ELSEVIER

International Journal of Medical Informatics 65 (2002) 225–241

International Journal of Medical Informatics

www.elsevier.com/locate/ijmedinf

Metadata-driven creation of data marts from an EAV-modeled clinical research database

Cynthia A. Brandt *, Richard Morse, Keri Matthews, Kexin Sun, Aniruddha M. Deshpande, Rohit Gadagkar, Dorothy B. Cohen, Perry L. Miller, Prakash M. Nadkarni

Center for Medical Informatics, Yale University School of Medicine, P.O. Box 208009, New Haven, CT 06520-8009, USA

Received 11 July 2002; received in revised form 16 August 2002; accepted 3 September 2002

Abstract

Generic clinical study data management systems can record data on an arbitrary number of parameters in an arbitrary number of clinical studies without requiring modification of the database schema. They achieve this by using an Entity-Attribute-Value (EAV) model for clinical data. While very flexible for creating transaction-oriented systems for data entry and browsing of individual forms, EAV-modeled data is unsuitable for direct analytical processing, which is the focus of data marts. For this purpose, such data must be extracted and restructured appropriately. This paper describes how such a process, which is non-trivial and highly error prone if performed using non-systematic approaches, can be automated by judicious use of the study metadata—the descriptions of measured parameters and their higher-level grouping. The metadata, in addition to driving the process, is exported along with the data, in order to facilitate its human interpretation.
© 2002 Elsevier Science Ireland Ltd. All rights reserved.

Keywords: Data marts; Entity-Attribute-Value databases; Clinical study data management systems

1. Introduction

Clinical Study Data Management Systems (CSDMSs) are a class of software that support centralized management of data generated during the conduct of clinical studies. Commercial CSDMSs include Oracle Clinical, ClinTrial and MetaTrial. Such systems, which are typically deployed at an institutional or organizational level, must accommodate diverse types of data from different clinical domains that is generated by different groups of clinical investigators. Large-scale CSDMSs typically employ a high-end database engine that is usually accessed over an intranet or the Internet using web-based technologies. Their architecture is *generic* in that, over time, their data content can expand to hundreds of

* Corresponding author. Tel.: +1-203-764-6713; fax: +1-203-764-6717

E-mail address: cynthia.brandt@yale.edu (C.A. Brandt).

1386-5056/02/$ - see front matter © 2002 Elsevier Science Ireland Ltd. All rights reserved.
PII: S1386-5056(02)00047-3

المنارة للاستشارات

www.manaraa.

clinical studies, encompassing several thousand clinical parameters, without requiring continual redesign of the database schema. They achieve this capability by using a data model called Entity-Attribute-Value (EAV), which will be discussed shortly. In addition, they usually enforce data privacy by ensuring that individual users are authorized to view the data only for those studies to which they are granted permission: ideally, such users should not be aware of the existence of studies that do not concern them.

One of the challenges facing the administrators and users of generic CSDMSs is that the EAV model makes the task of extracting data into an intuitively viewable and/or analyzable form non-trivial. The overwhelming majority of requests for such data are study-specific: ideally, all of the data for a single study should be extractable into a 'data mart'—a database that can reside on individual users' desktop machines, so that they can apply to it the tools with which they are most comfortable and familiar. Further, it is not enough to extract only the data. To be interpretable, the data must be accompanied by *metadata*—descriptions of the individual data elements and their relationship to each other.

This paper describes how Trial/DB, an EAV-modeled CSDMS that is used at Yale and other collaborating institutions, including the National Cancer Institute (NCI) supported Cancer Genetic Network (CGN) [1], uses clinical study metadata to automate the creation of study-specific data marts. The lessons learned should be readily applicable to other CSDMSs.

2. Background

Much of the published informatics research deals with tools and methods for ad hoc querying clinical and hospital administrative databases by clinicians [2–5] or comparative evaluations of different query approaches [6]. Safran and Chute identified four purposes for querying clinical databases: results reporting, case finding, cohort description and to analyze data patterns in terms of trends or relationships [7]. The work of Deshpande et al. [8], which describes metadata-driven ad hoc querying of Trial/DB partially meets these purposes.

An alternative to building custom query tools to query clinical data repositories directly is to extract all or part of the data from these repositories into a 'data warehouse' or 'data mart' respectively, and then use third-party tools to query such data. Detailed descriptions of true clinical data warehouses, which store multiple kinds of clinical data within the same database, possibly combined with non-clinical data, have not been published so far. The Intermountain Health Care Data Warehouse Project deploys a number of separate data marts, each containing a particular category of data from their EAV-modeled clinical database [9].

Other developers of CSDMSs have described data retrieval and analysis systems that use specific statistical software programs such as STATA or SAS [10–13]. These systems focus on providing a means for researchers to perform their analysis with the help of statistical advice or predefined SAS procedures.

The Clinical Data Interchange Standards Consortium (CDISC) is a group that focuses on XML-based data-interchange standards for CSDMS interoperability. The CDISC Operational Data Model (ODM) specifies an interchange model for both data and metadata [14]. The ODM is not fully mature, and does not presently address many important issues: these omissions will be discussed at various points in the text. In any case, generating one or more streams of XML is

www.manaraa.

not enough: as we shall see, considerable work needs to be done to make such data usable by its intended audience.

3. The EAV data model: pros and cons

In clinical data, there are hundreds of thousands of potentially relevant parameters (attributes) on which data can be recorded for patients across all clinical domains. If each attribute were represented conventionally, i.e. as its own column in a table, one would need to create numerous tables. For an individual patient participating in a given study, only a few hundred or less attributes would be applicable, and the vast majority of columns would be empty. To deal with this situation, a data modeling approach called EAV is used [15–19]. An EAV table can be thought of (conceptually) as a table with three columns: Entity (a combination of patient ID, Study ID, one or more timestamps when a clinical event occurred), Attribute (the ID of the clinical parameter being recorded at that event) and the Value of the parameter. In CSDMSs, an entity represents a single *clinical event* for a patient in a study. Fig. 1 shows sample data in both conventional and EAV forms. The conventional relational table at the top left has two rows of data for entity #001 and one row for entity #002, which record the parameters WBC (white blood cell count), Glucose, HCT (hematocrit), Weight and Systolic blood pressure as individual columns. When this data is converted to the EAV structure, top right, there are nine rows of data for entity #001 and five for entity # 002. In reality, rather than recording text in the 'Attribute' column, one stores an Attribute ID: this references an Attribute Definitions table that serves as a controlled vocabulary. Similarly, the Entity ID references an Entity Information table that records associated details such as the patient ID and timestamps.

Theoretically, one could have a single EAV table for all types of data, which would be stored as strings, irrespective of their actual data type—string, integer, decimal numbers, dates, etc. Trial/DB, however, uses multiple EAV tables, one for each data type. Segregation by data type allows more compact storage as well as indexing by value. It also allows one to define attributes based on binary large object (BLOB) data, such as images and signal data such as EKGs, which are recorded and archived during clinical studies. (The CDISC ODM does not currently address storage and exchange of BLOB data.)

EAV-modeling is a form of 'row modeling', where each row in a table stores *one* fact. In conventional tables, with one *column* per attribute, by contrast, each row of data contains *many* facts, one per column. The advantage of the EAV model is that if a new parameter/attribute is added—e.g. a newly devised lab test—a database programmer does not need to modify the database schema by adding a new column to a table. Rather, a new entry is made into the Attribute Definitions table. The ID of this attribute is then used in the existing EAV table. The EAV schema provides robust support for interactive browsing or editing by users inspecting small amounts of data selected by patient and case report form (CRF). That is, it supports *transaction-oriented* operations well.

The EAV model, however, has drawbacks. Briefly, its suitability for *repository* purposes is achieved through a somewhat counter-intuitive physical structure that makes it unsuitable for *analytical* purposes. Analytical programs, e.g. statistical packages, expect to receive their data as one column per attribute. In database terms, the logical schema of an EAV system—the way the data is conceptualized by its users—is markedly different from

www.manaraa.

Sample Conventional Relational Data

Entity	WBC	Glucose	HCT	Weight	Systolic
001	1.5	130	32	230	95
001	1.6		33	220	80
002	5	100	40	150	110

Sample EAV Data

Entity	Attribute	Value
001	WBC	1.5
001	Glucose	130
001	HCT	32
001	WBC	1.6
001	HCT	33
002	WBC	5
002	Glucose	100
002	HCT	40
002	Weight	150
002	Systolic	110
001	Weight	220
001	Weight	230
001	Systolic	95
001	Systolic	80

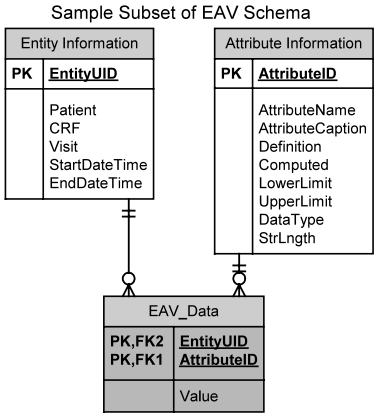

Fig. 1. This figure is a simplification of possible sample EAV data. The top left table shows a sample conventional table and the table on the top right is the same data represented as EAV. The EAV schema includes entity information such as patient, CRF, visit information, and date/times in entity tables and is linked to EAV_Data tables with many attribute value pairs. The Attribute information is stored in a separate table.

its physical schema—the way it is actually stored. As such, non-database experts cannot easily query EAV-modeled data. In particular, performing attribute-centric query, where one wishes to identify entities through complex Boolean criteria based on values of their attributes, requires queries based on set operations. Such queries, apart from being less easy to construct and poorly supported in third-party query tools because of their infrequent use in conventional scenarios, are also less efficient compared to queries on conventional tables [20].

Most analyses of CSDMS data are therefore best facilitated by bulk transformation of a suitable subset of that data into conventional format. The question is how to systematically perform such a transformation. We describe our approach below.

4. Using metadata with EAV

Users, like analytical programs, also take a conventional view of data: a usable EAV system must therefore create the user-interface illusion of conventional organization. This is achieved through metadata. In the account below, we will provide an overview of CSDMS metadata. We use the CDISC ODM terminology, though we will occasionally state where Trial/DB's metadata goes beyond this specification.

www.manaraa.

A study protocol consists of one or more Study Events: critical time points of the study during which CRFs (or simply, Forms) are administered to subjects. Trial/DB goes beyond the CDISC ODM in mandating that Study Events be given a serial number, which is optional in the ODM. In the vast majority of studies, Trial/DB Events must also be associated with a numerical value indicating their chronological offset from a 'time zero'.

A Form records the parameters that are measured in a subject. These parameters, or Items, may be grouped into logical categories called ItemGroups. For example, a 'Clinical Examination' form may be grouped into 'General Examination' and 'Physical Examination' parameters. Trial/DB's criterion for grouping Items into ItemGroups is fairly rigorous: a group of Items must have a common time-stamp that records when the facts represented in the ItemGroup actually occurred or were measured. Some ItemGroups represent clinical events with duration (e.g. adverse reaction to medications) and will have *two* timestamps, representing the start and end times. (The CDISC ODM does not currently address the temporal nature of clinical events.)

Sometimes, multiple instances of several items are recorded in a Form. For example, there may be multiple adverse events for a given patient/subject. For each event, one records the nature of the event, its severity, whether treatment was required, how long it lasted, and so on. A group of Items where multiple Item instances can occur is called a 'repeating' ItemGroup. In most cases, the number of instances is not known in advance. For example, some patients tolerate a drug well and have few or no adverse events, while others might experience numerous adverse effects.

The values of Items may be text, numbers, dates, and so on, but for certain Items, the values may be constrained by belonging to a CodeList. The latter is a set of discrete coded values (CodeListItems). CDISC only specifies that each CodeListItem shall have a value. In practice, however, it is useful to differentiate between the value that is recorded *internally*, which is typically a number, and a meaningful descriptive phrase that is *presented* to the user, e.g. in a pull-down list. For example, the response to the data item 'Result of Examination' might be the phrases 'Normal', 'Abnormal' or 'Not Done', which correspond to the internal values 0, 1 and 99, respectively.

Fig. 2 shows the correspondence between parts of a sample CRF and the corresponding metadata elements. Trial/DB goes beyond the CDISC requirements in mandating that, for a given study protocol, a given parameter/Item should be recorded in only a single place, i.e. in a single Form and within a single ItemGroup. When one is using a generic CSDMS to set up a study protocol, one is essentially simulating a relational DBMS design, and Trial/DB's restriction follows from the principles of normalized data design—that one fact shall, as far as possible, be recorded only in one place.

Fig. 3 shows part of the metadata schema of Trial/DB. (The terms Clusters, Question_Groups, Questions, Discrete_Value_Groups, Study_Phases are Trial/DB's internal terminology for CDISC's Forms, ItemGroups, Items, CodeLists and StudyEvents, respectively.) It is important to note that the same Form may be used in different Study Events, and that the same CodeList may be used in several Items, possibly across different forms. Because of these many-to-many relationships, when the metadata is exported into a data mart, it is most logically exported as a set of relational tables.

Trial/DB has additional metadata distinct from the CDISC model. For example, Inclusion and Exclusion criteria for a study are

www.manaraa.

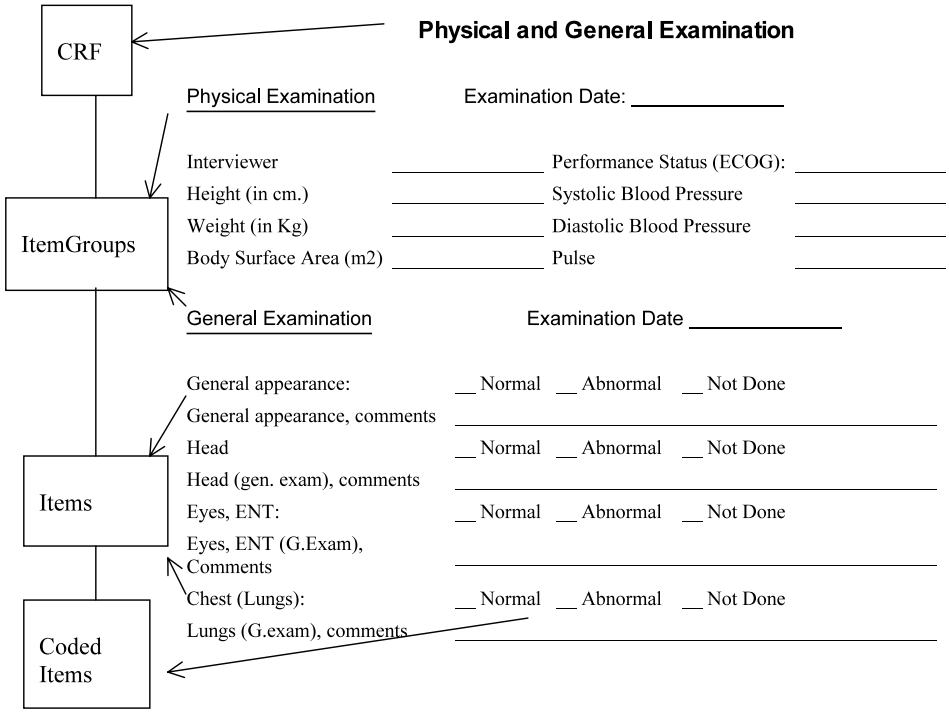

Fig. 2. Sample mapping of paper CRF (Physical and General Examination) into Trial/DB metadata, with groups (ItemGroups) of questions (Items) for physical examination and general examination.

treated as distinct from ordinary Items, because the responses to such criteria, phrased as questions, determines whether or not a given patient is eligible to participate in the study.

5. System objectives

The data export module of Trial/DB has several options, aimed at a variety of users:

Database analysts need to have the complete set of data generated in a study, including patient demographics, as well as BLOB data where applicable. For this purpose, it is necessary to create a data mart with a complete set of relational tables, with relationships between tables already set. The mart can then serve as the starting point for data exploration through creation of queries, some of whose definitions can be saved and reused, as well as reports. During the conduct of a study, the complete data may be exported several times. In this circumstance, one must preserve existing work (saved queries, reports) performed by the analyst. Therefore, creation of the output database schema must be separated from export of the data corresponding to those tables. This allows data in existing data marts to be purged and re-imported, while preserving the schema of the mart.

Statistical analysts need to reduce the number of steps required to prepare the data for analysis. It helps them if the data can be exported into the format used by particular statistical packages, with program scripts generated where appropriate—e.g. for SAS, PROC FORMAT scripts should be generated for variables that are based on CodeLists.

www.manaraa.

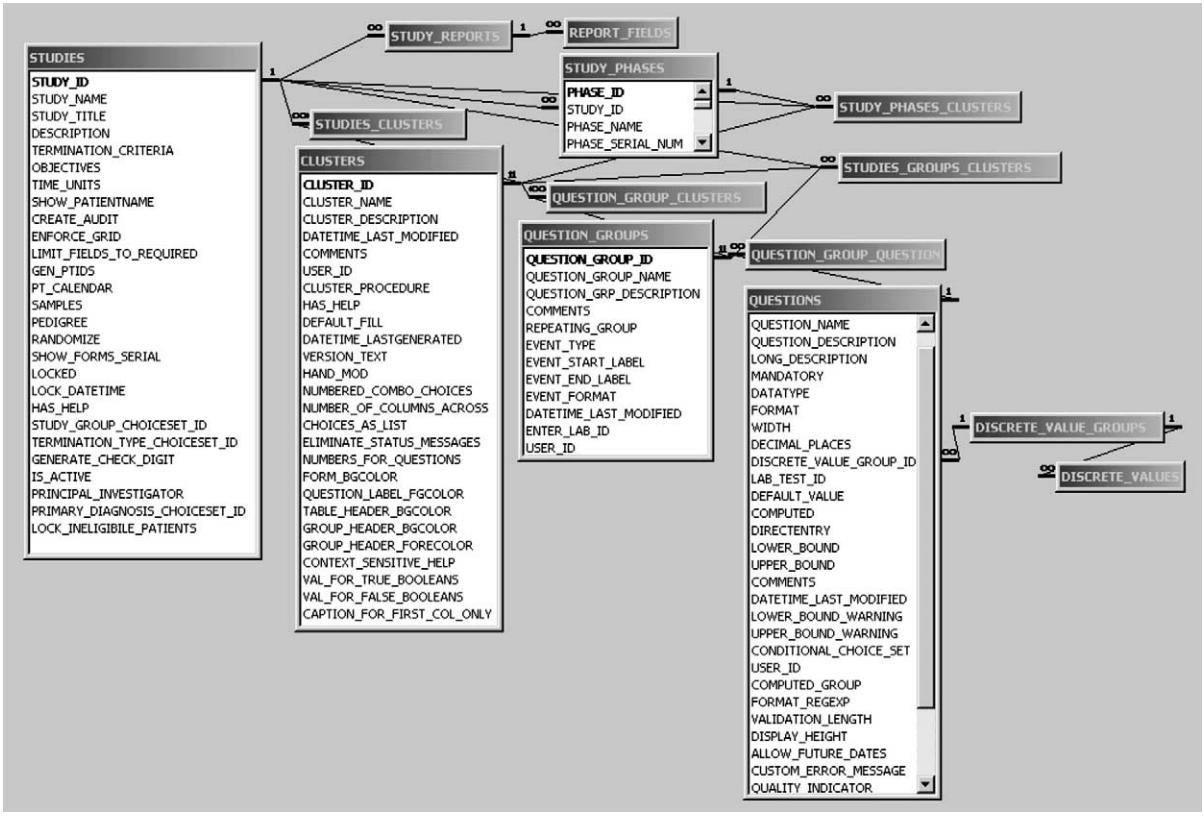

Fig. 3. Subset of metadata schema for the data mart. The STUDIES table stores metadata to describe the protocol, and STUDY_PHASES stores study event information. The bridge tables are minimized. The CLUSTERS table stores the CRF (Forms) information, QUESTION_GROUPS (ItemGroups), and QUESTIONS (Items). The DISCRETE_VALUES (CodedItems) is minimized.

Study administrators need to perform various monitoring tasks. For example, they may wish to determine adherence to the study protocol in terms of missed visits or missing data values, where the recording of particular parameters for a given Study Event is omitted partially or entirely. Another quality-control task is ensuring that data that has been gathered during individual Study Events is entered into the database in a timely fashion: in many cases, it is recorded on paper first and transcribed into the database later, but the time lag between the two should not be unacceptably long. For such users, standard reporting functionality must be bundled with the extracts: in other words, rather than just generate a set of data files, the system needs to be able to generate a mini-application with common functionality built in.

Advanced end-users need to be able to browse the data easily. When individual tables are inspected in spreadsheet mode or through forms, the English-language brief descriptions (captions) for individual parameters should be used for column headings by default, rather than the relatively cryptic column names that are used internally. For parameters based on CodeLists, the phrase for each Coded Item should be displayed rather than the numeric code: the latter should, by default, be hidden. Easy browsing is particularly helpful for quality-control tasks, such as identifying miss-

المنارة للاستشارات
www.manaraa.

ing data, which we shall discuss later. While easy browsing coupled with quality-control-monitoring reports assists data cleansing, cleansing should be performed on the source data rather than on the mart contents: otherwise, they would be lost with the next import of data.

Certain studies conducted by consortia are site-restricted: that is, most users are only permitted to look at data generated from their own study site. When such users do a 'complete' extract of the data, site restrictions must be enforced.

All groups of users need a data dictionary that is electronically searchable and cross-referenced. Thus, for example, given a parameter/Item that is identified by keyword search, one should be able to locate the table in which it lies. For this purpose, while the dictionary comprises a set of relational tables, it should be accompanied by user-interface components and code that facilitate its ready searching: once again, an application rather than data alone is what is needed. The data dictionary must not only contain information about Forms, ItemGroups, Items and CodeLists, but also about Study Events and which Forms are used in which Study Event.

Our experience has shown that the classes of users described above overlap significantly. For example, because the latest version of the widely used statistical package SAS has embraced SQL and the concept of relational databases wholeheartedly, most statisticians have to learn RDBMS technology, and they have found that microcomputer DBMSs such as MS-Access provide a relatively painless introduction. Further, data browsing almost invariably precedes analysis.

We determined that all of the above requirements could be met most effectively by generating a complete microcomputer DBMS application fully populated with related tables and the appropriate user-interface elements and code libraries. We chose MS-Access as the delivery platform because of its robustness as a standalone relational DBMS, ease of learning and use by non-database experts, and our extensive experience in generating applications that are based on it.

6. System description

We now describe how we use research study metadata to transform the contents of a Trial/DB study into conventional form.

7. Principles of data transformation

During conduct of a long-term study, certain data parameters/Items are recorded multiple times. Further, for reasons of cost, inconvenience and risk to the patient, individual Items are sampled at unequal frequency. For example, a liver biopsy or an MRI would be repeated much less frequently than a physical examination or standard questionnaire. Therefore, it is not possible in general to extract all of a study's data into the equivalent of a *single* spreadsheet, with one row per Study Event per Patient: doing so would result in 'missing' values for certain Items that are an artifact of the sampling process. Further, for repeating ItemGroups such as Adverse Events, the number of items per patient is highly variable and not known in advance. In other words, the output of transformation must be *multiple* tables. In the simplest case, therefore, one would create one relational table per Form. If however, a Form contains one or more repeating ItemGroups, the data for each such ItemGroup would need to go into its own table.

One can programmatically inspect the list of Forms across all Study Events and detect that certain Forms always go together for a

www.manaraa.

particular study. For example, along with a blood draw for Clinical Chemistry, one may also be administering a particular standard questionnaire. Such Forms are said to co-occur. Non-repeating Items on co-occurring forms may be combined into a single table, subject to database engine limits on the maximum number of columns per table. (Just because the number of output tables can be reduced does not mean that it should be. In general, this decision should be left to the user: a single table that consists of a very large number of columns of functionally unrelated parameters is not necessarily easier to browse than several smaller tables.) In any case, algorithmic inspection of the study data leads to a straightforward determination of how many output tables must be created, as well as the structure of each table in terms of which Items it is comprised of. Since we know the data type of each Item, it is possible to generate SQL (a CREATE TABLE statement) that defines each output table.

For extracts that serve as the basis for formatted reports created with third-party reporting tools, one may pick a subset of Items present in either a single Form or a set of co-occurring Forms. (Non-EAV data elements such as Patient Demographics and Enrollment occur just once per patient: such elements may be considered to co-occur with any other elements, and can be combined with them if desired.) These 'predefined extracts' are the analogs of views in a conventional database. During the early development of Trial/DB, predefined extracts were the initial means of data extraction: they have been briefly described previously [21]. The previously described algorithm is much slower than the one used at present, which will be described shortly. Currently, with the ability to bulk-extract the contents of the entire study, the use of predefined extracts has diminished greatly. However, as discussed later, they are still used to export data to particular statistical packages, because the description of a predefined extract includes certain metadata elements that are not recorded elsewhere.

Finally, one can use a static or dynamic approach to creating data extracts. In the static approach, one goes through the metadata to create 'compiled' definitions for one or more output tables. Predefined extracts use a static approach. Alternatively, the program can build such definitions on the fly each time. Experience showed that the use of predefined extracts for data was cumbersome for researchers who simply wanted all their data for a study exported. Another problem with predefined extracts was that, if the study design was altered during the course of the study by inclusion of new Forms or addition of new Items to a Form, certain extracts became out of date and had to be re-created. Therefore, even though recreating output table definitions anew takes a modest amount of additional time, we now prefer the dynamic approach to extract creation: most users wish to have all their data in one set rather than as multiple text files, and do not care to deal with changes in the study metadata manually.

8. Mechanics of data extraction into output tables

Prior to actually exporting the data, the user can filter it through such options as limiting data to a subgroup of patients that have been predefined (such as all the control subjects).

For a given output table, we know the list of all Items/parameters in that table. Further, the Entity columns for all data rows are tagged with the Study ID. It is therefore straightforward to gather all EAV rows where the Entities correspond to a particular Study

www.manaraa.

ID, and where the Attribute corresponds to a particular Item ID. We do this for all Items in the table, loading the values into a two-dimensional array in memory, where the columns correspond to the Items and the rows correspond to Entities. The process of assigning an arbitrary value to its corresponding row and column is facilitated using a data structure called a dictionary, also known as a hash table [22]. Missing values will result in empty array cells. The movement of data into the array essentially constitutes a row-to-column transformation, and the contents of this array are then written out to disk as a tab-delimited file, that is destined for import into the corresponding relational table. This process is repeated for every output table that is to be created.

Additional tables are created from conventionally structured data such as Patient Demographics/Enrollment data, by filtering on Study ID. Here, the output tables generated are simply miniature versions of the corresponding tables in Trial/DB. Setting of Primary-key/foreign-key relationships between the various output tables is straightforward. Every output table corresponding to a Form has at least two foreign keys: the Patient ID, which references the Demographics/Enrollment table, and the Study Event ID, which references the Study Event Metadata table. Tables generated from repeating ItemGroups are automatically related many-to-one to the table representing their enclosing Form.

Metadata is exported by starting with the definition of the Study Events comprising the study protocol/s, and then going on to the definitions of the Forms, ItemGroups, Items, CodeLists and Coded Items used in this study. Numerous Trial/DB-specific metadata elements, such as definition of inclusion and exclusion criteria for the study, are also exported. The list of such metadata elements has evolved with time as Trial/DB's features have evolved, and as our users have become more sophisticated. To avoid hard-coding the list of exported metadata elements within our subroutines, Trial/DB maintains metadata describing the metadata elements ('meta-metadata') This includes documentation about how particular elements are to be used by the study designer, as well as a Boolean flag indicating which metadata is 'exportable'. Our long-term objective here is that the Trial/DB application should contain as much of its own documentation as possible.

BLOBs such as arbitrary sized text, images and signal data are extracted using function calls to a custom software library (an 'ActiveX Control') into a local folder, with one disk file per BLOB. Each file is given an extension based on the type of data that it contains: in Trial/DB, the extension is stored as part of the metadata. In output tables that contain BLOB columns, we store the pathname in these columns rather than the data itself. These paths then serve as 'hyperlinks' that let the BLOB be viewed. When the data mart is created in Microsoft Access, the task of viewing such files is effortless and requires no special code on our part: After the user clicks on the hyperlink representing that file, Access simply opens that file and launches the appropriate default viewer that has been registered for that file type on that particular CPU. For example, JPEG files might be viewed using Adobe PHOTOSHOP or the Microsoft PHOTO EDITOR.

In situations where a complete data mart must be generated, the final step is to export certain standard predefined, parameterized queries, forms, reports and their associated code into the mart so as to create a full-fledged application. This provides added functionality such as the ability to browse the metadata.

The above code is CPU and memory-intensive—the machine performing the data

www.manaraa.

extraction is typically equipped with at least 1 GB of RAM—but economical of network traffic and DBMS resources. It minimizes the number of SQL statements that are issued to the database server, with each statement returning a relatively large amount of data that is then arranged in memory. The programming toolkit that we use, Microsoft ActiveX Data Objects (ADO) has a highly efficient function called getRows() for this purpose, which implements a 'fire-hose cursor', so-called because of the relatively high rate of data transfer.

Even with the optimization described above, extracting the complete data for a large study from our production CPU can impact response time for other concurrent users of Trial/DB who may be in the middle of performing interactive data entry. Because data entry takes highest priority, we schedule extraction tasks to run in batch-mode at off-peak-usage hours (i.e. between 11 pm and 6 am). The Web interface for data extraction, in fact, merely places the task in a queue: e-mail sent to the requester the next morning contains a hyperlink that lets the user download the requested data from a password-protected site.

The MS Access client can create a data mart for a typical study in about 15–16 min by retrieving data from the Oracle database over the network. The same study extraction performed on a machine that contains the Oracle database and the MS Access client, takes 4–5 min. The typical data mart contains about 80–90 data and metadata tables, and ranges from 3 to 5 megabytes in size. These measurements were performed on a typical study that contains over 70 patients, 25 CRFs and about 500 data items per patient. The data mart for the largest study in Trial/DB is about 22 megabytes in size and takes about 20–25 min to run on the stand-alone setup and over 35 min on the production system over the network.

We are in the process of creating a data warehouse for Trial/DB on a CPU that is separate from the production CPU, which will receive data dumps from the production system on a nightly basis. Requests for data extracts will be directed to the warehouse server, and result in fast response times, with the caveat that the data received is at most a day old.

9. The system in operation

Fig. 4 shows the interface in use for a study on an ongoing survey of breast cancer patients. (This interface illustrates the use of an MS-Access client program: similar functionality is available through a Web interface.) As shown, 13 extracts have been selected for export into an existing MS Access data mart (BrTest.mdb file).

Fig. 5 illustrates the Data Dictionary Browser, which is part of the generated data mart's functionality. This browser lets the user view table definitions and fields or questions within tables. In the figure, a list of columns is displayed for a Breast Cancer Pathology Reports table. The definition of a particular selected column based on a CodeList (Breast_CA_Margins) is also shown.

9.1. Exporting data to statistical packages

Newer versions of most statistics packages allow direct access to RDBMSs through ODBC (Open Database Connectivity) drivers. However, ODBC does not let the statistics package infer important facts about columns in a relational table—for example, that the values of a particular column are based on a CodeList. In general, exporting data to a statistical package requires one to provide,

www.manaraa.

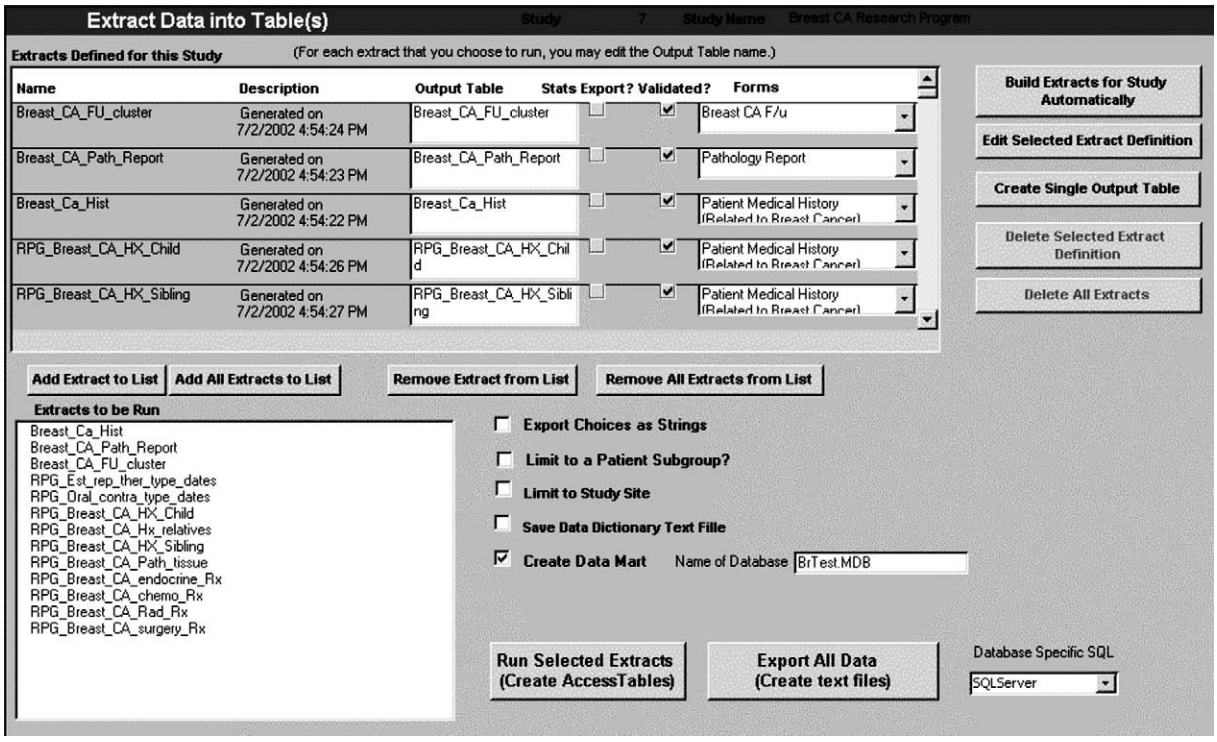

Fig. 4. Interface for creating data mart(s). The user selects specific sets of data (extracts) to export data for, checks various filtering options, selects the data mart to export the data into then clicks the button as appropriate to 'Run Selected Extracts' or 'Export All Data' as simple text files and the SQL for table creation for SQL Server or Oracle tables.

for each column, its name, data type, brief definition/caption, and set of values (if based on a CodeList). Many statistics packages, however, have limitations that interfere with seamless data interchange. For example, SAS is currently the only package that fully understands the concept of a database as consisting of multiple tables, and which allows the statistical programmer to fully utilize the SQL language to access more than one table. SPSS, while much easier to use for the non-expert, has a more restricted view of databases: each table requires its own 'data dictionary', and variable names cannot exceed eight characters in length—an anomaly in an era where most column names in relational databases are quite long in order to be self-documenting. (Oracle names, for example, can be up to 30 characters long, while MS SQL Server has a limit of 128.) Such idiosyncrasies must be taken into account based on the package one is exporting to.

When exporting to SAS, we create one 'library' (data dictionary) with all the required definitions. This dictionary, once created, can be re-used later if a new batch of data is to be exported, as long as the metadata has not changed. Export to SPSS requires the creation of pre-defined extracts with short names for each parameter/Items to be exported. The reason for creating predefined extracts is based on our conscious decision that 8-character short names for Items, which are idiosyncratic to the current version of SPSS, should not be part of an Item's definition, but should be stored with the extract definition, if needed. If subsequent versions of SPSS accommodate variable names of reasonable length,

المنارة للاستشارات
www.manaraa.

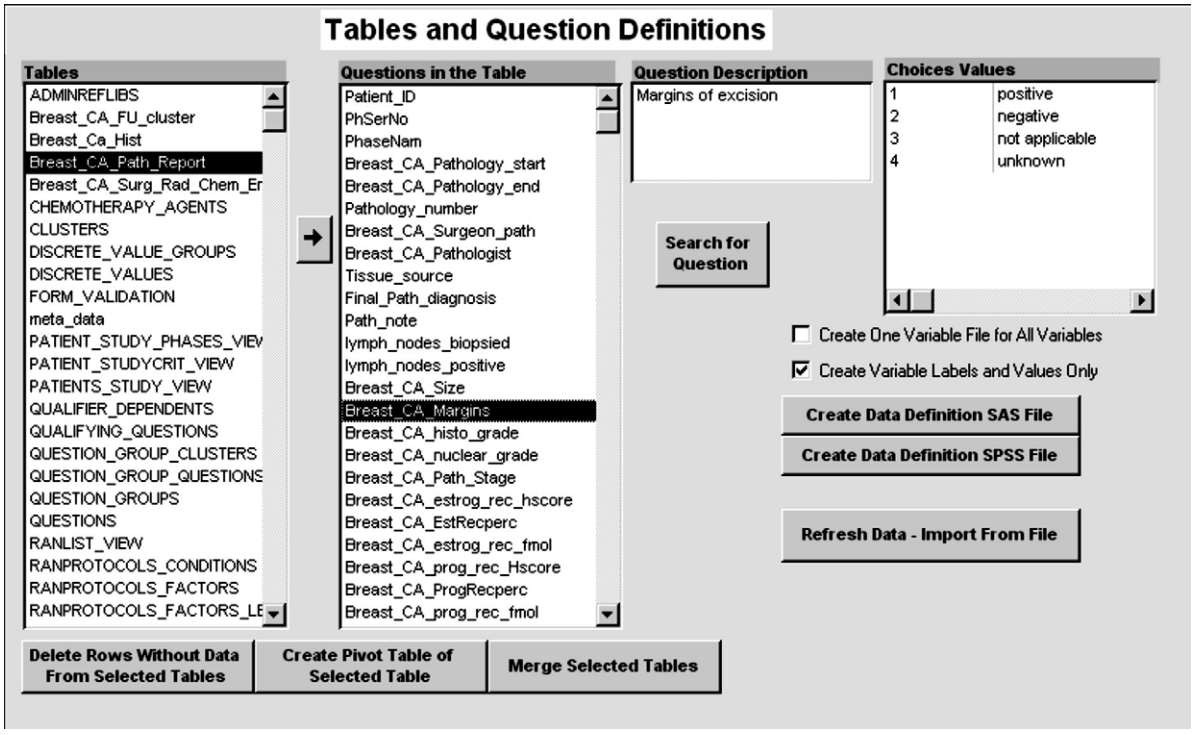

Fig. 5. Form for Metadata Viewing, Table Modification, Data Import and Export. This form is available in the data mart for use by researchers in order to view tables and questions, search for questions, manipulate tables, refresh data (from external sources) and to create statistical data definition files. The table Breast_CA_Path_report has been selected with the Breast_CA_Margins field selected, showing its description, Margins of excision, and the set list of choice values 1, positive ; 2, negative; 3, not applicable, and 4, unknown.

we will be able to dispense with predefined extracts for this purpose entirely.

Fig. 6 shows a sample of the data definition files created for SAS and SPSS. The top of the figure illustrates the simple PROC FORMAT text from SAS and comparable data label definitions on the bottom from SPSS for some of the fields from a table of Breast Cancer Pathology Reports.

9.2. De-normalizing data

In certain study designs, there are repeating ItemGroups where the number of repeats is known in advance. For example, in a long-term study of tamoxifen, an anti-estrogen used in breast cancer patients, a patient's bone mineral density (BMD) and high-density lipoprotein cholesterol (HDLC) may be measured at baseline, at 12 months and at 24 months. Such data is typically analyzed using a General Linear Model with repeated measures. In some statistical packages such as SPSS, this requires the creation of a de-normalized data set with three sets of variables, BMD0, BMD12, BMD24, and HDLC0, HDLC12 and HDLC24, which can then be contrasted with each other. (Creating columns with such names is anathema to RDBMS designers—such a design violates E.F. Codd's first rule of database normalization, 'no repeating values'[23].) Our program allows conversion of data in individual tables into such a de-normalized form if required, through a row-to-column transformation of the data.

www.manaraa.

```
PROC FORMAT;
VALUE Breast_CA_M_Stage 1 ='MX Presence of distant metastasis cannot be asse
 2 ='M0 No distant metastasis'
 3 ='M1 Distant metastasis (includes metastasis to ipsilateral supraclavicul
VALUE Breast_CA_Margins 1 ='positive'
 2 ='negative'
 3 ='not applicable'
 4 ='unknown';
VALUE Breast_CA_N_Stage 1 ='NX Regional lymph nodes cannot be assessed'
 2 ='N0 No regional lymph nodes metastasis'
 3 ='N1 Metastasis to movable ipsilateral axillary lymph node(s)'
 4 ='N2 Metastasis to ipsilateral axillary lymph node(s) fixed to one anothe
 5 ='N3 Metastasis to ipsilateral internal mammary lymph node(s)';
VALUE Breast_CA_Path_Stage 3 ='I invasive, LN negative'
 4 ='II  invasive, LN positive'
 5 ='IIIA locally advanced, resectable'
 6 ='IIIB locally advanced, non-resectable'
 7 ='IV metastatic'
 2 ='0 in situ carcinoma'
 1 ='NED'
 8 ='Unknown';
VALUE Breast_CA_T_Stage 1 ='TX Primary tumor cannot be assessed'
 2 ='T0 No evidence of primary tumor'
 3 ='Tis Carcinoma in situ: intraductal CA, lobular CA in situ, or Pagets di
 4 ='T1 Tumor 2 cm or less in greatest dimension'
 5 ='T2 Tumor more than 2 cm but not more than 5 cm'
 6 ='T3 Tumor more than 5 cm in greatest dimension'
```

```
VALUE LABELS
 M_Stage 1 'MX Presence of distant metastasis cannot be assessed'  2 'M0 No d
VALUE LABELS
 Margins 1 'positive'  2 'negative'  3 'not applicable'  4 'unknown'.
VALUE LABELS
 N_Stage 1 'NX Regional lymph nodes cannot be assessed'  2 'N0 No regional ly
VALUE LABELS
 Stage 3 'I invasive, LN negative'  4 'II  invasive, LN positive'  5 'IIIA lo
VALUE LABELS
 T_Stage 1 'TX Primary tumor cannot be assessed'  2 'T0 No evidence of primar
VALUE LABELS
 histo_gr 1 'I'  2 'II'  3 'III'.
```

Fig. 6. The text above shows two versions of the data dictionary definition files. The top section is the SAS PROC FORMAT version, the bottom is how the same formatting is done in SPSS with VALUE LABELS.

10. Current status and future direction

The Trial/DB tablespace in Oracle is over 300 megabytes in size. It contains data for over 4000 patients and 90 studies (since 1997). There are over 1.4 million EAV rows of data with the EAV integer table containing over a million rows followed by the EAV real and string tables. Several researchers and biostatisticians who are using Trial/DB are using the software described above: its feature set has evolved over time, and has been based on their continual inputs. Many of these users originally used data extracts corresponding to particular tables, but more of them are now requesting complete study exports and are beginning to use the data dictionary SPSS/SAS creation tools.

We have already discussed our intentions with respect to creating a read-only data warehouse to facilitate the extraction process. This warehouse is intended to contain pre-joined data tables, where the type-specific clinical data tables in Trial/DB will be physically combined with higher-level tables: it will still remain in EAV form, since it is intended to be cross-study. Such tables take up more space because the values of certain columns are recorded redundantly, but yield higher query performance. We will need to conduct benchmarking studies to determine the optimum level of pre-joining.

المنارة للاستشارات
www.manaraa.

We have not yet received requests for supporting data/metadata export into CDISC XML format. It is, however, reasonably straightforward to perform such export.

11. Discussion and conclusion

We will discuss some of the informatics issues and lessons we have learned in addressing the need of researchers to access data from an EAV-structured data repository.

12. The importance of independent data marts

For a given study, the complexity of data preparation increases with the complexity of the study protocol/s: certain studies in Trial/DB with numerous CRFs, for example, yield more than 50 data tables. The task of generating these tables clearly benefits from automation, and a relational data mart that inter-relates these tables is the natural store for such data. Further, such a data mart can serve as a starting point for a dataset to which value can be added, for example, by integrating additional information from software that assists study administration (e.g. tracking functions), as well as and data collected from legacy systems [24].

Such data marts can reside on the desktop computers of individual data analysts, forming 'personal' databases that facilitate data exploration [25]. Their use on standalone machines alleviates the load on the server that supports data entry, as well as decreases network traffic. The robustness of microcomputer DBMSs, which can manage significant data volumes, coupled with their ease of learning and use, allows users to use them to perform some of the work traditionally performed in statistical packages.

13. Data mart architecture

Data marts are widely used for on-line analytical processing (OLAP). One approach that has been widely used for OLAP is the deployment of multidimensional database (MDDB) technology [26]. Such database engines essentially operate by treating a particular data set as a multi-dimensional 'hypercube' whose axes are the parameters whose effect one seeks to measure. As an example, a hypercube containing adverse event data would have the following axes: nature of medication, patient, investigator, cumulative duration of therapy, dose of medication, and so on. MDDBs achieve high performance by pre-computing aggregates on multiple axes, so that, if a researcher wants a breakdown of number and severity of adverse events by drug category, that information has been pre-computed and may simply be looked up. MDDBs can, through pre-computation achieve an order of magnitude better performance than relational databases.

As we have stated before, the data marts that we generate are conventional relational databases. This is not necessarily a drawback. Bill Inmon et al., in their text, 'Data Warehouse Performance', [27], describe two different kinds of schema architectures, which are intended for two different kinds of users who they term 'farmers' and 'explorers'. The 'farmers' are concerned with producing standard reports on well-understood data sets: these reports must run as fast as possible, because they are run several times a day. 'Explorers' are concerned with discerning patterns in data and performing analyses whose exact nature is not determined in advance. Inmon et al. state that MDDBs represent an optimized structure that is useful for the 'farmers': the needs of 'explorers', by contrast, are better met by normalized relational structures. It is possible, however, that once data exploration has

www.manaraa.

identified particular characteristics of the data that particular critical reports are needed on a routine basis, some of the data may be exported to an MDDB. For the modest volumes of data that typically result from a single clinical study, however, the difference in performance between relational DBMSs and MDDBs is not dramatic.

Availability of Software: Trial/DB is open-source and is available on making a written request to either Dr Brandt or Dr Nadkarni.

Acknowledgements

Michael Reiss, now at the University of New Jersey/Robert Wood Johnson Medical School, is the principal investigator of the Breast Cancer Survey study. Timothy Rebbeck of the University of Pennsylvania provided valuable feedback that led to improvement of the software. *Grant Support:* NIH Grants K23 RR16042, M01 RR06022, G08 LM05583, T15 LM07056, R01 LM06843-01, U01 CA78266 and U01 ES10867.

References

[1] Cancer Genetics Network, NCI, National Cancer Institute, available from http://epi.grants.cancer.gov/CGN/, last accessed on 6/11/2002.
[2] D.J. Nigrin, I.S. Kohane, Data mining by clinicians, Proceedings of the AMIA Symposium, 1998 pp. 957–961.
[3] A. Wilcox, G. Hripcsak, C. Chen, Creating an environment for linking knowledge-based systems to a clinical database: a suite of tools, Proceedings of the AMIA Annual Fall Symposium, 1997 pp. 303–307.
[4] A. Wilcox, G. Hripcsak, CPMC Query Builder. 1998, available from http://www.cpmc.columbia.edu/arden/qbr/.
[5] F. Banhart, H. Klaeren, A graphical query generator for clinical research databases, Methods Inf. Med. 34 (4) (1995) 328–339.
[6] G. Hripcsak, et al., Access to data: comparing AccessMed with Query by Review, J. Am. Med. Inform. Assoc. 3 (4) (1996) 288–299.
[7] C. Safran, C.G. Chute, Exploration and exploitation of clinical databases, Int. J. Biomed. Comput. 39 (1) (1995) 151–156.
[8] A. Deshpande, C. Brandt, P. Nadkarni, Metadata-driven ad hoc query of patient data: meeting the needs of clinical studies, J. Am. Med. Info. Assoc. 9 (4) (2002) 369–382.
[9] P. Wang, et al., The Web enabled IHC Enterprise Data Warehouse for clinical process improvement and outcomes measurement, in: 1997 AMIA Fall Symposium. 1997. Hanley & Belfus, Philadelphia, PA, Nashville, TN.
[10] W. Dorda, et al., ArchiMed: a medical information and retrieval system, Meth. Inf. Med. 38 (1) (1999) 16–24.
[11] W. Gall, H. Heinzl, P. Sachs, Extracting a statistical data matrix from electronic patient records, Comput. Meth. Prog. Biomed. 66 (2–3) (2001) 153–166.
[12] A. Gouveia-Oliveira, Salgado N.C., A database system for integrated clinical trial management, control, statistical analysis and ICH-compliant reporting, Proceedings of the AMIA Symposium, 1999, pp. 72–76.
[13] N.C. Salgado, A. Gouveia-Oliveira, Towards a common framework for clinical trials information systems, Proceedings of the AMIA Symposium, 2000 pp. 754–758.
[14] Clinical Data Interchange Standards Consortium, C.D.I.S., Specification for the Operational Data Model (ODM) 2002, available from http://www.cdisc.org-/models/odm/v1.1/odm1-1-0.html, Last accessed on: July 1,2002.
[15] 3M Health Information Systems, The 3M Clinical Data Repository. 1998: Murray, UT, USA, available from http://www.3m.com/market/healthcare/his/us/products/care_inno/.
[16] S.M. Huff, et al., Evaluation of a SQL Model of the Help Patient Database, in: Proceedings of the 15th Symposium on Computer Applications in Medical Care 1991, IEEE Computer Press, Los Alamitos, CA, Washington, DC.
[17] S.M. Huff, et al., HELP the next generation: a new client-server architecture, in: Proceedings of the 18th Symposium on Computer Applications in Medical Care 1994, IEEE Computer Press, Los Alamitos, CA, Washington, DC.
[18] S. Johnson, et al., Using metadata to integrate medical knowledge in a clinical information system, in: Proceedings of the 14th Symposium on Computer Applications in Medical Care 1990., IEEE Computer Press, Los Alamitos, CA, Washington, DC.
[19] C. Friedman, et al., A generalized relational schema for an integrated clinical patient database, in: Proceedings of the 14th Symposium on Computer Applications in Medical Care 1990, IEEE Computer Press, Los Alamitos, CA, Washington, DC.
[20] R.S. Chen, et al., Exploring performance issues for a clinical database organized using an entity-attribute-value representation, J. Am. Med. Inform. Assoc. 7 (5) (2000) 475–487.
[21] P. Nadkarni, C. Brandt, Data extraction and ad hoc query of an Entity-Attribute-Value Database, J. Am. Med. Info. Assoc. 5 (6) (1998) 511–527.

www.manaraa.

[22] R. Sedgewick, Algorithms in C++, third ed., Addison-Wesley, Reading, MA, 2002.
[23] C.J. Date, An Introduction to Database Systems, ninth ed., Addison-Wesley, Reading, MA, 2001.
[24] S. Ruberg, M. McDonald, M. Wolfred, A proposal and challenge for a new approach to integrated electronic solutions, Appl. Clin. Trials February (2002) 42–49.
[25] W. d'Hollosy, et al., Semi-automated database design by the end-user, Meth. Inf. Med. 34 (3) (1995) 266–271.
[26] E. Thomsen, OLAP Solutions: Building Multidimensional Information Systems, Wiley, New York, NY, 2000.
[27] W. Inmon, K. Rudin, C. Buss, W. Anmon, R. Sousa, Data Warehouse Performance, Wiley, New York, 1998.

المنارة للاستشارات
www.manaraa.