Blockchain Technology and distributed databases

Stefan Cliff 2019/230449<efas of SQL - S

## Blockchain Technology – An intro

A **blockchain** is a list of records that is always growing, also called **blocks** that are then linked together using *cryptography*. Each block has the *cryptographic hash* of the previous block, along with a timestamp and the transactional data. The timestamp is there to prove that the transaction data existed at the time when the block was published which actually allows it to get into its hash. While this sounds semi- confusing, trust me it’s really not.

As each block contains the information about the previous block in line, they form a sort of chain for one another with each following block adding onto the ones before them. What this actually means is that blockchains are resistant/immune to their information being modified because, once recorded, the data in any single block cannot be altered without having to alter all the following blocks from that one.

Blockchains are managed by a *peer-to-peer* network most of the time so they can be used in a publicly *distributed ledger*, where every single node must follow the protocol to communicate and validate new blocks. While blockchain records are not unalterable, as *forks* it is actually possible. And while they may seem secure by their very design, they also support the example of a distributed computing system that has a high *Byzantine fault tolerance*.

Who actually invented the blockchain is actually unknown, be it a person or a group of people, it was created by *Satoshi Nakamoto* back in 2008 to serve as a public transaction ledger for the now very famous cryptocurrency ***Bitcoin***. Blockchains invention for bitcoin made it the first digital currency that solved the *double-spending* problem without actually needed some authority or centralized server. Its design inspired other applications and blockchains that are readable by the public and are now widely used for cryptocurrencies.

Private versions of blockchains have been proposed for businesses, but a few media sources for all things technology (i.e. *Computerworld*) have deemed the idea without an actual proper security model nothing more than *“snake oil”*. On the other hand, there have been some who argue that permissioned blockchains, if its design is made with care, might actually be more decentralized and therefore more secure in practice than the permission-less ones that we have now.

## Blockchain Technology – Brief History

The first proposition of blockchain esc protocols was written in 1982 by **David Chaum** in his dissertation: “*Computer Systems Established, Maintained and Trusted by Mutually Suspicious Groups”*. Further work on a cryptographically secured chain of blocks was described in 1991 by **Stuart Haber** and **W. Scott Stornetta**. They wanted to implement a system wherein document timestamps could not be tampered with. In 1992, **Haber**, **Stornetta**, and **Dave** **Bayer** incorporated *Merkle* trees to the design, which improved its efficiency by allowing several document certificates to be collected into one block.

As mentioned previously, the first ever actual conceptualization was by **Satoshi Nakamoto** in 2008. Nakamoto improved the design in an important way using a *Hashcash-like* method to timestamp blocks without requiring them to be signed by a trusted party and introducing a difficulty parameter to stabilize rate with which blocks are added to the chain.

## Blockchain Technology – Structure

A **blockchain** is a *decentralized*, *distributed*, and oftentimes *public* *digital* *ledger* consisting of records called *blocks* that is used to record transactions across many computers so that any involved block cannot be altered retroactively, without the alteration of all subsequent blocks.

This allows the participants to verify and audit transactions independently and relatively inexpensively. A blockchain database is managed autonomously using a *peer-to-peer* network and a distributed timestamping server. They are authenticated by mass collaboration powered by collective self-interests.

Such a design facilitates robust workflow where participants' uncertainty regarding data security is marginal. The use of a blockchain removes the characteristic of infinite reproducibility from a digital asset. It confirms that each unit of value was transferred only once, solving the long-standing problem of double spending. A blockchain has been described as a value-exchange protocol. A blockchain can maintain title rights because, when properly set up to detail the exchange agreement, it provides a record that compels offer and acceptance.

Logically, a blockchain can be seen as consisting of several layers:

* infrastructure (hardware)
* networking (node discovery, information propagation and verification)
* consensus (proof of work, proof of stake)
* data (blocks, transactions)
* application (smart contracts/dApps, if applicable)

There is a lot more to talk about when it comes to blockchains, but to save time we shall now move onto the second of the three parts of this paper.

## Distributed Databases – Intro

What is a distributed database? A **distributed database** is a database in which data is stored across different physical locations. It may be stored in multiple computers located in the same physical location (e.g. a data center); or maybe dispersed over a network of interconnected computers. Unlike parallel systems, in which the processors are tightly coupled and constitute a single database system, a distributed database system consists of loosely coupled sites that share no physical components.

System administrators can distribute collections of data (e.g. in a database) across multiple physical locations. A distributed database can reside on organized network servers or decentralized independent computers on the Internet, on corporate intranets or extranets, or on other organization networks. Because distributed databases store data across multiple computers, distributed databases may improve performance at end-user worksites by allowing transactions to be processed on many machines, instead of being limited to one.

Two processes ensure that the distributed databases remain up-to-date and current:

* **replication**
* **duplication**

**Replication** involves using specialized software that looks for changes in the distributive database. Once the changes have been identified, the replication process makes all the databases look the same. The replication process can be complex and time-consuming, depending on the size and number of the distributed databases. This process can also require much time and computer resources.

**Duplication**, on the other hand, has less complexity. It identifies one database as a **master** and then duplicates that database. The duplication process is normally done at a set time after hours. This is to ensure that each distributed location has the same data. In the duplication process, users may change only the master database. This ensures that local data will not be overwritten.

Besides distributed database replication and fragmentation, there are many other distributed database design technologies. For example: *local autonomy*, *synchronous*, and *asynchronous* *distributed database* technologies.

The implementation of these technologies can and do depend on the needs of the business and the sensitivity/confidentiality of the data stored in the database and the price the business is willing to spend on ensuring data security, consistency and integrity.

When discussing access to distributed databases, ***Microsoft*** favors the term distributed query, which it defines in protocol-specific manner as "any **SELECT**, **INSERT**, **UPDATE**, or **DELETE** statement that references tables and row sets from one or more external OLE DB data sources". Oracle provides a more language-centric view in which distributed queries and distributed transactions form part of distributed SQL.

## Blockchain vs Distributed Databases

To save on everyone’s time, here is a simple table that compares the two in an easy and readable manner:

|  |  |  |
| --- | --- | --- |
|  | Blockchain | distributed databases |
| 1 | Blockchain is decentralized because there is no admin or in-charge. | The database is centralized because it has admins and in-charge. |
| 2 | Blockchain is permission-less because anyone can access it. | The database required permission because it can be accessed only by entities who have rights to access. |
| 3 | Blockchains are slow. | Databases are fast. |
| 4 | It has a history of records and ownership of digital records. | It has no history of records and ownership of records. |
| 5 | Blockchain is fully confidential. | The database is not fully confidential. |
| 6 | Blockchain has only Insert operation. | The database has **CREATE**, **READ**, **UPDATE**, and **DELETE** operation. |
| 7 | It is a fully robust technology. | It is not entirely robust technology. |
| 8 | Disintermediation is allowed with blockchain. | Disintermediation is not allowed with the database. |
| 9 | Anyone with the right proof of work can write on the blockchain. | Only entities entitled to read or write can do so. |
| 10 | Blockchain is not recursive. Here, we cannot go back to repeat a task on any record. | The database is recursive. Here, we can go back to repeat a task on a particular record. |

This table I think nicely compares the core differences of these two styles of data storing, but in my honest opinion I think that the two shouldn’t be compared as if one is better than the other. Instead they both have their place in the worlds of technology, security and data storing. I do not think one shall surpass the other or make it obsolete, instead they shall simply both serve their purposes their way.

Thanks for reading/listening

Stefan Cliff 2019/230449<efas of SQL - S