The Evolution of the Meta-Data Concept: Dictionaries, Catalogs, and Repositories

Mark L. Gillenson
University of Miami

Raymond D. Frost
Central Connecticut State University

The concept of storing and using data about data -- metadata -- has been implemented in a wide variety of products. These include passive and active data dictionaries, relational catalogs, relational storage of dictionary data, an ANSI standard and, recently, repositories and object-oriented DBMS catalogs. This article traces the evolution of these developments, noting their origins, their contributions, and representative commercial products based on them. In particular, it goes into the repository and the OODBMS catalog and draws conclusions about the importance of their relationship for the future of the IS environment.

The need for some form of meta-data, the concept of storing data about a company's data and its information systems (IS) operations, goes back at least as far as the beginning of the 1960s. COBOL copy statements were used as early as that to provide consistent data definitions that could be referenced by different applications. Eventually, products known as data dictionaries developed as sets of files and associated tools that aided in the management of all aspects of the IS function. As time went on, data dictionaries developed capabilities that tied them to other software products and, in some cases, made them indispensable parts of the execution-time database environment (Mark and Roussopoulos, 1986; Navathe and Kerschberg, 1986). Relational database management systems (DBMS), which began to gain popularity in the early 1980s, included a feature called a catalog. The relational DBMS catalog seemed to be somewhere in the same ballpark as the data dictionary but it at once included more functionality in some respects and less in others. Now, as we enter the decade of the 1990s, a new idea, generally referred to as the repository is being touted. Furthermore, a new DBMS paradigm, the object-oriented DBMS (OODBMS), has been introduced.

In this article, we shall trace the evolution of the meta-data concept from its inception as the data dictionary through to its newest forms as the repository and the OODBMS catalog. We shall present the various developmental stages roughly in their chronological order of appearance, although anyone at all familiar with the IS world should reasonably expect that there have been overlapping and parallel developments among them. The stages are: passive data dictionary, active data dictionary, relational catalog, hybrid relational data dictionary, American National Standards Institute (ANSI) Information Resource Dictionary System (IRDS), repository, and OODBMS catalog. Figure 1 shows the flow among these stages, which we shall describe, below.

Passive and Active Data Dictionaries

The earliest data dictionaries were passive, in nature. The term passive means that the data dictionary is used in a substantially offline manner, relative to the running of the DBMS. Passive dictionaries are typically used as system documentation tools which are available for query by appropriate IS personnel. Several passive data dictionaries came onto the market in the early and mid-1970s and have been updated and enhanced since then (Lefkovits, 1977; Lefkovits, Sibley, and Lefkovits, 1983; Leong-Hong and Plagman, 1982; Narayan, 1988; Van Duyn, 1982; Wertz, 1989). Some, including University Computing’s UCC TEN, IBM’s DB/DC Data...
Dictionary, and Cincom’s Data Dictionary are designed to run as applications of specific database management systems. Others, including Synergetics’ Data Catalogue and Manager Software Products’ (MSP) Datamanager are “free-standing” collections of files.

The success rate of these passive dictionaries, as a class, has varied widely. Some companies have embraced them as an important part of their information systems environments. Such companies mandated, for example, that populating the dictionary was to become a part of the application development standard operating procedure and that all database control blocks were to be generated from the dictionary. Other companies found that their procedures, their personnel, or both, were incompatible with the data dictionary concept. It was not uncommon for a data dictionary to be brought into a company on a trial basis, only to languish, unused.

After the initial conception and development of the data dictionary concept, perhaps the most important idea for enhancing it was to make it active, particularly with respect to the database management system. A data dictionary is said to be active if it is a required part of the execution-time environment of a DBMS. That is, the meta-data in the dictionary is needed by the DBMS to complete some part of its functioning. An example of this is in data security (Narayan, 1988), in which the data dictionary contains authorization data about which personnel or terminals are authorized to execute which programs. The dictionary is active if, in the execution-time environment, the DBMS queries the dictionary for this authorization data. In general, when an active role for a data dictionary is conceived, care must be taken to assure that it will not become a bottleneck and therefore cause a performance problem. A variation on the active concept is when the dictionary is not a required part of the DBMS execution-time environment, but is (or may optionally be) a required part of the compile-time environment (e.g. to provide data descriptions during program compilation) (Allen, Loomis, and Mannino, 1982). If the dictionary was useful as a documentation device, it could be much more useful if it was actively connected to the DBMS. An example of an active (and integrated) data dictionary is the Integrated Data Dictionary (IDD) of Computer Associates’ IDMS/R database management system (Husband, McHenry, and Wooten, 1987; Martin, Derer, and Leben, 1990; Towner, 1989).

We also note that in addition to being described as active versus passive, data dictionaries are described as integrated versus non-integrated. Unfortunately, there is little agreement about the precise definitions of these terms and the distinctions between them (Allen, Loomis, and Mannino, 1982; Narayan, 1988; Wertz, 1989). The term integrated, as applied to data dictionaries, generally means that the dictionary can send data to another software tool or receive data from one, further automating the operational environment. For example, if a dictionary can send meta-data to a program that produces control blocks for a DBMS data structure, it might be said to be integrated with the DBMS. (If management mandates that all DBMS control blocks will be produced this way, some might argue that the level of integration is still higher.) But, integration can also refer to the dictionary’s ability to interact with software tools other than the DBMS in the IS environment. For example, if a report generator can use dictionary data as input in constructing reports, it would be said to be integrated with the dictionary. By this definition, virtually all data dictionaries have been integrated, to some extent, from their inceptions and the degree of integration has steadily increased over time. To complicate matters further, the terms active and integrated are sometimes taken to be synonymous, as are passive and non-integrated.

Another important advance in data dictionaries was the implementation of the concept known as, “extensibility,” as for example in Release 3 of IBM’s DB/DC Data Dictionary. This feature allowed the introduction of additional system entities and attributes and the ability to cross reference them with each other and with standard system entities. This development permitted the storage of much more information about the systems using the data, the business functions using the systems, and the data center staff.

Relational Catalogs and Hybrid Relational Data Dictionaries

The commercial explosion of relational database management systems in the early 1980s brought with it a new requirement for highly sophisticated query optimizers. Unlike the hierarchical, network, and flat-file integrated databases then in use, relational databases lacked physical linkages (i.e. direct address pointers) between related pieces of data. Query optimizers, which analyze a query to determine the most efficient strategy for responding to it, were developed to achieve reasonable execution-time performance, particularly when related data from different relational tables had to be integrated.

In order for relational query optimizers to work, they require information about the relational tables involved in the query, such as which fields are in which tables, which fields have indexes built over them, which indexes are “clustered”, which fields or indexes are specified to have unique values, and so on. This information must be easily and immediately accessible by the relational DBMS and it must be absolutely up-to-date. From this requirement was born the relational catalog, which is the term used in IBM’s DB2 (Date and White, 1989; IBM Corp., 1986) and SQL/DS relational DBMSs. Oracle’s ORACLE relational DBMS (Hoechst, Melander, and Chabris, 1990) and Ingres’ INGRES relational DBMS (Date, 1987) use the term dictionary to describe the same type of facility. Note that the relational catalog, in that it serves as the description of the application data structures, could be considered to be a descendant of the earlier CODASYL network schema concept, as indicated in Figure 1. However, as will be
Related Content

Leveraging Early Aspects in End-to-End Model Driven Development for Non-Functional Properties in Service Oriented Architecture  
[www.igi-global.com/article/leveraging-early-aspects-end-end/52994?camid=4v1a](www.igi-global.com/article/leveraging-early-aspects-end-end/52994?camid=4v1a)

Bug Fixing Practices within Free/Libre Open Source Software Development Teams  
[www.igi-global.com/chapter/bug-fixing-practices-within-free/39350?camid=4v1a](www.igi-global.com/chapter/bug-fixing-practices-within-free/39350?camid=4v1a)

Integrity Checking and Maintenance in Relational and Deductive Database and Beyond  
[www.igi-global.com/chapter/integrity-checking-maintenance-relational-deductive/24236?camid=4v1a](www.igi-global.com/chapter/integrity-checking-maintenance-relational-deductive/24236?camid=4v1a)

Energy and Latency Efficient Access of Wireless XML Stream  
[www.igi-global.com/article/energy-latency-efficient-access-wireless/39116?camid=4v1a](www.igi-global.com/article/energy-latency-efficient-access-wireless/39116?camid=4v1a)