Database Management in Blockchain

Under the guidance of
Professor Jhumma Dutta

Project Members:-
Arnab Dutta
Himanshu Kumar Prasad
Abdul Kuddus

Github Repo: https://github.com/slow-codex/blockchaindb

Project Idea in Brief

- Introducing **Blockchain Technology** to modern **Database Management Systems**.
- Made a blockchain using a **Local Database** created **hash** and **publicly accessible id**.
- Implemented the feature to add, read, find, update and delete the database via blockchain.
- Used OrbitDB and ipfs and implemented the solution in Javascript.

What is DBMS?

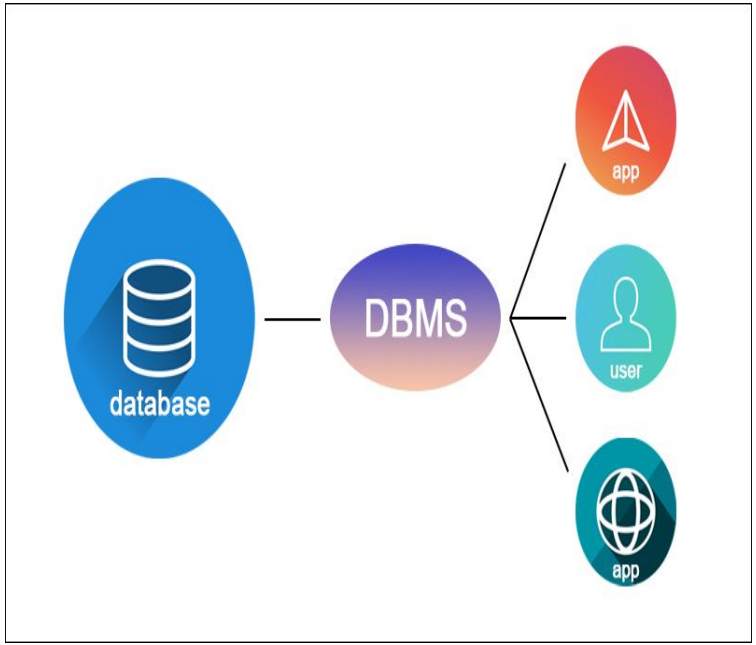
DBMS stands for Database Management System.

It is a software for storing and retrieving user data while considering appropriate security measures.

It consists of groups of programs that **manipulate** the database.

It allow user to create their own database as per their requirement.

It provides an interface between the data and the software application.



Drawbacks of Normal Database

- **Single Point of Failure**: A centralized database has a single point of failure. If the server goes down or is compromised, the entire database becomes inaccessible.
- **Limited Scalability**: Centralized databases can be difficult to scale as the size of the data and number of users increase because of limited Capacity.
- **Higher Security Risks**: In Centralized Database all the data are stored in a single point of storage so If a hacker gains access to the server they can easily manipulate the entire database.
- **Slower performance:** Centralized databases can experience slower performance as the volume of data grows, since all requests must go through a single server.

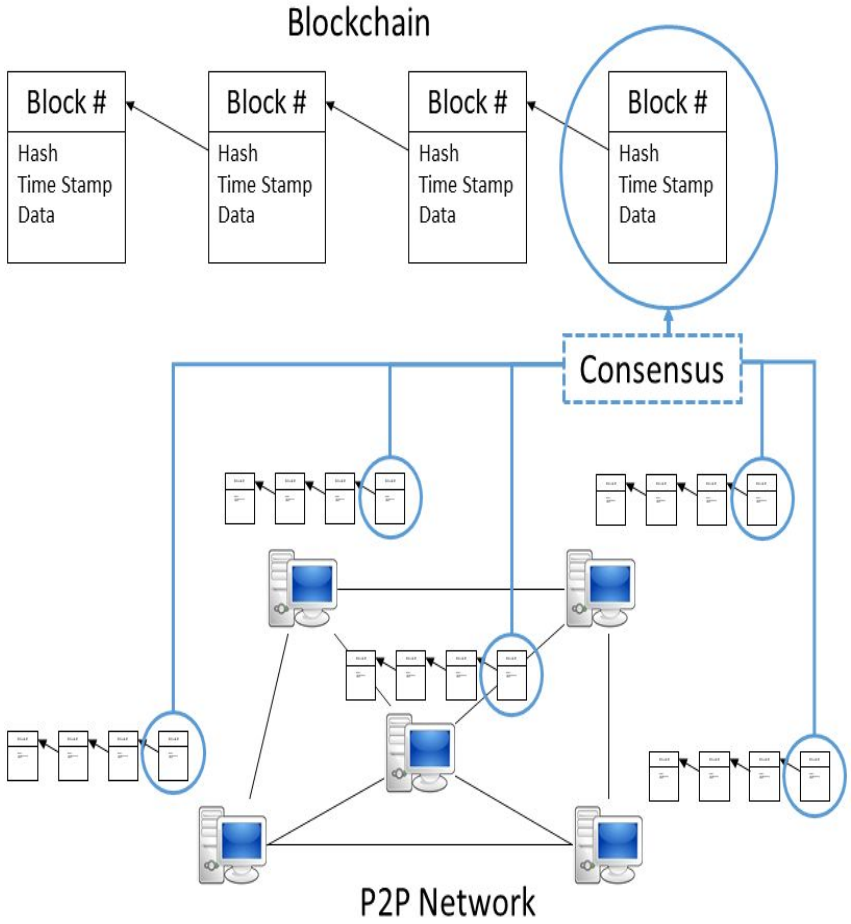
What is Blockchain?

A **shared database** blockchains store data in blocks that are then linked together.

As new data comes in, it is entered into a fresh block. Once the block is filled with **data**, it is chained onto the **previous** block, which makes the data chained together in **chronological** order.

Decentralized blockchains are **immutable**, for Bitcoin, this means that transactions are permanently recorded and viewable to anyone.

The most common use of Blockchain so far has been as a **ledger** for **transactions**.



Solution using Blockchain

As describe we can solve those problems using Decentralized Database by applying Blockchain Technology.

In a **Decentralized database,** data is spread across multiple nodes, so even if one node goes down, the rest of the network can still function. So here **single point of failure** never happen.

Using Decentralized databases, We can be easily scaled Up by adding more nodes to the network and increase its **scalability** .

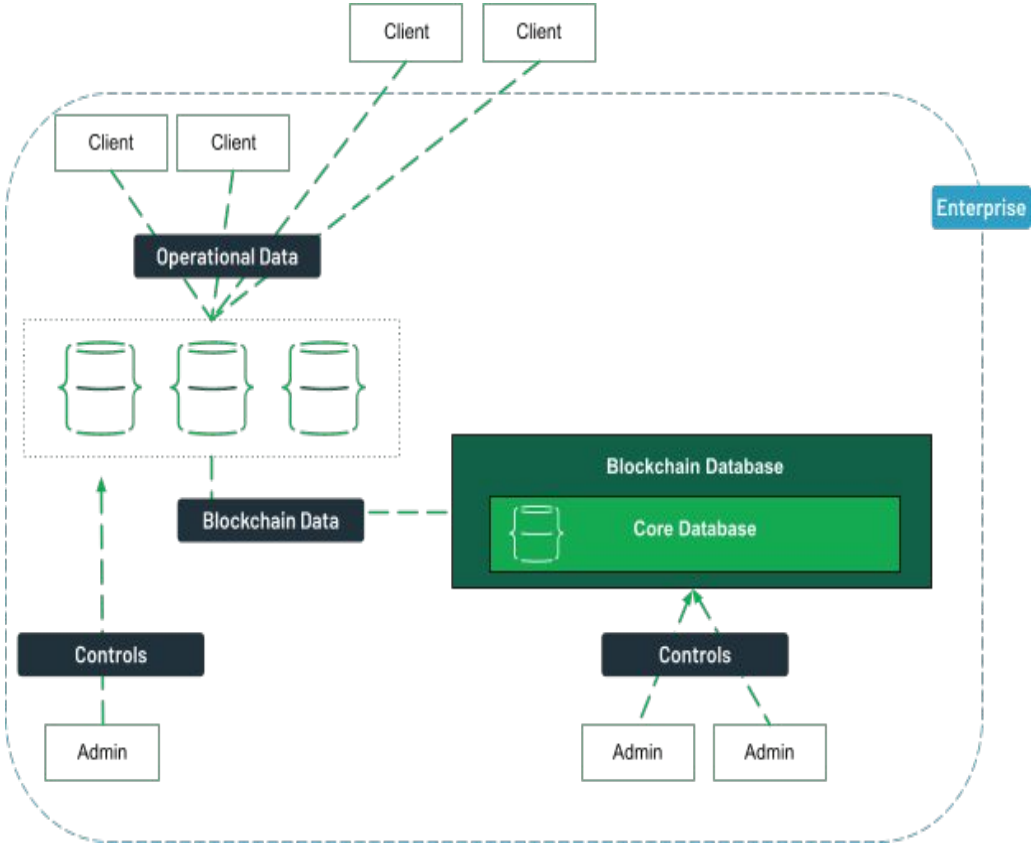
In a Decentralized database, data is stored across multiple nodes which makes it more difficult for a hacker to gain access to the entire network and it is more **secure** than the centralized database.

In Decentralized databases we can achieve faster performance by distributing data across **multiple nodes.**

Implementing Database in blockchain

We have implemented a blockchain in **JavaScript** using **OrbitDB** package where we are uploading the core database which is in **JSON** format.

Each section of the database converts to a block which has its own **hash, identity, signature, public key etc.**



What is OrbitDB

OrbitDB is an open-source, distributed, peer-to-peer database network.

It is designed to provide a decentralized way of storing, querying and replicating data across a network of nodes.

OrbitDB is designed to be used for building decentralized applications, and it is available as an npm package for use with Node.js.

Installing Orbit DB in node js:-

In terminal:

npm install orbit-db

This line of code adds a node package named ‘orbit-db’ from npm.

Core Database

```
StudentDB = [
 {
    _id: 1,
    name: "Jenna Smith",
    age: 20,
    course: "Computer Science",
  },
  {
    _id: 2,
    name: "Jacob Johnson",
    age: 19,
    course: "Mechanical Engineering",
  }
]
```

Blockchain Database


```
{
    hash: 'zdpuArsJWL4mkJJehgkEyJgCkChtTyF5NTBHUestQLr8Du2fD',
    id:
'/orbitdb/zdpuAtSL6WnotZZpKgXA9ePtRkaomJA8PBa6eGMZDZPrFHyrv/student-database',
    payload: { op: 'PUT', key: 1, value: [Object] },
    next: [ 'zdpuAtEZ2t7qqsxhbqH8R8AopW4maSHznFZZvBvVXeR2FhZSU' ],
    refs: [
      'zdpuArSPN84Gh8b9TNworKzJWZnKcMST5Gmhq76uXJqfucKZk',
      'zdpuAvoy1sgCzG5YhimTiNxXDWitF3f2sWo2diCuu78odR4CJ',
      'zdpuAtLTPVJHyByEVzBwzYAQnxu2Hx6GmAwfFJSBrkFwSB47g',
      'zdpuAqyigJbmsGmuu6J8wdKo6oDmfzYSQDAipYb8y17KX5RXv'
    ],
    identity: {
      id:
'03223b09e849e176b483ca6f404ab3e8cfd57251b4867934cb77b9fb87976a7783',
      publicKey:
'04b7dacabefe927b7b98fc10c8984517e70233409fd3e799b070fec56c9f41b1aa1aa3ec4a1e2
112251b3708c502040ca315a84c4551920a1c651f2237b280f074',
      signatures: [Object],
      type: 'orbitdb'
    },
    sig:
'3045022100ca3e94ac0c76babe84ab173f1133fde06893a1c6b12d38d1cdba7d7101024ac0022
0457ad356bd11307c61a90b4beac6806dc359dea2318e6772248cd06334dbc72c'
}
```

Blockchain Database

hash: zdpuAtSL6WnotZZp...
Key: 1
Next: zdpuAtEZ2t7qqasxhb...
Prev: zdpuArSPN84Gh8b...
id: 03223b09edfdfggdf849....
publickey: 04b54dfs6gd7d...

hash: adcuvAtSL6WncsYYp...
Key: 2
Next:zdpuBuFy346qqasxhb...
Prev: zdpuAtEZ2t7qqasxhb...
id:08076b09edfdejidief7653....
publickey:05h674djfdhisi8...

hash: vrxsAtSL6WnotZZx...
Key: 3
Next: sjiwizzcafths56k6hxk...
Prev: zdpuBuFy346qqasxhb...
id:0973b09edfdghdt66729....
publickey:076dfwf654av4u...

Student Database

Key -> Id
Id: 1
Name:"Jenna Smith"
Age: 20
Course: "Mechanical Engineering"

Key -> Id
Id: 2
Name: "Jacob Johnson"
Age: 19
Course:Mechanical Engineering

Key -> Id
Id: 3
Name: "john Smith"
Age: 22
Course: Computer Science

Steps Taken For Implementation

01 **Importing the DB**

Connecting the core database to blockchain using **docstore** function in OrbitDB.

02 **Reading the Blockchain**

Reading the blockchain database using db.get() where db is the name of the instance and .get() is an inbuilt function.

03 **Update query on Blockchain**

Finding the hash of a specific student using id then deleting the consisting hash and then add the updated hash with updated name.

04 **Disconnecting the Blockchain**

Disconnecting from the blockchain using db.disconnect() which is an inbuilt function in OrbitDB

Importing the core database in blockchain

In the below code:-
studentDB is the local **JSON** Database containing data of 20 random students.
Orbitdb is an object of **orbitdb** package

```
// Code Snippet of putting the local db in blockchain

// Open the database
const db = await orbitdb.docstore("student-database");

// Load the database
await db.load();

// Add the JSON data to blockchain from index of 20
for (var i = 0; i < 20; i++) {
  await db.put(studentData[i]);
}
```

OrbitDB function **docstore**:-

```
const db = await
orbitdb.docstore("student-database",
{ indexBy: "name",});
```

Here, **indexBy** is a property by which the blockchain connects the key value with the given attribute of the database.

Now, orbitdb takes the attribute named as **“_id”** as a default case if no such property is defined.

Gitup Repo:-https://github.com/slow-codex/blockchaindb/blob/main/ImportDB/UsingLocal.js

Reading the blockchain database

In the below code:-
studentDB is the blockchain Database containing data of 20 random students.

```
// Code Snippet of reading the blockchain db

// Creating a constant profile to get the  information from studentDB
const profile = studentDB.get("10");
console.log(profile);
```

OUTPUT

```
[
  {
    _id: 10,
    name: 'William Brown',
    age: 22,
    course: 'Mechanical Engineering'
  }
]
```

Github Repo:
https://github.com/slow-codex/blockchaindb/blob/main/ReadDB/ReadDB.js

Updating the blockchain database

In the below code:-
studentDB is the Blockchain Database containing data of 20 random students

```
// Code Snippet of updating the blockchain
// Changing the the name of the student with id = "10"
profile[0].name = "New name";
// Deleting the student details at id = "10"
await studentDB.del(10);
// Adding the new student details at id = "10"
await studentDB.put(profile[0]);
// Showing the new student details at id = "10"
const pr1 = studentDB.get("10");
console.log("After Changing updating the new name");
console.log(pr1);
```

OUTPUT

```
[
  {
    _id: 10,
    name: 'William Brown',
    age: 22,
    course: 'Mechanical Engineering'
  }
]
After Changing updating the new name
[
  {
    _id: 10,
    name: 'New name',
    age: 22,
    course: 'Mechanical Engineering'
  }
]
```

Github Repo: https://github.com/slow-codex/blockchaindb/blob/main/UpdateDB/UpdateDB.js

Comparative Analysis

In our project we have connected our local database in blockchain using OrbitDB. Now, **OrbitDB** is an open source node package with **latest version of 0.87**. Being fairly new there **hasn't been any research paper** regarding the use of OrbitDB for **Database Management System in Blockchain**.

Previously many researchers have tried to implement data managing system in blockchain and have achieved that goal using **DynamoDB, PingER** etc. But none of them connects the whole database in blockchain, instead uploads certain data in the block format and the rest in cloud storage system

For example, there's a research paper on "**A blockchain-based Data Storage using PingER**" where the authors have used **PingER** to achieve the same outcome but in their project they have also used the help of **cloud storage**.

Conclusion

We would like to thanks our project mentor Professor Jhumma Dutta, who help us a lots during this project.

In conclusion, blockchain technology offers a revolutionary new way to securely store, share, and transfer data and assets in a decentralized and trustless manner. Its potential applications are wide-ranging and impactful, including in areas such as Database Management System, finance, healthcare, supply chain management, and lot's of more.

During this presentation, we have explored the basics of blockchain technology, including the key components such as nodes, blocks, and consensus algorithms.

Thank You