Integrated Document-based Electronic Health Records Persistence Framework

Aya Gamal[1]*
Faculty of Computers and Artificial Intelligence
South Valley University, Hurghada, Egypt

Sherif Barakat[2], Amira Rezk[3]
Information System Department, Faculty of Computers and Information, Mansoura University, Egypt

***Abstract*—Electronic health record systems work beyond just recording patients` health data. They have multiple secondary functionalities, such as data reporting and clinical decision support. As each of these systems` workloads has contradictory different needs, managing a multipurpose electronic health record is a challenge. This paper proposes a unified healthcare data framework that can simplify health information system infrastructure. It investigates the suitability of the document-based NoSQL persistence mechanism, storing electronic health records data as a design choice for managing varied complexity ad hoc queries used in operational business intelligence. The performance of the most popular two document-based NoSQL back-ends, Couchbase Server and MongoDB, is compared according to the size of the database and query execution time. Results showed that while MongoDB can execute simple single-document queries nearly in milliseconds. It does not provide satisfactory response time for unplanned complex queries spanning multiple documents. By utilizing its analytics services and multi-dimensional scaling architecture, Couchbase Server multi-node cluster outperforms the response times of MongoDB for both simple and complex healthcare data access patterns. The primary advantage of the proposed tightly coupled EHRs processing framework is its flexibility to manage workload according to changing requirements.**

***Keywords*—Electronic health records; operational business intelligence; document data model; NoSQL; health information system; persistence framework; Couchbase** server

I. Introduction

Electronic health records (EHRs) system is a quintessential part of healthcare information system (HIS). It is an ecosystem for maintaining a long-term patient's health record. This system usually has multiple core functionalities. Primarily, it is used for storing and retrieving individual patient records for healthcare purposes. These EHRs data could be used in clinical decision support (CDS) to suggest the next steps for treatment or predict future conditions trends by analyzing transactional data [1]. EHRs data are not possessed by any particular healthcare provider. To interconnect these different healthcare practitioners, EHRs need to be interoperable through following certain standards to facilitate health information exchange (HIE) and sharing [2]. HIS workload usually encompasses two main practices: clinical use, which is regarded as a transactional workload, and research use which is dedicated to analytical workload.

As each of these workloads has a different access pattern, they have seemingly contradictory solutions. There is also the operational business intelligence (BI) workload, including ad hoc queries, which are in the middle between primary and secondary uses. This data access pattern tends to be unpredictable; it typically involves reading an extensive amount of data at one time and can include various complex joins [3]. There are various architectural methodologies [4,5] to manage those conflicting workloads. Traditional HISs could use transactional systems for providing answers to analytical queries in one engine, but the system performance may degrade dependent on the number and the complexity of queries submitted to the database. Therefore, following the "one size does not fit all" rule [6], a clinical data warehouse (CDW) [7,8] a specialized storage structure, is used for data analysis to segregate tasks and maintain the performance at an acceptable level. It utilizes extract, transform, and load (ETL) to integrate patient-level data from separate silos inside or across healthcare organizations, facilitating analysis and reporting. Thus, various healthcare application frameworks [9] have been introduced to offer diversified EHRs data analytical capabilities.

Earlier healthcare data persistence systems [10] were built depending on the relational schema approach. However, these traditional relational database management systems RDBMS are built for strong consistency level and data control [11]. With the rise of the "no one size fits all" concept [12], several alternative NoSQL stores have been developed [13] to address the shortcomings of RDBMS. NoSQL databases refer to a category of flexible data storage systems that manage data using a key-value structure. They are clustered into four main classes [14,15] based on their data model: (1) key-value data stores, (2) document data stores, (3) column-family stores, and (4) graph data stores. This categorization is necessary since each data storage architecture provides different solutions depending on the application's needs.

Healthcare data is complex, dynamic, intermittent, and diverse in nature. Furthermore, HIS applications` access patterns usually need their scale, performance, and flexibility requirements to surpass their transactional needs [16,17]. Thus, NoSQL data stores are more suitable to meet the specifications of distributed EHRs systems [18]. There are several criteria, such as data model, performance, data persistence, and CAP support[19], which must be considered when choosing which NoSQL store to be used. Various data modeling approaches [3,20–27] have been introduced for medical data persistence according to use case scenarios. These works investigate not only the type of NoSQL store that has to be chosen but which NoSQL products in that type will be used [19].

*Corresponding Author

Managing a fully functional EHRs system with the involvement of all healthcare participants is not a simple task because of the rapid development and growth of medical knowledge. To fulfill a variety of use cases, organizations nowadays commonly end up deploying different databases, leading to a "database sprawl" that causes delayed analytics. This layered architecture approach was challenged by the white paper [28] which argued that advances in memory technology enable data to be stored just once without compromising either transactions or analytical workloads. Massive parallel processing (MPP) database platform is considered as another architectural alternate to CDWs that could support medical data analytics [29]. It could be used for ad hoc population queries, which may not be quickly obtained from the CDW.

The capability to store data quickly is not a problem. But the challenge is the capability to do meaningful and quick insights with that data. Recently, Analytical application characteristics diverged from the typical characteristics of the online analytical processing (OLAP) system and become more real-time, operational, and proactive. Modern applications frameworks require blurring borders between operational and analytical workload [4]. Several terms, such as Hybrid Transactional and Analytical (HTAP) databases [30,31], are being used to describe this general trend in databases that supports hybrid workload processing requirements within a single logical database. Couchbase Server recently introduces multi-dimensional scaling (MDS) architecture [33] to support scaling workload independently according to the changing needs. It aims to offer a single, integrated platform that can be used for almost all varied complexity operational workloads as well as operational analytics. The motivation for this paper is that nevertheless, these hybrid data processing and management techniques [16,32] could provide solutions to a wide range of healthcare data problems, such as data silos. Few researchers evaluate their performance in the healthcare sector.

The main contribution of this paper is it proposes a unified multipurpose HIS framework that could improve healthcare services provision. This flexible Couchbase-based healthcare architectural framework provides different levels of services that could be efficiently adapted according to varied application workload requirements. As both database selection and its related schema architecture are aspects that affect the effective management of healthcare data, particularly in real-time usage systems [10]. This paper first discusses the suitability of the document data model as storage persistence for managing EHRs data. After that Couchbase Server and MongoDB which are the most popular document-oriented database management systems (DODBMSs) are evaluated for BI workload. These two storage back-ends storing EHRs are compared on the subject of their execution time and storage space requirements for handling varied queries complexities.

The rest of this paper is organized as follows: Section 2 reviews document-based data modeling techniques and their suitability for the healthcare domain. Section 3 describes the experimental environment, including datasets, workload specifications, and query implementation. Section 5 reports the experimental results for the two different size datasets. Section 4 briefly discusses the obtained results.

II. Document Data Modeling

A document database is a set of key-value in which each key corresponds to a complex data structure value known as a document. Each document holds a unique automatically generated key which not only enables gathering related documents. But also enables an application to perform keys-based document lookup, which is extremely fast. A document's structure in the document model database is made up of the arrangement of its internal attribute-value pairs [34]. In document-oriented databases, stored values are arranged, in a self-descriptive document that can be examined easily. There are several advantages of using document-oriented modeling, unlike relational databases that force applications to fit data into predefined models, regardless of their needs [35]. First, there is no impedance mismatch between application objects models and documents data models [36]. Second, they do not impose standard document structures, even across several documents. It allows a schema to gradually evolve by adding properties and structures to the document as required without the need to update other documents in the same way. Thus, the schema is explicit but variable, since attributes may vary among instances [37]. This allows applications to change their behavior without having to overhaul all the source data or take applications offline to make a basic change. Therefore, they offer faster write performance than the conventional relational model, besides efficient indexing features. In relational databases, [38] children use foreign keys to refer to their parents. On the other hand, parents refer to their children in document databases. This is because, unlike a field in a row, a field within a document can have several values.

Embedding and referencing are two techniques that could be used to link documents. Document embedding concerns nesting documents into another. It entails merging subclass entities into superclass entities and denormalizing relationships without the need for a reference table. On the other side, document refereeing is based on two separate documents, and one is refereeing into another. It involves adding a key to the object [34,39] and it is good when the referenced objects and relationships are static. Usually, deciding how to model related data documents normalized, denormalized or a hybrid is based on the type of relationship and the access pattern.

In today's market, there are a variety of document-oriented NoSQL databases; Couchbase [40] and MongoDB [41] are the most popular. Couchbase is a scalable in-memory NoSQL database, which was designed particularly for distributed processing and has native support for JSON documents[42,43]. It was created through merging two complementary technologies: Membase Server, a distributed key-value database based on Memcached, and CouchDB, a single-node document database supporting JSON. It has N1QL, a declarative query language that can manipulate JSON documents. A Couchbase node usually comprises of cluster manager and, optionally, data, query, index, analytics, search, and eventing services. Couchbase Server introduces MDS architecture, which is independent scaling architecture, for minimizing interference between services [33]. MongoDB uses a slightly modified version of JSON called BSON (Binary JSON). It has its own DBMS, including the full set of CRUD (create, read, update and delete) operations.

A. *Document-based EHRs Data Modeling*

Because of the special persistence policies of EHRs data. NoSQL systems fit better [20,21], as they allow healthcare data to be stored in a structure that is much closer to its real representation. NoSQL databases are aggregate-oriented data stores [36], as the aggregate is the unit of operation and consistency. They can be more scalable and faster where data volumes are extremely large, or when there are no internal document references that might degrade the performance or data consistency. On the other hand, healthcare applications could be an obvious example of an application type, where the presence of connections between various documents and their subcomponents has no impact on the application's basic functionality and consistency. This is because, if part of these data is updated during such medical treatment, a new extract with new information and their appropriate connections should be created, rather than overwriting any previously stored data elements [18]. This is a rigorous medical information constraint since it may be used to make medical decisions.When using a document-based store instead of a relational system, information about a particular patient is easily isolated from other patients' information.

So, healthcare tasks could be considered document-based tasks. Retrieving all patients' demographic data and linked hospital admissions. These flexible modeling approaches could efficiently adapt to the nature of healthcare data. Thus, several studies have been initiated along this path [3,20,24 ,44–47]. According to related work [20] result, MongoDB is best for handling single patient queries because of their concurrency. While Couchbase in [3,24] achieved better query response times and throughput. Thus, it is the best for handling the analytical workload of scalable, large data size.

III. Proposed Document-based Electronic Health Records Framework

In multi-node cluster topology, the Couchbase Server services are distributed across several nodes within the cluster, rather than a single node cluster. The main aim of this microservices architecture is deploying both query and analytics services, which are complementary services with contradicting workload needs, into two independent nodes within the cluster. Query service is used to support many users in making inexpensive operational queries. While analytics service is used to support complex and expensive analytical queries made by a much smaller number of users [43]. Data and query services provide user-facing applications with low-latency key-value and/or query-based access to their data. For analytics service, operational EHRs data is pushed into shadow buckets of the same data for immediate analysis by a dedicated massively parallel processing (MPP) analytics engine. These shadowed buckets are copies of data that are linked directly to the operational data in real-time. So, these analytical queries do not affect the performance of operational data queries. Cross data-center replication (XDCR) is used to replicate EHRs data per bucket basis between two data nodes in different clusters depending on application requirements. These replicated vBuckets can support read requests as they are kept constantly up to date by receiving a continuous stream of mutations from the active vBucket through database change protocol (DCP). Fig. 1 shows a high-level illustration of the deployed EHRs multi-node topology for varied access patterns.

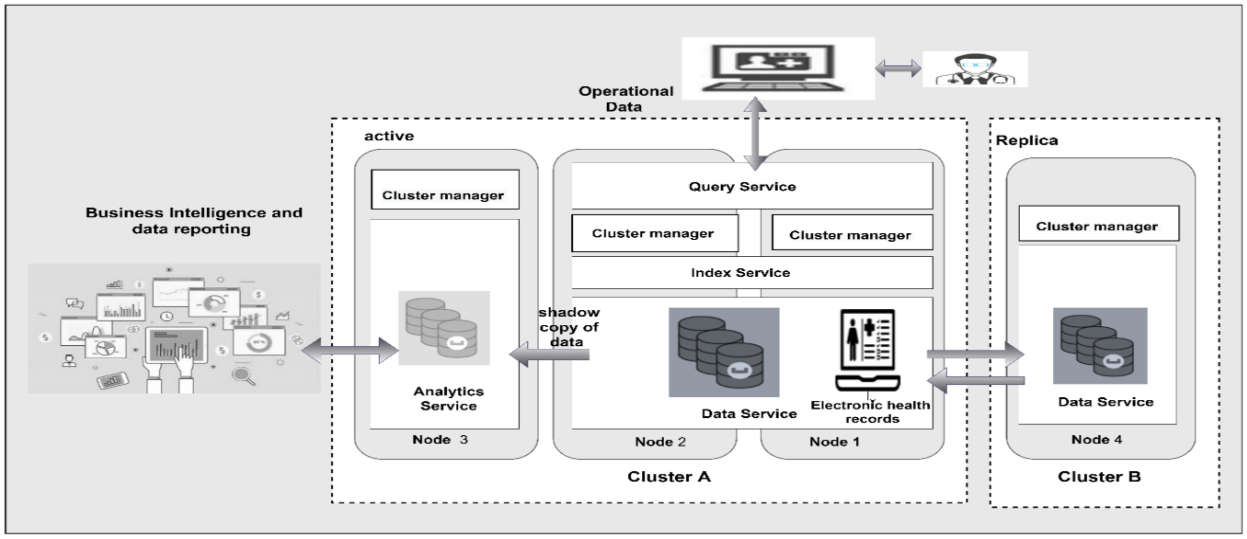

Fig. 1. Proposed Document-based Multifunctional EHRs Persistence Framework.

IV. Experimental Setup

Two NoSQL database management systems: Couchbase 6.6.2 and MongoDB 4.4.3 are used to assess the suitability of document-based persistence to store and retrieve EHRs objects. The two NoSQL products were deployed in a single machine with the following specifications: Intel core i7 -10750h CPU @ 2.60GHz, with 16GB of memory and an SSD storage system of 256GB, running windows 10 _64 bits.

The workloads were tested under the following conditions: data fit the memory, and replication is set to "1" signifying that just a single replica is available for each dataset. All executions are warm runs, so either caching techniques must be disabled, or each query must be performed individually to fill the cache. Queries must be prepared according to the native scripting language of the target database and run directly from the command line interpreter within the chosen system.

A. Dataset

Our experiment investigates the performance of the document-oriented database to store Fast Healthcare Interoperability Resources (FHIR) [48] compliant EHRs data. The used EHRs data is two different size synthetic patient datasets with COVID-19 [49,50]. These synthetic datasets have been generated by SyntheaTM [51], an open-source patient population simulation available from the MITRE Corporation.

SyntheaTM is a synthetic patient generator [52] that simulates the medical history of synthetic patients. Patient data and related health records covering several aspects of healthcare are realistic but not genuine. Every record in this EHRs dataset contains administrative, demographic, and healthcare information concerning a patient. These records are subdivided into different data sources or categories that encompass all the different aspects of healthcare. They also span practitioner, care team, device, organization, location, healthcare service, diagnostic, medications, as well as the financial tasks, such as insurance coverage. Core data elements are presented in Table I.

FHIR specification is a next-generation standard framework developed by HL7 to facilitate faster and more efficient integration, exchange, and retrieval of EHRs data. It bases on the concept of "resources". One of the key objectives of FHIR is that it does not enforce the exchange of whole documents, but it allows the exchange of specific pieces of information. Hence, enabling interoperability across various functional applications and ultimately reducing the cost of the implementation. FHIR data model is relational in nature. It centers around the 'patient' object or resource. Each object has a unique identifier 'id' field. This id is used to reference most resources to the 'patient' object. As a result, the patient resource collection has been normalized into individual resource-type documents in which the objects that are associated with a patient are linked by the patient's id in the subject field.

B. Workload

HIS workload usually involves clinical practice queries which are used to get single patient data and research queries that are dedicated analytical workloads.

TABLE I. Datasets Size and their Attribute Number

Objects	**10k patient**	**100k patient**
Allergies	5,417	51,592
Careplans	37,715	377,726
Conditions	114,544	1,143,900
Devices	2,360	23,694
Encounters	321,528	3,188,675
Imaging_Studies	4,504	45,609
Immunizations	16,481	168,160
Medications	431,262	4,227,723
Observations	1,659,750	16,219,969
Organizations	5,499	9,175
Patients	12,352	124,150
Payer_Transitions	41,392	409,553
Payers	10	10
Procedures	100,427	979,564
Providers	31,764	60,534
Supplies	143,110	1,389,858
Total Item Number	2928115	28419892
MongoDB Size	878.732 MB	8.17 GB
Couchbase Size	1.37GB	14.2GB
CSV Data File Size	504 MB	4.78 GB

Operational BI workload is read-intensive, with writes that do not conflict with reading queries. It resembles the workload of an OLTP application, where quick response times are desired. This workload includes a type of ad hoc query that is used for real-time reporting.

The motivation behind this paper was the belief that ad hoc queries are not planned before, thus having any query indexing. It investigates two back-ends behaviors for ad hoc queries that are examples of real-time analytical queries with different levels of complexity and may have joins and aggregations. The performance of each database was evaluated under queries range in complexity, from simple ad hoc read queries to complex join queries. Simple read workload usually includes filtering constraints and sorting operations and aggregate lookups. While join workload involves join operations with grouping and aggregation functions. These queries declared in Table II and Table III were identified and written according to each database query language. Specific queries whose behavior could differ widely from the others are chosen to differentiate the performance of the databases and avoid bias.

C. Query Implementation

Two database architectural topologies for the two storage back-ends storing EHRs were deployed to investigate for handling different queries complexities. MongoDB experiments were deployed to single unsharded data cluster architecture. While Couchbase Server investigates two database topologies, which are single node cluster and multi-node cluster.

TABLE II. SIMPLE AD HOC READ QUERIES

Q1	Retrieve patients' names and their age
Q2	Rank the encounter count of hospitals
Q3	Find all dead patients' data.
Q4	The number of inpatients.
Q5	Arrange conditions according to their co-occurring
Q6	Arrange encounters according to the number of conducted encounter class
Q7	Count of non-survivor
Q8	The age of dead patients
Q9	The average age of non-survivor
Q10	Max-age of non-survivors
Q11	Mortality by sex
Q12	Mortality by age
Q13	Total count of supplies
Q14	Medications dispensed to patients with COVID-19 since January 20, 2020
Q15	Total cost for every medication dispensed to patients with COVID-19
Q16	Number of patients required ventilator
Q17	All the conditions starting after January 20, 2020
Q18	Cumulative case count of covid over time
Q19	Number of patients with COVID-19 conditions
Q20	The max duration length of any COVID-19 patients
Q21	Care plans for all COVID-19 patient
Q22	Allergies ordered according to co-occurrence
Q23	Maximum observed temperature
Q24	The total number of patients who take influenza immunizations
Q25	The max cost of procedures

TABLE III. COMPLEX JOIN QUERIES

Q1	Retrieve patients' names and their current unresolved conditions
Q2	All patients who were at Beverley hospital on 2011-08-18
Q3	Patients' names with their insurance claim cost
Q4	Merge patients' names with their COVID-19 conditions
Q5	Period time every covid patient connected to a ventilator
Q6	Total cost for every covid patient
Q7	The total cost of medication for every covid patient
Q8	The number and total cost of immunization for every patient
Q9	Period time for every covid patient
Q10	Number of female and male patients with covid
Q11	Number of providers in every organization
Q12	All observations for patients with covid
Q13	All observations for patients who are not diagnosed

For Single node topology in MongoDB, the aggregation pipeline framework is used to implement queries using the concepts of multi-stage data processing pipelines. It uses the $lookup operator to apply left outer JOIN queries over an unshared collection in the same database [53].

For Couchbase Server single node cluster topology, all database services (query, index, and data) are contained in a single zone. In such architecture, the query service is used to handle all different complexity workload types. While for multi-node cluster topology, the Couchbase Server services are distributed across several nodes across several nodes rather than a single node.

V. EXPERIMENTAL RESULTS

A. Evaluation Criteria

The majority of the healthcare system's tasks usually entail searching the databases. An EHRs system's execution time is an essential performance parameter, and storage space efficiency could decrease future maintainability costs. Thus, reducing database query response time significantly improves the EHR system's performance and functionality [54]. This paper evaluates how DODBMSs store and retrieve EHRs in the context of storage space and response time.

B. Database Size

Both 10k and 100k patients' datasets files [55] are initially available in the SCV format with approximately 500 megabyte and 5-gigabyte sizes. Table I shows the two data sets' sizes when stored in each backend. To upload data into the Couchbase server, a command-line utility, $cbimport, is used to import synthase COVID-19 CSV files into JSON Couchbase format. While for uploading data into MongoDB, the mongoimport tool is used. Couchbase and MongoDB demand 2.9 and 1.7 times more space correspondingly than CSV storage space for both datasets of different sizes.

VI. RESPONSE TIME

Database performance is defined by the speed at which a database process workload. In our evaluation, the primary factors related to the performance of the database are the query complexity level and the size of the dataset. To investigate the scaling capability of the two back-ends, the same queries were applied to both 10k patients and 100 k patients' datasets, respectively. All queries` response time and the number of examined documents grouped by the database are represented in the following tables. Queries are mainly divided into two types. Simple ad hoc read queries that retrieve data from a single collection or bucket and complex join queries that span multiple collections to get data.

Table IV and Table V respectively show simple retrieval queries' execution times for the two datasets. There is a linear complexity increase in query response time as the database size grows. This is because the database execution time increases with the same queries, by scaling the dataset from 10k patients to 100k patients. However, query optimization techniques like indexing are not likely to be used for that type of ad hoc analytical query. Execution time was reported with and without documents indexing for the small dataset to declare the benefit of indexing to enhance query performance. Even Q23 scanned the observation document, which has the largest size. It achieved the smallest execution time value because the two attributes of the query are indexed to make a covering query. Covered queries usually retrieve results directly from the index without the need to access datasets documents. Without indexing, Q23 achieved the highest execution time for both databases. The same explanation is for Q7, Q8, Q9, and Q10 execution time. These queries utilized indexing deathdate and birthdate attributes as a covering index. Q24 achieved the lowest execution time because the number of scanned attributes is quite smaller than many other queries besides that it used the covering index. Even with indexing, Q6 reported high response time values. This is because of the low selectivity of the encounter class's attribute values. MongoDB reported observable better execution time than Couchbase in the case of simple queries scanning the small 10k patient dataset.

For the larger 100k patient dataset, MongoDB still reports better execution time than Couchbase except for Q4, Q6, and Q13. This is because of the large number of the queries` scanned attributes and usually, Couchbase is reported to be better for large-scale data. By utilizing Couchbase analytical services in multi-node architecture, database performance was enhanced significantly.

TABLE IV. SIMPLE QUERIES EXECUTION TIME FOR 10K PATIENT DATASET

	Execution time (ms) with index		Execution time (ms) without index		Number of examined documents
	Couchbase	***MongoDB***	***Couchbase***	***MongoDB***	
Q1	711	39	731.6	64	12352
Q2	411.4	12	419.1	42	5499
Q3	413.9	9	812.1	19	12352
Q4	22700	319	50200	871	321528
Q5	7300	288	5800	453	114544
Q6	20500	505	48900	806	321528
Q7	266.3	2	691.5	53	12352
Q8	158.6	14	721.5	22	12352
Q9	169.5	12	778.6	56	12352
Q10	241.4	17	840.8	26	12352
Q11	240.4	10	690.4	66	12352
Q12	173.5	18	670.4	77	12352
Q13	12900	276	7000	452	143110
Q14	25800	328	1m0.8s	631	431262
Q15	23800	393	1m11.1s	1073	431262
Q16	88.9	10	235	20	2360
Q17	7200	125	6200	246	114544
Q18	468.7	32	5800	102	114544
Q19	200.5	20	6200	142	114544
Q20	1300	91	5900	123	114544
Q21	2300	78	2400	328	37715
Q22	328.1	21	352.5	35	5417
Q23	3800	90	4m28s	3498	1659750
Q24	50	6	821.1	36	16481
Q25	6800	111	5400	200	100427

TABLE V. SIMPLE QUERIES EXECUTION TIME FOR 100K PATIENT DATASET

	Execution time (ms) with a primary index		Execution time (ms) without index	Number of examined documents
	Couchbase	***MongoDB***	***Couchbase analytics***	
Q1	4600	1169	1500	124150
Q2	377	129	111.90	3188675
Q3	3900	899	323.55	124150
Q4	4700	12137	1990	3188675
Q5	6600	3142	1830	1143900
Q6	7.900	15136	3720	3188675
Q7	5200	447	111.76	124150
Q8	4400	694	86.31	124150
Q9	4400	579	95.64	124150
Q10	4300	864	98.67	124150
Q11	4600	420	78.66	124150
Q12	4700	344	122.34	124150
Q13	1400	5136	867.99	1389858
Q14	24800	13236	4800	4227723
Q15	49200	8974	5.24s	4227723
Q16	919.1	46	128.67	23694
Q17	2800	1796	2210	1143900
Q18	17800	1400	673.41	1143900
Q19	8100	1876	557.71	1143900
Q20	7200	1678	757.81	1143900
Q21	13500	1211	607.60	377726
Q22	1700	439	146.02	51592
Q23	10m0s	31427	19510	16219969
Q24	5100	766	370.29	168160
Q25	21700	3039	781.44	979564

For complex join queries, Table VI and Table VII show execution time for the two datasets correspondingly. For the 10k dataset, Couchbase achieved better response time than MongoDB except for Q2 and Q7, which scanned few attributes. Q12 achieved the highest response time for both back-ends. As it both examined and retrieved many documents. Conversely, Q2 achieved the lowest response time for both back-ends besides Q6 for Couchbase. The number of scanned attributes results from the total number of joined documents attributes. MongoDB was considerably slower than Couchbase for join queries. It was not investigated against join queries for the 100k patient dataset, as it reported a very long response time for the small dataset. Even there is no indexing and there are many examined documents. Couchbase Analytics service enhances the execution time of ad hoc join workload dramatically. Q11 and Q12 reported the lowest and the highest execution time, respectively. This is because the Couchbase Analytics service is usually based on parallel join and those two queries examine the largest and the smallest number of scanned attributes. For two different queries with the same complexity and attribute number, the execution time is approximately the same.

TABLE VI. JOIN QUERIES RESPONSE TIME FOR 10K PATIENT DATASET

Join query	Couchbase execution Time	MongoDB execution Time	Number of examined documents
Q1	26500	6205489	321528
Q2	1800	691	5499
Q3	32800	562485	12,352
Q4	16400	192263	114544
Q5	2700	6725	2360
Q6	1100	31164	321528
Q7	19800	744	431262
Q8	6000	373802	16481
Q9	1600	195980	114544
Q10	1700	67823	321528
Q11	5000	322827	31764
Q12	51900	3085553	1659750
Q13	6900	3002136	1659750

TABLE VII. JOIN QUERIES RESPONSE TIME FOR 100K PATIENT DATASET WITH ANALYTICS SERVICE

Join query	Execution Time Couchbase analytical service	num. of docs scanned
Q1	10800	3312825
Q2	2400	3322000
Q3	15570	3312825
Q4	1920	1268050
Q5	13730	3212369
Q6	3030	3312825
Q7	3320	1050668
Q8	876.43	292310
Q9	2510	3312825
Q10	2210	3312825
Q11	350.12	69709
Q12	39300	19408644
Q13	37980	19408644

VII. DISCUSSION

This research investigates the appropriateness of two DODBMS storing EHRs data for BI workload. For single-node topology, MongoDB performed better for simple look-up queries, but their performance decreased with JOIN queries spanning multiple collections and involving aggregation. It was considerably slower than Couchbase for join queries of larger datasets. Conversely, Couchbase single node cluster reported better performance for complex join queries, but their performance decreased for simple lookup queries.

These results motivated us to utilize Couchbase server MDS architecture to distribute services across several nodes rather than a single node. An interesting result of the Couchbase multi-node cluster performance analysis is that, by using an analytics service with no index, the Couchbase server reported better response times than MongoDB for both types of ad hoc queries. This multi-service architecture decreased the query latency dramatically.

MongoDB uses the $lookup operator to apply JOIN operations, which traverse several collections. According to the result, this costly operation significantly increases latency and overall strain on the database. An alternative solution to handle these costly join operations in MongoDB is to denormalize the data model by embedding elements into their parent objects and performing a regular query. The major disadvantage of this denormalization approach, especially in a distributed system, is that it leads to data redundancy, which causes a significant write-performance downgrade to maintain consistency across multiple copies through multiple write operations. Furthermore, this modeling approach brings additional storage costs. It could be considered as an optimal response scenario with only one user submitting a single query at a time. Besides that, MongoDB provides a native connector for BI, which could provide support for real-time analytics by integrating with a leading BI and analytics tool.

The proposed converged Couchbase EHRs system infrastructure has significant advantages compared to the other loosely coupled alternatives. It supports using a common flexible document data model for both operational and analytical data with no transformation required for analysis. So, it could offer fast reporting and decision-making capabilities based on real-time and advanced analytics for large volumes of data without needing to move data around. Instead of managing and synchronizing many systems with several connections points [31].

The analytical workload is usually read and compute-intensive because it entails more massive costly calculations over data. So, it is typically ideal to perform analytical workload in isolation. Couchbase Server separates the competing workloads into independent services and isolates them from each other. Consequently, interference among them is minimized. Operational workload latency and throughput are isolated from analytical query workload slowdowns but without the complexity of operating a separate analytical database. It allows running analytics in extremely current data with no ETL process that delays data.

An important principle in medical systems is that when any data element is modified, it will not be overwritten. Rather, a new element with a suitable link is added. On the other hand, large-scale healthcare applications scenarios usually do not strictly enforce ACID properties. They deploy optimistic locking, as they desire isolation, but not at the rate of a significant performance penalty. For such scenarios, Couchbase Server, which is an append-only store that enforces the BASE (Basically Available, Soft State, Eventually Consistent) philosophy, seems to be a promising solution covering many EHRs systems use-cases. Their flexible MDS framework enables users to scale or shrink their cluster, involving data management resources as requirements change.

VIII. CONCLUSION

This paper investigates the appropriateness of DODBMS for handling operational BI workload. MongoDB does not provide satisfactory response time for unplanned join queries spanning multiple documents contrarywise Couchbase which reports better performance for these complex queries. By utilizing its analytics services and multi-dimensional scaling architecture, Couchbase Server multi-node cluster outperforms the response times of MongoDB for both simple and complex healthcare data access patterns. Couchbase is the ideal solution for our needs as long as we need to store unstructured health data in various EHR systems. This paper proposes a tightly coupled Couchbase healthcare framework that can efficiently adapt to different workload needs. Along this path, using complementary Couchbase services for architecting a monolithic framework enables new technical capabilities which serve as the foundation for a new class of intelligent applications such as machine learning, real-time operational reporting. Finally, the emergence of HTAP DBMSs does not imply the end of the massive, monolithic OLAP warehouse. But such systems will be required to serve as a universal back-end database for all an organization's front-end OLTP silos.

REFERENCES

[1] F. Winter, A., Haux, R., Ammenwerth, E., Brigl, B., Hellrung, N., Jahn, Health Information Systems: Architectures and Strategies, Springer-Verlag London, 2011. https://doi.org/DOI 10.1007/978-1-84996-441-8.

[2] G. Pradeep K. Sinha, A.D. Sunder, Prashant Bendale, Manisha Mantri, Electronic Health Record Standards, Coding Systems, Frameworks, and Infrastructures, Wiley, 2013. https://doi.org/10.1055/s-0038-1638463.

[3] S.M. Freire, D. Teodoro, F. Wei-Kleiner, E. Sundvall, D. Karlsson, P. Lambrix, Comparing the performance of NoSQL approaches for managing archetype-based electronic health record data, PLoS One. 11 (2016) 1–20. https://doi.org/10.1371/journal.pone.0150069.

[4] R. Jain, In Search of Database Nirvana:The Challenges of Delivering Hybrid Transaction/Analytical Processing, (2016).

[5] F. Özcan, Hybrid Transactional Analytical Processing : A Survey, (2017) 1771–1775. https://doi.org/https://dl.acm.org/doi/10.1145/3035918.3054784.

[6] M. Stonebraker, U. Çetintemel, 'One size fits all': An idea whose time has come and gone, Proc. - Int. Conf. Data Eng. (2005) 2–11. https://doi.org/10.1109/ICDE.2005.1.

[7] M.J.S. Benjamin M. Davis, Glen F. Rall, Using Electronic Health Records for Population Health Research: A Review of Methods and Applications, Physiol. Behav. 176 (2017) 139–148. https://doi.org/10.1016/j.physbeh.2017.03.040.

[8] L. Marco-Ruiz, D. Moner, J.A. Maldonado, N. Kolstrup, J.G. Bellika, Archetype-based data warehouse environment to enable the reuse of electronic health record data, Int. J. Med. Inform. 84 (2015) 702–714. https://doi.org/10.1016/j.ijmedinf.2015.05.016.

[9] V. Palanisamy, R. Thirunavukarasu, Implications of big data analytics in developing healthcare frameworks – A review, J. King Saud Univ. - Comput. Inf. Sci. 31 (2019) 415–425. https://doi.org/10.1016/j.jksuci.2017.12.007.

[10] K.K.Y. Lee, W.C. Tang, K.S. Choi, Alternatives to relational database: Comparison of NoSQL and XML approaches for clinical data storage, Comput. Methods Programs Biomed. 110 (2013) 99–109. https://doi.org/10.1016/j.cmpb.2012.10.018.

[11] S. Navathe, R. Elmasri, Fundamentals of Database Systems, 7th ed., Addison-Wesley, USA, 2016.

[12] M. Stonebraker, S. Madden, D.J. Abadi, S. Harizopoulos, N. Hachem, P. Helland, The end of an architectural Era (It's time for a complete rewrite), 33rd Int. Conf. Very Large Data Bases, VLDB. (2007) 1150–1160. https://doi.org/10.1145/3226595.3226637.

[13] N. Leavitt, Will NoSQL Databases Live Up to Their Promise?, Computer (Long. Beach. Calif). 43 (2010) 12–14. https://doi.org/10.1109/mc.2010.58.

[14] R. Cattell, Scalable SQL and NoSQL data stores, SIGMOD Rec. 39 (2010) 12–27. https://doi.org/10.1145/1978915.1978919.

[15] A. Davoudian, L. Chen, M. Liu, A survey on NoSQL stores, ACM Comput. Surv. 51 (2018). https://doi.org/10.1145/3158661.

[16] C.S. Kruse, C. Kristof, B. Jones, E. Mitchell, A. Martinez, Barriers to Electronic Health Record Adoption: a Systematic Literature Review, J. Med. Syst. 40 (2016). https://doi.org/10.1007/s10916-016-0628-9.

[17] T. Nguyen, Big data system for health care records, J. Sci. Policy Manag. Stud. 33 (2017) 146–156. https://doi.org/10.25073/2588-1116/vnupam.4101.

[18] M.Z. Ercan, M. Lane, Evaluation of NoSQL databases for EHR systems, Proc. 25th Australas. Conf. Inf. Syst. ACIS. (2014).

[19] P.P. Khine, Z. Wang, A review of polyglot persistence in the big data world, Information. 10 (2019). https://doi.org/10.3390/info10040141.

[20] R. Sánchez-De-Madariaga, A. Muñoz, R. Lozano-Rubí, P. Serrano-Balazote, A.L. Castro, O. Moreno, M. Pascual, Examining database persistence of ISO/EN 13606 standardized electronic health record extracts: Relational vs. NoSQL approaches, BMC Med. Inform. Decis. Mak. 17 (2017) 1–14. https://doi.org/10.1186/s12911-017-0515-4.

[21] K. Kaur, R. Rani, Managing Data in Healthcare Information Systems: Many Models, One Solution, Computer (Long. Beach. Calif). 48 (2015) 52–59. https://doi.org/10.1109/MC.2015.77.

[22] H.M. Kruse, A. Helhorn, L.A. Phan-vogtmann, E. Thomas, A.J. Heidel, K. Saleh, A. Scherag, Modeling a Graph Data Model for FHIR Resources, (2019) 398355.

[23] S. Kalogiannis, K. Deltouzos, E.I. Zacharaki, A. Vasilakis, K. Moustakas, J. Ellul, V. Megalooikonomou, Integrating an openEHR-based personalized virtual model for the ageing population within HBase, BMC Med. Inform. Decis. Mak. 19 (2019) 1–15. https://doi.org/10.1186/s12911-019-0745-8.

[24] D. Teodoro, E. Sundvall, M.J. Junior, P. Ruch, S.M. Freire, ORBDA: An openEHR benchmark dataset for performance assessment of electronic health record servers, PLoS One. 13 (2018) 1–22. https://doi.org/10.1371/journal.pone.0190028.

[25] S. El Helou, S. Kobayashi, G. Yamamoto, N. Kume, E. Kondoh, S. Hiragi, K. Okamoto, H. Tamura, T. Kuroda, Graph databases for openEHR clinical repositories, Int. J. Comput. Sci. Eng. 20 (2019) 281–298. https://doi.org/10.1504/IJCSE.2019.103955.

[26] H.Y. Yip, N.A. Taib, H.A. Khan, S.K. Dhillon, Electronic health record integration, Encycl. Bioinforma. Comput. Biol. 1–3 (2018) 1063–1076. https://doi.org/10.1016/B978-0-12-809633-8.20306-3.

[27] E. Choi, M.W. Dusenberry, G. Flores, Z. Xu, Y. Li, Y. Xue, A.M. Dai, Learning Graphical Structure of Electronic Health Records with Transformer for Predictive Healthcare, ICML 2019 Work. Learn. Reason. with Graph-Structured Data. (2019). https://graphreason.github.io/papers/38.pdf.

[28] H. Plattner, A common database approach for OLTP and OLAP using an in-memory column database, SIGMOD-PODS'09 - Proc. Int. Conf. Manag. Data 28th Symp. Princ. Database Syst. (2009) 1–2. https://doi.org/10.1145/1559845.1559846.

[29] E. Begoli, D. Kistler, J. Bates, Towards a heterogeneous, polystore-like data architecture for the US Department of Veteran Affairs (VA) enterprise analytics, IEEE Int. Conf. Big Data. (2016) 2550–2554. https://doi.org/10.1109/BigData.2016.7840896.

[30] N. Yuhanna, M. Gualtieri, Emerging Technology: Translytical Databases Deliver Analytics At The Speed Of Transactions Next-Generation Databases Seamlessly Support Both Transactions And Analytics, (2015). http://www.odbms.org/wp-content/uploads/2017/10/Forrester-report-Translytical-Databases.pdf.

[31] F. Biscotti, M. Pezzini, N. Rayner, J. Unsworth, R. Edjlali, S. Tan, E. Rasit, A. Norwood, A. Butler, W. Roy Schulte, Real-time Insights and Decision Making using Hybrid Streaming, (2015). http://www.gartner.com/technology/about/ombudsman/omb_guide2.jsp.

[32] K. Jee, G.H. Kim, Potentiality of big data in the medical sector: Focus on how to reshape the healthcare system, Healthc. Inform. Res. 19 (2013) 79–85. https://doi.org/10.4258/hir.2013.19.2.79.

[33] M. Al Hubail, A. Alsuliman, M. Blow, M. Careyl, D. Lychagin, I. Maxon, T. Westmannl, Couchbase analytics: NoETL for scalable NoSQL data analysis, Proc. VLDB Endow. 12 (2018) 2275–2286. https://doi.org/10.14778/3352063.3352143.

[35] C.P. Services, data modeling guide, Ann. Oncol. 32 (2021) iii. https://doi.org/10.1016/s0923-7534(21)01110-8.

[36] M. Sandeep Kumar, J. Prabhu, Comparison of nosql database and traditional database-an emphatic analysis, Int. J. Informatics Vis. 2 (2018) 51–55. https://doi.org/10.30630/joiv.2.2.58.

[37] P. Sadalage, M. Fowler, NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence, 1st ed., Addison-Wesley Professional, 2012. https://doi.org/0321826620.

[38] V. Herrero, A. Abelló, O. Romero, NOSQL design for analytical workloads: Variability matters, Lect. Notes Comput. Sci. (Including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics). 9974 LNCS (2016) 50–64. https://doi.org/10.1007/978-3-319-46397-1_4.

[39] Couchbase, Moving from Relational to NoSQL : How to Get Started, (2020).

[40] R.A.S.N. Soransso, M.C. Cavalcanti, Data modeling for analytical queries on document-oriented DBMS, in: Proc. ACM Symp. Appl. Comput., 2018: pp. 541–548. https://doi.org/10.1145/3167132.3167191.

[41] Apache, Apache CouchDB, (2018) 495. http://couchdb.apache.org/.

[42] MongoDB, (2020). https://www.mongodb.com/.

[43] D. Borkar, R. Mayuram, G. Sangudi, M. Carey, Have your data and query it too: From key-value caching to big data management, Proc. ACM SIGMOD Int. Conf. Manag. Data. 26-June-20 (2016) 239–251. https://doi.org/10.1145/2882903.2904443.

[44] Couchbase, Couchbase Under the Hood - An Architectual Overview, (2019) 1–32.

[45] A.M.C. de Araújo, V.C. Times, M.U. da Silva, PolyEHR: A Framework for Polyglot Persistence of the Electronic Health Record, He 17th Int. Conf. Internet Comput. Internet Things. (2016) 71–78. http://worldcomp-proceedings.com/proc/p2016/ICM3836.pdf.

[46] R. Sreekanth, G. Rao, S. Nanduri, Big Data Electronic Health Records Data Management and Analysis on Cloud with MongoDB: A NoSQL Database, Ijaegt.Com. (2015). http://ijaegt.com/wp-content/uploads/2015/05/409533-pp-946-949-venu.pdf.

[47] F. Khennou, Y.I. Khamlichi, N.E.H. Chaoui, Improving the use of big data analytics within electronic health records: A case study baseD OpenEHR, Procedia Comput. Sci. 127 (2018) 60–68. https://doi.org/10.1016/j.procs.2018.01.098.

[48] O. Schmitt, T.A. Majchrzak, Using document-based databases for medical information systems in unreliable environments, 9th Int. Conf. Inf. Syst. Cris. Response Manag. (2012) 1–10. https://www.researchgate.net/profile/Oliver_Wannenwetsch/publication/272885088_Using_Document-Based_Databases_for_Medical_Information_Systems_in Unreliable_Environments/links/5825780908ae61258e456ad3/Using-Document-Based-Databases-for-Medical-Information-.

[49] FHIR, Fast Healthcare Interoperability Resources, https://fhir.org/.

[50] J. Walonoski, S. Klaus, E. Granger, D. Hall, A. Gregorowicz, G. Neyarapally, A. Watson, J. Eastman, SyntheaTM Novel coronavirus (COVID-19) model and synthetic data set, Intell. Med. 1–2 (2020) 100007. https://doi.org/10.1016/j.ibmed.2020.100007.

[51] Synthea, (n.d.). https://synthea.mitre.org/downloads.

[52] The MITRE Corporation, Synthea by the Standard Health Record Collaborative, (2017). https://synthetichealth.github.io/synthea/.

[53] J. Walonoski, M. Kramer, J. Nichols, A. Quina, C. Moesel, D. Hall, C. Duffett, K. Dube, T. Gallagher, S. McLachlan, Synthea: An approach, method, and software mechanism for generating synthetic patients and the synthetic electronic health care record, J. Am. Med. Informatics Assoc. 25 (2018) 230–238. https://doi.org/10.1093/jamia/ocx079.

[54] MongoDB, MongoDB : Delivering Real-Time Insight with Business Intelligence & Analytics, MongoDB White Pap. (2017). http://s3.amazonaws.com/info-mongodb-com/MongoDB_BI_Analytics.pdf.

[55] S. Balsamo, A. Di Marco, P. Inverardi, M. Simeoni, Model-based performance prediction in software development: A survey, IEEE Trans. Softw. Eng. 30 (2004) 295–310. https://doi.org/10.1109/TSE.2004.9.

[56] CSV File Data Dictionary • synthetichealth/synthea Wiki • GitHub, (n.d.). https://github.com/synthetichealth/synthea/wiki/CSV-File-Data-Dictionary .