RAND

The RAND Metadata Management System (RMMS)

A Metadata Storage Facility to Support Data Interoperability, Reuse, and Sharing

Stephanie Cammarata, Iris Kameny, Judy Lender, Corinne Replogle

National Defense Research Institute
Arroyo Center
Project AIR FORCE

The research described in this report was conducted in RAND's three federally funded research and development centers: The National Defense Research Institute, sponsored by the Office of the Secretary of Defense and the Joint Staff; the Arroyo Center, sponsored by United States Army; and Project AIR FORCE, sponsored by the United States Air Force.

Library of Congress Cataloging in Publication Data

The RAND Metadata Management System (RMMS) : a metadata storage facility to support data interoperability, reuse, and sharing / Stephanie Cammarata ... [et al.].
p. cm.
"Prepared for the Office of the Secretary of Defense, The United States Army, and the United States Air Force."
"MR-163-OSD/A/AF."
Includes bibliographical references.
ISBN 0-8330-1515-X
1. Database management. 2. Database design. 3. Ingres (Computer file). I. Cammarata, Stephanie. II. United States. Dept. of Defense. Office of the Secretary of Defense. III. United States. Army. IV. United States. Air Force.
QA76.9.D3R247 1994
005.74´2—dc20 94-6663
CIP

Published 1995 by RAND
1700 Main Street, P.O. Box 2138, Santa Monica, CA 90407-2138
RAND URL: http://www.rand.org/
To order RAND documents or to obtain additional information, contact Distribution Services: Telephone: (310) 451-7002; Fax: (310) 451-6915; Internet: order@rand.org

RAND

The RAND Metadata Management System (RMMS)

A Metadata Storage Facility to Support Data Interoperability, Reuse, and Sharing

Stephanie Cammarata, Iris Kameny, Judy Lender, Corinne Replogle

Prepared for the
Office of the Secretary of Defense
United States Army
United States Air Force

National Defense Research Institute
Arroyo Center
Project AIR FORCE

Preface

This report discusses the results from a set of meetings held to design a repository system that stores descriptional and definitional information about databases (maintained in INGRES), models, and procedures supported by RAND's Military Operations Simulation Facility (MOSF). The resulting design, the RAND Metadata Management System (RMMS), is composed of its own INGRES database tables, called the "Data Encyclopedia" tables, which collectively maintain data about general characteristics of each of the individual INGRES databases. In addition, there are a set of INGRES tables, called the "Data Dictionary" tables, which augment each separate INGRES database and provide more detailed information about the contents of the database. This report describes the RMMS design. Implementation of the RMMS (based on this design) is an ongoing activity conducted by MOSF data and system administrators.

This research support activity was sponsored through RAND's three federally funded research and development centers—the National Defense Research Institute (sponsored by the Office of the Secretary of Defense, the Joint Staff, and the defense agencies), the Arroyo Center (sponsored by the U.S. Army), and Project AIR FORCE (sponsored by the U.S. Air Force). This document will be of interest both to users of the MOSF databases and to other relational database users who would like to develop an RMMS-based metadata repository for their set of databases.

Contents

Summary

The RAND Metadata Management System (RMMS) is a system that manages "metadata." Metadata denotes definitional and descriptional information about databases, simulation models, and procedures. Databases, such as those maintained in the INGRES database management system (DBMS) by the Military Operations Simulation Facility (MOSF), are prevalent throughout RAND. Similarly, many prominent simulation models are exercised regularly in the MOSF and require input data extracted from INGRES databases. However, most of these databases have little documentation or other descriptional information to go along with them. The absence of such information leaves users at a loss for understanding the definitions, abbreviations, acronyms, and descriptions of the pieces of data stored and maintained in a DBMS.

This report presents the design of the RMMS, a metadata management system for relational databases. Our goal was to produce a design document that could be used both by the users of MOSF relational databases (who may be future users of RMMS) and by users of other relational databases who would like to develop a similar RMMS metadata repository for their own set of databases. This work was motivated by the proliferation of databases stored and used in the MOSF and by a realization by the MOSF database management staff that such metadata is at least as important as, if not more important than, the actual data values. 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 report.

We have addressed five major issues or "needs" during the development of RMMS:

- The need to provide complete, thorough, and standard database documentation;
- The need to record and manage information about different versions of each database;
- The need to maintain a history of the changes made to database tables, schema, or data values;

- The need to facilitate derived databases for input to simulation models and to share among models; and
- 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.

Each of these issues responds to a limitation faced by users of the MOSF relational databases and simulation models that require input from those databases. The design of RMMS supports these requirements.

RMMS has two major components. One component is a database of metadata tables referred to as the Data Encyclopedia. In the RMMS, the database name given to the Data Encyclopedia metadata tables is "rmms". The information contained in these tables reflects the entire configuration of MOSF's databases and their versions, standard data elements and aliases, derived databases, simulation models that make use of the databases, and shared computer procedures, such as unit conversions and database derivations. General information about a particular database is recorded in the Data Encyclopedia.

The second component of RMMS is a set of metadata tables that augment the database of each external and derived database that is recorded in the Data Encyclopedia. This set of metadata tables is referred to as the Data Dictionary metadata tables. Each MOSF relational database will contain a corresponding set of Data Dictionary metadata tables. These tables record metadata about specific tables and columns in the corresponding data tables. Although other data models (such as semantic and object-oriented) and other information media (such as text, images, and voice) also require metadata, we limited our scope to metadata for relational databases.

The implementation of RMMS is an ongoing activity. Schemas for the Data Encyclopedia tables and Data Dictionary metadata tables have been developed and installed. However, acquiring metadata to populate the Data Encyclopedia and Data Dictionary is a continuing task. The initial user interface was developed in the C programming language 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.

The benefits of RMMS have been particularly significant for database users in applications that require ad hoc access and integration from multiple databases. Currently, users rely on RMMS for documentation, browsing, retrieving subsets,

formatting subsets, and verifying data values. The statistics metadata has also been judged very useful for determining the profiles of data samples. Although the RMMS prototype implementation is incomplete and continually evolving, it is both often used and well liked.

1. Introduction

Electronic databases and database management systems (DBMSs) are becoming commonplace in scientific and research communities. However, when scientific databases are installed in a DBMS, electronic documentation or other descriptional 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 originated, 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 data elements. 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 that is being designed. Historically, database metadata was not needed by business and financial enterprises because they relied on databases and DBMSs 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 report describes the RAND Metadata Management System (RMMS) whose function is the storage and maintenance of metadata to improve five major aspects of DBMS data administration: 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 database maintenance, in this document we focus primarily on the metadata storage structures within RMMS. Detailed discussion of user interface and database maintenance designs is beyond the scope of this report.

At RAND, RMMS maintains metadata about military databases, including data on 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.

In the next section we present examples of the use of metadata in different application areas. Section 3 cites specific problematic issues that RMMS is addressing and Section 4 discusses related work. In Section 5 we identify five categories of functionality supported by RMMS. The software architecture of RMMS is described in Section 6. We conclude with a discussion of future research issues for RMMS extensions. Appendices A and B contain detailed design specifications for the implementation of RMMS.

2. 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 explain 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 of data because through the years different data 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 contaminants across two such databases requires knowledge of the representations and conversion procedures each uses.
- 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 resource for browsing, making it possible to identify, for example, 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 that 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 various databases.

3. Issues

In this section we identify five specific issues or needs that 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 nonexistent. Some databases are delivered with hardcopy manuals or codebooks, but most often these must be acquired independently from 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 contents is frequently passed through word of mouth from experienced users to novices. This situation has many drawbacks leading to inconsistent and inaccurate applications of data. First, since precise and definitive semantics governing attributes are 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 databases. Administrators of scientific databases must manage different versions of databases that originate from a variety of sources. Many studies require databases that are acquired from outside agencies that release new versions regularly, e.g., weekly, monthly, yearly. For databases that are generated by a tightly coupled data collection activity such as the output of an experiment or simulation, the elapsed time between versions may be only 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 reapply to the new version corrections that 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 provided by a metadata management system. This capability will benefit those applications that require a complete audit trail (at the logical level) of all additions to, modifications of, and deletions from 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 to, modifications of, and deletions from either data values or the schema. For example, the addition of a new table to a new version should be reflected explicitly in the version management system along with 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 versions of a database. Finally, we needed to ensure 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, it acquires databases from external agencies and derives 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 nonderived databases, (4) applying transformations to subsets of data from nonderived databases, and (5) integrating subsets of data collected by methods (1) through (4). For the remainder of this paper, we refer to nonderived 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 external databases. 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 named differently or (2) named the same but conceptually different. This goal 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 U.S. commercial airfields and another representing Canadian airfields. Suppose the column name representing runway length in the U.S. 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 U.S. airfields is recorded in feet and that the length of runways on Canadian airfields is stored in kilometers. The user is faced with two decisions about the derived database: (1) what to name the column for runway length (i.e., "runwayln" or "rwlength" or something else); and (2) what units of measurement to use in 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 many 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 that represent runway length.

4. Related Work

In this section we first describe different methodologies adopted by two dictionary system approaches: DBMS 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 that 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 that is semantically correct and that achieves the intention of the query.

Information Resource Dictionary Systems (IRDSs) have also been the subject of considerable research (Dolk, 1987; Goldfine, 1985; Jones, 1991; Kossman, 1987; Navathe and Kerschberg, 1986). The scope of IRDS embodies the major activities, processes, information flows, organizational constraints, and concepts of an "enterprise model." In the past IRDSs 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, IRDSs are not amenable to scientific applications that 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 that 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 (Department of Defense, 1992). The Defense Data Repository System (DDRS) maintains standard data elements used throughout both DoD business applications and DoD command, control, communications, and intelligence 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 operating 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 (DMSO) 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 U.S. 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 U.S. Joint Chiefs of Staff. The OASIS mission is to develop a system that 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.

5. 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 were derived (even though they were 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 that 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 that 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 increased 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 increased 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 and Quang, 1986; Dadam, Lum, and Werner, 1984; Segev and Shoshani, 1987; Snodgrass and Ahn, 1986). The simplest approach, 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 include 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 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.

Database Derivation

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.

Data Element Standardization

Standard data elements are necessary to support interoperability among independent applications in an enterprise. Standard data elements are intended to represent those data values that 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 that 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 are 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, an "alias" is used to express the relationship between a standard data element name and a database name for the same concept. Therefore, 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 naming the new columns and with libraries of procedures for converting 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 comprehensibility 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 resolution, 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.

6. 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 has 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. Table 1 lists the RMMS tables illustrated in Figure 1. In this section, we describe the design architecture, implementation status, and user interaction supported by RMMS. (In Table 1 and Figure 1, "sde" stands for "standard data element," and "md" stands for "metadata.")

Table 1

RMMS Tables

RMMS Data Encyclopedia tables
database
version_history
standard_data_element
sde_aliases
standard_domains
sde_numeric_domain_range
values_<sde_domain_name>
RMMS Data Dictionary tables
md_table_extend
md_column_extend
md_dependencies
md_links
md_table_history
md_column_history
md_value_history_<table name>
md_value_enum
md_numeric_domain_range
md_range_statistics
md_enum_statistics

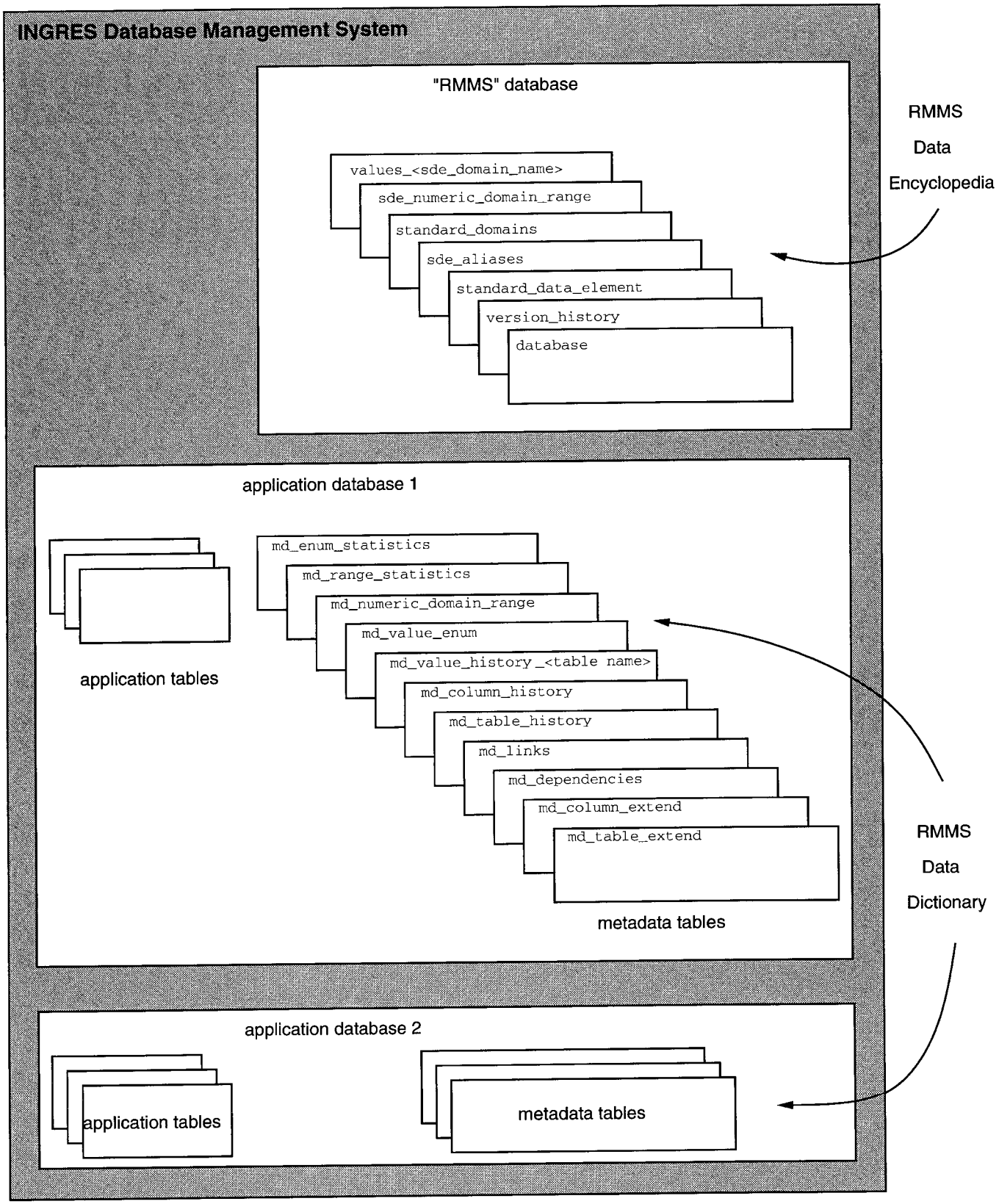

Figure 1—RMMS System Architecture

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. (For the remainder of this document, the Courier typeface is used to indicate specific INGRES table names and column names.) 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 `values_`<sde_domain_name> represent standard data elements and corresponding domains and their values (both character and numeric). The tables named `values_`<sde_domain_name> refer to one table for each standard domain where each table enumerates allowable values for that domain. Appendix A details each of these tables and their column names.

As we discussed earlier, INGRES maintains within each database its own "system" tables, 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 that contain RMMS metadata from those tables in the application database that contain actual data values. We use the terminology "metadata tables" to refer to tables that 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. For example, `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 are stored in `md_value_enum` and `md_numeric_domain_range`. Finally, statistics about

value distributions are maintained in `md_range_statistics` and `md_enum_statistics`. A detailed discussion of each `md_` Data Dictionary table and column can be found in Appendix B.

The implementation of RMMS is an ongoing activity. Schemas for the Data Encyclopedia tables and Data Dictionary metadata tables have been developed and installed. However, acquiring metadata to populate the Data Encyclopedia and Data Dictionary is a continuing task. RAND's data administrators are strongly recommending that newly registered databases 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 development 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.

7. Conclusions

RMMS responds to a majority of issues faced by scientific database users for integrating, sharing, and reusing application databases. In this final section, we present one issue that 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 nonatomic 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 two reasons: First, DBMSs, and especially relational DBMSs, 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 nondecomposable. 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 DBMSs 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 nonatomic 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 by (1) the proliferation of databases stored and used at RAND and (2) 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 that 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 that serve as input to simulation models. We have also proposed using 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. A detailed discussion of each table and its attributes follows the list of tables.

`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`
- `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`

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

sde_numeric_domain_range metadata table
- sde_domain_name
- sde_min
- sde_max

A discussion of each Data Encyclopedia metadata table now follows.

The `database` Metadata Table

This table contains basic information about each of the relational databases maintained in the MOSF and available for user applications. It may or may not include temporary and working databases, depending on the desires of the DBA (database administrator). This table contains one row for every available database. The columns in this table are listed below.

`db_name`

explanation: the name assigned to the INGRES database.

allowable values: fixed-length character string of 24 characters beginning with a letter.

`current_status`

explanation: a flag to indicate whether the database is active or inactive.

allowable values: {"A" "I"}.

`source_information`

explanation: information about the agency that supplied the database including agency name, address, point of contact; if this database is derived, then this field should indicate what databases it was derived from.

allowable values: freeform variable-length text string.

`current_rand_version`

explanation: the version number assigned and maintained by MOSF representing the data as of a particular date.

allowable values: {"1.0" "1.1" . . . "1.n" . . . "m.n"}.

`checkpt_version`

explanation: last backed-up or archived version.

allowable values: {"1.0" "1.1" . . . "1.n" . . . "m.n"}.

`valid_users`

explanation: list of login names for users who are allowed access to the databases.

allowable values: freeform variable-length text string.

`documentation`

explanation: what documents are available, where they are located, who distributes them.

allowable values: freeform variable-length text string.

`description`

explanation: a general overall description of the contents of the database.

allowable values: freeform variable-length text string.

Note that the field `db_owner` (the database owner) is an important piece of metadata but is not included in this metadata table because it can be found in the INGRES system tables.

The `version_history` Metadata Table

This table provides a complete history of all versions of each database, from their initial installation through their current version. This table contains one row for each new version (agency-generated or RAND-generated) of each available database. The columns in this table are listed below.

`db_name`

explanation: the name assigned to the INGRES database.

allowable values: fixed-length character string of 24 characters beginning with a letter.

`agency_version`

explanation: the version number assigned by the distributing agency.

allowable values: fixed-length character string.

`agency_date`

explanation: the date of the database, assigned by the distributing agency.

allowable values: INGRES date type.

`rand_version`

explanation: the version number assigned and maintained by MOSF representing the data as of a particular date.

allowable values: {"1.0" "1.1" . . . "1.n" . . . "m.n"}.

`rand_date`

explanation: the date when the database that corresponds to the `rand_version` number was installed in the MOSF.

allowable values: INGRES date type.

The `standard_data_element` Metadata Table

This table contains information about standard data elements (sde) which have been established in the MOSF. This table relates to the `sde_aliases` table though the column `sde_name` found in both this table and the `sde_aliases` table described next. This table contains one row for every standard data element. The columns in this table are listed below.

`sde_name`

explanation: a terse name for the standard data element.

allowable values: fixed-length character string.

`sde_long_name`

explanation: a full name for the standard data element.

allowable values: fixed-length character string.

`sde_units`

explanation: the unit of measure (if applicable) assigned to the standard data element.

allowable values: fixed-length character string.

`sde_domain_name`

explanation: the name of a standard domain found in the sde_domain metadata table.

allowable values: fixed-length character string.

`sde_source`

explanation: the source from which the standard data element information was obtained or derived.

allowable values: freeform variable-length text string.

`sde_composite_datatype_flag`

explanation: this flag indicates whether the standard data element is a composite data type.

allowable values: {"y" "n"}.

`sde_description`

explanation: a description of the standard data element (which could be an aggregated or derived standard data element).

allowable values: freeform variable-length text string.

The `sde_aliases` Metadata Table

This table contains the names of all columns in all MOSF database tables that denote a concept that has an associated standard data element. In this table, the `column_name` is considered an alias for the name given by `sde_name`. In general, an alias is a *different* name for referring to a particular concept. However, our use of "alias" also includes column names that are identical to a standard data element name. Therefore, this table can be used as a directory to determine which databases contain the same concept (regardless of whether its name is identical to or different from the name of the standard data element). The columns in this table are listed below.

`sde_name`

explanation: a terse name for the standard data element.

allowable values: fixed-length character string.

`db_name`

explanation: the name assigned to the INGRES database.

allowable values: fixed-length character string of 24 characters beginning with a letter.

`table_name`

explanation: the name of the table containing the alias.

allowable values: fixed-length character string.

`column_name`

explanation: the name of the column denoting the alias.

allowable values: fixed-length character string.

The `standard_domains` Metadata Table

This table contains domain information for each standard data element. Domain information specifies the allowable values that are acceptable for a standard data element. For each standard data element in the `standard_data_element` metadata table, there are one or more entries in this table. In cases where more than one domain is applicable to a standard data element, the domain sets will be related by subset relationships; that is, for a particular standard data element, a

subdomain will be a subset of the inclusive domain set. For example, if the set of countries in the world serves as a standard domain for the standard data element "country," then the set of European countries or the set of Eastern European countries is considered a *subdomain* of that standard data element. The columns in this table are listed below.

`sde_domain_name`

explanation: the standard name given to a domain; corresponds to `sde_domain_name` in the `standard_data_element` metadata table.

allowable values: fixed-length character string.

`sde_name`

explanation: the name of the standard data element for which this domain applies; corresponds to `sde_name` in the `standard_data_element` metadata table.

allowable values: fixed-length character string.

`sde_domain_category`

explanation: the type of the domain (enumerated list of character strings, a patterned string, or a numeric type).

allowable values: {"enum" "pattern" "num"}.

`values_table_name`

explanation: if the `sde_domain_category` is "enum," then this field names the metadata table containing the allowable values.

allowable values: the metadata table name is formed by prefixing the `sde_domain_name` with "`values_`".

`sde_valid_pattern`

explanation: if the `sde_domain_category` is "pattern," then this column records a pattern specification; for example, the pattern specified for a social security number may be "`NNN-NN-NNNN`" where `N` is one of 0,1,2, . . . 9.

allowable values: fixed-length character string.

`sde_domain_description`

explanation: a description of the standard domain.

allowable values: freeform variable-length text string.

The `values_<sde_domain_name>` Metadata Table

This table records the allowable values for enumerated types. There is one table for every enumerated type domain (whose name is qualified by the

sde_domain_name). Each table contains one row for each allowable value. The columns in this table are listed below.

standard_value
explanation: a legal and acceptable value.
allowable values: fixed-length character string.

expansion
explanation: an expanded form of the standard_value if it is abbreviated or encoded.
allowable values: freeform variable-length text string.

description
explanation: a description of the meaning of the value.
allowable values: freeform variable-length text string.

The sde_numeric_domain_range Metadata Table

This is a single table to express ranges for all numeric fields that are recorded as standard data elements. All numeric fields are treated as belonging to a floating-point domain. This table contains one row for each numeric domain corresponding to a standard data element. The columns in this table are listed below.

sde_domain_name
explanation: the standard name given to a domain; corresponds to sde_domain_name in the standard_data_element metadata table.
allowable values: floating-point number.

sde_min
explanation: the minimum value allowed.
allowable values: floating-point number.

sde_max
explanation: the maximum value allowed.
allowable values: floating-point number.

Appendix B

RMMS Data Dictionary Tables

The names of the tables and the corresponding columns that comprise the Data Dictionary are listed below. A detailed discussion of each table and its attributes follows the list of tables. 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 that contain RMMS metadata from those data tables that contain column values.

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
- sde_name
- valid_pattern
- sub_domain_class_name
- sub_domain_category
- sub_domain_class_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_dependencies metadata table
- dependent_table
- dependent_column
- independent_table
- independent_column
- description

md_links metadata table
- table_name1
- column_list1
- table_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_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

A discussion of each Data Dictionary metadata table now follows.

The `md_table_extend` Metadata Table

This table contains general information about data tables in the INGRES database. As discussed above, it has the suffix "`_extend`" because INGRES system tables also contain information about tables in the database including such information as number of rows, storage structure, and column names. This table contains one row for every data table in the database. The columns in this table are listed below.

`table_name`
explanation: the name of the table.
allowable values: fixed-length character string.

`unique_id`
explanation: columns in the data table that are unique and may be used as keys.
allowable values: freeform variable-length text string (which is machine readable).

`table_source`
explanation: records the source of the table; if the table was derived, then this includes by whom, when, and why it was derived.
allowable values: freeform variable-length text string.

`description`
explanation: a description of the contents of the table.
allowable values: freeform variable-length text string.

The `md_column_extend` Metadata Table

This table contains general information about the columns in data tables in the database. This table contains one row for every column, for every data table in the database. This table may contain multiple entries for an `sde_name`. The columns in this table are listed below.

`table_name`

explanation: the name of the table containing this column.

allowable values: fixed-length character string.

`column_name`

explanation: the name of the column.

allowable values: fixed-length character string.

`column_source`

explanation: documents the source of the column for derived databases.

allowable values: freeform variable-length text string.

`column_units`

explanation: a units metric, such as, feet, meters, etc.

allowable values: fixed-length character string.

`units_translation`

explanation: a procedure for converting units to and from the standard metric; if this value is prefixed by "PROC:", then the value is regarded as a specific software subroutine to be applied for translation.

allowable values: freeform variable-length text string.

`init_rand_version`

explanation: version of the database in which this column was initiated or originated.

allowable values: {"1.0" "1.1" . . . "1.n" . . . "m.n"}.

`composite_datatype_flag`

explanation: indicates whether this column is a composite datatype.

allowable values: {"y" "n"}.

`dependency_flag`

explanation: indicates whether this column is dependent on any other columns in this database.

allowable values: {"y" "n"}.

`column_description`

explanation: a description of the meaning of the column.

allowable values: freeform variable-length text string.

`sde_name`

explanation: the column's standard data element name; corresponds to `sde_name` in the `standard_data_element` metadata table.

allowable values: fixed-length character string.

`valid_pattern`

explanation: if the `sub_domain_category` is "pattern," then this field records the pattern specification.

allowable values: fixed-length character string.

`sub_domain_class_name`

explanation: name of the subdomain class that is acceptable for this column.

allowable values: fixed-length character string.

`sub_domain_category`

explanation: the type of the domain, which may be an enumerated list of character strings, a patterned string, or a numeric type.

allowable values: {"enum" "pattern" "num"}.

`sub_domain_class_description`

explanation: description of the subdomain class.

allowable values: fixed-length character string.

Note that the fields `column_type` (the datatype of the column) and `column_length` (the length of the field) are important pieces of metadata but are not included in this metadata table because they can be found in the INGRES system tables.

The `md_value_enum` Metadata Table

This table contains the valid enumerated lists that can be used as allowable values for columns in data tables whose domain is an enumerated type. This table contains one row for each value of each subdomain of type "enum" that is represented in the database. The columns in this table are listed below.

`sub_domain_class_name`

explanation: name of the subdomain class that is acceptable for this column.

allowable values: fixed-length character string.

`column_value`

explanation: the value found in the column.

allowable values: fixed-length character string.

`standard_value`

explanation: this is the standard value established in the `sde_domain` metadata table in the Data Encyclopedia database.

allowable values: fixed-length character string.

`description`

explanation: description of the mapping between the columns `column_value` and `standard_value`.

allowable values: freeform variable-length text string.

The `md_numeric_domain_range` Metadata Table

This table contains the valid ranges that can be used as allowable values for columns in the data tables whose domain is a numeric range. Note that all numeric fields are treated as belonging to a floating-point domain. This table contains one row for each numeric subdomain. The columns in this table are listed below.

`sub_domain_name`

explanation: name of the subdomain that is acceptable for this column.

allowable values: fixed-length character string.

`sub_domain_min`

explanation: the minimum acceptable value in this subdomain.

allowable values: floating-point number.

`sub_domain_max`

explanation: the maximum acceptable value in this subdomain.

allowable values: floating-point number.

The `md_dependencies` Metadata Table

This table contains entries describing dependencies between columns. The dependencies may be between columns in the same table or in different tables. This table may contain zero, one, or more entries for each row in the `md_column_extend` metadata table. The columns in this table are listed below.

`dependent_table`

explanation: the table name where the dependent column is found.

allowable values: fixed-length character string.

`dependent_column`
explanation: the name of the dependent column.
allowable values: fixed-length character string.

`independent_table`
explanation: the table name where the independent column is found.
allowable values: fixed-length character string.

`independent_column`
explanation: the name of the independent column.
allowable values: fixed-length character string.

`description`
explanation: a description of the dependency.
allowable values: freeform variable-length text string.

The `md_links` Metadata Table

This table records the "foreign key" to "key" connections between different tables in the database. This information is necessary for determining how you join two tables. This table contains one row for each pair of data tables that are related through a "foreign key" to "key" column. The columns in this table are listed below.

`table_name1`
explanation: the table name of one table participating in the link.
allowable values: fixed-length character string.

`column_list1`
explanation: one or more column names that form a key in table_name1 (which is machine readable).
allowable values: freeform variable-length text string.

`table_name2`
explanation: the table name of the other table participating in the link.
allowable values: fixed-length character string.

`column_list2`
explanation: one or more column names that form a key in `table_name2` (which is machine readable).
allowable values: freeform variable-length text string.

`cardinality`

explanation: indicates the number of rows in `table_name1` that correspond to one or more rows in `table_name2`.

allowable values: {"1–1" "1-N" "M-N"}.

The `md_table_history` Metadata Table

This table contains an account of all data table deletions, additions, or name changes. The purpose of this metadata is to record changes to the database that occur at the database level, that is, the addition or deletion of data tables. Because this metadata table maintains information about a data table, its format should correspond closely to the format of the `md_table_extend` metadata table. Therefore, an entry in the `md_table_history` table contains the same columns as in the `md_table_extend` table (excluding description) AND the additional columns listed below.

`init_rand_version`

explanation: version of the database in which this column was initiated or originated.

allowable values: {"1.0" "1.1" . . . "1.n" . . . "m.n"}.

`time_stamp`

explanation: automatic time stamp recording when the change was made.

allowable values: INGRES date type.

`table_change_type`

explanation: indicates the type of change made.

allowable values: {"table_delete" "table_add" "table_chg_name"}.

`table_change_source`

explanation: indicates where the new changed data came from.

allowable values: freeform variable-length text string.

`table_change_authorization`

explanation: name of person or agency who authorized the change.

allowable values: fixed-length character string.

`table_change_processor`

explanation: name of person who executed the change in INGRES.

allowable values: fixed-length character string.

`table_change_description`

explanation: description of reason the change was made.

allowable values: freeform variable-length text string.

The `md_column_history` Metadata Table

This table contains an account of all column deletions, additions, or format changes. The purpose of this metadata table is to record changes to the database that occur at the table or schema level—that is, additions, deletions, or changes to the column formats such as changing the data type of a column (e.g., from integer to float) or changing the length of a column (e.g., from 20 characters to 30 characters). Because this metadata table maintains information about columns in a data table, its format should correspond closely to the format of the `md_column_extend` metadata table. Therefore, an entry in the `md_column_history` table contains the same columns as in the `md_column_extend` table (excluding description) AND the additional columns listed below.

`init_rand_version`

explanation: version of the database in which this column was initiated or originated.

allowable values: {"1.0" "1.1" . . . "1.n" . . . "m.n"}.

`time_stamp`

explanation: automatic time stamp record of when the change was made.

allowable values: INGRES date type.

`column_change_type`

explanation: indicates the type of change made.

allowable values: {"delete" "add" "chg_len" "chg_type"}.

`column_change_source`

explanation: indicates where the changed data came from.

allowable values: freeform variable-length text string.

`column_change_authorization`

explanation: name of person or agency who authorized the change.

allowable values: fixed-length character string.

`column_change_processor`

explanation: name of person who executed the change in INGRES.

allowable values: fixed-length character string.

column_change_description

explanation: description of reason the change was made.

allowable values: freeform variable-length text string.

The md_value_history_<data table name> Metadata Table

This set of tables contains an account of all changes to a column value. The purpose of these metadata tables is to record changes to the database that occur at the column level, i.e., modifications to data values. In the discussion above describing the md_table_history and md_column_history metadata tables, note that there is one of each of these tables for each INGRES database. However, recording the changes made to values of specific columns requires metadata corresponding to the format of the specific data table. Therefore, one of these value history metadata tables exists for each data table in the database. Each table contains one row for each value change. To distinguish this metadata table from the data table, we prefix the metadata table name with "md_value_history_". Therefore, an entry in the md_value_history_<data table name> table contains the same columns as in the <data table name> table AND the additional columns listed below.

This table mimics the format of the data table, but stores only the old values of columns (along with some documentation explaining the change). In this way, RMMS is recording a complete history of the values of all versions of the database. In cases where the schema is changed (and therefore is recorded in the md_column_history table above), the change will also take effect in the md_value_history_<data table name>. For example, if a column data type changes, the md_value_history_<data table name> table is recreated keeping the old column definition, but also adding a column with the new definition. The name that RMMS assigns to this new column is yet to be determined.

init_rand_version

explanation: version of the database in which this column was initiated or originated.

allowable values: {"1.0" "1.1" . . . "1.n" . . . "m.n"}.

time_stamp

explanation: automatic time stamp recording when the change was made.

allowable values: INGRES date type.

`value_change_type`
explanation: indicates the type of change made.
allowable values: {"row_insert" "row_delete" "row_modify" "column_delete" "column_add" "column_chg_length" "column_chg_type"}.

`value_change_source`
explanation: indicates where the changed data came from.
allowable values: freeform variable-length text string.

`value_change_authorization`
explanation: name of person or agency who authorized the change.
allowable values: fixed-length character string.

`value_change_processor`
explanation: name of person who executed the change in INGRES.
allowable values: fixed-length character string.

`value_change_description`
explanation: description of reason the change was made.
allowable values: freeform variable-length text string.

The `md_range_statistics` and `md_enum_statistics` Metadata Tables

These tables contain useful statistics on the values stored in data tables. Because the statistics gathered depend on whether the values of a column are within a numeric range or are an element of an enumerated type, RMMS supports two different types of statistics metadata tables. These tables are generated automatically based on values found in the data tables; therefore, in the discussion below "allowable values" has been changed to "possible values." For range statistics, each table contains one row for each column of a data table for which statistics are recorded. For enumerate lists, each table contains one row for each value of each row for which statistics are recorded. The columns in each table are listed below.

The `md_range_statistics` Metadata Table

`table_name`
explanation: the name of the table containing this column.
possible values: fixed-length character string.

`column_name`
explanation: the name of the column corresponding to these statistics.
possible values: fixed-length character string.

`minimum`
explanation: the minimum value recorded in the column.
possible values: same allowable values and datatype as the column.

`maximum`
explanation: the maximum value recorded in the column.
possible values: same allowable values and datatype as the column.

`average`
explanation: the average of the values recorded in this column.
possible values: floating-point number.

`median`
explanation: the median value recorded in this column.
possible values: same allowable values and datatype as the column.

`standard_deviation`
explanation: the standard deviation of the values recorded in this column.
possible values: floating-point number.

`null_count`
explanation: the number of null values recorded in this column.
possible values: integer.

`sum`
explanation: the sum of the values recorded in this column.
possible values: same allowable values and datatype as the column.

`percentile_1st`
explanation: value of this column at the 1st percentile.
possible values: same allowable values and datatype as the column.

`percentile_99th`
explanation: value of this column at the 99th percentile.
possible values: same allowable values and datatype as the column.

`quartile_25th`
explanation: value of this column at the 25th percentile.
possible values: same allowable values and datatype as the column.

`quartile_75th`
explanation: value of this column at the 75th percentile.
possible values: same allowable values and datatype as the column.

The `md_enum_statistics` Metadata Table

`table_name`
explanation: the name of the table containing this column.
possible values: fixed-length character string.

`column_name`
explanation: the name of the column corresponding to these statistics.
possible values: fixed-length character string.

`unique_value`
explanation: value of the column for which statistics are recorded.
possible values: same allowable values and datatype as the column.

`count`
explanation: number of rows with the given value for this column.
possible values: integer.

`percent`
explanation: percentage of rows with the given value for this column.
possible values: floating-point number.

References

Adiba, M., and N. Gui. Quang (August 1986). "Historical Multi-Media Databases." *Proceedings of the Twelfth International Conference on Very Large Data Bases*, Kyoto, pp. 63–70.

Bruce, T. (1991). *Designing Quality Databases with IDEF1X Information Models*, Dorset House Publishing, New York.

Dadam, P., V. Lum, and H. D. Werner (August 1984). "Integration of Time Versions into a Relational Database System." *Proceedings of the Tenth International Conference on Very Large Data Bases*, Singapore, pp. 509–522.

Department of Defense. 8320.1-M-1. (1992). "Standard Data Element Development, Approval, and Maintenance Procedures," Department of Defense, Office of the Assistant Secretary of Defense (Command, Control, Communications, and Intelligence).

Dolk, D., and R. Kirsch (January 1987). "A Relational Information Resource Dictionary System," *Communications of the ACM*, Vol. 30, No. 1, pp. 48–61.

French, J. C., A. K. Jones, and J. L. Pfaltz (1990). "A Summary of the NSF Scientific Database Workshop." *Data Engineering*, Vol. 13, No. 3, pp. 55–62.

Goldfine, A. (October 1985). "The Information Resource Dictionary System," *Proceedings of the Fourth International Conference Entity-Relationship Approach*, Chicago, pp. 114–122.

Goldfine, A., and P. Konig (January 1988). "A Technical Overview of the Information Resource Dictionary System (Second Edition)," NBSIR 88–3700, U.S. Department of Commerce.

Jones, M. (November 1991). "Brave New World: A Vision of IRDS," *Database Programming and Design*.

Kerschberg, L., D. Marchand, and A. Sen (1983). "Information System Integration: A Metadata Management Approach," *Proceedings of the Fourth International Conference on Information Systems*, Houston, pp. 223–239.

Kossman, R. (April 1987). "An Active Information Resource Dictionary," *Proceedings of INGRES User Association Meeting*, San Francisco.

Mark, L., and N. Roussopoulos (December 1986). "Metadata Management," *Computer*, Vol. 19, No. 12, pp. 26–35.

McCarthy, J. L. (September 1982). "Metadata Management for Large Statistical Databases," *Proceedings of the Eighth International Conference on Very Large Data Bases*, Los Angeles, pp. 234–243.

Navathe, S., and L. Kerschberg (January 1986). "Role of Dictionaries in Information Resource Management," *Information and Management*, Vol. 10, No. 1, pp. 21–46.

OpenWindows Developer's Guide 1.1 User's Manual. (June 1990). Report Number 800–5381–10, Sun Microsystems, Inc., Mountain View, CA.

Segev, A., and A. Shoshani (1987). "Logical Modeling of Temporal Data." *SIGMOD Quarterly.*

Snodgrass, R., and I. Ahn (1986). "Temporal Databases." *Computer*, Vol. 19, No. 9.

MR-163-OSD/A/AF

ISBN 0-8330-1515-X