Information Technology Standard
Commonwealth of Pennsylvania
Governor's Office of Administration/Office for Information Technology

STD Number:	**STD-INF003B**	
STD Title:	**Data Modeling Basics**	
Issued by:	**Deputy Secretary for Information Technology**	
Date Issued: August 2, 2005	**Date Revised: November 18, 2010**	
Domain:	**Information**	
Discipline:	**Data Administration**	
Technology Area:	**Data Modeling**	
Referenced by:	**ITP-INF003**	

Revision History Date:	**Description:**
11/18/2010	**ITP Refresh**

Data Modeling Basics

Bureau of Enterprise Architecture
Information Domain Team

Why is Data Modeling Important?

Data modeling is a very vital part in the development process. One can compare this to creating a blueprint to build a house before the actual building takes place. As much as the blueprint takes time to prepare, and goes through multiple iterations of validation to ensure that the foundation, structure and aesthetics of the building plan conform to intended objectives and quality standards, so also data modeling is an intensive process which consumes a major part of the development time.

The model is built in a phased manner and goes through several iterations of validation to ensure that the structure and content of the model addresses the business objectives of the enterprise or application, meets quality standards, is modular, provides a solid foundation for future extensions and data reuse for other enterprise applications etc. That being the case, less time spent in this effort will only produce a weak unstructured model which will be very expensive to maintain in the future, may produce inconsistent and incorrect results and will be unfit for reporting or future extensions.

Major events in data modeling include:

- Identify entities, data requirements and processes
- Define attributes of the data such as data types, sizes, defaults
- Apply validation and business rules to ensure data integrity
- Define data management and security processes
- Specifying data archival and storage

How are Data Models Used in Practice?

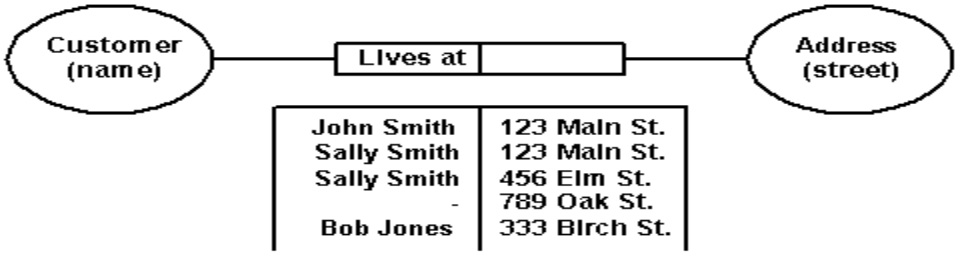
You are likely to see three basic types of data model:

- **Conceptual data models**. These models, sometimes called domain models, are typically used to identify and document business (domain) concepts with project stakeholders. Conceptual data models are often created as the precursor to Logical Data Models (LDMs) or as alternatives to LDMs.

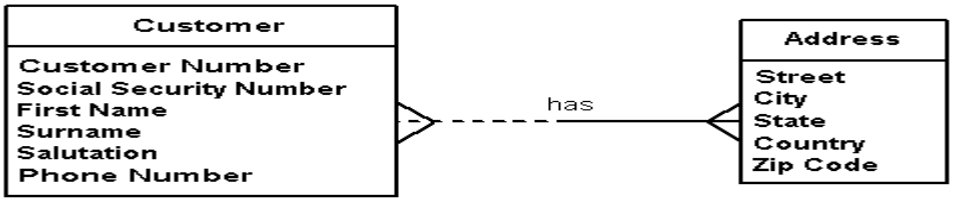
- **Logical data models (LDMs)**. Logical Data Models are used to further explore the domain concepts, and their relationships and relationship cardinalities. This could be done for the scope of a single project or for your entire enterprise. Logical Data Models depict the logical entity types, typically referred to simply as entity types, the data attributes describing those entities, and the relationships between the entities. DDL can be generated at this level.
- **Physical data models (PDMs)**. Physical Data Models are used to design the internal schema of a database, depicting the data tables (derived from the logical data entities), the data columns of those tables (derived from the entity attributes), and the relationships between the tables derived from the entity relationships).

The level of detail that is modeled is significantly different for each model type. This is because the goals and audience 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. Each of the various models should also reflect your organization's naming standards. A Physical Data Model should also indicate the data types for the columns, such as integer or character.

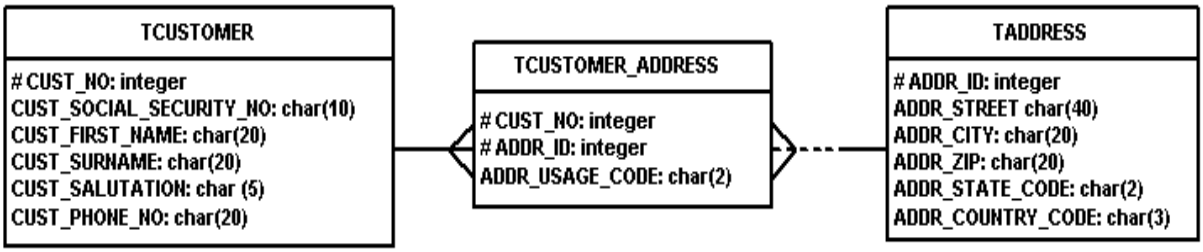
A simple conceptual data model:



A simple logical data model:



A simple physical data model:



Data models can be used effectively at both the enterprise level and on individual projects. Enterprise architects will often create one or more high-level Logical Data Models that depict the data structures that support the enterprise, models typically referred to as enterprise data models or enterprise information models. An enterprise data model is one of several critical views that the organization's enterprise architects will maintain and support; other views may explore network/hardware infrastructure, organization structure, software infrastructure, and business processes (to name a few). Enterprise data models provide information that a project team can use; both as a set of constraints, and as important insights into the structure of their system.

Project teams will typically create Logical Data Models as a primary analysis artifact when their implementation environment is predominantly procedural in nature, for example they are using structured COBOL as an implementation language. Logical Data Models are also a good choice when a project is

data-oriented in nature; perhaps a data warehouse or reporting system is being developed. However Logical Data Models are often a poor choice when a project team is using object-oriented or component-based technologies where the developers typically prefer UML diagrams or when the project is not data-oriented in nature.

When a relational database is used for data storage, project teams are best advised to create a Physical Data Model to model its internal schema. A Physical Data Model is often one of the critical design artifacts for business application development projects.

Data Modeling vs. Class Modeling:
Data modeling is different from class modeling because it focuses solely on data.
It is important to do data modeling and to develop the ERD (Entity Relationship Diagram) to insure that the relational database is properly designed.

From the point of view of an object-oriented developer, class modeling is a useful approach. Class models allow you to explore both the behavior and data aspects of your domain. Similar to data entities, there are associations between classes; relationships, inheritance, composition, and aggregation are all applicable concepts in modeling.

It is very important to get project stakeholders actively involved in the creation of the model. Instead of a traditional, analyst-led drawing session you can instead engage stakeholders by facilitating the creation of models.

Data Perspectives:

Data Models are a valuable source of information; providing a graphical depiction of data at different levels of abstraction. For example, the owner of a business process is interested in the conceptual view of data – the conceptual model. The designer of the data is interested in the logical view – the logical model. This view is sometimes referred to as the transformation layer. The data administrator is typically concerned with this model. The database administrator is typically more concerned with the physical model and the physical implementation of a relational database. The table on the next page summarizes the different perspectives.

Model	Perspective	Model Description	Type of Model	Entity	Type of Relation
Business Model	Owner	Semantic Model	Conceptual	Business Entity	Business
System Model	Designer	Logical Data Model	Logical	Data Entity	Data
Technology Model	Builder	Data Design	Physical	Table/ Segments	Key/Pointer
Detailed Representation	Developer	Data Definitions		Field	Address

Data Modeling Standards Supported:
There are three common data modeling notations: Information Engineering (IE), IDEF1X, and the Unified Modeling Language (UML).

From a notation standard, current product standards should support both IDEF1X and IE modeling standards as well as naming standards which allow you to create glossaries of approved words and enforce the way words are used in naming tables and columns, etc.

XML (which has a very loose industry standard) should also be supported in the tool. It is likely that UML based tools will co-exist with other data modeling notations for the near future. UML provides all of the syntax needed to perform data modeling, as well as behavioral modeling. Some developers see an advantage to using one modeling language for all modeling purposes.

Common data modeling notations:

Notation	Comments
IE	The IE notation (Finkelstein 1989) is simple and easy to read, and is well suited for high-level logical and enterprise data modeling.
IDEF1X	This notation is more complex. It was originally intended for physical modeling but has been applied for logical modeling as well.
UML	This is not an official data modeling notation (yet). Considering the popularity of the UML, the other data-oriented efforts of the Object Management Group (OMG), and the lack of a notational standard within the data community, it is only a matter of time until a UML data modeling notation is accepted within the IT industry.

The current standards either support the use of IE and IDEF1X notations or UML. (Reference Data Modeling Product Standards for the latest version.)

Model Reuse:
A logical data model 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.

Model Review Process:
A review process with appropriate team members and stakeholders should follow after each stage of model development. This process will ensure that the respective data model correctly and accurately represents the business requirements and maximizes data integrity. NOTE: A separate effort is currently underway to develop a formal data modeling methodology, including templates and checklists to validate logical and physical data models.

Data Modeling Best Practice Standards Supported:

A list of Data Modeling Best Practice Standards has been compiled by the Information Domain Team. These standards have applicability across all current standard products and are required to be used for all application development efforts of sufficient size and scope. If a specific standard applies only to mission-critical applications, it will be identified as such. Reference Data Modeling Best Practice Standards for the latest version.

Data Modeling Basic Steps

1. Identify entity types - an entity type represents a collection of similar objects. An entity could represent a collection of people, places, things, events, or concepts. Examples of entities in an order entry system would include *Customer*, *Address*, *Order*, *Item*, and *Tax*. If you were class modeling you would expect to discover classes with the exact same names. However, the difference between a class and an entity type is that classes have both data and behavior whereas entity types just have data. Ideally an entity should be "normal", the data modeling world's version of cohesive. A normal entity depicts one concept, just like a cohesive class models one concept. For example, customer and order are clearly two different concepts; therefore it makes sense to model them as separate entities.

2. Identify Attributes - each entity type will have one or more data attributes. For example, the *Customer* entity has attributes such as *First Name* and *Surname* and the *TCUSTOMER* table had corresponding data columns *CUST_FIRST_NAME* and *CUST_SURNAME* (a column is the implementation of a data attribute within a relational database). Attributes should also be cohesive from the point of view of your domain, something that is often a judgment call. If you wanted to model the fact that people had both first and last names instead of just a name (e.g. "John" and "Doe" vs. "John Doe") whereas we did not distinguish between the sections of a zip code (e.g.

90210-1234-5678). Getting the level of detail right can have a significant impact on your development and maintenance efforts.

3. Establish Data Naming Conventions - Standards and guidelines applicable to data modeling should be set and enforced. Commonly, this would be the responsibility of a data administrator. These guidelines should include naming conventions for both logical and physical modeling, the logical naming conventions should be focused on human readability whereas the physical naming conventions will reflect technical considerations. The basic idea is that developers should agree to and follow a common set of modeling standards on a software project. Just like there is value in following common coding conventions, clean code that follows your chosen coding guidelines is easier to understand and evolve than code that doesn't, there is similar value in following common modeling conventions.

4. Identify Relationships - entities have relationships with other entities. For example, customers PLACE orders, customers LIVE AT addresses, and line items ARE PART OF orders. Place, live at, and are part of are all terms that define relationships between entities. The relationships between entities are conceptually identical to the relationships (associations) between objects.

5. **Assign Keys** - A key is one or more data attributes that uniquely identify an entity. A key that consists of two or more attributes is called a *composite key*. A key that is formed of attributes that already exist in the real world is called a *natural key*. An entity type in a logical data model will have zero or more *candidate keys*, also referred to simply as *unique identifiers*. Both of these keys are called candidate keys because they are candidates to be chosen as the *primary key*, an *alternate key* (also known as a *secondary key*), or perhaps not even a key at all within a physical data model. A primary key is the preferred key for an entity type, whereas an alternate key (also known as a secondary key) is an alternative way to access rows within a table. In a physical database, a key would be formed of one or more table columns whose value(s) uniquely identify a row within a relational table.

6. Normalize Data - Normalization is a process in which data attributes within a data model are organized to increase the cohesion of entity types. In other words, the goal of data normalization is to reduce, and even eliminate, data redundancy.

7. Optimize Performance - Normalized data schemas, when put into production, may suffer from performance problems. This makes sense – the rules of data normalization focus on reducing data redundancy, not on improving performance of data access. It may be necessary to denormalize portions of your data schema to improve database access efficiency. It should be documented why changes were made to the model.

Definition of Terms:

Aggregation
Aggregation is a technique that optimizes data retrieval by summarizing rows of a fact table according to a specific dimension.

Business Rule
A business rule stipulates specific business-related information that is linked to database objects. The information can be in the form of business facts or descriptions; or it might be formulas or algorithms, either client-based or destined for the server. Once defined, business rules can be applied through the database or application code generation.

Cardinality
Cardinality indicates the number of instances (one or many) of an entity in relation to another entity. You can select the following values for cardinality:

- One-to-one - One instance of the first entity can correspond to only one instance of the second entity
- One-to-many - One instance of the first entity can correspond to more than one instance of the second entity

- Many-to-one - More than one instance of the first entity can correspond to the same one instance of the second entity
- Many-to-many - More than one instance of the first entity can correspond to more than one instance of the second entity

Data Attribute
A term used in logical data models to describe a kind of fact common to all or most instances of an entity. Student ID is an attribute of the entity Student. The corresponding physical data model generally implements the attribute as a database column or field.

Data Element
An entity, attribute, database table, or database column used to represent business information in logical or physical data models. Users should be aware that the literature also defines data element to explicitly mean an attribute of an entity. However, as defined in this document, the term encompasses both entities and attributes in logical data models as well as tables and columns in physical data models**.**

Data Entity
A term used in logical data models to describe a class of persons, places, things, concepts or events of interest to the business, about which the business intends to keep facts. The corresponding physical data model generally implements the entity in a database table or view.

Dimension
Defines the axis of investigation of a fact. Is attached to a dimension table.

Domain
A way of identifying and grouping the types of data items in the model. This makes it easier to standardize data characteristics for attributes/columns in different entities/tables. Some database management systems (DBMSs) will implement domains as "User Defined Datatypes". Another feature of domains is in the maintenance of similar columns. If all "name" columns (LastName, CityName, ProductName, etc.) are defined as a common domain, then changing the datatype from char(40) to char(50) is a one-step procedure, rather than having to visit each table and search for the correct columns.

Enterprise class database management system - integrates multiple business processes or applications into a single DBMS and hardware platform. This is in contrast to creating application specific database management systems.

Entity
Person, place, thing, or concept that has characteristics of interest to the enterprise and about which you want to store information.

Inheritance
Inheritance allows you to define an entity as a special case of a more general entity. The entities involved in an inheritance have many similar characteristics but are nonetheless different. The general entity is known as a supertype (or parent) entity and contains all of the common characteristics. The special case entity is known as a subtype (or child), entity and contains all of the particular characteristics.

Logical Data Model
A structured representation of the data of importance to the business, in terms of entities, attributes, and their relationships including the business rules that govern them. The representation includes both graphical depictions and textual definitions. Logical data models are used to translate business requirements into data representations that are understandable to information systems professionals.

Logical Data Name
A unique identifier of an entity or attribute as stored within a logical data model or data dictionary. Logical names should consist of English words and must be understandable by the end user. Also known as the Business Name or Functional Name.

Physical Data Model
A structured representation of the data of importance to the business, in terms of database tables and columns along with their relationships, formats, and business rules that govern the data. The representation includes both graphical depictions and textual definitions. Physical data models are used exclusively by information systems professionals to deploy database systems using appropriate database software.

Physical Data Name
A unique identifier of an entity or attribute as implemented within one or more database systems. Physical data names are generally constrained by the limitations of the database software.

Referential Integrity
Referential integrity refers to rules governing data consistency, specifically the interaction between primary keys and foreign keys in different tables. Referential integrity dictates what happens when you update or delete a value in a referenced column in the parent table and when you delete a row containing a referenced column from the parent table.

Relationship
A relationship is a named connection or association between entities. Each relationship is drawn as a line connecting the two entity types; each relationship is given a name that indicates what information it imparts (relationships are named in both directions); the *type* of relationship (*cardinality* and *optionality*) is specified as follows: the line style (dash or solid) indicates optionality and the relationship ends indicate cardinality.

Source:
Materials referenced and used from datamodel.org and Ambysoft's data modeling website.