A Metadata Management System to Support Data Interoperability, Reuse and Sharing

Stephanie Cammarata, Iris Kameny, Judy Lender, and Corrinne Replogle
RAND Corporation

Users of scientific databases need descriptional information for understanding the definitions, abbreviations, acronyms, and genealogy of data stored and maintained in a database management system. This type of knowledge or metadata includes a wide range of information to help users locate, access, browse, clean, and aggregate databases. In this paper we describe the RAND Metadata Management System (RMMS), which is the result of a project to develop and implement specifications for storing and maintaining descriptional metadata. RMMS promotes the sharing, reuse, and interoperability of scientific data by improving five major DBMS data administration activities: documentation, version management, history management, derived databases, and standard data elements. We discuss the motivation for the development of RMMS, the functionality of RMMS, and its implementation architecture.

Electronic databases and database management systems (DBMS) are becoming commonplace in scientific and research communities. However, when scientific databases are installed in a DBMS, electronic documentation or other descriptive information is rarely included. The absence of such information leaves users at a loss for understanding the definitions, abbreviations, acronyms, and descriptions of data stored and maintained in a DBMS.

To effectively use and share these databases, additional information or metadata is needed to define and describe databases and their contents (Mark and Roussopoulos, 1986; McCarthy, 1982). Metadata can take the form of (1) information about the database as an aggregated whole or (2) information about actual data elements stored in the database. For example, from an aggregate viewpoint, it is important to know the dates for which a database is valid, how the database was generated or where it was acquired from, and the relationship between a current version and a previous version of the database. Metadata describing specific data elements is based on the schema organization of the database and includes information such as the measurement units and value constraints of a data element. For instance, if a data element stores the weight of an aircraft fuselage, an engineer would like to know if the value represents pounds or kilograms. Similarly, if the fuselage should not exceed a certain weight, a user should have access to the exact weight limit and be notified if an updated value exceeds the limit.

Metadata covers a wide range of information that promotes the sharing, reuse, and interoperability of data, including information to help locate, access, browse, clean, and aggregate databases. The importance of metadata has intensified with the rapid growth of scientific and engineering databases. Users of these databases tend to access and integrate subsets of databases dynamically and interactively, depending on the study, simulation, or analysis which is being designed. Historically, database metadata was not needed by business and financial enterprises because they relied on databases and DBMS to record transactions applied by static application programs such as accounts payable, sales, and general ledger applications. However, even business users are taking advantage of ad hoc query capabilities and therefore will be requiring metadata repositories. For example, international financial services require metadata management for identifying currencies (e.g., Canadian dollars, Chilean pesos), recording exchange rates, and computing functions for currency conversion. Another important advantage of metadata is that it begins to facilitate data sharing and interoperability across functional areas of an application. To achieve interoperability requires agreement about the meaning and usage of domain concepts, domain entities, and corresponding data elements of the entities.

This paper describes the RAND Metadata Management System (RMMS) whose function is the storage and mainte-
nance of metadata to improve five major DBMS data administration activities: documentation, version management, history management, derived databases, and standard data elements. A metadata management system maintains not only metadata information, but also procedures needed to apply metadata, for example, to verify value constraints or to convert metric units (Kerschberg, Marchand, and Sen, 1983). Although a “management system” typically includes facilities for user interaction and maintenance, in this document we focus primarily on the metadata storage structures within RMMS. Detailed discussion of user interface and maintenance designs is beyond the scope of this paper.

At RAND, RMMS maintains metadata about military databases including weapon systems, military forces, and intelligence information. These databases are used as input to battle management and command and control simulation systems and analytical models. RMMS has also been proposed as a metadata management system for environmental databases that monitor the generation and management of industrial waste materials.

**Motivation**

A National Science Foundation study has identified the management of metadata as one of the primary issues in scientific database management (French, Jones, and Pfaltz, 1990). Below, we motivate the need for metadata management systems by identifying a number of examples of the use of metadata within a wide range of scientific disciplines.

- Social science studies integrating census data over many decades must be able to compare the schemas of different versions because through the years different fields have been recorded. For example, census surveys early in the century asked if households had a flush toilet.

- In military command and control simulations, it is important to maintain a history of database updates to analyze how a data field has changed over time.

- Different environmental waste databases maintain contaminant levels differently, for instance, as concentration percentages or as a pair of weights representing solid waste and total waste. To compare contaminates across two such databases requires knowledge of the different representations and conversion procedures.

- The dimensions of an aircraft fuselage are constrained by design specifications. These constraints can be stored as metadata to be used for cleaning and verifying data values.

- For a study analyzing international trends and costs of new technology breakthroughs, a researcher may need to retrieve data on foreign patents and will compare (1) calendar dates represented in different formats (e.g., mmddyy vs. yymmdd) and (2) different monetary currencies. Integrating data with these incompatibilities also requires metadata.

- In survey databases, most character fields are numerically encoded. If one survey codes 1 = female; 2 = male, and another survey is the opposite, then integration processes must recognize and reconcile these differences.

- Metadata is also an ideal vehicle for browsing, such as identifying databases that contain information on military airfield and runway assets. Metadata serves to link references from standard data elements such as “airfield” and “runway” to the databases which contain data for these data elements.

Effective sharing and integration of scientific databases depends on the availability of metadata. As exemplified above, metadata refers not only to descriptional information but also to procedures, e.g., for conversion of units and verification of value constraints. The goal of a metadata management system is to centralize and standardize metadata information and associated procedures. Once this goal is achieved, users will have an extensive set of capabilities for learning about the contents of a database, retrieving desired data, and combining data from different databases.

**Issues**

In this section we identify five specific issues or needs which we addressed during the development of RMMS. Each of these issues, discussed below, responds to a limitation faced by researchers who require data from application databases as input to simulation, modeling, and analysis systems.

The need for complete, thorough, and standard data documentation. In the past, hardcopy documentation has been limited, ad hoc, and sketchy; electronic documentation has been virtually non-existent. Some databases are delivered with hardcopy manuals or codebooks, but most often these must be acquired independently from the acquisition of the electronic databases. These manuals usually include system information for a database administrator but are rarely useful for casual users who want an overview of database contents. Instead, knowledge of the semantics or conceptual meaning of the database is frequently passed through word of mouth from experienced users to novices. This situation has many drawbacks leading to inconsistent and inaccurate use of data. First, since precise and definitive semantics of attributes is not stated explicitly, there may be subtle (or gross) differences in the meaning that users attribute to data elements. Second, users frequently need to integrate, aggregate,
or otherwise combine data from different databases to derive a new database. Therefore, users must guess what attribute in one database corresponds conceptually to an attribute in another database. Even when users spend much time learning about the organization and semantics of the data, their knowledge is not recorded and a new user must subsequently start from scratch to collect relevant information.

The need to record and manage information about different versions of different databases. Administrators of scientific databases must manage different versions of databases which originate from a variety of sources. Many studies require databases that are acquired from outside agencies which release new versions regularly, e.g., weekly, monthly, yearly. For databases which are generated by a tightly-coupled data collection activity such as the output of an experiment or simulation, the elapsed time between versions may only be minutes. In all applications, however, installing a new version means asking questions like:

- What should we do with the old version?
- How long should we maintain the old version?
- Should we re-apply corrections to the new version which were previously made to the old version?
- Can we recreate or reload the old version if an analyst needs to rerun an analysis or needs a series of old versions for a longitudinal study?

The answers to these questions constitute policies and automated procedures for version management and should be included as part of a metadata management system. Although the procedures may vary for different databases, they should be consistent across all versions of the same database.

The need to maintain a history of the changes made to database tables, schema, and data values. Facilities for version management (described above) maintain information about different versions. Facilities for recording the actual old values and the new updated values (or history of data values) should also be addressed by a metadata management system. This capability will benefit those applications which require a complete audit trail (at the logical level) of all additions, modifications, and deletions of the data or schema. Although DBMS logging facilities record all database transactions, these results are system level audit trails intended primarily for database administrators. We identified four major objectives for this facility. First, we wanted to minimize the amount of storage needed to maintain old versions. Frequently, in a new version only a small percentage of the database has changed. Therefore, simply maintaining complete old versions independently or as archived databases is not acceptable. A second objective was to explicitly record all additions, modifications, and deletions to either data values or the schema. For example, if a new table was added in a new version, this should be reflected explicitly in the version management system in addition to the registration of a new table in the DBMS schema structures. Recreating previous versions or snapshots was a third objective of a history management facility. With this capability, a user or database administrator can directly determine the differences between two different versions. Finally, we needed to insure that the underlying history management facilities are transparent to a user and that when accessing the current version of the database, no extra overhead is incurred.

The need to facilitate derived databases for input to simulation models and for sharing among models. RAND does not generally produce original source data; rather, they acquire databases from external agencies and derive subsets for specific studies. We define a derived database as one that is NOT generated or distributed by an external outside agency. That is, a derived database is one which is generated within RAND’s data management facility by some combination of (1) manually collecting or composing data, (2) automatically collecting data (such as output from a simulation model), (3) retrieving subsets of data from one or more non-derived databases, (4) applying transformations to subsets of data from non-derived databases, and (5) integrating subsets of data from sources (1) through (4). For the remainder of this paper, we refer to non-derived databases as “external” databases because these databases will generally be acquired from outside sources. The goal is to store and maintain derived databases as full-fledged external databases that can be referenced and accessed as if they were acquired as an external database. As full-fledged databases, these derived versions should include the same type of metadata information as their external database counterparts plus metadata documenting how they were derived. Ideally, a metadata management system should also automate the generation of metadata for a derived database, depending on the source of the external data.

The need to standardize the names of data elements that are (1) conceptually the same but are named differently or (2) named the same but are conceptually different. This issue addresses a common practice among scientific database users, namely aggregating, integrating, and combining data from different databases to produce a new database. For example, if a user is deriving a database of commercial airfields in the United States and Canada, he or she will retrieve data from at least two databases: one representing US commercial airfields and another representing Canadian airfields. Suppose the column name representing runway length in the US database is called “runwayln”, and the column representing runway length in the Canadian database is named “rwlength”. Further suppose that the length of runways on US airfields is recorded in feet and that the length of runways on Canadian airfields is stored in kilometers. The user is therefore faced with two decisions: (1) in the derived
database, what should be the column name for runway length (i.e., “runwayln” or “rwlength” or something else); and (2) what should be the units of the column representing runway length (i.e., feet or kilometers or some other metric)? Furthermore, once a metric is chosen, the user must convert some (if not all) of the values to a single metric unit. The implications of the decision may not be critical if the database is only single purpose and is not being installed for general availability. However, if the original user or another potential user may be interested in the same database in the future, providing a standard methodology for addressing database integration issues will improve productivity for future users. Standard data elements address this problem by establishing a standard name (independent of any database or data table) for a particular concept. Aliases are used to identify the name and usage of the same concept in each specific database. With standard data elements and aliases, a user deriving a new database is provided with guidelines and standards for naming new columns. For future users who want to peruse or retrieve data about a particular concept, say, runway length, a repository for standard data elements and aliases serves as a reference to all databases which represent runway length.

**Related work**

In this section we first describe different methodologies adopted by two dictionary system approaches: DBMS system catalogs and information resource dictionary systems. We also discuss specific data dictionary and data modeling efforts with functionality similar to RMMS.

Most DBMSs include a component called a data dictionary or system catalog. For relational DBMSs, this data dictionary maintains system tables containing information about database tables and columns. The information stored in this repository is generated at schema definition time and includes those database characteristics which are specified in the data definition language. These characteristics (for a table) include data such as who owns the table, when the table was created, and what storage structures are used for indices. Column metadata includes information such as the datatype and length of the column value and whether nulls or defaults are permissible. Much other system information is recorded in this data dictionary; however, most of this information is only relevant for the operation and optimization of the DBMS. These facilities cannot, for example, be used to store the units of a measurement value or any semantic information such as the relationship between a key attribute in one table and a foreign key in another. RMMS serves a different role from DBMS system catalogs and data dictionaries by managing semantic information required by users and applications for accessing and manipulating data in a manner which is semantically correct and achieves the intention of the query.

Information Resource Dictionary Systems (IRDS) have also been the subject of considerable research (Dolk, 1987; Goldfine, 1985; Jones, 1991; Kossman, 1987; Navathe and Kershberg, 1986). The scope of IRDS embodies the major activities, processes, information flows, organizational constraints, and concepts of an “enterprise model”. In the past IRDS were considered primarily as a design tool for information modeling and database design. “Active” data dictionaries were used only during batch DBMS operations or real-time transaction processing. Goldfine and Konig (1988) present IRDS specifications according to the National Institute of Standards and Technology (NIST). This standard describes a kernel set of basic data dictionary capabilities plus a collection of independent optional modules. So far, three additional modules have been specified dealing with security, application program interface, and documentation. The emphasis on program interfaces and the neglect of interactive tools are evident in the specification. Until recently, facilities provided by an IRDS were sufficient for information management in static business and financial applications. However, IRDS are not amenable to scientific applications which require ad hoc database manipulation. Furthermore, no IRDS implementations or tools currently conform with the NIST IRDS standards.

In the remainder of this section we will discuss four other data administrative efforts which are developing semantic data dictionaries for metadata management. First, the Center for Information Management within the Defense Information Systems Agency (DISA/CIM) has initiated an effort to catalog all Department of Defense (DoD) data elements (DoD, 1992). The Defense Data Repository System (DDRS) maintains standard data elements used throughout both DoD business applications and DoD Command, Control, Communications, and Intelligence (C3I) applications. Standard data elements in the DDRS are being derived from a top-down DoD data model identifying major entities and attributes of those entities. Although the DDRS is not related to a specific database or DBMS, it will serve as the standard set of DoD data elements for naming and interoperating among DoD information systems. A new DoD effort is approaching standard data elements from the bottom-up. The Joint Data Base Elements (JDBE) project sponsored by the Defense Modeling and Simulation Office is conducting reverse engineering efforts for existing subject area simulation models and databases to identify the subject area data requirements. Based on these requirements JDBE staff will produce an IDEF1X data model (Bruce, 1991). Data models from many different applications in common subject areas will be merged to identify common data entities and attributes. From this set of common data elements will emerge a standard data dictionary. In a third project, the US Army has implemented the Training and Doctrine Automated Data System (TADS) which supports the transformation of external databases into the format needed for Army
combat models. TADS requests data from about 20 suppliers and enforces standard nomenclature, standard output data files, and standard transformation processes. The Army’s modeling and simulation community requests data from TADS in specific formats required by approximately 12 existing models. The final system we present is the Operations Analysis and Simulation Interface System (OASIS) developed by the US Joint Chiefs of Staff. The OASIS mission is to develop a system which will significantly improve data collection, access, verification, analysis, reporting, and documentation for joint studies and analysis processes. OASIS is most similar to RMMS because it uses a centralized relational DBMS for its data element dictionary. OASIS contains an on-line dynamic data dictionary based on an entity-relationship model and uses an interactive window system for accessing the data dictionary.

RMMS Functionality

The primary objective of RMMS is to streamline the sharing, reuse, and interoperability of relational databases. In particular, we have designed RMMS to respond to the five needs identified in section 3. In the following section, we discuss how RMMS serves these needs. Although other data models (such as semantic and object-oriented) and other information media (such as text, images, and voice) also require metadata, we initially limited our scope to metadata for relational databases.

Documentation

RMMS provides a standard methodology for documenting relational databases. This capability supports a uniform representation for documentation yet allows enough flexibility to represent unstructured textual comments, such as “the values of this column were derived manually using background knowledge and human judgment”. Simply knowing how such data was derived (even though it was not derived in a rigorous fashion) is nevertheless helpful to a subsequent user. Documentation is stored for many different categories of database entity types, for example, database, table, and column entities all have associated documentation. Inter and intra-table relationships between columns are also identified and documented. Domain information, such as allowable values for a column, are maintained along with abbreviations and acronyms. Version information, historical data, and statistics reporting on the frequency of values all serve as documentation which helps a user decide what data to select.

Because of the structure imposed on database documentation in RMMS, it is appropriate not only as human-readable documentation, but also as machine-readable data specifications (e.g., translation routines for converting units) which further contributes to maintaining data consistency. All metadata in RMMS is considered documentation. Some of it is more relevant to interactive users; other metadata facilitates the four capabilities discussed below.

Version management

In RMMS, version management refers to the tracking of information such as the availability, source, and schema of different versions. RMMS is designed to accommodate two version tracks: an external version track and an internal version track. Since many of RAND’s databases are acquired from outside agencies, it is critical to be able to identify a database by its external (agency) version number. However, as a user of the data, RAND does not always acquire or install every newly released agency version. Furthermore, RAND’s data administrator may make changes to the data and internally produce a new version which is derived from the version which the external agency distributes. Therefore, internal version tracking is also necessary. The RAND internal version track is represented as a version number of the form p.s where p is the primary version number and s is the secondary version number. When a new agency version arrives, the p portion of the version number is incremented by 1 and the s portion is set to 0. When RAND data administrators make any changes to the data, the s portion of the version number is incremented by 1. Correspondence between the agency’s assigned version number and RAND’s version number is also maintained as part of version management metadata.

History management

History recording facilities maintain actual data values of current and old versions, both external and internal. Many approaches for maintaining historical data are practiced (Adiba & Quang, 1986; Dadam, Lum & Werner, 1984; Segev & Shoshani, 1987; Snodgrass & Ahn, 1986). The simplest, yet most costly in terms of resources, is to maintain complete copies of all versions in their own independent databases. Clearly, this is not a pragmatic solution. Other options suggest archiving complete copies or maintaining only the updates or changes to each version, rather than complete copies of each version. A disadvantage of maintaining only the updates is that whenever a user needs the current version, he or she must apply a series of updates to a baseline version. Because RAND users most often want to retrieve data from the most recent version, maintaining only updates is also not a practical alternative. The methodology we developed for RMMS maintains a history of all changes to a database by recording the previous value of a data item (before it has been updated by a new version) in a metadata table similar in structure to the table containing the modified data. These “value history” tables store old values from data tables that have been subsequently updated. Corresponding history metadata tables for recording changes to a table as a
whole (such as changing the name of a data table) and changes to the schema (such as changing the column length or data type of a column) are stored in “table history” and “column history” metadata tables.

Derived databases
A derived database is one that is generated internally from one or more external databases. Throughout the course of an analysis or study, researchers will produce many database variations derived from baseline data. In many cases, a user wishes to “permanently” store and maintain a derived database for future use. Facilities in RMMS support the derivation process by maintaining (1) specialized metadata relevant to derived databases, (2) procedures to automatically populate derived database metadata from external database metadata, and (3) a trigger mechanism for updating derived databases when external source databases have been updated.

If metadata for the source database is maintained in RMMS, then a subset of this metadata can be applied as metadata for the derived data. If the source is not a registered database or has no associated metadata, the user generating the derived database will be queried for relevant metadata. A derived database will contain an “audit trail” indicating the source of the data, the source’s version, a timestamp recording the date of derivation, and other supplemental metadata such as procedures used for transformations.

Linkages are maintained between derived databases and their respective source databases. In theory, when a new version of an external database is installed at RAND (or the current version is updated), a new version of the corresponding derived databases can be generated. In practice, however, this process is possible only if the derived database was produced automatically from external databases. Furthermore, it may not be desirable to automatically regenerate the derived databases. Instead, when an external database is updated or a new version installed, RMMS can determine which derived databases are affected and set a flag in the metadata of each derived database. This “trigger” mechanism will inform the user of a database when it needs to be “re-derived” from updated sources databases.

Standard data elements
Standard data elements are necessary to support interoperability among independent applications in an enterprise. Standard data elements are intended to represent those data values which are considered atomic and are not decomposable. For example, “employee last name” and “length of runway” are atomic data elements because they represent primitive elements which cannot be further decomposed. Many organizations are addressing the development of standard data element dictionaries for maintaining metadata and domain information relevant to data elements.

Metadata supporting standard data elements not only maintains a standard nomenclature for a given concept, but also prescribes standard units for measurement attributes. Procedures for converting units from one metric to another is part of the procedural component of a standard data element repository. When deriving a new database using the RMMS standard data element facilities, a user would apply the provided procedures to convert values to the standard metric. Note that the use of standard data elements does not require or suggest any changes to names or units in the original databases. Instead, this methodology provides a standard liaison between conceptually identical data elements among independent databases.

With the use of standard data elements, aliases, and conversion procedures, two objectives are achieved: (1) a user deriving a new database is provided with standards for how to name the new columns and with libraries of procedures to convert units, and (2) future users who want to peruse or retrieve data about a particular concept, say, runway length, can use a standard data element and its aliases as a pointer to all databases that represent runway length. This second benefit contributes to increased reusability and comprehension of the contents of application databases.

Maintaining data elements that are conceptually the same (and maybe even spelled the same) but which differ in other ways, such as format or granularity is an issue that remains to be addressed in RMMS. For example, one database may store the month and year of an event, but another database may store the month, year, day, and time. Facilities for comparing, translating, and combining data such as these must also be provided by standard data element facilities.

RMMS Architecture
RMMS manages metadata for relational databases that are maintained in the Ingres DBMS. Therefore, we designed RMMS as a set of Ingres relational tables that extend the application databases. RMMS is composed of two major components. One component, the “Data Encyclopedia” is a database of metadata tables that maintains general information about all application databases. The other component, called the “Data Dictionary”, is a set of tables that augments the tables for each application database and contains metadata about specific entities and attributes of a database. Although a single Data Encyclopedia database exists for an entire enterprise, each application database includes its own set of Data Dictionary tables. Figure 1 illustrates the two components and their relationship to application databases. In this section we describe the design architecture, implementation status, and user interaction supported by RMMS.

The Data Encyclopedia database, named “RMMS”, reflects the entire configuration of application databases and their versions, standard data elements and aliases, derived
databases, and shared computer procedures, such as unit conversions. This database contains metadata information that applies across all application databases. The tables contained in the Data Encyclopedia or “RMMS” database are shown in Figure 1 as a separate independent Ingres database. The tables named database and version_history contain information on specific databases; the tables standard_data_element, sde_aliases, standard_domains, sde_numeric_domain_range, and value_<sde_domain_name> represent standard data elements and corresponding domains and their values (both character and numeric). The tables named values_<sde_domain_name> connote one table for each standard domain where each table enumerates allowable values for that domain. Appendix A lists each of these tables and their corresponding column names.

As we discussed earlier, Ingres internally maintains its own “system” tables within each database which contain minimal metadata necessary for Ingres operation. However, Ingres system tables are not extensible and are not intended for storing semantic information about the database. Therefore RMMS supports additional metadata tables, called the “Data Dictionary” which augment the Ingres system tables. Data Dictionary metadata relates to one specific application database and is stored as metadata tables in the Ingres database containing the application tables. One set of Data Dictionary metadata tables exists for each Ingres application database. These tables are prefixed with “md_” so that users can distinguish those tables which contains RMMS metadata from those tables in the application database which contain actual data values. We use the terminology “metadata tables” to refer to tables which are part of the RMMS Data Dictionary system and contain definitions, descriptions, and information about specific data tables. In Figure 1, Data Dictionary metadata tables are shown with each application database.

Data Dictionary metadata extends the scope of the Ingres system tables. We use the suffix “_extend” as part of the Data Dictionary table name, e.g., md_table_extend and md_column_extend contain detailed information about application tables and columns. The tables md_dependencies and md_links identify inter and intra-column dependencies and join fields. The history of data values is maintained in the tables md_table_history, md_column_history, and md_value_history_<data table name>. Domain metadata is stored in md_value_enum and md_numeric_domain_range. Finally, statistics about value distributions is maintained in md_range_statistics and md_enum_statistics. A list of each md_Dictionary table and column can be found in Appendix B.

The implementation of RMMS is an on-going activity. Schemas for the Data Encyclopedia tables and Data Dictionary metadata tables have been developed and installed. However, acquisition of metadata to populate the Data Encyclopedia and Data Dictionary is a continuing task. RAND’s data administrators are strongly recommending that newly registered databases should be loaded into Ingres with metadata. Often however, all metadata is not available or projects do not want to assume the cost of developing metadata. Therefore, for many databases, the maintenance of Data Dictionary tables occurs over time. The initial user interface was developed in C using the OpenWindows Developer’s Guide (Devguide) interface system (OpenWindows, 1990). Future plans are to explore the use of INGRES Windows 4GL as the interactive windowing environment. Although RMMS is implemented in the INGRES DBMS, its schema can be ported to any relational system. Currently, users rely on RMMS for documentation, browsing, retrieving subsets, formatting subsets, and verifying data values. The statistics metadata has also been judged as very useful for determining the profiles of data samples.

**Conclusions**

RMMS responds to many significant issues faced by scientific database users for integrating, sharing, and reusing application databases. In this final section, we present one issue which has not been fully addressed by RMMS, namely, complex data types. We conclude with a brief discussion of another potential RMMS application.

One future RMMS goal is to develop an approach for modeling non-atomic or complex data as standard data elements. Complex data includes data elements that are derived, composed, or computed from other standard data elements, such as lists, repeating groups, matrices, probability distributions, and abstract data structures. Traditionally, standard data elements were atomic for 2 reasons: First, DBMS and especially relational DBMS, required a normalized representation of data in which each data slot was atomic. For example, in a relational database, an entry in a table should be a single atomic value, not a list or a repeating group or a concatenation of strings. Furthermore, the data type of the atomic value was limited to alphabetic or numeric. Second, the practice of data modeling and data administration regarded basic elements of information as atomic and non-decomposable. Conversely, if a piece of data was decomposable, then the data (in its composite form) could not be considered a standard basic element of data. Advanced data modeling techniques along with extended relational and object-oriented DBMS have made these rules obsolete. Complex data in the form of abstract data types and object data types are referenced and used in their composite form. Currently, standard data elements in the RMMS Data Encyclopedia are atomic elements. However, to promote better data modeling and interoperability in applications, RMMS must support complex data elements as both atomic elements and as non-atomic decomposable elements. In addition,
RMMS must also maintain the relationship between the whole and its parts for verifying, integrating, and deriving application databases containing complex data elements.

The development of RMMS was motivated (1) by the proliferation of databases stored and used at RAND and (2) by a realization that maintaining metadata is at least as important as maintaining the actual data values. Each of the five issues addressed by RMMS responds to a limitation faced by users of RAND’s relational databases and simulation models that require input from those databases. The benefits of RMMS have been particularly significant for database users in applications which require ad hoc access and integration from multiple databases. Although the RMMS prototype implementation is incomplete and continually evolving, it is both often used and well liked.

Until now, RMMS has been used to manage metadata for military databases which serve as input to simulation models. We have also proposed to use RMMS in environmental engineering applications. In these applications the goal is to improve the management, integration and analysis of environmental datasets collected for different purposes, at different levels of aggregation, by different agencies, and over different periods of time. In this environment, determining the availability, accessibility and location of critical data is a key challenge. RMMS will serve a major role in the maintenance of metadata necessary to meet these goals.

Appendix A. RMMS Data Encyclopedia Tables

The names of the tables and the corresponding columns that comprise the Data Encyclopedia database are listed below. The “RMMS” database, shown at the top of Figure 1, contains these tables.

```plaintext
database metadata table
  db_name
  current_status
  source_information
  current_rand_version
  checkpt_version
  valid_users
  documentation
  description

version_history metadata table
  db_name
  agency_version
  agency_date
  rand_version
  rand_date

standard_data_element metadata table
  sde_name
  sde_long_name
```

```plaintext
sde_units
sde_domain_name
sde_source
sde_composite_datatype_flag
sde_description

sde_aliases metadata table
  sde_name
  db_name
table_name
  column_name

standard_domains metadata table
  sde_domain_name
  sde_name
  sde_domain_category
  values_table_name
  sde_valid_pattern
  sde_domain_description

sde_numeric_domain_range metadata table
  sde_domain_name
  sde_min
  sde_max

values_<sde_domain_name> metadata table
  standard_value
  expansion
  description
```

Appendix B. RMMS Data Dictionary Tables

The names of the tables and the corresponding columns that comprise the Data Dictionary are listed below. One set of Data Dictionary tables exists for each Ingres database. These tables are stored in the same database as the data tables and are prefixed with “md_” so that users can distinguish those tables which contain RMMS metadata from those applications data tables which contain column values. These tables are shown in the middle of Figure 1 within “application database 1”.

```plaintext
md_table_extend metadata table
  table_name
  unique_id
  table_source
  description

md_column_extend metadata table
  table_name
  column_name
  column_source
  column_units
  units_translation
  init_rand_version
  composite_datatype_flag
  dependency_flag
  column_description
```

```plaintext
md_table_extends metadata table
  table_name
  unique_id
  table_source
  description

md_column_extends metadata table
  table_name
  column_name
  column_source
  column_units
  units_translation
  init_rand_version
  composite_datatype_flag
  dependency_flag
  column_description
```
Figure 1: RMMS System Architecture
sde_name
valid_pattern
sub_domain_class_name
sub_domain_category
sub_domain_class_description

md_dependencies metadata table
dependent_table
dependent_column
independent_table
independent_column
description

md_links metadata table
table_name1
column_list1
tables_name2
column_list2
cardinality

md_table_history metadata table
init_rand_version
time_stamp
table_change_type
table_change_source
table_change_authorization
table_change_processor
table_change_description

md_column_history metadata table
init_rand_version
time_stamp
column_change_type
column_change_source
column_change_authorization
column_change_processor
column_change_description

md_value_history_<data table name> metadata table
init_rand_version
time_stamp
value_change_type
value_change_source
value_change_authorization
value_change_processor
value_change_description

md_value_enum metadata table
sub_domain_class_name
column_value
standard_value
description

md_numeric_domain_range metadata table
sub_domain_name
sub_domain_min
sub_domain_max

md_range_statistics metadata table
table_name
column_name
minimum
maximum
average
median
standard_deviation
null_count
sum
percentile_1st
percentile_99th
quartile_25th
quartile_75th

md_enum_statistics metadata table
table_name
column_name
unique_value
count
percent

References


Bruce, T., Designing Quality Databases with IDEF1X Information Models, 1991.


Stephanie Cammarata, Ph.D., is a senior computer scientist at RAND Corporation in Santa Monica CA. She is currently an associate editor for the Journal of Database Management and has also served on the ANSI Object-Oriented Database Task Group. She has conducted information science and data management research in the areas of data dictionary systems, metadata management, data element standardization, object-oriented and semantic data modeling, and heterogeneous database systems. Her publications have appeared in the Journal of Object-Oriented Programming, and in the Proceedings of the ACM SIGMOD Conference, Very Large Databases Conference, and the International Joint Conference on Artificial Intelligence.

Iris Kameny is an Associate Director of the Applied Science and Technology program at RAND. She is currently supporting the Defense Modeling and Simulation Office in addressing data issues related to modeling and simulation and is a member of the C3I panel of the Army Science Board and a past member of the National Research Council Panel on Space Technology. She has over 30 years of experience in the areas of distributed and advanced database systems, knowledge based systems, computer security, simulation, natural language processing, speech understanding and information retrieval systems. Her publications have appeared in the National Academy Press, International Conference on Data Engineering, Winter Simulation Conference, and Conference on Very Large Databases.

Judy Lender, M.S., Computer Science, is a senior programmer analyst at RAND. Among her interests are object-oriented databases and data dictionary systems. She has worked in relational database management system product development, specifically semantic processing, at Sperry in Minnesota. She is currently a database manager of classified databases at RAND, and a database administrator on several RAND database systems.

Corrine Replogle, B.S., worked as a senior programmer/analyst for five years at RAND. During that time, she served as a database applications and systems analyst/programmer, where she was responsible for the implementation of the RMMS system. She also worked in software validation for the Raytheon Company in Bedford, MA. She is currently at the Pennsylvania State University School of Medicine in Hershey.
INDUSTRY NEWSBRIEFS

This section includes information collected from several sources, including press releases. The information was chosen by the Journal of Database Management editorial staff for its interest to our readers. Inclusion does not imply endorsement of the products or services by either the Journal of Database Management or Idea Group Publishing.

New Version of Database Design Tool Now Out Under Windows

The Canonizer, a database design tool from Six Sigma CASE, Inc., has just been released for the Microsoft Windows environment. Retailing for $1995, version 2.0.3 replaces the earlier character-based Canonizer and includes on-line support.

The new release also features ARD (Attribute Relationship Diagram) modeling instead of traditional entity relationship diagramming. ARD modeling is easier for a database designer to use because it accepts information and queries as the database is built. The designer does not need to know a methodology before beginning to build, which would be required with ERD modeling.

The Canonizer handles automatic normalization to Third Normal Form, produces a variety of reports and includes Postscript capability. The CASE tool is used to design new database models, re-engineer existing databases, generate code directly from design, produce design documentation and migrate database models across DBMSs. For more information, contact Six Sigma CASE Inc., 800/827-4462.

New Database Software Introduced for the Portable Newton MessagePad

The portable technology that inspired beaming, tapping and handwritten input will soon have powerful new database software for increased functionality. Code-name proFile, the first general-purpose flat-file database developed for the Newton™ MessagePad™ is due to hit the market in early 1994. Introduced by HealthCare Communications at the the recent Newton Platform Development Conference in Santa Clara, CA, proFile allows Newton users to create and modify databases customized to their business and personal needs.

According to HealthCare Communications, proFile will be bundled with Macintosh and Windows applications capable of direct data exchange. For more information, contact HealthCare, 402/489-0391.

English-Language Design Software for Database Developers Released

SerraCorp, an Ohio-based developer of natural-language database software, will release its English-based design tool, The Database Designer, on May 1, 1994. The Database Designer allows consultants to quickly design and automatically program normalized relational database prototypes using nothing more than English sentences. The Database Designer is a computer-aided software engineering tool that uses the Language-oriented Data Modeling (LODM) approach to database design pioneered by SerraCorp. LODM allows developers to automatically extract the logical schema embedded within a group of English sentences which describe a client’s data management needs. The LODM approach accomplishes this without requiring that consultants have sophisticated understanding of relational theory (e.g., of primary and foreign keys, referential integrity, and cardinality). For more information, contact SerraCorp, 216/677-1931.

Tattooed ID Plate with New Software Package Available

A patented indelible marking and registry system, with new database software called Asset Tracking and Management Program, is now available from S.T.O.P. (Security Tracking of Office Property) The system has reduced theft/loss rates by up to 97 percent.

“In our two and a half years, more than 50,000 PCs, other computers, and various office and lab equipment have been marked and registered—with only 10 reported incidents of theft,” said Berner Chesnutt, Director of Sales. “Now with our new software, companies will be able more easily to keep track of where their equipment is located.”

The Asset Tracking & Management Program is an inventory database which gives the user an overall picture of their equipment, its location, and user information. For more information, contact S.T.O.P., 201/359-9361.
Book Review

Review by Geoffrey Steinberg
Kent State University

Intelligent Database Tools & Applications
Kamran Parsaye and Mark Chignell
ISBN 0-471-57066-4

This is an exciting book from the start. The authors provide a wide and deep exploration of their interesting and forward looking ideas in this important area. Their thesis involves the “combination of various technologies to extend large databases with flexible features that compliment the user’s intellect.” In an earlier book the authors described the fundamental concepts of intelligent databases and in this book they focus on the application of new techniques to the implementation of intelligent databases.

The concept of intelligent databases is probably new to many people. It involves the integration of various software tools and applications to provide users with intelligent access to data. Integrating software products is not new to practitioners, but the underlying framework of what works best and why it does is generally now known. This book addresses that gap. Application developers will gain insights into what techniques work in harmony with others and why. They will learn about techniques, many of which are new, and will realize that integration of these techniques is easier than expected.

The authors stress that new techniques in and of themselves are not the solution. Rather, they suggest the importance of dialogue between domain specialists and computer professionals. That dialogue is crucial for the implementation of intelligent database applications.

Intelligent database applications are tools that assist the user in developing an understanding about the data they work with. As the volume of data that individuals depend on increases, the need for intelligent applications become more profound. Information discovery and data visualization tools become very important aids in providing users with insight into the information contained in their databases. The same set of underlying data may provide different types of information to different users. Intelligent database applications integrate diverse sources of information, create appropriate visualizations, and provide access to them within a navigable interface.

The authors correctly suggest that the input process in database management systems is well understood but that little emphasis is generally placed on the display or interpretation of data. The role of intelligent databases is to provide the user with tools that extend human reasoning about the data contained in large databases.

Major emphasis is placed on the use of hypertext for navigation and iconic interface for query entry. The importance of presentation is not overlooked. Information presentation is described as the activity that coordinates the assumptions of individuals about data. Treatment is given to spatial context, graphics, color coding and automatic presentation construction. Extensive examples make the understanding of their concepts easy.

The major contribution of this book is that of encouraging the reader to think creatively about computing. Traditional database methods and concepts are replaced with discussion about the seamless integration of new technologies and processing concepts: information discovery, data visualization, hyperinformation and hyperdata. In depth attention is given to the user interface including graphical techniques, information presentation, executive information systems and project management.

The book’s organization is thoughtful. The chapters are presented in two main thematic groupings: theory and application. The authors develop their thesis from the origins of database theory and graphical interface technology. After an introduction about intelligent databases and a general introduction to the topics covered in the book, they provide a deep discussion about graphical user interfaces. The most significant coverage is given to information discovery (rule generation and anomaly detection), data visualization (allowing the user to see patterns in complex data) and hyperinformation and hyperdata (nonlinear structuring of data). The section on information presentation provides a unifying conclusion to those topics. The remainder of the book covers applications with chapters on executive information systems, project management systems, marketing systems and quality control systems. The authors provide adequate background for the novice without suppressing the thirst of the experienced professionals.

Much of the examples presented in the book rely on software that is the product of the company of one of the authors. It is not clear from the discussion how widespread the presented techniques are in reality. Are there other vendors who supply the components that the authors discuss? This comment does not discredit the value of the concepts developed in this book rather it suggests that the time is ripe for others to jump into this market niche.
Related Content

Semantic Multigranularity Locking for Object-Oriented Database Management Systems
[www.irma-international.org/article/semantic-multigranularity-locking-object-oriented/51178/](www.irma-international.org/article/semantic-multigranularity-locking-object-oriented/51178/)

Mobile Commerce: Promises, Challenges and Research Agenda
Keng Siau, Ee-Peng Lim and Zixing Shen (2001). *Journal of Database Management* (pp. 4-13).
[www.irma-international.org/article/mobile-commerce-promises-challenges-research/3264/](www.irma-international.org/article/mobile-commerce-promises-challenges-research/3264/)

Multilevel Databases
[www.irma-international.org/chapter/multilevel-databases/11177/](www.irma-international.org/chapter/multilevel-databases/11177/)

[www.irma-international.org/article/rich-base-schema/3247/](www.irma-international.org/article/rich-base-schema/3247/)

BACIIS: Biological and Chemical Information Integration System
[www.irma-international.org/article/baciis-biological-chemical-information-integration/3338/](www.irma-international.org/article/baciis-biological-chemical-information-integration/3338/)