A health-care data model based on the HL7 Reference Information Model

T. J. Eggebraaten
J. W. Tenner
J. C. Dubbels

The integration of medical information from various sources is gaining in importance as hospitals and medical research centers attempt to gain new insights into existing data. The Health Level Seven® (HL7®) organization has developed an abstract information model for health-care data, the HL7 Reference Information Model (RIM). We describe in this paper our approach to implementing a physical data model based on RIM. Our approach, which combines elements of entity-relationship data modeling and entity-attribute-value data modeling, involves the modeling of base RIM classes, RIM inheritance, and RIM data types. We incorporated the resulting data model into IBM Clinical Genomics, a product that integrates clinical and genomic data in a way that enables medical researchers to carry out clinical research.

INTRODUCTION

Integration of enterprise-wide clinical information is getting more attention as hospitals and academic medical research centers attempt to gain new insights into existing data. Health-care data has many unique characteristics that differentiate it from other industries and that make it suitable for an abstract data model approach. These include data sparseness, a very large number of dimensions, nonadditive facts, a constantly changing set of attributes, and the need for near real-time data.[1]

Bill Inmon, the father of the data warehouse concept, defines a data warehouse as a "subject-oriented, integrated, nonvolatile, and time-variant collection of data in support of management's decisions."[2] Data warehouse architectures include components such as staging databases, operational data stores (ODS), atomic data stores, and data marts. Building and managing a data warehouse encompasses topics such as security, metadata, end-user access tools, team roles and responsibilities, and project principles. These important topics are covered elsewhere.[3] In building an effective warehouse, the design of the database models and populating the database with data play an important role.[2] Importing data into the warehouse, also known as the Extract/Transform/Load (ETL) process, is usually the most time-consuming and expensive part of any data warehouse project.

Whereas in other industries, generic data warehouse models have proven successful,[4] the characteristics of health-care data complicate both the ETL process and the design of the physical data model. Much work has been done by the Health Level Seven** (HL7**) organization to create object models for health care to support the exchange, management, and integration of health-care information.

■ Health-care data has many unique characteristics that differentiate it from other industries and which make it suitable for an abstract data model approach ■

The HL7 organization, which is accredited by the American National Standards Institute (ANSI**), has the mission to develop standards for the health-care industry worldwide. HL7 participants include the top health-care organizations in the world. The HL7 standards specify the transmission and exchange of health-care data between applications, systems, and organizations. The HL7 Version 2.x set of standards (Version 2.5 is the latest) is considered to be the workhorse in health-care data exchange and is the most widely implemented standard for health-care information in the world.

In 1997, HL7 began to work on HL7 Version 3 (HL7 V3), whose message development approach differs significantly from previous versions. All message specifications from HL7 V3 onward will be derived from the HL7 Reference Information Model (RIM). The HL7 RIM (RIM, for short) is a static model of health-care information that broadly and abstractly covers all aspects of a health-care organization's clinical and administrative information. In the HL7 V3 message development process, RIM is constrained to cover just the information needed for a particular message. Although it was not intended for the purpose of database design, RIM provides an integrated model for health-care data, and we found it to be a suitable basis for a data model used in a data warehouse architecture.

A RIM-based data model could be used in a number of ways in a data warehouse architecture. Although it could be used as the atomic data store or as an ODS, we have found that it is better suited as a staging database that feeds data to another layer in the data warehouse. This placement simplifies the process of getting data into the warehouse by enabling storage of all data from HL7 V3 messages without requiring query and reporting tools to deal with the complexities of a RIM-based data model for the ODS or atomic data store.

We describe in this paper our approach to implementing a physical data model based on RIM. We incorporated our solution into the IBM Clinical Genomics (CG) product.[5] CG integrates clinical and genomic data in order to enable medical researchers to carry out clinical research. CG is based on a message-oriented architecture that supports several messaging standards for health care, including HL7.

The rest of this paper is organized as follows. We first examine the two main data modeling methodologies that we used in implementing a physical data model for health-care data: the entity-relationship (ER) methodology and the entity-attribute-value (EAV) methodology. Then we describe our approach, which combines elements of ER and EAV models, and we discuss modeling of the base RIM classes, RIM inheritance, and the RIM data types. Next, we describe the special characteristics of health-care data and the ways we deal with the challenges that arise. We then describe the application of our design to CG. The last section contains concluding remarks.

DATA-MODELING METHODOLOGIES

Methodologies for modeling data include hierarchical models, network models, relational models, ER models, EAV models, and object-oriented models. Our RIM-based data model combines elements of relational, ER, and EAV models.

Entity-relationship model

The design of databases most often uses an ER data model. In an ER model, an entity represents a discrete object or concept, whereas a relationship represents an association between two or more entities. Both entities and relationships can have attributes.

Because the ER model is conceptual, it is typically implemented by using a relational model where an entity is typically implemented as a table and

attributes are implemented as columns within that table. Relationships can be implemented either as separate tables (this is necessary when the relationship is many-to-many) or as foreign key columns in an entity table.

Attributes can be thought of as facts about an entity. In the typical implementation of an ER model, all attribute values for an entity are stored in the same row in the entity table, in which the column names specify the attributes of the entity. A patient's street address, date of birth, and hematocrit percentage would each correspond to a different column. Whether two columns are in the same table or in different tables is a matter of database normalization.

Entity-attribute-value model

Because of the unique characteristics of health-care data, a common model used in health-care databases is the EAV data model.[6,7] In the EAV data model, each row of a table corresponds to an EAV triple: an entity, an attribute, and the attribute value. For example, the entity "patient" has the attribute "street address," a value of which could be the text "123 Chestnut Street." Similarly, the entity "patient" can also have the attribute "hematocrit percentage," a laboratory test result with an integer value of 41.

The use of EAV models for clinical data is appealing because adding new attributes to an entity does not require changes to the database design. This is an important consideration for clinical applications for which new attributes in the form of laboratory measurements and diagnosis variables are frequently added. Whereas EAV-based physical data models are easy to design and administer, the design and administration of ER databases are more complex and thus more costly than EAV databases. This is because there are thousands of different laboratory tests and EAV models are more efficient at representing sparse data.

Constructing queries for EAV models, however, is more complex.[6] Consider, for example, a query to retrieve the patient identifier (ID) and name for all patients who reside on a given street. In a conventional (ER) database, in which a column represents an attribute, the query would take the form:

```
SELECT patientID as 'Patient ID', name as 'Name'
FROM demographics
WHERE street LIKE '% Chestnut Street%'
```

In an EAV database, the query would instead look like this:

```
SELECT t1.patientID as 'Patient ID', t2.name as
'Name'
FROM patient t1, patient t2
WHERE t1.patientID = t2.patientID AND
t1.attribute = 'street' AND
t1.value = 'Chestnut Street' AND
t2.attribute = 'name'
```

The EAV database query requires more predicates because both the attribute and the column need to be specified. In addition, in the conventional design, because the name of the patient and the street address columns can be in the same table, only one table needs to be specified in the query. In the EAV database query, the table needs to be self-joined in order to return the name of the patient.

If we consider repeating elements such as laboratory tests and take into account the effects of normalization, the number of tables to be joined in the EAV design is not always larger than in the ER design. Consider for example the query that returns the patient ID and name for all patients with a glucose reading greater than 6.1 and a hematocrit percentage greater than 41. In the conventional design, the query would use three different tables:

```
SELECT patientID as 'Patient ID', name as
'Name', t2.value as 'Glucose'
t3.value as 'Hematocrit'
FROM demographics t1, glucose t2, hematocrit t3
WHERE t1.patientID = t2.patientID AND
t2.patientID = t3.patientID AND
t2.value > 6.1 AND
t3.value > 41
```

In the EAV design, the query joins three instances of one table and looks like this:

```
SELECT t1.patientID as 'Patient ID',
t1.name as 'Name', t2.value as 'Glucose',
t3.value as 'Hematocrit'
FROM patient t1, patient t2, patient t3
WHERE t1.patientID = t2.patientID AND
t2.patientID = t3.patientID AND
t2.attribute = 'glucose' AND
t2.value > 6.1 AND
t3.attribute = 'hematocrit' AND
t3.value > 41 AND
t1.attribute = 'name'
```

Because of the generality of the RIM model, which includes attributes without a predetermined data type, its implementation required the use of some aspects of EAV models, as discussed in the next section.

IMPLEMENTING THE HL7 RIM-BASED DATA MODEL

Implementing the RIM-based data model means mapping it to a physical database model. It is important to create a physical model that closely matches the RIM logical model so that the mapping between HL7 messages and the database is straightforward. We use an approach that combines elements of ER and EAV models.

Modeling the base HL7 RIM classes

The HL7 RIM is comprised of the following base classes[8]:

- *Act*—represents *actions* that have happened, are happening, or are scheduled to happen
- *Entity*—represents physical things or beings such as persons, places, or devices
- *Role*—represents the role that Entities play as they participate in a health-care act
- *RoleLink*—represents a connection between two Roles
- *Participation*—represents the association between a Role and an Act (for example, the context of an Act, such as who performed it, for whom it was performed, or where it was performed)
- *ActRelationship*—represents the association between two Acts (for example, the relationship between an order for a blood test and the result of a blood test)

Most of these classes have subclasses that further refine the concept represented by the class. For each base class in RIM, the subclasses form a hierarchy rooted in the base class. For example, the class LivingSubject is a subclass of Entity, and the class Person is a subclass of LivingSubject.

To illustrate the use of these classes consider a patient whose pulse rate is taken during a visit to the doctor. The patient is represented as an instance of class Person and an instance of class Patient, which is a subclass of Role (a person with the role of patient). The doctor visit is an instance of class PatientEncounter (a subclass of Act), the pulse rate is an instance of class Observation (a subclass of Act). The patient is linked to the visit by an instance of Participation; the pulse rate measurement is linked to the visit through an instance of ActRelationship.

With the exception of the Observation class, which we discuss later, all RIM classes are modeled using the ER approach. Each RIM class is considered an ER entity and mapped to a table in the physical model.[9]

To maintain the relationships between classes as defined in RIM, foreign keys were added to the appropriate class tables. When a relationship involves a base class that has subclasses, the relationship can involve any of the subclasses. In this case, an additional column is added to store the name of the table involved in the relationship. For example, class Role can have two relationships with any subclass of Entity, player and scoper. Therefore, in addition to foreign keys `player_id` and `scoper_id` in table ROLE, columns `player_type` and `scoper_type` are added to specify the table with which the foreign key has a relationship. As there are no many-to-many relationships in RIM, no additional mapping tables are needed to maintain this type of relationship, as is the case in many relational schemas.

Modeling HL7 RIM inheritance

Classes in RIM can inherit attributes from their parent classes. When these classes are mapped to physical tables, there are several ways to model the class inheritance structure.

- *Single table per class hierarchy*—A single table stores all the objects in each class hierarchy; the table contains all the possible attributes for the classes in the hierarchy.
- *Tables that do not contain inherited attributes*—A separate table for each class in RIM contains only the unique attributes for that class. When an object is stored in the database in this case, a row needs to be inserted in the base table and in each of the parent tables in order to store all of this object's information. Foreign keys need to be added to maintain the relationship between the parent tables.
- *Tables that contain inherited attributes*—For each class a separate table is created that contains all the attributes for that class plus all the attributes that it inherits from its parent classes. When an

object is stored in the database, all the information for the object can be stored in a single row of a single table.[9]

The single-table-per-class hierarchy is the simplest approach as it results in the smallest number of tables and table relations. This approach, however, is inefficient, as there are likely to be many unused attributes for each object. There are some data models in which this approach makes sense; for example, when subclasses only contain a few attributes. Because the number of attributes in many of the RIM subclasses is large, this approach is less attractive.

Using tables that do not contain inherited values is the most complex approach. There are no duplicated columns in this approach, which means the storage of the data is efficient, but inserting, updating, and querying data are complex. This is because the data for a particular object are spread across multiple tables in the database; consequently, storing an object requires multiple inserts, and retrieving an object requires joining multiple tables. Database views (i.e. one view per logical class) could alleviate the complexity of the queries, but would not avoid the performance degradation. The advantage of this approach is that referential integrity can be enforced because there is a clear relationship between tables, and all class hierarchies in RIM are linked together at the base class (with a few exceptions).

The approach that offers the most for a RIM-based model relies on tables that contain inherited attributes. With this approach all the data for a particular object are stored in a single table, which makes inserting and querying the data simpler than in the previous approach. Storage utilization is efficient as each table contains only the needed attributes for the class, and complexity is reduced because there is no need for foreign keys to connect parent and child tables. Although referential integrity cannot be enforced between class hierarchies, this is not a problem as long as the data are always loaded in a controlled and trusted manner. In this approach, the application inserting the data is required to enforce the appropriate referential integrity measures.

Figure 1 illustrates the inherited attributes in the RIM-based implementation of class Person. It shows owned and inherited attributes and their relations to data type tables. The Person table contains its own attributes, such as `marital_status_id`, attributes inherited from class Living Subject, such as `gender_code_id`, and attributes inherited from class Entity, such as `status_code_id`. To illustrate how data is stored with this approach, consider the

■ Data sparseness is a characteristic of health-care data that has to be considered when designing a physical data model ■

marital status attribute. Because marital status is a coded value, its data such as `code`, `code_system`, `version`, `display_name`, and so on would be stored in the DT_CE table, and a foreign key to this table would be stored in the `marital_status_id` column of table Person.

MODELING THE OBSERVATION CLASS WITH A HYBRID APPROACH

The Observation class in RIM captures many types of data in HL7 messages, such as diagnoses, laboratory results, allergies, and vital signs. An Observation instance has, in its most basic form, an ID, a code, and a value—a triple that is basically equivalent to an EAV triple. The code identifies the observation (e.g., glucose), and the value represents what was observed (e.g., 35 mg). The `value` attribute (class Observation) is defined as an ANY data type, which means it can be any valid HL7 data type. When messages are created using the HL7 refinement methodology, the `value` attribute, as with other attributes, can be constrained to a specific HL7 data type. However, a RIM data model is intended to store data from any and all HL7 messages and therefore, needs to accommodate any data type that may be associated with an Observation.

Although conceptually EAV models require only three columns, the physical data model normally uses one table for the entity (Observation in this case) and a separate attribute table for each possible data type. Each attribute of an entity is stored as an additional row in one of the attribute tables, depending on the attribute's data type. EAV models tend to be more difficult to query because self-joins are required to access all the information for a given object. The EAV modeling approach is intended to

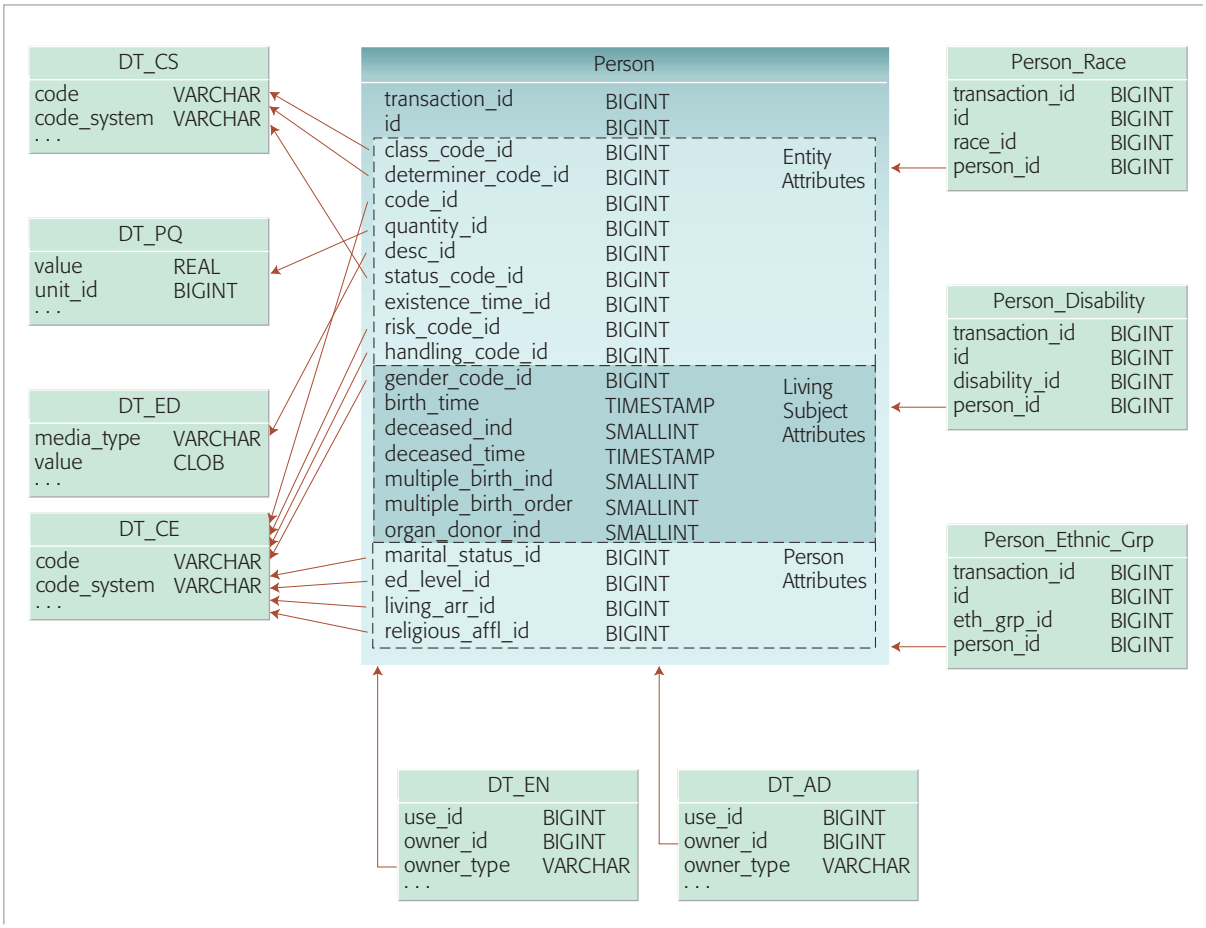

Figure 1
Inherited attributes in the Person class

allow for maximum flexibility in the number and type of attributes that an entity can have, but because RIM is fairly static in terms of attributes per entity, this flexibility is not required. The other valuable aspect of an EAV model is how it can efficiently handle attributes where the data type is not predetermined, such as the `value` attribute. A hybrid between an EAV model and a conventional ER model (hybrid approach, for short) combines the advantages of these approaches and is advantageous when modeling the Observation class due to the flexibility of the `value` attribute (ANY data type). In this hybrid approach, one table is created for each possible data type of the `value` attribute. Each table contains all the Observation's predetermined attributes (attributes where the data type is specified; i.e., it is not the ANY data type) and the value attribute constrained to a specific data type. This allows for all the attributes for a given observation to be stored in a single table, which makes data access easier. However, it also allows flexibility in the data types that can be supported for the observation's value attribute.

Figure 2 shows the EAV modeling approach and our proposed hybrid approach side by side. In RIM, an Observation class is linked to a Patient or other Role subclass by a Participation class. A standard EAV implementation of the Observation class is shown in blue on the left side of Figure 2. An Observation table is linked directly to a Participation table through a foreign key (a Patient table is connected to the Participation table but not shown). The Observation table does not contain any attributes. All attributes and their values are stored in one of the attribute tables depending on the attribute data type. Thus, the `Observation.classCode` attribute is stored in the ATTR_STRING table because it is a

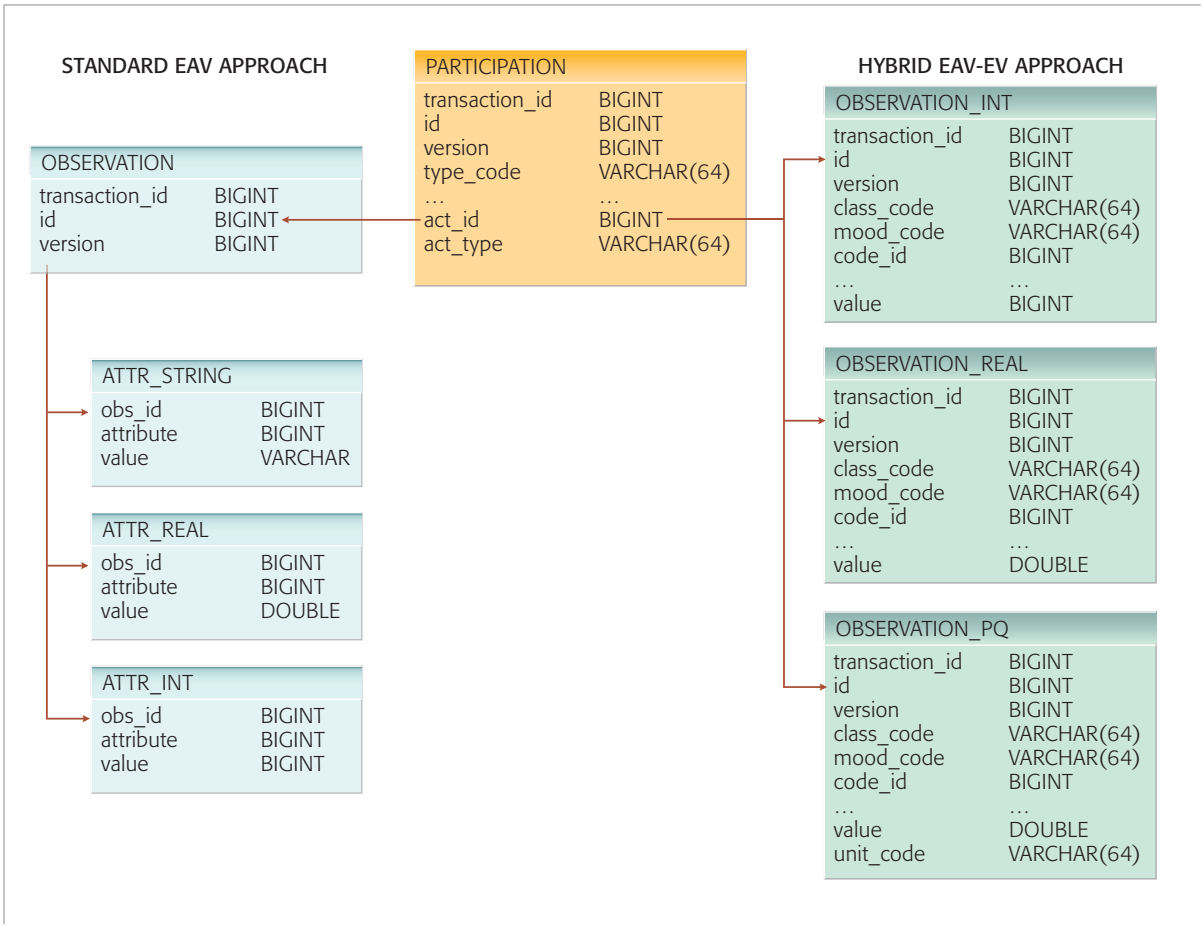

Figure 2
Modeling RIM inheritance: standard EAV-EV approach versus hybrid EAV-EV approach

string data type (the name of the attribute, `classCode`, is stored in the `attribute` column of table ATTR_STRING). A hybrid implementation of the Observation class is shown in green on the right side of Figure 2. Several observation-type tables are created, such as OBSERVATION_PQ, one for each possible data type of the attribute `value`. These tables are also linked to the Participation table through a foreign key. The entry in the `act_type` column in the Participation table identifies the observation-type table for that row. There is no need for separate attribute tables because all attributes that are not of data type ANY are added to each observation-type table.

Modeling HL7 RIM data types

A few of the HL7-defined data types can be mapped directly to a database management system (DBMS) data type, such as the string (ST) data type which can be mapped directly to a variable length character (VARCHAR) database column. Most of the HL7 data types are more complex and need to be modeled differently. When there are multiple parts to a data type, there are three approaches that could be used to represent the data type in a physical model, and all three were used for modeling some of the HL7 data types:

1. *Multiple columns in the class table*—Store each part of the data type as a separate column in the RIM class table. This is an appropriate approach for simple multipart data types where the data does not need to be referred to by other objects and there can be only one instance of the data type per instance of the RIM class. A data type that is suited for this approach is the physical

quantity (PQ) data type which contains two parts: a value and a unit code.

2. *A separate table for the data type and a foreign key in the class table*—Create a separate table to store the data type's data and add a foreign key in the class table that points to the new data type table. This approach is appropriate for more complex data types and data types that can be referred to by multiple objects. An example is the coded element (CE) data type, which has multiple parts (`code`, `codeSystem`, `displayName`, etc.) and which can be referred to by multiple objects. Consider a CE that represents Gender-Male: because it can be referred to by multiple person objects, it is more efficient to store the CE data once in a CE table rather than in each object and to store a foreign key in the person table.
3. *A separate table that includes a foreign key in the data type table*—Create a separate table that contains the data type's data and a foreign key to the owner class table. This approach is needed when the data type is defined as a set in RIM (i.e., `SET<II>`, which is a set of instance identifiers). In the case where a class can have multiple instances of a data type, the foreign key that maintains the relationship between the class and data type table needs to be in the data type table.

■ Operating a data warehouse with near real-time data is a key requirement for health-care applications ■

HEALTH-CARE DATA CONSIDERATIONS

Health-care data have certain properties that make them different from data in other industries, such as retail or insurance. The following characteristics of health-care data need to be considered when a data model is designed for health care:

- *Data sparseness*—Only a small subset of the possible attributes associated with a patient are used on any one patient. For example, the Logical Observation Identifiers Names and Codes (LOINC**) coding system has over 31,000 codes to represent unique laboratory tests,[10] yet most patients will have only a very small number of different laboratory tests performed on them over their lifetime.
- *Very large number of dimensions*[1]—Diagnostic, procedure, and laboratory coding systems contain tens of thousands of different codes for the various medical information items on a patient.
- *Nonadditive variables*—Many laboratory measures, such as glucose or body temperature, cannot be meaningfully added. In some cases, other aggregations, such as averages of nonadditive variables, can be performed. However, such averages are also nonadditive. Therefore, for nonadditive variables, the data must be categorized in order to perform dimensional analysis, such as roll-ups or drill-downs.
- *Rapidly expanding set of attributes*—New laboratory tests, diseases, drugs, genes, and so forth, are being invented or discovered every day. A data model for health care needs to be able to evolve as new types of information are created.
- *Deidentified data*—Although patient-identifying information is needed for some purposes (e.g., billing information but not patient health record sent to external organizations), privacy concerns require that health-care data be deidentified for most purposes.
- *Need to distinguish between source data and derived data*—Some data, such as laboratory measurements, are created at the source and other data are derived, such as annotations to a clinical note generated through a text analysis process or data that has been normalized (mapping of different codes to a unified code). Often it is important to be able to distinguish between source data and derived data, which means that the database system has to keep track of this information.
- *Need to associate some data with explanatory metadata*—In addition to traditional versioning of data, which is required to meet the definition of a data warehouse, a health-care data warehouse needs to be able to accommodate how data is captured over time. Procedures for a particular laboratory test can change over time, and different devices can measure the same data differently. An example is the difference between a body's temperature recorded from an oral thermometer in contrast to a rectal thermometer.
- *Requirement for near real-time data*—Operating a data warehouse with near real-time data is a key requirement for health-care applications. A data warehouse could be used to eliminate errors in the physician's order entry system.[11] Similarly, up-to-the-minute test results could be used to identify

patients present in the clinic in order to approach them for participation in a research study.

As an abstract model of health-care data, RIM offers significant benefits in that it can model any conceivable kind of data. This advantage represents, at the same time, its greatest challenge, which is managing and maintaining data consistency. As health-care data tends to have attributes that are nonadditive and its dimensionality tends to be high, a physical data model should be as general as possible.

Mapping information to RIM can be done in more than one way. A pregnancy complication, for example, could be stored as two objects: an Observation object for the pregnancy, another Observation object for the complication, and an ActRelationship object for the relationship between the two. Another way to record the same information would be to store the pregnancy as an Observation object and the complication as a qualifier code to attribute `value`. This mapping challenge is helped somewhat by use of the defined HL7 V3 messages, as these are constrained instances of RIM with a predefined structure. However, some of the HL7 V3 message standards such as CDA (Clinical Document Architecture) are still intentionally abstract so that they are flexible enough to support the exchange of unstructured health-care data, such as various types of clinical observations and services. Loading data into an HL7-based data model needs to be carefully managed to ensure consistent mapping of information to RIM structures.

In addition to message structure and semantics, it is important to maintain a consistent vocabulary for the terms used in the warehouse. Health-care organizations use many different coding systems for their vocabularies, including LOINC, SNOMED CT**,[12] and ICD-9-CM,[13] as well as their own local coding systems. RIM and HL7 messages support any and all of these coding systems, and they can coexist in a data model based on RIM. If the source systems that are sending data to the warehouse are using different coding systems, normalization of the vocabularies needs to be addressed. There are several ways to do this. One option is that each query and decision support application which accesses the warehouse would handle the vocabulary mapping, which would make these applications more complex. Another option is to normalize the vocabularies in the HL7 messages before the data is entered into the warehouse.

The clinical and genomic data for each patient tends to be sparse. Of the thousands of diseases, medical conditions, and laboratory tests in existence, any given patient is not likely to have information for a majority of the data elements. It would be inefficient to create tables and columns for each of these potential data elements. A more efficient choice is to use a hybrid approach based on the RIM Observation class as was previously described. This approach allows great flexibility in the data types supported for health-care data, but also does not require that each data element be defined as an explicit column.

■ IBM Clinical Genomics integrates clinical and genomic data for medical research purposes ■

A hybrid approach also works well in a health-care environment where new laboratory tests, diseases, medications, and so forth, are constantly added and changed. The data model does not require changes for the addition of a laboratory test as long as the data type for the laboratory test's value is already supported. The instruments and devices used for laboratory tests can change over time, and different instruments can record the same test differently. A simple example is a body temperature measurement, which is recorded slightly differently depending on whether a rectal or oral thermometer is used. This approach requires that metadata be stored along with a laboratory test observation in the warehouse, so that the device which made the observation is clearly defined. RIM supports this metadata by including a participation object to the appropriate device for each observation, which is possible in many HL7 messages and in CDA documents.

In some cases a clinical data warehouse needs to store deidentified data as well as patient-identifying data. Regulatory mandates regarding patient privacy require that patient data be deidentified before being used in patient population studies. Patients may give

permission to use their identifiable data for research purposes. HL7 messages, in general, cannot adequately represent deidentified data—there is no standard way to distinguish between deidentified and identifiable data. RIM does not currently address the concept of deidentified data. However, another possible approach to this challenge is to store the identifiable information in the warehouse and control access to this information by using privacy policies. IBM offers a solution for controlling access to private health-care data known as the Hippocratic Database.[14]

Data warehouses often contain data that is derived from other data. It is not always necessary to store a derived data item, but in many cases it is more efficient to do so rather than compute it each time it is needed. Data from multiple systems that use different coding systems, for example, need to be mapped to a common coding system in the data warehouse. The result of text analysis against clinical notes or pathology reports is yet another example of derived data.

In some cases, it is important to distinguish between source (nonderived) data and derived data. For example, a discrete diagnosis code as a source data item may be more accurate and treated differently than the same code as a derived data item, say, obtained by analysis of the free-text of a clinical document. Therefore, the system should allow the level of uncertainty associated with a derived data item to be specified and stored in the database. One way to approach this is to treat the certainty as a binary value and assume that data which originate from HL7 messages are certain, whereas other derived data are uncertain. Another method is to specify the uncertainty within the message itself by setting the uncertainty code, which is included in all subclasses of Act, including Observation.

One of the major advantages of the RIM data model is the straightforward mapping of HL7 V3 messages to the data model, but this advantage has been limited by the slow adoption of HL7 V3 in the United States. The vast majority of United States health-care organizations are still using HL7 V2 messages (which are not based on RIM), and it will likely take some time for them to move to the new standard because of the high cost associated with such a move. However, there are some major efforts underway that will promote and help expedite the move to HL7 V3.

The National Health System in the United Kingdom is sponsoring the Spine project, which attempts to develop a national, centralized repository for patient information. The Spine architecture is based on HL7 V3 messaging.[15] Other projects that are developing RIM-based data models include the Canadian Health Data Model from the Canadian Institute for Health Information[16] and Oracle Corporation's Healthcare Transaction Base product.[17] Even though currently there are few sources that generate HL7 V3 messages, the work on RIM-based data models will pay off as medical informatics system vendors eventually migrate to HL7 V3. In the meantime, traditional ETL functions are used to move data between health-care data sources and health-care databases. Mapping HL7 V2 messages into HL7 V3 messages is possible, but this transformation can be difficult as there is not always a direct one-to-one mapping between the two formats. The IBM Healthcare Collaborative Network solution can be used to map HL7 V2 messages to RIM-based CDA documents.[18]

In many industries, data warehouses are not updated on a continuous basis. Many are updated daily, weekly, or monthly, which is sufficient in most cases. In health care, decisions need to be based on the most current information, and decision support systems would greatly benefit from integrated real-time data. Integrating data into a data warehouse in real time is a huge challenge, but it will likely be an important requirement of future data warehouse projects. An event-driven architecture based on messaging standards such as HL7 is crucial to implementing a real-time data warehouse. Then data can be sent, as HL7 messages, to the warehouse as soon as they are generated rather than on the timetable of a traditional ETL process.

IBM CLINICAL GENOMICS PRODUCT

In the previous section we described our approach to designing a physical data model based on RIM and discussed its advantages for storing health-care data. In this section we describe the application of our approach to the IBM Clinical Genomics product.

We have implemented a RIM-based data model in CG by using the methodology described earlier. CG is based on a message-oriented architecture that

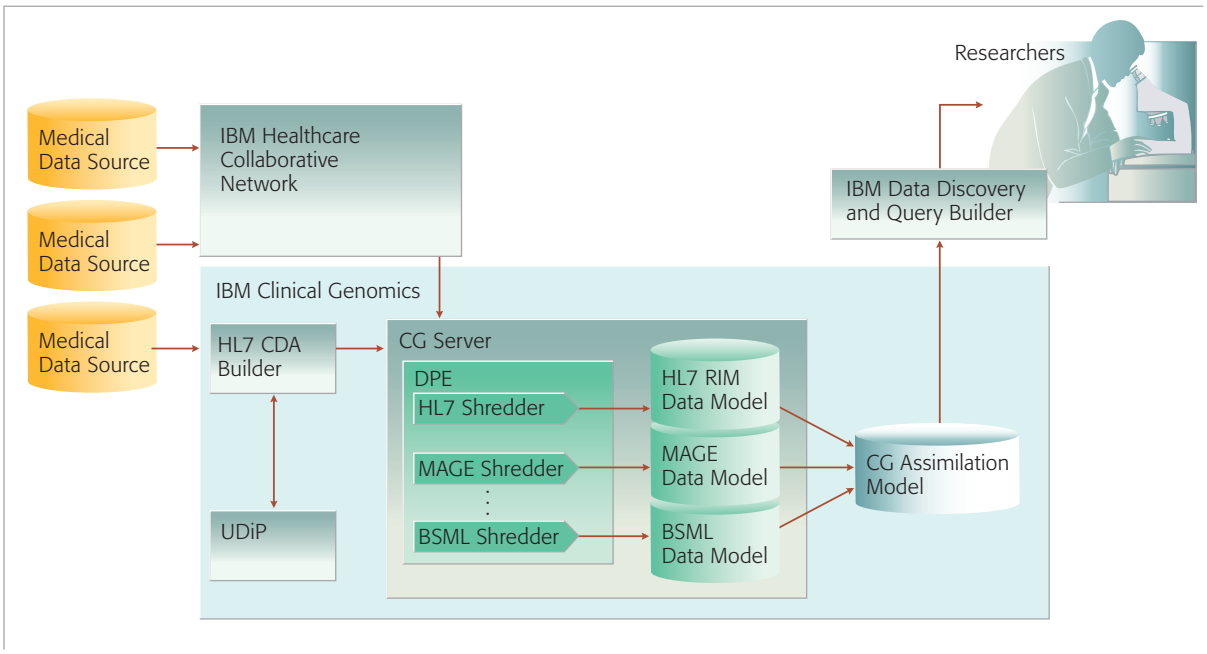

Figure 3
Architecture of IBM Clinical Genomics

supports several messaging standards for health care, such as HL7.

The diagram in ***Figure 3*** shows the CG architecture. The main components are the Healthcare Collaborative Network (HCN) product, the HL7 CDA Builder, the Universal Deidentification Platform (UDiP), the Clinical Genomics server, the Clinical Genomics Assimilation Model (CGAM), and the Data Discovery and Query Builder (DDQB) product.

The HCN product collects health-care information in standard messages such as HL7 V2.x, MAGE-ML (MicroArray Gene Expression Markup Language), BSML (BioSequence Markup Language), HapMap, and ODM (Operational Data Model). It performs deidentification of the data and assigns deidentified global patient identifiers that allow information from multiple institutions to be correlated. Another important feature of this component is the ability to aggregate a set of HL7 V2.x messages into a single RIM-based CDA message, which allows the data to be parsed and loaded into the RIM data model.

The HL7 CDA Builder enables creation of CDA and other HL7 messages in a standard and consistent format. It is a set of Java** APIs that can be used to load clinical data from source systems into a standard message structure for sending to the CG server. The UDiP component is an extendable platform for deidentifying data. It can be used with the HL7 CDA Builder to deidentify the patient information contained in a CDA message.

The data processing engine (DPE) of the CG server routes incoming messages to the appropriate shredder, which parses the message and loads the data into the appropriate data model. A data model is included for each messaging standard supported in CG, including HL7, MAGE, BSML, HapMap, and ODM. The HL7 shredder parses and loads HL7 V3 messages into the RIM data model. The most common HL7 V3 message standard used for transfer of clinical data is CDA, but other HL7 messages can be shredded and loaded into the RIM schema as well. The advantage of this message-based architecture is that additional data sources can be added without the additional cost of traditional ETL between the source system and the warehouse. The message standard (such as CDA) is the interface between the source systems and the warehouse system.

The CGAM data model is used to store current-valued health-care information from various data

sources, including the CG standards-based data models, for analysis and reporting. As previously discussed, the standards-based data models, such as the RIM model, are highly normalized and not optimized for query access. The CGAM data model has been denormalized to some extent to facilitate efficient querying and mining of data.

DDQB enables end users to query complex data, such as that found in the health-care industry.[19] DDQB employs a novel data abstraction technology that allows complex queries to be dynamically created in end-user terms. Because DDQB translates end-user queries into SQL (Structured Query Language), users do not need to specify table names or join predicates. DDQB also supports data access control that meets security, privacy, and auditing requirements. The DDQB data abstraction layer is especially helpful in dealing with databases such as the RIM-based data warehouse. It does this through a data abstraction specification called the Data Abstraction Model (DAM). Within the DAM, fields are defined and associated with access methods, which describe the mapping to the relational table or tables that hold the data for the field. Access method types include simple types (a field maps to a column), composed types (a field maps to an expression), or filtered types (a field relates to a subset of a column based on some condition). The filtered access method can be used with attribute-value pairs to define a field, such as glucose, based on the value column (when the attribute column is a code relating to the laboratory type of glucose). Although DDQB enables direct query of the RIM model (which may be necessary for research), reporting against the CGAM database for current-valued data or against a data mart once analytic dimensions are understood may provide better data access performance. Even though data marts and, to some extent CGAM, provide simpler database designs, DDQB is still helpful for queries against repositories using these models. DDQB features, such as support for generating complex queries without specifying joins and, for enforcing security, privacy, and auditing requirements, apply broadly.

CG and DDQB are key components in a solution that IBM is developing jointly with the Lupus Biomarkers Working Group—an interinstitutional and interdisciplinary group of researchers and clinicians working with patients suffering from lupus (systemic lupus erythematosus). Using HL7 CDA documents produced at each participating research center, CG accepts those documents and stores them in the RIM model by means of the CG HL7 shredder. A repository with a data model derived from CGAM is populated with data from the CG RIM-based repository. DDQB is then configured to access the CGAM-based repository, thereby enabling broad, consistent, collaborative research and analysis against a larger, more statistically significant, community-wide patient cohort. This solution is a path to increased understanding of lupus, and it helps to advance the research and treatment of lupus in ways that were, until now, not possible.

■ IBM Clinical Genomics is based on a message-oriented architecture that supports several messaging standards for health-care data, including HL7 ■

CG, which was initially released in July 2004, was updated in September of 2006 with new technology that includes the HL7 RIM data model. CG is being used on an experimental basis by one health-care provider and will be deployed later in 2006 in several joint research projects involving a number of health-care research organizations and IBM.

CONCLUSION

Health-care organizations have a wealth of data that can help discover new treatments and improve patient care. Our goal is to enable the health-care industry to tap into this data by providing a solution for integrating data from various sources for the purpose of analytical studies. Our approach focuses on taking advantage of an industry standard in the development of a solution that is flexible, robust, and meets industry requirements.

RIM is an information model for health care that broadly covers all aspects of an organization's clinical and administrative information. Health-care industry experts created RIM to support the definition of messages used for exchanging health-care information. By deriving our physical data model from RIM, we are able to take advantage of the expertise of the HL7 organization and enable the

handling of HL7 V3 messages in our RIM-based database. Our design combines aspects of ER modeling with a hybrid-EAV modeling approach. The HL7 data types and the inheritance structure of the RIM classes required additional modeling considerations beyond standard ER modeling. After health-care data populates our database, the data can be restructured into specific data marts to serve the specific analytical needs of the user community.

The RIM-based data model is a core component of CG, a solution for integrating clinical and genomic data. IBM, working with the Lupus Biomarkers Working Group, has applied CG to the integration of lupus data from various sources. We expect CG to deliver value to other multicenter research projects as well as intra-institutional integration projects.

CITED REFERENCES

1. T. B. Pedersen and C. S. Jensen, "Research Issues in Clinical Data Warehousing," *Proceedings of the Tenth International Conference on Scientific and Statistical Database Management*, Capri, Italy, IEEE Computer Society (July 1998), pp. 43–52.
2. W. H. Inmon, *Building the Data Warehouse*, Fourth Edition, John Wiley and Sons, New York (2005).
3. W. H. Inmon, J. D. Welch, and K. L. Glassey, *Managing the Data Warehouse*, John Wiley and Sons, New York (1996).
4. L. Silverston, *The Data Model Resource Book, Vol. 2: A Library of Data Models for Specific Industries*, John Wiley and Sons, New York (March 2001).
5. IBM Clinical Genomics, IBM Corporation (August 2006), http://publib.boulder.ibm.com/infocenter/eserver/v1r2/topic/ddqb/eicavcg.htm.
6. J. Anhøj, "Generic Design of Web-Based Clinical Databases," *Journal of Medical Internet Research* **5**, No. 4 (2003). http://www.jmir.org/.
7. P. M. Nadkarni, C. Brandt, S. Frawley, F. G. Sayward, R. Einbinder, D. Zelterman, L. Schacter, and P. L. Miller, "Managing Attribute-Value Clinical Trials Data Using the ACT/DB Client-Server Database System," *Journal of the American Medical Informatics Association* **5**, No. 2, 139–151 (1998).
8. HL7 Reference Information Model, Health Level Seven, Inc., http://www.hl7.org/v3ballot/html/infrastructure/rim/rim.htm.
9. R. Elmasri and S. B. Navathe, *Fundamentals of Database Systems*, Addison Wesley Publishing, Reading, MA (2006).
10. Logical Observation Identifiers Names and Codes (LOINC), Regenstrief Institute Inc., http://www.regenstrief.org/loinc/.
11. E. Bellin, "EHIT Uses Data Synchronization to Improve Healthcare," *What Works*, Volume 17, The Data Warehousing Institute (May 2004). http://www.tdwi.org/.
12. SNOMED CT, SNOMED International, http://www.snomed.org/snomedct/index.html.
13. ICD-9 Provider and Diagnostic Codes, U.S. Department of Health and Human Services, http://www.cms.hhs.gov/ICD9ProviderDiagnosticCodes/.
14. R. Agrawal and C. Johnson, "Securing Electronic Health Records without Impeding the Flow of Information," *Proceedings of the International Medical Informatics Association Working Conference on Security in Health Information Systems*, Dijon, France (April 2006). http://www.almaden.ibm.com/software/projects/iis/hdb/Publications/papers/imia06health.pdf.
15. R. Spronk, "The Spine, an English National Programme," Ringholm GmbH (March 29, 2005), http://www.ringholm.de/docs/00970_en.htm.
16. Canadian Conceptual Health Data Model (CHDM), Canadian Institute for Health Information, http://www.cihi.ca/cihiweb/dispPage.jsp?cw_page=infostand_chdm_e.
17. Oracle Healthcare Transaction Base: Sharing Information for Better Healthcare, Oracle Corporation, http://www.oracle.com/industries/healthcare/oracle-healthcare-transaction-base-htb.pdf.
18. Healthcare Collaborative Network Solution, IBM Corporation, http://www.ibm.com/software/solutions/LE/LHX01–02/solutions_overview.html.
19. Data Discovery and Query Builder, IBM Corporation, http://www-03.ibm.com/servers/eserver/iseries/db2/ddqb.html.

Accepted for publication August 9, 2006.
Published online December 12, 2006.

Thomas J. Eggebraaten
IBM Systems and Technology Group, 3605 HWY 52 N, Rochester, MN 55901 (tegge@us.ibm.com). Mr. Eggebraaten is a software engineer in the Healthcare and Life Sciences development organization of the IBM Systems and Technology Group. Soon after receiving a B.S. degree in computer science from the University of North Dakota in 1999, he joined IBM in Rochester, Minnesota, where he contributed to the testing of products for the iSeries® platform, such as WebSphere® and DB2®. He is now developing tools that support collaborative disease-specific research projects involving multiple health-care providers. He is also involved in the IBM Clinical Genomics Solution and various other life-science projects. He is a member of the IBM invention review board in Rochester.

Jeffrey W. Tenner
IBM Systems and Technology Group, 3605 HWY 52 N, Rochester, MN 55901 (tenner@us.ibm.com). Mr. Tenner is a Senior Technical Staff Member in the Healthcare and Life Sciences development organization of the IBM Systems and Technology Group. He is also Chief Architect for the IBM/Mayo Clinic collaboration. He led the architecture and development of the IBM Clinical Genomics Solution and Data Discovery and Query Builder products. Previously, he held

technical leadership roles in the development of DB2® for iSeries® and was lead architect for the Domino® portfolio of products for iSeries. He holds several patents in database technology and is a member of the editorial board of the *IBM Systems Magazine*.

Joel C. Dubbels
IBM Systems and Technology Group, 3605 HWY 52 N, Rochester, MN 55901 (dubbels@us.ibm.com). Mr. Dubbels is a senior software engineer in the Healthcare and Life Sciences development organization of the IBM Systems and Technology Group. In this role, he is responsible for development of service-oriented software solutions applied to e-business and the health-care industry. His current work is focused on solutions for disease-specific research collaboration across multiple enterprises. His previous work included measurement, analysis, and modeling of systems performance, operating system development, object-oriented-software framework development, and management. ■