Full/Regular Research Papers CSCI-ISHI

Healthcare Big Data Normalization Graph Theory Implementation

Atif Farid Mohammad
Senior Research Scientist
Ontrak Inc., 2120 Colorado Ave., Suite 230,
Santa Monica, CA 90404, USA
e-mail: amohammad@ontrak-inc.com

Peter Bearse
Chief Data Scientist
Ontrak Inc., 2120 Colorado Ave., Suite 230,
Santa Monica, CA 90404, USA
e-mail: pbearse@ontrak-inc.com

Intisar Rizwan I Haque (Contact Author)
Data Science Intern
Ontrak Inc., 2120 Colorado Ave., Suite 230,
Santa Monica, CA 90404, USA
https://orcid.org/0000-0003-3702-8923

Abstract— **This paper presents Healthcare Big Data Normalization using Computerized Provider Order Entry (CPOE) and application of Graph Theory. This is the process of entering physician orders directly into an electronic health record (EHR). CPOE replaces traditional pen and paper, email, fax, and telephone ordering methods. CPOE is an integral part of electronic medical records and a mandatory component for achieving Meaningful Use Stage 2 certification in health care. CPOE is vital because it helps reduce medical errors that can lead to morbidity and mortality and lowers health care costs. Relational databases are the most common type of database used in healthcare settings. The advantages of using a Relational Database Management System for CPOE are discussed, as well as the disadvantages. The Entity-Relationship diagram and schema for a medication CPOE system used in a small ambulatory medical clinic are provided. We also briefly discuss the potential use of a CPOE application and a NoSQL Open Source database, such as OrientDB, along with the benefits and challenges.**

Keywords— Relational Database, Healthcare, CPOE, Meaningful Use, NoSQL

I. Introduction

In 2009, the Patient Protection and Affordable Care Act (ACA) was passed into law. Along with the ACA came federal requirements for the “meaningful use” (MU) of certified health information technology. The long range goals of MU include improving clinical efficiency and patient outcomes, reducing preventable medical errors and reducing health care costs. CPOE is a software application that is embedded or interfaces with an electronic health record system designed to help achieve MU goals [1].

CPOE can be categorized as basic, intermediate or advanced. Basic CPOE is described as a passive system that pulls information from the database, provides links to medical reference guides for the health care provider to search through manually and can only print out prescriptions. Intermediate and Advanced CPOE are active systems that provide the practitioner with contextual information that serves as clinical decision support and provides convenience for patients, as they can push information and prescriptions to local networks (i.e., pharmacy, health systems). Intermediate CPOE uses standard drug tables and dosages, whereas Advanced CPOE systems suggest treatment options, checking for drug-drug and drug-allergy indications, and allowing dosages to be calculated based on weight. Advanced CPOE is ideal, as it has the added capability to directly link to regional Health Information Exchanges (HIE), which are networks of health information networks serving large geographical regions and key to the creation of a Nationwide Health Information Network (NHIN) [2]. Studies show that the more advanced the CPOE system, the greater percentage of error is avoided. To achieve MU Stage 2, providers must attest to using a minimum of an intermediate CPOE to electronically record and send prescriptions 60% of the time [3].

As previously mentioned, the ultimate goal of MU is the creation of a NHIN, hinting at what Garfinkel describes as a “nation of databases” [4], and which will eventually eliminate paper charting and afford patients greater continuity of care. Because most health data is unstructured, it must be stored as structured data to fulfill MU requirements. The most common form of database used in healthcare is the Relational Database [5]. The Relational Database Management System (RDBMS) is the cornerstone of most health information networks, because it is well suited for handling transaction data and its analysis. Because health care is episodic (one patient with multiple encounters) and each patient can have multiple other one-to-many relationships, the RDBMS must be normalized to third normal form to run efficiently [6]. This paper briefly explores RDBMS structure and describes some of the use cases for CPOE in a RDBMS.

II. BACKGROUND RESEARCH

Many current EHR systems (e.g., Epic, Veterans Health Information Systems and Technology Architecture (VistA), Meditech) use a hierarchical, array-based database called the Massachusetts General Hospital Utility Multi-Programming System, also known as MUMPS or M [7]. In use since the 1960s, this hybrid programming language-data architecture has created a massive amount of legacy applications within healthcare organizations. A hierarchical approach has been favored by many for healthcare applications, because unlike RDBMS, the tree structure allows for multiple redundancies and running simple queries is fast and does not necessitate the use of table joins and costly storage space. However, these types of databases do not support complex, non-traditional queries often used for healthcare data analysis [8]. They also fall short when it comes to data aggregation for reporting needs. In order to meet all the MU reporting and interoperability requirements, MUMPS users must purchase additional

978-1-7281-7624-6/20/$31.00 ©2020 IEEE
DOI 10.1109/CSCI51800.2020.00153

applications and use other platforms (e.g., Cache, Oracle, MS SQL) to perform those tasks. Another drawback to MUMPS is that there are fewer developers available to support and maintain these systems and subsequent mountains of code, and younger developers are not interested in learning the arcane language [9].

In 1970, Codd introduced the relational database model, which proposed a model for storing data in tables. Each table is organized into rows and columns that represent instances of attributes for each unique entity. Thus, a relational database is a collection of tables linked together by defined relationships. Because the rows of each table represent relationships among the set of data, mathematical equations, or relational algebra, can be used for data representation, fundamental set operations, and queries. The rise of RDBMS in popularity in the 1980s led to the development of high-level procedural and non-procedural query languages with which data can be stored, sorted, or manipulated. Specific records and groups of records can be sorted and analyzed with queries. Reports can then be generated from the data analysis. Besides elegance and functionality, RDBMS also gives data consistency, concurrency control, transaction control, and high-level data security, and supports data independence [10].

III. PROPOSED SOLUTION

A. *A Relational Database Management System*

First, we determine that the setting for our relational database would be a small, retail outpatient clinic with a limited scope of practice and patient services. Our product would be a component of the existing EHR and interface with an embedded web-based drug database (e.g., Medi-Span, EnterpriseRX) to provide clinical decision support and alert notifications. Because this is a conceptual model only, we will not discuss application or interface source codes or persistence layers for upstream/downstream applications. As with most commercial relational-databases, our database uses SQL to execute its functions.

Next, we examined the process flow of a clinic visit and produced an activity diagram (Figure 1.). Literature regarding CPOE design and implementation emphasizes the importance for CPOE programmers and developers to have a thorough understanding of the workflow of a clinical area, so that implementation can go smoothly and to minimize disruption [11]. With a desire to have a relatively simple relational database with regular structure, we then developed our database schema (Figure 2.).

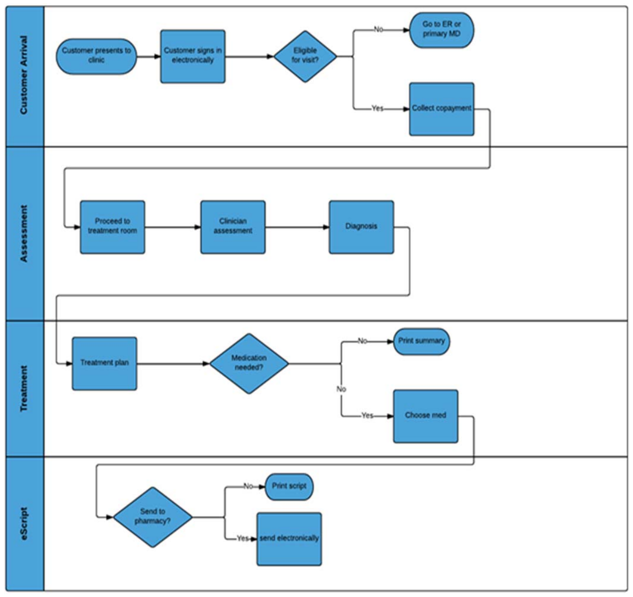

Figure 1. Clinic visit flowchart.

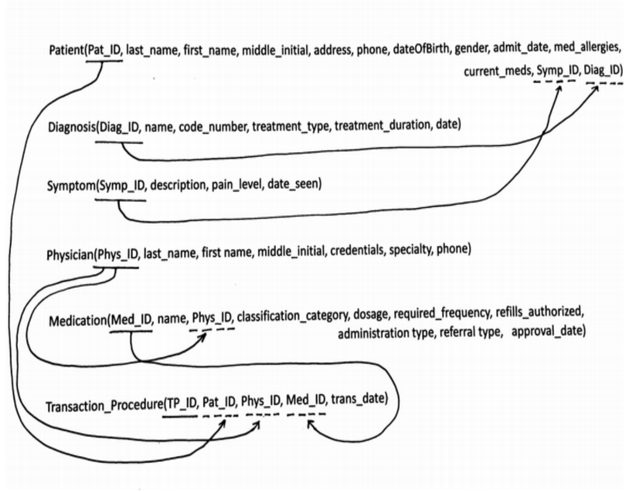
Patient(Pat_ID, last_name, first_name, middle_initial, address, phone, dateOfBirth, gender, admit_date, med_allergies, current_meds, Symp_ID, Diag_ID)
Diagnosis(Diag_ID, name, code_number, treatment_type, treatment_duration, date)
Symptom(Symp_ID, description, pain_level, date_seen)
Physician(Phys_ID, last_name, first name, middle_initial, credentials, specialty, phone)
Medication(Med_ID, name, Phys_ID, classification_category, dosage, required_frequency, refills_authorized, administration type, referral type, approval_date)
Transaction_Procedure(TP_ID, Pat_ID, Phys_ID, Med_ID, trans_date)

Figure 2. Database schema.

Next is our entity-relationship diagram (Figure 3.). Constraints and access permissions will not be discussed in detail. However, it should be assumed that access to write a medication order would only be allowed to a legal prescriber (e.g., physician, nurse practitioner, physician's assistant). Access to read and acknowledge orders only would be limited to appropriate users (e.g., medical assistants, nurses).

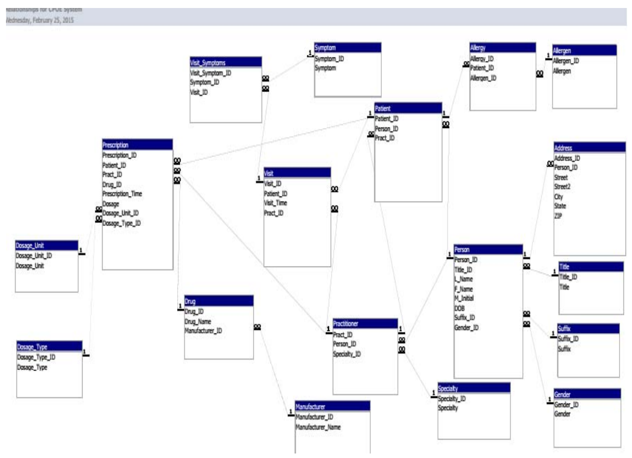

Figure 3. Entity-relationship diagram.

Based on recommendations from the literature we normalized the relations to third normal form to prevent update and consistency issues. The following are examples of Use Case scenarios of our RDBMS with relational algebraic equations:

Use Case – Physician Order

An existing physician orders a treatment from the database for an existing patient.

Relevant tables:
Primary: Transaction_procedure
Secondary: Login, Patient, Medication

Assumptions:

1) The provider is logged into the system with a unique login ID "user". (Need to add provider login ID to provider table, or better, create a login table, where we have (pk_loginID, fk_Phys_ID, roleID, sessionID, lastLoginTime)
2) The provider is within the record of the active patient "p".
3) The provider will select a treatment that is an existing element of the Drug table.
4) A single order will be created at a time

Execution steps:

1) Physician selects a treatment from the dropdown list:

$$MyMedName = \pi_{name}(Medication)$$

2) Given the medication name, administration type, dose, refills, are selected from the following lists:

$$MyAdminType = \pi_{administration\_type} (\sigma_{mymedID=Med\_ID} (Medication))$$
$$MyDosage = \pi_{dosage} (\sigma_{myMedName=name \cap MyAdminType = administration\_type} (Medication))$$

3) The data is available to populate the Transaction_procedure table:

TP_ID = Unique ID generated by the system
Pat_ID = Active patient

$$Med\_ID = \pi_{Med\_ID} (\sigma_{mymedName=name \cap myMedName=name \cap MyAdminType = administration\_type} (Medication))$$

Phys_ID = Active user account ID

4) Confirm that user role is sufficient to create this order. E.g. (prescribing physician has role type=4, and current user role must = 4)
5) Display any required alerts.
6) Insert record into TP table.

Use Case – Check Allergies

When placing an order, the system checks the new order against existing orders and generates an alert if there is a known allergy risk.

Relevant tables:
Primary: Transaction_procedure
Secondary: Patient

Assumptions:

1) The provider is logged into the system with a unique login ID "user".
2) The provider is within the record of the active patient "p".
3) The provider will select a treatment that is an existing element of the Medication table.
4) The patient's known allergies have been previously documented in the Patient table
5) A single order will be created at a time.

Execution steps:

1) Physician selects a treatment as described in the new order use case
   a) Physician selects a treatment from the dropdown list:

   $$MyMedName = \pi_{name}(Medication)$$

   b) Given the medication name, administration type, dose, refills, etc. are selected from the following lists:

   $$MyAdminType = \pi_{administration\_type} (\sigma_{mymedID=Med\_ID} (Medication))$$
   $$MyDosage = \pi_{dosage} (\sigma_{myMedName=name \cap MyAdminType = administration\_type} (Medication))$$

   c) The data is available to populate the Transaction_procedure table:

   TP_ID = Unique ID generated by the system
   Pat_ID = Active patient

   $$Med\_ID = \pi_{Med\_ID} (\sigma_{mymedName=name \cap myMedName=name \cap MyAdminType = administration\_type} (Medication))$$

   d) The data from c is inserted into a temporary table called Transaction_procedure_current.
2) Retrieve all med_allergies records for the current patient:

$$MyAllergies = \pi_{med\_allergies} (\sigma_{Pat\_ID=p} (Patient))$$

3) Retrieve all matches between med_allergies and Med_ID

$$MyAllergyAlerts = \pi_{Pat\_ID,\ Med\_ID} (MyAllergies \ X_{MyAllergies.med\_allergies = Transaction\_procedure\_current.Med\_ID}\ Transaction\_procedure\_current)$$

4) Display/return alerts for any allergies associated with a drug in the current order. User may click through text to override and populate transactions.

Use Case – Check Drug-Drug Interactions

When placing an order, the system checks the new order against existing orders and generates an alert if there is a known drug interaction risk.

Relevant tables:
Primary: Transaction_procedure
Secondary: Medication, Med_Interactions (This is a new table. Med_Interactions (pk_Interaction_ID, fk_Drug_ID1, fk_Drug_ID2, Risk_severity, Risk_text))

Assumptions:

1) The provider is logged into the system with a unique login ID "user". (Need to add provider login ID to provider table, or better, create a login table, where we have (pk_loginID, fk_Phys_ID, roleID, sessionID, lastLoginTime).

2) The provider is within the record of the active patient “p”.
3) The provider will select a treatment that is an existing element of the Drug table.
4) A single order will be created at a time.
5) Any existing drug interactions involving the treatment are recorded in the Med_Interactions table.
6) Previous drug pairings were tested against the interaction table when inserted.
7) Interactions are limited to “pairs” only, not sets of size > 2

Execution steps:

1) Physician selects a treatment as described in the new order use case
2) Retrieve all Med_ID of current order:

$$\text{MyMed\_ID} = \pi_{\text{Med\_ID}} (\sigma_{\text{mymedName=name} \cap \text{myMedName=name} \cap \text{MyAdminType = administration\_type}} (\text{Medication}))$$

3) Create subtable of all patient orders in txn database (We also need a way to discern which orders are “active”. The txn table seems like the best place to do this, though it will require stored procedures to update the active status daily.)

$$\text{MyMedTable} = \pi_{\text{MyMedID, MED\_ID}} (\sigma_{\text{Pat\_ID = P\_ID}} (\text{Transaction\_Procedure})$$

4) Create a union of this table with its inverse. $\pi_{\text{MyMedID, Med\_ID}}$ (MyMedTable) U $\pi_{\text{Med\_ID, MyMedID}}$ (MyMedTable)
5) Add a column to this table indicating the alert type to be returned from the interaction table.

$$\pi_{\text{risk\_severity,risk\_text}} (\sigma_{\text{fk\_Med\_ID1 = MedID} \cap \text{fk\_Med\_ID2 = MyMedID}} (\text{Med\_Interaction})$$

6) Display/return alerts for each non-null drug interaction with severity and text. User may click through text to override and populate transactions.

Use Case – Check if Drug is Appropriate

When placing an order, the system checks the ordered medication against the patient’s symptoms to determine if the medication is appropriate.

Relevant tables:
Primary: Transaction_procedure
Secondary: Patient, Symptom, Medication, Med_Appropriate (This is a new table. Med_Appropriate (pk_Appropriate_ID, fk_Med_ID, fk_Symp_ID))

Assumptions:

1) The provider is logged into the system with a unique login ID “user”.
2) The provider is within the record of the active patient “p”.
3) The provider will select a treatment that is an existing element of the Medication table.
4) The patient’s known symptoms have been previously documented in the Symptom table.
5) A single order will be created at a time.

Execution steps:

1) Physician selects a treatment as described in the new order use case

a) Physician selects a treatment from the dropdown list:

$$\text{MyMedName} = \pi_{\text{name}}(\text{Medication})$$

b) Given the medication name, administration type, dose, refills, etc. are selected from the following lists:

$$\text{MyAdminType} = \pi_{\text{administration\_type}} (\sigma_{\text{mymedID=Med\_ID}} (\text{Medication}))$$

$$\text{MyDosage} = \pi_{\text{dosage}} (\sigma_{\text{myMedName=name} \cap \text{MyAdminType = administration\_type}} (\text{Medication}))$$

c) The data is available to populate the Transaction_procedure table:

TP_ID = Unique ID generated by the system
Pat_ID = Active patient

$$\text{Med\_ID} = \pi_{\text{Med\_ID}} (\sigma_{\text{mymedName=name} \cap \text{myMedName=name} \cap \text{MyAdminType = administration\_type}} (\text{Medication}))$$

d) The data from c is inserted into a temporary table called Transaction_procedure_current.

2) Retrieve all Symp_ID records for the current patient:

$$\text{MySymptoms} = \pi_{\text{Symptom.Symp\_ID}} (\sigma_{\text{Patient.Pat\_ID=p}} (\text{Patient X}_{\text{Patient.Symp\_ID = Symptom.Symp\_ID}} \text{ Symptom}))$$

3) Check the currently selected treatment for a match in the Med_Appropriate table.

Loop through all of the patient’s symptoms. For each symptom: Project the Medication and Symptom

$$\text{MySymptomCheck} = \pi_{\text{Pat\_ID, Med\_ID, Symp\_ID}} (\text{Transaction\_procedure\_current X MySymptoms})$$

Check the Medication and Symptom for appropriateness (MySymptomCheck Left join Med_Appropriate looking for records in MySymptomCheck but not in Med_Appropriate).

$$\text{MyMedInappropriateAlert} = \pi_{\text{Pat\_ID, Med\_ID}} (\sigma_{\text{Med\_Appropriate.Symp\_ID = null}} (\text{MySymptomCheck} ⟕_{\text{MySymptomCheck.Med\_ID=Med\_Appropriate.Med\_ID} \cap \text{MySymptomCheck.Symp\_ID = Med\_Appropriate.Symp\_ID}} \text{Med\_Appropriate})$$

4) Display/return alert if the current medication is not associated with any symptom in the patient’s record. If any symptom is found that the ordered medication is appropriate for, there should not be an alert. User may click through text to override any alert thrown and populate transactions.

B. Applying NoSQL to CPOE

The transition from physical, paper-based medical records to electronic health records has been a recent phenomenon that has grown exponentially since the passing of Affordable Care Act and the advent of MU. There is understandably some reluctance to jump into the NoSQL arena, due to the regulatory and business needs of the medical community. CPOE systems rely on interfaces with clinical decision support systems (CDSS) that supply logic, rules, and information about medications and interactions. For that reason, we are proposing to add a NoSQL database for analytics support on top of the standard RDBMS. Given the complex needs of a medical practice, and the immense

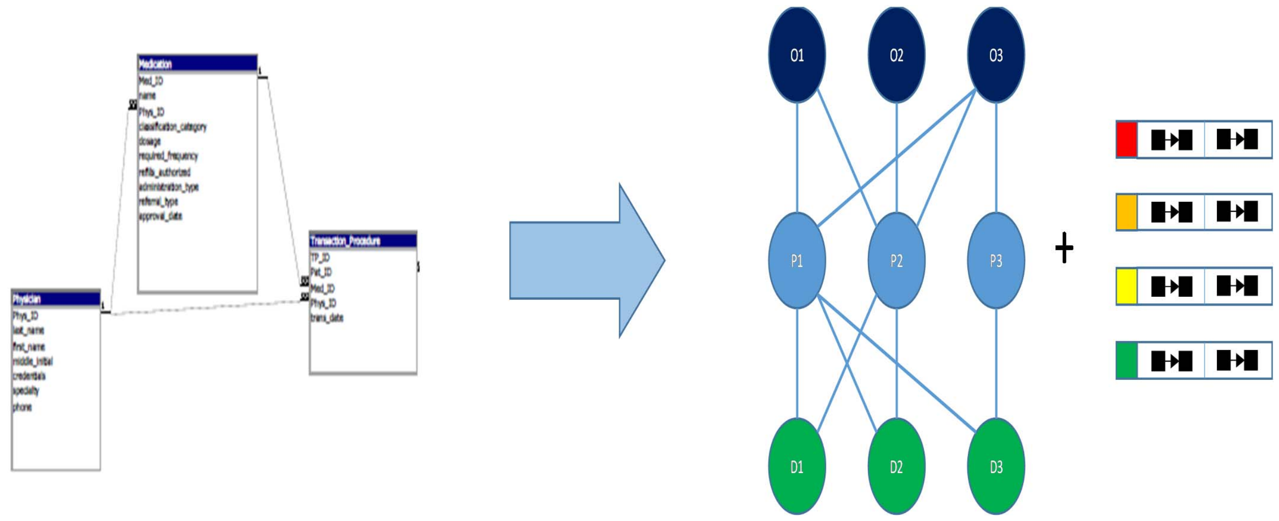

Figure 4. RDBMS – OrientDB diagram.

amount of data that can be generated for an individual patient, there are potential gains even from a smaller number of patients. OrientDB has been selected as the platform for the following reasons:

1) Benefit from all of the standard NoSQL advantages- speed, horizontal scalability, and schema development on the fly, and the ability to handle un- and multi-structured data.
2) Graph database functionality that suits medical records and prescription data as well.
3) Key/Value Document database functionality for handling written elements of medical records, such as patient histories, discharge notes, supplemental status notes, and imaging interpretations.
4) Atomicity, Consistency, Isolation and Durability (ACID) compliance.

The scope of gains listed above are well known and will not be described here. Rather, the key arguments relates to OrientDB platform's hybrid graph/document store database structure (Figure 4.).

Graphical functionality in OrientDB is implemented through the extension of Vertex(V) and Edge(E) classes, the main components of the graph. For example:

Create class Patient Extends V

Create class Physician Extends V

create vertex Patient content { "fname" : "John", "surname" : "Doe" }

create vertex Physician content { "fname" : "Douglas", "surname" : "Howser", "specialty" : "trauma surgery" }

The ability to create column families within each vertex should be familiar to any NoSQL user. Patient and Physician are subclass extensions of the generic Vertex class V. The next step is to create an edge linking the Patient and Physician.

create edge Treated from (select from Physician) to (select from Patient)

The fact that the patient sees this particular doctor is stored in the edge, and the edge itself can have attributes, such as visit dates, documentation generated, test results etc. Similar steps can be taken to instantiate the transactions in the patient's history, such as placing prescription order. The patient's entire history can be maintained in this graphical fashion.

Graphs can be constructed to facilitate the analysis of any particular attributes of interest. We can look at patient flows through the system and identify opportunities for optimization of shortest paths. Characterizing the graph with measures like betweenness and clustering coefficients can identify important factors like nodes of high risk in the network. Environmental factors from patient histories can be reviewed to assist with diagnoses, or clusters of infected patients can be connected to assist epidemiology studies.

The other strength of OrientDB is document support. For example, discharge notes can be stored as an attribute of an office or hospital visit, and those notes can then be mined and analyzed for content. A key element of the hybrid nature of OrientDB for this use is that edges can be created between patients, for example, when their histories share sufficient cosine similarity [12].

IV. CONCLUSION

Future Work and Challenges

RDBMS remains the predominant database in the commercial and healthcare sectors for myriad reasons previously mentioned. In the United States today, there are more than 700 EHR vendors that have created greater than 1,700 certified products, and only a handful of those companies control most of the market share. Although the federal government requires vendors to provide a method to utilize the EHR for data collection, analysis, and reporting [13], some of the oldest, well-known systems use older technology (MUMPS) and do not offer simple interoperability with other vendors or products, which is creating what amounts to a "monoculture" in EHRs [13]. In the near future, interest in open source NoSQL systems for healthcare applications may ignite, as the Federal Trade Commission and Office of the National Coordinator begin investigating and penalizing legacy vendors that engage in technology blocking and price-gouging for application interface software solutions. There is also much discussion about clarification, and perhaps relaxation, of HIPAA regulations regarding cloud storage of personal health information [14].

To be sure, there are many EHR systems that use enterprise cloud storage. Yet, there remains an undercurrent of fear within organizations related to system outages, loss of revenue, data breaches, and loss of reputation. Security breaches and outages caused by hackers, lax security, insider snooping, and mobile devices have solidified the notion, for the time being, that health organizations should keep RDBMS and limit cloud storage. But, because of the massive amount of unstructured health data that is being generated by EHRs, there is an ever-growing need for cheaper storage, along with more complex databases with deep analytic capabilities such as OrientDB, Cassandra, and Hadoop. RDBMS does not have the scalability or flexibility to provide big data mining, analysis or reporting. NoSQL products can perform all those tasks, but most fall short meeting enterprise security needs and ACID compliance. Though leaders in the industry say that as NoSQL matures, consistency and security will improve and remind us that there was once a time in the near past when RDBMS was an unproven technology [15].

REFERENCES

[1] D. Johnston, E. Pan, J. Walker, D. W. Bates, and B.Middleton (2002) The values of computerized order entry in ambulatory settings; HIMSS.

[2] (2015) HealthIT.gov website. [Online]. Available: http://www.healthit.gov/providers-professionals/meaningful-use-definition-objectives

[3] R. J. Campbell, "Database design: what HIM professionals need to know," AHIMA website. [Online]. Available: http://www.healthit-professionals/meaningful-use-definition-objectives

[4] S. Garfinkel, Database Nation: *The Death of Privacy in the 21st Century*. Sebastopol: O'Reilly & Associates, Inc., 2000.

[5] "Meaningful use criteria and healthcare IT infrastructure", October 10, 2010, Healthcare & Security Solutions blog, Available: https://healthcaresecurity.wordpress.com/category/healthcare-technology/privacy-security-solutions/

[6] L. S. Borok, "The use of relational databases in health information systems", J Health Care Finance, vol. 20 pp. 6-21, April 1995.

[7] "MUMPS"(n.d.),Available: http://en.wikipedia.org/wiki/MUMPS

[8] "Why are hierarchical databases like MUMPS still popular in healthcare?", October 12, 2010, Healthcare & Security Solutions blog. Available: https://healthcaresecurity.wordpress.com/2010/10/12/why-are-hierarchical-databases-like-mumps-still-popular-in-healthcare/

[9] R. Tweed, "Can a phoenix arise from the ashes of MUMPS?", January 22, 2013, The EWD Files blog. Available: https://robtweed.wordpress.com/2013/01/22/can-a-phoenix-rise-from-the-ashes-of-mumps/

[10] A.Silberschatz, H. F. Korth, and S. Sudarshan, *Database System Concepts* (3rd ed.). Boston: McGraw-Hill,1999, ch.4.

[11] K. B. Johnson and F. FitzHenry, "Case report: activity diagrams for integrating electronic prescribing tools into clinical workflow", J Am Med Inform Assoc, vol. 13(4) pp. 391-395, July-August 2006.

[12] "OrientDB manual – version 2.0", (n.d.). Available: http://orientdb.com/docs/last/

[13] (2015)HealthIT.govwebsite.[Online].Available: http://www.healthit.gov/sites/default/files/meaningfulusetablesseries2_110112.pdf

[14] K. DeSalvo, "Report to Congress: Report on health information blocking," (April 2015). Available: http://www.healthit.gov/sites/default/files/reports/info_blocking_040915.pdf

[15] K. D. Mandl and I. S. Kohane, "Escaping the EHR trap – The future of health IT", N Engl J Med, vol 2012(366), pp 2240-2242, June 14, 2012.