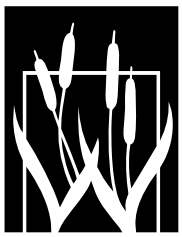

Whitemarsh
Information Systems Corporation

Data Architecture Reference Model

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

1.0 Introduction 1

2.0 Establishing the Right Data Architecture Terminology 2

3.0 The Whitemarsh Data Architecture Reference Model Components 4
- 3.1 Data Element Model 7
- 3.2 Specified Data Model 9
- 3.3 Implemented Data Model 10
- 3.4 Operational Data Model 11
- 3.5 View Data Model 13
- 3.6 XML Schema Data Model 14

4.0 Deployment of Data Architecture Models within Business Information System Development 15

1.0 Introduction

First things first: what is a reference model? From Wikipedia, *a reference model is a model in systems, enterprise, and software engineering is an abstract framework or domain-specific ontology consisting of an interlinked set of clearly defined concepts produced by an expert or body of experts in order to encourage clear communication.*

Given that, a data architecture reference model is in the area of data engineering and serves as a framework for understanding the scope and interrelationships among clearly definable sets of component models that address data. Accomplished, such a model enhances clear communication among deployed instances of the data architecture model's contained components.

After many years of evolution in understanding data from its natural conception through its deployment via databases, and business information systems, Whitemarsh has posited that a data architecture consists of four distinct technology independent models and then two technology-dependent interface models. The four main models depicted in Figure 1 are:

- Data Element Model
- Concepts (Specified) Model
- Logical (Implemented) Database Model
- Physical (Operational) Database Model

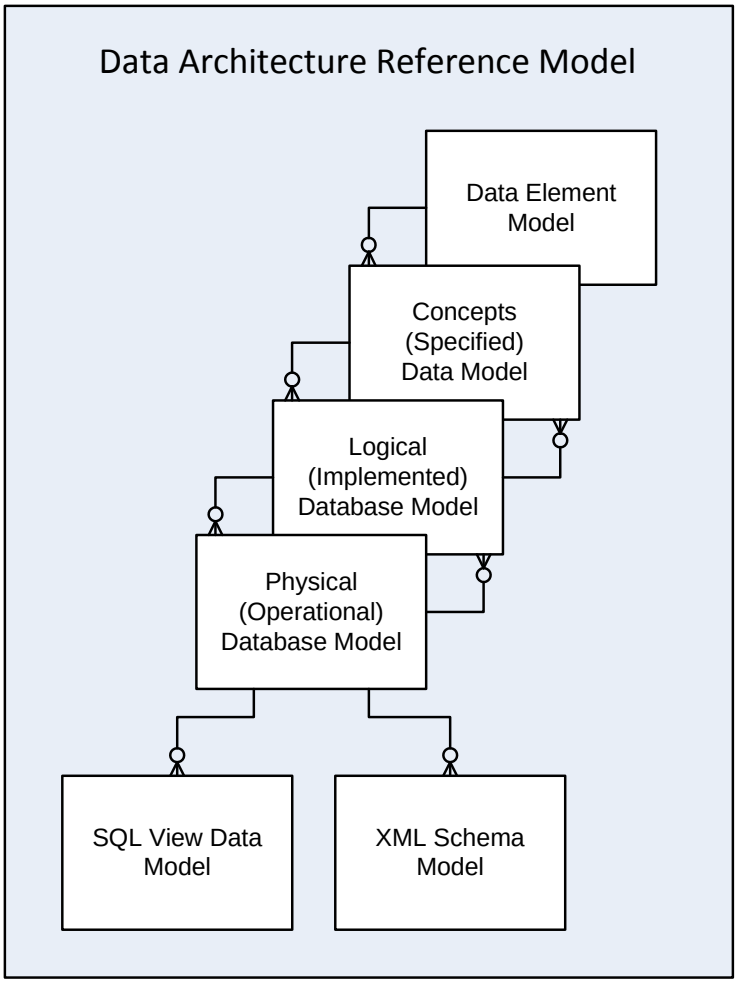

Figure 1. Data Architecture Reference Model.

The two technology dependent interface models are:

- SQL View Model
- XML Schema Model

2.0 Establishing the Right Data Architecture Terminology

The data management industry, as a whole, commonly refers to the first grouping of models as:

- Conceptual
- Logical
- Physical

The problem with this generally accepted industry understanding is that is it presumes that these models are three different forms of a same instance, that is, a Data Model. That is, for a given data model for example, Marketing, there is the Marketing data model's Conceptual Form, and thereafter, the Marketing data model's Logical Form, and finally, the Marketing Model's Physical form.

A saying from H. L. Mencken comes to mind: *For every complex problem there is an answer that is clear, simple, and wrong.* The problems with this understanding are:

- There is no concept of re-use of data-specified components within and across data models. Re-use is however very common.

- Because of this lack of re-use, there is a redundant set of design activities for the same data-specified component across a multitude of the enterprise's data models that inevitably leads to unnecessary semantic differences, which, in turn, cause "data problems" that have plagued us for scores of years..

- There is a generally accepted relationship of one-to-many between Conceptual to Logical to Physical. Not only is this inefficient and simply not born out in facts.

Because of the very closeness of the terminology,

Whitemarsh Data Component	**Data Management Industry Data Component**
Concepts	Conceptual Data Model
Logical Database Model	Logical Data Model
Physical Database Model	Physical Data Model

the terminology closeness inevitably leads to a conclusion of a "difference without distinction." The distinctions that are both clear and significant include:

First the industry term, Conceptual Data Model is completely misleading within Whitemarsh because the data management industry a Conceptual Data Model is a conceptual form of a data model. In Whitemarsh, a concept data model is simply the data model of an encapsulating concept. Thus, commonly existing within the data management industry, there would be a conceptual data model for Marketing that would contain whole collections of entities related to all the different aspects of marketing such as customers, locations including countries, states, and cities, marketing organizations including national, regional, districts, and the like, campaigns for sales and promotions, staffing results, contacts, products, and the like. Such a conceptual data model for Marketing could easily embrace several hundred entities.

In contrast, within Whitemarsh, a concept data model is simply the data model of a concept. An example might be Address. There might be two subtypes for commercial and residential. Another concept might be person name which would be Title, First Name, Middle Name, Last Name, and Name Suffix. These two examples illustrate that there is quite a significant difference in scope and intent.

Second, the industry commonly employed terms, Logical and Physical Data Model, in addition to being just two different development forms of the same "thing" (that is a data model), are without regard to the encapsulating concept, Database. A database is encapsulated in a Schema which is a "hard boundary" relationship mechanism that precludes syntactically expressible relationships between tables in different schemas.

In addition to the "database" difference, within Whitemarsh, the terms Logical and Physical connote technology independency differences. A Logical Database is without regards to the technology dependancy on a DBMS vendor (i.e., Sybase vs DB2 vs Oracle). In contrast, a Physical Database is dependent on such bindings and/or restrictions.

In this regard, the ISO/ANSI standard for SQL is the controlling mechanism for Logical Database model components while, for example, Sybase, DB2 or Oracle might be the controlling mechanism for Physical Database model components. This can be a difference with very dramatic distinctions as anyone knows who's been acquainted withe SQL standards across SQL:1986 , SQL:1989 , SQL:1992 , SQL:1999 , SQL:2003 , SQL:2008, and SQL:2012.

The most dramatic difference is that first three SQL standard conformed more than not to the relational model while there was a dramatic departure from the relational model starting with SQL:1999. Because of all these differences, Whitemarsh refers in all its materials as follows:

Data Model Type		**Whitemarsh Term**
Concepts	Is referred to as the	Specified Data Model,
Logical Database Model		Implemented Database Model
Physical Database Model		Operational Database Model

3.0 The Whitemarsh Data Architecture Reference Model Components

The Whitemarsh Data Reference Model consists of six classes of data models, which are all interrelated one to the other. These are depicted in Figure 1. The objective of these six models is to significantly increase semantic interoperability, increase re-use, eliminate redundancy, and reduce the overall cost of project specification, implementation, and maintenance. The effect of having these six models is a way to decrease overall project risk while at the same time increasing quality. The consequence of not having these models is the converse of these objectives and effects.

Table 1 enumerates and briefly describes each of these six data model classes. These six data model classes have real, practical, and existence implications for enterprises. To wit: The Y2K data debacles were a direct consequence of enterprise failure to have these six data models within a non-redundant, unambiguous and integrated data model management environment. The NASA 1999 Mars Climate Orbiter crash was directly tied to fact that one organization was computing with "metric" unit measurements and another was computing with "English" unit measurements. The existence of a Data Element Model semantics capstone would have controlled data specifications, implementations, and operations would have surfaced this semantic disconnect.

This data architecture reference model document does not address how to create each of these data models because the creation process is more than adequately addressed within other Whitemarsh books, courses, seminars, and methodology documents. Rather, the focus of this document presumes is to assess the necessary and sufficient interaction between these data models and the other key business information system work products.

Data Model Class	Description
Data Element	Data elements are the enterprise facts that are employed as the semantic foundations for attributes of entities within data models of concepts (Specified Data Models), columns of tables within database models (Implemented Data Models) that support the requirements of business and are implemented through database management systems (Operational Data Models), that, in turn, are employed by business information systems (View Data Models) that materialize the database objects necessary for within the resources of the enterprise that support the fulfillment of enterprise missions. Key components are Concepts, Conceptual Value Domain, Data Element Concepts, Data Elements and Value Domains. Semantic and data use modifiers can be assigned to every data element concept and data element.
Specified Data Model	Specified Data Models are the data models of concepts. These models consist of subjects, entities, attributes, and inter-entity relationships. Relationships can interrelate entities within multiple subjects. Each data model should address only one concept such as a person's name, or an address, etc. These Specified Data Models can be templates for use in developing database models (Implemented or Operational). Every entity attribute should map to its parent data element. Semantic and data use modifiers can be assigned to every entity attribute. Key components are subjects, entities, attributes, and relationships.

Data Model Class	Description
	A Specified Data Model is a data model of a specific concept, represented as a container such as student, school, organization, or address. These containers (e.g., student or school) must be specified before they can be implemented in one or more different database collections of tables that ultimately become operational through a DBMS such as Oracle.
Implemented Database Model	Implemented Database Models, are the data models of databases that are independent of DBMSs. These models consist of the data structure components: schema, tables, columns, and inter-table relationships. Relationships are restricted to tables within a single schema. While each Implemented Database Model can address multiple Specified Data Models from the collection of Specified Data Models , each Implemented Database Models should address only one broad subject. Every table column should map to a parent Attribute. Semantic and data use modifiers can be assigned to every column. There is a many-to-many relationship between the Specified Data Model and the Implemented Database Models. Key components are schemas, tables, columns, and relationships
Operational Database Models	Operational Database Models, are the data models of databases that have been bound to a specific DBMSs. These models consist of the data structure components: DBMS schema, DBMS tables, DBMS columns, and inter-table DBMS relationships. DBMS Relationships are restricted to DBMS tables within a single DBMS schema. Each Operational Database Models can address multiple Implemented Database Models. Every DBMS Column should map to a parent Column. There is a many-to-many relationship between the Implemented Database Models and the Operational Database Models. Key components are DBMS schemas, DBMS tables, DBMS columns, and DBMS Relationships. In this state, that is, dependent upon a particular DBMS and upon the performance requirements of a particular software application, this data model is termed “physical.” These data models are the Operational Database Models that are bound to application business information systems through view data models. These data models are often not in third normal form as a way to meet needed performance requirements. DBMS Columns from the DBMS tables from within these Operational Database Models are deployments of a single column of a table from a Implemented Database Model.
View Data Model	The View data models represent the interfaces between Operational Database Models and business information systems. View and their view columns can be characterized as Input and/or Output. Additionally, these views can be mapped one to the other on a view column basis and processes can be specified to define any appropriate data value transformation. Key components are Views, View columns, and view-column interrelationships. View data models are bound to the particular DBMS through which they are defined. View data models enable application systems to select, employ, and update databases according to their physical data models without having to include physical data model details within the application systems.

Data Model Class	Description
XML Interface Data Models	A XML interface model represents a specialized construction of importable or exportable data with respect to the business information system. Each XML data stream is defined through a XML Schema. Both XML schemas and XML data streams are independent of the software applications that create and/or use them. XML Schemas and XML data streams are DBMS represented in plain ASCII text. Key Components are XSDs, XML elements, and XML attributes. The structure of XML data is expressed through an XML schema that is employed to then understand the contents of XML data records. XML schemas are created through special software applications. XML data streams are created by source application business information systems and are subsequently read and processed by target application business information systems.

Table 1. Data Model Layers within the Data Architecture Reference Model.

Figure 1 shows that the core of the data management environment is the data model environment. Within this environment there are six distinct data models. For example, the Data Element model captures the once-only identification, specification, and definition of data elements that may be represented as database table columns in many different database tables.

Similarly, there may be Specified Data Models, for example, for students, schools, organizations, or addresses. These Specified Data Models can be deployed in one or more Implemented Database Models, which, in turn are operationally deployed in one or more DBMS specific Operational Database Models.

Each of these six data model classes serve a special purpose and is interrelated with the other data models in some integrity-enhancing and work-saving manner. Section 4 provides definitions and interrelationships.

Figure 1 shows a left-side set of one-to-many relationships going "down." This relationship supports two meanings. The first is the mapping of an individual component of a model, and the second is the mapping of a whole collection from within a data model. In the Data Element model there can be individual data elements such as Person First Name, and there can be collections of data elements within a specific data element concept collection, for example, Person Related Information such as Person Identifier, Person Birth Date, Person First Name, Person Middle Name, and Person Last Name.

In the first type of left-side one-to-many relationship, the individual data element, Persons First Name would be semantically mapped to zero, one, or more attributes within different entities. For example, to Employee First Name, to Customer Contact First Name, or to Causality Insurance Contract First Name.

In the second type of "left-side" relationship, a whole collection of data elements can be mapped to a whole collection of attributes across one or more entities. For example, all the Data Elements within a Data Element Concept collection called Biographic Data Elements might be mapped to the entity, Person Information, or to the entity, Customer Contact Information. In this

case, the mapping of the data elements, Person Identifier, Person First Name, etc., is mapped to a corresponding set of attributes within one or more entities.

On the right-side of Figure 1, there is also a set of one-to-many relationships. This set, like the left-side one-to-many relationships has two meanings: individual component, and whole collections. The meanings of the right-side one-to-many relationship are different from the left-side one-to-many. The first type of right-side relationship, the mapping of an individual component is not one-to-many, but one-to-one. Thus, an individual DBMS Column, for example, EmpFrstNam can be inherited from only one higher level component, for example, the single column, EmployeeFirstName.

The second type of right-side relationship, the mapping of collections can be one-to-many. That is, one collection can map to one set of columns within one table of a single Implemented Database Model while another collection from the same Implemented Database Model can be mapped to a different collection within a different Implemented Database Model. Hence, the collections can be seen as "from" one Implemented Database Model to zero, one, or more Implemented Database Model.

3.1 Data Element Model

The very first model within the Whitemarsh Data Architecture Reference Model is the Data Element Model. Here, a Data Element is a context-independent business-fact semantic template. Its function is to be the overarching container of semantics for:

1. Business facts that can be deployed across collections of entities from one or more entities within different subjects.

2. Business facts that can be deployed across collections of columns from one or more tables within different logical database schemas.

Data Elements are not simple. Rather, they are the result of a cascading set of data-related understandings and refinements. First there are data-related concepts that, when bound into Information Technology environments, become data. Here a concept that ultimately would results into specifications for databases might be Persons, Real Property, Abstract Property, Geography, Transportation, Finance, Manufacturing, and the like.

While Concepts are the abstract textual descriptions that ultimately form the basis for data specifications, they are not the descriptions of the data types that can be employed to build data elements. Needed are the Conceptual Value domains and examples include Dimensions, Weights, Geographic Coordinates, Strings, and Money.

When Concepts and Conceptual Value domains are joined they act as hosts for a more refined set of conceptual value-domain based semantics are called Data Element Concepts. Examples of Data Element Concepts include, for a person, Biographic Information, Skill

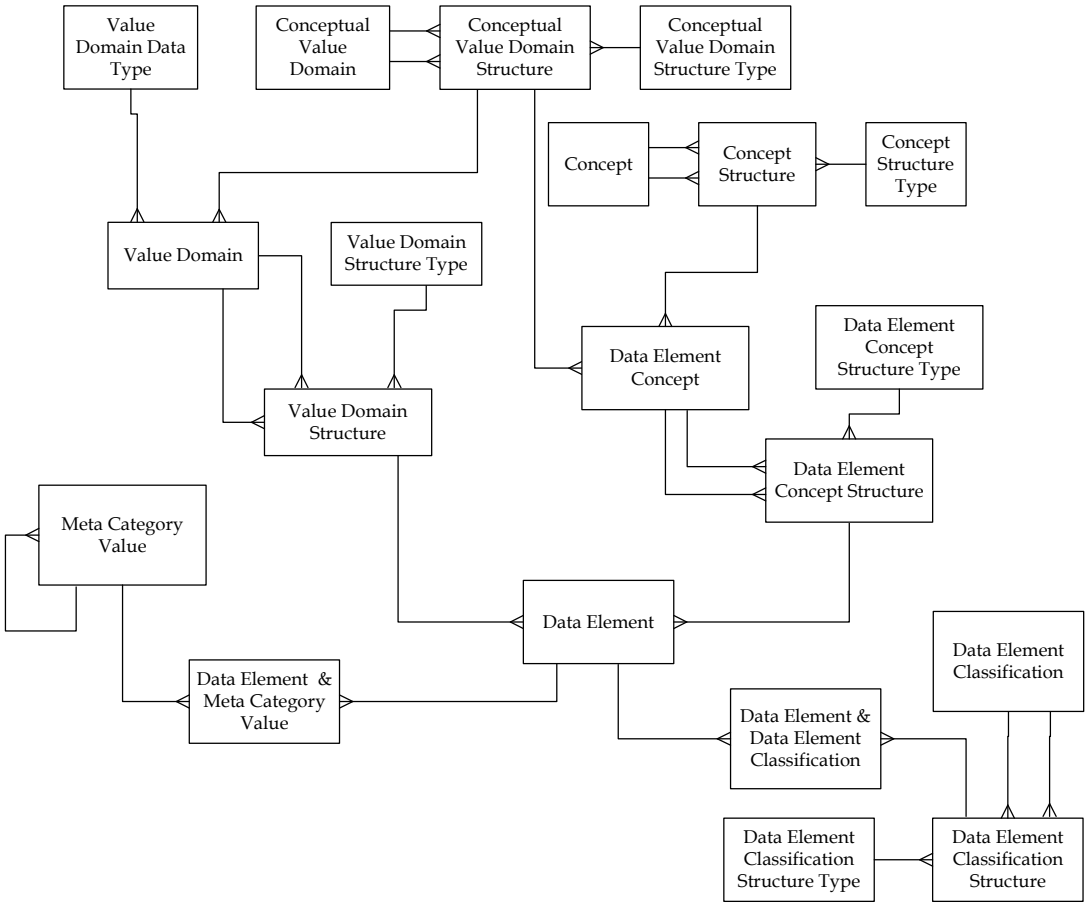

Figure 2. Data Element Model.

Information, Person Past Employment Information, and the like. For Finance it might include Ledger Specification Information, and Finance Transaction Information. For Geography it might include Geographic Locations. For each of these Data Elements there is a well defined concept, and assigned value domains.

Conceptual Value Domains are further refined into well known Value Domains. Included would be Money, Gender, Metric Dimension units, Weight Units, Geographic Point Specifications, and the like.

Subordinate to Value domains are explicitly allowed values. For example, a range of dates for allowed genders, Job Classifications, Employment Separation Codes, and the like. These Value Domain Value sets are thus bound into specific data elements, attributes, or columns.

The final joined-component pair for Data Elements are Data Element Concepts and Value domains. An enumeration of data elements results. Included might be PersonFirst, Street

Name, House Number, Financial Instrument Identifier, Financial Instrument Amount, Real Product Name, and the like.

Once the Data Elements are determined, they are ready to be employed as semantic templates for use in Specified, Implemented, and Operational Data Models.

3.2 Specified Data Model

A Specified Data Model is a data model of concepts that is independent of use within specific databases. Specified Data Models of concepts are defined through entities, attributes and relationships within specific subjects. There is a many-to-many relationship between the Specified Data Model and the Implemented Data Model. The meta model for the Specified Data Model is presented in Figure 3.

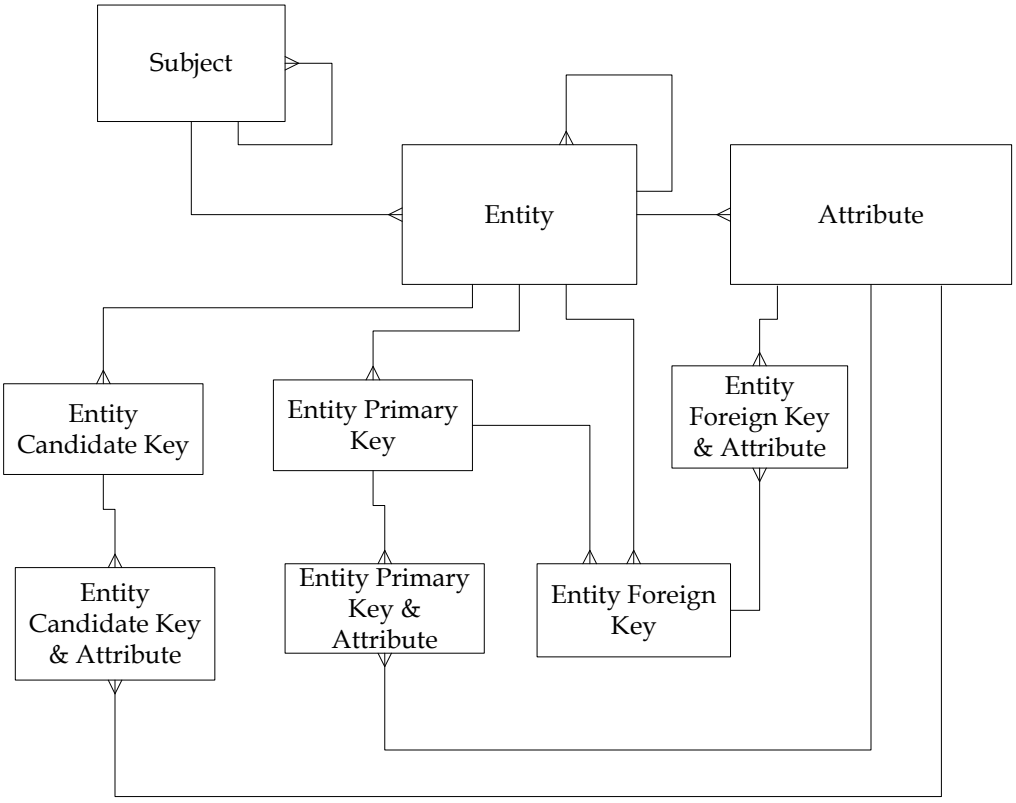

Figure 3. Specified Data Model.

In Specified Data Models, entities can have subtypes to any depth. An example might be for education which might have a common set of attributes for all classes of education, and subtypes for each different education class such as high school, college, or trade school.

Attributes represent the deployment of data elements within entities. Because of this one-to-many relationship between data element and attribute, users can see which attributes employ a data element's semantics regardless of the entity or subject.

Attributes can also be assigned various meaning modifier semantics and data use semantics. This enables automatic name construction for attributes. Automatic definitions and abbreviations are also enabled because there is a persistent relationship between the assigned semantic and the attribute. This too enables finding and reporting attributes regardless of entity and subject on the basis of either a semantic or data use modifier.

There can also be relationships among the entities within one subject, and there can be relationships between entities across subjects. If this were just a data model in a conceptual form, there could not be relationships across different subject-based data models. Each conceptual data model would be a stove-pipe that could only be transformed into a logical stove pipe data model and thereafter into a physical stove pipe data model.

Again, Specified Data Models are not conceptual forms of a data model. Rather, they are well-developed data models in their own right. Each entity within a Specified Data Model should be in third normal form. Specified Data Models are persistent. Their form remains as defined. Its structures, that is, the data models of the concepts, are available as templates for the construction of one or more Implemented Data Models.

3.3 Implemented Data Model

Implemented Data Models are models of databases that are independent of any particular DBMS such as Oracle or DB2. In the Whitemarsh Metabase System, Implemented Data Models can be exported to SQL DDL.

Implemented Data Models, illustrated in Figure 4 contain schema, table, column, SQL data types, primary key, primary key column, foreign key, foreign key column, candidate key, and candidate key column.

The Implemented Data Models obtain their data structures from one or more Specified Data Models, and in turn, provide these database data structures for use in the construction of DBMS specific database data models, that is, Operational Data Models.

Implemented Data Models are defined as tables, columns and relationships within schemas. Tables cannot be related across schemas. All the tables in this data model generalization level should be in at least third normal form to ensure data model clarity and quality design. There is a many-to-many relationship between an Implemented Data Model and an Operational Data Model. The reason is because a given Operational Data Model might be a data warehouse and some of its DBMS tables and columns could be drawn from multiple Implemented Data Models.

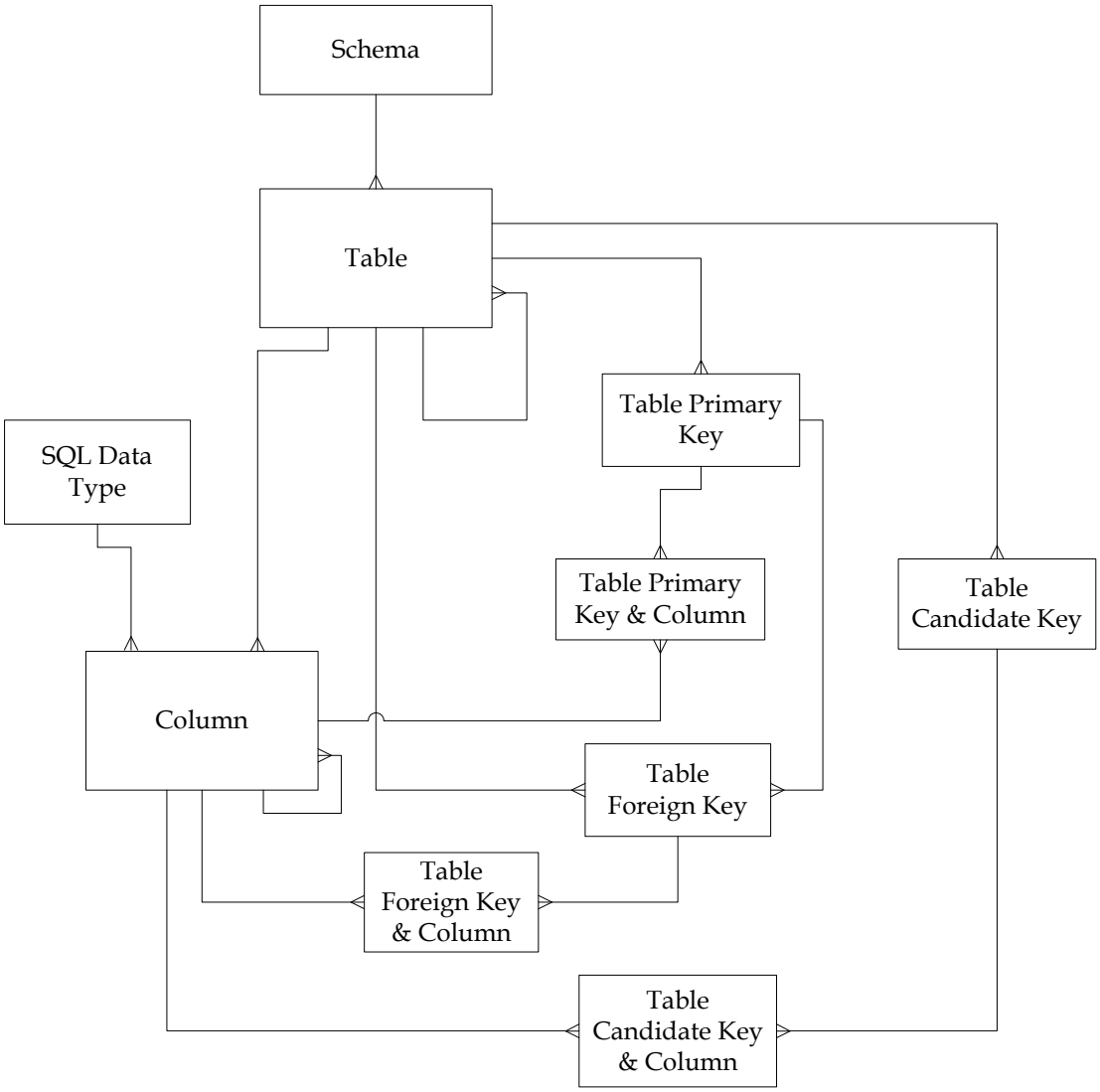

Figure 4. Implemented Data Model.

There can be relationships among the tables within one schema as well as data types, assigned value domains and other appropriate metadata for complete data models. There can also be assigned semantic modifiers, data use modifiers, and value domains assigned in the Implemented Data Model. Thus, as with the Specified Data Model there can be automatic name, definition, and abbreviation construction. Because the Implemented Data Model comes from a metadata management system database, any semantic assignment conflicts between the Specified and Implemented Data Models are immediately discovered and prevented. If the semantics

already assigned to attributes of entities within the Specified Data Model are sufficient, these are automatically inherited by columns of tables in the Implemented Data Model.

Each table may be constructed through the deployment of a collection of attributes from a single Specified Data Model entity, or collections of attributes from multiple Specified Data Model entities within one or more subjects. An entity or sub-collection of attributes from one entity can be used to form different data structures in multiple tables. The relationship between the Specified Data Model and the Implemented Data Model is thus, many-to-many, as it exists in real-world data modeling environments.

Tables in the Implemented Data Model can be subtyped. Columns can also be complex to support nested structures of arbitrary depth. The data types assigned to an Implemented Data Model are those allowed by the ANSI SQL standard.

Finally, because the Implemented Data Model is built from a foundation of the data models of concepts from the Specified Data Model, a complete cross reference between the two classes of models is easily reported.

3.4 Operational Data Model

Operational Data Models are models of databases that conform to the requirements of a particular DBMS such as Oracle or DB2. Additionally, the database's design must conform to the expected processing requirements of the particular set of database applications supported by the particular database. Operational data models are somewhat equivalent to the term, Physical Database. The meta model for the Operational Data Model is set out in Figure 5.

Operational Data Model designs vary because of many factors such as transaction volume, the host operating system, and the computer hardware size and throughput capabilities. Regardless, DBMS tables and/or DBMS columns of an Operational Data Model database should all relate back to tables and/or columns of an Implemented Data Model.

Operational Data Models are defined as DBMS tables, DBMS columns and relationships within DBMS schemas. "DBMS" is employed here to signify that these data model components are tied to a specific DBMS. There are many-to-many relationships between an Operational Data Model DBMS columns and View Data Model view columns.

The Operational Data Model is similar to the Implemented Data Model in its characteristics. The triple for the Operational Data Model is DBMS Schema, DBMS table, and DBMS column.

A key difference in the Operational Data Model from the Implemented Data Model is that it represents the data structures that result in SQL DDL streams that feed the DBMS to actually make the databases. In the Whitemarsh Metabase System, the Operational Data Model can export SQL DDL.

There may be multiple Operational Data Models for a given Implemented Data Model. Each Operational Data Model may contain a different subset of an Implemented Data Model's tables and columns, or in some cases, all its tables and columns. It may also be the case that a

complete set of a table's columns are transformed into differently named columns. For example, there might have been an Implemented Data Model table for telephone numbers that, in the Operational Data Model, are transformed into TelephoneNumber-1 through TelephoneNumber-5. This is a kind of Operational Data Model denormalization may be created to improve performance.

There may also have been multiple Implemented Data Models for a given Operational Data Model. This commonly occurs in data warehouse data model designs where the Operational Data Model for the data warehouse is sourced from multiple Implemented Data Models.

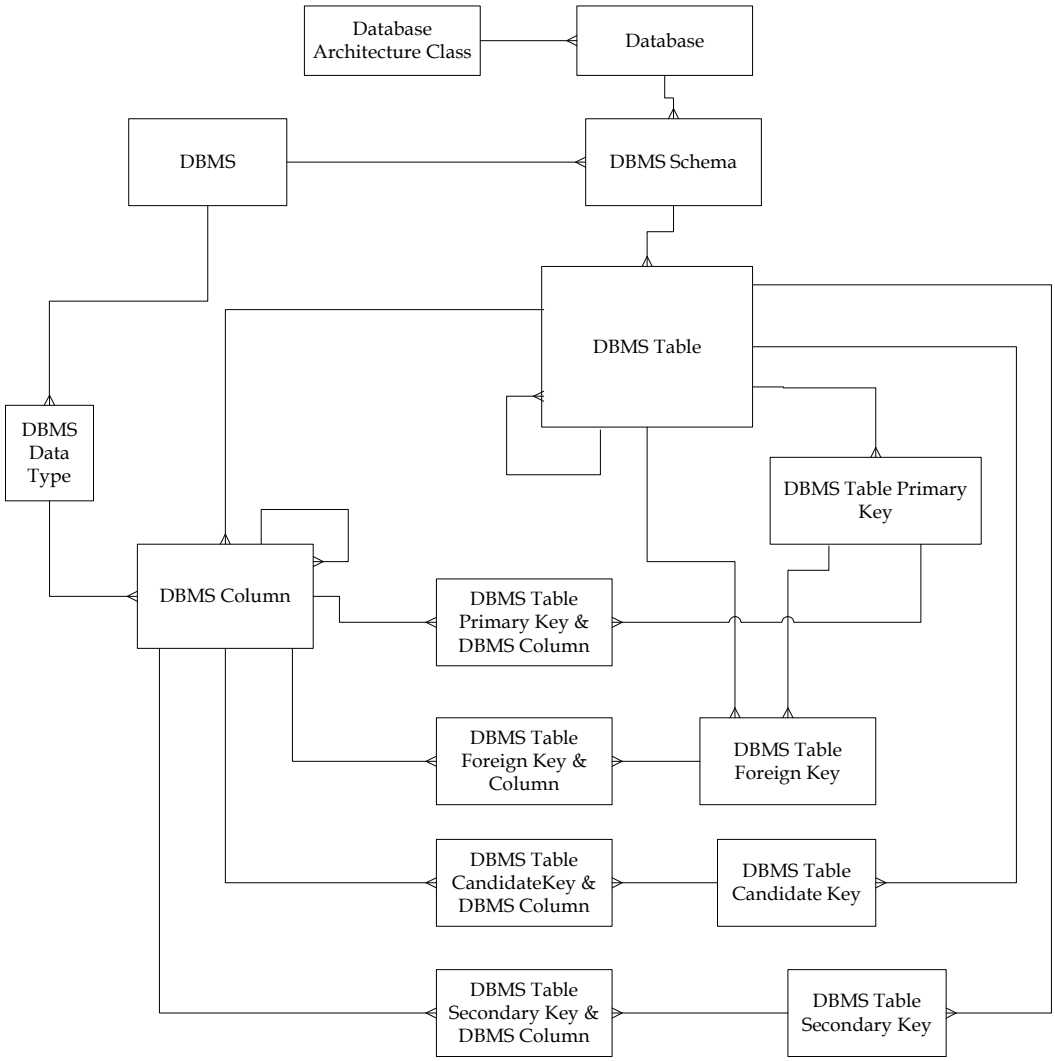

Figure 5. Operational Data Model.

The Operational Data Model can contain subtyped DBMS tables, and substructures in DBMS Columns. It can also have an assigned value domain and the data types reflect that of the containing DBMS.

3.5 View Data Model

View Data Models are the Business Information System and Operational Data Model intersection mechanisms. View data models, depicted in Figure 6, contain the following components: View, View Column, View Column & Compound Data Element, View Column & Derived Data Element, View Column & DBMS Column, View Column Structure, View Column Structure Process, and View Column Structure Type

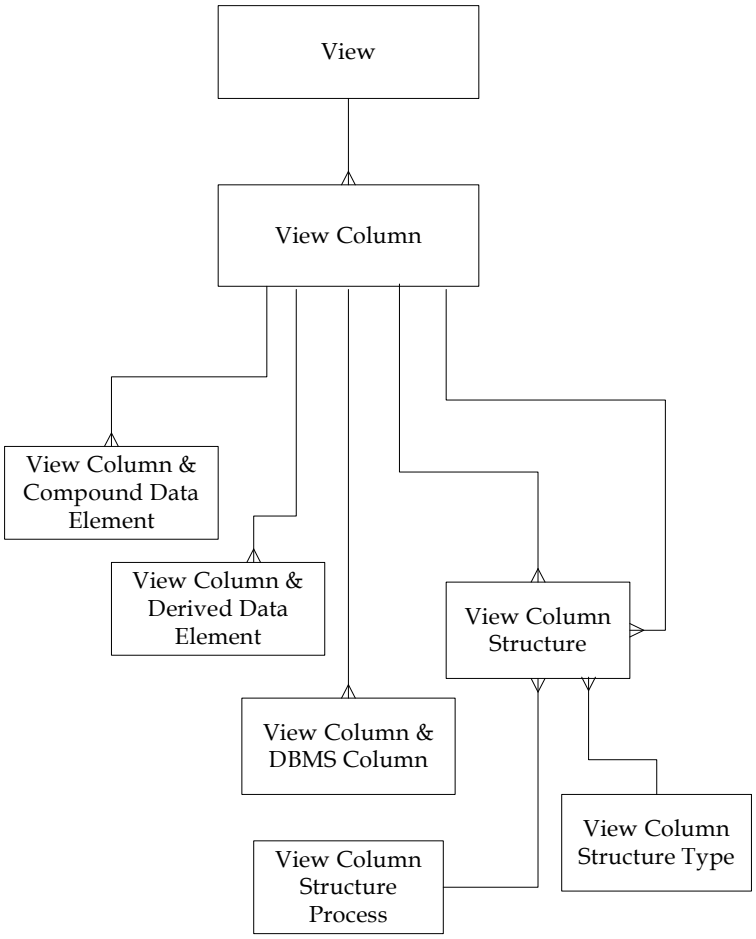

Figure 6. View Data Model.

View data models consist of views and view columns. They are also related to the business information systems that are the sources or targets of the database data. Views are specifically tied to specific DBMSs and view columns from a source view may be mapped to the view columns of a target view. Finally, view columns may be computed or derived.

View data models give applications a flat "record set" for processing. Included in views can be rename clauses and on-the-fly calculations. View columns not only map to DBMS columns but also, as appropriate, to compound data elements and derived data elements.

3.6 XML Schema Data Model

A XML interface model represents a specialized construction of importable or exportable data with respect to the business information system. Each XML data stream is defined through a XML Schema. Both XML schemas and XML data streams are independent of the software applications that create and/or use them. XML Schemas and XML data streams are DBMS represented in plain ASCII text. Key Components are XSDs, XML elements, and XML attributes.

The structure of XML data is expressed through an XML schema that is employed to then understand the contents of XML data records. XML schemas are created through special software applications. XML data streams are created by source application business information systems and are subsequently read and processed by target application business information systems.

4.0 Deployment of Data Architecture Models within Business Information System Development

We are all familiar with the collection of work products that are identified, engineered, created, evolved and maintained during the life cycle of business information systems. While there can be an endless array of names for some of these work products, they generally fall into the work product categories that are listed in column one of Table 2. A depiction of the interrelationships of the data models and of the work products is presented in Figure 7. Table 2 also shows the cross reference between the Data Architecture reference model data models and the typical work products of a business information system effort.

Business Information System Work Products, for example, Business Requirements, shown in Figure 7 can have a one-to-many relationship between one or more different business information system work products and zero, one or more of the data models. When two or more business information system work products are interrelated with one or more of the data models, there may exist relationship between those two business information system work products through artifacts contained in the data models. For example, there is a business requirement to capture student addresses. Additionally, there is a use case through which a student's address is

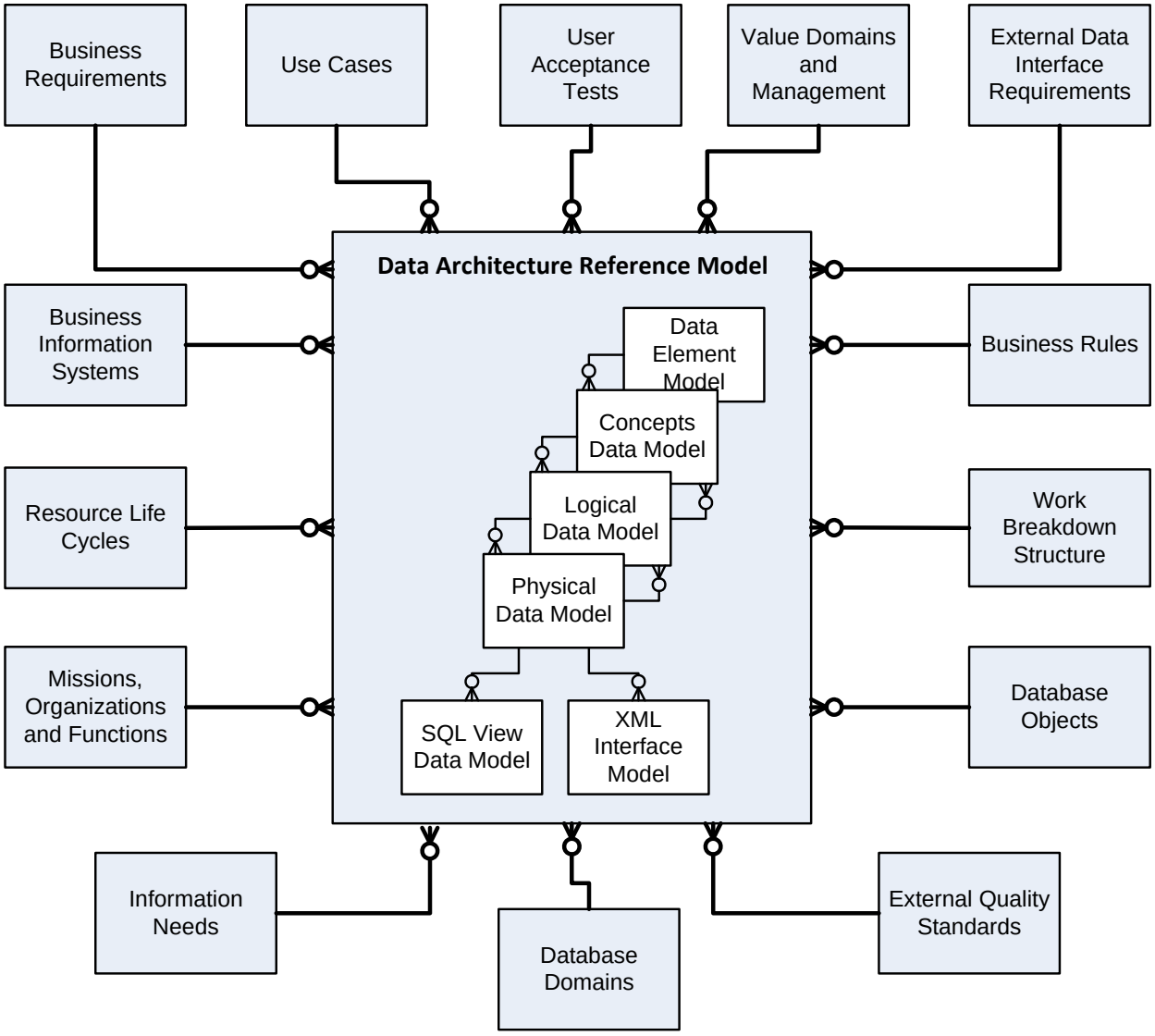

Figure 7. Cross Reference between Data Architecture Models and Business Information System Work Products.

captured. Apart from the obviousness of the interrelationship, the fact that there is a common data structure, Student Address, there is an interrelationship between the two business information system work products, business requirements and use cases. When both work products and the data models are stored in a comprehensive metadata management systems, the exposition of the business information system work product traceability of one work product to the other is a simple report request.

Table 2 identifies each of the business information system work products in the data management environment are listed in alphabetical order in Table 2. These are cross referenced with the data models with which they are involved in Table 2. The data models from Figure 2 that are identified in Table 2 are not accomplished in isolation. They result from an initial and then on-going analysis of the business information system work product. That is, Business Requirements, Business Rules, and the like. As each of these business information system work products are developed, reviewed, and evolved, the effect of those activities must be reflected in one or more of the data models depicted in Figure 1 and identified in Table 2.

Work Products	Data Architecture Reference Model Component					
	Data Element	Concepts Data Model	Logical Data Model	Physical Data Model	View Data Model	XML Data Model
Business Information Systems				✔	✔	✔
Business Requirements		✔	✔	✔		✔
Business Rules	✔		✔	✔	✔	✔
Database Domains			✔			
Database Objects			✔			
External Data Interface Requirements				✔	✔	✔
Eternal Quality Standards	✔	✔	✔	✔	✔	✔
Information Needs	✔		✔			
Mission Organization Functions			✔			
Resource Life Cycles			✔			
Use Cases			✔	✔	✔	✔
User Acceptance Tests				✔	✔	✔
Value Domains and Management	✔	✔	✔	✔	✔	✔
Work Breakdown Structure (WBS)	✔	✔	✔	✔	✔	✔

Table 2. Cross reference between work products and data architecture reference model component.

Each of the business information system work products are identified and described in Table 3. The last column of Table 3 enumerates the key data model artifacts that are involved with the work product. The data models that contain these artifacts are identified in Table 1.

This document does not address how to create each of these business information system work products because the creation process is more than adequately addressed within other Whitemarsh books, courses, seminars, and methodology documents. Rather, the focus of this document presumes is to assess the necessary and sufficient interaction among all the business information system work products.

Business Information System Life Cycle Work Products		
Component	**Description**	**Data Model Components**
Business Requirements	Business Requirements are the identification, specification, and definition of the components that must exist in the ultimate solution delivered by the contractor. Business Requirements form the foundation upon which all components of are engineered, implemented, and maintained. Business requirements will evolve over time. Thus, it is important to be able to track the initial and evolved requirements. It is unrealistic to initially have all requirements because new and/or revised requirements are discovered all during the project's architecture, engineering, and implementation.	Subjects, Entities Attributes Data Elements Database Domains Database Objects
Business Rules	Business Rules are assertions of truth-states in the database. Each business rule includes the identification, name, description, and exact specification of data-based rules that must either be true or that, after the execution of an information system process, results in a state of truth. There are two classes of Business Rules: data and process. Almost invariably, business rules depend on existing data, reference or control data, or data that is determined as a consequence of a process's execution. Almost all business rules are mappable to data, whether persistent or temporary. Business rules are almost always discovered during design sessions, and use-case walkthroughs. As business rules are discovered, the various data models need to be concurrently examined to determine whether the database can support the rules. Since every business rule has a process component, a key component of each rule is the specification of the process and a determination of where that rule is bound. That is, bound into the data model component (e.g., Data Element, or DBMS column), a low-level application component, a mid-level application process, or in the user presentation layer. Because of this multiplicity of possible bindings, business rules need to be centrally defined and managed, but bound only into the data model	Schema Tables Columns DBMS Schema DBMS Tables DBMS Columns Value Domains

Business Information System Life Cycle Work Products		
Component	**Description**	**Data Model Components**
	or business information system work product within which it is accomplished.	
Work Breakdown Structure (WBS)	Work Breakdown Structures are hierarchical representations of two classes of effort: What, and How. The "what" type of WBS contains an action phrase and a noun phrase that together describe what is to be done, and the name of the work product creation or evolution. The second class of WBS, the "how WBSs" are tuned to deliverables but to the actual techniques employed to accomplish the data model or business information system work product. Both the "What" WBS and the "How" WBSs need to be interlinked. Well developed metadata management systems include complete project management so that work plans and progress against work plans can be directly tied to and integrated with accomplishments.	Subjects, Entities Attributes Data Elements Database Domains Database Objects Schema Tables Columns DBMS Schema DBMS Tables DBMS Columns
External data interface Data Requirements	External data interface Requirements are the specifications of an interface between an internal database data structure and some external data source. Each interface essentially consists of a fully defined data model that defines every field in the interface to the extent that a software module can be created to read the records represented by the interface and take appropriate action against the database. Such actions can be to insert, delete, or modify database records.	DBMS Schemas DBMS Tables DBMS Columns DBMS Relationships Value Domains Views View Columns Relationships XML Schemas XML Elements XML Attributes
External Quality Standards	External Quality Standards are de jure and de facto standards through which data management products and/or processes can be judged. The ISO Standard 11179 enables assessment of the adequacy and completeness of the metadata associated with data elements. Included in this class of assessment are Concepts, Conceptual Value Domains, Data Element Concepts, Value Domains including mapping among equivalent values, Data Element Classifications, and Administrative Information (for Stewardship). The ISO/ANSI SQL standard enables the assessment of SQL data structures and languages employed in database designs and application program access.	Schemas Tables Columns Relationships Views View Columns Relationships XML Schemas XML Elements XML Attributes

Business Information System Life Cycle Work Products		
Component	**Description**	**Data Model Components**
	WC3 XML standards enable an analysis of the names and engineering of XML schemas and XML data streams so as to ensure maximum interoperability conformity to existing sets of XML schema models.	
Business Information Business Information Systems	Business Information Systems are the broad characterizations of the application systems that capture, update, and report data. Business Information Systems are additionally detailed into their specific components, and those that deal with database data are mapped to the appropriate data model component.	Views View Columns Relationships XML Schemas XML Elements XML Attributes
Use Cases	Use Cases are highly engineered pseudo-process models that clearly define the behavior of the users and business information systems, and also the responses provided from the databases as they take in, modify, or provide data to users. Use-cases are detailed to the level such that a programmer can interpret the process intent and write an application system module without semantic and/or process misunderstanding. Use Cases present behavior-based scenarios of the use of the database to accomplish the requirements. Because use-cases directly identify database data, mappings can be created between one use-case and another to identify redundancy and possible conflict. Mappings can also be made between the detailed data and process aspects of use-cases and business requirements, deployments of use-cases in software and hardware, value domains, business rules, and WBS.	Schemas Tables Columns Relationships Value Domains DBMS Schemas DBMS Tables DBMS Columns DBMS Relationships Views View Columns Relationships XML Schemas XML Elements XML Attributes
User Acceptance Tests	User Acceptance Tests are stylized user-application system interaction scripts that can be exercised to the extent that fully informed users can determine that all the requirements have been met and all use-cases are satisfactorily performed. User Acceptance Tests (UAT) are the ultimate mechanisms through which organizations determine that it has received the business information system that was specified. Because of all the different data models, and work products, the User Acceptance tests can be very comprehensive and very telling.	DBMS Schemas DBMS Tables DBMS Columns DBMS Relationships Value Domains Views View Columns Relationships XML Schemas XML Elements XML Attributes
Value Domains	Value domains relate to the allowed, disallowed, or other defined collections of values and interconnection of values that represent discrete choices (Gender = Male, Female, unknown), or sequenced	DBMS Schemas DBMS Tables DBMS Columns

Business Information System Life Cycle Work Products		
Component	**Description**	**Data Model Components**
	states such as Applied, Reviewed, Accepted, Rejected, or Appealed. Included as well are the mappings across time of evolved and/or changed value domains. Value domains commonly stand alone and are allocated to data elements, or to attributes, columns, or DBMS columns. In all cases of value domain allocation, the allocations must be such that semantics conflicts are prohibited.	DBMS Relationships Value Domains Views View Columns Relationships XML Schemas XML Elements XML Attributes
Resource Life Cycle Analyses	Resource Life Cycle of Analysis identifies, defines, and sets out the resources of the enterprises, the life cycles that represent their accomplishments, and the interrelationships among the different enterprise resource life cycles. Resource life cycle nodes represent the end-state data resulting from the execution of business information systems. The end-state data is represented through database object classes.	Schemas Tables Columns Relationships
Missions Organizations and Functions	Missions, organizations, functions, and position assignments represent the identification, definition, and interrelationships among the persons who, through their positions, perform functions within their organizations that accomplish enterprise missions. Missions define the very existence of the enterprise, and that are the ultimate goals and objectives that measure enterprise accomplishment from within different business functions and organizations. Functions represent the procedures performed by enterprise organization groups as they achieve the various missions of the enterprise from within different enterprise organizations. Organizations represent the bureaucratic units created to perform through their functions the mission of the enterprise.	Schemas Tables Columns Relationships
Information Needs	Information needs represent the identification, definition, and interrelationship of the information needed by various organizations in their functional accomplishment of missions and what databases and information systems provide this information?	Schemas Tables Columns Relationships
Database Domains	Database domains are the data-intensive bridge between mission descriptions and database object classes. While database object classes are non redundant, they may be referenced by several database domains.	Schemas Tables Columns Relationships
Database Object Classes	Database object classes represent the identification of 1) the critical data structures, 2) the processes ensure high quality and integrity data within these data structures, 3) the value-based states represented by these data structures, and 4) the database object centric information systems that	Schemas Tables Columns Relationships

Business Information System Life Cycle Work Products		
Component	**Description**	**Data Model Components**
	value and transform database objects from one state to the next. Database Objects are transformed from one valid state to another in support of fulfilling the information needs of business information systems as they operation within the business functions of organizations.	

Table 3. Data Management Components and affected Data Models.