Why Build A Logical Data Model

By Barbara A. Carkenord – The Knowledge Exchange Company
770-612-0020, www.tkec.com
Why Build a Logical Data Model?

Even though the concept of data modeling has been around for a long time, it is still not widely practiced. The phrase data modeling is interpreted differently by different people and methodologies. The purpose of this white paper is to discuss the importance of building a logical data model. It answers the questions: What is a logical data model?, Who uses the logical data model?, How is a logical data model different than a physical data model?, Why build one?, and What happens if you don't build a logical data model?

Data models

There are two types of data models: a logical and a physical. Everyone agrees that a good quality data structure is critical to a long lasting, easy to maintain system. This is referred to as the physical data model. Although there still are a few database administrators who say “Oh, it is just easier to put everything in one big table,” most people are aware of the data normalization rules and their benefits. Everyone involved in designing databases should familiarize themselves with the reasons for implementing a normalized database. Once educated on the techniques and justifications, it is difficult not to agree with such a rational approach. The purpose of this paper is not to discuss database design or implementation. The Logical Data Model refers to a higher level of data than the business data. Once we know the business data requirements, we can normalize and implement the data. However, if we do not have a clear understanding of the business requirements all of the design and implementation work in the world will not result in a good quality application.

Data is the most important part of an application system. A good, strong, accurate data structure allows application developers to design any processing, user interface, reporting, or statistical analysis ever needed. The only things more important in developing application systems than a quality data structure, are the business requirements. The most elegant, high tech system in the world will be shelved if it does not meet a business requirement. Therefore, logical data modeling combines the two most important components of application development: business requirements and quality data structure.
What is a logical data model?

A logical data model is a graphical representation of the information requirements of a business area, it is not a database.

The word logical is very critical because it modifies the phrase data modeling to a more specific activity. A logical data model is independent of a physical, data storage device. This is the key concept of the logical data model. The reason that a logical data model must be independent of technology is simply because technology is changing so rapidly.

Only 10 years ago, relational databases were still considered new and risky. It is possible that 10 years in the future we may look back on relational databases as dinosaurs. The improvements in hardware speed and sophistication may never stop or even level off to a pace we can keep up. It is very dangerous for application developers to continue to gather requirements with a specific technology in mind. Our legacy problem continues to grow and multiply in complexity. We need to build systems that are as independent of technology as possible. Build a system that is independent of technology; isn’t this an oxymoron? Yes and No.

There are components of a system that are intimately linked to the technology: the programs, the database management systems, and the screen components. There are also components of the system that can be technology independent: the logical data model, the logical process model, the business rules. These components are intimately linked to the business, not the technology.

Most business areas do not change as rapidly as technology. Think of industries that have been in business for 100 or 200 years. An insurance company provides coverage, receives payments, and pays claims; and has done so for their entire history. They performed this business without computers and now perform it on mainframes, minicomputers, PCs, networks, and the Internet. What the business does, has not changed, but how the work is accomplished, has changed significantly. This difference between WHAT the business requirements are and HOW they are accomplished describes the difference between a logical datamodel and physical database.
Using a Logical Data model speeds maintenance and eases the Transition to new technologies.

When a change request or new technology comes along, analysts with access to a logical data model do not need to go back to the business people and ask: “What do you do?” and “What data elements do you need to support your business?” We, as Information Systems professionals, should review our logical models - our understanding of the business requirements - and offer suggestions for implementing the changes, or suggest how the new technology could change the way the business requirements are accomplished. Building and maintaining a logical data model for the business area allows the opportunity to provide this service.

The logical data model is a picture of all of the pieces of information necessary to run the business. The logical data model is built using an Entity Relationship diagram (ERD) - a standard modeling technique used by data modelers around the world. Entity relationship diagramming is a structured technique used by business people and technical people as a communication tool. It includes the complete business requirements as opposed to the current physical data storage. It is the only place with a complete picture of the data required by the business.

The components of a logical data model include Entities, Relationships, and Attributes. Each Entity represents a set of persons, things, or concepts about which the business needs information. Each Relationship represents an association between two entities. Each Attribute is a characteristic or piece of information that further describes an entity. A name and a textual definition describe each of these components. These name and definitions provide ongoing documentation of the business rules and information requirements of the business area.

A Logical Data Model asks, “What do you and your business do”

Logical Data Modeling Terminology

The terminology used in logical data modeling is very important. It is technology independent and as such operates as a communication medium between business people and application developers. Zachman used the phrase Conceptual Model in his framework to refer to a high-level data model, a starting point for the complete logical data model. It represents a picture of the information requirements of the entire organization without any detail. It is not a normalized model. In contrast, an Enterprise Model is a logical data model of the entire organization including all data elements. Most organizations create the enterprise model by combining the logical models of several projects. This model is stored in a repository and used by all new application development project teams. A Project logical data model includes only the business requirements within the scope of the project. It is a subset of the enterprise model (if there is one). An End-User Model is a simplified view of the project data model used to present project progress to business area experts and their management. All of these models
are referred to as Logical or Essential Models. The words logical and essential describe the representation of information from an end-user perspective, without regard to its physical or technological aspects. They describe what the business requirements are, not how they are implemented, stored or processed.

Entity Relationship diagramming is a standard documentation approach for creating and detailing a logical data model. It was developed in the 1970s and can be notated using Information Engineering or IDEF1x symbols.

**Who uses the logical data model?**

The Business Area Experts own the logical data model. They describe their data requirements to the data modeler and review the models created. They use the models for impact analysis of changes to business requirements.

The Data Modeler conducts facilitated sessions with business area experts to gather the data requirements and build the logical data model. The data modeler also works with the process analyst to link data with processes. The data modeler is responsible for getting approval of the logical data model from the business area experts and then works with the DBA to transition the logical model to the physical model.

The Database Administrator (DBA) builds the physical data model from the logical data model. To create a good quality database design, the database administrator reviews the logical model to select technology appropriate keys, create indexes, detail data types, and build referential integrity to protect the data values. The database administrator may de-normalize the database for efficiency. DBA's also are responsible for writing stored procedures, triggers, maintaining referential integrity, and monitoring database performance.

**What is the difference between a logical data model and a physical database design?**

Reviewing the information in the table below, one could gloss over the importance of the differences between the models. Many people carelessly interchange the word entity with the word table. However, the distinction is significant. An entity represents a set of persons, things, or concepts that are important to the business. A table is a data structure that contains data values and is physically stored on a disk using a specific DBMS. The difference is the link to technology. An entity has no link to technology; a table is a technology. Also consider the difference between a unique identifier in the logical model and a primary key in the physical model. A unique identifier represents the data element(s) used by the business to discriminate between one occurrence of an entity and another.
"How do you tell the difference between one CUSTOMER and another?" A business area expert may answer this question: Customer Name. This is an important business requirement and must be captured as such. However, most database designers would not select Customer Name as a primary key. In the physical model, a primary key, such as a computer generated Customer Number, is selected because it uniquely identifies each row in the table and can serve as a foreign key in related tables.

**Why build a logical data model?**

The most important reason to build a logical data model is to confirm the users and analysts understanding of the business requirements to assure that the system developed satisfies the business need. Logical data modeling provides the analyst with a tool and technique to conduct analysis. Most business area experts can articulate problems, and oftentimes, solutions. Unfortunately their problems and solutions are often based on current system constraints, not true business needs. Assuming perfect technology forces users and analysts to look beyond the current system limitations. Asking the business people to detail every piece of data (attributes), requires them to understand and articulate every aspect of their business. This process allows the business to drive the system, not the other way around. It also stimulates discussion and thoughts. By identifying and detailing data in a model, further requirements and problem areas arise simply because of discussing and reviewing the model.

A logical data model is a foundation for designing a database that supports the business requirements. Database designers start their design with a complete picture of the business requirements and can then determine the best implementation approach. This allows designers to use their expertise in data access paths, data distribution and placement, and access efficiency to create a database that will satisfy business requirements for years to come.

<table>
<thead>
<tr>
<th>THE LOGICAL MODEL</th>
<th>THE PHYSICAL DATABASE DESIGN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Includes all entities, relationships, and attributes (and their information types) whether supported by a technology or not.</td>
<td>Includes tables, columns, keys, datatypes, validation rules, DB triggers, stored procedures, domains, and access constraints (security).</td>
</tr>
<tr>
<td>Uses business names.</td>
<td>Names may be limited by the DBMS.</td>
</tr>
<tr>
<td>Captures and records information necessary for the business.</td>
<td>Includes technology-specific data elements such as flags, switches, and timestamps.</td>
</tr>
<tr>
<td>Includes unique identifiers.</td>
<td>Includes primary keys, foreign keys, and indices for fast data access.</td>
</tr>
<tr>
<td>Is normalized to at least 3rd normal form.</td>
<td>May be de-normalized to meet performance requirements.</td>
</tr>
<tr>
<td>Does not include any redundant data.</td>
<td>May include redundant data elements.</td>
</tr>
<tr>
<td>Does not include any derived data.</td>
<td>May include results of complex or difficult to recreate calculations.</td>
</tr>
<tr>
<td>Business experts drive the model.</td>
<td>DBAs drive the model.</td>
</tr>
</tbody>
</table>
A logical data model also facilitates data re-use and sharing. Data is stable over time; therefore, the model remains stable over time. As additional project teams scope out their areas, they can re-use model components that are shared by the business. This leads to physical data sharing and less storage of redundant data. It also helps the organization recognize that information is an organization-wide resource, not the property of one department or another. Data sharing makes the organization more cohesive and increases the quality of service to outside customers and suppliers.

Building and maintaining a logical data model decreases system development and maintenance time and cost. Identifying all business requirements at the beginning of a project makes the design, coding, testing, and implementation phases go much smoother and faster. A model is easier and cheaper to modify during the development life cycle. Mistakes, missed data, and misinterpretations are less costly when corrected in a model than in an implemented system. It also decreases user requests for changes. When changes are necessary, the logical model can be used for impact analysis. A logical data model, documented in an Entity Relationship diagram, is a picture of the business area. Every entity, attribute, and relationship created and defined in the model is a piece of the resulting system documentation. The objects in the model contain textual definitions that describe their characteristics in business language. Because of data modeling, system documentation is an automatic output.

A logical data model confirms a logical process model and provides documentation of the information requirements of the business area for ongoing impact analysis. Each business process is tied to the logical data model to assure that all data and process model components have been discovered.

**What happens if you don’t build a logical data model?**

When users are not asked to focus on data as a critical part of a new system design, they talk about processes and activities. They may forget to tell designers about all data requirements. Traditionally designers create tables and files based on screen and report layouts, searching out data elements as they go. These data elements are not well organized or structured properly. Designing a model based on physical workflow could result in a model that does not fully represent the business requirements. This can occur when the technology of the workflow is forced into the physical model. The result is often a database that is missing critical data and must be changed immediately after implementation.

The database design task is a much longer activity and the database may be poorly structured. Since not all business data requirements have been thoroughly flushed out, database designs are unstable throughout the development process. During coding, test-
ing and even implementation, developers are finding additional data elements, requiring the DBAs to be re-active instead of pro-active. Resulting databases may be poorly structured, looking like a patchwork quilt, instead of being well planned and easy to maintain. Errors in the database design will cause the entire system to be unusable.

System documentation doesn’t get created, or is textual vs. graphical and difficult to maintain. User requirements are lengthy, textual documents that are time consuming to review. Without a logical model to refer to, data definition language must be referenced when planning system enhancements. DBMS systems often limit the length of column names making them difficult to decipher.

An ERD is a structured, graphical document that describes the business supported by the system.

Logical data modeling forces analysts to think about the current business requirements, independent of technology, thereby highlighting opportunities for business process improvement rather than simply automating an existing procedure or recreating a legacy system on a new platform. Users understanding of their current systems can prevent them from identifying the true business requirements. The main reason for missed target dates is a poor understanding of the business requirements. Building a complete, essential, logical data model (and linking it to an essential process model) forces the analysts and the business users to completely describe all information requirements of the business area. Without this rigorous analysis, data elements will be missed or defined improperly.

**Conclusion**

Completely understanding the data requirements of a business area before developing a solution is a proven approach to Application Development success. Contact TKEC for training, consulting, or assistance in selecting a data modeling tool.