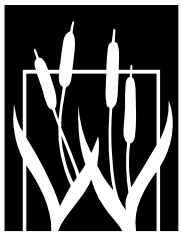

Whitemarsh
Information Systems Corporation

Data Model Management

Whitemarsh Information Systems Corporation
2008 Althea Lane
Bowie, Maryland 20716
Tele: 301-249-1142
Email: Whitemarsh@wiscorp.com
Web: www.wiscorp.com

Table of Contents

1. Objective

The objective of this ***Whitemarsh Short Paper*** is to present an approach to the creation and management of data models that are founded on policy-based concepts that exist across the enterprise. ***Whitemarsh Short Papers*** are available from the Whitemarsh website at:

http://www.wiscorp.com/short_paper_series.html

Whitemarsh's approach to managing data models is based on a "where-used" model versus the traditional "transformation-mapping" model. This difference brings a very great benefit to data model management.

2. Topics Covered

The topics is this paper include:

- Architectures of Data Models
- Specified Data Model Architecture
- Implemented Data Model Architecture
- Operational Data Model Architecture

3. Architectures of Data Models

If you are an IT professional but not very knowledgeable about database and data management and you ask the question, "What is a Conceptual Data Model?" you are likely to get confusing and conflicting explanations along the lines of any of the following:

Definition #1: From, http://www.1keydata.com

The features of conceptual data model include:

- Includes the important entities and the relationships among them.
- No attribute is specified.
- No primary key is specified.
- At this level, the data modeler attempts to identify the highest-level relationships among the different entities.

Definition #2: From http://en.wikipedia.org/wiki/Conceptual_schema

A conceptual data model is: A conceptual schema, or High-level data model or conceptual data model, is a map of concepts and their relationships, for example, a conceptual schema for a karate studio would include abstractions such as student, belt, grading and tournament.

Definition #3: From http://www.soi.city.ac.uk/~tony/dbms/conceptual_dm.html

Conceptual Data Modeling is the first stage in the process of top-down database design. The aim is to describe the information used by an organization in a way which is not governed by implementation-level issues and details. It should make it easy to see the overall picture so that nontechnical staff can contribute to discussions. A common method of analysis involves identifying:

- The entities (persons, places, things etc.) from the organization.
- The attributes - the items of information which characterized and describe these entities.
- The relationships between entities which exist and must be taken into account when processing information.
- In the abstract, or as illustrated by superficial examples, this looks like a very simple idea. An explanation may put forward a car as a typical entity, point to make, color, registration number as obvious attributes, and suggest owning and driving as relationships in which it takes part. But applying the method of analysis to some useful purpose in a working organization will be difficult, simply because the world does not fit so neatly into boxes.

Definition #4: From http://infogoal.com/dmc/dmcdmd.htm

Data modeling is the process of creating and extending data models which are visual representations of data and its organization. Data models exist at multiple levels including:

- The Conceptual Data Model describes data from a high level. It defines the problem rather than the solution from the business point of view. It includes entities and their relationships. Typically the conceptual data model is developed first.
- The Logical Data Model describes a logical solution to a data project. It provides more details than the conceptual data model and is nearly ready for the creation of a database. These details include attributes, the individual pieces of information that will be included. Typically the logical data model is developed second.

- The Physical Data Model describes the implementation of data in a physical database. It is the blueprint for the database. Typically the physical data model is developed third.

Upon hearing these four different definitions and supporting descriptions, the IT professional (but not Database or Data Management expert) would likely conclude that "Those data people just cannot get their stories straight!" No two are the same. Now, if that same question was given to a group of data management *experts*, a "civilized (hopefully) food-fight" would ensue. Each would insist that their definition was received by them from "on high." And finally, if a non-IT business user heard all these definitions, he might muse, "Thank heaven for the PC so I can do my own work, and not get a migraine from having to deal with these IT people."

There are however some common threads in these definitions.

- The conceptual design should be independent of any physical or DBMS details.
- The conceptual design should focus on concepts and relationships among concepts
- The conceptual design should include entities, relationships and possibly attributes

Whitemarsh does not disagree entirely with these common threads. Whitemarsh however, does not impost the restriction that the conceptual data model is the "first stage in the process of top-down database design." Rather, Whitemarsh believes that the conceptual design is not step #1 of a database's design, but rather a completely separate process of modeling concepts that, in turn, can be employed by the same or a different set of modelers in the modeling of one or more databases.

The Whitemarsh approach has five levels in the specifications of data-metadata. These are:

- Data Elements and Semantic Hierarchies (See Short papers 1, 2, and 3)
- Specified Data Models (commonly called conceptual data models)
- Implemented Data Models (commonly called logical data models)
- Operational Data Models (commonly called physical data models)
- View Data Models (commonly called SQL Views)

Whitemarsh calls the second through the fourth data models different from their common names for reasons that are explained in this short paper. This is done not just to be different but because of significant and beneficial data architecture reasons. The second through the fourth data model types are discussed. View models are not discussed as they have no direct bearing on data model management. The first model, Data Elements and Semantic Hierarchies are presented in Short Papers 1, 2, and 3. These can be found at:

http://www.wiscorp.com/short_paper_series.html

4. Specified Data Model Architecture

The Whitemarsh approach in the construction of the specified data model is similar to the approach for most conceptual data modeling in that it is based on concepts; there should be attributes about those concepts; and there should be relationships among the concepts. Since the Whitemarsh approach is not accomplished graphically, there has to be some strategy for materializing relationships. In Whitemarsh this is accomplished through the traditional relational data model concept: Primary and Foreign Keys. More on keys later in this paper. In actuality, that's what all ER modeling systems also do under the covers.

A first real difference in the Whitemarsh approach is that its conceptual models are set within subject hierarchies. A subject is a collection of related concepts. In traditional conceptual modeling there really is no "concept" within which the entities exist. Rather, in the traditional conceptual modeling it's really a database modeling process, where the first step is the database's conceptual form. In the Whitemarsh approach, it's a process to define the data models of distinct concepts. Each such concept is called a subject.

The goal then of any Specified Data Model effort is the construction of the models of data about concepts. That is, the subjects, entities, attributes, and relationships that fully define the data required for any concept or collection of related subject-based concepts. This stands in stark contrast to the traditional process that has a database's data model first in a conceptual form, then transformed into a logical form and then finally transformed into a physical form. It's database all the way along, just in different levels of refinement and/or transformation.

The traditional conceptual-logical-physical approach to data model development has as its goal the creation of *database* data models. A database ultimately is a "real" container. The *type* name for the container is "schema." An instance name might be Customer. Hence, Customer Schema. The schema represents the schematic of the data.

As a historical aside, the database term, schema, was used in the early 1970s by the ANSI/X3/SPARC Study Group on Database Management Systems. ANSI is the American National Standards Institute. X3 is the abstract name for the Computers and Information Processing ANSI subgroup. SPARC means Standards Planning and Requirements Committee. This group included persons from Honeywell, IBM, Eastman Kodak, Equitable Life, Boeing Computer Services, Deere and Company, University of Maryland, Exxon, Columbia University, CINCOM, the RAND Corporation, and NCR. Its report was never published in its entirety, but an edited version, just 17 pages long, was released by Dennis Tsichritzis and Anthony Klug. The edited report, sometimes called the ANSI Three-Schema Architecture, set fourth a generalized set of three-schemas, rather than the 42 that were identified and defined by the full committee. The three generalized schemas were:

- Conceptual Schema
- External Schema
- Internal Schema

The external schema is today's View Model and is the interface between application systems and the database managed by a database management system (DBMS). The internal schema is the

schema specifically tailored to and understood by the specific DBMS, e.g., Oracle or Sybase or DB-2. Its today's physical data model. The conceptual schema was to then represent the independent representation of what today would be all the tables, columns and relationships. In the early 1970s the relational model had barely been born. Today, there's a real battle as to what the ANSI SPARC committee really meant by the Conceptual Schema. Is it today's Logical Schema, or is it the Conceptual Schema without a Logical Schema. Or, was there one schema missing? Because of advances in technology and understandings, that question is really not worth answering. Suffice it to say, that today there are four commonly recognized schemas: Conceptual, Logical, Physical (internal), and View (external).

History aside, the key question then becomes, what should be the role of a conceptual schema, and what should its relationship be to the other schemas, that is, to the logical, physical, and view models? As stated above, the Whitemarsh approach has its conceptual models set within subject hierarchies, where the subject is a collection of related concepts, and then entities, attributes, and relationships are defined within the subject's scope. Within the Whitemarsh Specified Data Model approach there is no notion of a database and/or a schema. The Whitemarsh conceptual models of data are just that, data models about concepts.

A few examples serve to illustrate the point: There is the concept of a person, and within that there might be biographic information, family information, employment information, and the like. The Whitemarsh specified model concentrates solely on the creation of a data model around the person concept. Similarly there could be models for financial instruments, postal and physical addresses, and real or abstract property.

Each of these concepts is first expressed as a single subject or within a subject hierarchy. Then, the main entities about that concept are identified. Then, the attributes about those entities are identified, and finally, the relationships across the entities are defined.

In the example above for person, Person may well be both the subject and the first entity. Other entities related to the Person could be Biographic Information, Family Member, Employment Information, and the like. For each entity the attributes would be set down such as Biographic Information Name, Biographic Information Description, or more specially such as Birth Date, Death Date, City of Birth, Current Weight, Current Height, Gender, and the like. The relationships would be that there is:

- Zero, one, or more than one Biographic entity instance for every Person,
- Zero, one, or more Family Member entity instances for the Person
- Zero, one, or more Employment entity instances for the Person

Relationships between entities are of two types: implied or explicit. An implied relationship is merely a line between the two entities where one end of the line represents the "1" entity (a.k.a., source), and the other end of the line represents the "O, 1, or more" entity (a.k.a., target). Since data modeling tools don't really store "lines," what is actually stored in the "1" entity is the automatic creation of an attribute that might have the auto-generated name, <source-entity-name>_ID, as in Person_Id, or Biographic_Information_Id. What is then automatically created

and stored in the "O, 1, or more" entity is another version of the attribute that commonly has the name, <target-entity-name><source-entity-name>_Id. An example would be Biographic_Information_Person_ID. This is simple and clean. Most commonly, the Person_Id is referred to as the Primary Key, often has a key name of PersonPrimaryKey, and also has the Primary Key column, Person_Id. The key in the Biographic_Information entity is called a Foreign Key. That is because it is not "native" to the Biographic_Information entity. The key name is commonly, BiographicInformationPersonForeignKey with the foreign key column, Biographic_ Information_Person_ID.

Now, while this strategy is simple and easy to accomplish, it is not sufficient. The reason is that there is an implied semantic quality of instance uniqueness based on the *natural* values represented by the attributes within the entity. So, in the case of Person, if the only attribute within the primary key is Person_Id, then the values could be 1, 2, 3, and so on. Nothing however would prevent the same person being stored multiple times. So, while the instances would be unique based on the contrived primary key column values, there would be person duplicates based on the "real" attributes values. E.g, 17 Mary Smith's all numbered differently. Because of this potential real problem, Whitemarsh strongly recommends that natural attributes whose values taken together ensure uniqueness be the attributes that are assigned to the Primary Key. In this example, it might be the four attributes: PersonFirstName, PersonMiddleName, PersonLastName, PersonBirthDate, and PersonBirthCity. When this is accomplished in the specified data model then there is rigor behind the relationships. These business semantics based keys are sometimes called "Natural keys." The other type of key, the "ID" key, is sometimes called a Surrogate Key.

Collectively, the characteristics of a Whitemarsh Specified Data Model are that:

- A subject that relates to a concept, not to a schema or a database.
- There are collections of hierarchically organized subjects of concepts.
- Attributes exist within each entity of that subject concept.
- Relationships exist across entities, and thus, across concepts, and are manifest by primary and foreign keys
- The primary keys are based on natural attributes of data.

The value from this approach is the Specified Data Model represents data models of concepts, which are then employed in the construction of Implemented Data Models.

Finally, the reason why Whitemarsh does not employ the name Conceptual Data Model has mainly to do with too many conflicting definitions. Some of the differences are on the surface, but the two that really cause concern are that:

- A conceptual data model is just an "undeveloped or unrefined" database data model, or

- A conceptual data model is really just a model of entities, attributes and optionally relationships.

These characteristics of the Whitemarsh Specified Data Model are different enough from the normal uses of the traditional term, Conceptual Data Model, to cause Whitemarsh to use an entirely different term, Specified Data Model.

5. Implemented Data Model Architecture

An Implemented Data Model is not a refined Specified Data Model. Rather it is a completely different data model wherein its set of tables, columns, and relationships are derived from one or more Specified Data Models. That single statement alone is a most critical difference. Because of this very large difference and for other reasons set out below, Whitemarsh again uses its own term, Implemented Data Model rather than Logical Data Model. The Implemented Data Model consists of a traditional triple:

- Schema
- Table
- Column

The Implemented Data Model is supported by Primary and Foreign Keys, and also Candidate Keys. Primary and Foreign Keys are explained in the previous section. A Candidate Key, also called an Alternate Key because it is an alternate to the Primary Key. Columns are identified in each of these keys.

Traditionally, the difference between Conceptual and Logical Data Models is that the Conceptual Data Model is just a conceptual version of the Logical Data Model. Not so with the Whitemarsh approach. In the Whitemarsh approach, the only relationship that exists between the Specified Data Model and the Implemented Data Model is that every column in every table of every schema of the Implemented Data Model must be mapped either to a known attribute or to a special attribute called “Unknown” that exists within an “unknown” entity of an “unknown” subject of the Specified Data Model. Here are three “known” cases:

Case 1: One or more attributes from an entity of the Specified Data Model may be mapped multiple times to columns of a single table of an Implemented Data Model. For example, suppose the attributes for Telephone Number from the entity Telephone within the Subject Telephonic Communications are:

- Country Code
- Area Code
- Exchange
- Number
- Extension

These attributes could be mapped to three different columns in the Implemented Data Model that is, to:

- Salesman Phone Number
- Sales Manager Phone Number
- Billing Agent Phone Number

If the Implemented Data Model employed capabilities from the SQL:1999 standard, then each of those Implemented Data Model columns could be a RowType and have the attributes of Country Code, ..., Extension as contained subelements. Otherwise there would have to be the following:

- Salesman Phone Country Code
- Salesman Phone Area Code
- Salesman Phone Exchange
- Salesman Phone Number
- Salesman Phone Extension

And then the same for Sales Manger Phone Number and Billing Agent Phone Number.

Case 2: One or more attributes from an entity may be mapped to columns of multiple tables within the same schema. An example of this would be to have the Phone number associated with an Order Header and an Order Detail as follows:

- Order Header Table
  - Buyer Phone Number

- Order Detail Table
  - Product Manager Phone Number

This supports standard semantics across all the columns in all the tables regardless of the local column name.

Case 3: One or more attributes from an entity may be mapped to columns of multiple tables within different schemas. An example here would be reference data such as Corporate Product Names and Codes. If the Product Code attribute was mapped to a corporate standard set of values, then by inheritance, all the columns to which the attribute is mapped also have that same set of values.

The key value from the difference between the traditional mapping of conceptual to logical, is that through the Whitemarsh approach, it is not a “transform” mapping but a “where-used” mapping. And, the mapping can be one attribute to multiple columns in one table, or

multiple tables, or multiple tables in multiple schemas. This greatly enlarges the value and power of any conceptual data modeling.

6. Operational Data Model Architecture

The Operational Data Model is similar to the Implemented Data Model but with the following addition: There can be Secondary Keys. These are keys whose values are not required to be unique. These are commonly used as traditional indexes into a database. Similar to the implemented data model but specially tuned for a particular DBMS, the triple is:

- DBMS Schema
- DBMS Table
- DBMS Column

Now, in an Operational Data Model, it is very common to replace Natural Keys with Surrogate Keys. Given the Column [of the Implemented Data Model] to DBMS Column [of the Operational Data Model] mapping ability, that's perfectly reasonable. The Natural Key in the Implemented Data Model would be retained as a Candidate (or Alternate) key in the Operational Data Model. All Foreign Keys would then be based on Surrogate Key "ID" columns. That's clearly more efficient in DBMS performance.

There is an additional real benefit in the Whitemarsh where-used mapping versus the transform-mapping that relates to the Implemented and Operational Data Models. In data warehouse databases, its data model is a joining together and re-engineering of tables and columns from multiple databases. Thus, because of the where-used mapping capability, multiple tables and column sets from different Implemented Data Model Schemas can be mapped to just one DBMS schema, table and column set of the Operational Data Model. This mapping ability is impossible under a transform-mapping strategy.

7. Conclusions

The differences set out in this paper are NOT differences without distinction. The differences are great indeed. Foremost among them is that the mapping between data models is a where-used mapping, not a transform-mapping. All other differences, while important, pale in contrast.

If there is merely a transform-mapping among the three forms of essentially the same data model, then all we would really have in a metadata repository is a large collection of unrelated, unintegrated, and likely semantically conflicting data models. That's not much to brag about. What brings about the where-used mapping is a sophisticated data model metadata management environment.

As for pretty graphical pictures, any metadata data model repository environment can export any data model in SQL DDL and then have any ER modeling tool such as ERwin, ER-Studio, DeZign, or even WinSQL draw it handily.

8. References

The following references to Whitemarsh materials provide a more detailed exposition practical application of the significant content of this paper.

The following references to Whitemarsh materials provide a more detailed exposition practical application of the significant content of this paper.

The following documents are available free from the Whitemarsh website:

Paper	URL
Comprehensive Metadata Management	http://www.wiscorp.com/ComprehensiveMetadataManagement.pdf
Metabase Overview	http://www.wiscorp.com/Metabase.zip
Whitemarsh Data Modeler, Architecture and Concept of Operations	http://www.wiscorp.com/MetabaseDataModelerArchitectueandConceptofOperations.zip
Metabase User Guides	http://www.wiscorp.com/MetabaseUserGuides.zip

The following documents are available for Whitemarsh Website Members. The URLs that follow provide descriptions of the pages. Members should log in and proceed to the appropriate page, e.g., Enterprise Database, find the book, paper, or course and perform the download.

Paper	URL
Data Management Program - Metadata Architecture For Data Sharing Data Management Program - Database Interface Architectures Data Management Program - Projects And Data-Asset Product Specifications Data Management Program - Work Breakdown Structures Knowledge Worker Framework Database Objects Managing Database - Four Critical Factors	http://www.wiscorp.com/wwmembr/mbr_products_edb.html
Work Breakdown Structures	http://www.wiscorp.com/wwmembr/mbr_products_dp.html
Data Architecture Classes Guidelines for Data Architecture Class - Data Warehouse Iterations of Database Design	http://www.wiscorp.com/wwmembr/mbr_products_dd.html