



## DATABASE OBJECTS: THE FOUNDATION STONES OF ENTERPRISE DATABASE

Michael Gorman - Whitemarsh Information Systems Corp.

There are generally considered to be three classes of objects: display objects, wholly contained process objects, and business objects. Display objects embrace buttons on a screen, a drop list of menu choices, a graphical user interface (GUI), or complete engineering drawings. Wholly contained process objects are for example, the COSINE function, a nautical distance function that when given two geographical coordinates returns the geographic distance between them, or a well defined process that takes standard arguments and returns a specific value such as asking for the net asset value for a business given all assets and liabilities. Finally, business objects encompass business components like an insurance policy [information system] that accomplishes whole business transactions in a certain manner.

While three object classes have their proponents and detractors, what all three object classes have in common is that they are first and foremost self contained software modules/systems in the form of an executable that behaves according to certain fixed rules.

A database object is none of these. It is its own class. While database objects share some common names and definitions with the other three object classes, that is, encapsulation, inheritance, and polymorphism, database objects are unique to both database and DBMS.

Database objects are identified, designed, implemented, operated through, and evolved, or maintained through just one type of data processing facility, a database management system (DBMS). If the available DBMS is an ANSI SQL3 DBMS (download SQL\_BOM from the [www.wiscorp.com](http://www.wiscorp.com)) then database object definition and use can be direct. Otherwise, database objects can only be indirectly approached through proprietary facilities in one or more DBMSs.

Database objects are essential to the proper understanding, specification, implementation, and maintenance of world-wide heterogeneous databases. Database objects fit within the enterprise's Knowledge Worker Framework (download the Knowledge Worker Framework book from [www.wiscorp.com](http://www.wiscorp.com)).

Database objects are not new. They were started in certain DBMS types (e.g., IDMS, IDS, GIM, Inquire, and Adabas) in the late 1960s. Relational DBMS such as DB2, Oracle, Informix, and Sybase, however, stopped the march to database objects dead in its tracks. It was not until the ANSI SQL3 data model moved away from relational and not until a whole programming language was incorporated into ANSI SQL3 that the march to database objects restarted. The newest versions of IDMS, DB2, Oracle, Informix, and Sybase have all started to support the data structure and process features essential for database objects. Even if the twenty-year delay had not happened, computers, networks, languages and operating systems were just not sophisticated enough to make

database objects successful.

Database objects were formulated almost 20 years ago by the late Matt Flavin in his 1979 Yourdon Monograph, *Fundamentals of Information Modeling*. During the Seventies, Matt (who worked for Infodata of Rochester, NY and Fairfax, VA.), accomplished very early database management system research and development. Infodata's DBMS, Inquire, was widely used in the U.S. Federal Government. Matt represented Infodata to the X3H2, the ANSI Database Languages committee in the late Seventies.

Database objects existed only on paper only until the ANSI database languages committee, X3H2, working since early 1993 on the specification of SQL, formulated the essential linguistic components of database objects.

### **The Business Case for Database Objects**

Distributed, client/server data and processes are here to stay, and rightly so. Not only are they empowering, they are essential because enterprises are highly distributed and world wide. Enterprises must be able to respond to local needs, laws, customs and mores. But, if business are designed and tuned to respond to local situations, how can they act in concert within their world-wide communities? How can you have world-wide consistency and semantics without suffocating local needs and practices? How can both ends of the information resources spectrum be satisfied?

Business data needs far exceed today's DBMS's two dimensional table capabilities and simple column based constraints. Businesses cry out for semantically rich data management to meet business needs across world-wide, heterogeneous hardware and operating system environments. Business data management environments must behave consistently regardless of their host computing hardware environment, operating systems, or DBMS vendors, and must be easy to specify, implement, use and maintain.

Businesses require hierarchies of complex data tables, collections of integrated rules for data integrity, well defined procedure sets, and fixed transformations that move a business policy--data is just executed policy--from one well defined state to another. Examples of business needs include insurance policies and claims, court cases and documents, public safety incidents, sales and marketing databases that contain customers, sales organizations, forecasts, orders, deliveries, and product sales statistics, inventory control and deployment, and human resources.

### **Components of Database Objects**

Database objects "live" entirely within the domain of the DBMS. Database objects can be both persistent and non-persistent, and can span single or multiple-tables.

A persistent database object is one that is stored and is retrievable over long periods of time. An example is an insurance policy along with its full compliment of payments, renewals, and claims. Another example of a persistent object is the rotating-three dimensional views of a mechanical part.

A non-persistent database object is one that is materialized and displayed but is not able to be re-materialized because some of its components are not retained after the database object's display is terminated.

Non-persistent objects are dynamically produced from database data and exist only for the life of their "display." An example might be evening-news weather displays. The weather map, that is, the states, cities, streams, rivers, etc are all persistent database data. The actual streams of clouds, high and low pressure fronts, cloud formations, and the like are time-sequenced BLOBS that are dynamically displayed across the screen. While the displayed database objects may be recorded via videotape and redisplayed at a later time, the detailed components, which upon retrieval make up the non-persistent database objects, is not stored.

By the time the news-cast is over, the BLOB parts are discarded. Other than for a videotape replay, the complete set of non-persistent database objects is gone. The persistent part is traditional data structures with the appropriate quantity of indexes. The BLOBs are just non-indexed streams of binary data that are stored in a very primitive format.

Persistent database objects are those that are stored in a database on a permanent basis. Included are traditional "relational" data, abstract data types of complex structures (like an entire auto accident claim that might include BLOBs, free text streams, etc.).

Single table database objects are those that are fully defined within a single row of an SQL/3 table structure. With SQL/3, columns can support very complex structures, such as simple values, lists, sets, multi-sets, and abstract data types of arbitrary complexity. This capability is quite common in hierarchical DBMSs like System 2000 and in independent logical file data model DBMSs such as Adabas, Model 204, Inquire, and Datacom/DB.

For example, in a product sales database, the single table called sales has product number as the primary key with other columns for product name, product description, and the like. The sales table sales column in contrast, contains product sales by year by month by region, district and territory by salesman. That's a single column with six dimensions of values. Prior to SQL/3 such product sales information would require multiple tables with the attendant keys, joins, and computer processing melt-downs. Since the salesman's object identifier is contained as an integral component of the sales data, the salesman's full set of data is accessible through normal SQL language processing. A referenced database object, that is, the referenced salesman's data is not

considered a formal part of a single table database object.

Multi-table objects are those that are implemented across multiple tables. For example, an insurance policy may have several dozen tables that make up its full definition. One and only one table is considered as the database object's root table. The database object root table contains among many things, the object's identity column. A row from the root table is the head-row of the database object. All other related tables within the multi-table database object contain other information related to the object. In the insurance policy example, a claim might be contained in one or more database object tables. Other database object tables would contain the underwriting information regarding the person about whom the policy was issued. The person is a different single or multi-table database object. Each table within a multi-table object may consist of single valued columns, it can also support lists, sets, multi-sets, and abstract data types of arbitrary complexity.

Not only do most businesses contain multi-table database objects, the majority of business applications are examples of multi-table database objects. A quick look at business applications reveals that inescapable conclusion.

Database objects, regardless of persistence and regardless of whether single or multi-tables contain the same four-part composition:

- **Data Structure:** the set of data structures (simple and complex collections of tables) that map onto the different value sets for real world database objects such as an auto accident, vehicle and emergency medicine incident.
- **Database Object Process:** the set of database object processes that enforce the integrity of columns (simple or complex), references between database objects and actions among contained data structure segments, the proper computer-based rules governing data structure segment insertion, modification, and deletion. For example, accomplishing the proper and complete storage of an auto accident.
- **Database Object Information System:** the set of specifications that control, sequence, and iterate the execution of various database object processes that 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.) that accomplish entry of the auto accident information.
- **Database Object State:** The value states of a database object that represent the after-state of the successful accomplishment of one or more recognizable business events. Examples of business events are auto accident initiation, involved vehicle entry, involved person entry, and auto accident DUI (driving under the influence of

alcohol/drugs) involvement. Database object state changes are initiated through named business events that are contained in business functions. The business function, auto accident investigation includes the business event, auto-accident-incident initiation, which in turn causes the incident initiation database object information system to execute, which in turn causes several database object processes to cause the auto accident incident to be materialized in the database.

A database object is specified to the SQL/3 DBMS through the SQL/3 definition language (DDL). All four components of a database object operate within the "firewall" of the DBMS. This ensures that database objects are protected from improper access or manipulation by 3GLs, or 4GLs.

### **Database Objects Summary and Benefits**

The benefits derived from database objects include:

- Whole containment within SQL/3 DBMS
- Access to both type and instance components
- Complete expression through syntax
- Import and export through ISO/ANSI standard SQL/3 facilities
- Ability to be distributed and consistent operations via all SQL compliant DBMSs
- Independence from presentation-layer and operating-system bindings

Because database objects are wholly contained within SQL/3 DBMS they can be centrally accessed and manipulated regardless of the end-user environment, that is, batch, on-line, stand-alone "fat" clients, or traditional client/server.

*Michael M. Gorman, President of Whitemarsh Information Systems Corporation, has been involved in database and DBMS for almost 30 years. Mr. Gorman has been the Secretary of the ANSI Database Languages Committee, X3H2 for 19 years. X3H2 standardizes SQL. A full list of Whitemarsh's clients and products can be found on the web site, [www.wiscorp.com](http://www.wiscorp.com). The goal of the web site, WisWeb, is to make data management books, courses, methodologies, software, and metrics available to the database community through electronic publishing and downloading. WisWeb memberships are very reasonable and are designed for the individual, the ISD organization, universities/colleges, and professional training organizations.*

*Whitemarsh Information Systems Corporation  
2008 Althea Lane  
Bowie, Maryland 20716-1518 USA  
Phone: +1.301.249.1142  
FAX: +1.301.249.8955  
Email: [mmgorman@wiscorp.com](mailto:mmgorman@wiscorp.com)*

WWWeb: <http://www.wiscorp.com>

[\[The Article Archive\]](#)

**The Data Administration Newsletter**  
Robert S. Seiner - Publisher and Editor - [rseiner@tdan.com](mailto:rseiner@tdan.com)