Conceptual Data Model for a Central Patient Database

Stephen Johnson, Ph.D., Carol Friedman, Ph.D., James J. Cimino, M.D.,
Tony Clark, George Hripcsak, M.D., Paul D. Clayton, Ph.D.

Center for Medical Informatics, Columbia Presbyterian Medical Center

This paper presents methods used to develop a conceptual model for a patient database forming the centerpiece of a clinical information system under development. Various modeling techniques are discussed using a simplified fragment of the model. A method for mapping the model onto a relational design optimized for single patient retrievals is described. The results section discusses a number of issues pertaining to the flexibility and usability of this architecture.

Introduction

A conceptual data model is the integrated view of all data in an enterprise, and bridges the gap between the data organization as viewed by the database management system (the physical data model), and by individual user applications (logical data models). One of the primary motivations behind this three-level architecture is shielding applications from changes to the physical design ("physical data independence"), and changes to the conceptual design ("logical data independence") [1, 2, 3]. Another important aspect of this architecture is that the conceptual and logical levels can be employed effectively for communication with domain experts when validating the design, without necessitating that the experts possess detailed technical knowledge about databases. Feedback from domain experts is essential when developing databases in fields such as medicine in which data can be complex and the different types of data elements can be very large.

With relational databases, the relational schema can play the role of the conceptual layer, but in many respects relational tables are too close to the physical design. Tables are a suitable representation for application programmers, but obscure much of the semantics of the data model, and do not possess important modeling constructs, e.g. classification hierarchies. A higher level of abstraction for data modeling is available using methodologies such as entity-relation diagramming [4], fact-oriented modeling [5], knowledge bases of logical rules [6], and object-oriented analysis [7].

The application of any one of these techniques has the advantage that the same model can be used with different database management systems. The mapping of a conceptual design to the schema of a particular database system can be greatly facilitated by the use of database design software. For example, there are a number of commercially available computer-aided software engineering (CASE) tools that can generate a relational database design from an entity-relation model, and can even "reverse engineer" existing database designs for inclusion within the general entity-relation model.

Proposals for a standard data model of clinical information are just beginning to emerge, e.g., the model proposed for use in the HL7 and MEDIX standards for clinical information exchange [8]. A standard data model should contain information about institutional entities (e.g., organizations, health care professionals, and locations) and the events that take place in the clinical setting (e.g., requests for services for patients, administration of therapies, and performance of diagnostic tests). At the same time, standards for the items of clinical data that can be stored in clinical databases (e.g., diseases, medications, tests) are evolving [9, 10, 11, 12, 13]. Such general models can provide a framework for models being developed at different institutions, reducing the duplication of effort in database development, promoting the use of common software tools, and facilitating the sharing of patient records.

The Central Patient Database at the Columbia-Presbyterian Medical Center (CPMC) is the central component of the Clinical Information System (CIS) under development [14]. The database is implemented with relational technology on a mainframe, and is designed to provide good performance for patient-oriented queries for both results review applications and for a decision support system that monitors the storage of new patient data [15]. Since query performance was the primary concern from the outset, a physical design was developed first, to ascertain the feasibility of relational technology.

Methods

Entity-relation (ER) modeling was chosen as the methodology for creating the conceptual model for the Central Patient Database. The technique provides an adequate set of semantic primitives, works well with relational technology, and can be partially automated through the use of CASE tools (The Bachman Data Analyst tool [16] was used to build the model described here). The first phase of development involved the integration of a variety of existing databases and other information sources into a single ER model. In the second phase, a number of generalizations and simplifications were made using hierarchical

This work was supported by the International Business Machines Corporation and by a grant from the National Library of Medicine LM04419 (IAIMS).

classification techniques. The final phase specified how the conceptual design is mapped to the existing physical database design.

Rather than present the details of an entire conceptual model, we will examine a greatly simplified fragment (Figure 1) that exhibits a number of techniques that were found to be useful in organizing clinical data. In the diagram, rectangles represent entities, single ended arrows one-to-many relations, and double ended arrows many-to-many relations. There is a single one-to-one relation (no arrows) between the "order" entity and the "performed service" entity. Subclass entities (e.g., "patient" and "professional") appear inside the box representing the superclass entity (e.g., "person"). The diagram does not show names of relations or attributes of entities.

Populating the Model

The conceptual model was first populated by constructing an ER model of the The Medical Entities Dictionary (MED), a frame-based knowledge system for managing all items of clinical data ("medical entities") that can be stored in the Central Patient Database [17]. The current version of the MED contains primarily laboratory information, and has 2248 frames with 45 different slots. The ER model that was built from the MED consisted of 15 entities (e.g., disease, procedure, specimen, result, chemical) and five relationships. The algorithm used to transform the MED was carried out manually, but could easily be automated. Only two of these entities appear in Figure 1: "diagnostic procedure" represents both the procedures that can be performed on a patient, and the tests that analyze a specimen drawn from a patient; "result-type" represents the types of results that diagnostic tests may have. The entities are joined by a one-to-many relationship, indicating that a particular test may generate multiple types of results.

The next step was to reverse engineer the existing physical model of the Central Patient Database, implemented as a relational database in DB2. The version of the relational database that was used for this step contained only a few tables for demographics, orders, performed procedures (both diagnostic and therapeutic), and results of (diagnostic) procedures. This database required only a few tables because each table had a "generic" design, and could be used to store a wide variety of clinical data [14]. The two most important entities that were captured were a generic "performed service" entity, and a generic "performed service component" entity that represents detail information about the procedure such as test results, comments about the procedure, and the specimen analyzed by the procedure. These entities have a one-to-many relationship (inherited from "clinical event"), meaning that a performed procedure may have zero or more associated components.

In the integrated data model, the entities captured from the MED are used as "reference entities" or "code entities" for the entities captured from the relational patient database. Figure 1 shows how "diagnostic procedure and "result type" are used to control coded data entered for orders and performed service events: each distinct procedure and test is an instance of the "diagnostic procedure" entity. Each instance has a "MED-code" attribute with a unique integer value. An instance of "performed service" simply refers to the code of the particular procedure performed, and an instance of "performed service component" refers to the code of the particular test performed, and the code for the result type of the test. This arrangement permits the addition of new types of procedures and tests to the database without requiring a change to the database schema. This model is similar to one developed for objective clinical results at

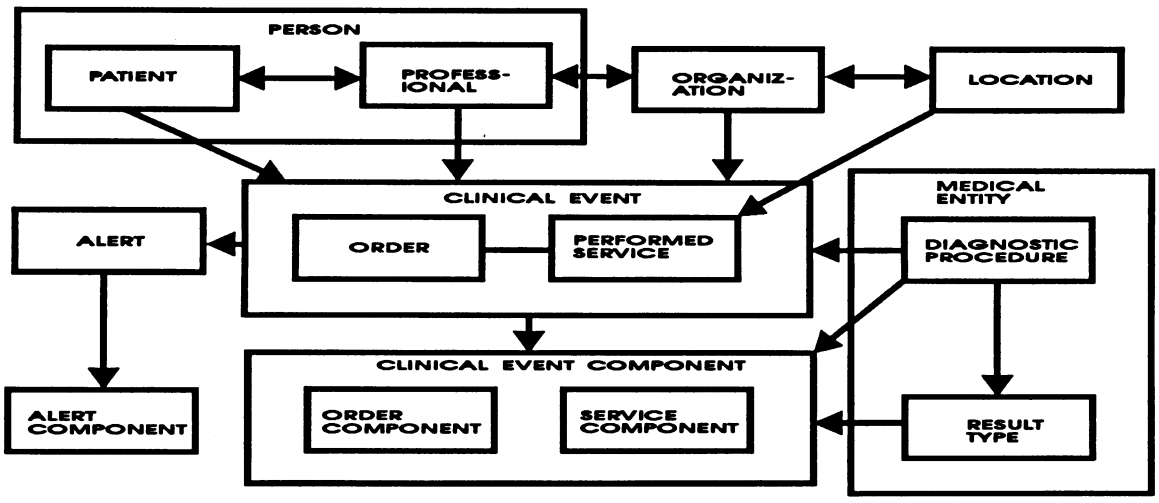

Figure 1: Fragment of Conceptual Model

Children's Hospital [18].

Additional clinical events were added to the ER model by incorporating entities from a number of sources: the proposed MEDIX data model [8], a hierarchical database for orders and results [19], and an Out-patient Database being developed at CPMC [20]. Clinical events from the proposed MEDIX data model include: encounter (e.g., an in-patient or out-patient visit), episode (which may span several encounters), patient problem, treatment, service request (an order), scheduled service, actual service, specimen collection, specimen analysis, result distribution. These were combined with similar entities from the CPMC relational database for out-patient information, adding substantial content, since the MEDIX model used did not supply attributes for entities.

Next, some of the "institutional" entities were incorporated from the MEDIX model [8] which included: location (offices, rooms, beds), person (e.g., patient, health care professional), and organization (e.g., family, company, society, provider organizations). Entities for departments (e.g., laboratory, pharmacy) were not present in the MEDIX model, and were added. The present CPMC model does not yet include a number of entities present in the proposed MEDIX model: institutional resources and schedules; financial information about payers and accounts; and entities comprising the actual paper chart (documents, folders, charts). Detail for institutional entities was provided by incorporating attributes of entities from an ADT database implemented in IMS [21]: patient, case, doctor, location, and discharge.

The MEDIX model suggests a few entities for describing the information system itself, but this portion of the model is not complete. In the CPMC system a separate relational database, the Metadatabase, is used for this purpose, storing connections between the components of the Clinical Information System [22]. Entities for alerts generated by the decision support system that monitors the Central Patient Database also had to be added to the ER model, and related to other clinical entities such as laboratory results. For example, if a laboratory result instance were stored in the database, and an alert were generated (due, e.g., to an outlying test result value), an alert instance would be created that referred to the result instance.

Constructing Type Hierarchies

In the second phase of developing the ER model, entity type hierarchies were constructed to capture generalities in the design, providing a clearer, more easily understood model. Type hierarchies provide inheritance for attributes of entities, reducing effort and the potential for introducing inconsistent attributes into the model. The MEDIX model makes some use of hierarchies. For example, the "person" entity has subclasses for patients, and for health care professionals (see Figure 1), which in turn has subclasses for nurses and physicians. Patients and health care professionals both have attributes for "first name" and "last name"; these attributes can be inherited from the common parent entity "person".

Relationships between entities can also be inherited, which is especially useful for defining events. A clinical event in the model is defined as the participation of a patient, an organization, a health care professional, and zero or more alerts associated with the event. All other events inherit these relationships. (Figure 1 shows only two subclasses of event "performed service" and "order", but the model contains several others, e.g. "patient problem", and "visit".) Without inheritance, every event would have to have an explicit relationship with patient, organization, etc., which would add redundancy and make the model more difficult to comprehend.

Frequently, several subclasses of an entity have attributes which are similar in meaning but have different data types. For example, it was observed that each subclass of "clinical event" had an attribute that was used to link the event in the Central Patient Database with another database system. The "order" entity had an "order-number" attribute for compatibility with an order entry system, and "laboratory service event" had an "accession number" attribute to link with a laboratory system. The model can be simplified by positing a single attribute for the superclass entity that is inherited by the subclasses. In order to handle the different data types, the attribute must be made "generic", i.e. have a character string data type of sufficient length to accommodate the sizes of the various subclass attributes.

Many of the institutional entities (e.g., location) are well-defined and unlikely to change much over time. The model uses traditional, normalized entities to define these. Some of the clinical entities are more difficult to structure, and are likely to evolve rapidly as new ancillary systems begin sending data to the central database. For these, a generic "header/component" structure is employed, in which a "header" entity has a one-to-many relationship with a "component" entity. The "header" entity participates in the various relationships that join the structure to the other entities in the model. Each component entity has a generic "value" attribute that is used to store a variety of data types, and a "value type" attribute used to specify what type of data a given component instance holds.

For example, "clinical event" is a "header" entity, and is related to "patient", "professional", "organization" , and "alert". The "clinical event" entity has a one-to-many relationship with the "component" entity "clinical event component", which can be used to store a wide variety of

clinical data, such as test results (numeric), or comments (text).

Mapping to Relational Design

The final phase was mapping the conceptual model (entity-relation) to the existing physical model (a relational database). This process was greatly facilitated by use of the Bachman DB2/DBA tool [16], which forward engineers a relational design from the ER model. Further physical design information can then be added to the relational design, e.g., the addition of indexes and specification of the physical clustering of rows within tables, to improve query performance.

The default physical mapping of an entity in the ER model is a table of the same name in the relational model. All the attributes of the entity become columns of the corespondent table. Relations are also realized as columns. A one-to-many relation becomes a column (foreign key) in the table on the "many" side of the relationship, that refers to the primary key of the table on the "one" side of the relationship.

Clearly, we do not want every entity in the conceptual model to become a table in the relational model. The relation model should only have a small number of tables (e.g., less than 50 for DB2) to reduce the number of join operations performed when retrievals are made. Therefore, for each type hierarchy in the ER model (e.g., the hierarchy of the various types of organizations, or the different kinds of people), some level must be chosen as the appropriate level of detail for that kind of entity. All the entities at this level become tables in the relational design. No entity above or below this level in the hierarchy will have a physical representation in the relation database.

For example, consider the hierarchy of clinical events in the model. If the top entity of this hierarchy is selected as the level of granularity for events, then the database will contain a single table for the storage of all the kinds of events. If the level below this is chosen, then orders, visits, problems, services, etc. will be stored in separate tables, but pharmacy orders, e.g., will be in the same table as radiology orders. At the next level, the decision might be made to differentiate various the kinds of services performed, in which case radiology exams, laboratory results, administration of medications, etc. would each be stored in a different table.

It is not necessary to generate tables for entities above the chosen detail level, since their attributes are inherited by all entities below them in the hierarchy. However, if an entity below the chosen level has an attribute (or relation) not present in the entities of the chosen level, an appropriate column must be added to one of the tables generated for the hierarchy.

Results

The introduction of the conceptual design did not interfere with the operations of the Central Patient Database. Reasonable performance for single patient queries has been achieved by keeping the number of tables small, and by the use of physical clustering on event tables using the patient identifier together with the primary date and time of events. The type hierarchy of the ER model helps to reduce mistakes in maintaining the tables. For example, the data type of the "event-id" attribute of the "clinical event" entity could be changed. After the altered ER model is forward engineered by the CASE tool, all the tables that inherit this attribute would be re-generated with columns with the correct data type. Without the inheritance mechanism of the CASE tool, each of these tables would have to be changed by hand.

Another nice feature is that it is fairly easy to make changes concerning the desired level of detail for hierarchies. For example, if initially different kinds of services were not distinguished and kept in a single table, and later it is decided to keep separate tables for services performed by different departments, the mapping can be changed, and the new tables generated. Rows from the old "performed services" table can then be copied into the appropriate departmental tables, and the "performed services" table can be dropped.

A significant problem remains with the use of SQL as the query language for the Central Patient Database. While a considerable degree of physical data independence is achieved, there are some difficulties in providing application developers an appropriate logical view of the data. This is a consequence of storing more than one type of entity in the same table. The application developer is forced know how to extract the type of entity his application needs from the table. This can be alleviated to some extent by creating appropriate "views" on the table using SQL. Since views are implemented by performing a query "behind the scenes", this method can reduce the performance of these retrievals to some extent. Potentially, a large number of views may be needed.

A more difficult problem exists in providing the decision support system (DSS) access to the database. The view required by the DSS is different from that needed by most applications, because the DSS knows only about medical entities as defined in the Medical entities Dictionary, and does not know about tables, columns, and SQL queries.

A possible solution to both of these problems is a layer of "data access modules" (DAMs), that lie between the database and application programs. Each DAM maps a generic table into a specific representation that is more appropriate to the needs of an application or the DSS. Application developers need only learn the interface to

the DAM, and do not have to be concerned with what the DAM does. The SQL in a DAM is maintained by database experts, and can therefore be written to optimize retrieval. But because a DAM is a program, it is less flexible for modification, should the data requirements of the application change. The issues of this architecture are the subject of future research.

Conclusion

Several techniques have been presented as methods for organizing clinical data in a conceptual model: "reference entities" represent coded data items such as procedures and drugs; entity type hierarchies provide inheritance of attributes; generic attributes combine similar attributes into a single attribute; header/component structures provide flexibility and extensibility. Simple mapping rules for type hierarchies were shown that allow the conceptual model to be mapped to relational database having a very different design.

This architecture allows alternative designs to be explored in the conceptual and physical models in a relatively independent manner. The use of a CASE tool to maintain the two models makes this architecture highly desirable. In addition, CASE tools facilitate incorporating other data models and database schemas into the central conceptual model. The model will continue to evolve as more ancillary systems are connected to the Central Patient Database, and as standards in modeling clinical data become established in health care computing.

References

[1] Date CJ: A Introduction to Database Systems, Vol. I. Addison-Wesley, Reading, Mass. 1986.

[2] Fleming CC, von Halle B. Handbook of Relational Database Design. Addison-Wesely, Reading, Mass. 1989.

[3] Ullman JD. Principles of Database and Knowledge Base Systems, Volume 1. Computer Science Press, Rockville, Maryland. 1988.

[4] Barker R: CASE*Method - Entity Relationship Modeling. Addison-Wesley, Reading, Mass. 1990.

[5] Nijssen GM, Halpin TA: Conceptual Schema and Relational Database Design a fact-oriented approach. Prentice Hall, New York. 1989.

[6] Walker A, McCord M, Sowa JF, Wilson WG: Knowledge Systems and Prolog. Addison-Wesley, Reading, Mass. 1990.

[7] Coad P, Yourdon E. Object-Oriented Analysis. Prentice Hall, New York. 1990.

[8] Spitzer, P: Proposal for a Generic Health Care Provider Data Model. Circulated in MEDIX (IEEE P1157) and HL7 working groups. Sept. 10, 1990.

[9] Humphreys BL, Lindberg DA: Building the Unified Medical Language System. Proc. of the 13th Annual SCAMC, Wash., D.C. 1989: 475-480.

[10] McCray AT, Hole WT: The Scope and Structure of the First Version of the UMLS Semantic Net. Proc. of the 14th Annual SCAMC, Wash., D.C. 1989: 126-130.

[11] Cote RA (ed): Systematized Nomenclature of Medicine, Second Edition. American College of Pathologists. Skokie, Illinois. 1982.

[12] United States National Center for Health Statistics: International Classification of Disease, Ninth Revision, With Clinical Manifestations. Wash., D.C.; 1980.

[13] Clauser SB, Fanta CM, Finkel AJ (eds): Current Procedural Terminology, Fourth Edition. American Medical Association, Chicago; 1984.

[14] Friedman C, Hripcsak G, Johnson SB, Cimino JJ, Clayton PD: A Generalized Relational Scheme for an Integrated Clinical Patient Database. Proc. of the 14th Annual SCAMC, Wash., D.C. 1990.

[15] Hripcsak G, Clayton PD, Pryor TA, Haug P, Wigertz OB, Van der lei J: The Arden Syntax for Defining Medical Decision Logic. Proc. of the 14th Annual SCAMC, Wash., D.C. 1990.

[16] Bachman Information Systems: Data Analyst and Database Administrator (DB2) Reference Manual. 1990.

[17] Cimino JJ, Hripcsak G, Johnson SB, Clayton PD: Designing an Introspective, Multi-Purpose Controlled Medical Vocabulary. Proc. of the 13th Annual SCAMC, Wash., D.C. 1989: 513-518.

[18] Stahlhut RW, Mcallie DP, Waterman DM, Margulies DM: A Relational Model for Clinical Objective Results. Proc. of the 14th Annual SCAMC, Wash., D.C. 1990: 354-358.

[19] International Business Machines: Patient Care System ORDERS Database Reference Manual. 1988.

[20] Shea S, Clark AS, Clayton PD: Columbia-Presbyterian Medical Center IAIMS Outpatient Clinical Information System Implemented in a Faculty General Medicine Practice. Proc. of the 14th Annual SCAMC, Wash., D.C. 1990: 730-734.

[21] International Business Machines: Patient Care System Patient Management Database Reference Manual. 1988.

[22] Johnson SB, Cimino JJ, Friedman C, Hripcsak G, Clayton PD: Using Metadata to Integrate Medical Knowledge in a Clinical Information System. Proc. of the 14th Annual SCAMC, Wash., D.C.; 1990.