Contextual Persistence in Complex Information Systems:

An analysis of storing and retrieving information in an electronic health record.

By:

Timothy W. Cook

Contextual Persistence in Complex Information Systems:

An analysis of storing and retrieving information in an electronic health record.

Version: 1.0.2 TIMOTHY WAYNE COOK Date:05/12/06

In partial fulfillment of a

Masters of Science in Health Informatics,

Lancashire School of Health and Postgraduate Medicine,

University of Central Lancashire, Preston, England

PREFACE

This thesis is being completed during 2006. It is based on material I have studied and experiences gained over several years working with various database applications across multiple operating systems and network environments. Special emphasis is being placed on the work performed during the study in this Masters programme and my recent research on health care applications.

I would like to thank my best friend, wonderful tutor and wife; Dr. Nikki Shaw. Her support, encouragement and belief in me is the foundation for my taking the chance to tackle this course of study.

I most appreciatively thank my supervisor Dr. Jean Roberts for taking on such a difficult task as reining in my far too broad ideas and helping me focus on an achievable goal.

When I examine myself and my methods of thought, I come to the conclusion that the gift of fantasy has meant more to me than any talent for abstract, positive thinking.

Albert Einstein

ABSTRACT

This thesis is the result of a qualitative research study undertaken in order to determine a well defined theory for choosing the correct data storage model for healthcare data as used in electronic health records. The researcher views an electronic health record application from the point of view of the patient as a primary care patient. This perspective is analogous to the current system in place with paper records in many Western countries. The primary care (family physician) maintains a record that is the point of consolidation of all of the patient's healthcare issues. It will contain information such as hospital discharge summaries, immunizations, reports from other healthcare/medical specialists and other care providers. The family physician and / or the patient is the single point where all of this information will ever come together. The results of this study indicate that while any type of storage philosophy may be deployed for electronic health records, there are certainly trade-offs as far as performance both in the short term and the long term. The key issues revealed are the long term (many years) of **contextual integrity** that must be maintained and the application and database maintenance costs of various approaches over time.

INTELLECTUAL PROPERTY DECLARATION

I, Timothy W. Cook do hereby declare that this work is entirely my own work except where appropriately attributed. I also affirm that this work has not been submitted for any other degree or professional certification.

Timothy W. Cook

Table of Contents

Index of Tables

"Everything should be made as simple as possible, but not simpler."

--Albert Einstein

Chapter 1: Introduction

1.1 Aim

To evaluate the most common methods used in storing and accessing data used in electronic health record applications and present a cogent argument that one approach is superior for use in electronic health record application development. This argument will be based on a theory that is grounded in the information analyzed.

1.2 Objectives

- To identify data persistence models used in electronic health record applications.
- To select representative documents that describes the advantages, disadvantages and foundational aspects of the identified models.
- To develop a grounded theory based on analysis of the selected documents.
- To develop recommendations based on the emergent themes discovered.

1.3 Background

The process used to permanently store data from one iteration of an application to another is known as persistence. The use of the word contextual in this thesis is referring to the temporal (time related) as well as the health (usually viewed as chronic) problem related associations and relationships between components of information within an electronic health record. The word complex in this title refers to the complexity of the data relationships. The full definition of the title and therefore the purpose of this thesis is to examine the way personal patient data is stored for long periods of time (legal requirements vary from jurisdiction to jurisdiction but seven years beyond death is not uncommon) so that the data may be accessed and processed by electronic health record applications (generally primary care health record) as well as various community health and other research applications that may have direct access or through data extract sharing (given proper administrative agreements). All the while maintaining a valid view of the data so that the healthcare provider can translate that data into information about the patient now (at the point of care) or at some other point in time.

This thesis is motivated and probably biased by a discovery I personally made while developing an electronic health record application in 1998. Though I have built several applications using relational database engines for data persistence, I had not encountered a situation as (temporally) complex and information rich as an electronic health record. This complexity was realized as I studied the importance of maintaining the informational context, as described in the last paragraph, of the stored data for extraordinary lengths of time. The longitudinal nature of patient health information is of particular significance as I will point out below. In order for healthcare providers to be able to make well founded decisions based on electronic health record data, they must be able to view that data from multiple perspectives. The application must be able to recreate each of the perspectives, in its appropriate chronological context.

Diagram 1 below depicts this contextual perspective problem. The numbers along the bottom represent the age of the electronic health record. Note that the record may have started well after birth and live well beyond the patient's end of life event. This is a complex issue and therefore the reader may need to refer and re-refer to the description provided while analyzing the diagram. The rectangles with the solid color background represent health problems(subsequently referred to as 'problems') that the patient has experienced throughout their life. Note some problems may re-occur over time in combination with different problems. In order to assess the health status, cause of events and therefore the care requirements of the patient at various periods in time . A healthcare provider may need to view the data from various perspectives (the objects with the transparent background). Those views may include various combinations of problems over various time periods. Sometimes, during interventions, only certain problems are viewed irregardless of the time period. The point is that these need to create these views of information cannot be predetermined by the application designers/developers because even the physician nor the patient knows today what combination of problems they may have in the future.

To help clarify this issue I will present a more concrete example and description of the diagram. The X axis (horizontal axis) indicates periods of years. The Y axis indicates potential information samples required by the healthcare provider in order to make a health management decision. The colored horizontal bars represent a specific health problem that exists for a specific period of time. Again, the outlined, though transparent (white) background elements of the picture represent specific information views of the patient's health by the healthcare provider. For example, the dark brown (more or less) rectangular outline demonstrates that a healthcare provider needs to look (or is looking) at a patient approximately 50 years old; back over time, to where this patient was 18 – 20 years old (onset of adulthood?) During that 30 – 32 year view, the healthcare provider will see that this patient has had the various health problems represented by the light blue, dark blue, yellow and gray 'problems'. This diagram/situation demonstrates how primary care providers assimilate bits of information into a diagnosis and treatment plan for the patient.

.

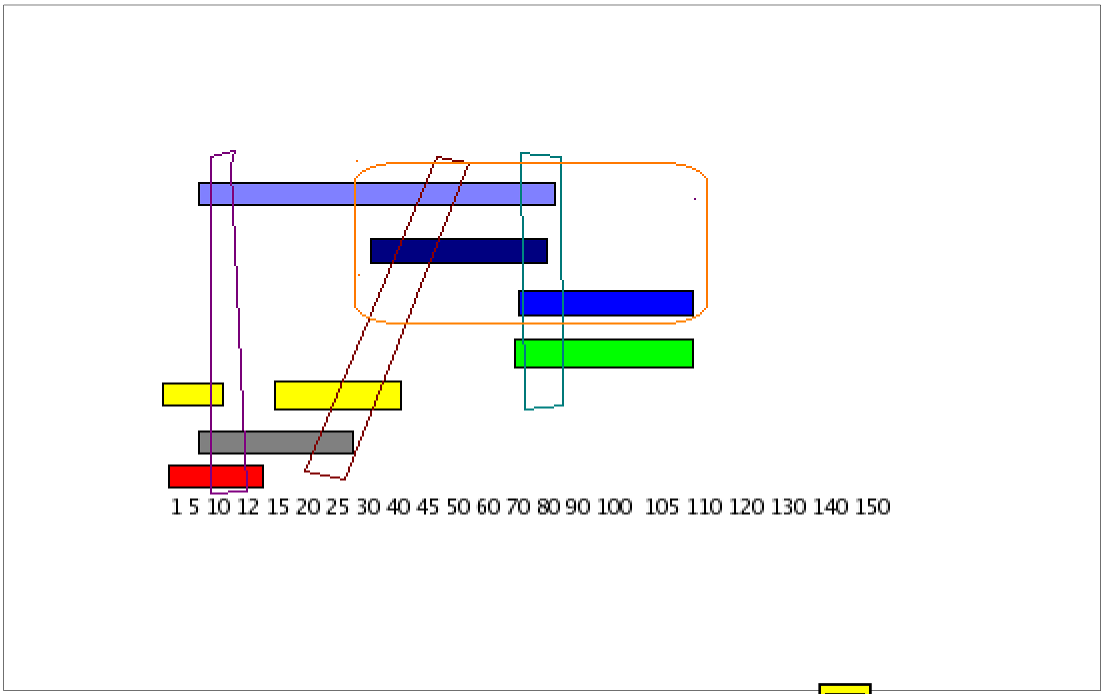

Diagram 1. The problem of contextual views in an electronic health record.

It must also be said that each of those problems indicated in Diagram 1 are composed of multiple encounters with what is likely input from a varied number of healthcare providers. Each encounter will likely represent multiple data points in multiple problems.

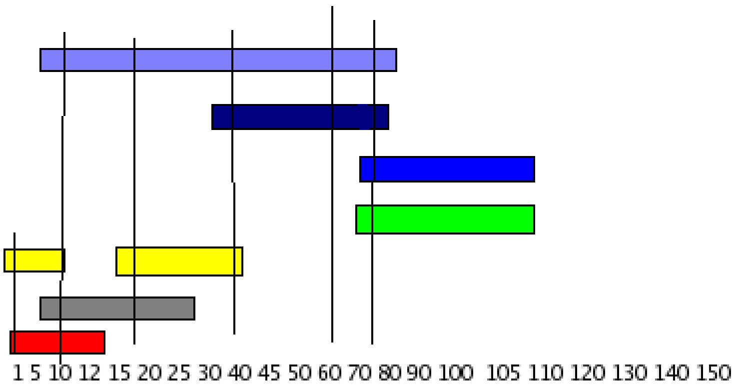

Diagram 2. Encounter cross-sectional view

Since the user needs cannot be specifically predetermined and therefore built into the application, the infrastructure must be able to provide a dynamic retrieval and view process.

A related but slightly different problem is in the use of external vocabularies. It is clear to me that when there is an error located in or a change made for clinical reasons to a external, linked vocabulary that these changes should be distributed to all end users of that vocabulary. However, those changes should only be reflected in future patient encounter entries of the electronic health record application. In no case should the historical content or context of the record be altered. This is especially important with regards to these vocabularies. But, as referential integrity constraints typically enforce in a relational database system and as is recommended by Quality

Control section of the NHSIA [NHSIAError2005], past patient encounter entries may be modified in ways that create a false contextual situation in the patient record. The actual encounter data may or may not be changed directly by the application. However, if the rubric of a coded entry is changed then the code no longer points to the text the clinician selected at the time of entry. Irregardless of the reason for the change this can cause changes in patient records with consequences (in both patient treatment terms and in statistical profiling of activity in general) that are unknowable at the time of the change.

This issue is relevant with any coding system. For example, using the International Classification of Diseases Ninth Revision, Clinical Modification (ICD-9-CM) this simple set of tables illustrate the concepts described above. Table 1-1 shows selected ICD-9-CM codes prior to the October 2005 update. Table 1-2 shows the same codes after the October 2005 update.

Code	***Text***
285.0	Sideroblastic anemia Anemia: sideroblastic : refractory
292.8	Other specified drug induced mental disorders
292.89	Other Drug induced sleep disorder

Table 1-1

Code	***Text***
285.0	Sideroblastic anemia Anemia: sideroblastic : Excludes: refractory sideroblastic anemia (238.7)
292.8	Other specified drug induced mental disorders
292.85	Drug induced sleep disorders Drug induced circadian rhythm sleep disorder Drug induced hypersomnia Drug induced insomnia Drug induced parasomnia
292.89	Other

Table 1-2

These two examples demonstrate how a diagnosis made in February 2005 can easily be misunderstood in November 2005.

In the first case 285.0 clearly is specific to refractory sideroblastic anemia prior to October 2005. Whereas after October 1st,2005 code 285.0 specifically EXCLUDES the refractory (resistant to treatment) type of sideoblastic anemia.

In the second case the 292.8 hierarchy of “Other specified drug induced mental disorders” demonstrates what happens when a new code is added to provide for a more specific diagnosis. Before the October 1st , 2005 change, 292.89 indicated a drug induced sleep disorder. After the change, 292.89 is now just an “Other” drug induced mental disorder. It is no longer associated as a sleep disorder with the addition of 292.85. One can easily imagine a point in the near future where the four instances of 292.85 will be separated out into individual new codes as well.

These examples are a minor example of the changes made in one vocabulary over a one year period. The issue cascades into a completely unmanageable nightmare over the lifetime of a health record utilizing multiple external vocabularies.

Clinical coding is only one area where I find these contextual discrepancies in electronic health record application designs based on the relational model. Another and possibly more obvious are in drug formularies. Typically a subset of all manufactured drugs are made available to a clinical application. This subset selected may be defined for many different reasons such as end user's choices, payor coverage or simply what's available locally. In any case these formularies typically (e.g. http://www.multum.com/Lexicon.htm) include at least information on name, category, dosing, ingredients, strength, route and form. What happens in the clinical application when a dosing change (50mg to 75mg) and/or a form (5mg/ml elixir to a 10mg tablet) is made in the formulary concerning the drugs available? We of course know that these will necessitate a new drug id number but what happens to the old entries? If their definitions are removed from the formulary database, how is the update handled in the patient record? This is not an easy question to answer and the question itself is often overlooked in electronic health record applications as there is not always a consistent replacement term. Typically the change may introduce a granularity or specificity not previously present. If the definitions are withdrawn rather that become unavailable for allocation to any other incidence, how are the older instantiations ever interpreted again? These questions do not have good answers in the existing systems I have reviewed. Over time, it is expected and has been previously the method of operation by the database distributors, that these no longer used entries will be phased out at some point. Though no further new use of a previous term does not imply explicitly that their previous uses are now unusable, therefore until all records that used the now defunct codes are trashed at the end of their useful life-cycle there will be a need to retain the codes and their definitions indefinitely . However, the database distributors of formularies and coding systems have no way too know when that might occur. In the case of a formulary, enough of the information to complete the prescription may be stored in the patient record as a prescription. But then we have to ask if the application depends on any of the other information from the formulary table(s) such as to do with drug-drug or other interaction testing. If so then what happens when the prescription still exists as a part of patient history yet the details are no longer available regarding the drug?

By way of one more example we will look at the strategies developed and promoted to update the Read Codes between versions and specifically in converting to SNOMED-CT [NHSCFHSnomedCT]. We can see that accepted concepts of referential integrity and relational database normalization (to some level) are desired and expected to be maintained. The recent alert from the National Health Service Connecting for Health programme (NHS CFH)[NHSIAError2005] regarding this issue demonstrates this retention approach as a mandatory core concept. In the paragraph describing the quality assurance procedures, the author of the NHSCFH paper clearly states that referential and semantic integrity are cornerstones of their efforts. I can only assume that semantic integrity in this case relates only to the SNOMED-CT tables. As is clearly apparent, semantic integrity cannot be assured for every end-user application that this vocabulary interacts with, as it is seminally defined in[McLeod1976].

The above are just some of the questions I asked myself when looking at existing applications and while I was doing 'worse case' data modeling for a new, open source electronic health record application. The easy answer is that the reference data such as drugs and vocabulary entries are never deleted from their tables. However, that is not the reality for the patient electronic health record, and as is currently seen, the electronic health record application designer can count on data being changed and deleted from these external sources of reference data over the coming years. This presents many problems for patients and clinicians. It is critical from a clinical point of view as well as a medico-legal point of view that a patient record can be viewed as it existed at any point in time. I call this maintaining the **contextual integrity** of the record. Having this capability is key so that a patient's condition can be reviewed in the context of what was known about the patient as well as what was known in medicine (reference vocabulary content) at any point in the life of the record. This capability is very important to the clinician in case of a legal dispute that involves one or more patient's records. This capability is very important to patients since it can effect the care they might receive in the future. It is also important to healthcare practitioners as use of the contemporaneous 'appropriate term / code' in generating activity profiles for facilities management and epidemiology is crucial.

This capability should also be important to community health specialists as they review issues arising in the future.

It was at this point, based on the above discoveries, that I realized that using a correctly modeled (fifth normal form) relational database to store patient information was not going to be desirable. In order to insure that patient data was valid throughout the life of the patient record, the application had to store duplicate data and not rely on the reference tables after the clinical code or drug was selected for use. This can be done using relational databases however, once an implementation strays from the rigor of the relational model then there is additional complexity required in the application in order to insure the proper integrity of the data is maintained. This integrity management is typically the domain of the relational database engine's data manipulation language (DML), where a set of constraints can act on data independent of the number of applications accessing that data. Once that basic concept of a relational model implementation has been broken then the key values of using the relational model no longer applies.

Using my anecdotal approach I have encountered many obstacles in sharing this information and point of view with developers and academics involved in data modeling. It has therefore become my mission to validate this position in a thorough academic manner. I may find that I am incorrect in my assertions. But, whatever the outcome of this research, I will have established a solid foundation for electronic health record data persistence choices.

1.4 Ethics

This study reviews only publicly available information that does not contain personally-identifiable data therefore, after taking expert advice at the university, no ethical review was undertaken.

In literature as in love, we are astonished at what is chosen by others.

Andre Maurois

Chapter 2: Literature Review

A thorough search for literature on the subjects at hand was conducted yielding many thousands of documents. Several hundred documents were read and many are included as additional resources for the reader in the Bibliography even though they were not directly analyzed and or referenced in this thesis. The directly referenced documents are found in the References section.

The search terms, number of matches and electronic databases/journals searched are provided in Table 1.1 below. The terms used, journals and databases searched were chosen based on an attempt to extract as broad a range of publication type and location as possible. While reviewing the descriptions and abstracts of the tens of thousands of documents resulting from the searches, I used a mental selection method of content pertinence that was bounded by the goal to select information describing specific issues around data modeling using the relational model, an object-oriented approach or combination approaches. As described in the Background section of Chapter 1, other data persistence modeling approaches (such as the network and hierarchical models) had been removed from consideration prior to the literature review. This removal process was based on technical applicability in modern computing environments.

Database/Journal	***Search Terms***	***# of Matches***
Journal of the ACM[1]	database relational object	346
Journal of the ACM[1]	object persistence	1,923
Journal of the ACM[1]	database “object persistence”	1,291
Journal of the ACM[1]	“object database” “query language”	355
Amazon.ca – Books	relational database theory	686
Amazon.ca – Books	Object database theory	885
Amazon.ca – Books	Database query language	2232
Amazon.ca – Books	SQL "object database"	194
ACM Transactions on Information Systems	"object database" "query language"	57
ACM Transactions on Information Systems	"relational database" "object query"	67
ACM Transactions on Database Systems	“query language”	216
ACM Transactions on Database Systems	"object query language"	5
ACM Transactions on Database Systems	“object persistence” database	411
Wilson Web; Library, Literature & Information Management	relational AND database	471
Wilson Web; Library, Literature & Information Management	object AND database	327
Google Scholar	"object database" theory	5220
Google Scholar	"relational database" theory	17,300

1 Using Advanced Search requiring all words and phrases

Database/Journal	***Search Terms***	***# of Matches***
Google	"object database" "market failure"	111
Google	"relational database" "market failure"	293
Google	"object query language" “object database”	63,900
Google	"relational query language" "relational database"	901
Google Scholar	"object database" AND "query language"	2550
Google Scholar	"relational database" AND "query language"	11,100

Table 1.: Literature Review Process

From the matches in Table 1.1, I selected the most applicable documents for review based on subject matter applicability and newest publication dates. From those documents I categorized each where it would best fit into the methodology framework as described in Chapter 3. Once each document was categorized I selected the most definitive in each category on which to perform the content analysis.

If you don't know where you are going, any road will take you there.

Lewis Carroll

Chapter 3: Methodology

The participatory action research methodology was first considered as it has proven application in information systems research as described by [Avision1999] and [Baskerville1999]. Though there is a great deal of local expertise in this field with luminaries such as Professor Trevor Wood-Harper at the University of Manchester (http://www.informatics.manchester.ac.uk/school/staff_details_ac.php?staff_id=ATWH#profile) close by, the scope of such a project would be beyond manageability in the current environment of this thesis. I made the decision to use a document analysis methodology since it still captures the essence of the conundrum between the various (and specifically two selected) data persistence models. This decision preceded the literature review as described in Chapter 1. The primary difference is that document analysis will capture what has been done and said over the past many years by many experts. Whereas, participatory action research would only capture the effects and affects of the limited projects that I could be involved in personally. I then considered this a major positive effect and concluded that document analysis was the more appropriate methodology to engage.

This qualitative study was undertaken using document analysis as an unobtrusive means of analyzing communication and interaction artifacts between information technology professionals. This method of analysis provides insight into complex models of human thought and language regarding the implementability and use of complex information systems. Without analyzing specific applications this research looked at the logical models of persistence and what factors play into the decision to use each model. It is expected that this research and subsequent analysis will yield theory and strategy for selecting a proper model that is grounded in the content of the documentation and expert discussion for use ultimately in practical application solution implementations and possibly continued academic debate .

The history of data modeling has yielded many models or model approaches. Most of which have proven to be unusable or obsolete in terms of modern computing and have therefore been dropped from popular usage as applications of technology became more complex there were limitations discovered causing obsolescence. These models include the Network, CODASYL and Hierarchical models. Though not generally in use, these models can be considered foundational as many of their better features are found in the newer models. From the models reviewed, two were chosen for detailed analysis based on the fact that there are database engine implementations in use in practical applications.

The chosen models are the relational model and the object model. There is the hybrid object-relational which I had originally considered a separate model. During documentation review it later became apparent that the object-relational model was simply an attempt of vendors of relational database engines to overcome the problems with the relational model in incremental steps in order to address marketing issues without actually correcting the overall problems.

This research contrasts the advantages and possible pitfalls of each of the two selected models. I utilized information from the review of relevant text books, peer-reviewed journal articles, white papers, marketing materials, public mailing list entries and website entries (expert discussion), standards documents and white papers as detailed in the tables below.

	Advantages	***Disadvantages***	***Foundational***
Peer Review	[Codd1979]	[Kim1992-2]	[Codd1970]
Text Book	[Date2005]	[Dietrich2005]	Not Applicable[1]
Expert Discussion	[Debunk02]	[Debunk01]	Not Applicable[1]
White Paper	[Oracle2005]	[Objectivity2002]	Not Applicable[1]
Standards	[SQL2002]	Not Applicable[2]	Not Applicable[1]
Marketing Material	[Oracle2003]	[Versant2001]	Not Applicable[1]

Table 2: Relational References

	Advantages	***Disadvantages***	***Foundational***
Peer Review	[Dingle1998]	[Codd1979]	[Dahl2001]
Text Book	[Kim1992]	[Date2004]	Not Applicable[1]
Expert Discussion	[DBTheory2006]	[SlashDot2005]	Not Applicable[1]
White Paper	[Srinivasan1997]	[Oracle2005]	Not Applicable[1]
Standards	[ODMG2005]	[SQL2002]	Not Applicable[1]
Marketing Material	[Versant2001]	[Oracle2003]	Not Applicable[1]

Table 3: Object References

The process of discovering and selecting each of these documents was presented in Chapter 2.

The justification for choosing each of the document types is as follows:

1 These materials,lacking peer review, cannot be considered foundational as a scientific piece of work.

2 Not applicable in this context.

1. Peer Reviewed Papers: academically credible sources of information

2. Text Books: review what students have been and are being taught regarding data modeling choices

3. Expert Discussions: discussions, while anecdotal, representing practical experience of professionals at work

4. White Papers: studies based on practical application implementation experience though often affected by company market positions

5. Standards: nationally and internationally recognized best practice methodologies for implementation

6. Marketing Materials: materials that have the effect of coloring the technical perception of experts with or without provable foundations in order to present in best light the perspective of a particular vendor solution

Each selected document was read first for appropriateness and then re-read for the purpose of developing categories (or themes) that may be used in comparative analysis. Comparative analysis is a process used to develop either a substantive or a formal theory. In this case I am looking to develop a substantive but practical application theory. The elements of theory development generated using comparative analysis are:

1. Conceptual categories and their corresponding conceptual properties.
2. Generalized relations among the categories and their properties.

By definition, categories stand on their own as a concept. Properties are tied to a specific category or categories and do not produce any information outside of that context.

My process for developing the categories will begin with paragraph by paragraph analysis of the documents recording keywords and phrases. I will determine significant categories exist in cases where the same subject area (via the keywords and phrases) is being used and developed by some majority of the documents. It is expected that categories will be developed in different ways and with different approaches (the properties) but the subject will still emerge.

Once these categories emerge and their properties are explored I will examine relationships between them and begin to integrate the categories into one or more concepts or hypothesis.

The generation of theory grounded in data requires joint and continuous activity in data collection, coding and analysis. This notion of theory development, as a process, is key to the successful use of this methodology.

Statistician: A man who believes figures don't lie, but admits that under analysis some of them won't stand up either.

Evan Esar

Chapter 4: Analysis

I utilized information from the review of the selected relevant text books, journal articles, application documentation, public mailing list entries and white papers. This information was coded, categorized and analyzed in order to ascertain themes using the grounded theory methodology as described by Glaser and Strauss[Glaser1967] and later (and further) clarified by Glaser[Glaser1992]. The outcome of this analysis should reveal the best and most appropriate method of persisting clinical data for an electronic health record application. The analysis process was meticulously tracked and recorded as described in this chapter.

4.1 Category Development

Categories are extracted from data through a process of careful reading and understanding of the author(s) intent. Each of the categories consist of properties that may be made up of words or short phrases. However, according to Glaser and Strauss[Glaser1967] these properties can be, and generally are, presented in a discussion type format so they maintain their context. It is important that each of these categories have appeared in multiple documents but that they present essentially the same context from one document to another. This draws the relationships between the documents and enforces the validity of the concepts as categories.

The categories that emerged from the data are:

- Evaluating Implementations
- Data Quality
- Purpose of a Database
- Confusion Between SQL and a Relational Database
- Data Complexity
- Data Reliability and Availability
- Generalized Query Language
- Data Storage Models
- Data Independence
- Database Design

Below I provide my analysis of each of these categories and its properties.

Evaluating Implementations

The relational model provides normal forms (preferred formats or approaches) as a standard to evaluate real world implementations against the mathematical model. SQL (see below) provides the common data access and data manipulation language. Object databases have no such underlying formal model. as part of the Object Management Group (http://www.omg.org) there is an Object Data Management Group (ODMG) 'standard' that is really a feature list rather than a true technical standard. Work is under-way on development of a 4th Generation OODB Standard.[ODMG2005]

Essentially a relational database is judged on it's structure. An object database typically stores data as it is created inside the application. This is where the data model or structure directly translates from the application to the storage media and framework. The object oriented application that stores it's data in a relational database will be required to translate that object data into the structure of the relational database.

The relational database implementation uses a unique primary key to identify tuples of data inside each of it's relations (relations are sometimes called tables). That primary key is made up of one (or more in combination) attribute (sometimes called a column). This primary key is 'prime' or 'unique' within the relation (table) and the data has a relationship to the remainder of the tuple. The object database typically uses an object identification property (attribute) that is said to have no semantic relationship (in the real world) to the data (properties) of the object it is associated with. Since there is no data manipulation language that can apply to all objects in a database then the application language usually takes care of those tasks.

Data Quality

The relational model uses normal forms (preferred formats) to insure data consistency by preventing data duplication as well as avoiding update anomalies. However, due to poor performance by database engine implementations it is common practice to sacrifice many of the data quality insurance aspects of the relational model in order to provide the end user with acceptable performance metrics.

Purpose of a Database

Interestingly enough this issue comes up quite often. The purpose of any database is to store information (usually large amounts) in a manner where that information can be retrieved according to the needs of the user. Admittedly there are many contextual issues surrounding this definition.

For many practical reasons object oriented design using object based languages have become the chosen approach to new application development. The primary reason is considered to be the re-usability of source code thereby reducing application complexity. Therefore it is essential that applications be able to persist the collected and calculated data in a manner that is valid and consistent with the application. The act of persisting this data is usually in some sort of database that helps insure the quality of the data.

In a relational database it is assumed that the standard data manipulation language (SQL) can and will handle those constraints. In an object database there is no specific data manipulation language therefore these functions must be handled by the application language and/or a custom application programming interface (API) for the database.

The common thought process is to use a database to store persistent data for use from one execution to another of an application or a process of an application. The three approaches of storing object data from applications are:

a) adding object persistence mechanisms to a relational database

b) adding object extensions to older type storages (i.e. Hierarchical).

c) using an object oriented database with an API in the chosen application language

As is usual with the kinds of decisions, each approach has it's pros and cons and must be evaluated in the context of the chosen application.

Since object-oriented modeling and development have become the accepted standard in application development it is important to be able to persist object data. Object data really consists of attribute values that are maintained in context with the object itself.

<u>Confusion Between SQL and a Relational Database</u>

There is widespread confusion between the relational model and databases that operate using the Structured Query Language (SQL) standard. There are no known implementations of database engines that truly operate as relational databases. A simple concept but major factor is at the root of this confusion. As the relational complexity of data increases, the ability of database engines to manage joins and still perform at an acceptable level diminishes. These joins are necessary to logically connect relations in order to present attributes from multiple relations as a new relation. This new relation is often called a view. This performance problem has led vendors to make compromises in adherence to the relational model. One of the key compromises is in allowing the use of NULLS. The relational model is a two valued model only. The NULL provides for a third value which is not defined in the relational calculus. There exists in the industry what are called 'flavors of NULL'. These flavors can be defined as; empty, not defined, unknown value or other definition that is application specific. The use of NULLS can have serious negative consequences on the data retrieval process.[Date2006]

There are many implementations of SQL databases. Some of them use 'extensions' to the standard in order to perform certain proprietary functions. When used in this mode, those engines are no longer enforcing relational integrity nor are they standard SQL databases.

There are those that profess that object-relational (O-R) systems extend the relational model. However, the relational model is really quite simple as it defines relations that are composed of tuples of attributes. O-R systems simply add type extensions to existing database engines. Typically these suffer the same fate as other SQL database implementations.

Data Complexity

Medical and even more so, healthcare data is very complex in the various relationships and the conceptual richness of it. The ability of an object-oriented database to aggregate data into many different views is a large advantage over the relational database. The relational database engine suffers serious performance issues as the number of joins increases due to these complex relationships.

In addition to healthcare data, these issues are well known in the fields of computer-aided manufacturing, computer-aided design and geographical information systems[Peuquet2002]. In these fields the complexity comes more from spatially complex data[Rigaux2002] but the issue is the same or similar to the temporal complexity found in healthcare. This concept is covered in later chapters of[Peuquet2002] as well.

Data Reliability and Availability

Certainly a corner stone of the relational database community is the ability to maintain data reliability and availability across multiple applications accessing the same database. This is possible because of the implementation of a standard generalized data definition and data manipulation language (SQL). SQL can manage constraints on the data inside the database. Whereas in object oriented databases these task fall to the applications themselves and must be synchronized across applications. One other approach using object-oriented databases that is gaining favor is to build a service application layer for all applications to go through to access data in the database. This Service Oriented Architecture (SOA) approach is increasing in popularity but still suffers from the fact that the application interface must be written for the design of each individual database.

Generalized Query Language

Having a common query or data access language is said to be important for ad-hoc queries and is one of the prime motivators for using a relational database instead of an object-oriented database. This ad-hoc query capability is supposed to allow end-users to write queries and retrieve information from a database without having to go to the programmers for assistance. In the early days of relational databases there were several query languages. It seems that Structured Query Language (SQL) won the market when IBM selected it as the language of choice for their DB2 engine in the late 1980's and at that point became a 'de-facto' standard.

SQL later became a formal standard but even today each implementation of it is slightly different requiring modifications to the data and the constraint instructions when an application is moved from one database engine to another. Of key importance are the semantic differences in how the database engines handle NULL values, column (attribute?) function and duplicate row deletions. It is clear that SQL is optimized for each platform and does not adhere to the relational model as defined by E.F. Codd.[Codd1970]

In [Oracle2005] the authors describe enhancements, that are really non-conforming extensions to the SQL standard, that are essential for creating scalable high performance applications. Examples of these applications include bio-informatics, clinical medicine and eBusiness. 'Regular Expressions' (a common search syntax used in many computing environments) are used in the database to move some of the programming from the client or middleware into the database. However, the client or middleware interface must be developed specifically to understand the format of the returned values. This certainly is not part of a standard query language implementation.

SQL does bring the ability to define cross application data integrity constraints as well as provide ad-hoc query capability. Indications are that it would be fair to say that very few end-users actually implement their own SQL queries. These queries are sometimes created by end-users via a report writing tool but more often that not it takes someone with special knowledge of SQL and or the report tool to do this. These specialists are usually experts on a specific database engine or junior programmers familiar with one or more database products.

The lack of an ad-hoc query language as well as data integrity constraints being written in the object-oriented programming language are two of the major detractors of object-oriented databases. The Service Oriented Architecture approach to data access is a certain answer to the data integrity issue. The use of runtime scripting languages like Python (http://www.python.org) could easily be an answer to an ad-hoc query language for an object-oriented database.

Data Storage Models

In preparing to select a data storage model it is important to be aware of the various approaches that have been used.

File management systems maintain data items as simple blocks of data on the storage media. There is no knowledge in the file system as to what is contained in the file. Only that it exists in a certain space on the storage media.

In hierarchical databases each record, aka. component of interest, is linked to a parent record and one or more children records. This creates a tree that can be navigated very rapidly. The downside is that it is difficult to see other relationships in the stored data other than the parent-child connection.

The network database extends the hierarchical model by allowing records to be linked to multiple parents. This increases the ability to create cross record relationships. These relationships can be expressed as sets of data and provides for much more powerful applications.

The chief disadvantage of the hierarchical and network storage models is that the application contains all of the knowledge about the structure of the data. This generally means that the application programmers must create all of the queries for the application. This can create bottle-necks in production and use of the application.

The relational model abstracts the data structure in the database from the application data. A data model implementing the relational model will consist of relations defined by one or more attribute name:type pairs. The relation will consist of tuples containing data matching those attribute name:type pairs. For the convenience of two dimensional drawing, these relations are often drawn as tables made up of columns and rows. There are no true implementations of the relational model as defined and refined by E.F. Codd. While many of these SQL databases exhibit the flavor of being relational there are compromises made due to performance issues when managing complex data. Where complex data is defined as data that has multiple many to many relationships. This is an important point since many supporters of SQL databases will point out that the relational model is based on a mathematically provable model. This is a true fact. However, no database engine fully implements the relational model and SQL allows and even encourages use of NULLS and non-optimized relations in order to improve performance.

Data Independence

Data independence as expressed between the application and the database is a desirable concept. This feature provided by a database engine provides a capability of adding multiple client applications to one database while maintaining the structure and the integrity of the data separate from all of them. The use of an object-relational mapping tool provides a middleware solution to matching up the object model of the application to a database using a different (i.e. relational) model. However, it must be noted that the process of object-relational mapping is estimated to add 30% or more to the line count of code in an application. A large majority of maintenance hours on applications that use object relational mapping are spent maintaining those links between the application object model and the relational database model.

Database Design

As described by[Date2005] , database design theory is a separate theory process from the relational model. Then Date goes on to say that, “it is appropriate to think of it as part of relational theory overall”. Comments such as this in text books and other publications can lead developers to believe that there is only one database design approach. Database design is a fairly subjective and almost creative process. Database design is also called database modeling. Arguably the definitive text on data modeling[Simsion2005] describes database design as the process of physically implementing the conceptual-cum-logical design. These designs are the result of the modeling process so it is impossible to separate the two in any real sense. Whether called modeling or design, the significant concepts are the differences between the conceptual design where all of the required data elements are discovered and recorded as a communications device between the domain experts and the database experts. The logical logical design is the follow-on component of the database design process where the application data is diagrammed and documented regarding relationships, data types etc.

It must be pointed out that nothing in object modeling precludes good relational design. In[Dietrich2005] the authors promote using object-oriented design using the extended entity relationship approach. However, there is nothing to prevent the use of the Unified Modeling Language, with it's richer syntax and extensibility, as a relational modeling tool.

It is the mark of an educated mind to be able to entertain a thought without accepting it.

Aristotle

Chapter 5: Discussion

In a general discussion surrounding what this research means in the context of electronic health records and other clinical application development I would like to present the following assertions as truths discovered during this research.

1. Any data persistence model may be valid for clinical applications.
2. Project budget (e.g. overall economics) and the project time line will ultimately control the selected approaches.
3. Expected lifetime of the application will have a large impact on the persistence decision.
4. SQL is seldom if ever implemented by end-users.

How do these four concepts relate to the findings from the analysis? Certainly the relational model is the most valid and appropriate theoretical model to use for any application. **"IF"** the issues surrounding outside vocabularies are addressed by the vendors of the vocabularies. In addition to the outside vocabulary problem related in the Introduction, there are no true implementations of the relational model. This results in the fact that there are concessions made for the benefit of acceptable performance levels. This means that 'many to many' joins that are required by healthcare applications in a truly relational application must be sacrificed so that there is data duplication and other contradictions to the normal forms required for a relational implementation.

Facing reality we must also realize that there are always economic constraints to a project. Far too often these are short-term (short-sighted) project goals so that project managers and bureaucrats may meet their quarterly, semi-annual or at best annual goals.

Certainly we can see that temporal context degrades over time in a typically modeled relational (SQL) database system. This should tell us that it is important to build applications that will exist for the duration of a patient record (c. 125 years). Though short-term (c. 20 years) thinking is beneficial on an individual basis, it should be morally incumbent upon us to develop applications that may need to last hundreds of years into the future to provide data for epidemiological and population studies yet to come.

There are risks and costs to a program of action. But they are far less than the long-range risks and costs of comfortable inaction.

John F. Kennedy

Chapter 6: Recommendations

Recommendations to industry and academia that are based on the results of this research:

1. Stop naming SQL database engines as relational database engines. To do so subverts the relational model and discredits the late Dr. Codd's excellent work.
2. Correctly present to students and new database information technology workers that there are trade-offs between object oriented and relational databases but that both are valid approaches if the proper approach is utilized in the design.
3. Teach students and new database designers that relations are not tables, attributes are not columns and tuples are not rows. While the two-dimensional versions (tables, columns, rows) or views are convenient they are not the same thing.

A hero is no braver than an ordinary man, but he is braver five minutes longer.

Ralph Waldo Emerson

Chapter 7: Future Plans and Reflections

Things I want to pursue in this area in the future are to review curricula in the way students are taught about data storage. To teach the current generation of educators not to follow industry for industry's sake. Yes, an entire industry can go in a wrong direction based on marketing money and a sheep like mentality.

A few years ago a French diet allowing adherents to eat whatever they wanted in any quantity became very popular. A US nutritionist interviewed about it on TV commented that the diet's author learned what Americans have known for a long time: you can become popular and rich telling people what they want to hear.

Chapter 8: Conclusion

The results of this research are supposed to be substantiation of the premise that a theory, grounded in the data, emerges that can be used to select the correct type of persistence model for electronic health record applications.

It would be wonderful if I had succeeded in that quest. I however, must admit that there are many more issues involved in selecting the appropriate persistence model than simply selection **the best one**. I have however shown that the relational database (really the SQL database) should not be automatically chosen instead of the object or network database. I have also cautioned the industry against blind acceptance of the relational model as being appropriate for clinical applications and more specifically primary care electronic health record applications.

Appendix A

While there are scores of related reading materials on this subject I would like to point out a couple that have assisted me in determining where the roots of these issues lie.

Joint Cognitive Systems, Foundations of Cognitive Systems Engineering, (2005) Erik Hollnagel & David D. Woods, ISBN 0-8493-2821-7

Qualitative Research in Information Systems, A Reader, (2002) Michael D. Myers and David Avison, Editors. ISBN 0-7619-6632-3

Advances in Databases and Information Systems, 7^{th} East European Conference, ADBIS 2003, LNCS 2798, ISBN 3-540-20047-9

Glossary

Below are some terms and phrases that may not be understood by the casual reader or that may be defined or understood with different meanings based on their context within this thesis.

Contextual: The ability of the application to reproduce all information, that has been entered, about a patient as it would have existed at any point in the past.

Persistence model: The approach used to manage the long term storage and retrieval of data from within an executing computer program so that the data exists over long periods of time beyond that of the executing program.

Complex: Levels of complexity exist in the context of the multiple contexts that the user deals with. This is in contrast to complexity in the data itself.

Information System: A system, whether automated or manual, that comprises people, machines, and/or methods organized to collect, process, transmit, and disseminate data that represent user information.

Regular Expressions: Regular expressions consist of constants and operators that denote sets of strings and operations over these sets. There is a common syntax and the language is often abbreviated as 'regexp'. In this setting it can be considered a language used for searching for string patterns.

References

NHSIAError2005 NHS CFH ALERT NOTICE: NOTIFICATION OF ERROR July 2005 SNOMED CT UK Edition2005

NHSCFHSnomedCT NHS Staff Snomed CT FAQ 2006 http://www.connectingforhealth.nhs.uk/technical/s

McLeod1976 D. McLeod HIGH LEVEL EXPRESSION OF SEMANTIC INTEGRITY SPECIFICAATIONS IN A RDBMS1976

Avision1999 David Avison, et.al Action Research 1999

Baskerville1999 Richard L. Baskerville Investigating Information Systems with Action Research 1999

Codd1979 E.F. Codd Extending the Database Relational Model to Capture More Meaning 1979

Kim1992-2 Won Kim On Unifying Relational and Object-Oriented Database Systems 1992

Codd1970 E.F. Codd A Relational Model of Data for Large Shared Data Banks 1970

Date2005 C.J. Date Database In Depth; Relational Theory for Practioners 2005

Dietrich2005 Suzanne Dietrich & Susan Urban An Advanced Course in Database Systems; Beyond Relational Databases 2005

Debunk02 Fabian Pascal, et.al. 2005 http://www.dbdebunk.com/page/page/2532683.htm

Debunk01 2003 http://www.dbdebunk.com/page/page/754911.htm

Oracle2005 Oracle Corporation Oracle Database 10g Release 2A; Revolution in Database Technology 2005

Objectivity2002 Objectivity Corporation Hitting the Relational Wall 2002

SQL2002 James R. Groff & Paul N. Weinberg SQL The Complete Reference 2002

Oracle2003 Oracle Corporation New Capabilities in Oracle 10G 2003

Versant2001 Versant Corporation Objects End-toEnd The ODBMS Advantage 2001

Dingle1998 Adair Dingle, David Umphress & Russ Moul Explication of Object-Oriented Databases: A Software Engineering Perspecti 1998

Dahl2001 Ole-Johan Dahl The Birth of Object Orientation: the Simula Languages 2001

Kim1992 Won Kim Introduction to Object-Oriented Databases 1992

Date2004 C.J. Date An Introduction to Database Systems 2004

DBTheory2006 grigorie...@yandex.ru, et.al. comp.databases.theory thread on 'The OverRelationalManefesto ' 2006 http://groups.google.com/group/comp.databases.the

SlashDot2005 Why Aren't You Using an OODB? 2005 http://developers.slashdot.org/article.pl?sid=01/

Srinivasan1997 V. Srinivasan & D.T. Chang Object persistence in object-oriented applications 1997

ODMG2005 Bernhard Haas, et.al. 4th Generation Standard for Object Databases On the Way 2005 http://www.odbms.org/index.html

Glaser1967 Barney G. Glaser & Anselm L. Strauss The Discovery of Grounded THeory: strategies for qualitative research 1967

Glaser1992 Barney G. Glaser Basics of grounded theory analysis 1992

Date2006 C.J. Date & Hugh Darwin Databases, Types and the Relational Model; The Third Manefesto 2006

Peuquet2002 Donna J. Peuquet Representations of Space and Time 2002

Rigaux2002 Phillippe Rigaux, et.al. Spatial Databases with Application to GIS 2002

Simsion2005 Graeme C. Simsion & Graham C Witt Data Modeling Essentials 2005