DATA IS EXECUTED POLICY

A Case Study

Michael M. Gorman

ABSTRACT

Data is executed policy in that data represents the “proof” that policies are executed. From that it follows that the data definitions systems employ, that is, database schemas, are really policy definitions. And from that, it can be concluded that if a database’s definition is incorrect or incomplete, then, in essence, the enterprise’s policies do not exist. Or at least, enterprise policy executions cannot be proven.

This paper, Data Is Executed Policy, shows via a case study that if the underlying database design and the attendant software system’s infrastructure are not synchronized with the policy of the enterprise, then the enterprise operates at best in a suboptimal fashion. What is presented is the nature of the enterprise, the existing database and information system’s environment and consequences of that incompletely specified environment.

The paper then presents the work accomplished to complete both policy definition and the development of the business system to mirror enterprise policy. Because Clarion for Windows was employed, the complete requirements analysis and design through production system implementation was able to be accomplished in nine staff months, not counting data conversion and database administration activities. The key statistics of the effort are: 60 tables, 240 “programs,” and about 90K lines of code.

1. RATIONALE FOR “DATA IS EXECUTED POLICY”

Quality database is an expression of organization, clarity, and precision. It may or may not be computerized. If it is, it may exist on a desktop, a server, or a large mainframe. Finally, a database may or may not be centralized. Notwithstanding the mode, the mechanism, or the form of database implementation and operation, the codification of and adherence to data semantics, which are the rules for meaning, validity, and usage, are prerequisite to a successful database.

When a database is on a computer, it represents the automation of the knowledge component of a business, which is manifest through the business's quality operation, planning and management. With quality database, management of a business can research the past, organize the present, and plan for the future. To have quality database is to have defined policy because each database object class's data structure within the database is the data representation of a policy's definition.

- The database object class's processes through which rows are added, deleted, and modified are the mechanisms necessary for policy execution, that is, the policies procedures through which database objects are transformed from one valid state to another.
- A fully defined policy includes both its complete definition and its necessary steps for coherent execution.
- Interrelated collections of rows across multiple tables of one object class and across multiple database object classes form more comprehensive policies.

When data is seen as executed policy, and is realized through database object classes, then quality databases support following within the enterprise:

- Business information systems that are a coherent union of the policy and then execution of the procedures that represent the accomplishment of the policy.
- Consistent collection and/or modification of policy instances through the life cycle of the policy.
- Consistent execution of policies when ever, where ever, and how ever deployed as the essence of the policy and the totality of its critical procedures are encapsulated within the database object class itself.
- Minimized redundancy and consistent policy implementations across distributed environments as the database object class can be distributed through encapsulated strategies.
- Comparable instances of deployed policies as they are independent of hardware architectures and operating systems.

But, what forms the basis of an database object? Simply, it is a business' policies and procedures. While policies can exist without procedures, the converse is not true. This ontological priority dictates that procedure is dependent on policy. Not only, in this case, do they go together like hand and glove, the glove (procedure) serves no useful purpose without the hand (policy).

A database object is a person, place, or thing that has internal consistency, and is transformed from one valid, predefined state to another through well-defined rules. The minimum value states are null and valued. The internal behavior of a database object class as it transforms database objects from one state to another is immaterial to its user. Database object classes conform to the requirements of business rather than the converse. Database objects are the corporate memory of the enterprise. All the rest is anecdotes.

The internal specifications of a database object are independent of its implementation. Because database object specifications are ANSI SQL1999 standard, different SQL1999 DBMS vendors (here "DB" stands for database object) are free to implement the ANSI SQL1999 specifications as they like just so long as two conditions are true: the database object specifications are portable from one SQL1999/DBMS to