Converting a Legacy System Database into Relational Format to Enhance Query Efficiency

Jonathan C. Prather, B.S.[1], David F. Lobach, M.D.,Ph.D.,M.S.[1], Joseph W. Hales, Ph.D.[1], Marvin L. Hage, M.D.[2], Seth J. Fehrs, B.S.E.[1], and William E. Hammond, Ph.D.[1]

Duke University Medical Center, Durham, North Carolina
[1]Division of Medical Informatics
[2]Department of Obstetrics and Gynecology

The analysis of clinical data collected over time can provide important insight into the health care process. Unfortunately, much of the electronic clinical data that exists today is stored in legacy systems, making it difficult to access and share the information. An approach is needed to improve the accessibility of electronic data stored in legacy system databases. In this study, a legacy database is converted into a relational format in the personal computer environment. The impact of such a conversion on query performance is evaluated, and issues that need to be considered when converting a legacy system database are identified.

INTRODUCTION

With advances in information technology, access to clinical data is becoming increasingly important both for evaluating health care delivery and for setting health care policy [1-6]. Clinical data that has been accumulated over time is particularly valuable for monitoring trends in health care delivery, for assessing the effectiveness of medical therapies, and for evaluating outcomes. Unfortunately, much of this long term clinical data has been collected in systems with proprietary database structures that make data accessibility and portability difficult. To make this valuable data easily and economically accessible, alternative methods are needed to query the data. One approach to improve accessibility is to convert the proprietary database into a relational format on commercially available database software which supports SQL or other ad-hoc querying mechanism.

The potential advantages of using commercial database software on a personal computer (PC) include improved query performance, access to standard ad-hoc query tools, enhanced documentation, and the ability to export query results to a statistical software package [7]. In the past, converting an entire legacy system to the PC environment was constrained by technological limitations both in commercial database software and in PC hardware. Conversion to the PC environment is now potentially feasible due to recent advances in search engines for relational database software and improvements in the processing power and storage capacity of PCs.

Little work has been published that addresses the conversion of patient databases from legacy systems to increase query performance. Huff et al. [7] converted a portion of a proprietary clinical database to a relational format. However, no attempt was made to convert the entire legacy database or to normalize the relational representation. This conversion also depended on main-frame technology.

The purpose of this project is to demonstrate that a large legacy database could be successfully converted to a relational format in the PC environment. In this paper we describe the conversion of the database from TMR® (The Medical Record) into three commercial relational database packages. We also report the results of our evaluation of query speed, single record retrieval time, and disk storage requirements using the relational database packages and the VAX-based legacy system. Finally, we discuss several issues that need to be considered in converting legacy databases to relational databases in the PC environment.

METHODS

Computer Based Patient Record System

TMR is a comprehensive longitudinal computer-based patient record system (CPRS) developed at Duke University over the last 25 years. While the functionality of TMR continues to evolve, the underlying data structure has remained fundamentally the same [8]. The existing database structure provides efficient and secure storage; however, it makes performing complex queries slow and labor-intensive. In short, the existing TMR query system does not meet current research needs. Consequently, a customized program needs to be written to extract the data for each query.

The TMR database selected for conversion in this project was the perinatal database used by the Department of Obstetrics and Gynecology at Duke University Medical Center [9]. This database serves as the repository for a regional perinatal computerized

0195-4210/95/$5.00

Table 1. Example nodes from a patient record with two TMR problem data blocks.

Description of Node	Node Number	Actual Stored Data
patient identifier	1	555-55-5555
problem code	170	60
modifier	171	marginal- f/u 24-26 wks
onset date	177	93130
date resolved	178	93181
problem code	180	707
modifier	181	transverse abd incision
onset date	187	93181
date resolved	188	93181
recurring dates	189	91130-91130

patient record that is used in both inpatient and outpatient settings. The current Duke perinatal database contains comprehensive data on 40,854 patients seen in the department since 1988. The data collected includes demographics, registration, study results, problems, therapies, subjective and physical findings (SAPs), and encounter summaries.

TMR has a proprietary data structure. This structure employs a modular approach to store all of the patient's information in a single record. An individual TMR patient record is composed of variable length delimited character strings called nodes. While this data structure is flexible and easy to expand, it is not a standardized database management system [10]. Retrieval of TMR data is only accomplished by using application program code written in GEMISCH, the native programming language for TMR [8].

Conversion to Relation Format

The primary TMR record was mapped and exported in ASCII into relational tables in third normal form. These tables formed the relational databases on the PC and contained 84 entities and 460 attributes. Lengths for the fields in the relational tables were not assigned before the data was imported. The longest data element of each field in the database was automatically used by the relational software for the field size in the relational tables.

An example of how the data was mapped from the TMR database into relation tables is shown in Tables 1 and 2. TMR, as a problem-oriented CPRS, focuses on tracking patient problems. Each problem is assigned 10 nodes in the patient record. These nodes store the problem code and the associated descriptors which include a free-text modifier, onset date, resolved date, recurring dates, certainty factor, a DUSOI (Duke University Severity of Illness) score, causes, and manifestations. Table 1 shows patient data contained from two problem entries in a TMR patient record. These two entries mapped into three relational records (Table 2). The lack of one-to-one mapping occurred because the second problem occurred on two separate dates.

Three off-the-shelf PC relational database packages were used in the project: Borland Paradox® for DOS v 4.0, Borland Paradox® for Windows v 1.0, and Microsoft Access® v 2.0. These packages were chosen because each package had features to import and export ASCII tables with the delimiter of choice. Stand-alone versions of these database packages were installed on a PC with a 60 MHz Pentium CPU, 1700 megabytes of hard disk, 16 megabytes of RAM, and using the Windows NT™ 3.5 operating system and file system. To avoid confusion with the many database files involved, a relative addressing scheme to group related files in separate directories was utilized for each relational database.

Evaluation Methods

To evaluate query speeds, three queries of increasing complexity were constructed to run on the four systems. A "first order" query was defined as a query which involved data contained in only one table; a "second order" query involved data contained in two tables that

Table 2. Example relational records created by the nodes in Table 1.

Patient Identifier*	Onset Date*	Problem Code*	Problem Modifier*	Certainty Factor	DUSOI Score	Date Resolved
555-55-5555	05/10/91	60	marginal- f/u 24-26 wks	N/A	N/A	02/17/91
555-55-5555	02/17/91	707	transverse abd incision	N/A	N/A	02/17/91
555-55-5555	02/03/93	707	transverse abd incision	N/A	N/A	02/03/93

** indicates keyed fields which combine to uniquely identify the record*

were linked together; and the "third order" query involved data contained in three linked tables. The content of the response to each of the three queries across all four databases was found to be identical.

A second evaluation involved the time to retrieve a single patient's record by an indexed field. Retrieval of a single record is sometimes required for research queries. For this evaluation, records for three different patients were retrieved by the patient's name from the same locations within each database. The patient records selected for retrieval were chosen because they were the first, middle and last records of each database. In the relational packages, secondary indexes were created on the field of interest. Measurement of the retrieval times included the time required to link several patient tables together in a form.

Storage efficiency was evaluated by comparing the physical size of the relational tables (including indexes) with the size of the current structure of the same data in TMR. To compare table sizes for all three relational packages simultaneously within the size constraints of the hard drive, three sample sections of TMR data of different sizes were imported into the relational packages. The smallest section contained information about referral physicians. The mid-sized section contained patient problems (structure shown in table 2). The largest section for the comparison contained demographic information.

Comparing the size of relational tables with only the raw TMR data and delimiters would be misleading because the overhead of the TMR data structure would not be included. Therefore, an estimation of the size of the TMR data prior to conversion was made using the above equation.

$$TMRDataSize = R + \frac{P[(\frac{16}{5}N) + \sum^{i}(5M_i)]}{8}$$

In the formula, R is the total bytes of data (including delimiters) of the nodes containing the raw data, P is the total number of patient records, N is the number of applicable nodes to the table, and M is the number of non-empty nodes for the *i*th patient record.

RESULTS

Query Times

The queries of the relational databases were completed in a fraction of the time required to query the same information using GEMISCH programs in TMR. All three queries using the legacy programs took over an hour to complete. In contrast, the three queries on the relational systems all took less than 4 minutes (Figure 1).

Among the relational packages, Paradox® for DOS had the best average query time of 76 seconds per query. Access® averaged 106 seconds per query, while Paradox® for Windows averaged 110 seconds per query. The average query time on the TMR system was 5691 seconds, or approximately 1.5 hours. Paradox® for DOS performed best on the first and second order queries (61 and 54 seconds respectively), while Access® performed best on the third order query (63 seconds). Paradox® for Windows recorded the slowest times on the second and third order queries (122 and 146 seconds respectively), while Access® was slowest on the first order query (196 seconds).

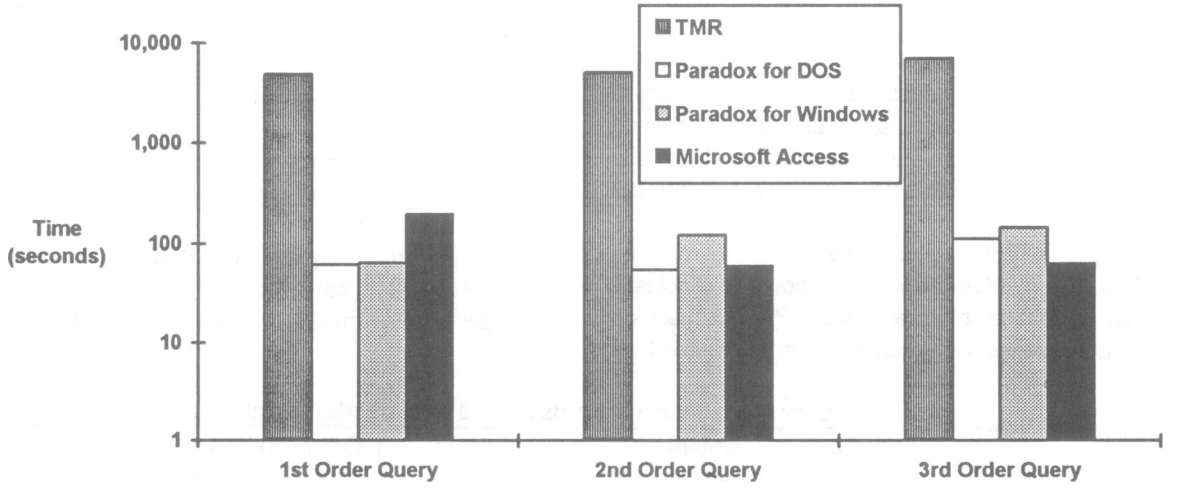

Figure 1. Logarithmic graph of query times by degree of complexity.

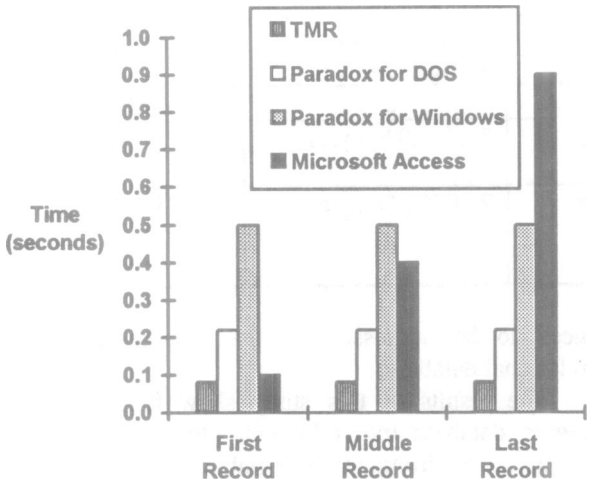

Figure 2. Patient record retrieval times by an indexed field (patient name).

Single Record Retrieval

The times for retrieving a single record by an indexed field varied among the relational packages (Figure 2). TMR access times were less than a tenth of a second for all three patients regardless of where their records were stored in the database. This finding was expected because the TMR application program code has been optimized to retrieve single records. Among the relational packages, Paradox® for DOS had the fastest search times and averaged 0.2 seconds per retrieval. Microsoft Access® was second fastest with an average of 0.4 seconds per retrieval, with the retrievals becoming progressively slower the deeper the patient record was located in the database. Paradox® for Windows was the slowest at finding a single patient record and averaged 0.5 seconds per retrieval.

Storage Efficiency

The physical size of the data files increased substantially when the patient data was converted from TMR into relational format. All three relational databases more than doubled the size of the original TMR data section (Table 3). For the smallest section, the Paradox® databases were the most efficient for storing the patient data. For larger sections, the Access® database was the most efficient. The smallest section of TMR data grew from 54 kilobytes in TMR to an average size of 162 kilobytes in relational format (301% the size of the original data). The mid-sized data section went from 8.8 megabytes to an average 29.6 megabytes (337% the size of the original data). The largest section showed the largest proportional increase following conversion: 18.2 megabytes increased to an average of 66.4 megabytes (365% the size of the original data).

The largest file imported in the entire project was the subjective and physical findings section of the TMR database; this section contained approximately three million entries. The file containing the SAP data was more than 74 megabytes. All three relational database packages had difficulty converting this large file. In Paradox® for Windows, the table became corrupted after 2 queries. Paradox® for DOS and Microsoft Access® did not complete the import process. In order to successfully convert the TMR SAP data, we divided SAP data into four smaller tables. Although this step caused the database to deviate from normal form, it provided the end-user with the tables required to run queries on all SAPs in the database. Though a partitioning of the SAP data was required for these three database packages, partitioning might not be necessary with other relational database packages.

DISCUSSION

The primary benefit of the conversion of the legacy database to relational format was the significant reduction in query times. While the relational databases improved query speed, they did so at the expense of decreased storage efficiency and increased retrieval times for a single patient record by an indexed field. The TMR data structure provides extremely efficient storage because space is not allocated for data that are not present in the record. In contrast, the relational database structures required that substantial space be allocated whether or not data was stored in the field. In spite of the increased size, the relational database tables were still manageable in a PC environment. The amount of time required to retrieve a patient record by an indexed field, however, could be problematic for accessing patient data in a clinical setting. Fortunately, single record retrieval is not frequently required in research queries, thus this drawback may not impact most of the database queries.

Entering queries on the relational systems was more expedient than programming queries directly in TMR. The relational database packages evaluated incorporated query-by-example with graphical user interfaces. This querying style was more interactive and more flexible than the query system in TMR. If a query-by-example did not return the desired results, the query could be modified and re-run in a matter of seconds. In TMR, a modification in a program would need to be made to alter the query.

The relational approach has several potential problems. First, the commercial relational database package may not accommodate vague temporal representations commonly used in a CPRS. If the exact date or time of an occurrence is not known, TMR allows

Table 3. Table sizes in bytes including overhead.

Database tables	TMR® Data Structure	Paradox® for DOS v 4.0	Paradox® for Windows v 1.0	Microsoft Access® v 2.0
Referring MDs	54,032 bytes	143,360 bytes	147,456 bytes	196,672 bytes
Referring MDs + Problems	8,797,693 bytes	42,121,216 bytes	33,751,040 bytes	13,074,496 bytes
Referring MDs + Problems + Demographics	18,220,912 bytes	88,733,696 bytes	73,056,256 bytes	37,584,960 bytes

the user to input only the known information. For example, if a patient knows he had an immunization in June 1995, but does not remember the specific day of the month, the date can be stored as 6/??/95. The relational database packages we tested only accepted complete dates and times in their designated field types. Consequently, time and date data exported from TMR had to be stored in alphanumeric fields and built-in date and time functions could not be used. Second, the relational format makes it difficult to store fields of variable length in the CPRS. The longest occurrence of the data can be measured and used for the field length; however, this may cause an unnecessary overhead in the database. For example, in TMR, a 200 character narrative explanation might be entered in the modifier field instead of a coded data phrase. In the relational database, the modifier field would be set at 200 characters. This overhead for free text data elements could be minimized by storing narrative modifiers in a separate table. The third potential problem relates to maintaining concurrence following the conversion of the TMR database. If data were changed in TMR, it would need to be changed in the relational version. Because the legacy database is the initial repository for the data, it must be considered the "gold standard," the official and most reliable version of the data. To avoid converting the entire CPRS database to maintain concurrence after a modification of the data, a daily log file of modified records could be used to identify the records requiring re-conversion in order to capture revised or new data. This approach presents a problem when database activities are not recorded in the daily log, such as when a programmer fixes a corrupt file or when system modifications are made outside of normal application code. A mechanism would need to be put in place to record these changes in the legacy system as well. Fourth, relational conversion introduces security issues. Multiple copies of the legacy system data present a potential for improper access and use of confidential patient data. In a CPRS, security precautions are already in place because the data exist in one central repository with password protection. In TMR specifically, patient data are naturally encrypted by the data structure, providing an extra layer of security. Confidentiality will need to be addressed by the administrators of the relational databases.

The results of this study show that converting a legacy database into relational format in the personal computer environment is possible. Such a conversion enhances query efficiency and improves accessibility to data by eliminating the barriers imposed by the legacy system. The storage of long-term clinical data in relational format permits health care professionals to use commercially available software to perform retrospective studies. Studying this clinical data will potentially lead to improvements in the quality and cost effectiveness of health care delivery.

This work supported in part by NLM Training Grant #LM07071-3.

REFERENCES

1. Fisher LD, Gillespie MJ, Jones M, McBride R. Design of Clinical Database Management Systems and Associated Software to Facilitate Medical Statistical Research. *Cril Rev in Med Informatics,* 1988;1:323-31.

2. Fries JF. The Chronic Disease Data Bank Model: a Conceptual Framework for the Computer-based Medical Record. *Comp and Biomel Res,* 1992;25:586-601.

3. Hlatky MA, Lee KL, Harrell FE Jr, Califf RM, Pryor DB, Mark DB, Rosati RA. Tying Clinical Research to Patient Care by Use of an Observational Database. *Stat Med,* 1984;3:375-87.

4. Pryor DB, Callif RM, Harrel FE, Hlatky MA, Lee KL, Mark DB, Rosati RA. Clinical Data Bases: Accomplishments and Unrealized Potential. *Medical Care,* 1985;23:623-47.

5. Safran C. Using Routinely Collected Data for Clinical Research. *Stat Med,* 1991;10:559-64.

6. Tierney WM, McDonald CJ. Practice Databases and Their Uses in Clinical Research. *Stat Med,* 1991;10:541-57.

7. Huff SM, Berthelsen CL, Pryor TA, Dudley AS. Evaluation of an SQL model of the HELP patient database. *Proc Symp Comp Appl Med Care,* 1992;15:386-90.

8. Hammond WE, Stead WW. The Evolution of GEMISCH and TMR. In: Orthner HF, Blum BI, editors. Implementing Health Care Information Systems. Springer-Verlag, New York, 1989;33-66.

9. Burkett, ME. The Tertiary Center and Health Departments in Cooperation: The Duke University Experience. *J Perinat Neonat Nursing,* 1989;2:11-19.

10. Hales, JW, Hammond, WE, Straube, MJ. Reverse Engineering Objects into the TMR Record Structure. *AMIA 1994 Spring Congress Program,* 1994:82.