Comparison of Database Management Systems in the Context of the Healthcare Industry

Anchit Sharma
Department of Computer Science and Engineering
University of Minnesota
Minneapolis, MN, United States
sharm598@umn.edu

Aviral Bhatnagar
Department of Computer Science and Engineering
University of Minnesota
Minneapolis, MN, United States
bhatn042@umn.edu

Joseph Lach
Department of Computer Science and Engineering
University of Minnesota
Minneapolis, MN, United States
lach0101@umn.edu

Manish Rai
Department of Computer Science and Engineering
University of Minnesota
Minneapolis, MN, United States
rai00007@umn.edu

Shreya Datar
Department of Computer Science and Engineering
University of Minnesota
Minneapolis, MN, United States
datar010@umn.edu

Shriya Rai
Department of Computer Science and Engineering
University of Minnesota
Minneapolis, MN, United States
rai00016@umn.edu

***Abstract*—This paper presents a high-level overview of Database Management Systems (DBMS) within the context of the healthcare industry. Areas of emphasis are electronic health records, billing processes, inventory maintenance, and digital imagery considerations for healthcare facilities and organizations. Each of these areas will explore the data requirements, the challenges of managing that data, and a review of selected DBMS solutions available in the open source community or from commercial vendors.**

I. Introduction

Healthcare is a vast and complex system from the perspective of data. There are a wide variety of data that must be captured, processed, queried, stored, and archived. Given that much of the data relates to the most intimate details of an individual, namely their medical and financial information, special considerations and government regulations must be included in the planning, development and eventual implementation of a database solution. Furthermore, the ability to handle data effectively and efficiently can have a profound impact on the budgets of private, non-profit, or government entities involved in healthcare; it is incumbent upon the Information Technology (I.T.) manager to weigh these and other factors when deciding which DBMS to use and for what purpose.

II. Electronic Health records

A. The data

A large portion of a healthcare organization's data is patient information. An electronic health record is essentially a longitudinal record of a person's health information. It includes self-reported demographics, medical history, immunization dates, allergies, clinical notes, diagnoses, laboratory and test results including radiology reports, medications and treatment plans, and billing and insurance information. Until recently, most of this information was stored on paper. However, as expected, this led to a lot of inconsistencies in storing the data, data loss and redundancies. Hence, a shift to databases was made.

B. Considerations

Owing to the sensitivity of healthcare information, it is important to understand patient rights and provider responsibilities. The Health Insurance Portability and Accountability Act (HIPAA) sets forth regulations in this regard.

- HIPAA Privacy Rule- The HIPAA Privacy Rule establishes national standards to protect individuals' medical records and other personal health information and applies to health plans, health care clearinghouses, and those health care providers that conduct certain health care transactions electronically. The Rule requires appropriate safeguards to protect the privacy of personal health information, and sets limits and conditions on the uses and disclosures that may be made of such information without patient authorization. The Rule also gives patients rights over their health information, including the right to examine and obtain a copy of their health records, and to request corrections. [5]

- HIPAA Security Rule- The HIPAA Security Rule establishes national standards to protect individuals' electronic personal health information that is created, received, used, or maintained by a covered entity. The Security Rule requires appropriate administrative, physical and technical safeguards to ensure the confidentiality, integrity, and security of electronic protected health information. [5]

Hence, it is important to implement data usage and access controls, log and monitor use, encrypt data and back up the data to an offsite, secure location.

C. Comparison of Database Systems

When the shift to database systems was made, a natural choice was the OLTP architecture because of its ease of use. However as the amount of data started increasing, certain drawbacks became evident. The following figure shows a comparison of various database systems and their functionality in a healthcare scenario.

TABLE I. COMPARISON OF DATABASE SYSTEMS FOR EHR'S

OLTP	• backup and external storage • quicker processing of transactions such as lab results, payment claims, etc. • *overwhelming amount of raw data* • *data silos*
OLAP	• sophisticated analysis on data from a variety of sources: EHR, billing, population ourcomes, patient satisfaction, etc.
Health Catalyst Data Operating System	• combines features of data warehousing, clinical data repositories and health information exchanges in a single platform
NoSQL	• higher accessibility due to distributed nature and replication • flexible models allow storage of unstructured or semistructured data (free-text notes, images, etc.) • horizontal scalability

Considering the amount of data, the different types of data and the need for multiple location access, NoSQL may provide more functionality for an EHR application.

III. BILLING

A. The data

Billing is the process of submitting and processing of claims to insurance companies in order to get reimbursement for services rendered by a healthcare provider. [4] The data constitutes facts about health insurance such as eligibility and membership, dual coverage when relevant, copayments and deductibles for a given benefit package.

The data for example would have attributes similar to the ones listed below:[2]

1. Patient: id, name, birth date, address , phone no ,gender
2. Insurance: insurance_id, insurance name, address, phone no
3.Doctor: doc_name, doc_id, NPI, address , phone no
4.Diagnosis: dx_code, diag_desc
5.Procedure: px_code, proc_desc
6.Bill: bill id , bill date, due date, bill amount, bill type
7.Payment: transaction id, date received, pmnt type, bill_id, payment amount
8.Encounter: visit date, pt id

B. Considerations

Billing requires validation and execution of business rules, constraints and triggers. For instance, a patient's insurance coverage should only kick in after they have paid the deductible amount. Moreover the billing software needs to backup the data religiously and have tenacious crash recovery mechanisms in place. However, there's relaxation in terms of the type of queries. There's a standardized set which has to be executed fast such as updating a patient's pending amount. This can be solved using OLTP engines.
The database tables should be highly normalized (at least 3NF).

C. Comparison of Database Systems

Hekaton is a new database engine optimized for memory resident data and OLTP workloads. Hekaton is fully integrated into SQL Server; it is not a separate system.

To take advantage of Hekaton, a user simply declares a table memory optimized. Hekaton tables are fully transactional and durable and accessed using Transact-SQL (T-SQL) in the same way as regular SQL Server tables. A query can reference both Hekaton tables and regular tables, and a transaction can update data in both types of tables. T-SQL stored procedures that reference only Hekaton tables can be compiled into machine code for further performance improvements. The engine is designed for high concurrency. To achieve this it uses only latch-free (lock-free) data structures and a new optimistic, multiversion concurrency control technique.[1]

Apache Spark is an in-memory computation engine built on top of the Hadoop Distributed File System. It speeds up transaction execution by delaying operations up to the point where they impact an I/O operation. It processes records in a micro-batching fashion.

a) Storage and Indexing: Hekaton is used to store frequently accessed tables in the main memory while others remain on disk, same as that of SQL Server. A Hekaton table can have several indexes and two index types are available: hash indexes and range indexes. Hekton indices are designed and optimized for memory resident data. The engine uses latch-free data structures to avoid physical interference among threads and a new optimistic, multiversion concurrency control technique to avoid interference among transactions.[1]

Spark distributes each mico-batch of data records across the main memories of worker nodes. This is referred to as RDD.[3] It writes data to HDFS post computation on RDDs.

b) Read, Write and Update: Every read operation in Hekaton specifies a logical (as-of) read time and only versions whose valid time overlaps the read time are visible to the read; all other versions are ignored. Different versions of a record always have non-overlapping valid times so at most one version of a record is visible to a read. [1]

Update in Hekton inserts a new version of a record with begin time as begin time for the transaction that updated it and changes the end time of the previous active version to begin time of the same transaction.

Delete in Hekaton changes the end time of the previous active version to begin time of the same transaction.

Spark builds a Directed acyclic graph (DAG) of operations and delays their execution until an action is called. [3] An action is an operation that performs an external API call or writes data to HDFS. The delayed execution of backlog transformations on the moving data speeds up the reads, writes and updates specified by the lambda functions. For instance, a sequence of operations such as modifying a person's address, updating the pending amount would actually be executed only when the next DAG step is to update the HDFS file.

c) Query Optimization: Hekaton maximizes run time performance by converting statements and stored procedures written in T-SQL into customized, highly efficient machine code. The generated code contains exactly what is needed to execute the request, nothing more. As many decisions as possible are made at compile time to reduce runtime overhead. [1]

Spark is coupled with a rule engine instead of writing lambda functions. This is preferred because in order to make any changes to the business logic (say, reducing dual coverage limit), we just have to edit a file.

d) Crash Recovery: The data in Hekaton is stored on external storage consisting of transaction log streams and checkpoint streams. Log streams contain the effects of committed transactions logged as insertion and deletion of row versions. Checkpoint streams come in two forms: a) data streams which contain all inserted versions during a timestamp interval, and b) delta streams, each of which is associated with a particular data stream and contains a dense list of integers identifying deleted versions for its corresponding data stream. The combined contents of the log and checkpoint streams are sufficient to recover the in-memory state of Hekaton tables to a transactionally consistent point in time.[1]

Upon system crash, Spark re-executes the operations performed during the last I/O operation to restore the main memory data to a consistent state. However, this can be further improved by introducing checkpointing of RDDs[3] post computationally intensive transformations.

IV. Inventory Management

A. The Data

Inventory and supply chain management in the healthcare industry lends itself quite well to a relational model for a database system. Many of the entities, attributes and relations in inventory management can be mapped to the common concept of one or more Products tables. An example implementation may be described in an ISA relationship for distinction between durable medical equipment, medical devices, and consumables, each with distinct descriptions.[7]

In a clinical setting with storage limited to a few supply closets, capturing location information is likely of little importance, but this changes when considering a hospital complex spread across multiple floors and departments, or even multiple campuses. Logging quantities per location becomes crucial to keeping a well-stock supplies cabinet in dozens or even hundreds of places, and can lead to optimizations for reallocating supplies, particularly if items have expiration dates.[7]

Finally, an inventory database could be used to store electronic versions of manuals, calibrations and servicing intervals to maintain compliance data.[7]

B. Considerations

Because inventory and supplies for a clinic or hospital can represent a significant portion of the budget with a high degree of loss, security must weigh heavily as a top factor to consider for a DBMS for inventory and supply management. If expensive equipment or controlled pharmaceutical substances (such as narcotics) are captured in the database, security should extend beyond typical database-based access control methods. An example from the Federal Information Processing Standard (FIPS) Publication 140-2 could include smart-card type token access for user authentication for meeting the lowest Level 1 specification.[8]

Another top consideration for selecting a DBMS is cost. Open source options are available as an alternative to a commercial product. On the one hand, choosing the open source route will be low- to no-cost for the basic software platform. On the other, it will entail hiring a team of developers and engineers to design and then maintain the database for its lifespan. With these points in mind, it is easy to see why the contractor solution is frequently preferred over developing in-house.

Finally, additional considerations evaluated include ACID-compliance (Atomicity, Consistency, Isolation, and Durability), crash recovery, and XML support as a means to extend the database beyond a rigid relational model while maintaining a schema structure. For example, it may be useful to include additional extended attributes about certain devices or equipment that, if included as defined columns on a table, could result in wasted space on disk for many items that do not need those extended attributes. Because these are generally well-understood criteria, they were selected for

inclusion as a means of describing interesting contrasts between the evaluated databases.

C. *Comparison of Database Systems*

Four DBMS were selected for evaluation:

a. Oracle chosen to represent a top-tier commercial DBMS for inventory management.
b. MongoDB is a popular document NoSQL-style database.
c. MySQL is a popular open-source relational DBMS.
d. PostgreSQL is another open-source relational DBMS.

The methodology chosen for this comparison was to set each consideration at three points and deduct based on a review of the documentation. The five considerations were ranked in order one to five, and a final score derived from the following calculation:

$$\text{Final Score} = \text{SUM}[\ (6 - \text{RankVal}) * \text{Score}\ ] / 5$$

Oracle

As a commercial vendor, Oracle offers a variety of methods to secure the database to meet the client's needs, including advanced services such as smart card access, which requires both a cryptographic token for identity and a personal identification number (PIN) to decrypt it for identity verification.[9]

Oracle is also expensive. While pricing will vary heavily depending on implemented services and non-profit or governmental status, the healthcare facility or organization may determine the logistical improvements more than offset the cost, particularly when factoring development costs for a software team to support an open source project.[10]

This DBMS also earned top marks for the three remaining categories; a review of the documentation did not merit any further deductions.

Criteria	Ranking	Oracle
Security	1	3
Cost	2	1
ACID compliance	3	3
Crash recovery	4	3
XML support	5	3
Total		7.4

MongoDB

MongoDB focuses mainly on the availability of service to be more suitable for websites where the usage is more extensive. It does not perform as well as its peers in key security fields like confidentiality, access control, data integrity etc., thus being awarded two points.[11]

MongoDB is an open-source software, with cloud storage that comes at a price.

As a document database, MongoDB has always supported ACID compliance at the document (row) level. The recent 4.0 release in June, 2018, extends this to the multi-document transaction level.[12]

Because MongoDB was evaluated as an open source DBMS, crash recovery will necessarily be handled by the IT team's Database Administrator (DBA), either on site or in a cloud environment. Given this, there will be added complexity and overhead relative to a commercial solution, and liability shifts from the contractor to the healthcare facility or organization.

There is no need to evaluate XML support in a document database. For the purpose of the final score, three points were awarded in this category.

Criteria	Ranking	MongoDB
Security	1	2
Cost	2	2
ACID compliance	3	3
Crash recovery	4	2
XML support	5	N/A
Total		6.8

MySQL

Data integrity and confidentiality are well-preserved through FIPS support, using OpenSSL Library. Support for transactions and rollbacks makes it an appropriate fit for inventory management.[11]

MySQL is open source, released under the GNU General Purpose License.[13]

It is possible to configure MySQL with a non-ACID compliant database engine, which could result in inaccurate records. For many years, the MyISAM storage engine was the default, so existing implementations may still be configured this way for legacy systems. The current default InnoDB engine supports ACID transactions, but the risk posed by this flexibility merits a deduction.[14][15]

The same evaluation for crash recovery for MongoDB also applies to MySQL, with the identical rating.

MySQL offers limited support for XML. The reference documentation includes a note that the XML functions described in the manual remain under development. Without

a thorough examination of what and how XML might be used with MySQL for inventory management in a healthcare facility or organization, it suffices for the purpose of this comparison to evaluate this consideration the lowest score.[16]

Criteria	Ranking	MySQL
Security	1	2.5
Cost	2	3
ACID compliance	3	2
Crash recovery	4	2
XML support	5	1
Total		7.1

PostgreSQL

PostgreSQL has all the essential security features from MySQL along with the support for Materialized Views, that play a key role in caching the data pertaining to a specific role remotely, for quicker and more efficient access. Due to these reasons, no deductions were warranted.

PostgreSQL is open source, released under the PostgreSQL Licence.[17]

PostgreSQL is an ACID-compliant open source DBMS upon installation, as such no points were deducted.[18]

As with the two other open source databases, one point was deducted from crash recovery due to the need to internally develop and maintain a crash recovery plan.

PostgreSQL supports XML data types and a variety of XML functions in currently supported versions of the DBMS. No points were deducted.[19][20]

Criteria	Ranking	PostgreSQL
Security	1	3
Cost	2	3
ACID compliance	3	3
Crash recovery	4	2
XML support	5	3
Total		8.6

V. Healthcare Images

A. *The Data*

Medical images have become critical to healthcare industry. New healthcare diagnosis methods have come up that use advanced medical imaging devices to generate more and more images. Some of these medical imaging devices include X-Ray, CT Scan, Ultrasound, Magnetic Resonance, etc. These devices generate extremely large images that represent the interior of the body and are very crucial for any diagnostic process and clinical analysis. The medical images also have interpretation reports. Based on an estimate, 30% of the world's data will be medical images by 2020[24] and that will call for efficient storage, retrieval and transfer systems for these images.

Figure 1. Magnetic Resonance Image

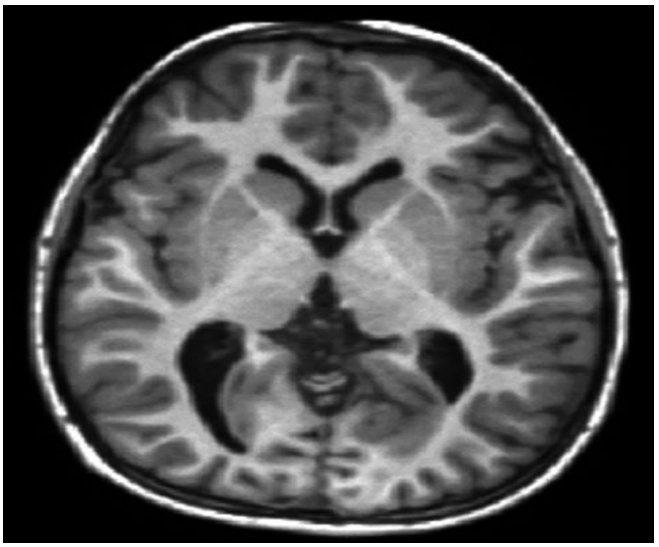

Image Source - Wikipedia

Medical images and their accompanying interpretation reports are stored in a Picture Archiving and Communication System (PACS) that provides the secure storage and convenient access to images generated from multiple modalities of healthcare devices. PACS has four major layers that ensure secure distribution of healthcare images, easy viewing, processing and interpretation, and electronic archiving for storage and retrieval.

Figure 2:PACS Architecture

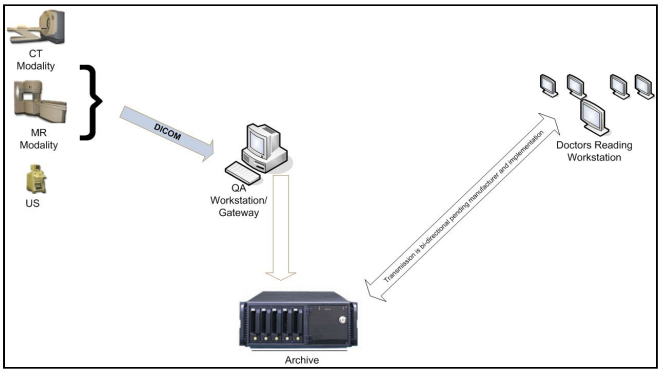

Image Source - Wikipedia

PACS uses the Digital Imaging and Communications in Medicine (DICOM) standard as the de-facto protocol for image storage and transfer. DICOM is the default standard for image data management in healthcare and is supported by major medical imaging equipment manufacturers. The DICOM standard was initially created by the Americal College of Radiology(ACR) and the National Electrical Manufacturers Association(NEMA) in 1993 in order to facilitate a standardized format for creation and exchange of images generated by various medical imaging devices.

B. Considerations

To start with, images generated by the medical imaging devices were stored in separate databases and only the directory of the images were stored in relational databases. Since then, there has been an evolution in the needs of people and the technology available, so new storage techniques have been developed. New data objects called Binary Large Objects (BLOB) is one such development that facilitated the storage of images within the same relational database. BLOB stores a collection of binary data as a single object, and it has made it easier to store the images encoded as a binary. [22] However, there are several challenges that come with storing the medical images in relational databases:

i. The quality of medical images is increasing leading to demand for more storage per cell of a relation.
ii. Chunking the images into small sub-files for storing and reassembling of the files is another problem that is not easily managed by relational databases. [22]
iii. Medical images are generally semi-structured or unstructured and relational model doesn't work well for them.

These challenges with storing medical images in a relational model has led people to explore the Document (NoSQL) databases such as MongoDB, Cassandra, HBase, etc. NoSQL databases are adequate at handling semi-structured and unstructured data. They also open source and provide a great deal of flexibility and a community of researchers.

This section will explore Oracle Multimedia for DICOM, and a brief survey of research on NoSQL databases.

C. Comparison of Database Systems

Oracle Multimedia provides full support for storing, retrieving, and manipulating DICOM format medical images and other objects in a database. It also provides support for DICOM protocol which enables DICOM applications and devices to easily access DICOM data in the Oracle Database. This enables Oracle Database to store and manage DICOM content as part of a clinical workflow. Oracle Multimedia provides the object type, **ORDDicom**, which natively supports DICOM content produced by medical devices. This object type holds the DICOM content and extracts the metadata, and implements the methods to manipulate the DICOM content. It also provides a multitude of features for image processing, apply adequate security features, and many more. This is a great tool for storing medical images in relations and it has SQL-like queries for selecting, updating, etc. of the metadata.

Figure 3: Oracle Multimedia DICOM architecture

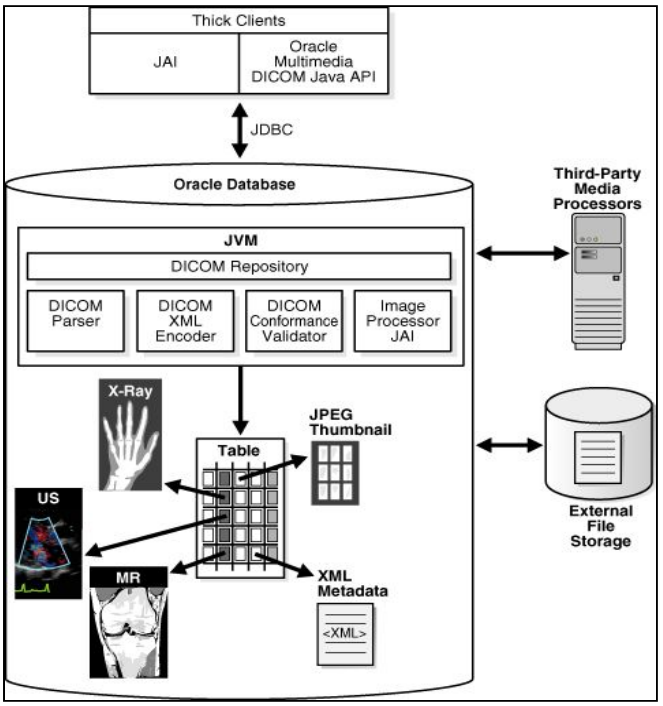

Image Source - Oracle Multimedia DICOM

Several NoSQL databases have been studied by researchers for storing medical images. One study evaluated the performance of MongoDB for storing medical images. [22] Some of the striking features about MongoDB that the original paper cited are:

a. MongoDB manages large files through the GridFS API that automatically chunks a large file into a group of small files. When a query is made, this API reassembles the chunks into original file.
b. MongoDB stores files into Binary JSON, which is schemaless and keeps all related data in one place.

Automatic handling of large files is a very important feature required for medical images, and is provided by MongoDB. This can reduce a lot of programming overhead from the developer.

Another NoSQL database reviewed is Cassandra, which is a distributed storage system developed by Facebook. [24] It was designed to work on cheap hardware and handle high write throughput while not sacrificing the read efficiency. It has a massively scalable architecture which scales linearly. One of the observations made by the original study regarding Cassandra for medical images was that storing large images with a single set of operations is difficult. However, chunking is possible in Cassandra using the utility Astyanax. [24]

The literature surveys and comparison of MongoDB and Cassandra indicated that MongoDB performs better than Cassandra on large size images. [24] This is a very critical metric that makes MongoDB stand out for further consideration.

VI. Conclusion

It is clear in the healthcare industry that there exists a wide variety of data to support: no one database solution can handle all of the disparate needs of a facility or organization. Some aspects will do well with the traditional relational model for structuring data, while others will scale and perform better with a document (NoSQL) database. Additionally, there are financial considerations as to whether the facility or organization should adopt a commercial or open source solution, which are beyond the scope of this research.

Choosing a database management system for a particular function will depend on being cognizant of both what data are to be modeled and the systems available, as well as applicable governing regulations and industry standards. This paper serves as a high-level review of this process, and has provided examples of the data that will be encountered, developed criteria for comparison, and contrasted different DBMS solutions.

Conflict of Interest Statement

Co-author Joseph Lach is privately employed as a software engineer while in graduate studies at the University of Minnesota. His employer uses Microsoft SQL Server. Mr. Lach was involved in the research of the inventory management section, and the SQL Server DBMS was deliberately not chosen for evaluation to avoid a potential conflict of interest.

References

[1] Cristian Diaconu,Craig Freedman, Erik Ismert, Per-Åke Larson, Pravin Mittal, Ryan Stonecipher, Nitin Verma, Mike Zwilling "Hekaton: SQL Server's Memory-Optimized OLTP Engine"

[2] https://www.slideshare.net/NoelleVaughn/medical-billing-database

[3] Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauley, Michael J. Franklin, Scott Shenker, Ion Stoica, "Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster ComputingK. Elissa"

[4] Aftab Ahmed, Umair Abdullah and Mohammad J. Sawar "Software Architecture of a Learning Apprentice System in Medical Billing"

[5] https://www.hhs.gov/hipaa/for-professionals/security/index.html

[6] Michael Lane, Mehmet Ercan, "An evaluation of NoSQL databases for electronic health record systems", December 2014, University of Southern Queensland

[7] "Computerized maintenance management system", June 2011, World Health Organization, http://apps.who.int/iris/bitstream/handle/10665/44567/9789241501415_eng.pdf

[8] Federal Information Processing Standards Publication 140-2, May 25, 2001, https://csrc.nist.gov/publications/detail/fips/140/2/final

[9] Oracle database security guide 18c, October 2018, https://docs.oracle.com/en/database/oracle/oracle-database/18/dbseg/index.html

[10] Oracle pricing, https://cloud.oracle.com/en_US/logistics-cloud/wms-pricing

[11] Omar Al-Ithawi, "A security comparison between MySQL and MongoDB".

[12] James Kobielus, "MongoDB drives NoSQL more deeply into enterprise opportunities", June 27, 2018, https://wikibon.com/mongodb-drives-nosql-deeply-enterprise-opportunities/

[13] MySQL documentation, "Commercial License for OEMs, ISVs and VARs", July, 2010, https://www.mysql.com/about/legal/licensing/oem/

[14] MySQL 8.0 reference manual, "15.2 InnoDB and the ACID model", https://dev.mysql.com/doc/refman/8.0/en/mysql-acid.html

[15] Ronald Bradford, "Q: Does MySQL support ACID? A: Yes", June, 29, 2016, http://ronaldbradford.com/blog/q-does-mysql-support-acid-a-yes-2016-06-29/

[16] MySQL 8.0 reference manual, "12.11 XML Functions", https://dev.mysql.com/doc/refman/8.0/en/xml-functions.html

[17] PostgreSQL documentation, "License", https://www.postgresql.org/about/licence/

[18] PostgreSQL documentation, "About", https://www.postgresql.org/about/

[19] PostgreSQL 11 reference manual, "8.13 XML Type", https://www.postgresql.org/docs/11/datatype-xml.html

[20] PostgreSQL 11 reference manual, "9.14 XML Functions", https://www.postgresql.org/docs/11/functions-xml.html

[21] A new database for medical images and Information - "Dave Tahmoush"

[22] A NoSQL Solution to efficient storage and retrieval of Medical Images - "Revina Rebecca"

[23] Understanding and Using DICOM, the data interchange standard for biomedical images - "Dean Bidgood"

[24] Analysing the suitability of Storing Medical Images in NoSQL Databases - "Revina Rebecca"

[25] https://docs.oracle.com/database/121/IMDCM/toc.htm - Oracle Multimedia for DICOM