Webology (ISSN: 1735-188X)
Volume 19, Number 3, 2022

Data Modeling: A Perspective In Changing Database Scenario

Dr Sunita Dwivedi[1] , Leeladhar Chourasiya[2]

[1]Associate Professor, MCRPV Bhopal.

[2] Research Scholars, MCRPV Bhopal.

Abstract

Data modeling is a decisive skill for every data scientist, whether you are doing research design or architecting a new data base for any organization. The skill to think clearly and systematically about the key data points to be stored, retrieved, and also how they should be grouped and related, is what the data modeling element of data science is all about. A data model describes information in a organized way that allows it to be stored and retrieved efficiently in a Relational Database Management System (RDBMS), such as SQL Server, MySQL, or Oracle. The model can be thought of as a way of translating the logic of precisely relating things in the real world and the relationships between them into rules that can be followed and enforced by computer code. Data Models and Data Modeling Methodologies have been around since the beginning of time. At least since the beginning of computer technology. The data needs a structure that provides the computer with a way to understand the data and process its bits and bytes. Sure, we're also dealing with unstructured and semi-structured data today, but for me it means we've evolved into a more sophisticated paradigm than previous computing. Therefore, the data model remains and forms the basis for building sophisticated business applications.

Introduction

When we come across terms such as modeling and design in the context of data, we think about the logical design and physical implementation of a database. Data modeling can mean documenting the design of software and business systems. Diagrams, symbols, and text references are used to represent how data flows within an organization or software application. Data modeling is also involved in the decision-making and conceptual business processes of the entire organization. Data modeling is commonly used to design data structures at various levels of abstraction, from conceptual to physics. As a result, data modeling also leads to effective design. The model easily determines the logical structure and style in which the data is organized, accessed, or manipulated. Organizations are now

interested in collecting big data about their business, storing this data in repositories, and applying analytics to make decisions. The underlying DBMS allows you to model your data in a variety of ways. This modeling depends entirely on the structure of the application data and the requirements of the application. Most of the DBMS are built around the particular data model even it is possible to extract more from it. A good data model will allow repository to grow easily, as well as allowing for good performance. User requirements are main factor for development of logical data model, and then it is translated into the physical data model. Ingredient of the data modeling exercise is often the identification of data sources. Sometimes this step is delayed until the ETL step. However, it is better to find out where the data exists, or, better yet, whether they even exist anywhere in the enterprise at all. If this activity is somehow delayed until the ETL phase, rectifying it will become a much tougher and more complex process. In this paper we try to highlight the data modeling techniques from its inception to recent Big Data families.

History of Data Models

In the 'Computing Dark Ages', we used flat record layouts, or arrays; all data saved to tape or large disk drives for subsequent retrieval. However, in 1958, J. W. Young and H. K. Kent described modeling information systems as "a precise and abstract way of specifying the informational and time characteristics of a data processing problem". Soon after in 1959, CODASYL or the 'Conference/Committee on Data Systems Languages', a consortium, was formed by the Charles Babbage Institute at the University of Minnesota which led to standard programming languages like COBOL and the 'Integrated Data Store' (IDS); an early database technology designed in the 1960's at GE/Honeywell by Charles Bachman. IDS proved difficult to use, so it evolved to become the 'Integrated Database Management System' (IDMS) developed at B. F. Goodrich (a US aerospace company at the time, and yes the tire company we know today), marketed by Cullinane Database Systems (now owned by Computer Associates). These two data modeling methodologies called the 'Hierarchal Data Model' and the 'Network Data Model' respectively, were both very common across mainframe computing for the next 50 years. You may still find them in use today. Wow!

In the late 1960's, while working at IBM, E. F. Codd in collaboration with C. J. Date (author of 'An Introduction to Database Systems'), mapped Codd's innovative data modeling theories resulting in the 'Relational Model of Data for Large Shared Data Banks' publication in 1970. Codd's campaign to ensure vendors implemented the methodology properly published his famous 'Twelve Rules of the Relational Model' in 1985. Actually, thirteen rules numbered zero to twelve; Codd was clearly a computer geek of his day.

The Relational Model also introduced the concept of 'Normalization' with the definition of the 'Five Normal Forms'. Many of us talk about '3NF' or the '3rd Normal Form', but do

you know how to define it? Read up on these two links and find out if you really know what you think you know. There will be a quiz at the end! Not …

The next significant data modeling methodology arrived in 1996, proposed by Ralph Kimball (retired), in his groundbreaking book co-authored by Margy Ross, 'The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling'. Kimball's widely adopted 'Star Schema'

Data model applied concepts introduced in the data warehouse paradigm first proposed in the 1970's by W. H. (Bill) Inmon (named in 2007 by Computerworld as one of the ten Most influential people of the first 40 years in computing). Inmon's 'Building the Data Warehouse', published in 1991 has become the defacto standard for all data warehouse computing. While there has been some history of disagreement between Inmon and Kimball over the proper approach to data warehouse implementation, Margy Ross, of the Kimball Group in her article 'Differences of Opinion' presents a fair and balanced explanation for your worthy consideration.

Recently a new data modeling methodology has emerged as a strong contender. The Data Vault! Its author and inventor, Dan Linsdedt, first conceived the Data Vault in 1990 and released a publication to the public domain in 2001. The Data Vault model resolves many competing Inmon & Kimball arguments, incorporating historical lineage of data, and offering a highly adaptable, auditable, and expandable paradigm. A critical improvement (IMHO); I invite you to read my blog on 'What is "The Data Vault" and why do we need it?'. Linstedt's Data Vault proved invaluable on several significant DOD, NSA, and Corporate projects. In 2013, Linsdedt released Data Vault 2.0 addressing Big Data, NoSQL, unstructured, semi-structured data integration coupled with SDLC best practices on how to use it. Perfect timing, I'd say. So here we are …

2.1 Data Modeling Process

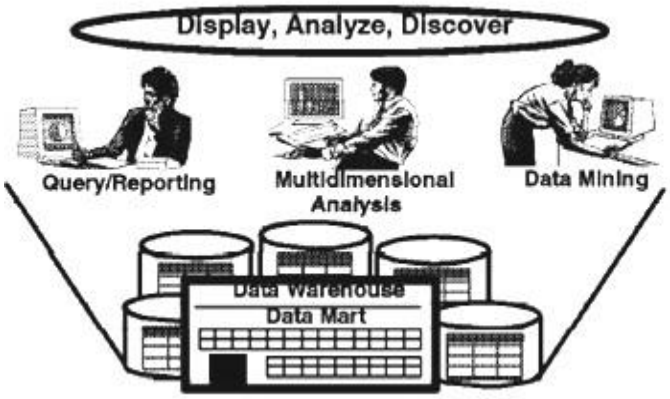
Data modeling process starts with analyzing the situation. Here the analysts are able to gather requirements, when designing a proper data model it's important to communicate with the stakeholders about the requirements. Data modeling is the act of exploring data oriented structures, which can be used for multiple purposes. Mainly data modeling is a communication tool among users, which considers as the blue print of the database system. (Merson, Paulo F.).

A data model consists of three different phases. (West)Those are:
Structural part – Consisting a set of rules
Manipulating part – Types of operations allowed, such as updating, retrieving, and changing the database Integrity part – which validates the accuracy of data.

2.2 Data Analysis

The techniques of data analysis can impact the type of data model selected and its content. For example, if the intent is simply to provide query and reporting capability, a data model

that structures the data in more of a normalized fashion would probably provide the fastest and easiest access to the data. Query and reporting capability primarily consists of selecting associated data elements, perhaps summarizing them and grouping them by some category, and presenting the results. Executing this type of capability typically might lead to the use of more direct table scans. For this type of capability, perhaps an ER model with a normalized and/or demoralized data structure would be most appropriate.



Reason to Data Model

We'll look at how data models are easier to change than databases, why data models are easier to review than database designs, and consider how data modeling principles will help you succeed in a wider range of software projects.

– Data Models are Easier to Change than Databases
– Data Models Are Easier to Review than Database Designs
– Data Models Will Help You Succeed On More Projects

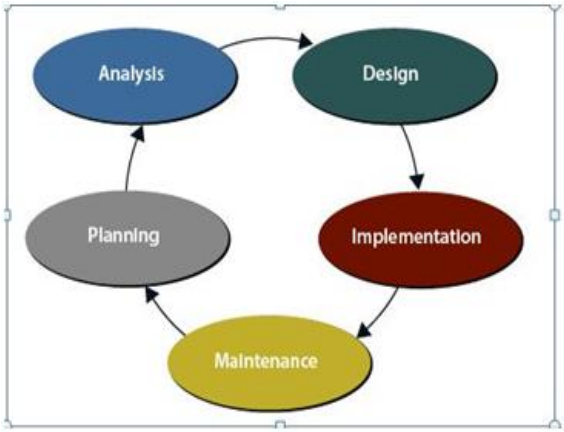
Stages of Data Modeling

There is a horde of related terminology including application data models, conceptual modeling, enterprise modeling, logical models, physical models, entity-relationship models, object models, multi-dimensional models, knowledge graphs, statistical models, canonical data models, business requirements models, enterprise data models, integration models, business information models, taxonomies, non-relational models, semantic modeling, and many others.

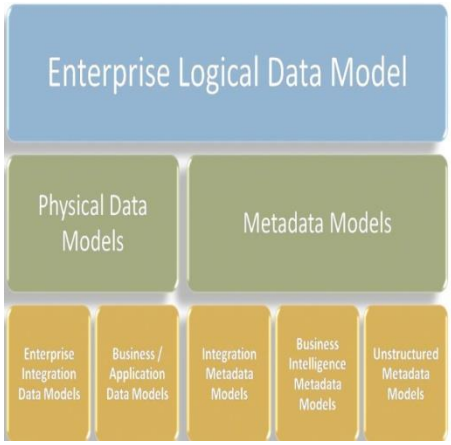
A model is not just a manner of structuring data but also defines a set of operations that can be performed on the data. For example the relational model defines operations such as select and join. These operations may not be explicit in a particular query language they provide the foundation on which a query language is built.

One can implement various physical data models from given logical model. While doing physical implementation most of the DBMS allow some degree of control on this, which then improve the performance.

Conceptual – This is the first step in the modeling process, which imposes a theoretical order on data as it exists in relationship to the entities being described, often real-world artifacts or concepts. These models are also called domain models and are typically used to explore domain concepts with project stakeholders. On traditional teams conceptual data models are often created as the precursor to Logical Data Models or as alternatives to Logical Data Models.



Logical – Taking the semantic structure built at the conceptual stage, the logical modeling process attempts to impose order by establishing discrete entities, key values, and relationships in a logical structure that is brought into at least 4th normal form (4NF). This could be done for the scope of a single project or for your entire enterprise.



Physical – Actually not physical at all, but it would be confusing to use “logical” twice, this step breaks the data down into the actual tables, clusters, and indexes required for the data

store. Physical data models are used to design the internal schema of a database, depicting the data tables, the data columns of those tables, and the relationships between the tables.

Although logical data model and physical data model sound very similar, and they in fact are, the level of detail that they model can be significantly different. This is because the goals for each diagram are different – you can use a logical data model to explore domain concepts with your stakeholders and the physical data model to define your database design. Figure 1 presents a simple logical data model and Figure 2 a simple physical data model, both modeling the concept of customers and addresses as well as the relationship between them. Both diagrams apply the Barker notation, summarized below. Notice how the physical data model shows greater detail, including an associative table required to implement the association as well as the keys needed to maintain the relationships. More on these concepts later. Physical data model should also reflect your organization's database naming standards; in this case an abbreviation of the entity name is appended to each column name and an abbreviation for "Number" was consistently introduced. Physical data model should also indicate the data types for the columns, such as integer and char(5). Although Figure 2 does not show them, lookup tables (also called reference tables or description tables) for how the address is used as well as for states and countries are implied by the attributes.

Figure 1. A simple logical data model.

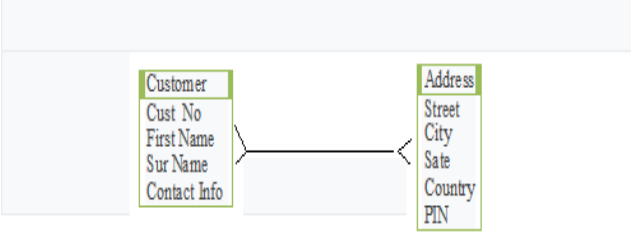
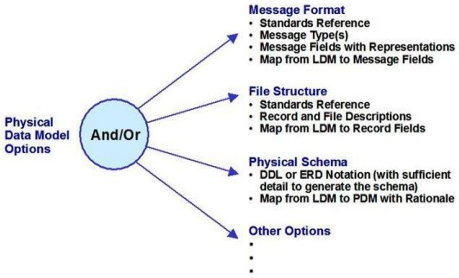
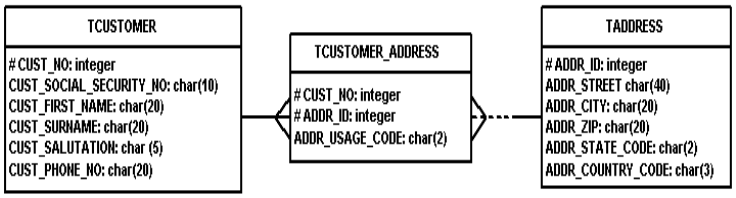


Figure 2. A simple physical data model.





The following is the comparison of these three models based on different features:

Feature	Conceptual	Logical	Physical
Entity Names	✓	✓	
Entity Relationships	✓	✓	
Attributes		✓	
Primary Keys		✓	✓
Foreign Keys		✓	✓
Table Names			✓
Column Names			✓
Column Data Types			✓

The complexity increases from conceptual model to logical model to physical model hence we always first start with the conceptual data model to understand at high level what are the different entities in our data and how they relate to one another then move on to the logical data model to understand the details of our data without worrying about how they will actually implemented and finally the physical data model to know exactly how to implement our data model in the database of choice.

There are many possible visual representations of data models, but the primary one used in database design today is the classic entity-relationship model. This is simply a flowchart of boxes, describing entities with their attendant data points inside, and lines between the boxes, describing the relationships between entities.

Modelers might find themselves using other specialized modeling methodologies on specific projects, however, and data scientists will be expected to learn several of them, including:

Bachman Diagrams

Object-Role modeling

Zachman Frameworks

Data Modeling for Non RDBMS

Massive datasets have pulled back the dominance of RDBMS's, whether the data being stored can easily be modeled relationally or not. The RDBMS store archetype depends on the database system itself to maintain coherence and internal consistency of the data being held

in it, and while the relational model, when properly applied, can achieve this, the process comes with overhead. When huge data points are being stored, the price of this internal consistency can bring performance grinding to a halt.

NoSQL databases such as MongoDB, HBase and Cassandra and have been one of the most promising industry answers to this problem. These use sometimes fundamentally de-normalized data stores with the sole objective to improve performance. They rely on queries and calling code to handle the sort of integrity, consistency and concurrency that come baked-in to the RDBMS approach, offering blinding speed and scalability over ease-of-use.

To do this, they adopt simplistic data stores such as:

- Key-value stores
- Document stores
- Graphs

Modeling these types of stores is a significant departure from the RDBMS method. Data scientists may start from the result side of the process, asking themselves, "What question am I trying to answer?" instead of "What data do I need to store?" They'll completely disregard duplication of data and have to plan to handle concurrency conflicts and other integrity issues on the output end rather than in the design itself. They might choose to aggregate data rather than breaking it down discretely, shoving complete sales transactions into a flat document store, for instance.

NoSQL data modeling puts your education to the test as you put to use advanced techniques such as:

- Atomic updates
- Dimensionality reduction
- Inverted search patterns
- Tree aggregation

Understanding these techniques and the capabilities offered by NoSQL, allow data scientists to make the best choices for what type of data store to use and how to model it. In almost every case, data scientists in the real world will end up using a combination of RDBMSs and NoSQL or other exotic data sources as a daily part of their work. Understanding how to apply the models that allow those systems to record a picture of the world is the only thing that makes the job even distantly possible.

Database Model Types

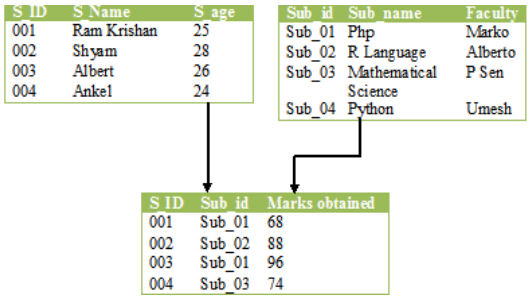
The main work of data modeling is to categorize data or any kind of information that is required by the system so it can store it, maintain it or let others access it when needed. There are several types of data models available in database. Most common models are:

- Relational
- Hierarchical
- Entity-relationship
- Network

- Object-oriented database
- Document
- Entity-attribute-value
- Star schema
- The object-relational, which combines the two that make up its name

Relational model

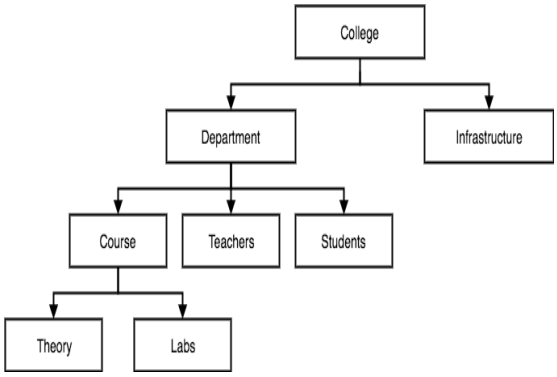
The data is organized in two-dimensional tables and the relationship is maintained by storing a common field in relational model. In 1970 E.F Codd has introduced this model and is most widely used model today also. Table is basic structure of the data in this model. The information related to a particular type is stored in rows of that table. Thus, tables are also known as relations in relational model.



Hierarchical Model

In this model, a child node will only have a single parent node and data is organizes into a tree-like-structure, with a single root, to which all the other data is linked. The hierarchy starts from the Root data, and expands like a tree, adding child nodes to the parent nodes.

In hierarchical model, data is organized into tree-like structure with one one-to-many relationship between two different types of data, for example, one department can have many courses, many professors and of-course many students.



Entity-relationship

In entity-relationship model, relationships are created by dividing object of interest into entity and its characteristics into attributes. Different entities are related using relationships. ER Models are defined to represent the relationships into pictorial form to make it easier for different stakeholders to understand. This model is good to design a database, which can then be turned into tables in relational model.

Network model

The network model builds on the hierarchical model by allowing many-to-many relationships between linked records, implying multiple parent records. Based on mathematical set theory, the model is constructed with sets of related records. Each set consists of one owner or parent record and one or more member or child records. A record can be a member or child in multiple sets, allowing this model to convey complex relationships.

It was most popular in the 70s after it was formally defined by the Conference on Data Systems Languages (CODASYL).

Object-oriented database model

This model defines a database as a collection of objects, or reusable software elements, with associated features and methods. There are several kinds of object-oriented databases:

A multimedia database incorporates media, such as images, that could not be stored in a relational database.

A hypertext database allows any object to link to any other object. It's useful for organizing lots of disparate data, but it's not ideal for numerical analysis.

The object-oriented database model is the best known post-relational database model, since it incorporates tables, but isn't limited to tables. Such models are also known as hybrid database models.

Object-relational model

This hybrid database model combines the simplicity of the relational model with some of the advanced functionality of the object-oriented database model. In essence, it allows designers to incorporate objects into the familiar table structure.

Languages and call interfaces include SQL3, vendor languages, ODBC, JDBC, and proprietary call interfaces that are extensions of the languages and interfaces used by the relational model

Other database models

A variety of other database models has been or are still used today.

Inverted file model

A database built with the inverted file structure is designed to facilitate fast full text searches. In this model, data content is indexed as a series of keys in a lookup table, with the values

pointing to the location of the associated files. This structure can provide nearly instantaneous reporting in big data and analytics, for instance.
This model has been used by the ADABAS database management system of Software AG since 1970, and it is still supported today.

Flat model
The flat model is the earliest, simplest data model. It simply lists all the data in a single table, consisting of columns and rows. In order to access or manipulate the data, the computer has to read the entire flat file into memory, which makes this model inefficient for all but the smallest data sets.

Multidimensional model
This is a variation of the relational model designed to facilitate improved analytical processing. While the relational model is optimized for online transaction processing (OLTP), this model is designed for online analytical processing (OLAP).
Each cell in a dimensional database contains data about the dimensions tracked by the database. Visually, it's like a collection of cubes, rather than two-dimensional tables.

Semi structured model
In this model, the structural data usually contained in the database schema is embedded with the data itself. Here the distinction between data and schema is vague at best. This model is useful for describing systems, such as certain Web-based data sources, which we treat as databases but cannot constrain with a schema. It's also useful for describing interactions between databases that don't adhere to the same schema.

Context model
This model can incorporate elements from other database models as needed. It cobbles together elements from object-oriented, semi structured, and network models.

Associative model
This model divides all the data points based on whether they describe an entity or an association. In this model, an entity is anything that exists independently, whereas an association is something that only exists in relation to something else.
The associative model structures the data into two sets:

- A set of items, each with a unique identifier, a name, and a type
- A set of links, each with a unique identifier and the unique identifiers of a source, verb, and target. The stored fact has to do with the source, and each of the three identifiers may refer either to a link or an item.

Other, less common database models include:

- Semantic model, which includes information about how the stored data relates to the real world
- XML database, which allows data to be specified and even stored in XML format
- Named graph
- Triple store

NoSQL database models

In addition to the object database model, other non-SQL models have emerged in contrast to the relational model:

The graph database model, which is even more flexible than a network model, allowing any node to connect with any other. The multivalve model, which breaks from the relational model by allowing attributes to contain a list of data rather than a single data point. The document model, which is designed for storing and managing documents or semi-structured data, rather than atomic data.

Conclusion

A Data Model is a common and essential ingredient of Business Applications, Data Integration, Master Data Management, Data Warehousing, Big Data, Data Lakes, and Machine Learning. The Data Model is the backbone of almost all of our high value, mission critical, business solutions from e-Commerce and Point-of-Sale, through Financial, Product, and Customer Management, to Business Intelligence and IoT. The paper presented above gives an understanding of Data modeling techniques; along with it the paper gives a review of the research and developments in the field of Data and Modeling techniques. The paper also provides the suggestion regarding the future researches in the field of Data and Modeling.

Reference

1. Xiaoyue Han, Lianhua Tian, Minjoo Yoon, Minsoo Lee, "A Big Data Model supporting Information Recommendation in Social
Networks", 2012 Second International Conference on Cloud and Green Computing. IEEE.
2. Imran Khan, S. K. Naqvi, Mansaf Alam, S. N. A Rizvi, "Data Model for Big Data in Cloud Environment", 2015 2nd
International Conference on Computing for Sustainable Global Development (INDIA Com). IEEE.
3. Abdullah, M. F., & Ahmad, K. (2013, November). The mapping process of unstructured data to structured data. In Research
and Innovation in Information Systems (ICRIIS), 2013 International Conference on (pp. 151-155). IEEE.

4. Li Kang, Li Yi, LIU Dong, "Research on Construction Methods of Big Data Semantic Model", Proceedings of the World
Congress on Engineering 2014 Vol- I, WCE 2014, July 2 - 4, 2014, London, U.K. IEEE.
5. Gruber TR, "A translation approach to portable ontology specifications[J]" Knowledge acquisition, 1993, 5(2): 199-220.
6. Hemant Kumud, "Handling Unstructured Data for Semantic Web – A Natural Language Processing Approach", Scholars
7.Journal of Engineering and Technology (SJET) ISSN 2321-435X (Online) Sch. J. Eng. Tech., 2014; 2(2A):193-196