A DATA MODELING PROCESS FOR DECOMPOSING HEALTHCARE PATIENT DATA SETS

By

Der-Fa Lu, PhD RN[1]

W. Nick Street, PhD[2]

Faiz Currim, PhD[2]

Ray Hylock, BS[2]

Connie Delaney, PhD RN FAAN FACMI[3]

[1]The University of Iowa College of Nursing

[2]The University of Iowa Management Sciences Department

[3]University of Minnesota School of Nursing

Citation:

Lu, D. Street, W., Currim, F., Hylock, R. & Delaney, C. (February, 2009). A data modeling process for decomposing healthcare patient data sets *Online Journal of Nursing Informatics (OJNI), 13*, (1). Available at http:ojni.org/13_1/Lu.pdf

INTRODUCTION

Healthcare information technology can help healthcare sectors control costs and improve patient safety.[1] In his 2004 State of the Union Address President George Bush said, "By computerizing health records, we can avoid dangerous medical mistakes, reduce costs, and improve care." Subsequently, he established the position of National Coordinator of Health Information Technology to facilitate the goal of building interoperable electronic health records for most Americans within ten years. The implementation of these electronic health information systems will generate an increasing amount of patient care data. The issue of improving healthcare by effectively utilizing clinical patient data via database management therefore becomes paramount to this goal.

Clinical data sets capture data from the point of care throughout the patient care process. This process of data collection is very different from data created from a highly standardized and controlled research protocol. Furthermore, researchers or clinicians using clinical data to answer important questions for effectiveness research and quality improvement do not have control over data collection, retrieval, and storage issues. The users are often confronted with difficult tasks in assessing system capabilities and defining retrieval and storage procedures.[2] For example, clinicians often have to rely on a busy computer programmer to develop a customized program to retrieve relevant data elements from the hospital central server, often delaying data retrieval. Once the clinician has all the data elements, he or she must reconstruct the data into a logical structure before feeding them into a data analysis program to produce useful information.

One solution to the problem of data accessibility is the design and construction of patient care databases using the relational database model. End users can use common database query languages to manage, manipulate, and analyze data. Using knowledge discovery from data sets and data mining procedures, the relational databases can uncover useful practice patterns, thus improving quality of care. Data is transformed to useful information and knowledge in an efficient manner. To "build once

—use many times" maximizes the use of data components already in existence, since the design of the database is publicly known.[3]

Most of the existing healthcare literature on relational database topics is focused on research databases rather than on clinical databases.[4-6] For example, Boyington[4] and others designed a relational database for a research project to manage data for a six-year clinical research project establishing a nursing model for older rural women with urinary continence difficulties. McCormick and McQueen[5] reported their database design for clinical geriatric research. In Ward's[6] article, he reports using a relational database design to organize data from an intensive care unit clinical information system. Building enterprise relational databases from existing clinical data sets needs more research.

The purpose of this paper is to present an example of a data modeling process for large healthcare data sets. We use entity-relationship (ER) modeling to analyze the database requirements and normalization principles to decompose and validate effective data set design for the relational schema. This model, when applied to other similar large data sets, provides a framework for creating a clinical data warehouse. Clinicians can use this data model to manage the large volume of clinical data for decision support and quality control at the point of patient care.

BACKGROUND

Minimum Data Sets

One way of managing large amount of data in healthcare setting is by minimum data sets. Minimum data sets in healthcare are designed to identify essential, common, and core data elements to be collected for all patients receiving care. Moreover, it is a standardized approach that facilitates the abstraction of these core data elements to describe the patient care practice.[7] The Health Information Policy Council [8] Subcommittee on Data Comparability and Standards identified nine criteria for inclusion in a minimum data set: orientation to multiple users, collectible and reliable data, non-duplication of data from other sources, confidentiality, cost of data collection, clear definition, guidelines for procedural definitions, uniformity of definitions across minimum data sets, and generic definitions. The development of the Uniform Minimum Health Data Sets (UMHDS) and the Nursing Minimum Data Set

(NMDS) are two examples. They both have a set of standardized variables in the data set for capturing important healthcare information. The variables selected in this study use the UMHDS and NMDS as their models. The advantages of large data sets for outcomes research include: ready access to large numbers of subjects, provision of information on episodes of patient care as well as data on settings, provision of longitudinal records of episodes of care, access to samples more representative of the universal population than studies relying on volunteer subjects, machine-readable forms that are easily transferable between sites, and relatively inexpensive retrieval compared to primary data collection. Limitations of these data sets include data validity, degree of completeness, and potential for deficient clinical information.[9, 10]

The UMHDS is a set of minimum specifications for the content of information systems. It is a concept that can be applied to healthcare or to nursing. The UMHDS "defines the central core of data needed on a routine basis by the majority of decision-makers about a given facet or dimension of the health care delivery system and it establishes standard measurements, definitions, and classifications for this core." [11p263] Patient demographic and services data are two such examples.

The Nursing Minimum Data Set, built on the concept of the Uniform Minimum Health Data Sets, includes the essential nursing data used on a regular basis by the majority of nurses across all settings in the delivery of care. The purposes of the NMDS are as follows: 1) Establish comparability of nursing data across populations, settings, geographic areas, and time; 2) Describe the nursing care of patients or clients and their families in a variety of settings, both institutional and non-institutional; 3) Demonstrate or project trends regarding nursing care needs and allocation of nursing resources to patients or clients according to their health problems or nursing diagnoses; and 4) Stimulate nursing research through links to the detailed data existing in nursing information systems and other healthcare information systems.[12p107]

The NMDS is comprised of sixteen elements: 1) nursing diagnosis, 2) nursing intervention, 3) nursing outcome, 4) intensity of nursing care (1–4 capture nursing care data), 5) personal identification, 6) date of birth, 7) sex, 8) race and ethnicity, 9) residence (5–9 are patient or client demographic data), 10) unique facility/service agency number, 11) unique patient health record number, 12) unique registered nurse provider number, 13) admission or encounter date, 14) discharge or termination date, 15) disposition of patient or client, 16) expected payer (10–16 are service data).

The Genesis Data Set

The Genesis Medical Center (formerly known as Mercy Hospital before consolidating with St. Luke's Hospital in Davenport, Iowa, in 1995) generated the research data sets used in this paper. This center is a 500-plus bed community hospital with sites located on two campuses in Davenport, Iowa. Approximately 650 registered nurses provide acute and skilled care across the inpatient, outpatient, and home care delivery areas. Genesis Medical Center (GMC) has had a computerized nursing information system involving all aspects of patient care planning since 1983. Staff nurses receive standardized nursing language training during orientation and routine training to ensure the reliability of data from different raters. Regular chart audits ensure data validity. Care planning includes all nursing diagnoses approved by the North American Nursing Diagnosis Association — a list of defining characteristics, etiologies, expected patient outcomes, and nursing orders sets containing common nursing interventions for each diagnosis. Clinical nurses train to use these care plan systems. A nursing data repository was developed in 1991 as part of research collaboration between the University of Iowa NMDS Research Team and Genesis Medical Center. Since then, the Genesis system has incorporated other nursing classifications (NANDA, Nursing Interventions Classification [NIC], and Nursing Outcomes Classification [NOC]).[13-15]

The Genesis data set comprises patient demographic and services data drawn from the Uniform Hospital Discharge Data Set (UHDDS). The Genesis Medical Center built the UHDDS for routine billing purposes. The Genesis data set included all the outpatient and inpatient information, with the exception of those patients having diagnoses related to AIDS, chemical dependency, and psychiatric disorders (per researchers' choice to protect those patients' confidentiality). The privacy of patients was protected by changing the unique patient identifier using an algorithm known only to the researchers at the clinical site. The structure of the GMC data set included all of the elements from NMDS except the unique number of the principal registered nurse provider.

The initial data were transported in text files (minus a data structure) at the beginning of each year, so that the data set contained all discharge data from the entire previous year. The researcher collected data from the Genesis system from 1991 to 2000. This clinical data repository contains more than one million patient records. Two large data sets were set up for patient data (P Table) and care data (T Table). The P Table lists variables in Table 1; those for the T Table are in Table 2.

The text data files extracted by the hospital programmers were sufficient for hospital data transportation and data containment (being all in one flat file was helpful for the data manager in monitoring the completion of data transferals). However, when researchers or clinicians wanted to use the data to answer critical research or clinical questions in order to improve quality of care, they needed a transformation of the flat file to a relational database design since writing queries on the text files would be very time consuming and may even lead to incorrect results. In the current T Table (nursing care data) format, there is no meaningful relationship between fields 3 to 13. These fields are the placeholders for nursing care information (nursing diagnoses, interventions, and outcomes labels). It is not possible to construct a query to find important information regarding patient care data in the current T Table format. Therefore, the entire data set was redesigned using a conceptual database structure—the entity-relationship model. The normalization process decomposed the variables in both the T and P Tables to ensure data consistency.

Table 1. Variables in P Table (demographic data)

Name	Description
SEQ	Unique encounter number (patient billing number)
PT_ID	Patient medical record number (one per patient)
SSN_R	Social security number
GENDER	Female or male
RACE	Ethnic background
ZIPCODE	Zip code
LOS	Length of stay, number of days between admission and discharge dates
SERVICE	Patient service type, e.g., surgical, orthopedic
DISSTAT	Discharge status; e.g., home, nursing home
INSUR 1	Primary health insurance
INSUR 2	Secondary health insurance
INSUR 3	Secondary health insurance when different from previous one
ICD 1	Primary disease code
ICD 2	Secondary disease code
ICD 3	Secondary disease code (co-morbidity)
ICD 4	Secondary disease code (co-morbidity)
MARSTAT	Marital status
RELIG	Religious preference
PROCED 1	Procedure received for this visit
PROCED 2	Another procedure received for this visit
ADMDATE	Admission date
BRTHDATE	Birth date
DISDATE	Discharge date
AGE	Current age (calculated from birth date and admission data)

Table 2. Variables in T Table (nursing care data)

Name	Description
SEQ	Encounter number for the visit
Field 3 to Field 13	All symptoms, defining characteristics, expected outcomes (goals), and nursing orders for the nursing diagnoses

Entity-Relationship Database Design, Relational Design and Normalization Process

This section provides some background for the major steps undertaken in the design and implementation of a relational database. More details can be found in a variety of standard texts on the topic[16-18].

Organizations today consider databases a critical part of their information technology (IT) infrastructure, and database management systems provide the back-end for a variety of applications including those in government, manufacturing, and the large service sector (including healthcare). Databases supporting the operations of a large establishment are complex and often store millions of records. This size is determined not only by the number of data entities in the database, but also by the number and intricacies of the relationships between data[19p.108]. With the importance of databases in providing IT services, and the consequent justification for the creation of large databases to support organizational informational needs, comes the requirement for robust design. Designing a database is a difficult process, and involves several stakeholders and decision-makers at different levels in the organization. Database design starts upon the completion of requirements gathering and analysis, and is typically divided into *conceptual, logical* and *physical design* [20, 21]. In this paper, we focus on the conceptual (the ER diagram) and logical design (building tables) stages since these are critical for data integrity and the foundation for subsequent implementation and tuning.

The ER model, developed by Chen[22], and the subsequent variants based on it, have become a widely used tool for presenting the conceptual structure and semantics of data objects. These conceptual models have three main constructs: entities (the objects in the database, representing physical or abstract objects in the real world being modeled), relationships (the associations and roles among entities), and attributes (the properties of an entity, each of which has a value). The ER diagram developed as the end-result of the modeling process serves as a valuable means to understand and verify the user requirements, as well as to communicate the database design among analysts. After generating the conceptual schema, the database analyst proceeds to translate it into the logical schema. While this could theoretically involve any data model, in practice the relational model is most commonly used, and the one we adopt. In the relational schema, tables typically represent entity classes, and entity class attributes become columns in the tables. Relationships map to their own tables or as attributes within existing tables (depending on the relationship cardinality). Each entity class has a special attribute called an *identifier* (in relational terms, a *primary key*), which uniquely identifies it; for example, a patient entity class might have an ID number. Weak entity classes are those that do not fully contain their own identifiers and depend on one or more other entity classes for identification, and, in effect, existence from a data-modeling standpoint. Relationships among entity classes have an associated cardinality that indicates the actual number of related occurrences for each of the entities.

Integral to logical design is the normalization process. Normalization is a formal methodology for designing the attributes that should (or should not) belong in a particular relation (table), and was first proposed by E. Codd[23]. The purpose of normalization is to achieve progressively higher levels (or norms) of data integrity, and in doing so eliminate insert, update, and delete anomalies. This process prevents two common pitfalls in relational-database design: the repetition of information and the inability to represent certain information. Normalization is part of the database design process. There are two approaches to performing the database design process: top-down and bottom-up. The top-down approach focuses on initially identifying entities and then assigning attributes[24]. Researchers using the bottom-up approach identify relevant attributes from existing data and assign them to the relational table's entities. This project includes both processes in order to reengineer a large healthcare database.

After the relational database is built and populated, users can employ a front-end or directly write Structured Query Language (SQL) statements to insert, update, and retrieve data. The relational data model (using normalized tables) has proven highly effective for managing very large volumes of data. It also provides users with powerful query capabilities (through SQL) to address specific research questions. SQL is a high-level programming language that has become a standard for querying, creating, and modifying data. When querying, it primarily uses three relational-algebra operations:

- projection, which extracts certain columns or derived expressions from each tuple (using the SELECT clause);
- selection, which applies filtering criteria to restrict the output of specific rows or records from the relations in question (using the WHERE clause); and
- join, which connects tables across relationships (can be performed in the FROM or WHERE clause).

Additionally, the GROUP BY and HAVING operators in SQL allow for querying and reporting aggregate information. The results and discussion session presents examples of queries and the results from this project.

METHODS

This study used the principles of entity-relationship modeling, logical design and normalization to create a relational database from a large healthcare data repository. The procedures for redesigning the database are as follows:

A. Obtain a data dictionary defining and explaining the classes and properties of data and the relationships among data entities.

B. Study the relationships among variables.

C. Resolve any questions with the domain experts at both clinical and research facilities.

D. Design the ER diagram according to the clinical relationships between variables. This diagram serves as the blueprint for the normalization of the data set.

E. Decompose the original data sets into the various tables according to the ER diagram.

F. Verify the design with a database management expert.

G. Build and populate tables.

H. Test relational queries.

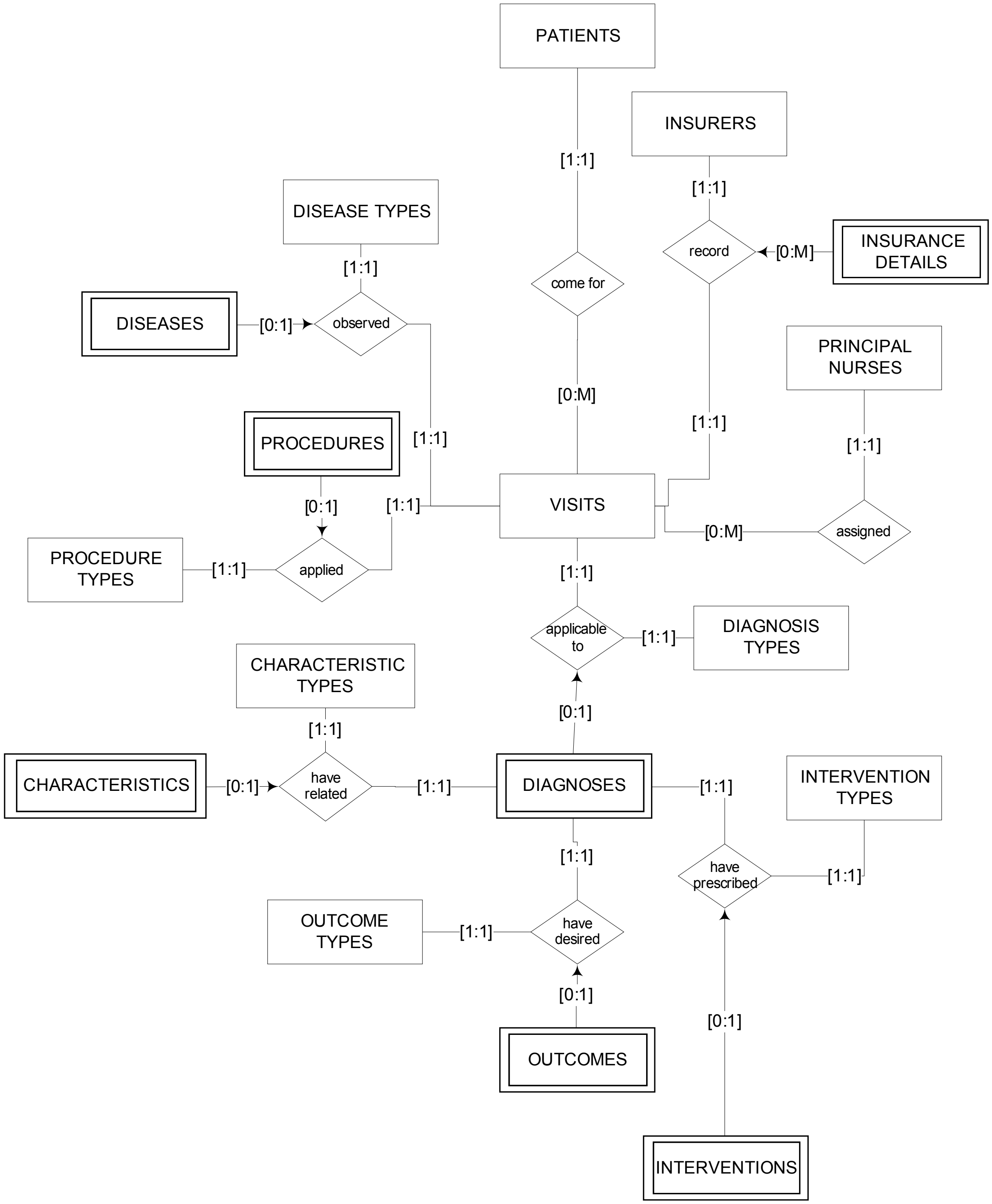
RESULTS

ER Diagram

When developing the conceptual schema (in ER notation), we focused on the idealized or model state of data. An advantage of this approach is it allows us to develop a conceptual diagram to apply to a class of such datasets, and have it be applicable independent of the specific implementation platform or data limitations and constraints (which may differ with each application). Our ER diagram (Figure 1) defines the entity and relationship classes relevant for our nursing care dataset. The notation we use describes entity classes with rectangles and relationships with diamonds. Rectangles representing strong entity classes have a single line, while a double line denotes weak entity classes. An arrow points from the weak entity class to its identifying relationship (i.e., the relationship connecting a weak entity class to the defining class(es) that provide part of its identifier). In Figure 1, we see that Diagnoses and Insurance Details are weak entity classes. The Diagnoses class depends on Visits and Diagnosis Types, and each diagnosis needs a single visit sequence number and nursing diagnosis type code to identify it. For convenience, we could add an artificial identifier to Diagnoses, but we chose not to do so in the current schema. Similarly, insurance information by itself is not unique, and depends on identifiers from Visits and Insurers.

Figure 1 does not show attributes to preserve readability (but the reader can infer these from the relational schema). Each patient has a set of demographic variables that usually do not change, and we listed them in the entity class Patients. A set of attributes describing a patient's specific hospital experience for each visit is collected; these are contained in the entity classes Visits, Insurance Details, Diseases, Procedures, and Nursing Diagnoses. Visits are central to our schema in that they link patient visits to the nurses assigned during the visit and the diagnoses delivered by them, as well as the observed medical diseases and procedures performed by the hospital.

Figure 1. ER Diagram for the database



The other central class in the schema is (nursing) Diagnoses. Although it is weak due to its identification through visits and diagnosis types, it allows us to link with a visit the specific characteristics, outcomes and interventions that form part of the nursing care plan for the patient. The cardinality constraints are listed along the line connecting the entity classes to the relationships. In our case, each visit is associated with exactly one patient (minimum and maximum of 1), for example. Conversely, there could be many visits per patient (minimum of 0, maximum of many—for which the symbol "M" is used). A weak entity class member is linked to exactly one entity from each of its identifying classes; hence, the cardinality on the identifying sides is always [1:1]. Using the example of the ternary relationship "applicable to" leading to the definition of diagnoses, the interpretation of weak cardinality is as follows. For a given visit (e.g., sequence number '123456') and diagnosis type (e.g., impaired mobility), there is either no such diagnosis combination in the database (minimum cardinality 0) or at most one diagnosis. This corresponds to the cardinality of [0:1] attached to Diagnoses in the applicable to relationship. For a given diagnosis (identified by a visit sequence number and diagnosis type) and diagnosis type, we can have a minimum and maximum of one matching visit. Likewise, for a given combination of a specific diagnosis and visit, we have exactly one diagnosis type. This also illustrates how cardinality in a ternary relationship differs from that in a binary relationship. If we were only modeling the relationship between Visits and Diagnosis Types (with no intervening weak class based off the relationship), then we would end up with a many-to-many relationship. This is because each visit has multiple (possible) diagnosis types associated with it, and each diagnosis type is linked to many visits. We can record the binary constraint in a ternary relationship by listing it in the data dictionary associated with the ER schema. For additional information about the kinds of cardinality constraints, their semantics for relationships, and their translated SQL equivalents in relational design, we refer the reader to prior work[25].

Admission to an inpatient unit and care provided by professional nursing personnel generated a nursing care plan for each patient. The T Table transcribes hospital information about nursing diagnoses, defining characteristics, expected outcomes (goals), and nursing activities. The order of nursing diagnoses (ND) entered in the data set (first, second, up to an eleventh ND) served as the link in the legacy system to the associated information for defining characteristics, outcomes, and nursing interventions. In our design, the link was preserved through the nursing diagnosis code, so it was no longer necessary to store the order.

Table Design

In the logical design phase, we translated the ER diagram into the corresponding relations. The Patients relation includes the attributes: PT_ID, SSN, BirthDate, Gender, Race, Zipcode, MarStat, and Relig. (Table 1 lists a detailed description of these attributes.) If one patient had multiple hospital visits, then he/she would have one unique encounter number for every visit represented in SEQ. The Visits entity class contains the information about each specific hospital experience. The attributes listed in the corresponding Visits table are SEQ, AdmDate, DisDate, LOS, DisStat, and Service.

The insurance information consists of three properties (INSUR1, INSUR2, INSUR3). The first is the patient's primary insurance policy, with INSUR2 and INSUR3 being the secondary insurance providers. The Insurance Details weak entity class captures this information. For simplicity, we leave out details about insurers themselves and the appropriate billing mechanisms (as these would tie into the billing systems of the hospital, and our focus is on nursing care). Upon translation, the insurance details table contains the visit and insurer identifiers (the primary keys of the corresponding strong entity classes), and an additional attribute Ordering indicating whether the specific insurer was the primary, secondary, or tertiary organization on that visit. Ideally, we would want to track insurance history, with start and end dates as related to a patient rather than a visit, but our data set does not contain this information.

A similar approach was taken for disease diagnoses. A reference table stores information on each ICD code (based on the International Statistical Classification of Diseases and Related Health Problems published by the World Health Organization) and associated disease classification data. The table structure for the disease details is: (SEQ, ICD, Ordering). SEQ ties to a visit, while ICD corresponds to a (medical) disease diagnosis. Ordering corresponds to a ranking (primary, secondary) of the ICD code for the patient on that visit. The source data provides medical diagnoses for each visit as ICD1, ICD2, ICD3, and ICD4, with ICD1 representing the primary diagnosis, and ICD2, ICD3, and ICD4 as the secondary ones. A visit that had four disease codes would previously have had all attributes ICD1 through ICD4 populated, but a visit that had one diagnosis would only have a value in ICD1 and have null values in ICD2, ICD3 and ICD4. In the new design, a single diagnosis would have one tuple (SEQ, ICD, Ordering), while multiple diagnoses would be represented by a row for each such diagnosis. This design also provides flexibility for the future in the event that we had to record more than four disease codes per visit for any reason. Similar to medical diagnosis data, we capture procedure information in its own table.

In summary, the following are the significant tables and their attributes (with the primary key for each table in bold):

1. Patients (**PT_ID**, SSN, BirthDate, Gender, Race, Zipcode, Marstat, Relig)
2. Visits (**SEQ**, PT_ID, Service, Admdate, Disdate, LOS, DisState, Age)
3. Insurance_Details (**SEQ**, **Insurer**, Ordering)
4. Diseases (**SEQ, ICD**, Ordering)
5. Procedures (**SEQ, Procedure**, Ordering)
6. Nursing_Diagnoses (**SEQ, NDiagnosis_code**, NDiagnosis_date, Ordering)
7. Characteristics (**SEQ, NDiagnosis_code, NCharacteristics_code**, NCharacteristics_date)
8. Outcomes (**SEQ, NDiagnosis_Code, NOutcome_code**, NOutcome_date)
9. Interventions (**SEQ, NDiagnosis_Code, NIntervention_code**, NInterventions_date)

Additional reference tables are not shown here for the sake of brevity, since the focus is to illustrate the management of the nursing data set. These tables include look-up data for characteristic codes, outcome codes, intervention codes, diseases, and procedures (and their descriptions). In order to present diseases in a clinically meaningful manner, the research team mapped the individual ICD codes to the Clinical Classifications Software (CCS) for ICD-9-CM.[26] A set of reference tables also stores the mapping between the Clinical Classification Software (CCS) (single-level diagnoses codes) and ICD-9-CM codes that correspond to each of them.

Queries

Once the tables were created and populated with data from the former P and T Tables, we evaluated their utility with sample queries. The following are examples of queries and their results. Details of the SQL code used are available in Table 3. Further discussion of the queries (including additional explanations and a discussion of vendor-specific syntax) is in Appendix 1.

1. What are the most frequently prescribed nursing diagnoses for patients aged 65 and above?

Nurses can use SQL to find out the five most frequently prescribed nursing diagnoses for the elderly in clinical care plans as documented by nurses (Table 3, query 1). These diagnoses are: Altered Health Maintenance (28), Knowledge Deficit (52), Risk for Infection (37), Pain (12), and High Risk for Injury (39). This information can be useful in designing patient care education, and nurses' continuing education content to address the needs of healthcare for elderly patients.

2.*What are the residential locations of elderly patients?*

Results of this query showed that most of the elderly patients lived in downtown senior citizen apartment/housing projects. There are two important reasons for this community hospital to know where the majority of their elderly patients live. First is to be able to design a seamless care plan, from acute hospital to community health organizations. Second, for marketing purposes, the hospital will know where to focus their recruiting efforts with their elderly clientele. Examples of SQL and the results are in Table 3, second query.

3.(A) *Which diseases are common for senior citizens (age 65+) but are not common for the age group 50-64?*

(B) *What are the common disease co-morbidities for these cases?*

In this complex query, our research team found important diseases for patients age 65 and older by using a clinically meaningful coding system (CCS). Two heart and lung related diseases were found (Table 3, Query 3). More specific co-occurring diseases associated with these two primary diseases were also queried. Cardiac dysrhythmias was highly associated with Coronary atherosclerosis and Essential hypertension. Chronic obstructive pulmonary disease was found with smoking cigarettes (as part of CCS 663), Coronary atherosclerosis, Essential hypertension, and Fluid and electrolyte disorders. Healthcare cost-effectiveness research and education program design specific for elderly population can be developed based on the results of this query.

These are only three examples of how SQL provides readily useful information. Once the relational database is built, users can run SQL queries of varying complexity to create a database subset that answers their research questions. Operational maintenance of the system to allow for thousands of updates a day is sustainable through a variety of commercial database management systems. This is evidence in support of the "build once, use many times" model. Retrieving clinical information and the knowledge embedded in the large data set improves quality of care. The discussion section presents more ways of querying this database.

Table 3. Examples of queries SQL codes and results

SQL code

Results

Query 1

```
SELECT NANDA, Freq FROM (
SELECT ndiagnosis_code as NANDA, count
(*) as Freq, rank() over (ORDER BY count
(*) DESC) as rank
FROM nursing_diagnosis N JOIN visits V ON
(N.seq = V.seq)
WHERE V.age>=65
GROUP BY ndiagnosis_code )
WHERE rank <=5
```

NANDA	Freq
28	5,914
52	5,252
37	3,738
12	3,140
68	1,729

Query 2

```
SELECT zipcode, freq FROM (
SELECT zipcode, count(*) as Freq,
rank() over (order by count(*) desc) as Rank
FROM patients P JOIN visits V ON (P.pt_id =
V.pt_id)
WHERE AGE > 65
GROUP BY zipcode)
WHERE Rank <= 5;
```

** We used " xx" to replace the middle 2 zip code digits for confidentiality.*

ZIPCODE*	Freq
5xx04	2,069
5xx22	1,941
5xx03	1,536
5xx06	1,317
5xx02	638

Query 3 (A)

```
SELECT C.ccs_code, count(*) as Freq
FROM ccs_icd C JOIN diseases D ON (C.icd =
D.icd) JOIN visits V ON (D.seq = V.seq)
WHERE V.age >= 65
AND C.ccs_code IN (
SELECT ccs_code
FROM (
SELECT ccs_code, rank() over (order by
count(*) desc) as R
FROM diseases D JOIN visits V ON (D.seq =
V.seq) JOIN ccs_icd C ON (C.icd = D.icd)
WHERE V.age>=65
GROUP BY ccs_code )
WHERE R <= 5 )
AND C.ccs_code NOT IN (
SELECT ccs_code
FROM (
SELECT ccs_code, rank() over (order by
count(*) desc) as R
FROM diseases D JOIN visits V ON (D.seq =
```

CCS	Freq
106	2,086
127	1,826

```
V.seq) JOIN ccs_icd C ON (C.icd = D.icd)
WHERE V.age>=50 and V.age<=64
GROUP BY ccs_code )
WHERE R <= 5 )
GROUP BY C.ccs_code;
```

Note: we used CCS codes in conjunction with ICD codes for this query.

Query 3 (B)

```
SELECT CCS, CoCCS, Freq
FROM (
SELECT C1.ccs_code as CCS, C2.ccs_code as
CoCCS, count(*) as Freq, rank() over
(partition by c1.ccs_code order by count(*)
desc) as rank
FROM ccs_icd C1 JOIN diseases D1 ON
(C1.icd=D1.icd) JOIN diseases D2 ON
(D1.seq = D2.seq) JOIN ccs_icd C2 ON
(C2.icd = D2.icd)
 WHERE D1.icd <> D2.icd
 AND C1.ccs_code IN ('106', '127')
 AND D1.ordering='1'
 AND D2.ordering IN ('2', '3', '4')
 GROUP BY C1.ccs_code, C2.ccs_code
)
WHERE rank <= 5;
```

CCS	CoCCS	Freq
106	101	168
106	98	135
106	106	109
106	108	81
106	96	76
127	663	199
127	101	190
127	98	182
127	108	112
127	55	95

Relevant CCS codes and names: 106 (Cardiac dysrhythmias), 127 (Chronic obstructive pulmonary disease and bronchiectasis), 101 (Coronary atherosclerosis and other heart disease), 98 (Essential hypertension), 108 (Congestive heart failure; nonhypertensive), 96 (Heart valve disorders), 663 (Screening and history of mental health and substance abuse codes), 55 (Fluid and electrolyte disorders).

DISCUSSION

Most published literature related to entity-relational database design in healthcare concentrates on research data sets[4-6]. This study presents a process and method for building an enterprise entity-relational database using a standard modeling method. We also demonstrate how, once the database is populated with clinical data, it can be used to answer research and management-related queries.

The use of the data in the database can be organized into seven categories[27]. They are: 1) aggregates, 2) utility, 3) abstractions, 4) correlations, 5) linear profiles, 6) comparisons, and 7) quality. Since this is a patient care database, the information generated from the data set has patient care as a focus. Each category presents specific examples.

1) Aggregates: The relational database provides frequency data of nursing diagnosis, signs and symptoms, assessment needs, procedures, outcomes, and activities.

2) Utility: Since standardized nursing language was used (NANDA for nursing diagnosis, and Nursing Interventions Classification for nursing actions) in the data sets, the consistency of nursing data is ensured, and the results can be used to compare care quality across different units and facilities.

3) Abstractions: Abstracted nursing data can be used in the areas of prevention, deterioration, and stabilization of patients.

4) Correlations: Correlations between nursing data and medical data, between nursing actions and doctors' orders, and between nursing actions and patient expected outcomes are readily executable. Complex queries can help identify the association between patients' living arrangements and their nursing diagnosis on admission. Associations between nursing diagnoses and patient profiles (age, gender, insurances, etc) can be tested. The research team is developing a methodology to validate clinically the linkage between nursing interventions and outcomes for cost-effectiveness research. The relational database from this project serves as the foundation.

5) Linear Profiles: Monitoring linear profiles and longitudinal data reflecting nursing data from admission to discharge is more encompassing. Likewise, expected

outcomes/patient responses to nursing actions and doctors' treatments throughout a given hospitalization are readily obtainable.

6) Comparisons: Comparisons between nursing actions and other allied health professionals' actions are more accurately drawn, as are patients' responses from specific treatments to expected outcomes.

7) Quality: As shown by the results of this ER model database, monitoring the reliability of both structured and variable content within the data sets is possible. Safety or incident reports related to specific nursing actions or doctors' orders increases quality control.

The results of this project (ER diagram, relational tables, and complete data dictionary) are essential in facilitating users who desire a basic understanding of the data set, and in performing various queries to answer questions. The ER diagram, in keeping with the goals for conceptual database modeling, captures the semantics for the data and models a design that is consistent with the ideal situation (if we had all relevant data, for example) and free from implementation considerations. Not all data is available in every clinical facility and each scenario varies. The tables we developed take into account the specifics of the data we had available, and implementation considerations such as the database platform.

Much patient care data is temporal in nature. While the theory behind and practical implications of managing temporal data have been well researched [28], it is only recently that commercial database management systems have begun providing better support for temporal data. With the continued advances, we see improvements in being able to provide temporal SQL support to end-users. Queries like: "Which patients have had symptom X and symptom Y overlap" or "List the coalesced disease history for patient 2341", are easy to write with temporal SQL (though not trivial with traditional SQL). This information will be important to manage patients with chronic complex disease, e.g., diabetics or heart related diseases. Another area for future work is to use XML (Extensible Markup Language) to provide effective interchange for temporal patient data, building upon some previous work that considers the use of temporal extensions to XML Schema [29]. This application would be important in managing complex chronic diseases for patients with chronic heart failure or diabetics.

The approach outlined here serves as the foundation for a comprehensive infrastructure that uses sound data storage and retrieval principles to aid clinicians and researchers in their quest to improve health care quality and contain costs. As a first step, our relational model can easily be expanded to include other healthcare standardized data sets such as the Nursing Management Minimum Data Set, creating a more complete version of the patients' health records. Data from multiple sites and multiple periods can then be collected into a *data warehouse*, a multi-dimensional database that facilitates more sophisticated analysis of data for knowledge discovery.[30] However, healthcare data that includes many multi-valued attributes (such as nursing diagnosis) presents challenges for the traditional "star" and "snowflake" data models used in standard warehouses. Our current research focuses on overcoming these challenges. Finally, a stable and complete warehouse would allow the use of data mining techniques to answer questions such as: What sets of nursing diagnoses, interventions, and outcomes appear together most (or surprisingly) often? How do these groups compare to the prescribed care plans in standardized references? What interventions lead to the most consistent achievement of desired outcomes, and does the answer depend on demographic or other data related to the patient?

Acknowledgment

The authors wish to acknowledge Patricia Ramstad and Barbara Lewis for their assistance in editing the manuscript.

REFERENCES

Authors' Bios

Der-Fa Lu, PhD RN, is an Assistant Professor in Nursing Informatics at College of Nursing, University of Iowa. Her research focuses on data mining in large patient care datasets with standardized nursing languages.

W. Nick Street, PhD, is an Associate Professor at Management Sciences Department, University of Iowa. His research focuses on developing data mining algorithms for both health and business datasets.

Faiz Currim, PhD. is an Assistant Professor at Management Sciences Department, University of Iowa. His research focuses on designing relational databases schema for complex datasets.

Ray Hylock, BS, is a doctoral student at Management sciences Department, University of Iowa. He is trained in building relational databases.

Connie Delaney, PhD RN FAAN FACMI, is the Dean at School of Nursing, University of Minnesota. Her research focuses on developing minimum nursing datasets.

APPENDIX 1: DISCUSSION OF SQL QUERIES

For convenience, we have written our queries using some functions provided by Oracle since it was the relational DBMS used for implementing and testing our design. In particular, when examining the queries that perform *Top-N* calculations, the reader may find that the syntax in Table 3 is different from what they may be used to seeing (even though it is SQL:2003 compliant). One option we could have taken was to write the top-N queries for products conforming to the ANSI SQL-92 standard. However, this ends up being longer (and typically not as efficient in implementation). Therefore, programmers typically use the syntax extensions provided by their RDBMS of choice for such queries. Rewriting such queries in the specific platform of implementation (or in ANSI SQL) is always a feasible alternative however, and we provide them for Query 1 to illustrate.

SQL Server Syntax Query 1

```
SELECT TOP 5 N.ndiagnosis_code AS NANDA,
count(*) AS Freq
FROM [nursing_diagnosis] N JOIN visits V
ON (N.seq = V.seq)
WHERE V.age>=65
GROUP BY N.ndiagnosis_code
ORDER BY count(*) DESC
```

ANSI Compliant SQL Query 1

```
SELECT n1.Nanda, n1.Freq
FROM
(SELECT ndiagnosis_code as nanda, count (*) as freq
FROM nursing_diagnosis N JOIN visits V ON (N.seq = V.seq)
WHERE V.age>=65
GROUP BY ndiagnosis_code) n1,
(SELECT ndiagnosis_code as nanda, count (*) as freq
FROM nursing_diagnosis N JOIN visits V ON (N.seq = V.seq)
```

```
WHERE V.age>=65
GROUP BY ndiagnosis_code) n2
WHERE n1.freq >= n2.freq
GROUP BY n1.nanda, n1.freq
HAVING (select count(*) from
(SELECT ndiagnosis_code as nanda, count (*) as freq
FROM nursing_diagnosis N JOIN visits V ON (N.seq = V.seq)
WHERE V.age>=65
GROUP BY ndiagnosis_code)
) - COUNT(*) < 5
ORDER BY n1.freq desc
```

In connection with the second query, the original data had some zip codes in a 5-digit format, while others had a 4-digit suffix attached. To obtain all the zip codes in the standard 5-digit format, we first parsed the data (easily done by creating a view that takes a substring containing the first five characters; alternatively the data can be physically transformed) to get it in a suitable format, and then ran the query on the transformed values. In addition, a view for patients was created that contained their age (derived from their date of birth). A point to keep in mind when creating such date transformations is that functions for date manipulations tend to be DBMS-specific. When porting the schema and associated data over to a different platform, one may have to rewrite some of the code.

Given the relative complexity of Query 3, we felt it would be useful to provide additional explanations for it. The first sub-query finds the top five diseases for people over 65 years of age. The query then removes from this list (using the "NOT IN" operation) those diseases that are among the top five in the 50-64 (inclusive) age group. This leaves the (top commonly occurring) diseases unique to people over 65. The interpretation of a disease is based on the CCS codes. A single CCS code may encompass multiple ICD codes. We felt we would get a bigger picture of the diseases affecting patients in the sample dataset using a higher level of granularity. This also explains why we find CCS code "106" co-occurring with itself. Two or more diseases within the same CCS code (but distinct ICD codes) frequently co-occur with each other.

We noticed there were thirty-four such combinations such as ICD 42731 (Atrial fibrillation) and 42732 (Atrial flutter) appearing as the primary and secondary disease diagnoses respectively. This resulted in a total of one hundred and nine patient disease diagnoses where CCS 106 was the category for the primary and secondary disease codes.

For query 3(B), we could also write it using the code for 3(A) embedded within it as a sub-query (rather than using literal values for ICDs 106 and 127), but we chose not do so to preserve simplicity in display. Also for a specific historical dataset such as this one, the results will not change due to updates over time (thus, we do not potentially end up with inaccurate data due to dataset evolution even with the values hard-coded).