themselves, they need access to the data models. They need to know what the data means and what all the rules are that apply to the data.

Because users are more directly involved in determining what data they need, they may also be involved in developing and maintaining the data models. And if they're accessing external data sources, they are fully responsible for developing the data models that describe the meaning of these new data sources.

Data models must not be locked away in the ivory tower (IT)

The time has passed that data models reside in the ivory tower (IT). IT and the business users have equal rights to the data models. Because it's their data, the data models are theirs as well.

A Data Model is Pivotal – It's no discussion anymore, data plays a crucial role in organizations. Therefore, data models have become crucial. Without proper definitions and descriptions, users are lost, and subsequently the business is lost. It would be like a processing plant that receives all the ingredients in non-labeled cans.

For many reasons the data model is crucial:

- The data model is crucial for users to develop their own reports, because they must understand the data.

- The data model is crucial for users to analyze reports and results. If sales are up according to a report, the data model can indicate whether tax is included in the sales figures, whether it applies to sales of all the products or just a specific subset. The data model defines how currency conversions are applied to calculate the total sales, and so on.

- The data model is crucial for developers to develop new big data applications to understand the structure of the incoming sensor data, and all the exceptions that can occur.

- The data model is crucial for BI specialists to integrate data from multiple systems to present a correct 360 degrees view of customers.
- The data model is crucial for data stewards to determine the correctness of data.

Without data models, organizations are data-rich, but still information-poor.

Summary – Data models have become pivotal for many within an organization. Everything revolves around the data model. Without data models, organizations are *data-rich*, but still *information-poor*.

4 Agile Data Modeling

In the Beginning – Traditionally, data models were developed by IT specialists and by no one else. Business users could validate them and comment on them, but the IT specialists were responsible for them. Complex and rigid procedures existed to maintain and change existing data models. These data models were created for large systems. The data model had to be ready before any form of functional or technical design and implementation would start. Nowadays, this is called the *BDUF approach* (Big Design Up Front).

In most cases, these were mission critical systems for the organization. Changing them could only happen under the strictest procedures and controls. The same applied to the corresponding data models. These could only be changed or extended when approved by several specialists. Therefore, the models were quite static.

Static Data Modeling – Static data modeling won't work anymore, data models and data modeling must be *agile*. The world of data has become extremely fluid, so data modeling had to follow in its footpath. Agile data modeling is not an option anymore, it's essential.

Agile data modeling is not an option, it's essential.

Agile in the context of data modeling can mean several things. Agile can refer to the agility of the resulting data model itself; it can refer to the way data models are developed, in other words to the process itself (agile or BDUF); and it can refer to how the models are developed and maintained (is everyone allowed to maintain them?). In this whitepaper, agile data modeling refers to all three.

Agile Data Models – In principle, whether a data model was created twenty years ago or last week, it's agile. Regardless of how the data model is recorded, whether on paper in a binder or in a data modeling tool such as ER/Studio, making changes is easy. If the *information needs* change, making a corresponding change in the data model can be done in minutes or hours. A data model may be somewhat non-agile when it's chiseled in a large concrete tablet. Then, changing it requires muscle power, a chisel, a hammer, and lots of patience.

The real challenge, however, is that there is an entire IT system developed based on this data model. A change of the data model could mean that the data structures of operational tables must be changed, which involves an unloading and reloading of all the data; derived data structures in data warehouses and data marts have be changed accordingly; ETL programs must be changed; applications responsible for

inserting and updating the data must be altered; and so on. Even a minor change of the data model can lead to an avalanche of changes throughout various systems. This can be very costly.

An agile data model has been designed in such a way that when new information needs arise, the changes to the data model are minimal or negligible, and that, therefore, the changes to be made to the IT systems are minimal as well. For example, data models developed according to the *data vault* principles are easy to extend and do not lead to changes in the current system. See the material by Dan Linstedt[7] and Hans Hultgren[8]. Also, by developing certain structures, a model can be more agile as well. More on this in Section 8.

Data models should not be regarded as frozen documents, or as models that are cast in concrete. On the contrary, data models have become dynamic specifications that can change continuously. In a way, data models are much more like cities. Cities are never finished, they continuously grow, change, adapt, and so on.

Data models must not be cast in concrete.

Agile Data Modeling – The processes used by organizations to model data can differ enormously. As indicated, in the old days, data modeling was a highly structured and procedure-rich approach. Today, a much more dynamic approach is required. For example, self-service users want to be involved in defining and modifying the data models. It has become a much more dynamic process. In the coming sections the various aspects and requirements of agile data modeling are described.

Data Modeling Tools – It's impossible to support all the requirements for agile data modeling when models are only available on paper. A professional *data modeling tool* is indispensable. Such a tool allows multiple users to work on the same data model concurrently, it is familiar with the new data storage concepts, and it supports collaboration features. Without such a tool, everyone involved in data modeling is handicapped.

5 Agile Data Modeling is Data Storage Technology Agnostic

The first requirement for agile data modeling is related to all the new data storage technologies available.

From SQL to NoSQL – There was a time when SQL-based database servers were dominating the market. Almost every IT system used a SQL database server. Not anymore, the times have changed, especially the last few years. So called *NoSQL database servers*, such as MongoDB, Cassandra, and Apache HBase, have received a lot of attention and have been successfully implemented by numerous organizations. The products are used in prototypes but also in mission-critical systems. According to DB-Engines[9], MongoDB is currently the fourth most popular database server after Oracle, SQL Server, and MySQL.

What's special about these NoSQL products is that they don't organize the data in relational tables consisting of records and columns. In NoSQL, data can be organized in hierarchical structures. Also, not all the records (read documents) of the same table have the same structure. In fact, each record of a table can have a (slightly) different structure.

[7] Dan E. Linstedt, *Data Vault Series 1 - Data Vault Overview*, July 2002, see http://www.tdan.com/view-articles/5054/
[8] H. Hultgren, *Modeling the Agile Data Warehouse with Data Vault*, Brighton Hamilton, 2012.
[9] DB-Engines Ranking, March 2015; see http://db-engines.com/en/ranking

The reason why these products have become so successful is that they can support application requirements that most existing database servers can't. For example, some can process more transactions, others can handle a larger ingestion rate, some improve developer productivity, and there are those with a lower price/performance ratio. This doesn't mean that they will replace SQL database servers, it's just that for specific applications they have something unique to offer.

Polyglot Persistence – The consequence of all the new products is that each organization is slowly moving to a *polyglot persistent* environment. Polyglot persistence means that different data storage technologies are used by different applications. The one that has the best price/performance ratio for a specific workload is deployed; see Figure 1. For example, Riak may be used for weblogging, Hadoop to offload cold data warehouse data, and MongoDB to store and analyze sensor-based data.

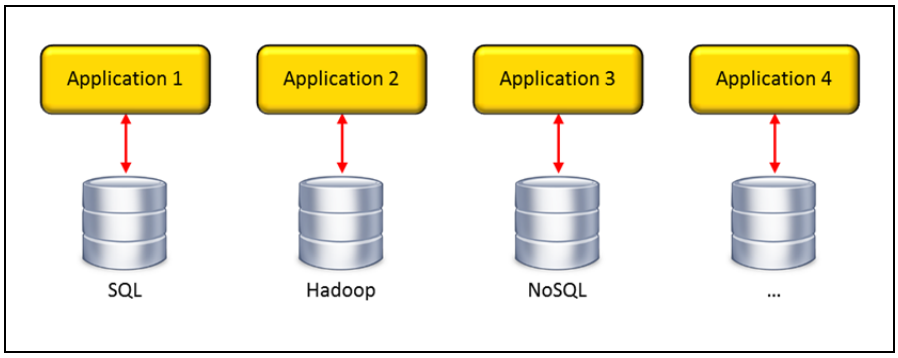

Figure 1 *Different data storage technologies are used to support different applications.*

Data Storage Technology Agnostic – The consequence for data modeling is that it must be *data storage technology agnostic*. The assumption that all databases are developed with SQL is not valid anymore. Data modeling must be agile enough to support any data storage technology and their concepts, including NoSQL and Hadoop. This means that a data modeling tool must support the following features:

- **Rich set of data model concepts:** Support for NoSQL starts by allowing data modelers to use a larger palette of modeling concepts, especially the ones supported by the NoSQL products. For example, many of them support the concept of a *collection*. A collection can indicate a hierarchical relationship between concepts. For example, a publisher contains an address attribute and that address consists of a city name, state name, street name, and zip code. Here, a hierarchical relationship exists between publisher address and city name. Figure 2 contains a data model that shows the use of such NoSQL concepts in a data model. The blue squares indicate so-called *embedded collections*.

- **Forward engineering:** It must be possible to generate NoSQL data structures based on the extended data models; see Figure 3. In fact, it must be as easy to generate data structures for NoSQL as it is for SQL.

- **Reverse engineering:** It must be possible to drive the data model from existing NoSQL databases. In other words, the data modeling tool must be able to reverse engineer an existing MongoDB data structure to a logical data model. As an example, the diagram in Figure 2 has been reversed engineered from a MongoDB database using ER/Studio.

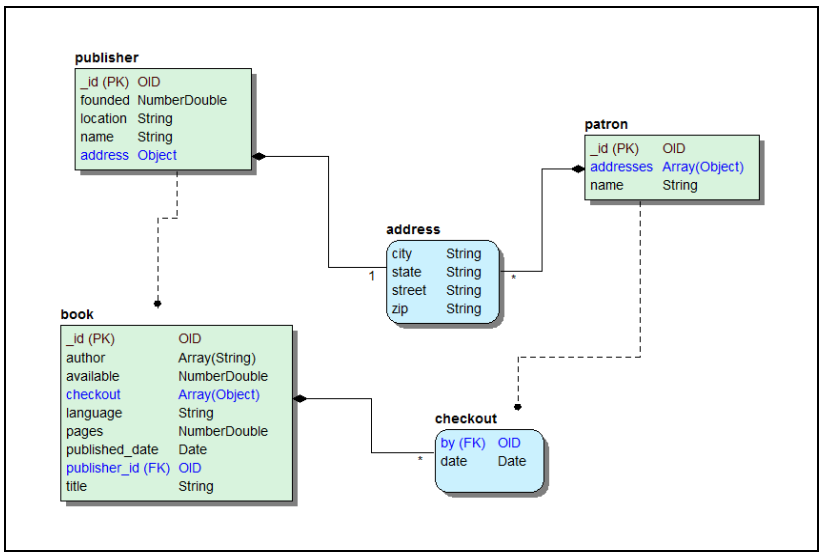

Figure 2 *This data model contains typical NoSQL concepts. The blue boxes represent embedded collections that can be used by several entities.*

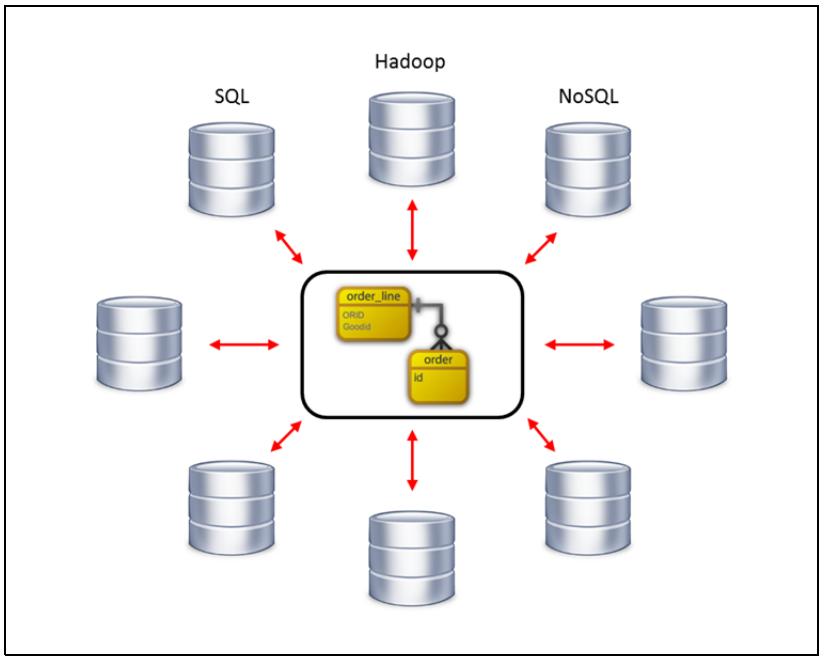

Figure 3 *A data modeling tool must be able to reverse and forward engineer NoSQL databases.*

6 Agile Data Modeling is Collaboration

A Collaborative Exercise – As indicated, there was a time when only IT specialists developed and maintained the data models. Not anymore. Business analysts, data stewards, business users, and data architects/modelers are all involved. They want to change models, extend them, and they all want to have a say in it. Development and maintenance of data models is a collaborative exercise involving the following stakeholders:

- **Data stewards:** The term *data steward* is defined by Wikipedia[10] as follows: "A data steward is a person responsible for the management of data elements - both the content and metadata." Data stewards deal with definitions and data quality. Therefore, they must have influence on the data structures and definitions residing in the data models. Data stewards can be initiators of changes, such as a definition change or a change of the allowed set of values of an attribute. It would be counter-productive to let the data stewards first inform the IT specialists of the intended changes, followed by a long explanation, and then the specialists come up with a proposed solution that they discuss together leading to some additional changes, and finally the final change is implemented in the model. This can be a long project. With agile data modeling data stewards must be able to make these changes themselves and inform the IT specialists about them.

- **Business analysts:** Business analysts have a very close relationship with the business and the users. If the business changes and if users' needs change, they usually are the first to hear it. Again, as with data stewards, business analysts must be able to work on the data models.

- **Business users:** Business users must be able to add descriptions, notes, alternative definitions, abbreviations, synonyms, and must be able to redefine a definition if a concept has changed. For example, the introduction of a new law or regulation may influence the definition of a concept. Business users must be able to introduce that new definition. There should be no need to wait for the IT specialists to make these changes.

- **Data architects /modelers:** This group has always been the traditional group of specialists involved in developing and maintaining the data models. This won't change, but increasingly they'll work in cooperation with one of the other functions.

There has been a shift of responsibilities. IT specialists used to be in the driving seat with respect to data models. It will be more and more the business users and analysts and the data stewards who will initiate changes and amendments to data models.

The business is more and more in the driving seat with respect to changing data models.

Versioning of Data Models – Collaboration demands *agile change management*. It must be easy to change data models and experiment with data models. Agile change management requires that data modeling tools support *versioning of data models*. Agile data modeling can't work without versioning. Business analysts, data stewards, business users, anyone, must be able to experiment with a data model. They must be able to make changes and add new aspects to see what the impact is on the entire data model. But such an experiment should never override the current data model. Making permanent changes requires consultation with all the stakeholders.

As an example, Figure 4 shows that two versions of the Sales Order System data model have been created. They are specified behind the symbol and are called version 1 and 2.

[10] Wikipedia, *Data Steward*, March 2015; see http://en.wikipedia.org/wiki/Data_steward

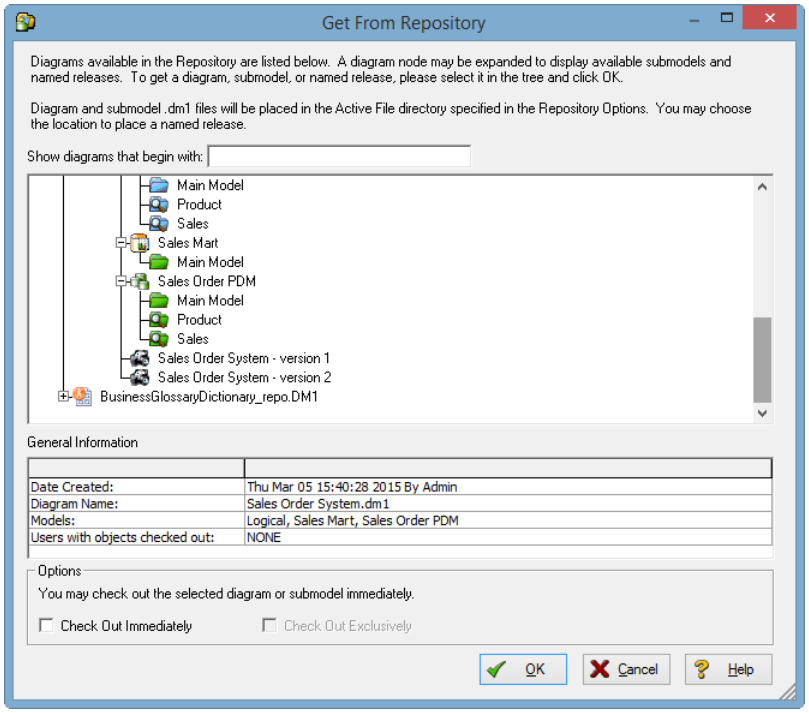

Figure 4 *Versioning of data models is a prerequisite for collaborative data modeling.*

An aspect of agile change management is that versions can be compared to identify the differences. Figure 5 indicates two differences between versions 1 and 2 of the Sales Order System data model. First, the length of the ProductNumber attribute has been changed from 25 to 30 characters, and second, the attribute BrandName has been added in version 2.

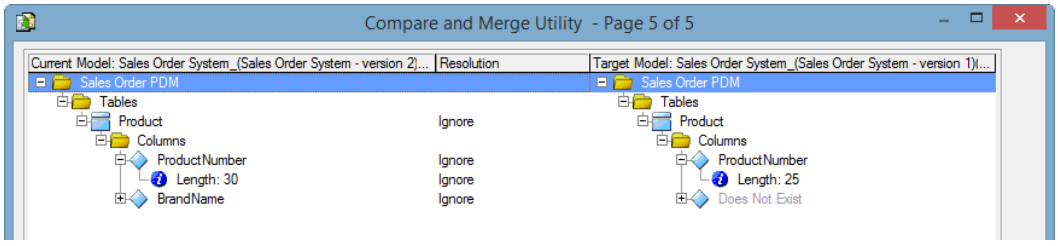

Figure 5 *Data modeling tools must allow different versions of data models to be compared.*

Comparing data model versions is an important aspect of agile change management and is therefore an important feature of data modeling tools. This feature must make it possible to *check-out* a data model to make changes to it. After these changes have been studied and discussed, and when everything is correct, the model can be checked back in.

Fine-grained Checking-Out and Checking-In of Data Models – Because more and more people are involved in the continuous development of data models, it would be unpractical that every time someone wants to make a change or add something, the entire model is *checked-out*. It would block all other users to experiment with other parts of the model. A *fine-grained check-out* (and check-in) mechanism must be supported by data modeling tools. For example, it must be possible that a data steward checks out the definition of an entity to refine its definition and that at the same time a business analyst changes the set of permitted values of an attribute. This demands a fine-grained form of checkout. Else, a change of one user blocks out the entire data model for all other users. This was acceptable when data models were owned and maintained by a limited number of IT specialists, but not anymore.

7 Agile Data Modeling is a Business Glossary

A Data Model is an Online Source of Information – A data model is no longer a passive description that is only studied when an IT system must be changed. It has become an active source of information for a large group of users. Parts of a data model may be investigated by hundreds of users every day.

The data model is an active source of information to business users.

Now that more self-service users develop their own reports and access all kinds of data sources directly, data models must be available to them like any other online source of information. Users that create their own reports must develop their own integration logic when data from multiple data sources must be combined, they must prepare their own data, and they must deal with the cryptic and technical table and column names assigned by IT. For business users it's not obvious that a table called `DMW_SAL_23` contains sales data for the North-American stores. Access to the data model explains to them what it all means. It's their Rosetta Stone[11].

Without access to a data model, the following business problems may arise:

- Because users have access to so much data distributed over so many data sources, they may not even be aware that certain types of data are available. This may lead to missed and unexploited business opportunities. Users do not and cannot analyze data they have no knowledge of.
- Users may interpret the data incorrectly, and thus develop the wrong integration logic and reporting logic. This can lead to erroneous report results.
- If data is "hidden" in a large set of tables, it may take users a long time to locate the right tables for their analysis, and a long time to convince themselves that they are really using the right tables. In the process, they are losing valuable business time.
- Data structures may be so complex that users can't locate the right data, although they know it exists.
- Users may not be sure of what particular data values represent.

The Business Glossary – To have access to the typical concepts of a data model diagram can be useful to users. It tells them how the tables are linked together, the type of data that the tables and columns contain, what each record means, and so on. Still, such a diagram only helps if users know what they're looking for.

Self-service BI users may be logging on to several data sources and this may potentially give them access to literally hundreds of tables. The effect may be that they have no idea where to start; it may stifle them. Self-service users need a starting point. They need to be able to search for business objects, such as patient, factory spill, and part-time employee. For example, when they search for the word sales, they should receive a list of all the tables containing sales data. Next, descriptions and definitions formulated in plain English, and not in IT-English, should tell them which table they need for their analysis.

[11] Wikipedia, *Rosetta Stone*, March 2015; see http://en.wikipedia.org/wiki/Rosetta_Stone

A business glossary contains a non-technical description of business objects.

This is where the *business glossary* comes in. A business glossary contains definitions and descriptions of business objects. It describes relationships between the business objects and contains categorizations and classifications of business objects, and all this in plain English. A business glossary is not supposed to contain names such as `DW34EMPL` or `DM20DEP`, nor technical and cryptic definitions, such as "A record in the `DW34EMPL` table contains employee data containing a unique employee number of 15 bytes long, and address data." This is too vague, too technical, and far from descriptive. The following definition is more in line with what users expect to find in a business glossary: "An electronic health record (EHR) is an electronic record of health-related information on an individual that conforms to nationally recognized interoperability standards and that can be created, managed, and consulted by authorized clinicians and staff across more than one health care organization[12]." As an example, Figure 6 shows the business object called *Customer Lifetime Value* including its definition, abbreviations, and additional notes.

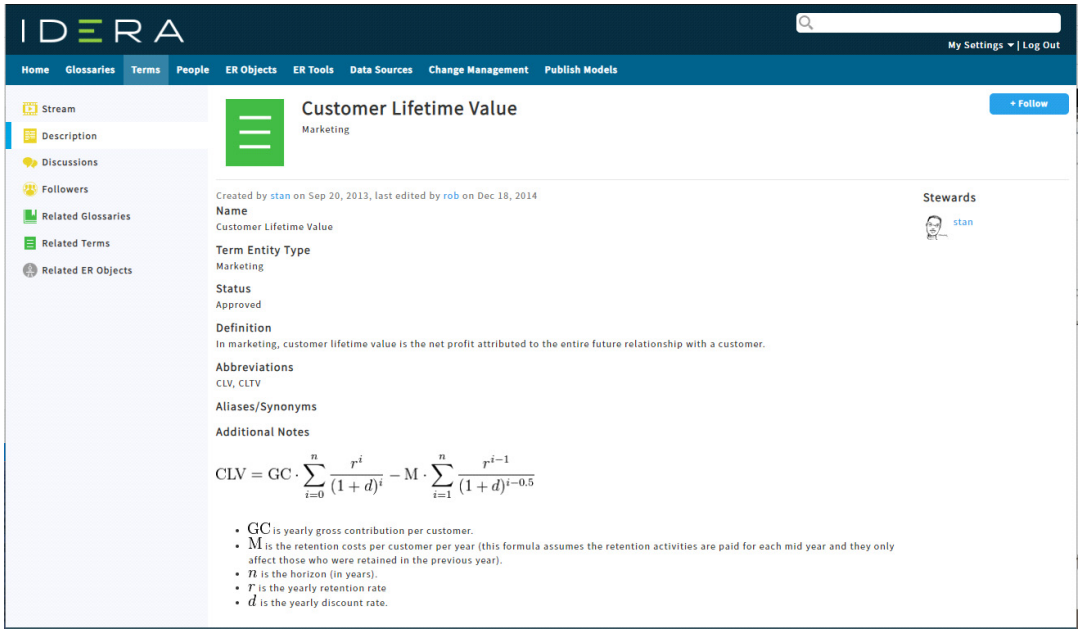

Figure 6 *A full definition and description of the business object Customer Lifetime Value.*

Searching the Business Glossary – The most important feature of a business glossary is *search*. In a simple way, users must be able to search for the right business objects, the ones they need for their reports and analysis. If they are interested in, for example, customer lifetime data, simply typing in the words customer life must be sufficient for the business glossary to show all the business objects that are somehow related to this term, as shown in Figure 7.

A business glossary offers an extensive search feature.

[12] See http://www.hitechanswers.net/key-definitions/

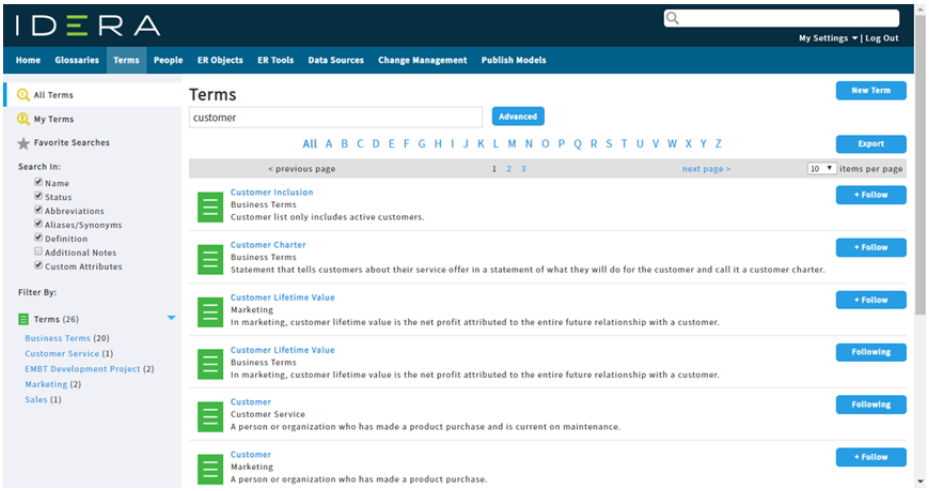

Figure 7 *The result of a search on the "Customer" leads to a list of defined terms that includes Customer Inclusion, Customer Charter, and Customer Lifetime Value, among others.*

Note that the quality of a search result is fully dependent on the quality of the business glossary itself. Therefore, keeping the glossary up to date, is crucial.

An Integrated Business Glossary – A stand-alone business glossary can be very valuable to an organization, but an *integrated business glossary* even more. Integrated means that the business glossary is somehow integrated with the data model concepts, definitions, and descriptions, and also with the data sources. When users have discovered the business objects they're looking for, the business glossary must be able to point them to the related data model concepts and the implementation concepts as well as to the data sources where the data is stored. As an example, Figure 8 shows that the business term Customer Lifetime Value links to the column DateFirstPurchase within the table DimCustomer in the data model for Adventure Works DW. In other words, the integrated glossary guides users from the business world seamlessly to the right data sources and data models. This allows users to develop reports with the right tables and columns.

An integrated business glossary guides users to the right data sources.

Especially for business users, it's important that they can start their analysis of data by using their own business terminology. It's not wrong to assign a name to a table in a data warehouse that is according to a very rigid naming convention. The problem is that these names may be too cryptic for business users and may make a search for the right data difficult and maybe even impossible.

Integration of a business glossary and the data models and data sources is indispensable in an agile world, in which more and more data becomes available, in which data sources change more and more quickly, and the business must respond more and more quickly to business opportunities and challenges.

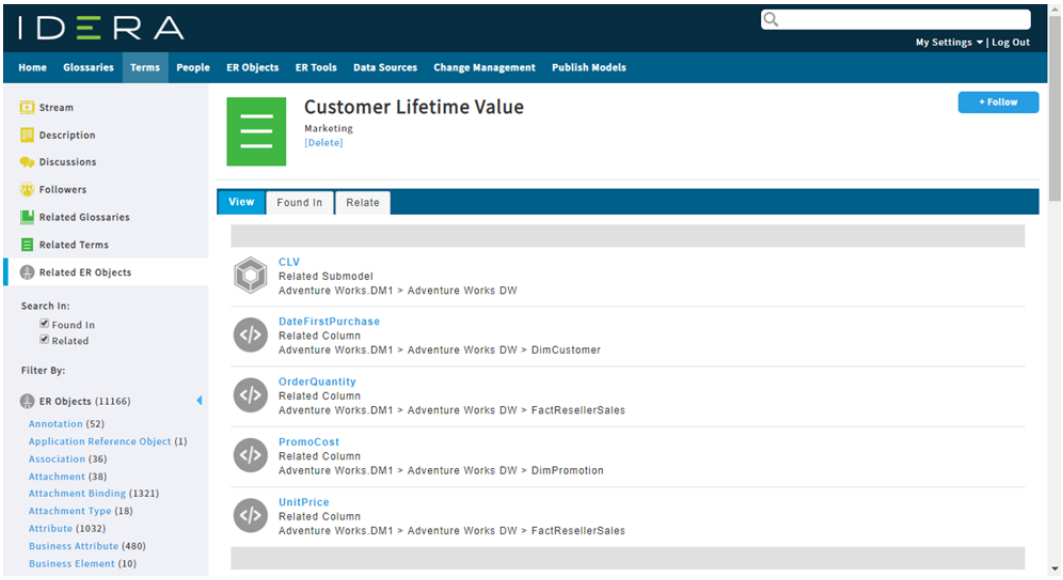

Figure 8 *With an integrated business glossary business objects are linked to the data model concepts and the data sources.*

8 Agile Data Modeling is a Flexible Data Model

Agile data modeling also requires that the following aspects of the data models are flexible: the data modeling notation technique, the data modeling rules, and the data modeling principles.

Data Modeling Notation Technique – Many different data model notation techniques exist. Figure 9 shows four alternative notation techniques. In some cases it's purely the symbols that are different, but some use unique modeling concepts. Users must be able to use the notation technique they prefer, the one that fits best with their job. Data modeling tools can assist by allowing the same data model to be presented with different notation techniques.

Users must be able to use their preferred notation technique.

In addition, different users may want to see different aspects of a data model. For example, IT specialists may want to see all the technical details of a data model, including data types and foreign keys, whereas business users may be much more interested in a data model that shows as many business aspects as possible, such as the definition and description. Data modeling tools must be flexible enough to show different views of the model depending on the person who studies it.

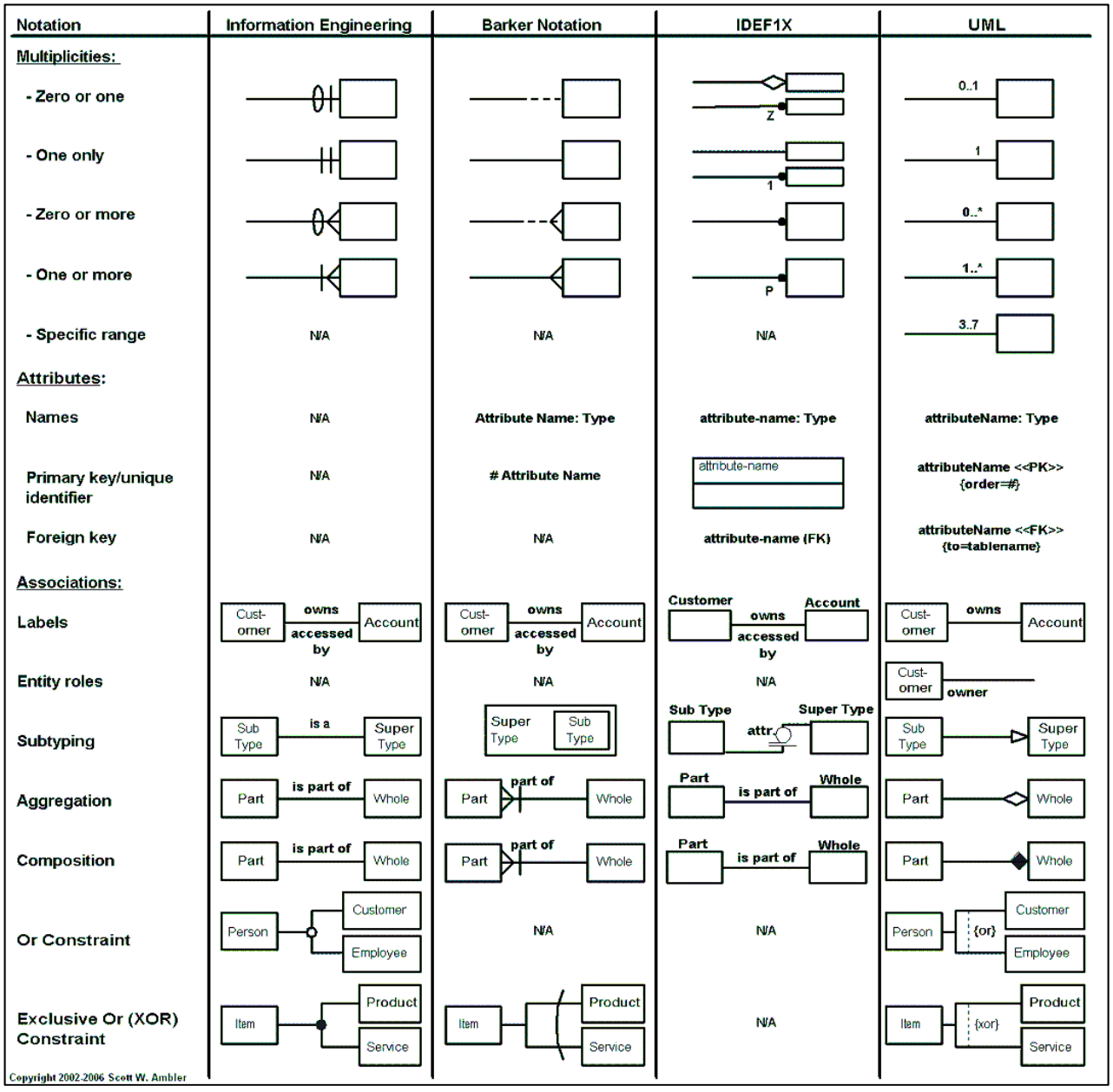

Figure 9 *Several different data model notation techniques exist*[13].

Data Modeling Rules – Different rules can be applied when developing data models. For example, if a classic ERM data modeling notation technique is used, there's no guarantee that the same information needs always result in the same data model. This depends on the modeling rules. For example, if the rules for star schema design are applied, the final model will look different than when the rules belonging to the data vault approach are deployed. The same notation technique is used, but different data modeling rules apply.

[13] Scott Ambler, *Data Modeling 101*; see http://www.agiledata.org/essays/dataModeling101.html

A data modeling tool must not enforce a specific set of data modeling rules, in other words, it should allow all users to apply their preferred rules.

Data modeling tools must not enforce specific data modeling rules.

It's important that data models are flexible so that new information needs can be added to the model easily with minimal (preferably) changes to the rest of the model. For example, in the data vault approach the data structures are *over-normalized*. The effect is that new information needs never lead to a change of an existing part of the model, but always to an extension of the model. The big benefit is that not only the data model that can be changed easily, but the operational database that's derived from it, as well. The applications for which the change is irrelevant probably also remain unchanged.

Data Modeling Principles – But whatever notation technique and set of modeling rules is used, whether the data model itself is really agile, depends for a large part on how the building blocks in the data model have been organized, or in other words, what the *modeling principles* are. Let's illustrate this with a simple example.

The diagram on the left hand side of Figure 10 represents a small part of a data model for an old-fashioned library. In this library, books can be loaned to members and members are categorized as junior or senior members. The annual subscription is currently fully dependent on the member category. Junior members pay $75 per year and senior members $125. The left hand side of Figure 10 is designed to deal with these information needs. But imagine that new rules are added. For example, all new members who subscribe in the coming three months get a 10% discount. This is hard to model in the tables derived from this data model. An artificial solution is to introduce two new member categories: junior members with a 10% discount and senior members with a 10% discount. But imagine another new rule is introduced: members who are also member of another library in the city, get a 25% discount. Eventually, we must accept that the initial model is not agile enough. The model must be changed. Unfortunately, it's not only the model that must be changed, but the database and the applications as well.

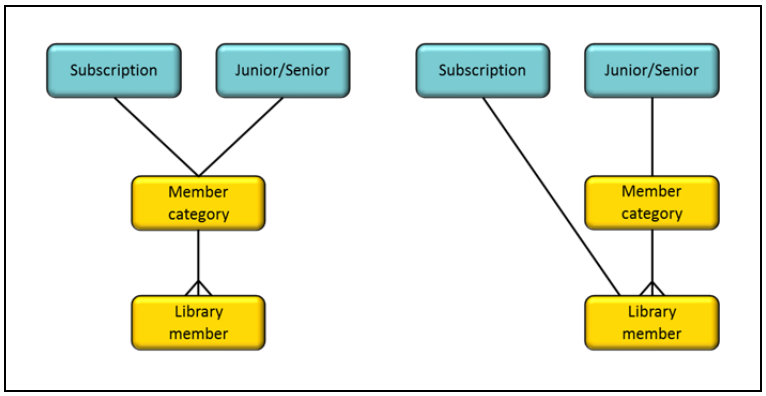

Figure 10 *Applying different sets of data modeling principles can lead to different data models where one is more agile than the other.*

The model presented on the right hand side of Figure 10 is a better and more agile solution. Here, the annual subscription and the member category have been decoupled. Now, any annual subscription can be modeled, regardless of the member category. When in this situation the rules change, the data model doesn't have to be changed nor the database. Maybe the applications must be changed a little. This is clearly a more flexible data model. More changes of information needs can be implemented without drastic changes to the existing systems.

Modeling principles have a major impact on how flexible a data model is, and indirectly they determine how flexible the databases and applications are.

9 Agile Data Modeling with ER/Studio

The data modeling tool *ER/Studio* by IDERA was first released in April 1996. It started out as a simple tool that allowed data model diagrams to be entered and maintained by IT specialists only. It was designed during and for the small data era. Since then ER/Studio has evolved. Most of the features that have been added over the years have made the tool more suitable for agile data modeling. This section describes briefly its agile data modeling features.

Data Storage Technology Agnostic – From the beginning ER/Studio has always been data storage technology agnostic. So, the product now supports forward and backward engineering to one of the most popular NoSQL products MongoDB. For this, it has extended the data modeling concepts; see Figure 2. It also supports Hadoop Hive and several other big data platforms through the *MetaWizard Bridges*.

Collaboration – ER/Studio supports agile change management through versioning of data models. With the latest version, small components of a data model can be checked-out and checked-in allowing large groups of users to work on the same data model. The changes can also be associated to tasks and/or user stories. Data models can be compared to locate, for example, the differences between two versions of a data model; see Figure 5.

Business Glossary – Business glossary functionality has been implemented in IDERA's product called *Team Server*. It allows business objects to be defined, described, and categorized; see Figure 6. In addition, Team Server offers an integrated business glossary. Business objects can be linked to data source concepts, such as tables and columns; see Figure 8. Its extensive search feature is not restricted to searches on the name only. A search for a term includes a search on the name, the definition, the available notes, the abbreviations, and so on; see Figure 7.

Flexibility – ER/Studio supports different data model notation techniques. Figure 11 shows how users can switch from one notation technique to another by selecting the preferred one. They can choose between IDEF1X, and three different forms of IE. In addition, they can indicate which aspects of the data model must be included; see the `Available Options` in Figure 11. For example, Figure 12 shows a data model that primarily contains the objects, their definitions, and their relationships, whereas Figure 13 shows the same data model, but now including all the technical details.

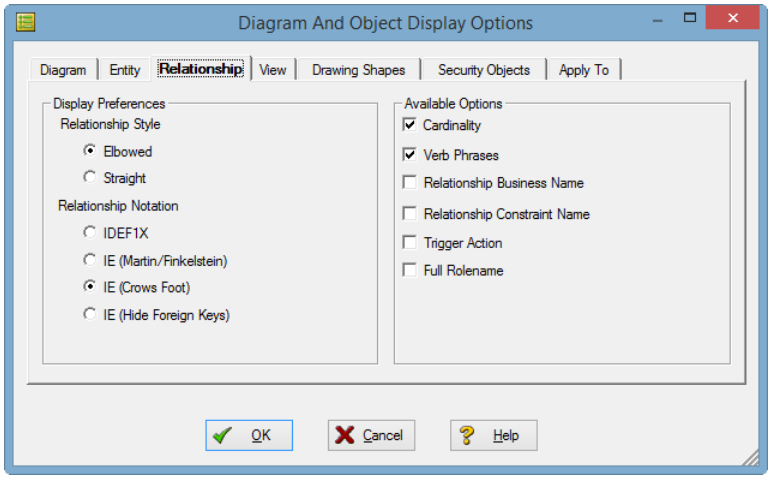

Figure 11 *Users of ER/Studio can select between different notation techniques and can select the aspects of the data model that must be included.*

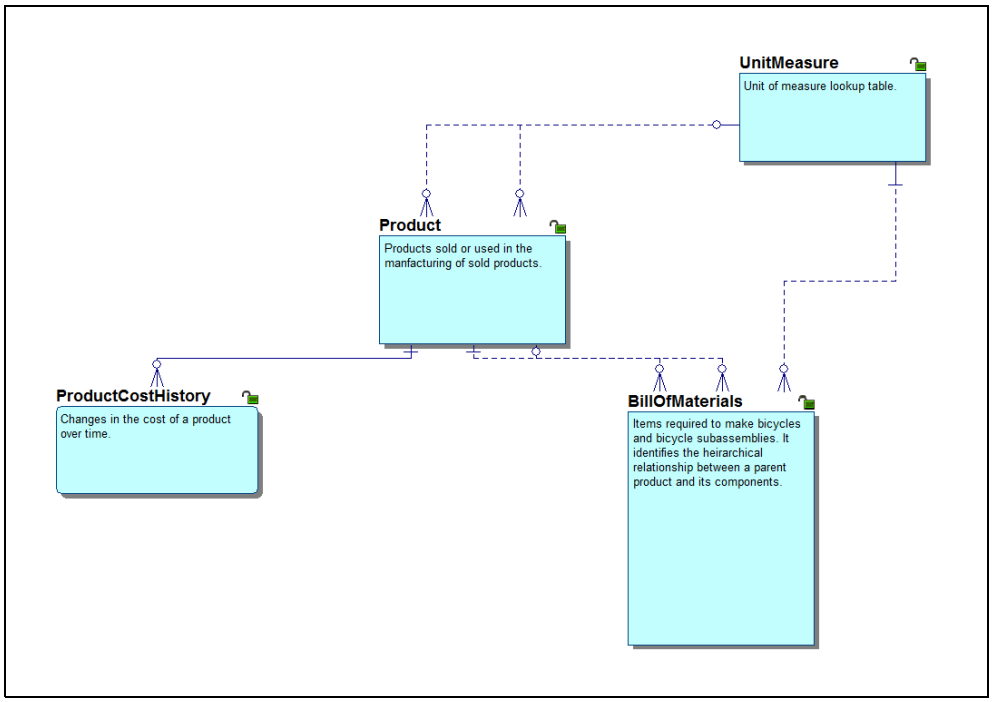

Figure 12 *A data model showing only high-level aspects such as definitions and relationships.*

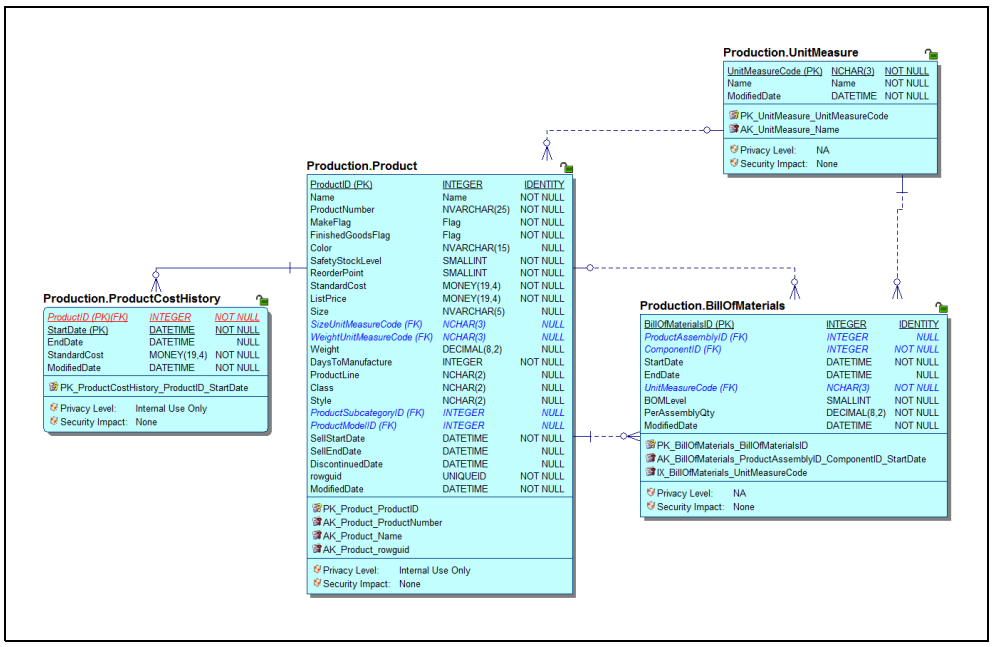

Figure 13 *A data model showing all the technical details.*

About the Author Rick F. van der Lans

Rick van der Lans is a highly-respected independent analyst, consultant, author, and internationally acclaimed lecturer specializing in data warehousing, business intelligence, big data, database technology, and data virtualization. He works for R20/Consultancy (www.r20.nl), which he founded in 1987. In 2018 he was selected the sixth most influential BI analyst worldwide by onalytica.com[14].

He has presented countless seminars, webinars, and keynotes at industry-leading conferences. For many years, he has served as the chairman of the annual *European Enterprise Data and Business Intelligence Conference* in London and the annual *Data Warehousing and Business Intelligence Summit* in The Netherlands.

Rick helps clients worldwide to design their data warehouse, big data, and business intelligence architectures and solutions and assists them with selecting the right products. He has been influential in introducing the new logical data warehouse architecture worldwide which helps organizations to develop more agile business intelligence systems. He introduced the business intelligence architecture called the *Data Delivery Platform* in 2009 in a number of articles[15] all published at B-eye-Network.com. The Data Delivery Platform is an architecture very similar to the logical data warehouse.

Over the years, Rick has written hundreds of articles and blogs for newspapers and websites and has authored many educational and popular white papers for a long list of vendors. He was the author of the first available book on SQL[16], entitled *Introduction to SQL*, which has been translated into several languages with more than 100,000 copies sold. More recently, he published his book[17] *Data Virtualization for Business Intelligence Systems*.

For more information please visit www.r20.nl, or send an email to rick@r20.nl. You can also get in touch with him via LinkedIn and Twitter (@Rick_vanderlans).

About IDERA

IDERA understands that IT doesn't run on the network – it runs on the data and databases that power your business. That's why IDERA designs their products with the database as the nucleus of your IT universe.

Their database lifecycle management solutions allow database and IT professionals to design, monitor and manage data systems with complete confidence, whether in the cloud or on-premises. They offer a diverse portfolio of free tools and educational resources to help you do more with less while giving you the knowledge to deliver even more than you did yesterday.

Whatever your need, IDERA has a solution.

[14] Onalytica.com, *Business Intelligence – Top Influencers, Brands and Publications*, June 2018; see http://www.onalytica.com/blog/posts/business-intelligence-top-influencers-brands-publications/
[15] See http://www.b-eye-network.com/channels/5087/view/12495
[16] R.F. van der Lans, *Introduction to SQL; Mastering the Relational Database Language*, fourth edition, Addison-Wesley, 2007.
[17] R.F. van der Lans, *Data Virtualization for Business Intelligence Systems*, Morgan Kaufmann Publishers, 2012.