ISSN: 2229-6956 (ONLINE)
DOI: 10.21917/ijsc.2024.0489

AN ETHEREUM-BASED ELECTRONIC HEALTH RECORD SYSTEM WITH SQL QUERY SUPPORT

Suvam Tamang[1], M.S. Srinath[2], Pallav Kumar Baruah[3]
Department of Mathematics and Computer Science, Sri Sathya Sai Institute of Higher Learning, India

Abstract

Blockchains' power of decentralization, immutability, and transparency has found its application in many fields. The blockchain is an append-only structure. As the blockchain grows, accessing blocks from the past in an efficient manner has become a challenging task. Further, blockchains were not initially envisioned to be read-heavy systems. With the passage of time, more and more applications are using blockchains and therefore blockchains need to provide active support for high read loads concerning the history state as well. To facilitate data querying in blockchain, we have proposed an SQL query processing feature in the Ethereum blockchain through a decentralized application. More specifically, we build an Ethereum-based Electronic Health Record (EHR) system with SQL query support. The following approaches for query processing have been explored and implemented: (i) linearly scanning blockchain, (ii) scanning only from a user-specified block, (iii) replication in database, (iv) indexing in a database, and (v) using smart contracts. Our timing analysis across these implementations reveals that the smart contracts-based approach has reasonable performance gains compared the other approaches.

Keywords:

Electronic Health Records, Blockchain, Smart Contracts, Web3, IPFS.

1. INTRODUCTION

Health records have been used by humans ever since the Bronze Age. Its first traces found in the form of ancient Egyptian inscriptions date back to 1600 BC [1]. It has witnessed significant transformation over the past three and a half millennia. In recent years, governments have taken massive initiatives to maintain the health records of their citizens. In the year 2009, the HITECH Health Information Technology for Economic and Clinic Health act passed by the

U.S. government to incentivize people for the usage of EHR was a great boost for the spread in the usage of EHR [2] as also the affordability of EHR was dropping down. The Indian government launched The Ayushman Bharat Digital Mission (ABDM) in response to the emerging need to digitize healthcare in India. COVID-19 was the main reason for reconsidering the healthcare ecosystem as the system needed to be more interoperable and streamlined. ABDM has eased the process of data sharing and achieved Universal Health Coverage (UHC) [3]. Wisner et al. [4] have discussed the issues that the paper-based system is prone to and how the EHR system solves those. The EHR system aims to mitigate the issues existing with the traditional paper-based records system, such as:

- **Security Compromise**: The traditional health record system where a centralized authority or a cloud service keeps track of paper-based records is an easy target for attackers and hackers [5]. Further, insider attacks are quite a cause of concern here. Mazurek et al. [6] and Argaw et al. [7] also talk about how hospitals have become a target of data breach attacks. Since October 2009, 173 million data entries have been compromised.
- **Lack of Data Integrity**: Data loss, deletion, and corruption are other causes of concern. Tamper-proof availability cannot be guaranteed in such paper-based systems as a mere paper needs to be manipulated.
- **Difficulty of Accessibility**: The patients themselves could be unable to access their medical histories in paperbased systems simply because the paper document went missing or was destroyed. Yao et al. [8] talks about the vital need of the availability of patient data.
- **Lack of System Interoperability**: Also the medical records tend to get scattered in quite a lot of places and different forms. Also, cross-region sharing may not be possible due to the laws and regulations differing across various regions. This severely affects system interoperability. Reisman et al. [9] also talk about the varying system functionalities across healthcare institutions and interoperability issues.
- **High-Cost Requirements**: Centralized databases setting up takes up quite a lot of planning, funds, and maintenance costs thereafter, though this seems to get minimized when using cloud services.

EHR solves all the above issues as it is a digital append-only document that tracks every clinical visit of a person. Information stored in this document includes the patient's symptoms, diagnostics, medications, past medical history, demographics, laboratory data, immunizations, MRI images, and radiology reports [10]. If we have to jot down the benefits of EHR in brief, they are as follows:

- EHR leads to the improvement in the quality of healthcare by reducing medical errors that creep in easily because of paper-based documents and can also contain measures specific to a patient that can highly specify the care appropriate to the patient.
- EHR leads to the minimization of cash flows in the long run as opposed to paper-based documents that require a lot of paperwork and thereby improve both the financial and operational performance.
- EHR also easily provides many research institutes to research as the data is available on the web thereby drastically improving the health of the population.

Blockchain was not conceived for any sophisticated querying mechanism. The major focus is on data storage, there are hardly any user-friendly avenues to query a blockchain. We would like to bridge this gap. We provide a SQL querying ability for querying the Ethereum blockchain for the details of patient health records through a decentralized application. We have made use of IPFS (Inter Planetary File System) as the major storage component and blockchain as the metadata storage layer. To this end, we have made the following contributions through this work.

- A survey of the existing literature on Electronic Health Record Systems on the blockchain.
- A comparative study of the existing frameworks of the existing query processing system in the blockchain.
- Proposed four different methods of query processing in the blockchain each with its drawbacks and characteristics.
- Provided a dApp that incorporates all the features mentioned above and that runs on the Ethereum blockchain.

The remainder of this paper is organized as follows. Section 2 deals with all the background details regarding blockchain and its implementations. Section 3 talks about the literature review that aligns with the problem of our interest. The proposed system architecture along with all the system components with their respective functionalities have been discussed in Section 4. The performance analysis and the results are presented in Section 5. The last section comprises conclusions thus made and the references that were used.

2. BACKGROUND

The term blockchain came into the light way back in 1991 when Stuart Haber and W. Scott Stornetta thought of a system consisting of a cryptographically secured chain of blocks that ensured that the document timestamp could not be tampered with. Merkle Trees or Hash Trees named after Ralph Merkle found its collaboration in the following year. Satoshi Nakamoto, an individual or a group of individuals [11] adapted the technique in 2008 and created the first cryptocurrency bitcoin having introduced Bitcoin blockchain through a white paper. The paper mainly discusses an electronic payment system where in the absence of a centralized authority monitoring the payment, the cryptographic proofs formed a basis of trust. The Proof of Work (PoW) consensus algorithm found its first formal application in this paper. The decentralized network has the power to prevent double spending of the digital assets, where always the longest chain wins and gets finally committed to the network in case of forking. All the information is stored in blocks that are linked to each other forming a chain, thereby making it hard to tamper with. Thus, blockchain is a distributed, decentralized ledger of transactions that enables a secure, transparent, and tamper-proof way of storing and sharing information between multiple parties.

2.1 BLOCKS

The basic data structure of a blockchain is the blocks [12]. They are stored in a linked list style where the latest block gets attached to the previous block. A block contains transaction information and other details such as the nonce value, timestamp of the block, previous block hash, etc. The block contents depend on the type of blockchain used. The first block in the blockchain has a special name and is called a genesis block. The most important content of a block is its hash. The hash is what provides authenticity to a block and determines whether the block needs to be added or discarded. The block hash is its fingerprint. If a malicious user tries to change the block contents, its hash value also changes. This warns all the other nodes in the network regarding malicious activity. The Fig.1 shows a typical block structure.

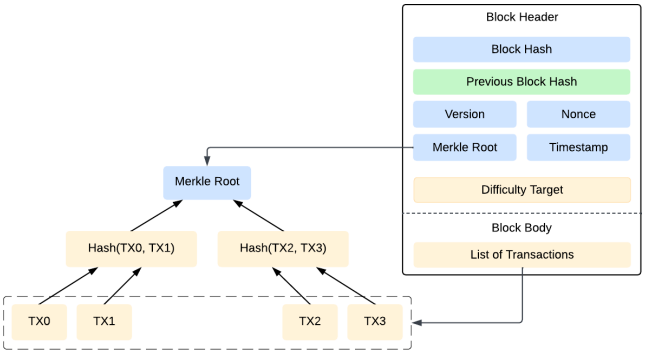

Fig.1. A block structure

2.2 TRANSACTIONS

A transaction is created whenever there is a data transfer from a sender to a receiver. The main contents of a transaction are sender, receiver, and value although it consists of other information such as timestamp, input value, transaction hash, etc. A transaction thus fired results in a blockchain state change. A block as discussed above stores all these transactions. For a transaction to be valid all the nodes in the network verify the transaction based on the consensus protocol that the network follows.

2.3 CONSENSUS

A transaction gets validated through consensus protocols. Two of the most popular ones are Proof of Work (PoW) and Proof of Stake (PoS). Bitcoin blockchain uses PoW whereas the Ethereum blockchain uses PoS. Consensus is a set of rules that needs to be followed in the network. Consensus is achieved when multiple nodes in the network participate. The more the number of participants stronger the consensus. Whenever a block proposed by a miner goes through the consensus protocol as a valid and genuine block, the miner is now eligible for a reward in the form of the blockchain's native currency. Theoretically, attackers can collaborate to control 51% of the network computation power after which they have a higher chance of proposing the next block even if it is faulty. This scenario in the blockchain is known as a 51% attack. PoS is a consensus mechanism in which validator nodes stake some amount of asset as "collateral", and based on the amount of asset staked and the time duration for which the amount was staked, the next validator node is chosen and the node gets to propose a new block into the system [13]. The validator node receives a reward on a genuine proposal whereas if it tries to execute the foul play, the asset kept at stake will be confiscated and it will be detained from participating in the future consensus. Since PoS doesn't involve any complex mathematical problem-solving, it is more energy-efficient and less expensive than PoW.

2.4 ETHEREUM

Ethereum is a programmable blockchain. Unlike bitcoin blockchain which mainly focused on transactions and only transactions Ethereum was introduced to make blockchain more lively by being able to use blockchain in many fields such as healthcare, education, social media, supply chain, etc. Ethereum was first conceived by a Russian programmer, Vitalik Buterin in the year 2014 when we wrote a paper titled Ethereum: A Next-

Generation Smart Contract and Decentralized Application Platform [14]. Ethereum was formally released in the year 2015 by Buterin alongside Joe Lubin. Ethereum allows developers to write and deploy their code in blockchain in the form of Smart Contracts and thus enables the creation of decentralized applications or dApp in short. Thus, the dApps built can run with no fraud, downtime, or thirdparty interference. Ethereum is a blockchain with a built-in Turing Complete Language. Ethereum initially used PoW as its consensus mechanism but it moved on to PoS on September 15th, 2022 06:42:42 UTC, at block 15537393 known as The Merge.

2.5 ETHER

Ether is the native currency of the Ethereum blockchain. It is used for transaction payments. When a user has to get his transaction verified and added to the network, then a transaction fee, also known as gas fees needs to be paid, depending on the amount of computation that transaction is going to involve and the status of the network. Ethers are created by the underlying Ethereum protocol and cannot be created by a user. The transaction fee is calculated using the formula:

$$Transactionfee = Gasprice \times Gaslimit \quad (1)$$

where the gas price is the price of a unit of gas, and the gas limit is the maximum amount of gas that the transaction is willing to spend.

2.6 SMART CONTRACT

Smart Contracts (SCs) are pieces of code that are deployed and executed in the blockchain. In fact, in simple words, a smart contract is a simple computer program that gets executed when certain predefined conditions are met and thereby facilitates the transfer of assets between two parties. SCs was first introduced by Nick Szabo, an American computer scientist, and cryptographer, in the year 1994 [15]. It is the revolutionizing technology that gave Ethereum its usefulness in almost every field of application in today's world of web3. The business logic can well be coded in the SCs and deployed in the blockchain after which whenever the condition as specified in the logic is satisfied, the contract executes automatically and performs necessary actions of fund transfer or any other activity. Ethereum Virtual Machine (EVM) is responsible for smart contract execution.

3. RELATED WORK

Blockchain as such has been applied in many ways in the field of medical science but the real issue lies in the fact that most of them don't talk about the query processing feature in the blockchain. Some of the papers do talk about query processing but they haven't described query processing in specific fields like healthcare which is our problem of interest. Daraghmi et al. [16] proposes a system called MedChain that is mainly concerned with the secure sharing and interoperability of patients' medical records among healthcare providers. Advanced encryption techniques to maintain security aspects have been discussed and timely based medical records access after which the access rights will be revoked has also been discussed.

Shuaib et al. [17] implement a secure, sharable medical record system on Hyperledger Besu using Practical Byzantine Fault Tolerance (PBFT) consensus and found their performance improving greatly in terms of transaction throughput, and latency than existing blockchain systems. Rouhani et al. [18] also talk about the blockchain being more used for cryptocurrency aspects and consensus protocols like PoW adding to the performance overhead. They talk about an asset management system that stores and manages the medical records of the patients in the Hyperledger fabric. Azaria et al. [19] propose a system called MedRec that allows easy access of blockchains to the patients. They also propose a modular architecture that can easily integrate with the data storage of the local existing systems for easy adoption. One of the very interesting features of this system is that it allows the stakeholders to participate in the system as a miner and by maintaining the network security using consensus protocols they get patient records as rewards in return. Shanaz et. al. [20], also discuss building an Electronic Health Record System on the blockchain but they don't provide any hints as to how queries can be performed.

Bragagnolo et al. [21] very much aligns with our goal of providing SQL query support to the Ethereum blockchain. Their work does support SQL, select from where clause but they are mostly concerned with fetching the blocks, accounts, and transactions information and there are many sources available online that provide such kind of information. Cheng et al. [22] proposes a system that can perform range queries over a Boolean range. Both the techniques of Bloom filters and Merkle Tree have been used for efficient querying of data. They define a bloom filter of the size of the query range and the exact blocks are directly accessed. Merkle trees help to determine the authenticity of the data. This system can be used in a wide variety of fields in daily life. Off-chain storage and computation in the form of a Hadoop Distributed File System (HDFS) have been proposed by Linoy et al. [23]. Their main idea was to provide a wide variety of efficient query support and easy dApp integration of their system.

Przytarski et al. [24] discusses the complexities involved in querying a blockchain. It talks about how blockchain has come a long way from its inception in 2008 by Satoshi Nakamoto, where the focus was only on Bitcoin use. Today it has found its use in various fields like healthcare, supply chain, education, e-voting, etc. and emphasis has to be laid on the query aspects as well. For correct and faster diagnosis of a patient, a doctor may need a past medical history of the patient. For a land registration application, the land buyer may need access to the past ownership chain of the land he wants to buy. In Supply Chain related applications a product's authenticity may be determined by tracing back its supply chain of deliveries, from whom and at what time the product was delivered in the delivery process to the actual patient. It describes the need for complex queries supporting framework in blockchain in various fields such as Health Data Management, Financial Accounting, Registries, Food Supply Chains, and E-voting.

Przytarski et al. [25] also discuss the concepts of an object-based data model in blockchain and introduce Constant Objects and Expandable Objects. The paper concludes with the mention of state-of-the-art technologies such as some database that has blockchain properties such as BigchainDB [26], and some blockchain that have database properties such as blockchainDB [27].

Li et. al. [28] talk about adding a query layer on top of an Ethereum blockchain. They raise the concern about the lack of developer-friendly APIs for accessing data from the blockchain. EtherQL provides highly efficient query primitives for analyzing Ethereum blockchain data. They have included range queries, and top-k queries, and also claim that their work can be integrated with other applications with high flexibility. Pratama et al. [25] is an extension to the work done by Li et al. [28] as discussed above. Their extension is in terms of the variety of queries supported. Their search parameters can have multiple parameters in a retrieval query. Apart from these multiple parameter extensions, their system also supports aggregrate queries that perform min, max, count, and sum operations and ranking queries that give results in ascending or descending as specified by the user.

Han et al. [29] make it clear that an Ethereum node stores data in LevelDB and it is not suitable for query purposes as it stores data internally in the form of a Trie. The authors built the system using an embedded database SQLite. This paper very aptly describes standard transactions and smart contract transactions. The query manager uses SQLite to perform SQL queries. This method of query processing though improved efficiency and scalability but the queries supported were only SELECT and INSERT operations.

Kaur et al. [30] have also discussed the querying ability using blockchain but with a private blockchain network using Hyperledger Fabric [31] and have not designed any query processing components as such. The data that they store is written to CouchDB, which is a key value store, and they perform queries from CouchDB. Peng et al. [32] gave another idea as to how a database can be populated as and when new data gets written to the blockchain. They have used MongoDB for this purpose. Their concern is that the query system is highly inefficient, and the authenticity of the query result is not very secure. They propose a Verifiable Query Layer where the database in the middle layer is not just filled with the relevant data but at the same time, a database fingerprint is maintained in the blockchain so that whenever a query request comes the authenticity of the database is first checked before the data fetch. This ensures that the data coming from the middleware layer is indeed valid.

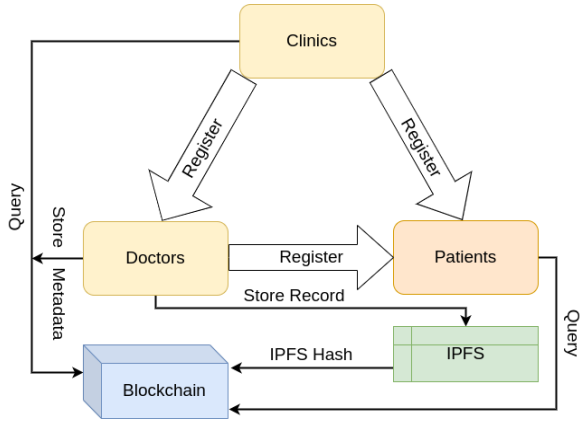

Fig.2. A Basic Schematic of the Proposed System

4. PROPOSED SYSTEM

The proposed system mainly consists of three main actors. The clinics, the doctors, and the patients. A clinic is at the top of the hierarchy. A doctor becomes a valid doctor only after he has been registered by a clinic. Similarly, a patient becomes a valid patient only if he has been registered into the system first, by a clinic, or by a valid doctor. Not all the queries for a clinic, doctor, or patient are the same. The details of the query supported will be discussed below. The Fig.2 shows a basic schematic of the proposed system. It shows that a clinic can register both the patient and the doctor whereas a doctor can register patients alone. Furthermore, the doctor takes the help of IPFS to store the actual patient record whereas the metadata information is stored in the blockchain. The clinics, doctors and patients can query the blockchain to retrieve relevant information to which they have access.

4.1 CLINIC FUNCTIONALITIES

In the three-role system of ours, the clinic is at the top of the hierarchy. Now let's summarize some of the clinic functionalities as shown in the use case diagram.

- **Add Clinic**: It handles the registration of clinics to the system. The inputs that need to be given for successfully registering a clinic to the system are clinic name, clinic metamask address, location, and password.
- **Check Clinic**: This function gets called when the clinic logs in to the system. It takes input as the clinic's metamask address, and password and performs a check in the blockchain whether a clinic exists with the same metamask address and password. Upon successful login, the clinic is routed to its dashboard.
- **Add Patient**: The clinic uses this function to register a new patient to the system, only then is the patient a valid patient. The same patient cannot be added again.
- **Clinic Add Doctor**: The clinic uses this function to register a new doctor to the system, only then is a valid doctor. The same doctor cannot be added again.
- **Get Registered Patients**: Returns all the registered patients along with the department name of their registration.
- **Get Registered Doctors**: Returns all the registered doctors along with the department name of their registration.
- **Get Registered Patients in a Department of a Clinic**: Returns all the patients registered or getting treated in a specific department in a clinic.
- **Get Registered Doctors in a Department of a Clinic**: Returns all the doctors belonging to a specific department in a clinic.
- **Doctor Natural Join Patient**: Gives the clinic a single view of the doctors' and the patients' details. This operation returns records representing patients and doctors from the same department.

4.2 DOCTOR FUNCTIONALITIES

In our three-actor system, the doctor is in the middle of the hierarchy. Some of the available doctor queries and functionalities are as follows:

- **Add Patient**: This function handles the registration of clinics to the system. The inputs that need to be given for successfully registering a patient to the system are the patient metamask address, patient name, clinic address,

name of the registering department, and a new password to set up for the patient.

- **Check Doctor**: This function gets called when the registered doctor logs in to the system. It takes input as the doctor's metamask address, and password and performs a check in the blockchain whether a doctor exists with the same metamask address and password. Accordingly, the doctor will be routed to its dashboard or will be greeted with an error.
- **Doctor Add Record**: This is the method for a doctor to add patient medical records, and metadata, into the blockchain. The input fields that need to be provided are IPFS CID of the patient's actual record, patient metamask address, patient name, doctor name, doctor metamask address, symptoms shown by the patient, diagnosis as performed by the doctor, the department from the which the medical record has been added.
- **Get Registered Patients Records By ID**: This is a query that a doctor can perform to find the medical records of a patient given the patient's metamask address, provided that the address is a valid one.
- **Get Patient Records From Specific Date**: This is a query that a doctor can perform to find the medical records of a patient given the patient's metamask address and specific data (inclusive) after which the doctor would like to see the patient's medical history, provided that the address and date is a valid one.
- **Group By Clause**: This query can be used by the doctors to get a single view of the patient's medical history by grouping a set of doctors who have treated a specific patient as doctors need to specify the patient metamask address when performing group by clause.

4.3 PATIENT FUNCTIONALITIES

- **Check Patient**: This function gets called when the registered patient logs in to the system. It takes input as the patient's metamask address, and password and performs a check in the blockchain whether a patient exists with the same metamask address and password. Accordingly, the patient will be routed to its dashboard or will be greeted with an error.
- **Get Registered Clinics**: Returns all the registered clinics in the system.
- **Get Registered Clinics by Location**: Returns all the registered clinics in the system filtered by the location parameter as specified by the patient. This query is very helpful for the patients in the sense that a patient gets to know which clinics are available in the system.
- **Get Registered Doctors by Clinic Name**: Returns all the registered doctors in the system filtered by the clinic name parameter as specified by the patient. This query helps patients to know all the available doctors as specified by the patient in a specific clinic of interest.
- **Get Registered Doctors by Clinic Name and Department**: Returns all the registered doctors in the system filtered by the clinic name parameter, and doctor's department as specified by the patient.

5. EXPERIMENTS AND RESULTS

5.1 APPROACHES EXPLORED AND IMPLEMENTED

This section discuss the approaches that we have considered and implemented for this problem of performing query processing in the Ethereum blockchain for electronic health records.

5.1.1 Linearly Scanning the Blockchain and Searching for the Required Data:

The first approach carried out is the naive way of performing query processing. That is scanning through all the blocks in the blockchain and searching and accessing data that is relevant. This way of accessing the blockchain is not at all efficient as there are more than 400 million blocks and more than 100 million blocks already in the Polygon Mumbai test net and Goerli test net respectively and accessing them linearly doesn't scale up to our needs. Thus as an approach to start with, we started with linearly scanning the blockchain and found that linearly scanning is not at all practical in reallife case scenarios. Because in order to satisfy just a single query, all the blocks need to be traversed which is highly time consuming and not practical for use cases like ours, which is a time-critical application. The Fig.3 below summarizes this approach.

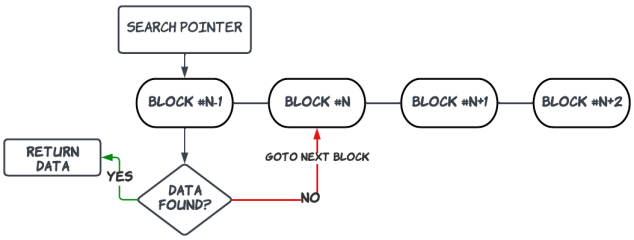

Fig.3. Linearly Scanning the Blockchain

5.1.2 Scanning the Blockchain only from a Specific Block:

The second approach would be to chop the blockchain. By chopping what we would mean to get data from the blockchain only from a user-specified block. Now how to identify the user-specified block is based on the timestamp that the user inputs. The user would specify from which year onward block scanning should be performed. This will reduce the search space by some thousand blocks but will still not be very useful for real-time sensitive applications like our system. As the user needs to be very precise about from which date onwards he/she wants to start scanning the blocks we can't really rely on this way of scanning the blocks as it is not very practical in real-life scenarios as shown in Fig.4.

For both cases as mentioned above the overhead is going to increase as it is not just a matter of scanning each block, there may be a certain number of transactions within each block, and each of them also needs to be scanned and checked whether the transaction we found is the transaction of our interest. This is going to render our searching algorithm to O(kn), where k is the number of transactions in each block and n is the total number of blocks. In the worst-case scenario, this is inefficient especially when the number of records to be searched is huge in number.

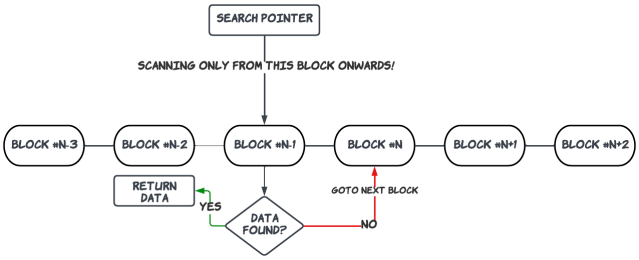

Fig.4. Scanning from a specified block

5.1.3 Using Database Alongside the Blockchain:

The idea behind this approach is that instead of using blockchain alone for fetching data, we used a database for performing query processing. Now, how are we populating the database for this use? To answer this question we take the help of the event feature of smart contracts. An event is simply a signal that a smart contract emits to inform that some activity has happened in the blockchain. The events emitted are written to event logs in a special data structure. It is not feasible for smart contracts to read these values. The data relating to smart contracts are stored in the State Trie whereas events emitted are stored in the Transaction Receipt Trie. For us to be able to query event logs, we need to emit events with some parameter(s) set as indexed parameters, all other parameters are encoded as ABI encoded data portion of the logs. Events are also a form of a communication method in which a smart contract performs some function and the front end can subscribe to these events, by continuously listening to them to know in real-time any activity that has occurred in the blockchain.

As and when a patient's records are input into the blockchain, an event gets emitted. and we know the data has been successfully written to the blockchain. Now, we further read the parameters of the events emitted which will effectively contain the parameters written to the blockchain. These event parameters are then used to populate various database technologies that will be discussed below. After having populated the database we then compare the performance of the queries fired. The Fig.5 summarizes this approach.

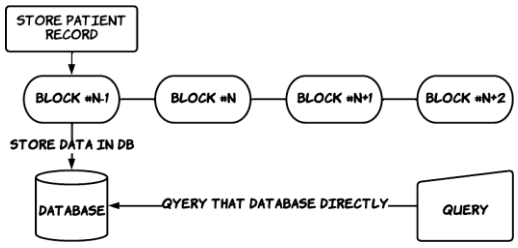

Fig.5. Using Database for Querying

5.1.4 Using Database as an Index:

Another approach that was considered was to use some kind of indexing technique such as storing the information of our interest in a database, MongoDB, to be specific. Here by information of our interest, it is meant that the transaction hash and the block number of transactions committed by the user using the web3 library which is an Application Programming Interface (API) to interact with the Ethereum blockchain. This will again significantly reduce the search time as it becomes a matter of storing the block number in which the information of our interest is stored in the database and then later when fetching the data we perform a lookup in the database first, find the appropriate block numbers and directly go and hit those particular blocks and retrieve the relevant information. This is one more approach to performing query processing which is not quite an efficient approach. Again, the performance of this approach was compared with the smart contracts. The Fig.6 summarizes this approach.

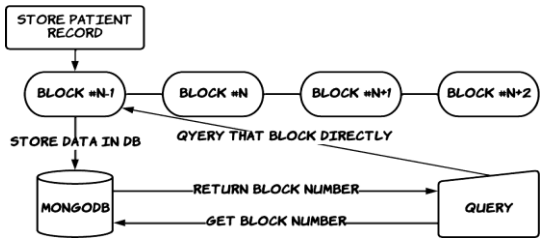

Fig.6. Using Database as an Index for Querying

5.1.5 Using Smart Contracts:

The final approach that we took was to use Smart Contracts (SCs) alone for storing data and fetching data from the blockchain. These structures are stored in the blockchain using solidity data structures and then are used and manipulated based on the logic of the function invoked. The actual patient data which is the JSON files will be encrypted using a dummy key that acts as the patient's private key and is stored in IPFS. Now unless the doctors or any clinicians have access to this private key, won't be able to make anything out of the file and hence won't be able to view the patient's records unless granted by the patient himself. The state change functions consume gas and the user invoking the function pays for it whereas for performing any query processing no gas cost needs to be paid as it just results in the invoking of view functions in the contract.

5.2 PERFORMANCE EVALUATION

5.2.1 System Specifications:

For the most part of the work, the Ubuntu desktop was used. The system specifications are as follows:

- OS Name: Ubuntu 22.10 LTS (Jammy Jellyfish).
- OS Type: 64 bit.
- Memory: 16.0 GiB.
- Processor: Intel Core i5-4670 × 4.
- Hardware Model: Dell Inc. OptiPlex 9020.
- Disk Capacity: 1.0 TB

The Table.1 summarizes all the tools used for this project along with their specifications.

5.2.2 Results and Discussion:

Firstly, a comparative study was made for a single SQL query: select * from patients where patient-id = "some-meta mask-address" using four different storage entities.

- MongoDB
- MySQL
- BigchainDB
- Blockchain (Smart Contracts)

The analysis was carried out such that a single type of query was fired initially followed by a mixed type of query. The number of queries fired increased from 1 to 10,000. When the number of queries was less, the database technologies tended to perform better but when the number of queries increased the smart contracts showed much better execution time. To perform the above-mentioned analysis, we have used axios npm library to make HTTP requests to the database backend servers. To invoke methods of the smart contract, i.e. for the blockchain backend we have made use of the web3 library.

Table.1. Version table

Tool Used	Version
Truffle	v5.6.5
Ganache	v7.5.0
NodeJS	v18.16.0
Metamask	v10.28.3
Solidity	v0.5.16
Web3.js	v1.7.4
IPFS	ipfs version 0.16.0
MySQL	Ver 8.0.32
MongoDB	db version 6.0.5
BigchainDB	v2.2.2
ReactJS	v18.2.0
Docker	Docker version 23.0.4, build f480fb1
Locust	v2.15.1

- **Case 1(a)**: The number of records stored was limited to 100. A single type of query being fired into the system, meaning, a query of the form select * from patients where patient-id= “some-patient-meta maskaddress”. Here “some-patient-meta mask-address” has been kept identical in all the query files. This is what we mean by a single type of query. The queries eventually increased in numbers from 1 to 10K queries and the performance has been summarized in Fig.7.

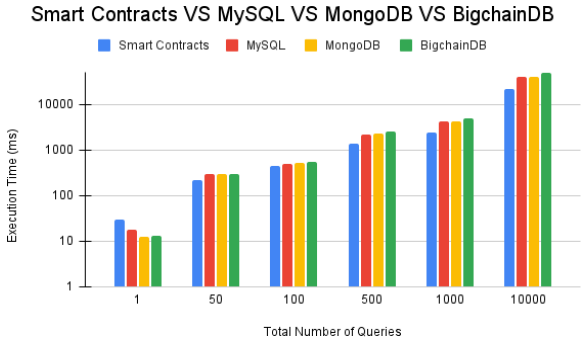

Fig.7. Scenario when a single type of queries used

- **Case 1(b)**: The number of records stored was limited to 100. A Mixed kind of query was fired. By mix, we mean a file of queries that contains queries such as select * from patients where patient-id=“some-patient-meta mask-address” but here “some-patient-meta maskaddress” keeps changing in all the query files. This is what we mean by a mixed type of query. The queries eventually increased in numbers from 1 to 10K queries and the performance has been summarized in Fig.8.

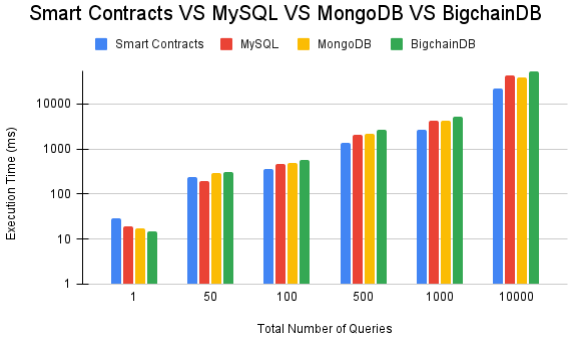

Fig.8. Scenario when mixed type of queries used

- **Case 2**: The analysis for this case was carried out only for mixed kinds of queries as discussed above. The number of records stored in the system has increased to 500. The performance of MySQL, MongoDB, BigchainDB, and smart contracts have been summarized in Fig.9. The number of queries was less the databases as well as bigchainDB gave better execution times whereas when the number of queries increased then it was the smart contract that provided us with better results.

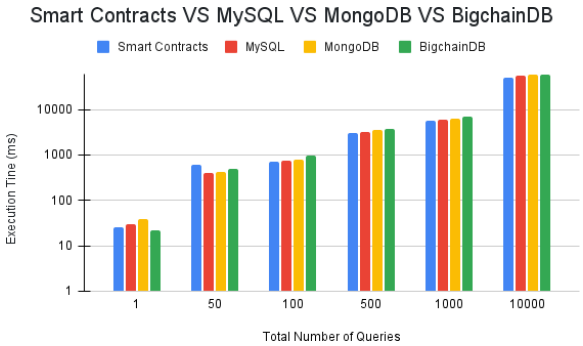

Fig.9. Mixed Queries fired when 500 records were stored Now in all the figures that we see, we can see that when

The second half of the analysis consists of the comparisons of performances between the smart contracts and the database index. Let us go through all the scenarios that we have considered for this analysis part.

- **Scenario 1**: The data quantity in the blockchain is limited to one or a few records. When a single type of query is fired: select * from patients where patient-id =0xD42D954A5aC1fa52706d26Cc26FaDBA5cfC8f773 (this is a dummy metamask address of the patient). The Fig.10 shows the outcome.

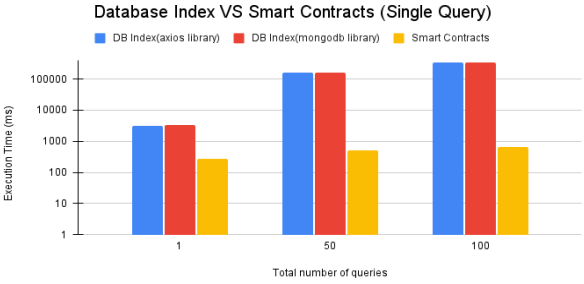

Fig.10. DB Index VS Smart Contracts Single Query Type

- **Scenario 2**: The data quantity in the blockchain is limited to one or a few records. When a mix of queries was fired. The Fig.11 shows the outcome.

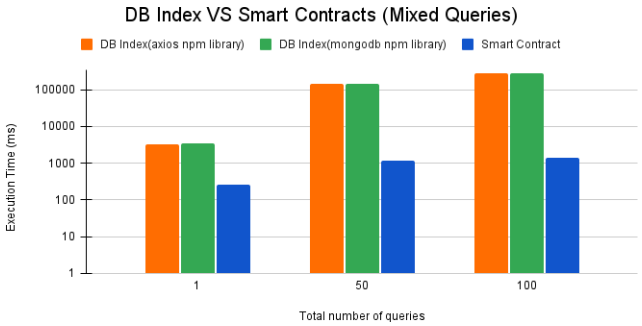

Fig.11. DB Index VS Smart Contracts Mixed Query Type

- **Scenario 3**: When the amount of data stored in the blockchain is increased to many records specifically to 100 records. When a single type of query is fired. The Fig.13 shows the outcome.
- **Scenario 4**: When the amount of data stored in the blockchain is increased to many records specifically to 100 records. When a mixed type of query is fired. The Fig.13 shows the outcome.

Now let us see the throughput as given by the smart contracts and by using the database as an index. These are the average of all the throughput measurements that we took. The Fig.14 summarizes the throughput measurements made. The Fig.15 summarizes the latency measurements made. If we see both the throughput and latency measurements, we find smart contracts outperforming various other methodologies in every aspect of our measurements.

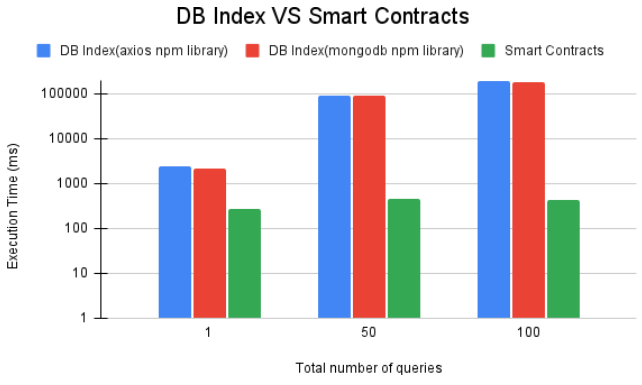

Fig.12. DB Index VS Smart Contracts Single Query Type

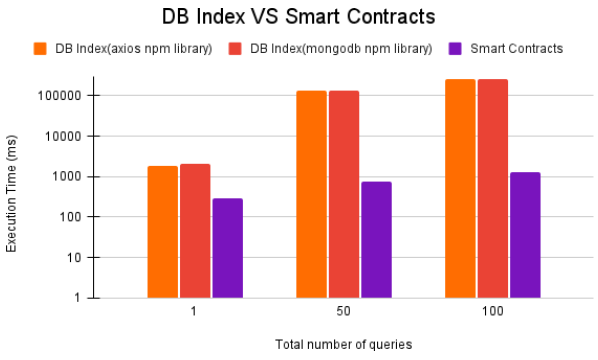

Fig.13 DB Index VS Smart Contracts Mixed Query Type

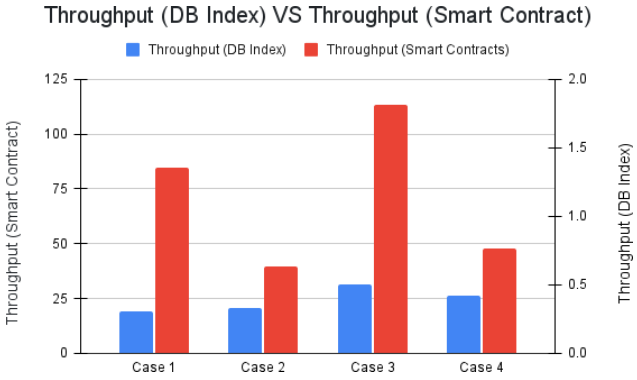

Fig.14. Throughput Comparison

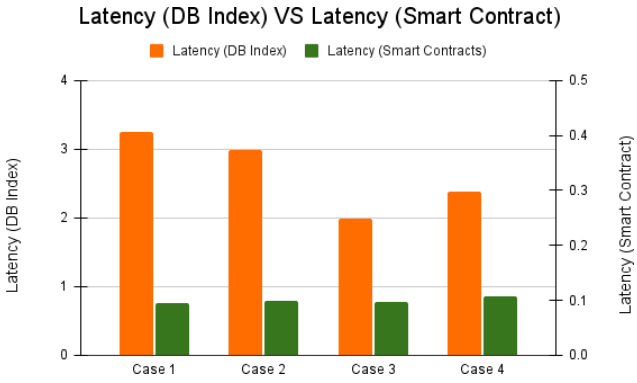

Fig.15. Latency Comparison

6. CONCLUSION

In this paper, we study the problem of efficient query processing in the Ethereum blockchain. We gave a detailed survey of the existing literature first for the existing EHR system in the blockchain and later followed by the existing query processing paradigms available using blockchain. We also explained how some of the literature truly aligns with the problem of our interest and how they differ as well. To handle the issue of inefficiencies creeping in because of linearly scanning the blockchain, we used a method that doesn't scan the blocks from the starting block but only from the block as specified by the user. Linear scanning from a specific block, also was not a practical approach, as it was too block number dependent and can cause havoc if the user first doesn't get the block number right, again quite an impractical approach for a sensitive application like ours. We cannot expect either the clinics, doctors, or patients to remember the block numbers for using our system. Another approach that we came up with was to use the database as an index to the blockchain. The performance of this approach was quite reasonable, although we explored one approach that could give better results than this approach. One more approach was to use blockchain as the main storage layer, but at the same time populating the database so that we use the query facility provided by the database itself to perform query processing. The final approach that we had considered was to use smart contract alone for performing query processing with the front-end taking care of the query semantics and invoking appropriate SC functions. Finally, we proposed a decentralized application that can perform SQL Query processing for three actors, clinics, doctors, and patients with a very simple user interface. With the system specifications that we had and

with the tools we used, we found that the approach based on smart contracts performed much better in almost all aspects, such as execution time, throughput, and latency irrespective of the multiple approaches that we had taken.

REFERENCES

[1] R.S. Evans, "Electronic Health Records: then Now and in the Future", *Yearbook of Medical Informatics*, Vol. 25, No. 1, pp. 48-61, 2016.

[2] "An Act", Available at https://www.govinfo.gov/content/pkg/PLAW-111publ5/html/ PLAW-111publ5.htm, Accessed in 2023.

[3] J.S.S.D. Sharma and A. Rohatgi, "The Ayushman Bharat Digital Mission: Making of India's Digital Health Story", *CSI Transactions on ICT*, Vol. 11, pp. 3-9, 2023.

[4] K. Wisner, A. Lyndon and C. Chesla, "The Electronic Health Record's Impact on Nurses Cognitive Work: An Integrative Review", *International Journal of Nursing Studies*, Vol. 94, No. 3, pp. 1-12, 2019.

[5] M. Mehrtak, S. Seyed Alinaghi and M. Mohsseni Pour, "Security Challenges and Solutions using Healthcare Cloud Computing", Vol. 14, No. 4, pp. 1-8, 2021.

[6] M. Mazurek, D. Strzałka, A. Wolny Dominiak and M. Woodbury Smith, "Electronic Health Record Breaches as Social Indicators", *Social Indicators Research*, Vol. 141, pp. 861-871, 2019.

[7] S. Argaw, N. Bempong-Ahun, B. Eshaya-Chauvin and A. Flahault, "The State of Research on Cyberattacks Against Hospitals and Available Best Practice Recommendations: A Scoping Review", *BMC Medical Informatics and Decision Making*, Vol. 19, No. 1, 2019.

[8] Q. Yao, X. Han, X.K. Ma, Y.F. Xue, Y.J. Chen and J.S. Li, "Cloud based Hospital Information System as a Service for Grassroots Healthcare Institutions", *Journal of Medical Systems*, Vol. 38, pp. 1-7, 2014.

[9] M. Reisman, "EHRs: The Challenge of Making Electronic Data Usable and Interoperable", *Journal for Formulary Management*, Vol. 42, pp. 572-575, 2017.

[10] N. Menachemi and T.H. Collum, "Benefits and Drawbacks of Electronic Health Record Systems", *Risk Management and Healthcare Policy*, Vol. 45, pp. 47-55, 2011.

[11] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System", *Decen Tralized Business Review*, pp. 1-9, 2008.

[12] D. Cosset, "Blockchain: What is in a Block?", Available at https://dev.to/damcosset/ blockchain-what-is-in-a-block-48jo, Accessed in 2017.

[13] A.A. Monrat, O. Schelen and K. Andersson, "A Survey of Blockchain from the Perspectives of Applications, Challenges and Opportunities", *IEEE Access*, Vol. 7, pp. 117134-117151, 2019.

[14] V. Buterin, "A Next-Generation Smart Contract and Decentralized Application Platform", Available at https://github.com/ethereum/wiki/wiki/White-Paper, Accessed in 2014.

[15] N. Szabo, "Smart Contracts: Building Blocks for Digital Markets", *Extropy: The Journal of Transhumanist Thought*, Vol. 18, No. 2, pp. 28-33, 1996.

[16] E.Y. Daraghmi, Y.A. Daraghmi and S.M. Yuan, "Medchain: A Design of a Blockchain-based System for Medical Records Access and Permissions Management", *IEEE Access*, Vol. 7, pp. 164 595-164 613, 2019.

[17] K. Shuaib, J. Abdella, F. Sallabi and M.A. Serhani, "Secure Decen Tralized Electronic Health Records Sharing System based on Blockchains", *Journal of King Saud University-Computer and Information Sciences*, Vol. 34, No. 8, pp. 5045-5058, 2022.

[18] S. Rouhani, L. Butterworth, A.D. Simmons, D.G. Humphery and R. Deters, "Medichain TM: a Secure Decentralized Medical Data Asset Management System", *Proceedings of International Conference on Internet of Things and IEEE Green Computing and Communications and IEEE Cyber, Physical and Social Computing and IEEE Smart Data*, pp. 1533-1538, 2018.

[19] A. Azaria, A. Ekblaw, T. Vieira and A. Lippman, "Medrec: using Blockchain for Medical Data Access and Permission Management", *Proceedings of International Conference on Open and Big Data*, pp. 25-30, 2016.

[20] A. Shahnaz, U. Qamar and A. Khalid, "Using Blockchain for Electronic Health Records", *IEEE Access*, Vol. 7, pp. 147782-147795, 2019.

[21] S. Bragagnolo, H. Rocha, M. Denker and S. Ducasse, "Ethereum Query Language", Proceedings of IEEE/ACM International Workshop on Emerging Trends in Software Engineering for Blockchain, pp. 1-8, 2018.

[22] C. Xu, C. Zhang and J. Xu, "Vchain: Enabling Verifiable Boolean Range Queries Over Blockchain Databases", *Proceedings of International Conference on Management of Data Association for Computing Machinery*, pp. 141-158, 2019.

[23] S. Linoy, H. Mahdikhani, S. Ray, R. Lu, N. Stakhanova and A. Ghor-bani, "Scalable Privacy-Preserving Query Processing Over Ethereum Blockchain", *Proceedings of International Conference on Blockchain*, pp. 398-404, 2019.

[24] D. Przytarski, C. Stach, C. Gritti and B. Mitschang, "Query Processing in Blockchain Systems: Current State and Future Challenges", *Future Internet*, Vol. 14, No. 1, pp. 1-6, 2021.

[25] F.A. Pratama and K. Mutijarsa, "Query Support for Data Processing and Analysis on Ethereum Blockchain", *Proceedings of International Symposium on Electronics and Smart Devices*, pp. 1-5, 2018.

[26] T. McConaghy, M. O'Conner, R. Sarry, N. Bennet and E. Marvici, "BigchainDB 2.0 the Blockchain Database", Available at https://www.bigchaindb.com/whitepaper/bigchaindb-whitepaper.pdf, Accessed in 2018.

[27] M. El-Hindi, C. Binnig, A. Arasu, D. Kossmann and R. Ramamurthy, "BlockchainDB: A Shared Database on Blockchains", *Proceedings of International Conference on VLDB Endowment*, Vol. 12, pp. 1597-1609, 2019.

[28] Y. Li, K. Zheng, Y. Yan, Q. Liu and X. Zhou, "EtherQL a Query Layer for Blockchain System", *Proceedings of International Conference on Database Systems for Advanced Applications*, 2017, pp. 556-567, 2017.

[29] J. Han, H. Kim, H. Eom, J. Coignard, K. Wu and Y. Son, "Enabling SQL Query Processing for Ethereum-based Blockchain Systems", *Proceedings of International Conference on Web Intelligence, Mining and Semantics*, pp. 1-7, 2019.

[30] J. Kaur, R. Rani and N. Kalra, "Blockchain-based Framework for Secured Storage Sharing and Querying of Electronic Healthcare Records", *Concurrency and Computation: Practice and Experience*, Vol. 33, No. 20, 2021.

[31] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman and Y. Manevich, "Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains", *Proceedings of 13th Conference on EuroSys*, pp. 1-15, 2018.

[32] Z. Peng, H. Wu, B. Xiao and S. Guo, "VQL: Providing Query Efficiency and Data Authenticity in Blockchain Systems", *Proceedings of International Conference on Data Engineering*, pp. 1-6, 2019.

[33] GitHub repository: https://github.com/Tamang-Suvam/QueryBCCode/tree/master/finaldApp