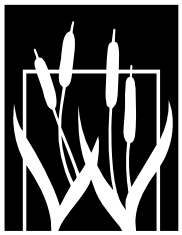

Whitemarsh
Information Systems Corporation

Business Event 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 1

2. Topics Covered 1

3. Rationale for Business Event Management 1

4. Business Event Data Structures 5

5. Solution Approach 8
   5.1 All Possible Business Event Context Data 10
   5.2 Actual Business Event Context Data 11

5. Whitemarsh Methodology Support 13

6. Metabase System Support 15

7. Conclusions 17

1. Objective

The objective of this paper is to present the Whitemarsh approach to deal with business event sequences of two forms: possible and actual. Such business event sequences are critical for database applications that must maintain detailed transaction histories rendering and event auditing. Example applications include "chain of custody," finance, workflow, ordering, manufacturing, and human-activity tracking.

This paper distinguishes between two classes of data: Content and context. Content data refers to the proper columns of data such as a person's social security number, birth date, proper name, and the like. Context data refers to the six interrogatives that describe the who, what, when, where, why, and how of the content data's capture, storage, interrelationships and maintenance. Context data is critical to the proper recording of business events and interrelationships between and among business events.

2. Topics Covered

The topics is this paper include:

- Rationale for business event management
- Business event data structures
- Solution approach
- Whitemarsh methodology support
- Whitemarsh metabase system support

3. Rationale for Business Event Management

Often, we see database schemas as large collections of database tables without recognizing the discrete database table patterns for employees, companies, orders, invoices, and customers. Figure 1 illustrates just such a database. This entity-relationship style data warehouse data model appears quite traditional and addresses customers, sales organizations, [product] component items, customer locations, orders and shipments, and sales statistics. The majority of data for this database comes from original data architecture-based databases and business information systems. The data for the database depicted in Figure 1 is extracted from the operational system databases, likely recast, and stored in this data warehouse.

Each well-designed database table represents an encapsulated data-based policy instance such as an address, an employee's biographic information, an invoice header or detail, or a customer's key contact information. Each such table is characterized by a name, a set of

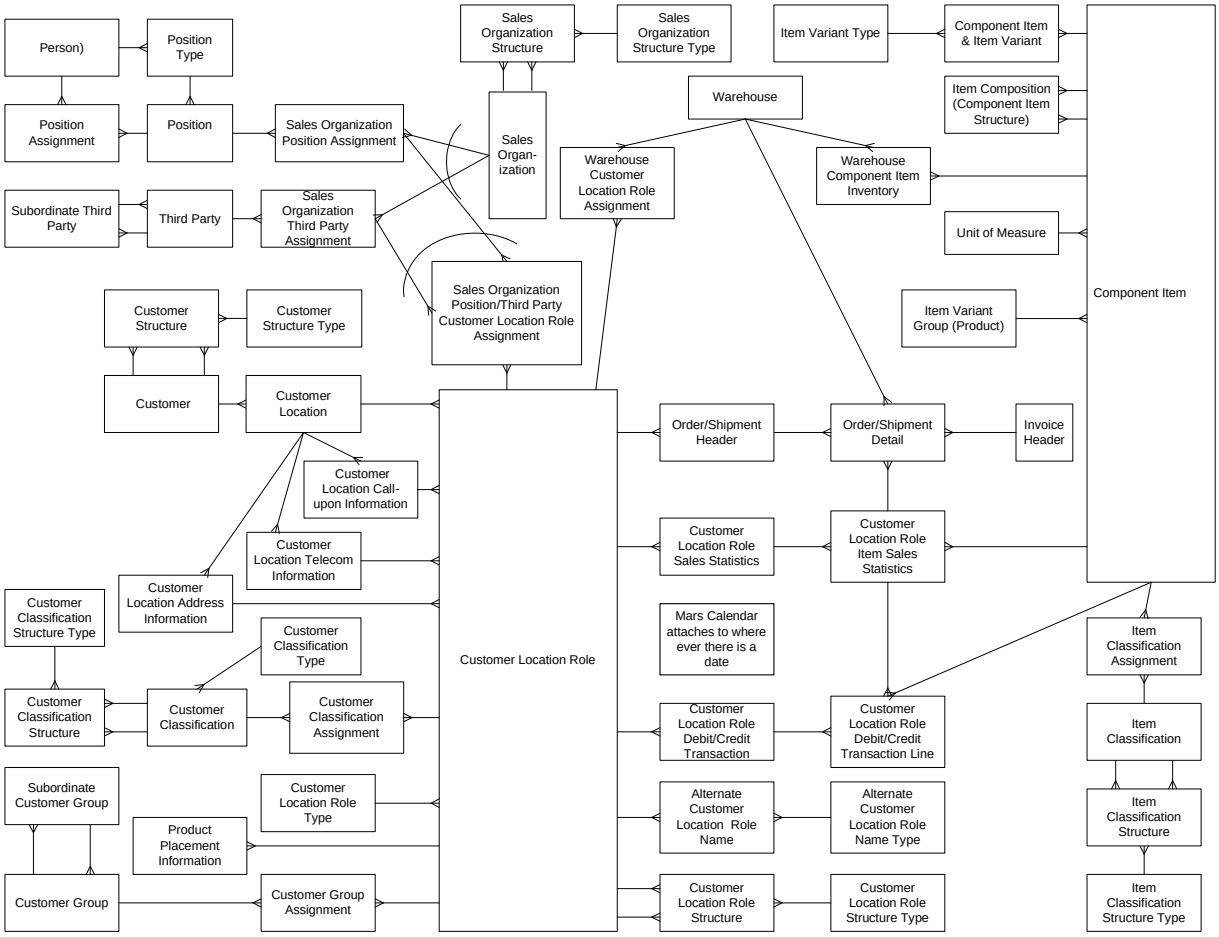

Figure 1. Sales Data Warehouse.

columns, and if well designed, a primary key that is based on column-based business facts that, when valued, selects one and only one row.

A database table is, however, often part of a larger and more comprehensive set of business-based data known as a database object class. A database object class almost always is comprised of multiple functionally related database tables for something like Employees, Companies, Orders, Invoices, and Customers. Within Customer, for example, there would be a customer's basic information, customer contacts, customer addresses, customer locations, and the like. Each of these represents a table within the context of the Customer. Database objects, regardless of persistence and regardless of whether single or multi-tables contain the same four-part composition:

- Data Structure: Database object class data structures are the set of data structures that map onto the different value sets from multiple database tables for real world database.

- Database Object Process: Database object class processes are the set of computer software based processes that enforce the integrity of values from simple or complex columns, references between database objects and actions among contained data structure segments, and the proper computer-based rules governing data structure segment insertion, modification, and deletion.

- Database Object Information System: Database object class information systems are the collections of specifications that control, sequence, and iterate the execution of various database object processes that, in turn, cause changes in database object states to achieve specific value-based states in conformance to the requirements of business policies. For example, the reception and database posting of data from business information system activities (screens, data edits, storage, interim reports, etc.).

- Database Object State: A database object class state represents the database object after-state of the successful accomplishment of one or more recognizable business events. Database object state changes are initiated through named business events that are contained in business functions.

A set of data from a database object class that represents a policy-based well-defined instance is called a database object. Examples of database objects for Employee might be Employee Requisition, Employee Candidate, Employee Interviewed, Employee Hired, Employee Assigned, Employee Reviewed, Employee Promoted, and Employee Separated. Each named database object consists of a subset of rows of valued columns across the database object's tables that match an engineered set of business rules.

Database object classes almost always have a root database table for a database object class and a set of subordinate database tables related to the root table through primary and foreign keys. Database object classes can have multiple levels to their hierarchies. From Figure 1, Persons is a database object class. Position Type and Position is also a database object class. So too are Sales Organization, Customers, Invoices, and [product] Component Items.

From Figure 1, some database object class data structures are networks. Included in these network structures are customers, sales organizations, customer groupings, customer classifications, component items, and item classifications.

Database object classes are often derived through use-case analyses. Use-cases are seen as collections of related business functions that employ discrete, recognized business processes that are focused on employees, companies, orders, invoices, and customers. Use cases often end up as software module collections that collect, store, update one or more database objects and report business information from these database objects.

Database objects are not completely isolated one from another or from other enterprise artifacts. Rather they are collected into subject related groups and are allocated to the essential resources of the enterprise. From Figure 1, these enterprise resources include Persons, Customers, Component Items, and Sales Organizations. Other examples of enterprise resources might be assets, facilities, finance, and reputation. Each of these resources proceed through well-defined resource life cycle states. Each resource life cycle state is evidenced through a collection of database objects from one or more database object classes. Each resource life cycle state is achieved through the execution of one or more business information systems.

Clearly, a parallel structure between database objects and identifiable business information systems functions exists. As business functions, via their business information systems, are executed, database objects are created, stored and updated. As the database objects are created, stored, and modified, the states of the enterprise resources (persons, facilities, customers, etc.) change.

In spite of the existence of the database object classes, database objects, resources, and their resource life cycle states, the questions that cannot be answered are:

- When did the represented underlying database objects occur?
- In what sequence did the underlying database objects occur?
- What were the total set of all possible database objects?
- What, from all possible database object classes, were the specific database objects that did occur?
- Who accomplished the various database objects and in what sequence?

For longitudinal, auditing, or historical research, it is necessary to address these questions.

Missing from this two-part (business functions (via business information systems) and database object classes) paradigm are business events. Here, a business event is the formal identification of the execution of the business information system software that captures or updates database objects. Business functions are fundamentally human-based processes. Business events are the instigators of database object class actions that effect discrete, identifiable, recognizable, trackable, inter-relatable, time-sequenced, and reportable database records. For example, the creation of Employee Requisitions are the business functions. The creation of a specific Employee Requisition within the database is not only the instantiation of one or more database objects through the business function supported business information systems, it must also be the instantiation of the “who, what, when, where, why, and how” information related to the employee requisition business event itself.

Each business event execution represents a business event transaction. Once developed into executing software, business event transaction content data is captured through the business information system software layer and is stored in the database as database objects.

The business event transaction’s data is the "who, what, when, where, why, and how" about each time-sequenced, business event-driven execution. This business event transaction data is created with each business event execution.

Database designs, however, seldom support the capture of data about 1) the business event transactions themselves, 2) the history of business event transactions, or 3) the interrelationships among business event transactions. Rather, the database objects are often merely stored in the database without all the "who, what, when, where, why and how" context data.

Business events exit in two forms: possible and actual. The possible business events are those that are anticipated to occur. These are identified through the business information system methodology tasks of requirements, analysis, and design. In contrast, actual business events are those that actually occur. These exist as a consequence of end-user behavior.

Seldom defined and stored as "real" database data are the business events that are "possible" to occur versus which business events "actually" occur. Even more rarely stored are the interrelationships between possible and actual business events.

If the context data is stored only at the database table level, lost will be the context data for whole database objects. If context data is stored only at the actual business event level, lost will be the ability to compare and contrast actual verus possible business events that would support the development of patterns that might be critical to the very missions served by these business information systems.

If the business events are not explicitly defined and manifest within the business information software and database objects, the context of the business event transactions will be lost during the capture of the database content data. Even more importantly, lost also will be the context and interrelationships of the business' operations themselves.

4. Business Event Data Structures

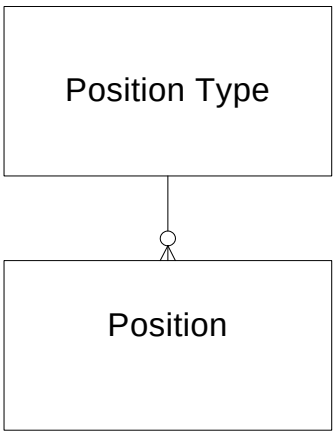

Figure 2. Multi-table Hierarchy

Business event context data is needed for three types of commonly seen database structures: single tables, hierarchies, and networks. Single tables such as Person should contain all the necessary data to reflect its purpose and intent. The captured business event context data only needs to relate to one row from the content table.

Hierarchical data structures commonly exist across multiple tables such as Invoice Header and Detail, Order Header and Detail, and the like. Hierarchical business events often mirror the database object data structures of database object classes. Figure 2 illustrates a multi-table business event for Position Type and Position. In this type of hierarchy, the identifier (most likely) of the Position Type is a proper column in the Position table.

Another form of hierarchical data structure are self-referencing tables such as Organization contains Organization. In this case the value of the parent organization is contained as a value of a column such as Parent Organization Id within the Organization

Figure 3. Self-referencing hierarchy.

table. This is illustrated in Figure 3. In hierarchical data structures, only one parent is allowed for each table.

Network-based business events are quite different from hierarchical business events. Whenever a business event can be invoked by multiple parent events, the business event conforms to a network. For example, the business event, Order Entry, might invoke the child business event, Make a Payment. A different business event, Invoice Processing, accomplished by an on-line customer might also invoke the same contained business event, Make a Payment. What makes this a network is that the "child," Make a Payment, has two parents, Order Entry and Invoice Processing.

Other network examples include Organization Management. One organization might morph into multiple organizations and thereafter be absorbed into a consolidated organization. An organization that purchases three other organizations is a hierarchy. In contrast, an organization that is the result of the merger of two other organizations is a network. Another common network example is reference data. A value set that takes one value and divides it into two others is hierarchy, while the merger of three values into one value is a network.

Because business event transactions such as Make a Payment can have multiple parents as well as multiple children, the business event transaction's data structure has to be a network. Without a network structure, that is, with only a hierarchical data structure, data redundancy and subsequently, conflicts occur. That ultimately leads to update errors as multiple records require update when there should have only been one update.

There are two types of network structures that need to be addressed with respect to business events. The first, shown in Figure 4, is a traditional many-to-many network. It has two "parent" tables and one "interrelationship" table.

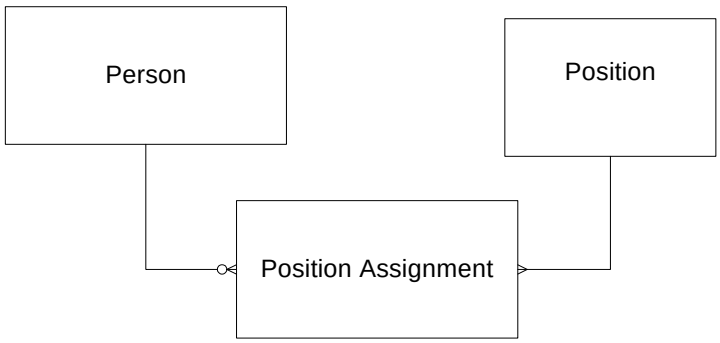

Figure 4. Interrelationship-type Network Structure

In this form of a network there are three tables. The Person table is essentially an employee. The Position table represents the functional jobs of the enterprise. The Position Assignment table represents an association between a person and the position that the person

occupies. Commonly, the Position Assignment table has a Position Assignment Start Date and a Position Assignment End Date. Often, these dates are not allowed to overlap.

The minimal quantity of business event context data that must be represented here must address the Position Assignment table. While the other two tables, Person and Position, exist independently from the Position Assignment data, the Position Assignment business event might require knowledge of the Person and Position for there to be a complete set of business event context data.

The second type of network structure is the bill-of-materials data structure. This data structure contains three tables. It is illustrated in Figure 5. While this bill-of-materials network data structure is commonly employed with product-based manufacturing applications, it has applicability to personnel, financial tracking, and workflow applications. This data structure is organized to represent the relationships among instances of a single object. All three tables comprise the database object class.

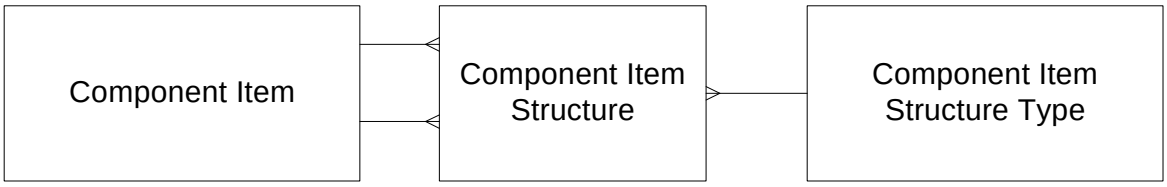

Figure 5. Bill of Materials Data Structure.

The Component Item table, in product composition applications holds all the product's parts. The Component Item Structure table holds the interrelationships among all the parts and supporting part usage information such as quantities among the parts that comprise specific assemblies. The Component Item Structure Type table enables the classification of the collections of Component Item Structures.

As an example, suppose the component item is a bicycle. For a bicycle, there is a collection of discrete contained components. Some of these contained components can be employed multiple times throughout the bicycle. An example is a spoke for either a front or rear wheel, or a bolt and nut. The strategy of a bill of materials data structure is that a given component item is represented once-only in the database no matter how many different times it is employed. The once-only representation is contained in the Component Item table.

There are two relationships between Component Item and Component Item Structure. The top relationship can be termed as an "Active" relationship such as "Contains." An example is that the bicycle contains a frame, a front wheel, and a back wheel. In this example, there would be four component items: Bicycle, Frame, Front Wheel, Back Wheel. There would be a contains relationship from Bicycle to Frame, to Front Wheel, and to Back Wheel. The Component ID value of the bicycle is stored in the Passive Id column of the Component Item Structure table. The Active Id of the three relationship instances of the Component Item Structure table that contains the Id of the Frame, the Front Wheel, and the Back Wheel.

Similarly, there is a second relationship that is passive in form. In this example, the relationship is "Is contained in." There is thus a relationship from each of the contained component items, that is, Frame, Front Wheel, and Back Wheel to the containing Bicycle component item. The Component Id of the Frame, or Front Wheel, or Back Wheel is stored in the Active Id column. The Component Id of the bicycle is stored in the Passive Id.

The purpose of the Component Item Structure Type table is to support classifications of collections of the Component Item Structure records. For example, "Contains" and "Is Contained In." There could be different collections of Component Items such as metal, plastic, imported, or domestic.

Similar to the multi-table network, the minimal required data necessary to represent the business event is the Component Item Structure table. Most likely, however, the rows from the Component Item and the Component Item Structure Type tables are also encapsulated within the business event's context data. The creation and maintenance of the Component Item data and the Component Item Structure Type data is independent from the Component Item Structure relationships that exist between and among Component Items.

Because of the myriad of different database table structures, that is, single tables, and collections of interrelated hierarchical and network tables that map to enterprise business data base objects and because of the inherent complexity of enterprise business functions and their corresponding business events, the need for a robust solution to business event context data is real and must be comprehensive.

5. Solution Approach

A comprehensive solution for business event management must address:

- Both possible and actual business events.
- Simple, hierarchical and network based business events.

The overall strategy is to capture the context data for all business events that are possible to occur, and to capture the context data for all business events that actually do occur. The context data for each business event exists independently from the content data. The business event data takes on an "about-data" or metadata viewpoint. The context data must be related to the various instances of the content data.

Figure 6 illustrates an approach for 1) all possible business event transactions, 2) the history of actual business event transactions, and 3) the interrelationships between all possible business events, all actual business events, and between possible and actual business event transactions, and finally 4) the interrelationship between context data and content data. All three blocks have internal relationships among their rows instances and also relationships between rows across the top, middle, and bottom blocks.

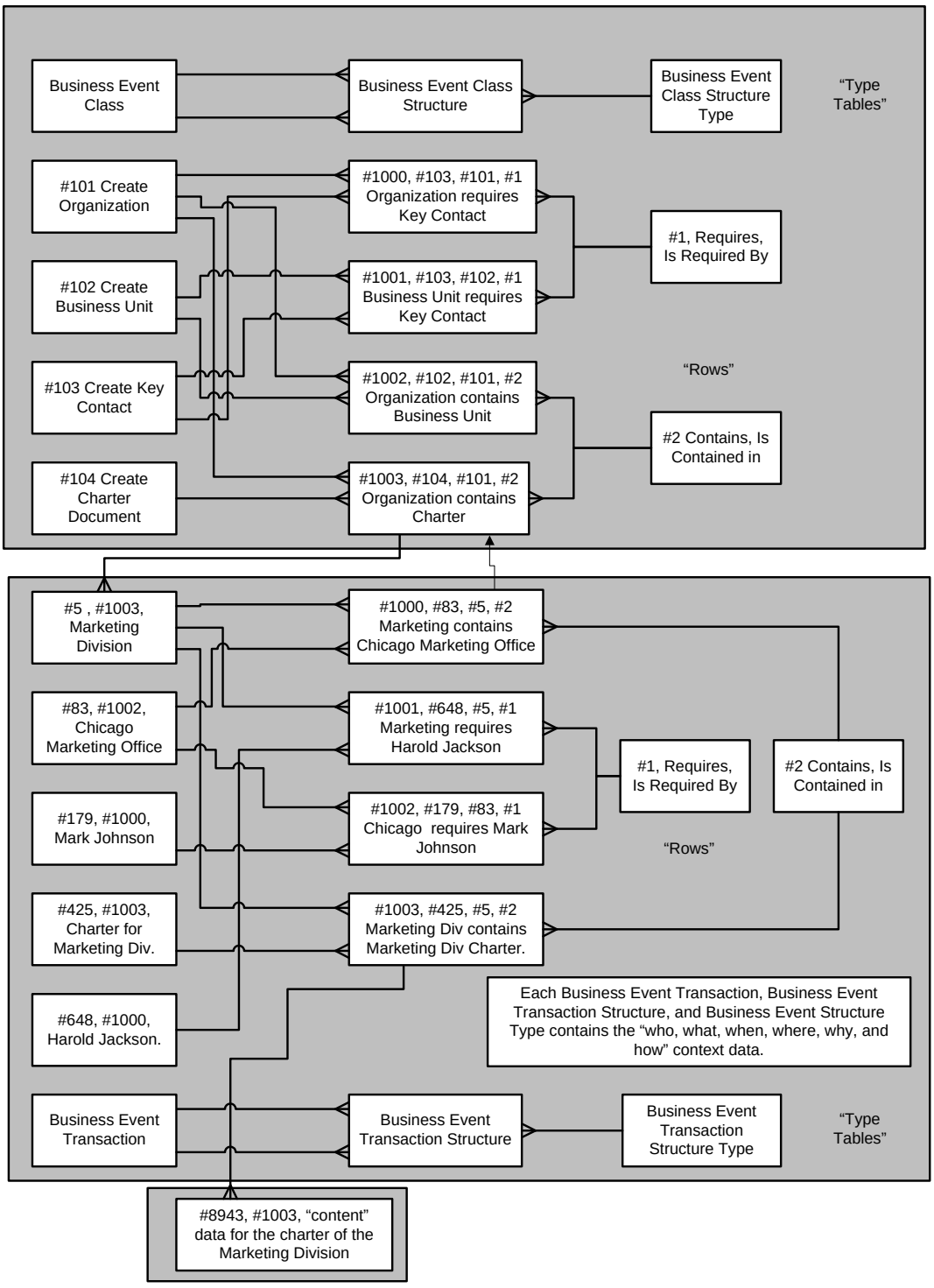

Figure 6. Double network structures to capture both business events and their contexts.

Figure 6 shows three major blocks of tables. The top block represents all possible business event transactions. The middle block represents all actual business event transactions including the business event's contact data. That is, the "who, what, when, where, why and how" of each. The bottom block represents that actual "content" data for a given business transaction.

5.1 All Possible Business Event Context Data

The top block from Figure 6 contains two classes of tables: "Type Tables" and "Instance" rows. In the top part of the top block there are three tables: Business Event Class, Business Event Class Structure, and Business Event Class Structure Type. These tables, when instantiated, represent all ***possible*** business events.

The Business Event Class table rows hold the name and descriptive data for each different business event that can possibly occur. These different business events are identified during the business information system's requirements and use case creation activities. Essentially, the Business Event Class table represents a data-based version of the business information system's processes.

The middle table, Business Event Class Structure, enables the capture of the interrelationships among business event transaction classes. This represents the contextualized history of all possible business events. Essentially, the Business Event Class Structure table represents a data-based version of the interrelationships between and among all the business information systems's processes.

The middle table consists, at a minimum, its own identifier, the ActiveId (of the contained part), the PassiveId (of the containing part), and a foreign key value to the Business Event Class Structure Type table. The stylized name string in this table represents a <part> <relationship> <part> sentence, as in Organization Contains Key Contact.

In both the top block (i.e., Business Event Class) and in the middle block (Business Event Transaction), the two middle tables show the Contains (or Requires) relationship as the "top" relationship line. The column that stores the Business Event Class row Id-value is the Passive Id. For example, the row ID value "101" from the Business Event Class row, #101 Create Organization, is shown in the middle table rows, #1000 and #1002. The value "101" is contained in the third column of each of those two rows. Stylistically, the relationship line is the "top" line.

Two stylized set of names, Requires and Contains, were created to support the semantics between the two related business events. That is, the meaning of the relationship between the ActiveId referenced business event and the PassiveId referenced business event. The table, Business Event Class Structure Type, provides a descriptive name for the inter-relationship but also the applicability to textually describe the meaning of the inter-relationship. In the first instance, it is "Requires" or "Is Required By." From the top block of Figure 6, the Business Events Classes are:

- Create Organization
- Create Business Unit
- Create Key Contact
- Create Charter Document

As a consequence of the development of the various uses cases of the business information system, the Business Event Class Structures, which represents the interrelationships among the business events are:

- Organization requires Key Contact
- Business Unit requires Key Contact
- Organization Contains Business Unit
- Charter contains Organization

The Business Event Class Structure, has two child-to-parent relationships with Business Event Class. This enables a network. The top relationship line is an "active" relationship. For example, the record, “#1003, 104, 101, 2" essentially means “Organization contains and Charter Document.” The “passive” relationship “bottom line” means “Charter Document is contained in Organization.”

As another example and read in the “passive way, the relationship row, #1001 is Key Contract is required by Organization. Note also that #103 Create Key Contact is also required by #102 Create Business Unit. Because the key contact is a "child" of both Organization and Business Unit, the data structure is called a network. That's because a network uniquely enables a child to be "owned" by multiple parents.

5.2 Actual Business Event Context Data

The middle block from Figure 6 also contains "type" and "instance" tables. In this second block, however, the "type" tables are at the bottom of the middle block so as to minimize “crossed lines.” The tables are Business Event Transaction, Business Event Transaction Structure, and Business Event Transaction Structure Type.

Note, the key naming differences between the top and middle block. The top block contains the word, Class, for all three of its tables. The middle black contains the word, Transaction, for its tables. “Class” is proper for the top block because it signifies business event types. “Transaction” is proper for the middle block because it signifies business event instances. In short, the top block is a “Type” layer and the middle block is an “Instance” layer.

From the middle block, Business Event Transactions that actually occurred are:

- Create Marketing Division
- Create Chicago Marketing Office

- Create Key Contact for <Mark Johnson>
- Create Charter for Marketing Division
- Create Key Contact for <Harold Jackson>

In these examples, the Business Event Transaction table not only contains its own Id value and the descriptive information about the business event transaction, it also contains a column that identifies the Id value of the Business Event Class Structure row. This enables the mapping from an actually occurring Business Event Transaction back to its type Business Event Classification Structure row.

As a consequence of the execution of the business information system based use cases, the Business Event Transaction Structures, which represents the interrelationships among the business event transactions that actually occurred are:

- Marketing contains Chicago Marketing Office
- Marketing requires Harold Jackson
- Chicago requires Mark Johnson
- Marketing Division contains Marketing Division Charter

The instance example here is for the Marketing Division. The method of reading the rows is the same as in the top block.

The value from having these “possible” and “actual” business event network data structures is significant. For example, fFrom top to bottom, here’s what one set of data structures “says.”

- Organization contains Charter Document (1003, 104, 101, 2)
- Marketing Division contains Marketing Division Charger Document (1003, 425, 5, 2, 2)
- Content data for the charter for the marketing division (8943, 1003)

What can then be determined are all the business event transactions that occurred at the same time that are related to each other, or are related to their corresponding “possible” business event classes.

Here, once the capture of a Marketing Division Charter is started, a prior step has to be the capturing of the Charter's organization, which in this case is the Marketing Division. Each organization (e.g., the Marketing Division) is represented in the Business Event Transaction table.

The “real” data for the Marketing Divisions data is not contained in any of these Business Event Transaction, Business Event Transaction Structure, and Business Event Structure Type tables. Rather, what is contained is the business event context data for the Marketing Division. That is, the "when, how, where, why, who, and what" of each business event transaction. This is shown in the “box” that is below and to the right of the *Charter contains Marketing Division* business event transaction structure row.

The content data for the Marketing Division is represented by the #8943 row at the bottom of Figure 6. This row has the foreign key value, #1003, back to the Marketing Division business event transaction structure row that, in turn, has a foreign key value #5 back to the Business Event Transaction row, Marketing Division. That row, in turn, has a foreign key #1003 back to the Business Event Class Structure row, Charter contains Organization, which, in turn, maps to the business events, Create Charter Document and Create Organization.

Figure 1, that is subseted in Figures 2 through 5, is the representation of the collection of content tables. The inter-relationships among these content tables naturally exist in the database's design and are traversed through normal business information system processes. In each of these tables, there is a column that contains the foreign key value of the business event's transaction table from the middle block. The role the middle block serves is to be able to access all the other business event transactions that came before or that occurred after the creation of update of the Marketing Division's content data. The role of the top block is to set the actual business event transactions within the context of all possible transactions.

All in all, these three collections of business event tables, that is, possible, actual, and content provide non-redundant, discrete, identifiable, recognizable, trackable, inter-relatable, time-sequenced, and reportable business event information across all business functions that invoke business events.

While this approach may appear indirect and somewhat complex, it represents industry standard best practice for capturing business events, the contextual relationships among business events, and the various collections of business events.

The top and middle blocks of network structures are required because the first represents all ***possible*** business events and interrelationships, and the second represents all ***actual*** business events and relationships. Married together, business reports can be produced that not only indicate what is possible and even required, but also what has actually occurred. That provides the ability to create critical reports of what actually exists and what remains missing.

Through this approach what can be captured are 1) the business event transactions themselves, 2) the history of business event transactions, and 3) the interrelationships among business event transactions.

5. Whitemarsh Methodology Support

This short paper set out a clear interdependence between a business' process model that are manifest through business events and specific database data representations of those business process based business events. Figure 6 clearly shows that a form of the business event processes are records of data in both the top and middle blocks. These processes are identified through examining the artifacts produced from the Whitemarsh Methodology that contains about 125 pages of work tasks. These tasks exist in all comprehensive database project methodologies. That table that follows contains the start of the methodology tasks that relate to the

identification, specification, and ultimate implementation of the business information system processes and supporting database structures for capturing business event data.

Methodology Phase	Task Number and Name	Description of Effort
Preliminary Analysis	1.02.02.04 Identify, name, and briefly describe the business information systems	This task identifies and describes all the appropriate business information system that need to be involved in any existing maintenance and/or interfacing effort. When the business information systems are not yet developed then they are merely identified here but are further detailed as to the business functions, database objects, and business events that are involved.
	1.02.02.06 Identify, name, and briefly describe the business functions ...	This task is the first real task to identify the human functions that are to be performed. Ultimately these functions are manifest as use cases that, when implemented cause the execution of business events that in turn have to be properly captured and tracked.
	1.04.01 Create Resource Life Cycles	This task enables the creation of all the resources and resource life cycle node. It also support the allocation of database object classes and business information systems to the identified and defined resource life cycle nodes.
Conceptual Specification	2.03.02 Create business event model	This task accomplishes the identification and specification of all th business events that are to be accomplished that result in updates to the various databases. These business event definitions occur in conjunction with the definition of the detailed business function model.
	2.03.03 Create detailed business function model	This task is a follow on task to the Preliminary Analysis Phase task that accomplished a high level identification and definition of all the business functions. Once the detailed business functions are determined the business events that are to be accomplished are allocated. The business events, in turn, are associated with business information systems and these detailed mission-organization-functions. As the use cases are detailed, the events within each use case are mapped to the mission-organization-functions as well.
	2.09 Validate through Prototyping	This task is very important. That is because within the prototype a large majority and possibly even all the business events will have been identified. Consequently, the business events can be cast as data and stored in the database tables that correspond to

Methodology Phase	Task Number and Name	Description of Effort
		the top and middle blocks of Figure 6.
Implementation	4.03.03.02.01.01 Create general design for a transaction driven program	This task provided additional detail and a firmer identification of the business events that need to be tracked within the business information systems.
	4.03.03.02.01.03 Create design of update transaction	This task provided additional detail and a firmer identification of the business events that need to be tracked within the business information systems updated transactions.
	4.03.03.02.01.06 Create programming specifications for each update transaction	This task provided additional detail and a firmer identification of the business events that need to be tracked within the programming specifications of the business information systems updated transactions. This is needed to know which business event tracking database tables need to be accessed including the appropriate columns, the setting of certain values for dates, person who has performed the updates, and the like.
	4.03.03.02.01.07 Create programming specification for transaction rollback	This task is very important because in the event a database transaction is rolled back there may be a need to create a business transaction cancelling transaction that is stored in the business event database tables. It may be that the business event was attempted but only that there was a computer failure. In this case, the attempt would likely still need to be recorded.

6. Metabase System Support

As the methodology tasks from Section 5 are accomplished, specification and implementation artifacts are created, interrelated, and are stored into the metabase. The metabase system modules that are directly involved in the accomplishment of business event management are:

Module Name	Role in Business Event Management
Mission, Organization, Function Position Assignment	This module enables the capture of all the business functions that are involved in the identification of the business events that need to be captured.
Business Information System	This module enables the capture of the hierarchies and networks of business information systems and their modules that ultimately are to be executed to accomplish enterprise missions. This data becomes many of the Figure 6 top

Module Name	Role in Business Event Management
	and middle block records.
Data Element Model	This module enables the specification of the business data elements that become the foundation semantics of attribute of entities and columns of table that in turn will form the data structures necessary to capture the business event data.
Specified Data Model	This module enables the creation of the data models based on the concepts that must be captured within the business event data models. Included in each data model are entities, attributes, and relationships. Attributes are mapped back to data elements from the data element model.
Implemented Data Model	This module enables the creation of the data models based on the real databases that are to exist in support of the data structures for business event classes and transactions. Included in each database models are tables, column, and inter-table relationships. Columns are mapped back to data elements from the data element model.
Operational Data Model	This module enables the creation of the DBMS bound data models that will contain the actual business event class and transaction that support business event capture, tracking and management. Included in each database models are DBMS tables, DBMS columns, and DBMS interrelationships.
View Data Model	This module enables the creation of the specifications of specific interchanges between application systems that are tracking business events and the databases that are storing the tracked business events.
Resource Life Cycle Analysis	This module enables the identification, capture, and interrelationships of the enterprise resources and their resource life cycle nodes that reflect the result of the execution of a collection of business events to achieve the resource life cycle states necessary by the organizations as they execute their functions to accomplish enterprise missions.
Database Object Model	This model enables the complete specification of all the database tables and processes that are involved in the capture of the business events from across all the business information systems. The database object classes are then allocated to the Resource Life Cycle nodes.
Use Case Model (Spring 2010)	This model is created during requirements analysis and design. It essentially becomes the detailed function model and is interrelated with the Mission-Organization-Function model, Database Object Classes, and the Business Information Systems models. Each use case ultimately becomes the employment of a business event.
Document and Form Model (January 2010)	This model is created during the requirements phase and identifies, describes and details the various forms and/or documents and contents of the forms and documents that must be addressed during the development of a complete inventory of business events. This model, like the use case model is interrelated with the Mission-Organization-Function model, and the Business Information Systems models.

7. Conclusions

The practical application of the points made in this paper include:

- There is a clear rationale for the capture of critical business event data. That is, the "who, what, when, where, why, and how" of each business event transaction.
- There is a clear distinction between all possible business event transactions and all actual business event transactions.
- The difference between the all-possible and the actual transactions need to be tracked and interrelated so that critical behavior reports can be generated.
- The specifications necessary to capture business event transactions are independent from those that capture the content data associated with a database transaction.
- The database tables for a business event transaction are independent from the database transaction. Additionally, there are relationships among all the business event transactions, relationships to business event classes, and among the business event transactions and the database content transactions.

8. References

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

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

Paper	URL
Iterations of Database Design	http://www.wiscorp.com/iterations_of_database_design.pdf
Data Management Conferences	http://www.wiscorp.com/dama2002.zip http://www.wiscorp.com/dama2003.zip http://www.wiscorp.com/wrad2000.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

Paper	URL
Work Breakdown Structures Database Project Work plan Templates Information Systems Development Methodology Phases 1 and 2 Whitemarsh Project Estimating Work plan Development	http://www.wiscorp.com/wwmembr/mbr_products_dp.html
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