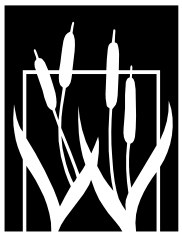


Whitemarsh
Information Systems Corporation

A Column by Any Other Name Is Not A Data Element

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

Background

This article describes an approach to achieve enterprise-wide data standardization through the specification, implementation, and maintenance of data elements within the context of a metadata-repository, CASE-like environment. Data elements, posited by this paper, are *context independent business fact semantic templates*. A full exposition of the presented in this paper is provided through the "URL" references within the last section. This paper then is merely intended to have you answer the question, "does this approach make common sense?" If your answer is yes, then consult the references for the much deeper presentations.

1. A Michael Brackett Story from DAMA 2001

Michael Brackett sat in on the my Enterprise Wide Data Standardization presentation at DAMA 2001 in Anaheim. The presentation had been a last minute substitution for Terry Moriarty's presentation on Business Rules. While I too would have liked to hear her presentation, I was "secretly grateful" that she couldn't make it. Davida Berger suggested that I substitute and I hope that I acceptably filled the "Terry-able-void."

Since the fundamental concepts of the DAMA presentation of the approach--with respect to data elements--had been "creatively acquired" from a May 1995 Michael Brackett talk that was given to the New Jersey DAMA chapter, it was sort of like having Michelangelo sit in our your one-person art show. You're both wanting his review and critique but are deathly afraid that he will give it, or even worse, just walk out half way through. I got the critique and review, and Michael stayed to the very end.

After the presentation was over, a meeting with Michael produced the observation that the approach was fundamentally sound but that he disagreed with my assertion that an enterprise only had 2000 data elements. Wow, I thought, he agrees with the approach. That's great!

2. The Approach

The approach is simply this: create a re-usable cache of data elements and then establish a metadata repository and CASE environment that enables the cache to be used to 1) define database columns, 2) interrelate the defined columns through the data element metadata, and 3) to support both forward engineering of new databases and reverse engineering of existing databases.

The approach to creating a re-usable cache of data elements begins with the adoption of a data element definition that fosters reuse. That is, *a data element is a context independent business fact semantic template*. But how does it become that? It does when the data element represents an amalgamation of familiar concepts expressed as single words brought together under a single name and then represented through a set of values. The value is not the data

element, the collection of semantics is. The value is merely a discrete value-based alternative representation of the semantics.

Critical to reusing the underlying data element concepts that form the semantics of a data element is a metadata model that exists within a repository type database. Finally, since all data elements are not just elementary atomic fact templates, both compound facts, and derived facts must be represented. Full name, and age are common examples.

Putting all this together into a single data element meta model is depicted in Figure 1. The five collections of meta entities in this meta model are:

- Semantic Hierarchies (meta category value entities)
- Data Element Domain
- Data Element
- Derived Data Element
- Compound Data Element

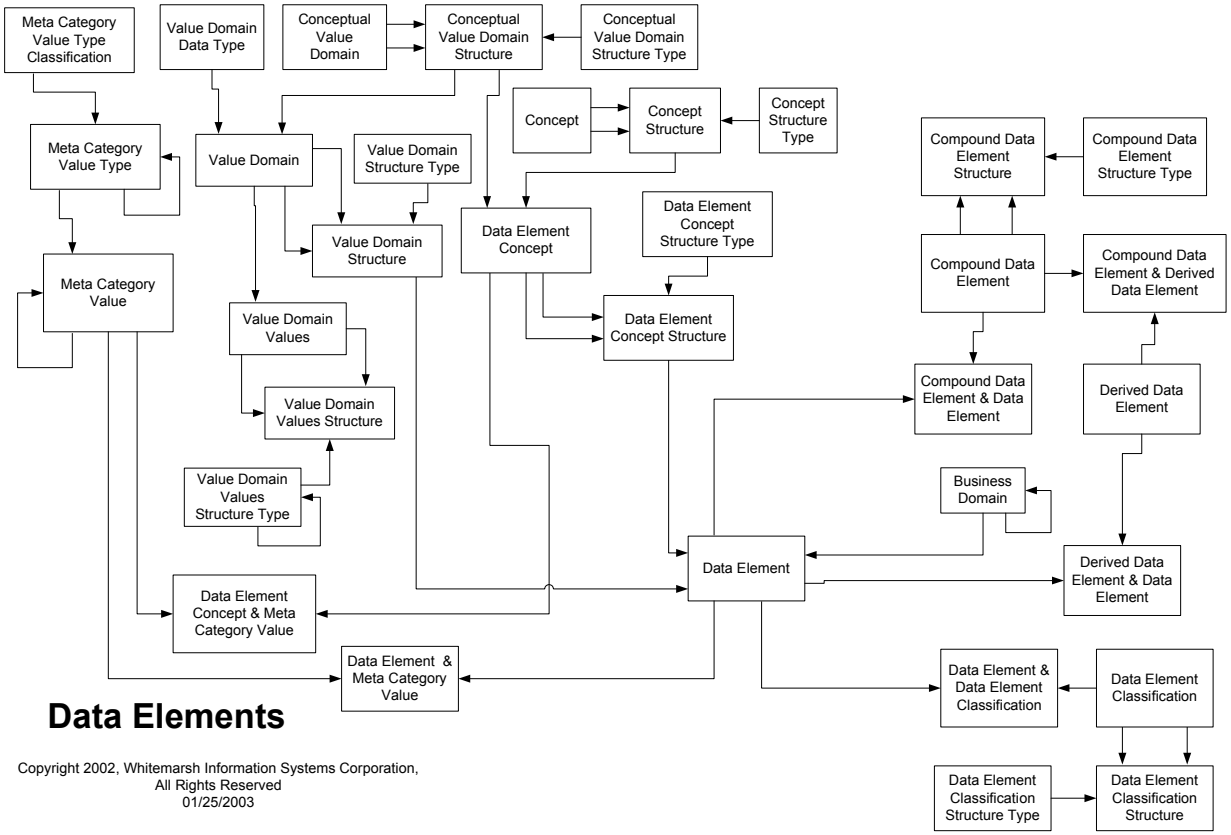
A key observation in all Michael Brackets materials is that data elements exist within natural classification hierarchies and the names of data elements are comprised of well known business domain words. This led to the set of meta entities that comprise the words that ultimately represent the semantics of the data element. The meta entities, Meta Category Value and Meta Category Value Type, are the full exposition of the relevant ontologies. The meta entity, Meta Category Value Type Classification enables the segregation of chosen word combinations to be "before" the data elements common business name or "after" its common business name.

An examination of any large inventory of columns clearly shows that there is an implicitly restricted word list that is employed to contrive the column's names. These words exist within higher categories of collections, such as accuracy, geography, and time. For accuracy, the set of words might be estimated, final, or projected. These are a sibling ontology. For geography it might be World, Western Hemisphere, North America, United States, New England, Vermont, and Burlington. These represent a hierarchical ontology. Both hierarchy types exist and are representable within the meta model of Figure 1.

Given that you now have a business based set of word hierarchies, it is an easy step to then create a full set of abbreviations for full (no abbreviation), long, medium, and short word variations. Each word should also have a definition within its context. And, while all the words that comprise the data element's name are a "short hand" for the assigned semantic concepts, the data element's full definition is the collection of definitions assigned to the collected set of data element words.

Data elements are not always simple atomic facts. Sometimes the are derived or compound business facts. Both these are represented in the meta model of Figure 1. Compound data elements are represented via a Bill of Materials meta model, and derived data elements are represented as algorithmic transformations of assigned atomic data elements. The model also supports the existence of compound data elements that are in turn derived data elements.

Data element domains represent hierarchical classification of the business semantics of values associated with a data element. For example, if there is a data element, salary, it clearly represents some form of compensation that is received by an employee within the context of the enterprise. While it is true that compensation is a type of money or currency, these are data use modifiers that identify the form of the compensation. For example, the form of compensation could be land, days off with pay, free travel, and the like. From the point of view of a data element domain, there is first Compensation and then the data element Salary. Each has its associated meta category value type and values as appropriate.



3. Benefits to this Approach

Let's review some generally common statistics about database environments for a typical state government, or multiple business line enterprise. These are provided in the table below.

Unit	Components
1 database	100 tables
1 table	15 columns
Each agency	100 databases
Each state	40 agencies

The total count of columns for each database is 1500. The total columns for an agency is 15,000 columns, and the total for the state, for example, like Washington, New York, or Maryland would be 600,000. Thus, if Michael Brackett's assertion is true, which seems correct because of his long association with the State Government of Washington, then each of the asserted 20,000 data elements is reused about 30 times. For sure many are used hundreds of times and some are used only once. Clearly then, a key benefit is that there is significantly reduced effort in specification, implementation and maintenance.

In another example of this approach, an agency within the United States Department of Defense, tasked a contractor (bordering on a sentence of eternal damnation, some would suggest) to synthesize the unique set of data elements from among the complete set of columns. The result is presented in the following table:

Data Standardization Alternative for a multi-site, multi-application Government MIS	Final Quantity of columns, fields, cells, etc.	Cost via technique employed for definition
Accomplished traditionally (prime + modifier + classword) across all systems	19,000	$6.75 million
Alternatively, if accomplished by standardizing closely named columns and fields	3,000	$1.06 million
Alternatively, if accomplished through Comprehensive Data Standardization Techniques–Eliminates redundant–but different--representations of the same concept	560	$200,000

In this particular example of determining data elements from columns, the ratio was 34 to 1. This ratio is similar to the first example. The third column in this table clearly shows the effect on cost incurred to define the original 19,000 columns. The effect is dramatic, simply dramatic.

A third example comes from another United States Department of Defense agency. A study was undertake to determine the cost of Extract-Transform-Load (ETL) efforts. Each effort is characterized by a requirement, design, software implementation and maintenance. Each such ETL represents a failure in data standardization. Columns that were supposed to be the same were represented through different names, semantics, data types, levels of granularity, time-sequencing, and the like. While an enterprise-wide data element standardization approach would not solve all these problems, it would clearly affect different names, semantics, data types. The agency spends about $175,000,000 each and every year on such ETL activities. If the data element approach resolved 50% then that would represent savings of about $90 million per year. Extended to the U.S. DoD as a whole the savings would be about $450 million, and to the U.S Government as a whole, about $1.5 Billion. Given that the U.S Government spending represents about 10% of the total economy, then the savings to the economy as a whole is about $15 Billion. These savings are not small.

A final fourth example was provided me by a reviewer of this article. An effort was mounted to bring together sets of rows from a large number of different SQL tables within a health care environment. The columns, whose actual data values were "mined" were all expected to have values of Yes or No. The table speaks for itself.

Yes|No Column Frequency	**Code values**
1296	1 = yes…..0 = no…..
899	Y = yes…..N = no…..
740	1 = yes…..…..
75	Y = yes…..…..
17	0 = yes…..1 = no…..
2	1 = yes…..0 = no…..2= error
104	Y = yes…..N = no U = unknown
1	Y = yes…..N = no…..D = does not apply
3	0 = false…..1 = true…..
5	Y = yes…..N = no…..N/A = not applicable
90	1 = yes…..2 = no…..
6	1 = yes…..2 = no…..3 = not applicable

Yes|No Column Frequency	Code values
1	1 = yes…..2 = no…..N = none
2	1 = yes…..2 = no…..0 = do not use default
88	1 = yes…..0 = no…..99 = unknown
1	1 = yes…..0 = no/negative…..99 = unknown
1	1 = yes…..0 = no…..2 = unknown
24	Y = yes…..N = no…..NA = not applicable
17	Y/N items to be deciphered…..
11	1 = yes…..0 = no…..2 = not applicable
1	1 = yes…..0 = no…..U = undetermined
1	1 = yes…..2 = no…..3 = unknown
2	Y = yes…..N = no…..blank = no
1	0 = yes…..1 = no…..2 = not applicable
1	2 = yes…..1 = no…..
1	1 = yes…..0 = no…..U = unknown
1	Y = yes N = no A = ask
1	Y = yes N = no R = restricted
1	yes = yes no = no
1	1 = yes 0 = no 2 = ask
3394	**Total Columns**

All of these stories and savings that result from an enterprise-wide approach to data elements are *just* for database applications. What are the savings for other types of access methods like spread-sheets, non-database file access such as COBOL and ISAM, and non-automated "systems" such as form, person-to-person communications? What about savings in documentation, training, systems analysis, and design? What about the probably uncountable calls to technical support to ask, "What does that field mean?" Those costs are probably incalculable by any reasonably defendable means.

4. ISO Standard for Data Element Metadata

With all the potential savings in both analysis, design, implementation efforts as a consequence of implementing the data element approach to enterprise-wide data standardization, it would not be too surprising that some sort of standard exists to govern all this. There is, and it is *ISO 11179, Information Technology - Data Management and Interchange - Metadata Registries (MdR).* The standard, known by it's shorthand, 11179.

The standard is developed and administered by Working Group 2, Metadata, of the Standing Committee 32, Data Management and Interchange. SC32 is part of the ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission). National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in such work.

In the United States, the National Body is ANSI, the American National Standards Institute. ANSI has accredited standards organizations such as IEEE and NCITS. NCITS, the National Committee for Information Technology Standards itself has a number of Technical Committees. The L8 Technical Committee for Metadata participates in the development of the 11179 standard. Another committee, for example, H2, is the Technical Committee for Database participates in the development of database language standards such as SQL.

The FCD (Final Committee Draft) of the 11179 standard that is under development by L8 and WG2. This document represents a revision of the existing 11179 standard, that while important, did not sufficiently address the required metadata models for data element metadata. The following material is taken directly from the draft document. From the Introductory section of the standard:

> Data processing and electronic data interchange rely heavily on accurate, reliable, controllable and verifiable data recorded in databases. A prerequisite for correct and proper use and interpretation of data is that both users and owners of data have a common understanding of the meaning and representation of the data. To facilitate an understandable shared view, a number of characteristics, or attributes, of the data have to be defined.
>
> This ... International Standard specifies the structure of a metadata registry, which is a place to keep facts about characteristics of data that are necessary to clearly describe, record, analyse, classify and administer data. The conceptual structure of a metadata registry is specified in the form of a conceptual data model. This Part also describes the basic attributes of data elements for purposes where a complete registry is not appropriate.

> This ...standard is of interest to information developers, information managers, data administrators, standards developers and others who are responsible for making data understandable and shareable. It is also of interest to manufacturers of metadata registry and CASE tool products.

From Section 4.1, Meta model for the content of a metadata registry:

> Data elements, data element concepts, conceptual domains and value domains are specific types of administered items. Each of these types of administered items is represented within a metadata registry by administration records that document the common administration and identification, naming and definition details together with their administered item-specific details.
>
> A meta model is a data model that is about data. This standard is specified as a conceptual data model. A conceptual data model describes how relevant information is structured in the natural world. In other words, it is how the human mind is accustomed to thinking of the information. As a conceptual data model, there is no one-to-one match between attributes and fields, columns, objects, et cetera in a database. There may be more than one field per attribute and some entities and relationships may be implemented as fields. There is no intent that an implementation should have a table for each relationship or entity. The meta model need not be physically implemented as specified, but it shall be possible to unambiguously map between the implementation and the meta model in both directions.
>
> The structure described by this meta model may be distributed over several implementations. These implementations may be databases, repositories, registries, dictionaries, etc. The importance of this meta model is the exposure for understanding, sharing, and reusing of the contents of implementations.

From Section 4.2, Application

> A meta model is necessary for coordination of data representation between persons and/or systems that store, manipulate and exchange data. The meta model enables systems tools and information registries to store, manipulate and exchange the metadata for data attribution, classification, definition, naming, identification, and registration. In this manner, consistency of data content is assured among systems tools and information registries.

> Using the meta model, mappings to the schema of each tool set can be developed. The meta model constructs can be translated into the language of each tool set, preserving the concepts represented in the original model.
>
> It is assumed that an implementor will use this conceptual data model to develop a more specific logical data model of the identical sphere of interest. A logical data model describes the same data, but as structured in an information system. It is often referred to as a Model of the Information System. A logical data model can be directly used for database design.

From Section 4.3, Extensibility:

> It is not expected that this meta model will completely accommodate all users. For example, scientific data requires metadata attributes not addressed in this standard. Such extensions shall be considered conformant if they do not violate any of the rules inherent in the structure and content as specified by the meta model in this standard. Entities, relationships, and attributes may be added to this conceptual data model.

The standard's meta model consists of the following five regions:

- Administration and identification Region
- Classification Region
- Data element Concept Region
- Conceptual and value domain Region
- Data element Region

Administration and Identification Region

The Administration and Identification region supports the administrative aspects of administered items in a registry. The meta model for this region is presented in Figure 2. This region addresses:

- The identification and registration of items submitted to the registry

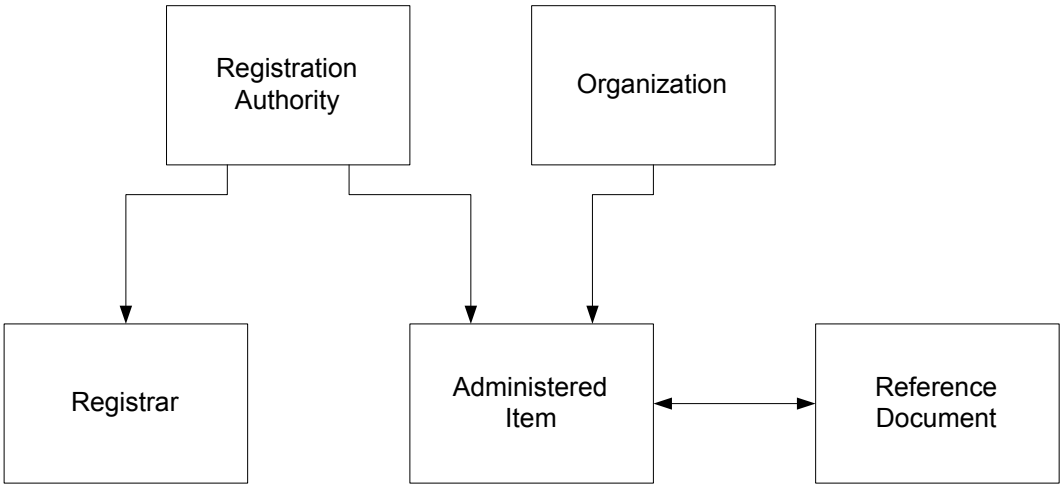


Figure 2.11179 Meta Model Administration and Identification Region

- The organizations that have submitted items to the registry, and/or that are responsible for items within the registry, including Registration Authorities
- Contact information for organizations
- Supporting documentation

Classification Region

The Classification region is used to manage *classification schemes* and the classification scheme items that are in the classification schemes. The meta model for this region is presented in Figure 3. *Administered items* may require classification under one or more schemes. The *classification scheme* may be a taxonomy, a network, an ontology, or any other system for systematizing where the categories are mutually exclusive. The classification may also be just a list of controlled vocabulary of property words (or terms). The list might be taken from the "leaf level" of taxonomy.

The classification scheme allows for a full bill of materials structure supporting both network and hierarchical classification schemes in support of an administered item. Administered items relate to one or more data elements.

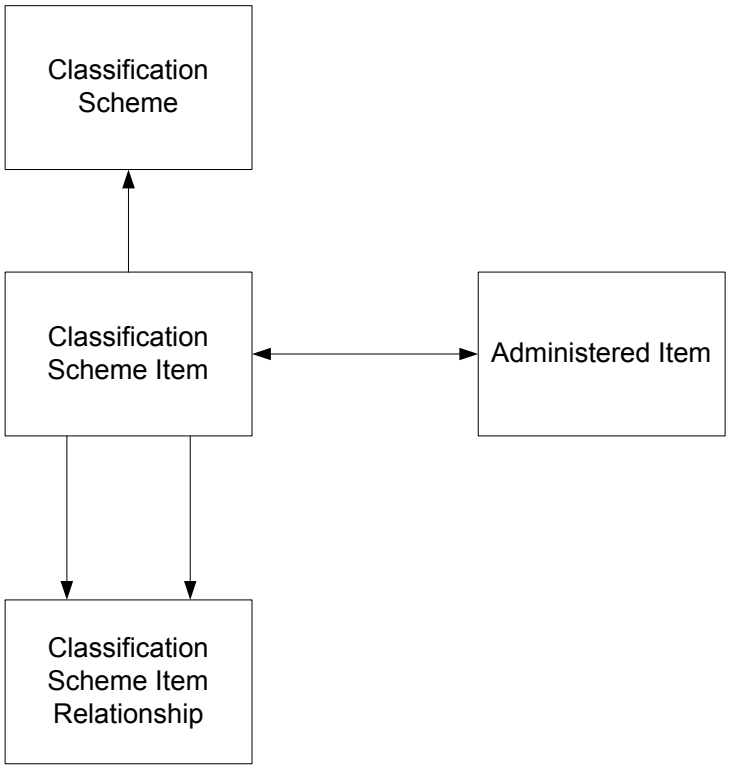


Figure 3. 11179 Meta Model Classification Scheme Region

Data Element Concept Region

The data element concept region maintains the information on the concepts upon which the data elements are developed. The meta model for this region is presented in Figure 4. The components of this region concentrate on semantics. The concepts are independent of any internal or external physical representation. The major components of this region are concepts and properties, which are combined to form data element concepts.

Data element concepts also allows for a full bill of materials structure supporting both network and hierarchical data element concepts in support of data elements.

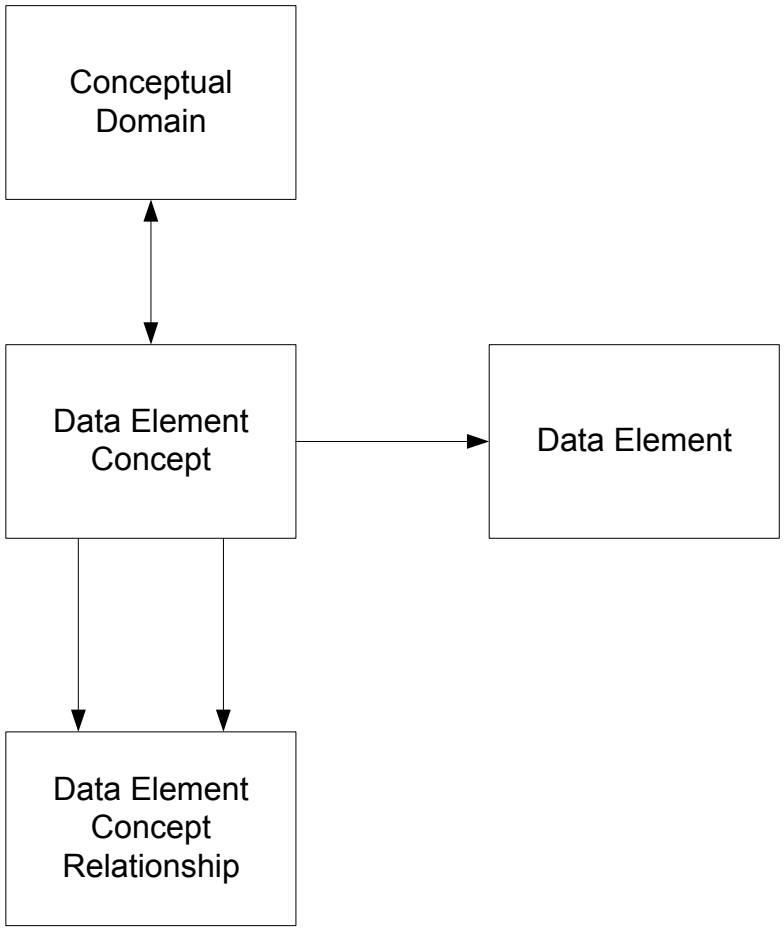


Figure 4. 11179 Meta Model Data Element Concept Region

Conceptual Domain and Value Domain Region

The meta model for conceptual domains and value is presented in Figure 5. These domains can be viewed as logical code sets and physical code sets. Conceptual domains support data element concepts and value domains support data elements.

There are two bills of material data structures in Figure 5. The first supports the conceptual domains that are tied to data element concepts, and the second are the value domains that are tied to conceptual domains.

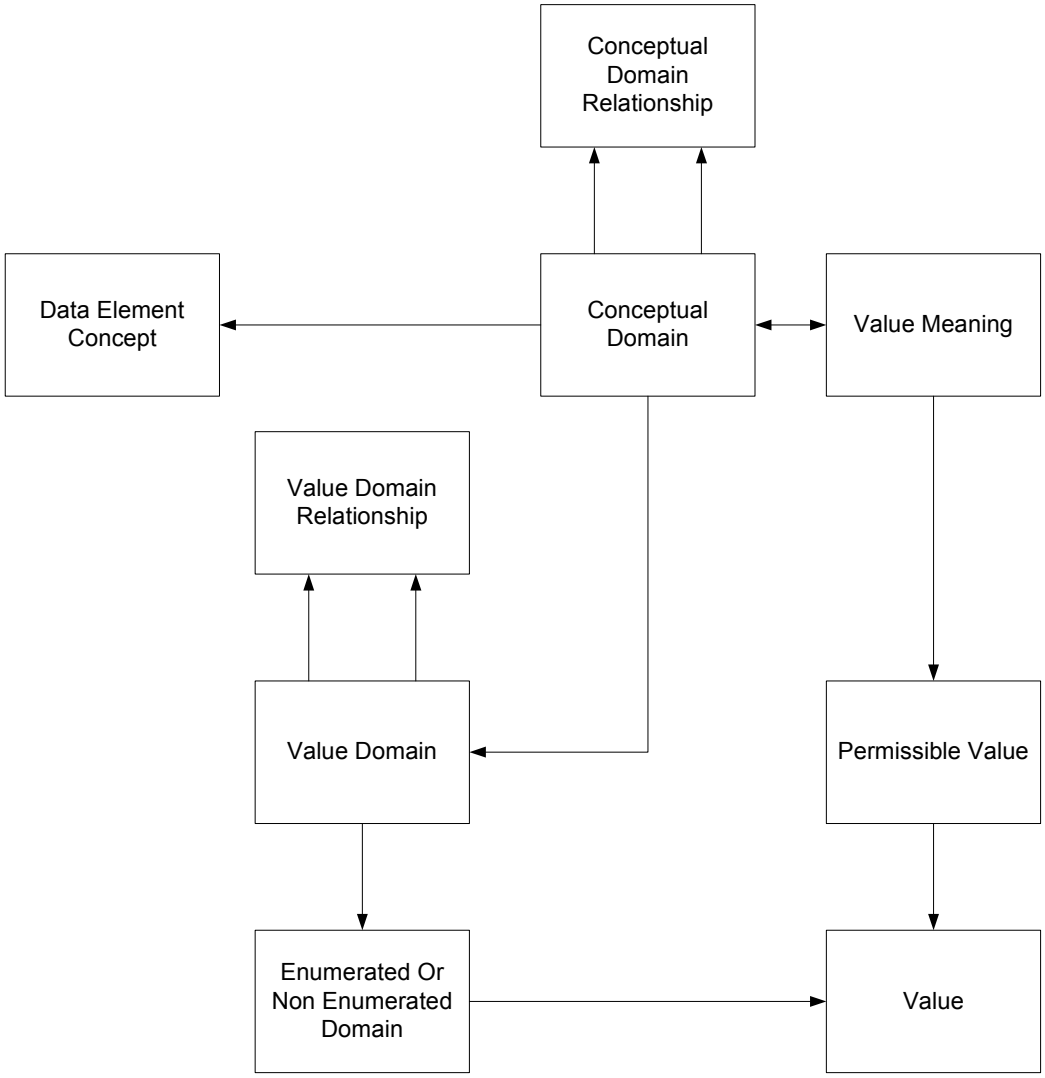


Figure 5. 11179 Conceptual Domain and Value Domain Region

Data Element Region

The Data element meta model is presented in Figure 6. *Data elements* provide the formal representations for some information (such as a fact, a proposition, an observation, etc.) about some concrete or abstract thing. *Data elements* are reusable and shareable representations of *data element concepts*.

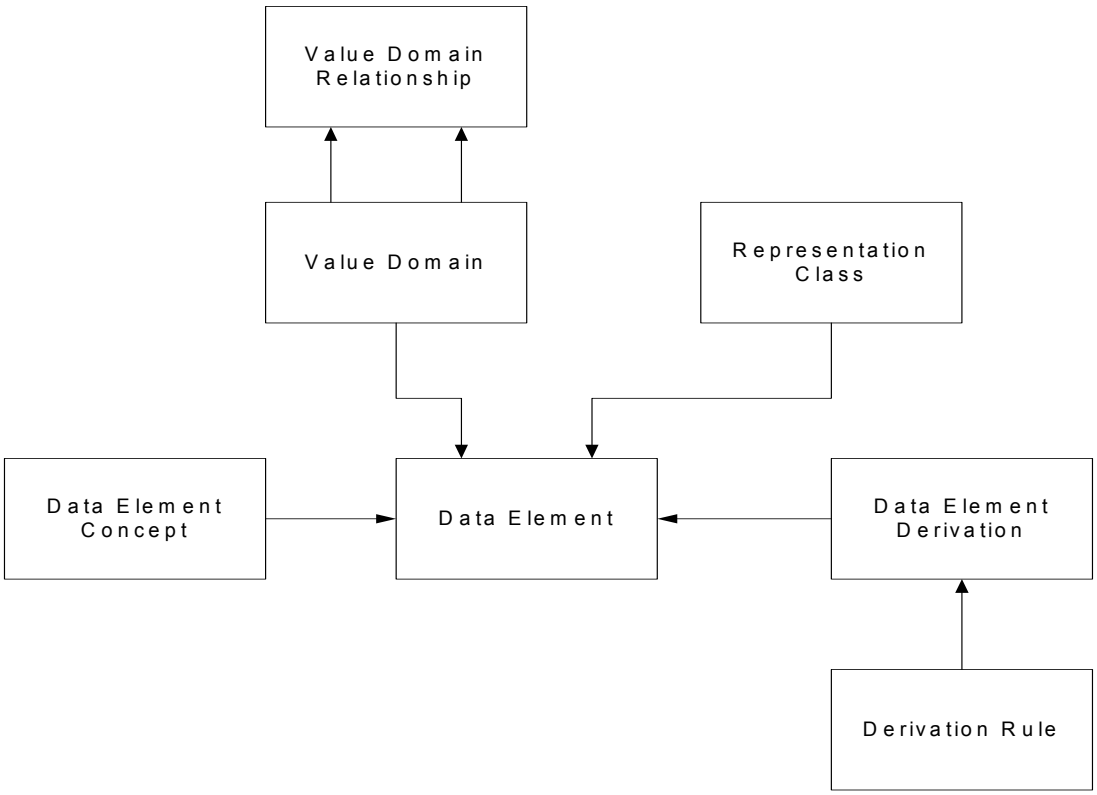


Figure 6. 11179 Data Element Region

5. Achieving the Benefits in the Database Environment

A necessary first step in implementing the conceptual meta models of 11179 is its transformation into relational tables that can be implemented through a DBMS. The transformation process is well known to data modelers and is not presented. An acceptable transformation of the 11179 conceptual entities to relational tables data element meta model is presented in Figure 7. The key tables of that transformation are depicted. There may be additional tables to make it more conforming to 11179 (for example, the registration tables are not shown) and different transformations just so long as its contained data element metadata can be materialized for loading into a 11179 compliant data element registry or that be can be shared with other 11179 data element registries.

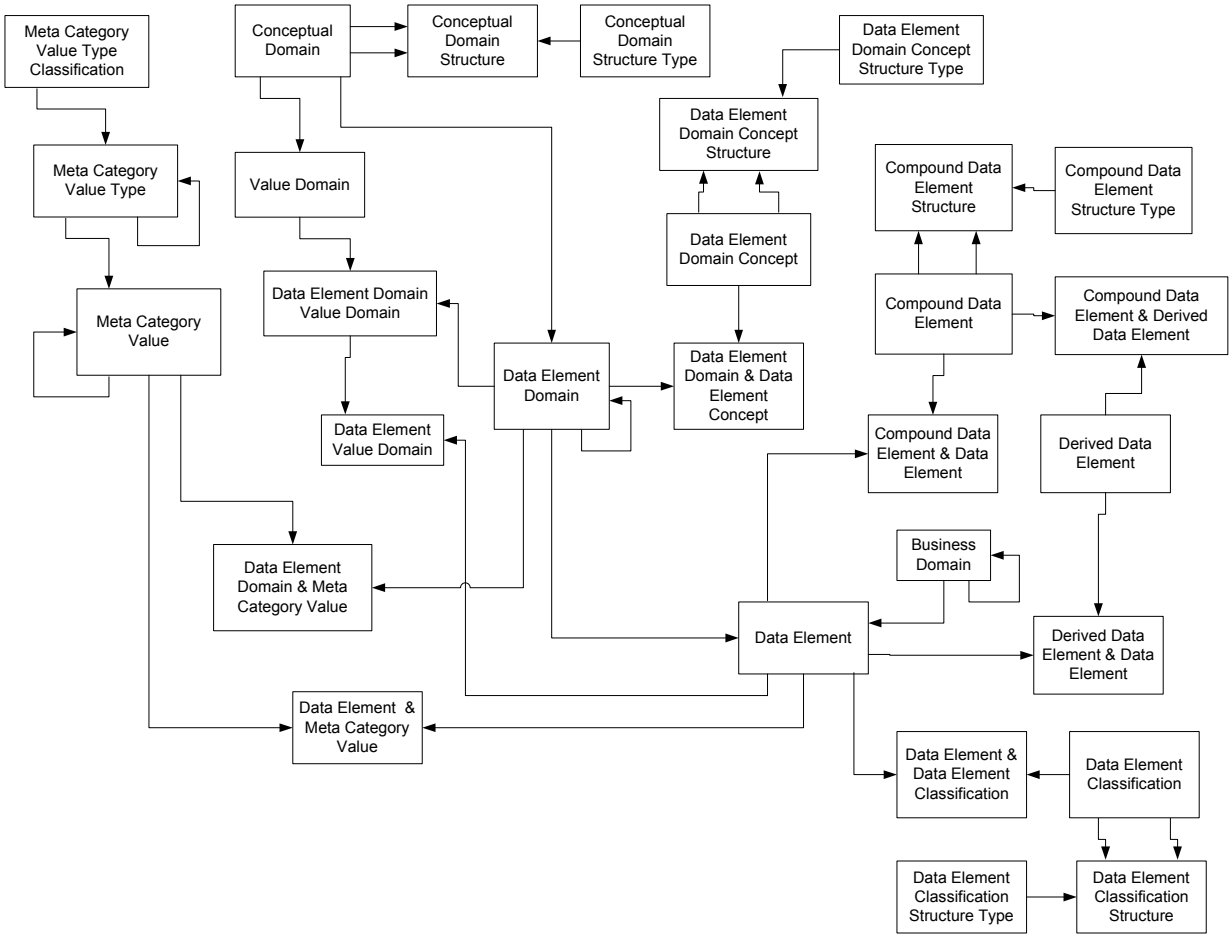


Figure 7. Transformed "relational" data model for ISO 11179.

It is not critical nor practical that an enterprise have one metadata repository. After all, if the ideal is reached, that is all the metadata is defined once, stored once and then used throughout the enterprise, and if that metadata repository suffers a catastrophic disaster then conceivably all IS development and maintenance would cease. While some might argue that humanity would be better off, it is not appropriate to either advance or refute such arguments.

If the enterprise supports a distributed metadata repository environment, then sophisticated distributed DBMS synchronization concepts would have to be employed. Shared for certain would be the certain of the meta models that are identified below, such as semantic hierarchies and data elements, and specified data models at least at the subject level.

Once implemented, if data element metadata resides solely within the confines of a metadata repository as opposed to being employed within a complete CASE (both lower and upper) environment, the benefits of the data element metadata will be greatly limited. In short, the more "active" the data element metadata is, the greater its positive effect on the goal of achieving enterprise-wide data standardization. Figure 8 presents an integrated environment in which data element metadata is integrated within a CASE-Repository environment. Within this environment there are the following submodels:

- Semantic Hierarchies
- Data Elements
- Specified Data Model
- Implemented Data Model
- Operational Data Model
- View Data Model

These six meta models and their inferred data represent the executed policy of the IT organization of the enterprise with respect to data metadata. If key data metadata is not defined, collected or is not able to be properly employed in the efficient and effective development, use, and maintenance of databases and information systems then blame should squarely lay at the feet of data administration, database administration, and subject matter functional experts.

There are many other meta models besides these six that comprise full IT systems development. Collectively these six and the other meta models should form the complete metadata repository that is at the heart of a completely integrated CASE environment. If all these meta models are fully defined and integrated into the critical path of information technology then productivity and quality will increase, and costs and risk will decrease to such an extent that the overall cost of building such environments will be negative.

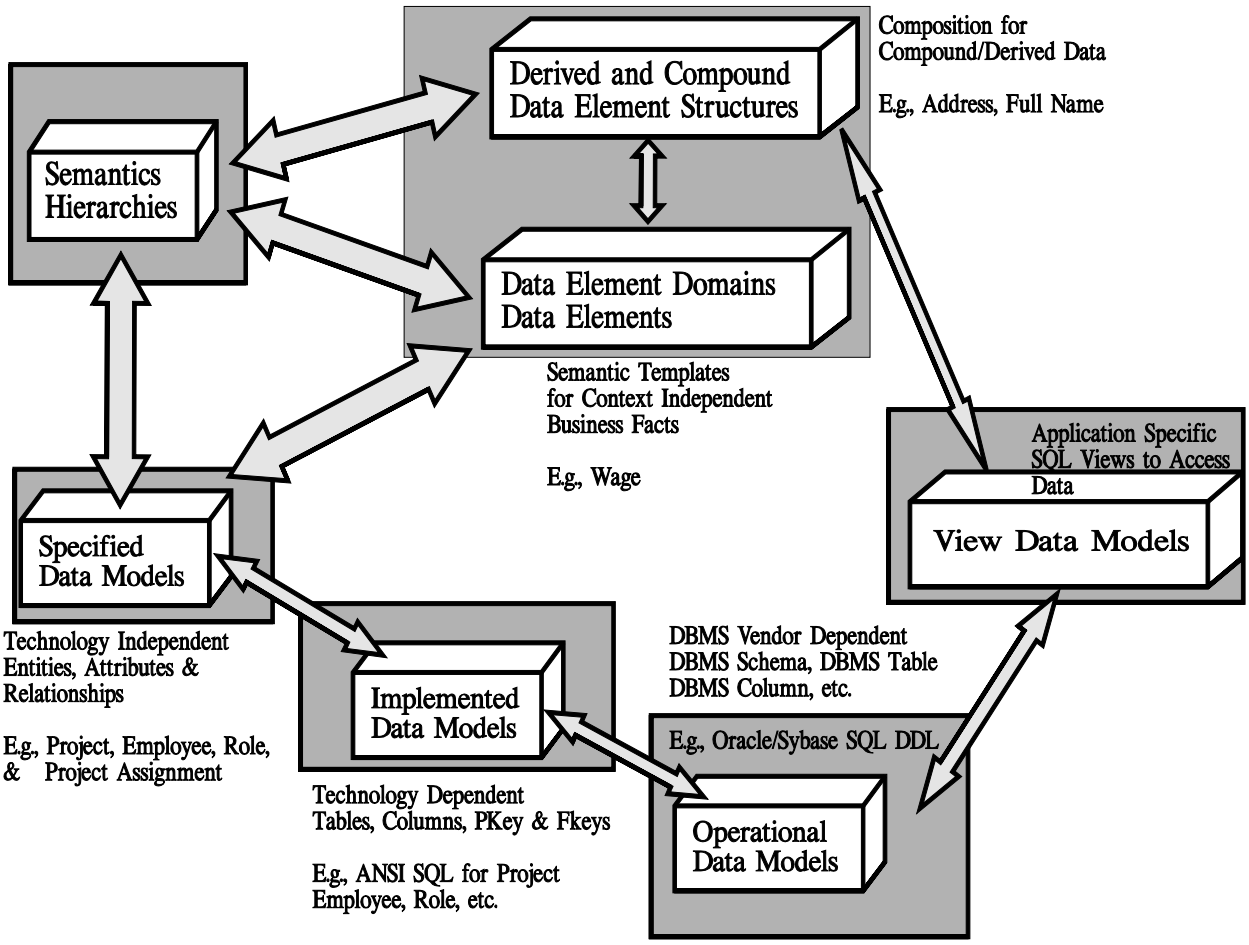


Figure 8. Integrated Metadata Model for Integrated Data Models.

Semantic Hierarchies and Data Element Model

The semantic hierarchy and data element meta model database should be a data administration project. That is, it should be their responsibility to determine requirements, build the database design, specify and possibly implement the data element metadata repository subsystem, to develop training, documentation, and to assist users in its proper use as they work to achieve enterprise-wide data standardization. The other meta models that are described below should be a joint responsibility of data administration, database administration, and subject matter functional experts.

The semantic hierarchy and data element models are essential first models to be built and implemented because flowing out of these models is the ability to achieve enterprise-wide data standardization, and the ability to "see" where data sharing can occur.

Specified Data Models

The specified data model represents a first step in achieving data standardization in support of enterprise-wide database. That is, the creation of data model templates. These represent commonly used collections of entities, attributes, and relationships that are organized by subject areas.

Each entity is a coherent policy-based collection of attributes that are in at least third normal form so as to guarantee policy based homogeneity. Data elements map to the attributes across and within these specified data model templates.

Semantic hierarchies from the data element meta model in Figure 8 are mapped to attributes that more precisely define the semantics within the attributes of their containing entities. The semantics of the attributes must, of course, represent a subset of the semantics previously assigned to the data elements.

Specified data models are not database designs. Its paradigm is subject, entity, attribute. Relationships connect various entities, and can related entities across subject areas. These models are best accomplished by functional data administrators or experts. These specified data models are then available to database designers as they employ these data model templates to create implemented databases.

The specified data model attributes are quickly created through the identification of one or more data elements that are to be the semantic templates for attributes within entities. The inherited semantics from the hierarchies of meta category value types, meta category values, data element domains, data elements, and entity subject areas hierarchies all contribute critical components to the semantics of an attribute. Attribute definitions can be completely automatic through the definition fragments that are immediately available from the attribute's inherited semantics.

In the case where a data element is associated with more than one attribute in an entity as would be the case for home-, office_, cell_, and fax_telephone_number, the ability to create a

local name and even a local definition is present. Available, of course are all the already inherited semantics.

Because of these work saving assists, creating a specified data model is an effort that is characterized by:

- Significantly shorter times to create entities because of all the inherited semantics
- Lowered risk because when things are the same they will be semantically defined the same
- Increased quality because there will be more time for important semantic component parts
- Increased productivity because the process of creating the specified data model can be accomplished by functional experts rather than just data administration staff.

Implemented Data Models

Database data models exist in two necessary forms: DBMS independent and DBMS dependent. Both these models are needed because databases that are actually deployed may have table designs changed to meet the needs of different DBMSs, operating systems, data architecture classes, or computer hardware capacities. In such cases, the "real" database design is not intrinsically changed, just deployed differently. Consequently, the DBMS independent database designs are needed.

Independent database data models are thus database data model templates for operational database data models deployed across the enterprise. The triple of the implemented data model are schema, table, and column. Data elements are mapped to columns. Attributes are mapped to the columns. Semantic hierarchies mapped to columns give a more precise layer of semantic subsetting of the attributes and in turn of the data elements that are mapped to the columns. A database is thus a collection of tables of columns that map to attributes of entities. Attributes from more than one entity may map to columns of a single table.

A table then is not necessarily a homogeneous set of attributes within a policy based entity as would be the case in a well designed specified data model entity, but may be artificially contrived to meet the needs of a particular data architecture class such as original data collection, Transaction Data Staging Area (TDSA), subject area database, data warehouses (wholesale or retail), or reference tables.

The implemented data model is distinct from the specified data model. It may merely be the technology dependent transformation of a collection of entities from within the specified data model whereby subject areas, entities, and attributes serve as data structure templates for tables

and columns within an implemented database. The implemented data model may also represent a stand alone database design that has not been mapped to the specified data model. That of course is not recommended. The implemented data model may have a multiple subject-area scope and thus may represent interrelated collections of entities, attributes and relationships from the different subject areas.

Another key difference is that tables within the implemented data model are able to belong to database objects, while, in the specified data model, entities are merely standard data structures that are able to be deployed as tables (possibly within database object classes) of a particular database.

A key value of the implemented data model is to convey a COTS (commercial off the shelf) package vendor's data model in a form that is understood by the rest of the enterprise. If TDSA data architecture databases are built with push-data from the COTS package, then downstream pull-database applications, such as subject area databases and data warehouse databases, can be built.

The implemented database's main purpose and is to represent a database that is to be implemented on some technology dependent platform. That is, through one or more DBMSs and one or more specific computers. If a human resources schema is implemented under a SQL DBMS on three different platforms, for example, MVS, Unix, and Windows/NT, then while there would be only one Schema (and tables and columns), there would be three different sets of DBMS Schemas, DBMS Tables, and DBMS Columns.

The greatest benefit from this approach to the development of the implemented data model is the ability to employ predefined semantics from the specified data model. That is, from subject areas, entities, attributes, data elements, meta category value types, and meta category values. After a period of time, this approach should enable enterprises to develop entirely new schemas, tables, columns, etc., with a minimum of original effort. The data architecture classes that benefit most are subject area databases, wholesale and retail (data mart) databases, and TDSA databases. Original data capture and reference data databases benefit the least from this approach as these databases are the source of the majority of data structures and semantics for the enterprise.

Operational Data Models

DBMS data models for the operational set of schemas that exist within specific hardware, operating systems, and DBMSs. The triple here is DBMS schema, DBMS table, and DBMS column. DBMSs columns are mapped to columns from the implemented data model tables. Because of the divergence of DBMS vendor implementations of SQL standards because of different performance characteristics of operation systems and hardware, there may be different operational data model variations of the same implemented data model.

As stated at the outset of this section, the operational data model is not only DBMS specific, it is also targeted to a specific operating environment. If the only set of data models that an enterprise retains are those that reflect the operating DBMSs, then there will be no context independent business semantics from which the business can be understood and changed.

Thus, there can be multiple operational data models, each with a somewhat different design due to DBMS characteristics and performance requirements for every database. Additionally the data models contained in the data structures within the specified data models could appear in multiple implemented and operational data models.

Because most enterprises do not define, interrelate and track the different deployments of the same set of business semantics they do not really have any control over the most valuable resource, information.

Operational data models are created through three avenues: forward engineering, original creation (due to capabilities that may be missing in certain DBMSs) and reverse engineering. Most often, the enterprises will employ a combination of top-down and bottom-up techniques.

Bottom up techniques proceed via reverse engineering to create the realistic portrayals necessary to give implemented databases a real rather than a theoretic existence. Once the top-down and bottom-up are joined, enterprise database begins to emerge.

What may seem to be an inordinate amount of work in order to achieve enterprise database is really not. The model building strategy described in this approach ensures that there is maximum reuse of already defined business based.

View Data Models

Application view data models, that is, view models that interface the operational data models to the database applications consist of views and their view columns with the associated hierarchies of joins and selects to give applications a flat “record set” for processing. Included in the views are rename clauses and on-the-fly calculations. View elements map to DBMS columns and as appropriate compound data elements and derived data elements.

Approach Summary

Data elements mapped through attributes of the specified data model and/or through columns of the implemented data model enable fact based semantic homonyms regardless of name changes. Semantic hierarchies that are attached to data elements, which in turn are mapped to attributes or columns enable the identification of semantic homogeneous or related context dependent business facts that exist ultimately as DBMS columns.

Data elements form the semantic templates that can be employed to control the semantics of attributes within specified data model entities. This saves great time in specifying attributes and also enables rapid where-used reports across entities of different subject areas.

Specified data model templates, in turn, serve as templates for complete or partial tables within the design of databases. This too saves resources in specifying database tables, and because the data elements are, by inheritance, mapped to the database table columns, where-used reports across tables from different schemas can be produced. And, because implemented data models are mapped to specified data models through the mapping of columns to attributes, cross reference subject, entity, and attribute reports can be produced.

Implemented data model collections of schemas, tables, and columns can finally be employed to deploy different operational data models. Different operational data models may be needed to accommodate different DBMS-based database designs that had to be created as a consequence of difference in hardware configurations, operating/system differences, and in the different types of data models that are able to be created by any given DBMS.

To complete the process, DBMS data model columns are mapped to application view columns to thus enable the full complement of IT staff to understand which business information systems are collecting, reporting, and updating enterprise business facts.

The complete integration of these models, that is, semantic hierarchies, data elements, specified data models, implemented data models, operational data models and application view models finally gives enterprises the metadata through which integrated, shareable, enterprise-wide data standardization can be achieved.

7. Summary and Conclusions

The strategy for researching, identifying, defining, and deploying data elements within an enterprise can be highly efficient. It is fundamentally based on the premise that enterprises have a finite set of business facts that are used over and over in different contexts and computing environments. That this assertion is true is supported by the examples in this paper.

While the sensibility of this approach is intuitive, and widely recognized, it is almost never employed. That is because the approach for data element metadata development has been isolated into stand-alone repositories and not integrated within a CASE environment through which the benefits of "define-once use many times" can be realized. Without wide deployment and without CASE integration, isolated data element repositories have almost always fallen out of favor, and in times of financial distress, have been difficult to support and are often discontinued. Given that prior proponents of such projects are then identified as having wasted scarce corporate resources, future proponents of such projects are seen as being foolhardy at the very minimum.

Make no mistake, however, the benefits have always been present, but seldom realized due to poor implementation and integration strategies. Thus, properly designed and implemented meta data, CASE-based, repository projects have always been justifiable. The implementation

costs from such projects, if implemented through pre-existing designs, CASE and code generators are commonly returned on the first use project.

A key determining characteristic as to whether your organization has implemented the true spirit of 11179 is **not** the existence of the core 11179 meta model that's depicted in Figure 1, or a relationally transformed model as depicted in Figure 7, but the existence of a use environment that parallels the other data meta models such as Specified, Implemented, Operational, and Application View. Without these other data meta models, the title of this article*, A Data Element by Another Name Is Not a Column* will not be true as the inventory of data elements will essentially be the same as the inventory of columns. You will not have achieved the 20-30 to 1 ratio between columns and data elements. Rather, if you have 50,000 columns you will probably have about 30,000 data elements. The reason for the difference in quantities is that you are 20,000 "data element" definitions behind in your work. By the time that recognition occurs, you will probably then realize that for every database of 1500 column that comes into existence you now have an additional set of 1500 data elements to define. Not a "pretty picture." In short, the faster you go, the "behinder you get."