# Blockchain on Relational and NoSQL Databases

Vedant Chauhan (892758), Siddharth Malhotra (934336), and Saksham Agrawal (866102)

**Abstract**—Blockchain is a novel technique that chain blocks together as a public ledger on the nodes of peer-to-peer network. It is a tamper-resistance technique which technically is a database with no query capabilities. If blockchain is measured with database, it has low throughput, latency, and capacity. Traditional databases are centralized with greater capacity, higher throughput, and sub-second latency. This paper explores further about the underlying storage requirements associated with blockchain while exploring the current ecosystem of databases and form a comparative review of what would be the best architecture suited for blockchain.

**Index Terms**—Blockchain, NoSQL, Relational Databases

## 1 Introduction

As we know that data is increasing day by day exponentially, it is tough to maintain the data and retrieve it from the large server as per user requirement. Due to this large data, decentralized, peer-to-peer is improving as a viable option. The technique introduced is known as Blockchain [1]. This technique is more than just cryptocurrency. Blockchain is more efficient in terms of security, retrieving and storage of data.

Database is broadly divided into the centralized and decentralized database. In the centralized database, the database is physically in one location and users can use them using internet connections. These databases are mainly structured database which has a predefined schema. Banks prefer to keep a centralized database. It is maintained by a database administrator only. It can be accessed by different a communication network such as LAN, WAN, and Terminals provided by the authorized person. It has various advantages such as high reliability and availability, modular growth, lower communication cost and faster response. However, it comes with some disadvantages like the cost of software and complexity, overhead complexity and data integrity. A relational database is an example of a centralized database.

On the other hand, a decentralized database is a single logical database that is spread physically across computers in various locations that are connected by a data communication link. In this most of the processing take place on local device which requires local ownership of data and data sharing is also required. NoSQL [4] and google's DBMS called Bigtable are examples of the decentralized database. It has various advantages such as it minimizes communication which results in a decrease in cost, provides local control over data and is scalable with the incoming of data. However, it also has disadvantages such as software cost and complexity, overhead processing and data integrity as every physical location has to manage their security.

This technique can use either centralized or decentralized database. However, there are many disadvantages of using a centralized database with blockchain than using a decentralized database with blockchain which will be mentioned in the following sections.

The rest of the paper is divided into following sections. Section 2 explains background of technologies used in the paper. Section 3 includes relation between blockchain and database. Section 4 and 5 describes relation of blockchain with relational and NoSQL, respectively. Section 6 concludes the paper.

## 2 Background

### 2.1 Blockchain

A Blockchain is a technique which is a decentralized to store data in a completely new manner either using centralized or decentralized database. It comprises mainly three parts **Blocks**, **Chain** and **Network**.

**Block** is a list of transactions recorded in the ledger over a given period. The size, period and triggering of the block varies with every blockchain.

**Chain** helps in linking one block to another block like a linked list by calculating hash function mathematically. This is the most difficult concept in blockchain to understand. SHA-256 is the most common algorithm used in the blockchain.

The **network** is composed of all the nodes. Each node has a record of all the transactions that occurred in blockchain. The nodes are located all over the world and the whole chain can be operated by anyone from anywhere in the world.
**Working of blockchain** [2] starts when someone requests the transaction, it is then broadcast to a network which is Peer-to-Peer consisting of computer known as nodes. The block contains all the data such as cryptocurrency, contracts, records and other information. Then the system verifies the user using the algorithm. After this verification, the block is then added to the existing blockchain such that it is permanent and unalterable. Your transaction is completed now. Every transaction gets updated in the nodes all over the world.

### 2.1.1 Advantages

a. **Disintermediation**: Database is a tangible thing which stores in memory and disk of computers. These data can be altered or corrupted if an unauthorized person gets the access. Due to this reason companies asked the third-party to take care of the security which attracts a great amount of time and money. However, in the blockchain, we can replace the third-party security system by clever cryptography.
b. **Empowered users**: Blockchain allows the user to control all their data and transactions.
c. **Transparency and Immutability**: Changes are publically viewable in blockchains and transactions cannot be altered or deleted.
d. **Ecosystem Simplification**: As all the transactions are in a single ledger, the complication we encounter and the maintenance get reduce from using multiple ledgers.
e. **Lower transaction cost**: In order to avoid transactions fees and charges for exchanging
f. the assets third-party is replaced by blockchain mechanism.
g. **Faster transaction**: While doing interbank transactions, transactions usually take a day to successfully complete which is resolved here as any transaction on blockchain can be done in minutes without any restriction as you can perform transaction any time of the day.

### 2.1.2 Disadvantages

a. **Performance**: Blockchain does all the similar things like a regular database except query capabilities, however, with additional three burdens which are
   Signature verification: Every transaction must be digitally signed using a public-private cryptography.
   Consensus Mechanism: This mechanism does lots of computational work.
   Redundancy: Transaction is processed in every node of the chain.
b. **Cost**: Initial investment is high in blockchain.
c. **Updates**: Updates to decentralized applications can be quite challenging as all the distributed peers on the network providing services need to be updated.

### 2.1.3 Concepts used in blockchain

In a centralized database system, all the decisions are taken by a leader however, blockchain is a decentralized database system due to which it needs a different mechanism to take the decision so that they can come to a consensus that mechanism is called "Consensus Mechanism". The importance of this mechanism is to make sure that the block belongs to particular blockchain or not. This mechanism can be achieved by two concepts.

a. Proof-of-work [5]: Miners solve the cryptic hash to mine the block, which is eventually added in blockchain. It requires massive computation. This work should be unbreakable. After this, blocks are verified. If it is verification is successful then the block will be added into the chain.
b. Proof-of-state: This concept will help in making the whole mining process virtual. Also, it will replace the miners with validators. The validators are supposed to keep some of their coins as a collateral deposit. After that, they will start validating the blocks. This means that after they discover a new block which can be added to the chain, then they have to validate it by betting on that block. If the block gets appended, then the validators will get reward proportionate to their bets.

## 2.2 Relational Databases

A relational database [3] is based on the relational model of data which digitally implemented. This database is easily manageable as it is structured. This database is controlled by a single entity rather than opened for everyone. This database is an unencrypted database as the responsibility of this database is of the centralized authority. It is built on a single server. So it is easy to avoid the mistake of duplicating data. It is easy to manipulate. A relational database is made up of rows and column. In this database, it is mandatory to define the schema before implementation as adding new columns to the same table is difficult in future. Each table can be updated without disturbing the others. It's quite simple to extend the database by adding new data in the new row of the table. Relational database mainly works on operations such as union, intersection, difference, Cartesian product, joins and others. It also has some constraints which help in retrieving data as required such as primary key, foreign key, stored procedures, and index. All the tables are connected to each other through the foreign key. If you need to fetch data from more than one table then use joins to fetch them. This property makes relational database more complex. As it works on a single server because of which it is difficult to extend it after a limit also the costing server makes this database more costly to upgrade.

## 2.3 NoSQL Databases

Now-a-days developers are working on many different types of applications which creates a large volume of data and data types which are changing rapidly such as structured, semi-structured, unstructured and polymorphic data. Users are in demand of always-on services, accessible from any device and can accommodate data of millions of user across globally. Also, organizations start using commodity server and cloud computing instead of large monolithic servers and storage infrastructure. To fulfill all the above demands NoSQL comes in the picture. NoSQL databases are scalable. Also, they provide superior performance over a relational database. The data model of NoSQL covers many issues that the relational model does not design to address. There are three types of data models:

a. **Document Model**: Instead of storing data in rows and column, it allows to store data in documents. These documents contain one or more fields, where each field contains a typed value such as string, date, binary, decimal value, or array. It also does not allow to use joins and a foreign key to avoiding complexity instead of that it uses a single hierarchical document to store records and related data. Every document can contain different fields which makes them dynamic schema.
b. **Graph Model**: This model uses graph structure with nodes and edges and different properties to represent the data. The main advantage of this model is that it makes it easier to model and navigate relationships between entities in an application.
c. **Key-Value and Wide column Model**: In this model, every item in the database is stored either as an attribute name or key, with its value. This model is very useful for representing unstructured and polymorphic data, this is because the database does not enforce a set schema across key-value pairs. Also, data can be queried by the key only.

## 2.4 Relational Database v/s NoSQL Database

| | *Relational Database* | *NoSQL Database* |
|---|---|---|
| *Data Models* | Schema is important to define before entering data in the system | Allows building application without defining the schema which makes easier to update with the change in requirements. |
| *Data Structure* | Relational database was built in an era where data was fairly structured and clearly defined by their relationships. | NoSQL is defined to handle unstructured data such as text, social media posts, video or emails. |
| *Scaling* | Relational database requires | It is much cheaper to scale a NoSQL |

| | | |
|---|---|---|
| | a single server to host the entire database. To scale, you need to buy a bigger server. | database than the relational database because you can add capacity by scaling out over cheap servers. |
| *Development Model* | The relational database is a closed source with fees. | NoSQL database is open source |

Table 2.4.1 Comparison between two different databases

## 3 BLOCKCHAIN V/S DATABASE

### 3.1 Comprehensive review

The regular databases are centralized and use a client server architecture. Clearly, there is a fundamental difference between blockchain v/s the regular databases. Blockchain may well be related as a distributed database, which have the objective of splitting larger problems into sets of smaller problems, to solve them effectively. Later, we shall see in the where does blockchain fit into the entire ecosystem of databases figuratively.

1. Broadly, distributed databases may be classified in a master-slave format or mult-master format. In the master-slave format, the master serves as the single point of failure or the main bottleneck. In multi master information is copied into other nodes but the problem is double replication can happen in this. Due to the inherent properties of blockchain, problems such as double spending and single point of bottleneck are avoided.
2. The design of these databases and replication across several nodes, allows the following advantages: better reliability and availiability, imporved performance and throughput, easier expansion.
3. The idea of smart contracts is another difference when compared to blockchain and databases, as the blockchain usually run their own virtual machine, it is possible to run the smart contracts over them. The smart contracts can be compared as stored procedures in the regular blockchain.
4. Blockchain allow permanent, immutable recordkeeping and are much slower than data stores designed to handle and distribute more perishable data. In terms of operations, blockchains are write heavy and only work with Insert operations.
5. In contrast, the traditional databases are said to perform CURD operations. The architecture of a traditional database may be master-slave or mult-master.
6. While in blockchain, a full replication of block is performed on every node. A consensus mechanism such as a 2-phase commit (distributed transaction) can be found on a traditional database, however, in a blockchain, majority of the peers have to agree on the outcome of the transactions.

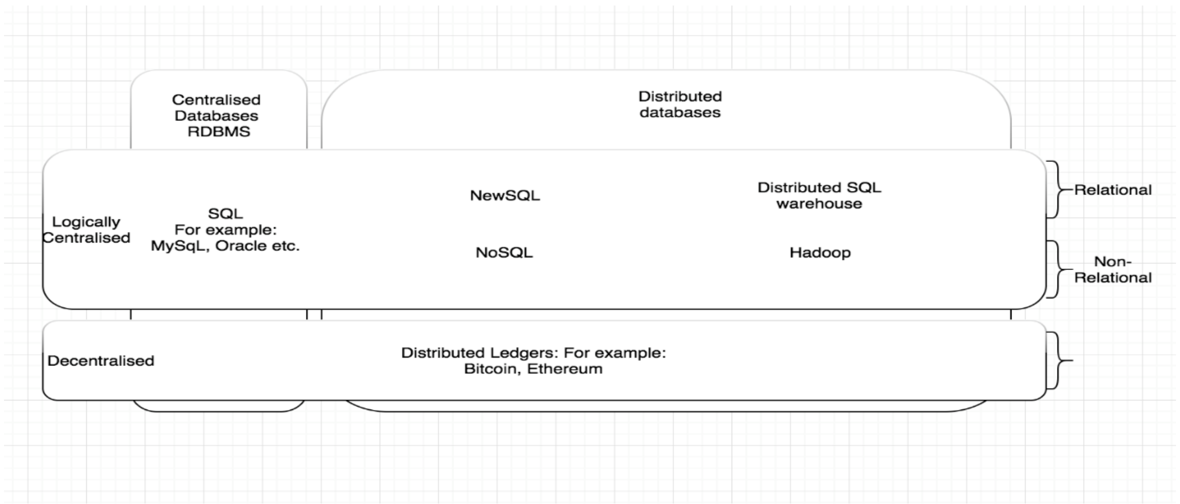
Centralised Databases RDBMS
Distributed databases
Logically Centralised
SQL For example: MySqL, Oracle etc.
NewSQL
Distributed SQL warehouse
NoSQL
Hadoop
Relational
Non-Relational
Decentralised
Distributed Ledgers: For example: Bitcoin, Ethereum

Fig. 3.1.1 Blockchain and database

## 3.2 NOT ACID, NOT BASE, but SALT: A Transaction Processing Perspective on Blockchain

Traditional RDBMS databases are built on the philosophy of ACID properties, i.e. Atomicity, Consistency, Isolation and Durability.

- A transaction consisting of multiple operations is conducted as all or nothing.
- Each transaction transforms the data from one consistent state to another, which adheres to the database integrity constraints.
- Concurrent transactions maintain isolation.
- Once the transaction has been committed the results become permanent.

There is a reliance on transaction managers, to perform various tasks- such as coordination between resource managers, 2 phase commit protocol. Isolation requires protocol such as 2 phase locking and durability is left on the hands of database.

With the rise of modern Web applications, an alternative philoshpophy came into picture, namely BASE i.e, Basically avilalable, Soft state and Eventually Consistent.

- The system should be available in partial failures without causing a complete failure.
- The state may be classified as 'soft' even though there may be no further updates.
- If no new updates are being made, the system will eventually become consistent.

Further, the idea of sharding was introduced which worked basically on top of peer-to-peer systems where all peers where equally responsible for subset of the overall data. There is no transaction manager in BASE systems. Based on a distributed hashtable client request is handled. BASE systems use optimistic replication thus sometimes comprising on consistency. Overall the aviliability is high in BASE systems as data can be read and written by multiple clients at the same time. The clients interact with a load balancer or resource managing unit. Thus, BASE systems induce scaling to allow large number of clients to interact and meet large demands.

Blockchains, as we have previously discussed, allow a trust-based system and are based on absence of a central authority. Transactions in blockchain may be classified as Squential, Agreed, Ledgered and Tamper-resistant.

- All transactions are processes sequentially. There is no parallel-processing of transactions as in ACID.
- The majority of the nodes have to agree on the transaction for it to be classified as valid. This brings in the use of consencus and removes the need for central authority.
- All the agreed transactions are added to the ledger and cannot be revoked.
- Transactions cannot be manipulated and are tamper-free. No user/block can change the sequence of pending transactions. Furhtermore, since all nodes keep a history of the transaction, the other nodes will never agree to any tampered changes.

The systems using blockchain maybe further defined as symmetric, admin-free, ledgred and time-consensual.

Overall, the system is simpler than ACID as there is no grouping of multiple operations into an atomic piece of work for the client. There is however, a way to create stored procedures, for instance, using smart contracts, which would relate to the atomic property of ACID. The isolation property is quite similar to ACID that there is a well-defined order consistent with the network. In ACID cetain parties check for transaction integrity and violations, where in blockchain all parties have to acknowledge. Like BASE, two nodes can be mining different blocks and hence maybe found in a conflicting state. In BASE, the systems are guareenteed to converge when there are no pending transactions, whereas in SALT based blockchain exacltly one block is to be agreed upon by all the peers. Persistancy is common in both ACID and BASE when speaking of durability, the blockchain however rely on a probabilistic model. The users wait for a certain number of blocks until they consider the transaction to have been completed.

## 4 Blockchain with Relational Databases

### 4.1 Relation between blockchain and relational database

As we know traditional databases are set up in a client-server network architecture. The control of the database remains in the hands of the admininstrators, allowing access and permissions to the centralized servers. The data is modified can thus be altered by the administrators at any point. It is in direct contrast to the blockchain which constitute a set of decentralized nodes, wherein all nodes can verify data that is recorded on the blockchain.

Blockchain databases have a fault-tolerant and robust way of storing critical data. Relational databases have a huge advantage when looking at the performance. In addition, in relational databases data can be easily modified or deleted. Blockchain are considered as distributed ledgers, i.e. distributed databases which uses cryptography for concurrency control and consensus in an environment which is compromised. Blockchain provides byzantine fault tolerance, pseudo-anonymity, immutable, accountability and non-repudiation at transaction level.

We shall now explore the possible advantages of integrating a relational database with blockchain.

1. Memory management may become a problem as the blockchain scales. As the size of the network scales storing history of all the transactions taking place becomes a challenge.
2. Blockchain are usually designed as a Merkle tree, thus allowing effiecient and secure verification of content in a large body of data. It also helps in the content and consistency of the data. It summarizes all the transactions that occurred in a block, thus making it easier for the user to verify whether the transaction was included in the block. This design structure may work better by subtituing it with relational databases thus avoiding complexity and navigation of a particular pathway.
3. Impovement in access time of data as relational databases are faster to query and do not need a supplementary database for holding a set of indicies for lookups.
4. Blockchain are good in establishing trust and provenance. Database are meant for current state of the object, thus making it possible to delete the history. Hence, trust and immutability can be associated with blockchain, whereas, confidentiality and performance may be best associated with databases.

### 4.2 Techniques

We shall now consider a study wherein blockchain is associated with a relational database. The idea here is to replace the existing architecture of Bitcoin- a blockchain based cryptocurrency with SQLite RDBMS database and try to understand **Why RDBMS may be a better fit with blockchain?** Secondly, **How RDBMS can be changed to integrate with blockchain?**

1. Reducing Disk Storage with SQLite into Bitcoin Architecture [9]
   a.) The objective here would be, to improve memory management and access time in the bitcoin protocol using SQLite RDBMS database supporting SQL queries. It also explores the current bitcoin architecture. Flat file/Level DB are the primary DBs which store and retrieve information. The databases under Bitcoin are Berkeley Database Engine, Flat file database and LevelDB.
   b.) The Berkeley DB keeps history of all wallets that are used in the protocol. It is a high performance embedded database engine of key/value produced by Oracle. It contains vital information such as key, name, address, account balanaces and more. The flat file is a set of blocks containing transactions and are written into file in a raw database format. LevelDB is a persistant key/value store which puts index values which are used for accessing information on a file.
   c.) The block in database can thus be descrived as its Index value (from LevelDB) and the data which is stored in serialized format.
   d.) SQLite in Bitcoin: SQLite implements a SQL database engine which is serverless, self contained, zero-configuration, and transactional. It can be run in minimal stack space (4KiB) and very little heap (100KiB). Applications interact with the SQLite library

through function calls this making it faster than inter process communication.

e.) For each node to have the entire history of transactions, the usage of LevelDB comes handy interms to indicies and faster retrievals. However, with the advantages relational databases have to offer, the need for additional disk space is eliminated. The access time can be reduced by reducing the module of serialization/deserialization.

f.) When designing the database keep in mind the main goal, which is to store and retrieve blocks. The block contains a header and some transactions. The header contains information such as the version number (nVersion), previous block reference (PrevHashBlock), timestamp i.e. time at which the transaction is recorded by a computer, root of the Merkle Tree (MerkleHashRoot) and a nounce (nNounce).

g.) The blocks have input and output to and from the database by parsing SQL statements.

h.) The creation involves writing scripts for creation of the database file and its handler, this handler is further passed to functions. Wrapper methods to use Select, Insert, Update and Delete are built.

i.) Thus, the idea is to store data in a structured format making it easier to find a sat of data/record from multiple set of records. Creating ad-hoc queries becomes easier with such a model. Also, it becomes easier to perform operations on a large set of data.

j.) Through this, there is a large decrease in the block size (over 90%), which can be attricuted to the removal of redundant data during serialization, removal of indicies used to search block.

**4.3 Comparison:**

Following comparison shows the relation of blockchain and Relational databse mentioned in this section:

| | *Blockchain* | *Relational Database* | *Blockchain with Relational* |
|---|---|---|---|
| *Immutability* | O | | O |
| *No Central Authority* | O | | O |
| *Assets over Network* | O | | O |
| *High Throughput* | | O | O |
| *Low Latency* | | O | O |
| *High Capacity* | | O (Expensive) | O (Expensive) |
| *Schema less* | | | |
| *Integrated Caching* | | | |

Table 5.3.2 Comparison between blockchain and Relational Databases

## 5 Blockchain with NoSQL Databases

### 5.1 Relation between blockchain and NoSQL

In earlier sections, we had detailed discussion about blockchain and NoSQL. But, there are few questions to answer before blockchain with NoSQL techniques provide better understanding about the relation of blockchain and NoSQL databases. First, **Why NoSQL is a better fit with blockchain?** Second, **How NoSQL can be changed to integrate with blockchain?**

First question can be answered with NoSQL advantages that elevate when integrated with blockchain.

1. Schema gives structure to data. Predefined templates can help in faster development. Traditional databases use this generic approach to handle data. Due to this predefined structure additional scenarios are hard to accommodate and permissioning is affected. But, NoSQL is schema less. It means it has a schema which can be key-pair like JSON, document, and graph. It usually helps in type of data that can be stored by blockchain.
2. Blockchain is a ledger which stores contracts, certificates, numbers, facts, etc. all categorized mostly towards unstructured data. NoSQL supports unstructured data, i.e. text-heavy, facts, numbers, etc. These type of data don't have a predefined structure. Traditional databases are not comfortable with such data. As, they focus more on systematic structure with rows and columns.
3. In modern applications, scaling and agility are some common challenges. Relational databases fail to cope with these challenges. Relational databases use scale-up architecture which increases load on the machine and makes adding servers expensive for the organization. But, NoSQL solve this problem of scaling. As the architecture of NoSQL allows horizontal scaling, i.e. sharding. Documents of the database are objects. These objects are key-pairs unlike rows and columns of relational. Therefore, they can be placed in part or complete on different nodes easily. This helps in handling the load easily and inexpensive.
4. In any database, querying is one of the fastest ways to know the expected result. One of the problem with blockchain is, it does not have query capabilities. NoSQL and relational databases provide querying capabilities, but NoSQL has huge processing power to iterate the results and produce the result instantly, however large is the database size.

Second question can be answered by tackling blockchain requirements and how those requirements can be framed using NoSQL.

1. One of the main features of blockchain is decentralization. Multiple nodes (anyone can become a miner) are interconnected with each other over peer-to-peer network performing proof-of-work to make a valid chain. Blockchain with NoSQL can provide voting permission to nodes on p2p network and operate over database's consensus.
2. Another feature is tamper-resistance of blockchain. Once an information is stored in blockchain, it can't be changed. Databases are not immutable. But, can be transformed in order to perform the activity. One such way is that sequence of blocks in the form of documents in database containing order of transactions are ordered. Each hash of the block is derived from data, transaction and previous hash, forming a chain of blocks.
3. Scalable capacity is important as chain is constantly growing after a transaction is performed by a miner. NoSQL as mentioned above, has more scaling power. Nodes can be easily added to increase the capacity. Hence, certificates and legal contracts will directly stored on blockchain database.
4. High performance is needed to mine the block and store the contents before any failure occurs. NoSQL already has high throughput and low latency which can help in optimization and computation of the process. Once these advantages integrates with blockchain, blockchain's own throughput and latency in comparison to database elevates and provides fully-featured NoSQL query capability.

Both questions are significantly needed to be answered in order to check the relationship of blockchain with NoSQL databases. The questions gives a starting point about current understanding of the concept (NoSQL as a database for blockchain) and how the concept can be improved to enhance the needs of current blockchain technique.

## 5.2 Techniques

Blockchain with NoSQL Database is explored with the research in the area. The techniques provide how database is mined with blockchain to improve the current needs of the system. Initial technique emphasis on database with blockchain while other technique builds on top of it to improve security of the system. Following techniques provides clear understanding:

1. BigchainDB: A Scalable Blockchain Database [6]
   - a.) It combines both blockchain and database resulting in database style decentralized storage. Decentralized control is obtained via cluster of nodes with voting permissions. This leads to a peer-to-peer network where permissioning is operated above database layer. This permissioning open doors for private blockchain databases to link with public blockchain databases.
   - b.) Along with permission query capabilities to question the database is must for a database. BigChainDB provides the NoQL query capability to query the database what blockchain was lacking.
   - c.) Scalability is an issue with traditional databases. BigchainDB emphasis on scaling. It combines NoSQL scaling advantages with blockchain to store certificates, legal contracts, and transactions on the database. It focuses on fractional replication, i.e. storing part of data on each node instead all data on one. This gives better scalable options.
   - d.) Blockchain immutability is handled by sequence of blocks maintaining sequence of transactions. This ordering of blocks helps in validating the chain. If any unauthorized transactions is conducted, hash of the blocks will change and chain will be declared unvalidate. This operation is directly taken from the blockchain.
   - e.) Creation and transferring of assets are improved with additional security implementation. Only entities which have permission can access the assets, any other compromised entity cannot change the data. This is called no single point-of-failure.
   - f.) If number of nodes are increased in the chain, throughput increases to 1 million writes per second, and sub-second latency leading to petabytes of storage.
   - g.) Due to all the above features BigchainDB can be applied to many use cases: directly storing legal contracts and certificates on blockchain database, supply chain management by tracking and creating high volume asset thereby reducing cost, fraud, and latency, intellectual property tracking, improving database reliability with no single point-of-failure, time stamping legal contracts reducing legal conflicts, and offers certificate of authenticity for security purposes.
   - h.) With so many advantages BigchainDB is certainly a viable technique for combining blockchain and NoSQL databases. It presents usually all the advantages of blockchain and merges with the NoSQL database. The focal point was to improve the performance. But, this leads to some security compromises. One such disadvantage is to make concession on data integrity. If large number of miners are malicious, then consistency and accuracy of the system becomes critical. As, all the malicious miners will try to make the chain invalid by creating fake transactions. Normal miner will not be able to distinguish the valid transaction. Hence, next technique comes into picture which solves this major problem.
2. Blockchain based database to ensure data integrity [7]
   - a.) Main focus is on data integrity in cloud computing environments. Due to low throughput, weak stability, and high latency of blockchain data integrity is compromised. As mentioned above this problem is solved by using two-layer blockchain based database as shown in the figure below.
   - b.) First layer is database interface alongside blockchain. This layer stores every operations on the database. Due to the quick and reliable storage throughput is increased and latency is reduced. Each permissioned miner when executes the transaction stores the evidence on DB replicas. Private and public key pairs are used as a digital signatures of the

messages. This layer has good performance but lacks data integrity due to no proof-of-work feature.

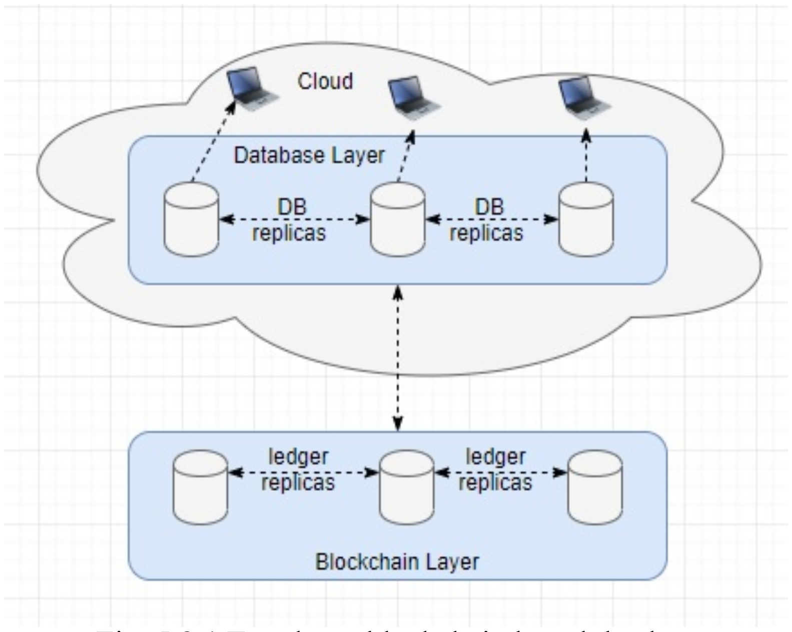

Fig. 5.2.1 Two-layer blockchain based database

c.) Second layer solves the problem of first layer. It is a proof-of-work based blockchain that stores part of database operation from first layer. Part of operation of first layer, i.e. hash of current operation are send to second layer for storage making it immutable. These hashes ensures the validity of the data stored in blockchain. This layer provides data integrity with weak performance.

d.) Hence, both the layers play the crucial part of improving performance and ensuring data integrity by communicating with each other.

e.) Data integrity provided by the technique is strong as Bitcoin's, it takes approximately infinite time for the attacker to break the chain.

f.) Bitcoin and Ethereum are permissionless blockchains. In these chains anybody can become a miner. These blockchains are adequate for the second layer of the model. Due to market fluctuations and mining mechanisms stability can be questioned. But, the stability is depended on application context. In this application, data integrity with performance is considered using database mainly non-relational database. Hence, can be a viable option.

g.) This technique includes some disadvantages. When the operation happens in the first layer, all members are made aware that database operation is occurring. It is notification which malicious user can also get access.

h.) Private keys can also be stealed by attackers in first layer while digital signature are applied in the first layer. The miner needs to be reactive while signing the messages.

i.) Collusion attack aimed to change the information stored in first layer which was approved by all members makes the operation authenticated. If a valid member finds that the claim was wrong, then nothing can be done because in blockchain longest chain is the correct chain. Hence, this technique solves some problems but affected by problem of its own.

## 5.3 Comparison

Comparison between two techniques and overall relation between blockchain and NoSQL database is the ideal way to verify the claims made in this section.

Table 5.3.1 shows the comparison between BigchainDB and Blockchain based database. The table shows the current dynamics of both techniques.

| | *BigchainDB* | *Blockchain based database ensuring data integrity* |
|---|---|---|
| *Layers* | One- Both for blockchain and database | Two- One for database related operations and another for blockchain proof-of-work |
| *Description* | Database style decentralized storage | Data integrity in cloud environments |
| *Goal* | Emphasis on scale | Ensure data integrity |
| *Concept* | Practical | Theoretical (prototype in work) |
| *Performance* | High throughput and sub-second latency | Throughput and latency are improved (implementation in work) |
| *Capacity* | Petabytes of storage leads to increased capacity | Not mentioned |
| *Security* | As emphasis is more towards performance and scale, it suffers from data integrity problems | Solves data integrity problems but suffers from collusion attacks |

Table 5.3.1 Comparison between two mentioned techniques

The techniques have both advantages and disadvantages. BigchainDB provides the relation between blockchain and NoSQL databases, and blockchain based database builds on top of it to provide data integrity. From both techniques, BigchainDB is the technique which answers the goal we have mentioned in the abstract of this paper. This technique revamps the use of NoSQL database in blockchain and makes NoSQL a better contender for blockchain designs. Following comparison shows the relation of blockchain and NoSQL mentioned many times in this section and proves NoSQL is a better fit:

| | *Blockchain* | *NoSQL* | *Blockchain with NoSQL* |
|---|---|---|---|
| *Immutability* | O | | O |
| *No Central Authority* | O | | O |
| *Assets over Network* | O | | O |
| *High Throughput* | | O | O |
| *Low Latency* | | O | O |
| *High Capacity* | | O | O |

| | | |
|---|---|---|
| *Schema less* | O | O |
| *Integrated Caching* | O | O |

Table 5.3.3 Comparison between blockchain and NoSQL

Table 5.3.3 shows the merits of blockchain, NoSQL, and their combination. Blockchain is immutable and decentralized, but has low throughput, latency, and capacity compared to NoSQL database. NoSQL on the other hand is schema less and supports integrated caching, i.e. in-memory storage which is better compared to relational database. As, BigchainDB suggested blockchain with NoSQL combines merits of both and provides a stable and efficient system with decentralization, tamper-resistance, high throughput, capacity, and query capabilities.

## 6 Conclusion

Collection of data and processing that data is constantly being refurbished with different technologies. With blockchain, there is security and immutability factor added while storing the data. Due to constant increase in the data and individual needs, just maintaining a public ledger with instability is not enough. Databases are always integral to store petabytes of data. In this paper we focus on relational and NoSQL databases to elevate current blockchain technique. Relational databases are schema-based centralized structure which provides high performance, but lacks in scaling, integrated caching, etc. NoSQL on the other hand is scalable, powerful and schema-less architecture. After comparing respective techniques based on these databases and blockchain, NoSQL provides the edge. It is debatable that which database technique is better with blockchain, but our research gives NoSQL an advantage. It is cost-effective to use NoSQL with blockchain and distributes multiple applications like supply chain, intellectual property, cryptocurrencies, legal contracts and certificates.

## References

[1] Blockchain. https://en.wikipedia.org/wiki/Blockchain

[2] Blockchain working. https://medium.com/@micheledaliessi/how-does-the-blockchain-work-98c8cd01d2ae

[3] Relational databases. https://en.wikipedia.org/wiki/Relational_database

[4] NoSQL databases. https://www.mongodb.com/nosql-explained

[5] Proof-of-work. https://cointelegraph.com/explained/proof-of-work-explained

[6] McConaghy, Trent, Rodolphe Marques, Andreas Müller, Dimitri De Jonghe, Troy McConaghy, Greg McMullen, Ryan Henderson, Sylvain Bellemare, and Alberto Granzotto. "BigchainDB: a scalable blockchain database." *white paper, BigChainDB* (2016)

[7] Gaetani, Edoardo, Leonardo Aniello, Roberto Baldoni, Federico Lombardi, Andrea Margheri, and Vladimiro Sassone. "Blockchain-based database to ensure data integrity in cloud computing environments." (2017)

[8] Peters, G.W. and Panayi, E., 2016. Understanding modern banking ledgers through blockchain technologies: Future of transaction processing and smart contracts on the internet of money. In Banking Beyond Banks and Money (pp. 239-278). Springer, Cham

[9] Mbinkeu, R.C.N. and Batchakui, B., 2015. Reducing Disk Storage with SQLite into BitCoin Architecture. International Journal of Recent Contributions from Engineering, Science & IT (iJES), 3(2), pp.10-14

[10] Tai, S., Eberhardt, J. and Klems, M., Not ACID, not BASE, but SALT-a transaction processing perspective on blockchains. In *Proceedings of the 7th International Conference on Cloud Computing and Services Science, CLOSER* (Vol. 1, pp. 755-764)