UNIVERSIDADE FEDERAL DO CEARÁ

CENTRO DE CIÊNCIAS

DEPARTAMENTO DE COMPUTAÇÃO

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

CARLOS SERGIO DA SILVA MARINHO

MOON: AN APPROACH TO DATA MANAGEMENT ON RELATIONAL DATABASE AND BLOCKCHAIN

FORTALEZA

2020

CARLOS SERGIO DA SILVA MARINHO

MOON: AN APPROACH TO DATA MANAGEMENT ON RELATIONAL DATABASE AND BLOCKCHAIN

Dissertation submitted to the Graduate Program in Computer Science of the Center of Science of the Federal University of Ceará, as a partial requirement to obtain the Academic Master Degree in Computer Science. Concentration Area: Computer Science.

Advisor: Prof. Dr. Javam de Castro Machado

Co-advisor: Prof. Dr. Leonardo Oliveira Moreira

FORTALEZA

2020

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

M29m Marinho, Carlos Sergio da Silva.
MOON: : An Approach to Data Management on Relational Database and Blockchain / Carlos Sergio da Silva Marinho. – 2020.
82 f. : il.

Dissertação (mestrado) – Universidade Federal do Ceará, Centro de Ciências, Programa de Pós-Graduação em Ciência da Computação, Fortaleza, 2020.
Orientação: Prof. Dr. Javam de Castro Machado.
Coorientação: Prof. Dr. Leonardo Oliveira Moreira.

1. Blockchain.. 2. Relational Databases.. 3. Distributed Architecture.. 4. Data Management.. I. Título.
CDD 005

CARLOS SERGIO DA SILVA MARINHO

MOON: AN APPROACH TO DATA MANAGEMENT ON RELATIONAL DATABASE AND BLOCKCHAIN

Dissertation submitted to the Graduate Program in Computer Science of the Center of Science of the Federal University of Ceará, as a partial requirement to obtain the Academic Master Degree in Computer Science. Concentration Area: Computer Science.

Approved on: 11 23, 20

EXAMINERS COMMITEE

Prof. Dr. Javam de Castro Machado (Advisor)
Universidade Federal do Ceará (UFC)

Prof. Dr. Leonardo Oliveira Moreira (Co-advisor)
Universidade Federal do Ceará (UFC)

Prof. Dr. Flávio Rubens de Carvalho Sousa
Universidade Federal do Ceará (UFC)

Prof. Dr. Paulo Henrique Mendes Maia
Universidade Estadual do Ceará (UECE)

To my family.

ACKNOWLEDGEMENTS

To my parents for their support and motivation to move forward in the most important moments of my life.

To my advisors and mentors, Javam Machado and Leonardo Moreira, for the advice, guidance, and opportunities granted that were fundamental to the success of this research.

To professors Flávio Rubens de Carvalho Sousa and Paulo Henrique Mendes Maia for their availability to participate in the evaluation committee for my master's degree.

To the Laboratório de Sistemas e Banco de Dados (LSBD) for having provided all the necessary structure for the development of the scientific research contained in my dissertation.

To my friends Serafim Costa, Denis Cavalcante, Lucas Falcão and Daniel Praciano for their support in the discussions that we had at LSBD during the evolution of this research.

To all my friends of LSBD for their help in several situations that directly or indirectly contributed to the construction of this scientific research.

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

“Let everything happen to you
Beauty and terror
Just keep going
No feeling is final”

(Rainer Rilke)

RESUMO

O modelo relacional tem sido amplamente utilizado por décadas e foi importante para a popularização do uso de bancos de dados. Atualmente, várias aplicações de diferentes domínios continuam orientadas a dados, entretanto alternativas ao modelo relacional vêm se solidificando. Uma delas é a tecnologia Blockchain, que é considerada disruptiva e possui propriedades relevantes, como imutabilidade e ausência da necessidade de terceira parte confiável. Portanto, algumas aplicações que usam Bancos de Dados Relacionais (BDR) podem se beneficiar dessas propriedades migrando parte de seus dados para Blockchains. Esta pesquisa apresenta a abordagem *the approach to data Management on relatiOnal database and blOckchaiN* (MOON). Nesta abordagem, aplicações clientes usam a linguagem SQL DML para se comunicar com o MOON. Consequentemente, o cliente envia operações de *insert*, *update*, *select* e *delete* para o MOON, que trata a requisição independentemente de os dados estarem armazenados em Blockchain ou BDR. Além disso, o MOON realiza mapeamento entre o modelo relacional e a Blockchain, integrando as duas tecnologias, além de indexar dados de Blockchain. Experimentos foram realizados para validar este estudo. O primeiro gerou carga de trabalho a partir de dados reais de um hospital português. A segunda parte foi realizada usando dois *benchmarks* validados: Voter e Twitter. Em todos os experimentos, três cenários foram utilizados: i) dados armazenados apenas no BDR; ii) dados particionados usando MOON; e iii) dados armazenados apenas na Blockchain. As métricas utilizadas nas avaliações foram tempo de resposta e corretude. A conclusão é que o MOON responde a solicitações corretamente e fornece características BDR e Blockchain aos dados, como suporte a consultas complexas e imutabilidade de dados. Além disso, seu tempo de resposta foi intermediário entre BDR e Blockchains.

Palavras-chave: Blockchain. Banco de Dados Relacional. Arquitetura Distribuída. Gerenciamento de Dados.

ABSTRACT

The Relational model has been used widely for decades and was valuable for the popularization of the use of Databases. Nowadays, many applications of several domains continue to be data-oriented, but alternatives to the Relational model have been solidifying. One of them is Blockchain, which is considered a disruptive technology and has relevant properties, such as immutability and no need for centralized third parties. Therefore, applications that use Relational Databases (BDR) can benefit from these properties by migrating part of their data to Blockchains. This research presents the approach to data Management on relatiOnal database and blOckchaiN (MOON), which its client applications use SQL DML to communicate with the MOON. Then, clients send inserts, updates, selects, and deletes queries to the MOON, which execute them regardless of whether the data is in a Blockchain or RDB. Furthermore, the MOON performs mapping between relational and Blockchain model, integrating the two technologies, and Blockchain's data indexing. There were two experiments to validate this study. The first generated workload from real data from a Portuguese hospital and the second used two validated benchmarks: Voter and Twitter. In all experiments, there was an execution on three scenarios: i) data stored only in the BDR; ii) partitioned data using MOON; and iii) data stored only on the Blockchain. The metrics used in the evaluations were response time and correctness. The conclusion is that the MOON responds to requests correctly and provides RDB and Blockchain features to the data, such as supporting complex queries and data immutability. Moreover, their response time was intermediate between BDR and Blockchains.

Keywords: Blockchain. Relational Databases. Distributed Architecture. Data Management.

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

LIST OF ABBREVIATIONS AND ACRONYMS

EQL	Ethereum Query Language
SEBDB	Semantics Empowered BlockChain DataBase
MOON	data Management on relatiOnal database and blOckchaiN
P2P	peer-to-peer
RDB	Relational Database
SQL	Structured Query Language
IoT	Internet of Things
UTXO	Unspent Transaction Output
PoW	Proof of Work
PoS	Proof of Stake
DPoS	Delegated Proof of Stake
PoET	Proof of Elapsed Time
DBMS	Database Management System
RDBMS	Relational Database Management System
SEQUEL	Structured English Query Language
ANSI	American National Standards Institute
JVM	Java Virtual Machine
JPA	Java Persistence API
BST	Binary Search Tree
JSON	JavaScript Object Notation
VM	Virtual Machine
PL/SQL	Procedural Language extensions to SQL

CONTENTS

1 INTRODUCTION

A database is a collection of related data, and the relational model is widely used in several contexts (ELMASRI; NAVATHE, 2015). This model defines a database as a collection of one or more relations composed of rows and columns. On the other hand, Blockchain provides distributed, reliable, and secure support for large-scale peer-to-peer (P2P) network transactions (NAKAMOTO, 2008). It is a disruptive technology as there is a decentralized trust entity, with no need for trustworthy centralized third parties, such as notary offices. Hence, network nodes do not necessarily need to trust each other for the system works correctly (GREVE *et al.*, 2018).

The relational model was substantial to the popularization of the use of databases (PAVLO; ASLETT, 2016), as this model met the applications' requirements when it was developed. In these times, many applications continue to be data-driven and use the relational model, as it is suited to their demands (MAENHAUT *et al.*, 2015). However, alternatives to the Relational model have been solidifying, as some applications may need requirements that are better suited to other models. Then, Blockchain emerges as an option, being used by applications that benefit from its properties, such as immutability and irrefutability.

The use of Blockchain is advantageous to Relational Databases (RDBs) in aspects such as security and trust-building. However, RDBs usually have a higher throughput than Blockchains (CHOWDHURY *et al.*, 2018). It probably occurs because, unlike Blockchains, RDBs do not implement expensive consensus, such as Proof of Work, used by several Blockchains in the literature. Ruan et al. (2019) argue that the two technologies' primary objective is different because while Blockchains focus on integrity and security, databases center on performance.

In different domains, such as e-health or e-commerce, an application uses datasets that are more suited to the relational model and others to the Blockchain. Inherent data to Blockchain are in contexts in which there are no trusted third parties and no trust between network nodes or immutability of the data. On the other hand, the data most suited to the relational model are those that are desired to perform more complex queries using joins or analyses.

Given this context, some applications may benefit from using a hybrid approach. Based on a survey of the current state of the art, it was found that there is a shortage of studies that propose approaches to hybrid data management, with data storage in RDB and Blockchain. Moreover, this work proposes the use of Structured Query Language (SQL) as an interaction

method between client applications and the approach, which makes access transparent to the data model used.

1.1 Main Objective

The goal of the research is to develop an approach that manages data stored in relational databases and Blockchain infrastructures.

1.2 Specific Objectives

- Explore data management solutions to store data in Blockchain and RDB;
- Design a relational data mapping for Blockchain;
- Validate the proposed approach through an experimental evaluation.

1.3 Hypothesis

An approach for managing data that uses the relational and Blockchain model can provide the applications: i) Complex queries and data mutability, relational model characteristics, and ii) Immutability, transparency, and lack of reliable third party, inherent to Blockchain.

1.4 Research questions

- **RQ1**: What are the advantages of developing an approach that maintain data disjointly in Blockchains and RDBs?
- **RQ2**: Given a collection of data, which aspects should be considered for decision-making on how best to store it: Blockchain or RDB?
- **RQ3**: How to map relational data to a Blockchain infrastructure?

1.5 Scientific Papers

Tables 1 and 2 include the work done throughout the past two years. Five articles were submitted, evaluated, and accepted by the scientific community. The work published in JIDM is less related to the this dissertation. However, it was relevant in this research's progress, as it produced the necessary knowledge for important areas of this dissertation, comprising topics of databases and Distributed Systems. The study published in the RSMD introduced the research

Table 1 – Published Papers on Journals - Own authorship

Journal	Title	Authors
JIDM	LABAREDA: A Predictive and Elastic Load Balancing Service for Cloud-Replicated Databases	**Carlos Sérgio da Silva Marinho** Leonardo Oliveira Moreira Emanuel Ferreira Coutinho José Serafim da Costa Filho Flávio Rubens de Carvalho Sousa Javam de Castro Machado
RSMD	Oportunidades de Pesquisa para o Uso de Infraestruturas Blockchain na Gestão de Dados Distribuídos	Leonardo Oliveira Moreira **Carlos Sérgio da Silva Marinho** Maurício Moreira Neto Emanuel Ferreira Coutinho José Neuman de Souza Javam de Castro Machado

Table 2 – Published Papers on Conferences - Own authorship

Conference	Title	Authors
SBBD 2019	MOON: An Approach to Data Management on Relational Database and Blockchain	**Carlos Sérgio da Silva Marinho** Leonardo Oliveira Moreira Javam de Castro Machado
ICSA-C 2020	Using a Hybrid Approach to Data Management in Relational Database and Blockchain: a Case Study on The E-health Domain	**Carlos Sérgio da Silva Marinho** José Serafim da Costa Filho Leonardo Oliveira Moreira Javam de Castro Machado
ICSA-C 2020	Research Opportunities for E-health Applications with DNA Sequence Data using Blockchain Technology	Maurício Moreira Neto **Carlos Sérgio da Silva Marinho** Emanuel Ferreira Coutinho Leonardo Oliveira Moreira Javam de Castro Machado José Neuman de Souza

questions exposed in this dissertation.

The other works are strongly correlated with the present research. The article published in SBBD describes the approach proposed in this document. The first article published in the ICSA is an extension of it and contains part of the experiments exposed in this dissertation. The second article published in the ICSA contributed to a better understanding of the domain of e-health applications, which was used to carry out part of the experiments.

1.6 Dissertation Structure

The following chapters of this document are structured as follows:

– Chapter 2: Provides background information;

– Chapter 3: Discusses related work;

– Chapter 4: Explain in detail the proposed; approach, including architectures, algorithms, flows, and technologies adopted;
– Chapter 5: Presents the experimental evaluation performed in the present study, identifying the workloads, the technologies and the environment used;
– Chapter 6: Presents the conclusions of this research and proposes future work.

2 BACKGROUND

This chapter aims to prepare the reader with background information for understanding the context and the formulation of problems and the proposed solution. Defining and discussing Blockchain technology is the central focus of the chapter as it is a recent technology, less discussed in the literature than other consolidated technologies, such as RDB. Therefore, there is an address for aspects related to properties, categories, and contexts in which Blockchains are used. Moreover, there is an explanation of technical topics used by Blockchain, such as consensus and cryptography.

This chapter introduces and discusses other topics. There is a conceptualization of databases, database management systems, and relational model, explaining their principal characteristics relevant to this research. Finally, there is the definition of e-health, since it is addressed in this document in part of the experiments carried out to evaluate the proposed approach.

2.1 Blockchain

This section discusses Blockchain, its technologies, properties, categories, and concepts. First, there is a Blockchain overview in subsection 2.1.1. Then, the Blockchain properties are discussed in subsection 2.1.2. The categories of Blockchain are explained in subsection 2.1.3. Blockchain's central elements are cryptography, transactions, smart contracts, blocks, ledger, and client software. It is explained in subsection 2.1.4. Finally, subsection 2.1.5 shows some consensus algorithms.

2.1.1 Blockchain Overview

Blockchain is a technology that provides reliable, secure, and distributed support for transactions between nodes in a large-scale P2P network (NAKAMOTO, 2008). Blockchain is a disruptive technology since it has a decentralized trust entity that there are no centralized third parties (i.e., bank, notary's office, and government) to mediate transactions between them. Moreover, the network participants do not necessarily have trust with each other. Applications from different domains use Blockchain, such as e-health, finance, Internet of Things (IoT), and cloud computing (GREVE *et al.*, 2018; HUH *et al.*, 2017; CROSBY *et al.*, 2016; SAMANIEGO; DETERS, 2016; AZARIA *et al.*, 2016; SWAN, 2015).

Figure 1 compares financial transactions in two scenarios: a centralized model with third parties and a Blockchain network. There is a traditional situation in the first scenario in which a user must send money to a bank to transfer it to someone. The bank has a record of all transactions that occurred, and the structure that stores this data is the ledger. So this structure is centralized and owned by the bank. In the second scenario, the user sends a transaction to some node that sends it to the other network nodes. Next, these nodes make a consensus, and then the operation is confirmed. One network node does not need to trust the other nodes, as consensus must ensure that the entire process occurs honestly (METTLER, 2016; YLI-HUUMO *et al.*, 2016). Although there is a use of ledger in both scenarios, the second is decentralized, and each node keeps it up to date. The discussion of the advantages of using a distributed ledger is in section 2.1.4.6.

Blockchain was first defined by Nakamoto (2008) along with Bitcoin. It is a P2P network in which there are transactions of a digital currency, also called Bitcoin. Moreover, the network nodes propose transactions involving Bitcoins. These transactions are received by nodes that decide the order in which transactions will be executed and stored in a chained structure by consensus (SWAN, 2015; CAHILL *et al.*, 2020). This structure's name is Blockchain, which is replicated on each node of the network. Once published on the network, the transactions cannot change. The paradigm rupture in Nakamoto's proposal is in the elimination of the third party, which in this case would be some bank, needed in conventional financial transactions (CROSBY *et al.*, 2016).

Figure 2 presents a Blockchain overview. Blocks contain a set of n transactions, where n may vary depending on the Blockchain's implementation, as this defines the maximum size of a block. Each block has its hash and a hash pointer to indicate its previous block, forming a chained structure of blocks. Also, the first block has height 0, being called genesis. The others have the height of their previous block plus one. The nodes make a consensus to publish a new block with a set of transactions on the network. It is discussed in subsection 2.1.5 (XIAO *et al.*, 2020; LI *et al.*, 2017; SWAN, 2015).

2.1.2 Blockchain Properties

Blockchains have technical properties that contribute innovatively to the development of applications that use this technology. They are:

– **Decentralization:** According to Ray et al. (2020), Greve et al. (2018), and Zheng et al.

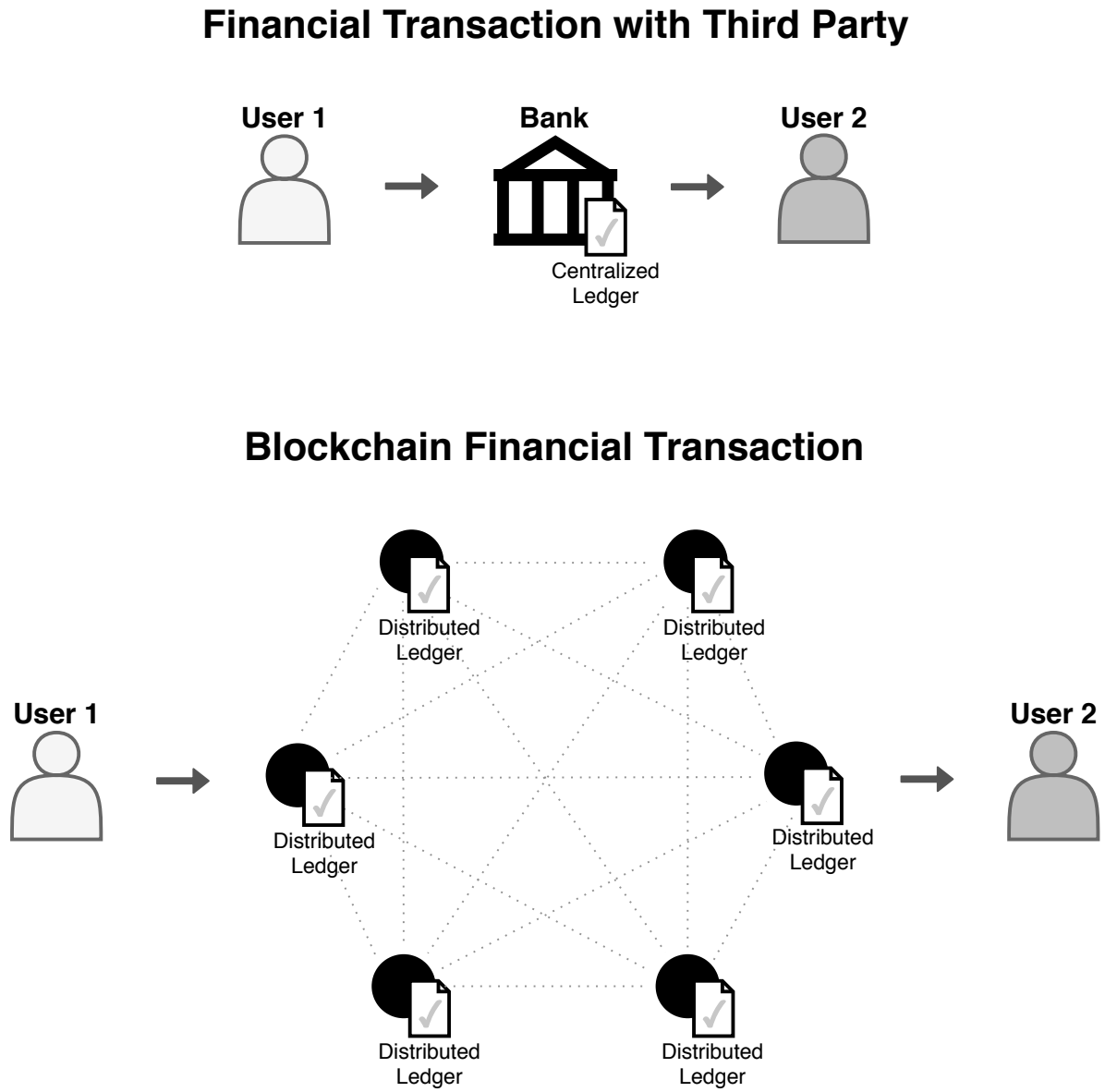


Figure 1 – Third party financial transaction and Blockchain financial transaction - Own Authorship

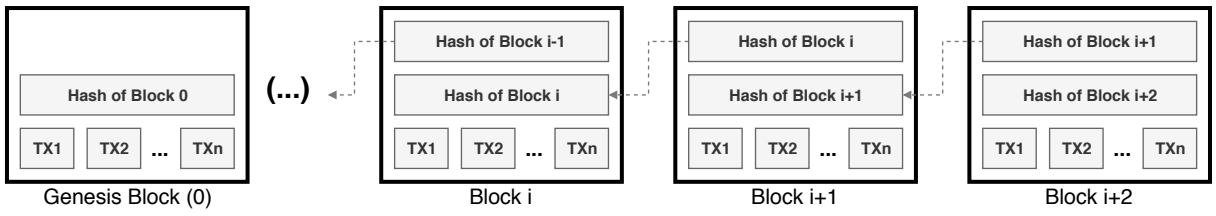


Figure 2 – Blockchain Overview - Based on Zheng et *al.* (2018), Nofer et *al.* (2017), and Sikorski et *al.* (2017)

(2017), applications and systems are executed in a distributed way by establishing trust between nodes, without third parties. For Braga, Marino, and Santos (2017), the concept of Blockchain decentralization is centered on the non-existence of the sole owner of the

ledger. Each node of the P2P network is co-owner and is responsible for maintaining its ledger replica, contributing to the global system state update.

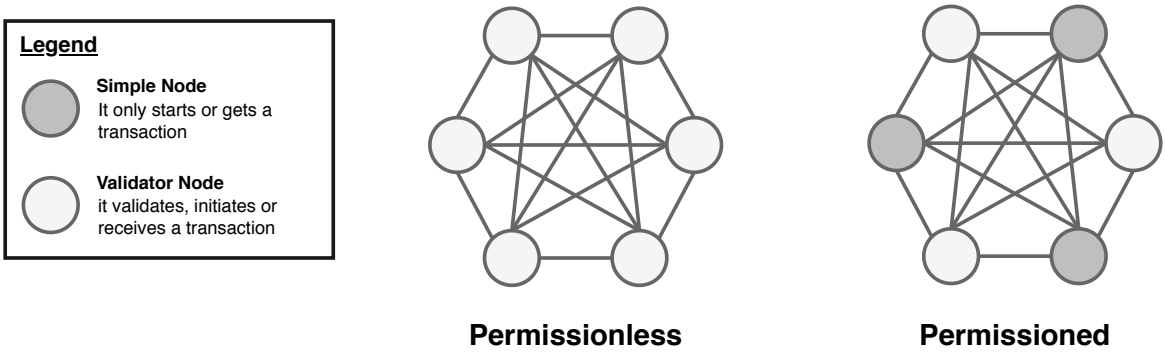
Availability: Data and transactions are replicated to network nodes securely, maintaining the system always available (GREVE *et al.*, 2018). Braga, Marino, and Santos (2017) include that the network is always available even if some nodes become offline.. Thus, the system must never be inoperable and must be able to make a consensus. However, it is worth noting that each consensus algorithm requires a minimum number of nodes online to enable consensus.

- **Integrity:** According to Greve et al. (2018), this feature is strongly related to availability. The data replication between nodes keeps the system available and consistent. Zheng et al. (2017) focus on using consensus algorithms, which is necessary to maintain Blockchain data consistency. Drescher (2017) discusses integrity as seen from three perspectives: data integrity, security, and behavioral integrity.
- **Transparency:** All transactions recorded on the ledger are visible to all network nodes and client software with read-only access (RAY *et al.*, 2020; GREVE *et al.*, 2018; BRAGA *et al.*, 2017).
- **Auditability:** According to Greve et al. (2018), Braga, Marino, and Santos (2017), and Zyskind, Nathan, and Pentland (2015), this is the property that defines that all network nodes can check the ledger. However, Braga, Marino, and Santos (2017) name this property public visibility. Zheng et al. (2017) discuss the Unspent Transaction Output (UTXO) model, which is adopted in some Blockchain networks, such as Bitcoin. UTXO enforces that a transaction must reference the last transaction that mentioned the resource being transacted. Thus, the referenced transaction, which was marked as not spent, is now marked as spent and can no longer be mentioned. It facilitates transaction validation and tracking.
- **Immutability:** Once registered in the ledger, a transaction cannot be changed. The only possible update should be adding new transactions through consensus (RAY *et al.*, 2020; XIE *et al.*, 2020; GREVE *et al.*, 2018; CHOWDHURY *et al.*, 2018; BRAGA *et al.*, 2017).
- **Irrefutability:** Once stored by the Blockchain network, a node can not contest the veracity of a transaction sent by itself (GREVE *et al.*, 2018; BRAGA *et al.*, 2017).
- **Disintermediation:** Blockchain efficiently integrates multiple systems. Thus, this technology is considered a connector for complex systems, which simplifies the design of

systems and processes (GREVE *et al.*, 2018; BRAGA *et al.*, 2017; XU *et al.*, 2016).

2.1.3 Blockchain Categories

There are two categories for Blockchain network: i) permissionless; and ii) permissioned. The main difference between these two categories is the permissions given in it as if anyone can join or publish blocks. Subsections 2.1.3.1 and 2.1.3.2 detail the two categories mentioned. Figure 3 compares the main aspects of the two categories.



Access	Public	Need Authorization
Legislation and regulation	It may have its own rules	According to legislation and regulation
Validators	Anonymous	Preselected group
Potential Applications	Open Access Applications	Restricted Corporate Environments
Example	Bitcoin	Hyperledger Fabric

Figure 3 – Comparison between permissioned and permissionless networks - Based on Braga et *al.* (2017)

2.1.3.1 Permissionless

It is a Blockchain network open for everyone to enter and publish blocks without needing permission from anyone. Since there is no authentication to join and leave the network, participating nodes are considered unknown and can randomly join and leave. Commonly, this kind's Blockchain platforms are open source, available to anyone who wants to inspect the code (YAGA; MELL, 2018; WÜST; GERVAIS, 2018; BRAGA *et al.*, 2017; XU *et al.*, 2017).

Since any node can send transactions and publish blocks, there is usually a complex

consensus in permissionless Blockchains to add blocks to the ledger. A consensus algorithm widely discussed in the literature is Proof of Work (PoW), probably because it was proposed in the fundamental article of Blockchain (NAKAMOTO, 2008). Consensus on Blockchain networks are explained in subsection 2.1.5. Bitcoin and Ethereum are examples of permissionless Blockchains (YAGA; MELL, 2018; BASHIR, 2018; Dinh *et al.*, 2018; WÜST; GERVAIS, 2018).

2.1.3.2 Permissioned

It is a Blockchain network in which nodes are identified, authenticated, and authorized. The network structure is known, with *n* participating nodes that depend on permissions to join or leave the network. Permissioned Blockchains usually best serve corporate or private interests, in which participants have well-defined roles and can organize themselves into groups. Besides, some of the nodes may function as consensus validators (VO *et al.*, 2018; WÜST; GERVAIS, 2018; LI *et al.*, 2017; XU *et al.*, 2017).

In a permissioned network, it is not common to use computationally costly consensus algorithms, such as PoW (CHICARINO *et al.*, 2017). As network nodes are known and are usually in a controlled scenario, network participants are probably trusted. Thus, it is common to use round-robin or Proof of Identity consensus, for example. An example of a permissioned Blockchain is Hyperledger Fabric (RAY *et al.*, 2020; CHICARINO *et al.*, 2017; CACHIN, 2016).

2.1.4 Central elements of Blockchain

To discuss Blockchain properties and concepts, it is necessary to understand some of the key components and techniques used. These components are cryptography, transactions, smart contracts, blocks, and ledger. The following subsections discuss these points.

2.1.4.1 Cryptography

Using encryption ensures system and application security. Two features are most prevalent in the Blockchain context: cryptographic hash functions and digital signatures.

2.1.4.1.1 Cryptographic Hash Functions

These are functions that take as input data, such as text or image, of any size and calculate an output, known as a digest. The digest is intended to be unique. As shown in Table 3, if a minimal change is made to the input, such as a single bit, the output of the function results in something substantially different (GATTESCHI *et al.*, 2020; GONCZOL *et al.*, 2020; YAGA; MELL, 2018).

Input	**Output SHA-256**
1.01	0xcb5d2011975d7a70e93f7cf9d2934fc752c4f1c5013a80cd34b8d2deb5ded6b0
1.02	0xf42b383a2bf402ee84ccf15180546c6a5906c8e54c939fd633be87fac485f225
1.011	0xb6f3f5f534166b4471c6e1f2d0f9b8fabb6f566bf1448d55bcb39ea2294d042f

Table 3 – Very distinct outputs for similar inputs - Based on Yaga and Mell (2018)

Cryptographic hash functions have the following security properties:

- **They are preimage resistant**: For $f(x) = y$, it is infeasible to find the value of x. It means that, given an output, it is possible to discover the input. Because of this, these functions are known as unidirectional (YAGA; MELL, 2018; ROGAWAY; SHRIMPTON, 2004);
- **They are collision resistants:** Given $f(x) = y$, it is impossible to find a z where $f(z) = y$. That is, you cannot find two entries that applied to the same hash function produce the same output (YAGA; MELL, 2018; ROGAWAY; SHRIMPTON, 2004).

2.1.4.1.2 Digital Signatures

Digital signatures must be irrefutable and unfalsifiable, which is also valid for handwritten signatures on documents. Thus, a digital signature should only be made by one but validated by anyone who wishes. Consequently, when the signature is valid, its owner cannot dispute its validity (GREVE *et al.*, 2018).

These signatures are implemented using asymmetric key encryption. This cryptographic model uses public and private keys. Public keys are widely disseminated and are used to verify the authenticity of the signature. A private key is known only to its own and is used to sign documents (BRAGA; DAHAB, 2015).

Implementing digital signatures typically has at least three methods (GREVE *et al.*, 2018):

- Generate keys: It receives the wanted size for the keys. Returns a pair of keys;
- Sign: It receives a message and the private key. Returns the encrypted message;

– Verify: It receives the message, the public key, and the encrypted message. Returns whether the encrypted message is valid or not.

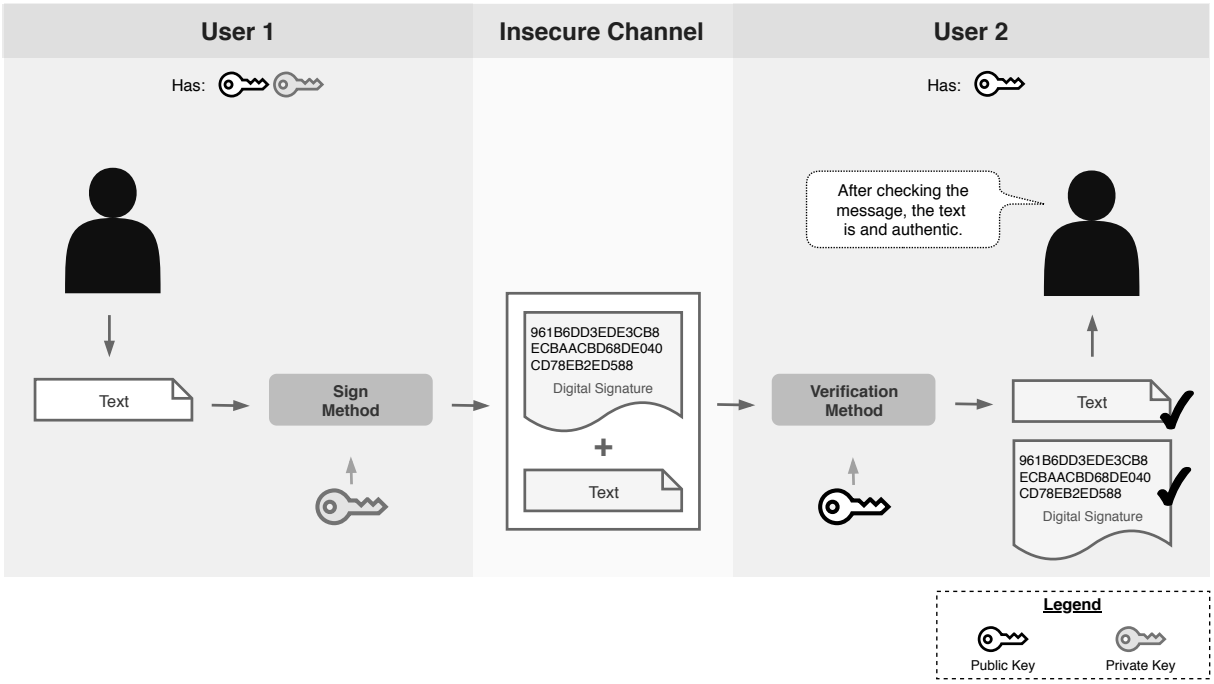


Figure 4 – Operation of digital signatures - Modified from Braga and Dahab (2015)

2.1.4.2 Transactions

A transaction is an interaction between the parties involved (GATTESCHI *et al.*, 2020). The data of a transaction varies by Blockchain implementation and domain. In a financial context, a transaction can be a transfer of an asset between users in the network. In a healthcare context, a transaction may be a laboratory test result. Besides, depending on Blockchain's implementation, transactions may involve smart contracts, which are discussed in section 2.1.4.3.

Transactions must fulfill ACID properties, widely discussed in the database, as follows:

– Atomicity: It is based on the idea of "all or nothing", which means that either all operations in the transaction perform or none at all (RAMAKRISHNAN; GEHRKE, 2003; ELMASRI; NAVATHE, 2015; ÖZSU; VALDURIEZ, 2011);
– Consistency: The consistency of a transaction is its correctness. It means that a correct transaction transforms the system from one consistent state to another consistent state (RAMAKRISHNAN; GEHRKE, 2003);

- Isolation: It ensures that even if multiple transaction operations are interleaved, the effect of executing them is the same as executing all these transactions one after another in some serial order (ELMASRI; NAVATHE, 2015);
- Durability: It guarantees that the effects of a committed transaction are maintained permanently. That is, once a transaction has been committed, its results are permanent regardless of failures such as power outages, crashes, or errors (ÖZSU; VALDURIEZ, 2011).

To perform a transaction, a user sends the following information to the network: i) a sender identifier; ii) his public key and digital signature; and iii) transaction inputs and outputs (YAGA; MELL, 2018). Entries in a transaction make up a list that contains the assets to be transacted. The sender must prove that they have access to the transacted assets using a private key (YAGA; MELL, 2018; NARAYANAN *et al.*, 2016). The outputs of a transaction are commonly the assets' addressees, and the amount each will receive (YAGA; MELL, 2018).

Nodes validate transactions in a few steps, which usually include: i) digital signature validation; ii) inspection of assets, and iii) confirmation that any previous transaction did not spend the asset. This verification is simplified by implementing hash pointers to previous blocks, making it easy to verify the outputs of previous transactions (NARAYANAN *et al.*, 2016). One could notice that this process occurs independently on each node, reinforcing the decentralized characteristic of Blockchain (GREVE *et al.*, 2018).

Figure 5 shows an example of transactions and the use of their properties in cryptocurrencies. It is worth mentioning that, to simplify the example, each block has only one transaction, which is not common in a real scenario. In block 1, there is the creation of assets for Alice. In block 2, there is a division of an asset of value 25 into two smaller assets, one of value 17, transferred to Bob, and another of value 8, which Alice transferred to herself. The second transaction is necessary to make it explicit when there are surpluses in a transaction. That's because after transferring 17 to Bob, Alice remains with 8 out of 25 she had initially. Thus, block 3 is invalid, causing inconsistency because there are 2 assets left, so the sum of the outputs of block 3 must be 17. However, the sum of the outputs of block 3 is only 15, missing 2 to the initial values. Block 3 would be valid if Bob transferred 9 to himself, which is what remained after transferring 9 to Carol. Finally, the owner of traded assets signs each transaction.

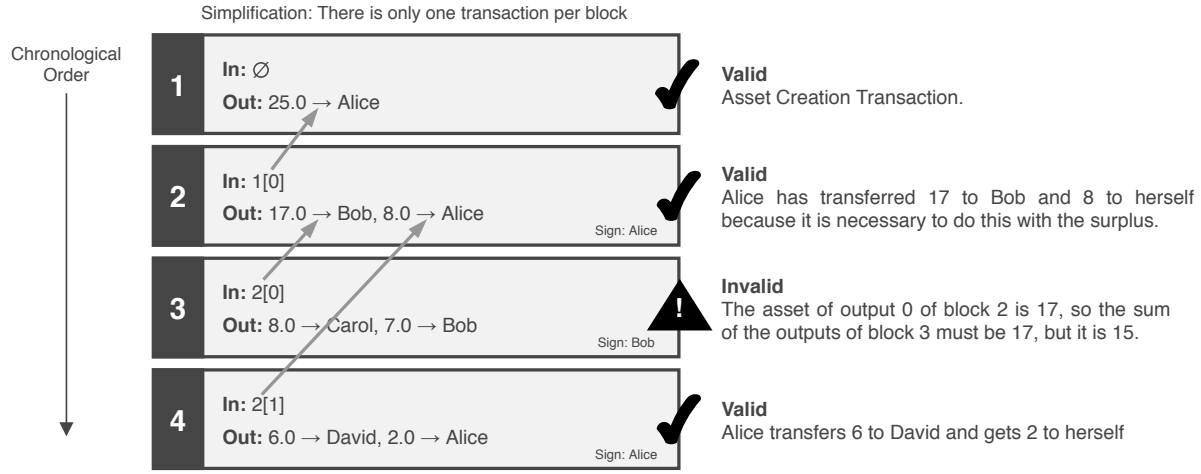


Figure 5 – Example of Asset Transactions on a Blockchain Network - Modified from Narayanan et *al.* (2016)

2.1.4.3 Smart Contracts

Szabo (1994) first defined smart contracts, and the Ethereum project inserted it into the Blockchain context (WOOD, 2014). A smart contract is a protocol that performs the terms of a contract, that is, a programming code that is transacted between network nodes to execute it at some predefined time. Its objective is to satisfy contractual conditions such as payment terms and confidentiality. Besides, it minimizes the need for third parties (GREVE *et al.*, 2018; HUH *et al.*, 2017). Table 4 explains the difference between hand-signed contracts and smart contracts in the following criteria: the language used, who is responsible for executing the contract, and where it is registered.

	Traditional Contract	**Smart Contract**
Language	Juristic	Script
Execution	By Stakeholders or Authorities	Self-executing (at the time specified by the entry)
Register	Notary Office	Blockchain Network

Table 4 – Comparison between traditional contracts and smart contracts - Own authorship

There are several uses for smart contracts. Some examples are calculus, data storage, and automatic asset transfer. However, not all Blockchain networks support the use of smart contracts. Examples of it are Ethereum and Hyperledger Fabric (GREVE *et al.*, 2018; BRAGA *et al.*, 2017).

2.1.4.4 Blocks

Blocks are sets of transactions and other elements in the block header. In some Blockchain networks, such as Bitcoin, it is common for the first transaction of each block to assign assets to the node that created that block. This type of transaction is known as coinbase (BASHIR, 2018). The structure of a block depends on the Blockchain implementation, but there are more similarities than differences. To exemplify the structure of a block, the Bitcoin network block model is explained, as it is widespread in the literature and was defined in Blockchain's original article. According to Yaga and Mell (2018) and Greve et al. (2018), a header has the following fields:

- **prev**: It is a hash pointer to the previous block since the Blockchain is a chained list of blocks. If the block is the first one, it has no value and is called a genesis block;
- **mrkl_root**: It is a pointer to the root of the Merkle Tree in which the block's transactions are. It is because, in Bitcoin, each block has its transactions structured in Merkle Tree. In this structure, transactions are only on leaf nodes, and each has a hash identifier;
- **nonce**: It is an integer to be discovered in solving a cryptographic puzzle during consensus realization. This field is specific to the PoW consensus model, which will be detailed in the subsection 2.1.5;
- **timestamp**: It is the time which the block was added to the Blockchain;
- **target**: Like nonce, it is specific of the PoW model and defines the difficulty for a node to publish a valid block during consensus.

There are other fields in the block: the metadata. However, this information is not transmitted on the network. It is i) the block header hash, which is unique and identifies the block on the network, and ii) the height of the block in the Blockchain, being the first block called genesis, which has a height of 0. The other blocks of the network have the height of the previous block plus one (ANTONOPOULOS, 2017).

There are some steps to validate the blocks. They are: i) to analyze if the block structure is well-formed; ii) check if the block hash is valid; iii) examine whether the block size conforms to network specifications; and iv) validate block transactions. Like transactions, the validation process occurs independently on each network node (GREVE *et al.*, 2018).

2.1.4.5 Ledger

Financial professionals have long used the term ledger. Accountants typically used these paper records to record and track expenditures or revenues on goods and services. The ledgers then went digital and were stored in databases operated by a centralized third party.

Ledgers are not necessarily centralized. They can be distributed to be spread between nodes of a P2P network, as occurs in Blockchains. In this scenario, deploying a distributed ledger eliminates a third party (YAGA; MELL, 2018). Thus, in a Blockchain context, the ledger is a replicated and immutable data structure that keeps track of all blocks validated by network nodes. Given this replication feature, the ledger maintains the global state of the system. Encryption and consensus are essential to ensure ledger authenticity, integrity, consistency, and availability (NARAYANAN *et al.*, 2016; ANTONOPOULOS, 2017; GREVE *et al.*, 2018).

The main reasons for using the distributed ledger are based on safety and reliability aspects of centralized ledger use. Some relevant arguments for using distributed ledgers are:

- Data stored centrally can be lost or destroyed. Thus, users need to believe that the system backup is being performed repeatedly by those responsible for the centralized ledger. In a distributed ledger Blockchain network, each node maintains its ledger replica, which makes data loss difficult (ZHANG; BOULOS, 2020; YAGA; MELL, 2018);
- Centralized ledgers are typically in a homogeneous network, where software, hardware, and infrastructure are often the same. It makes the system more susceptible to attack since a successful attack on one network node also works on any network node. Blockchain networks are heterogeneous in software, hardware, and infrastructure, so there is no guarantee that attacks on one node will work across all nodes (XIE *et al.*, 2020; ZHANG; BOULOS, 2020);
- Transactions in a centralized ledger are not auditable and may not be valid. Therefore, users must trust that each transaction received is valid. However, in a Blockchain, transactions are validated by network nodes, and if any nodes are transmitting invalid transactions, the others detect and ignore (YAGA; MELL, 2018);
- In a centralized ledger, the transaction list may not contain all transactions that occurred. The user must be trusting that all valid transactions received are being added. In a Blockchain network, all accepted transactions are stored in the distributed ledger, as a new block must reference the previous block when it is added. If it does not, the other nodes reject the proposed block (YAGA; MELL, 2018);

– There is no guarantee that a centralized ledger owner did not tamper with previous transactions. On the other hand, Blockchain technology uses cryptographic mechanisms to provide tamper-proof records (YAGA; MELL, 2018);
– A central system may have a security hole. Therefore, the user must believe that necessary security corrections are being made to prevent breaches and theft of personal information. On the other hand, given their distributed characteristics, Blockchain networks do not provide centralized attack points. The main difficulty for such attacks is the existence of honest nodes in the network. If an individual node is not fixed, only that node is affected, not the entire system (YAGA; MELL, 2018; ZHANG; BOULOS, 2020).

2.1.4.6 Client Software

The user uses a blockchain client software, which usually is a web or mobile application, to transact his assets. When used in the context of cryptocurrencies, these systems are called eWallets. These applications implement mechanisms to i) store cryptographic keys securely; ii) sign transactions; iii) encrypt transactions, and vi) transmit data securely. Therefore, this system's main purpose is to manage the client's private key set so that it manipulates its assets. So, the client's assets are associated with the keys he used to sign them, which assures that only the assets owner can manage them. However, if a private key is stolen, the wrongdoer will have full access to all assets controlled by that private key. Therefore, the user is unable to manage their assets if they lose their keys (BRAGA *et al.*, 2017; LI *et al.*, 2017).

There is a flow to add a new transaction on a Blockchain network. Initially, the user creates a new transaction in the client software. Next, the client software sends the transaction to some network node, which validates it for integrity and authenticity. So, it is transmitted to the other nodes of the Blockchain network and is considered pending. There is a consensus, and each node adds the new block containing the proposed transaction to its replica. It is important to note that the client software does not participate in the consensus as it is usually not a node in the P2P network. However, this software is critical for protecting user data and assets (BRAGA *et al.*, 2017; SWAN, 2015; NAKAMOTO, 2008).

The client application is developed to meet the application's logic, respecting the languages and business rules. So, good usability in cryptographic key management is pertinent. It is achieved by using metaphors and clear abstractions of traded assets, enabling the transparent use of encryption and its aspects. Thus, the user does not need to have cryptographic knowledge

to use the client software (BRAGA *et al.*, 2017). However, Krombholz (2017) points out that the main obstacle to the widespread dissemination of Blockchain technology is the high complexity of using these cryptographic systems. Many of these tools are still not clear enough.

2.1.5 Consensus Algorithms

A central point in Blockchain is to decide which node should publish the next block, and that is the purpose of consensus algorithms. Thus, consensus enables participants to agree on their actions for joint decisions, guaranteeing the Blockchain's consistency and progress, even if failures occur (GREVE, 2005). It is related to safety and liveness properties. In permissionless Blockchains, nodes compete to publish the next block for their interests, as rewards are common for those who create new blocks, such as coinbases in the Bitcoin. In permissioned Blockchains, the motivation is to make sure that is the Blockchain network works honestly, as there are no benefits to those who post new blocks (YAGA; MELL, 2018; GREVE *et al.*, 2018).

According to Yaga and Mell (2018), four properties are required for consensus:

- The initial state of the system starts with the genesis block
- The nodes agree on which consensus algorithm is used to define who adds the next blocks
- Each block is chained to the previous block using a hash pointer except the genesis block
- Each block can be independently verified

The consensus model varies by domain. For example, in permissioned networks, there is often some trust between blocks, and so it is not necessary to use a costly consensus algorithm to determine which node adds the next block. On the other hand, in permissionless networks, it is common to use consensus that requires excessive resource consumption to publish blocks, such as PoW. However, Yaga and Mell (2018) explain that using a resource-intensive consensus is not so good. One reason is the high energy consumption since the Bitcoin network consumes energy as the country of Ireland. Therefore, proposing new models that are safe and use less computational resources are common in the literature, such as Proof of Stake (PoS), which was developed to be an option to PoW.

This section describes some consensus models in the literature. Table 5 shows the algorithms discussed, explaining the advantages and disadvantages of each and examples of implementations.

Algorithm	Advantages	Disadvantages	Domains	Example
Proof of work (PoW)	Difficult to perform denial of service by flooding network with bad blocks.	Computationally expensive, high power consumption	Permissionless	Bitcoin
Proof of stake (PoS)	It's not very computationally intensive.	The formation of a stakeholder pool to create centralized power cannot be prevented	Permissionless	Ethereum
Delegated PoS (DPoS)	Elected nodes are economically encouraged to remain honest.	There may be an incentive for elected nodes to accept bribes	Permissionless, Permissioned	Bitshares
Round Robin	Low computational power	Nodes must have a lot of trust with each other	Permissioned	MultiChain
Proof of Identity	Fast confirmation time	It depends on the assumption that the current validation node is honest	Permissioned	POAChain
Proof of Elapsed Time (PoET)	Low computational power	Given speed-of-late latency limits, real time synchronism is impossible in distributed systems	Permissioned	Hyperledger Sawtooth

Table 5 – Consensus algorithms summary - Based on Yaga and Mell (2018)

2.1.5.1 Proof of work (PoW)

Used in permissionless networks, this model proposes that the node which publishes the next block is the one that first solves a computationally costly puzzle. However, while the challenge is difficult to overcome, it is easy to see if the solution is valid. All other nodes quickly validate the proposed block, refusing it if it is not valid (YAGA; MELL, 2018; LI *et al.*, 2017; XIAO *et al.*, 2020). This model was presented in the first article to define Blockchain and is widely discussed in the literature.

In the PoW, each node that wants to propose a new block applies a hash function to that block. The challenge imposes that for a block to be valid, its hash must be equal to or less than a predefined value, named target. Since the nonce is the only field that can be changed in a block without changing the data itself, the node needs to find a correct nonce value to overcome

the challenge. This process is done by brute force, using trial and error, and is designated mining. Consequently, the nodes that perform this process are called miners. The miner which overcomes the challenge sends its block to other nodes, which validate the received block. If the block is valid, each node includes the block in its replica and discards the one it was creating. Otherwise, the block is discarded, and the consensus continues (ZHENG *et al.*, 2017; GREVE *et al.*, 2018; XIAO *et al.*, 2020).

Figure 6 exemplifies the PoW algorithm. Considering that block one is already published in the network and is valid, Figure 6 shows the calculation of the appropriate nonce value for block 2, called mining. As is the rule of this Blockchain network, the hash of a block must start with four zeros to be considered valid. Hence, a nonce value must be found so that the hash of block 2 starts with four zeros. So, the miner tries the value 0 for the nonce of block 2 and verifies it is not valid. It will then increment one and try again, seeing that this value does not satisfy the condition. In attempt 97105, finally, the node finds that a satisfactory value for the nonce is 97104. The node then proposes the block to the other network nodes, which validate and accept the block.

2.1.5.2 Proof of stake (PoS)

PoS is based on the premise that the more resources a node brings into a system, the less likely it is to corrupt the system while keeping the system honest. Resources can be of various kinds, such as a cryptocurrency value, and generally can no longer be spent after being entered into the system. In this algorithm, there is a higher probability that the winning node is the one that invested the most resources in the Blockchain network (YAGA; MELL, 2018; ZHENG *et al.*, 2017; MINGXIAO *et al.*, 2017; ZAMFIR, 2015).

The Ethereum project introduced PoS as an alternative to PoW as it generates excessive energy costs. For example, in 2018, the Bitcoin network spent the same amount of energy as Ireland. PoS is used on permissionless networks and promotes fewer resource expenditures such as time, electricity, and processing than in PoW (ZHENG *et al.*, 2017; YAGA; MELL, 2018). Examples of applications of this model are Blackcoin (VASIN, 2014), Peercoin (KING; NADAL, 2012), and Ethereum Casper (ZAMFIR, 2015).

There are some PoS implementations. In all of them, there is the assumption that the nodes with the most investments are more likely to be chosen to publish new blocks. In one of them, there is a random selection with weights according to each node's assets. For example, if a

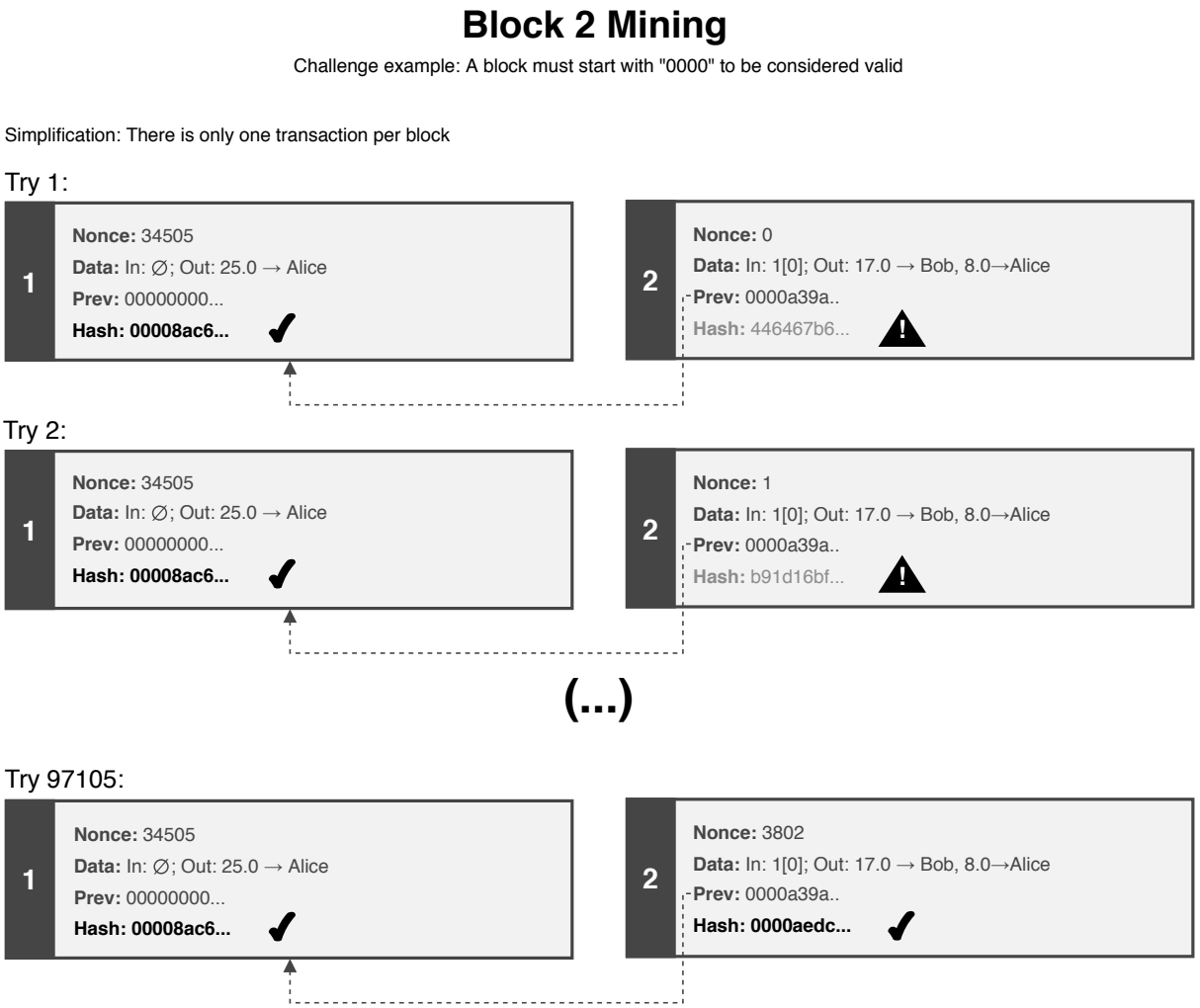


Figure 6 – Second block mining example with simplification of one transaction per block - Own authorship

node has 30% of all network assets, it will overcome 30% of consensus. Another implementation is multi-turn voting: The Blockchain network selects wealthy nodes, which vote on proposed blocks. There may be several turns to decide on a new block (YAGA; MELL, 2018).

2.1.5.3 Delegated Proof of Stake (DPoS)

Larimer (2014) presents a variation of the PoS, named Delegated Proof of Stake (DPoS). In this algorithm, each node votes on one node, delegating block creation to it. Hence, the elected node becomes responsible for creating the next block. It is important to note that each vote has a different weight related to the voter's investment in the network. Thus, the votes of the most resourceful nodes in the network are of greater importance. As for performance, DPoS provides good scalability, providing higher throughput than PoW and PoS algorithms

(MINGXIAO *et al.*, 2017; LARIMER, 2014).

There may be some implementation differences for DPoS. First, some implementations allow a node to vote against a node instead of voting in favor of one. Besides, there may or may not be a financial incentive for elected nodes, designated delegates. It is common for rewards to be given on permissionless networks. Another implementation is to choose a node group instead of just one node. This group takes turns publishing blocks in the round-robin model (WENTING *et al.*, 2017; YAGA; MELL, 2018).

2.1.5.4 Round Robin

Nodes take turns creating blocks in this model. Besides, there is a limit time for a node to publish a new block, losing the turn if it does not respect it. This avoids throughput decrease if some nodes are unavailable. Its advantage is the low energy use and the guarantee that no node will create most blocks (YAGA; MELL, 2018).

This model is used in permissioned Blockchain networks because it needs trust between nodes. If a permissionless network uses round-robin, malicious users can add more nodes to increase their chances of publishing new blocks and subverting the network (YAGA; MELL, 2018).

2.1.5.5 Proof of Identity

Used on permissioned Blockchains, this model correlates nodes that publish blocks to real-world people or corporations' identities. Each publishing node must have its proven and auditable identity, so it bets its reputation to publish new blocks. If the node does something that others do not agree with, it loses its reputation. However, if it does something others agree with, his reputation improves. Reputation matters because the probability of a node publishing a block is directly proportional to its reputation (YAGA; MELL, 2018).

2.1.5.6 Proof of Elapsed Time (PoET)

In a permissioned Blockchain network, each node that wants to publish a new block requests a random wait time from a hardware time counter on its computer. Then, the requesting node is idle for the set period. Then, after waiting, the node creates the next block and sends it to the others. Those nodes that are still inactive become active, validate the block and then the

process is restarted (CHALAEMWONGWAN; KURUTACH, 2018) (CHICARINO *et al.*, 2017). Proof of Elapsed Time (PoET) requires some guarantees as if the choice and wait of random time has happened place properly. If it does not occur correctly, a malicious node is idle for the minimum time and subverts the system (YAGA; MELL, 2018).

2.2 Relational Database

A database is a collection of data that describes the activities of one or more related organizations. For example, in a university domain, a university database might include information from i) entities, such as students, classrooms, and courses; and ii) associations between entities, such as students in courses (RAMAKRISHNAN; GEHRKE, 2003).

A Database Management System (DBMS) is a software developed to support the maintenance and the use of large data collections. Therefore, a DBMS provides operations to insert, update, delete, select data on a system. It must also guarantee the security of the stored data, preventing data loss and prohibiting access to unauthorized data. In addition to these features, the DBMS has others, some of which are: i) ensuring redundancy and concurrency control; ii) providing structured storage for efficient query processing; iii) creating backups and recovering after failures; iv) providing access interfaces for users; v) representing complex relations between data, and vi) ensuring restrictions of integrity (ELMASRI; NAVATHE, 2015; ÖZSU; VALDURIEZ, 2011; RAMAKRISHNAN; GEHRKE, 2003; DATE, 1995).

The relational model proposed by Codd (1970) is based on mathematical concepts, such as set theory and predicate logic. This model emerged with the need to expand data independence in database management systems. Moreover, it provides a set of functions supported by relational algebra to store and retrieve data, which was not provided by the previously available databases. This model is currently widely used, and several commercial solutions implement this model, such as Access, Oracle, MySQL, PostgreSQL, and others.

Figure 7 shows the basic structure of a relational database. The main element that composes it is the relation, called table informally. A relation is made up of one or more attributes with a specific type of corresponding data, such as numeric or textual. Each instance of the scheme is called a tuple or record (EF, 1970).

SQL is the standard query language for Relational Database Management Systems (RDBMSs). Initially named Structured English Query Language (SEQUEL), SQL was designed by IBM Research in the early 1970s and is based on relational algebra and relational calculus.

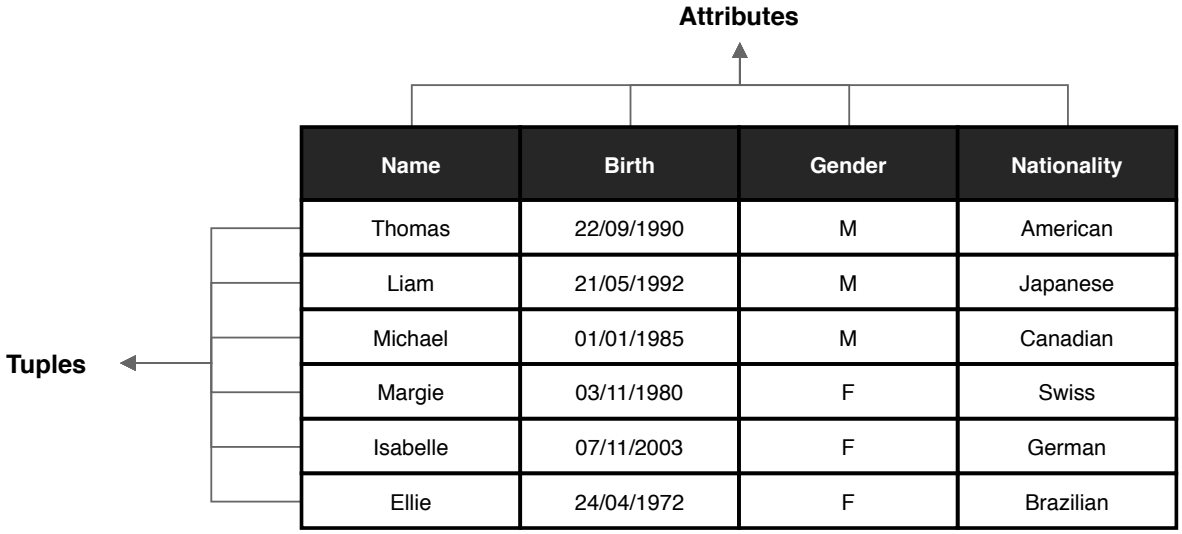


Figure 7 – Example of a relation Student with four attributes - Own authorship

Although there are some variations according to the implementation of the DBMS, the SQL language is regulated by the American National Standards Institute (ANSI) and generally does not present serious problems when porting data from one platform to another (ELMASRI; NAVATHE, 2015; DATE, 1995; EF, 1970).

2.3 E-health

Ross et al. (2016) define e-health as the use of computing, data, or communication technology in health or healthcare aspects. The use of e-health is essential for solving problems faced by healthcare systems, such as the aging population, providing better treatments (ROSS *et al.*, 2016). According to Zhang et al. (2017) and Kassab et al. (2019), e-health applications are solutions that traverse multiple contexts in the health area. Examples of e-health applications are i) a system for monitoring patients' physiological data; ii) software to manage drugs in the hospital infrastructure; iii) electronic medical records; iv) personal health records; and v) health education programs, patient portals, and patient applications (KASSAB *et al.*, 2019; Zhang *et al.*, 2017; EYSENBACH, 2001).

Eysenbach (2001) presented a conceptual framework to address the potential impact and the central factors of e-health. In its seminal article, the ten E's of e-health were proposed. The E's defined by Eysenbach indicate a combination of 'e' that together characterize the term e-health (MELCHIORRE *et al.*, 2018), they are:

– **Efficiency**: It must reduce costs without decreasing quality, avoiding duplicates of data,

for example. Moreover, e-health applications lessen overheads, such as waiting times and encounter times;

- **Enhancing quality**: E-health can improve the quality of healthcare, for example, by providing comparisons between different health providers and directing patient flows to the best quality providers;
- **Evidence-based**: Evidence should support eHealth applications. Rigorous scientific evaluations must prove its effectiveness and efficiency;
- **Empowerment**: Patients have more access to knowledge and their data, which makes care and assistance more patient-centered;
- **Encouragement**: Patient participation becomes proactive. For example, an application that supports monitoring blood glucose levels and helps people with diabetes with diet encourages and raises awareness;
- **Education**: E-health applications can be educational for doctors, through online sources and patients, through health education and preventive information for patients, for example;
- **Enabling information exchange**: E-health applications enable standardized communication between healthcare facilities, professionals and patients;
- **Extending**: Health care must go beyond conventional limits, geographically, and conceptually. Hence, there is a facility to access healthcare services, including in rural areas, for example. These services range from simple advice to complex interventions;
- **Ethically**: It deals with issues related to doctor-patient interaction, such as privacy, informed consent, and equity issues;
- **Equity**: It must be egalitarian, regardless of financial conditions, gender, whether you live in rural or urban areas, age, and illness.

Casado-Vara and Corchado (2019) discuss that e-health systems are currently centralized. It obligates all users, such as patients and healthcare staff, to trust the intermediary that stores the data. Furthermore, healthcare professionals need to trust that their patients have authentic medical records so that they have not altered or falsified anything to obtain medications or treatments illegally.

Kuo, Kim, and Lucila (2017) complement Casado-Vara (2019) and present some benefits of using Blockchain in an e-health domain. The first is decentralized management, which supports applications in which independent stakeholders wish to collaborate without giving up control to a centralizing intermediary. Examples of stakeholders are hospitals, providers,

and patients. The second is that, due to the decentralizing characteristic, there is no single data ownership with full authority to change them. In Blockchain, only the data owner can make changes using cryptographic protocols. The last advantage is related to auditability and immutability, making fraud difficult and enabling a check of all network states. A practical example of it is the inspection of insurance transactions.

3 RELATED WORK

This chapter compares this research with related approaches found in the literature. For each work, there is a discussion of its advantages and limitations. The following search criteria were used to collect some related works: i) work that use Blockchain or related concepts; and ii) work that use RDBs or their properties.

The search for papers from 2015 to 2020 was carried out in the following repositories: ACM Digital Library, IEEE Xplore Digital Library, ScienceDirect, Proceedings of the Brazilian Symposium on Databases, and Journal of Information and Data Management. At the end of the chapter, in section 3.6, there is a comparison between the works collected in four predefined topics.

3.1 R3 Corda

Hearn (2016) describes R3 Corda, a permissioned Blockchain specific to the financial sector, which stores data in relational databases. Corda's goal is to provide Blockchain features, such as immutability, and relational database aspects, such as complex queries using SQL. Moreover, Corda limits access to data in a transaction using need-to-know protocols, which only share data with those directly included in the transactions, not with all nodes on the network. Furthermore, this approach uses the UTXO model, avoiding double-spent. Finally, Corda supports smart contracts, running it on the Java Virtual Machine (JVM). It is limited to validate transactions, accepting or refusing them. That is, another smart contract's possibilities are not practicable, such as connect with a web service and execute transactions in specific future moments.

Corda has some significant correlations to this research, but some points differ. First, the mapping done by Corda is Object-Relational, by annotations defined in the Java Persistence API (JPA), which makes it dependent on that technology. Additionally, there is no data partitioning between databases and Blockchains, but merging the two technologies to create a new approach. Finally, as it is specific to the financial domain, Corda does not support other contexts' applications. The design of the proposed approach in this dissertation is general-purpose, supporting, for example, data from e-health, social networks, and e-commerce.

3.2 A general framework for blockchain analytics

Bartolleti et al. (2017) present a framework for analytics on Blockchain data in the cryptocurrency domain, supporting Bitcoin and Ethereum. The approach is a Scala library to construct a Blockchain view and store it in SQL or NoSQL database. In this approach, there is a representation of Blockchain primitives entities in Scala classes, such as block and transactions. Next, these representations are stored in a database. So, sending analytic queries is available and occurs using the database's interface, which can be SQL language, if the database uses the relational model.

Another feature of the presented framework is that it enables the integration of Blockchain data with data from external sources. For example, it is possible to integrate the Blockchain data with exchange rate data between the traded cryptocurrency and the dollar, supporting conversions and analysis. However, the research does not discuss how to manage data stored in different infrastructures. It focuses on the need to provide ways to simplify Blockchain access using consolidated and widely used technologies. Hence, this framework's main contribution is to provide complex queries and analytics on data from Blockchains.

3.3 Ethereum Query Language (EQL)

Bragagnolo et al. (2018) developed the EQL, which allows users to get data from an Ethereum Blockchain using queries analogous to SQL ones. Without it, to find specific data in Ethereum, it is necessary to access the blocks using a unique identifier or search multiple blocks sequentially, one by one. Moreover, the EQL provides a rich syntax, supporting to specify data elements to search for information spread across multiple records. The study discusses some advantages of using that language, such as describing filters to get information, ordering query results, and limiting the number of results returned. Figure 8 shows an example of query in the EQL.

To fast manipulation of Blockchain data internally, the approach that implements EQL uses index files. The data structure used in the indexes is the Binary Search Tree (BST) since it is efficient for retrieving a range of values in any comparison operation, such as "greater than" and "less than". It is relevant to the cryptocurrencies domain as there is wide use of numerical values. Finally, it is worth noting that the approach creates its indexes automatically, without the need for user configurations.

```
1 SELECT block.parent.number, block.hash,
    block.timestamp, block.number,
    block.amountOfTransactions
2 FROM ethereum.blocks AS block
3 WHERE block.timestamp BETWEEN date('2016-01-01')
    AND now() AND block.transactions.size >10
4 ORDER BY block.transactions.size
5 LIMIT 100;
```

Figure 8 – Example of query EQL - From Bragagnolo et *al.* (2018)

Bragagnolo et al. (2018) use a robust data query language similar to SQL, which is one relevant aspect of the relational model. However, the work does not present hybridism between the relational model and the Blockchain. Nevertheless, it indicates future work that has a strong correlation with MOON. It is proposed to use the language to query data from different sources, not only Blockchains.

3.4 ChainSQL

Muzammal, Qu, and Nasrulin (2019) developed the ChainSQL, a Blockchain-based log database system that aims to provide Blockchain's integrity and fast query processing of distributed databases. The ChainSQL domain is financial institutions geographically separated with a large volume of business transactions. In this approach, the Blockchain stores transactions, while the database stores current data.

Figure 9 shows the architecture of ChainSQL and its workflow. First, it receives a transaction and sends it to a Blockchain node. After consensus, the transaction is forwarded to the database. Hence, there is a synchronization between database and Blockchain, which occurs in two ways: synchronize-at-each-transaction and synchronize-at-each-interval. Some use cases for ChainSQL are i) a multi-active database middleware for connecting a user application with a database; ii) a failure recovery middleware, and iii) an audit middleware to process transaction traces.

In ChainSQL, there are three node profiles. The first one is the complete-node, which participates in the Blockchain and the consensus. The second is a partial-node, which is only interested in transactions associated with itself. The last one is a light-weight client connected to a complete-node for network operations. Moreover, ChainSQL provides an API that supports

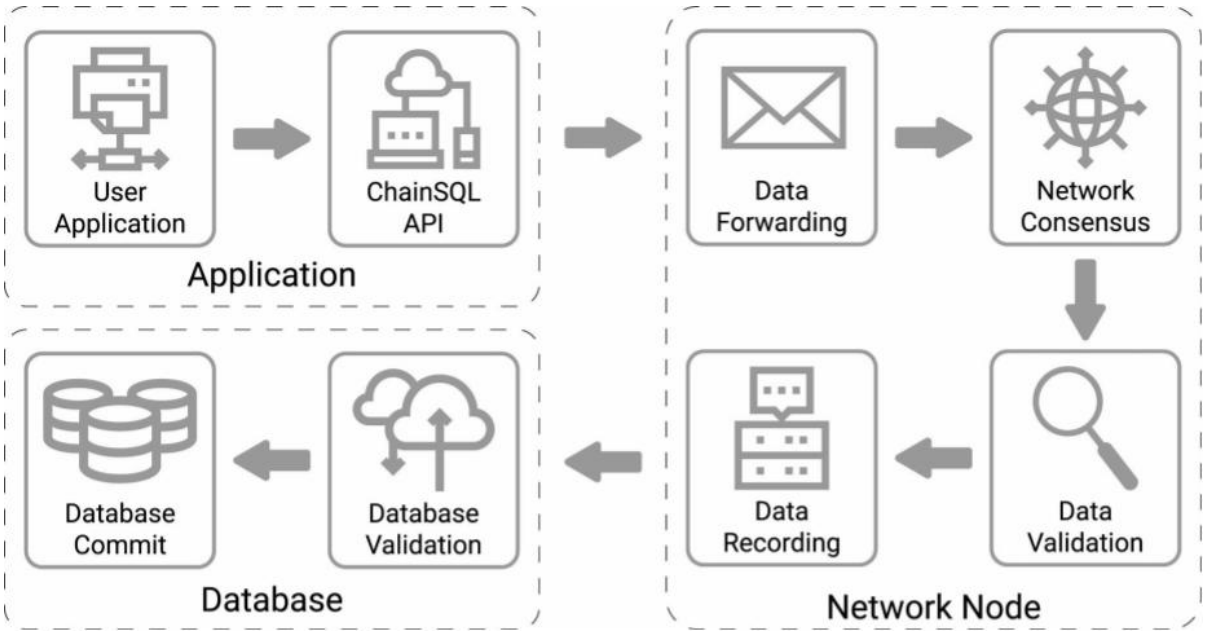


Figure 9 – Architecture of ChainSQL - From Muzammal et *al.* 2019

queries on SQL and JavaScript Object Notation (JSON) with a specific format. In ChainSQL, the Blockchain network is accessed: i) directly by the client for write operations; ii) using a database for reading operations, and iii) configuring a database on a Blockchain node for reading a database or writing on a Blockchain.

Although it has several points similar to ChainSQL, MOON has some significant differences. First, MOON does not propose to perform database-blockchain synchronization since there is no data replication because the purpose is to partition them. It is because keeping the data current in both models is likely to be computationally costly. Second, the article presenting ChainSQL does not discuss Blockchain data indexing or relational model mapping to the Blockchain, which are features of the present research.

3.5 SEBDB: Semantics Empowered BlockChain DataBase

Zhu et al. (2019) proposed SEBDB, an approach that includes the relational data semantics in Blockchains. SEBDB associates each transaction with a tuple that represents one table and uses a language similar to SQL as a communication interface. Moreover, this approach uses RDB to store off-chain-data, that is, that are not in the Blockchain network. Hence, there are no trust aspects in the use of RDBs. Furthermore, SEBDB allows joins between Blockchain data and RDB data, which is called on-off join. SEBDB's architecture is in Figure 10.

SEBDB indexes Blockchain data using files as it is inefficient to scanning blocks

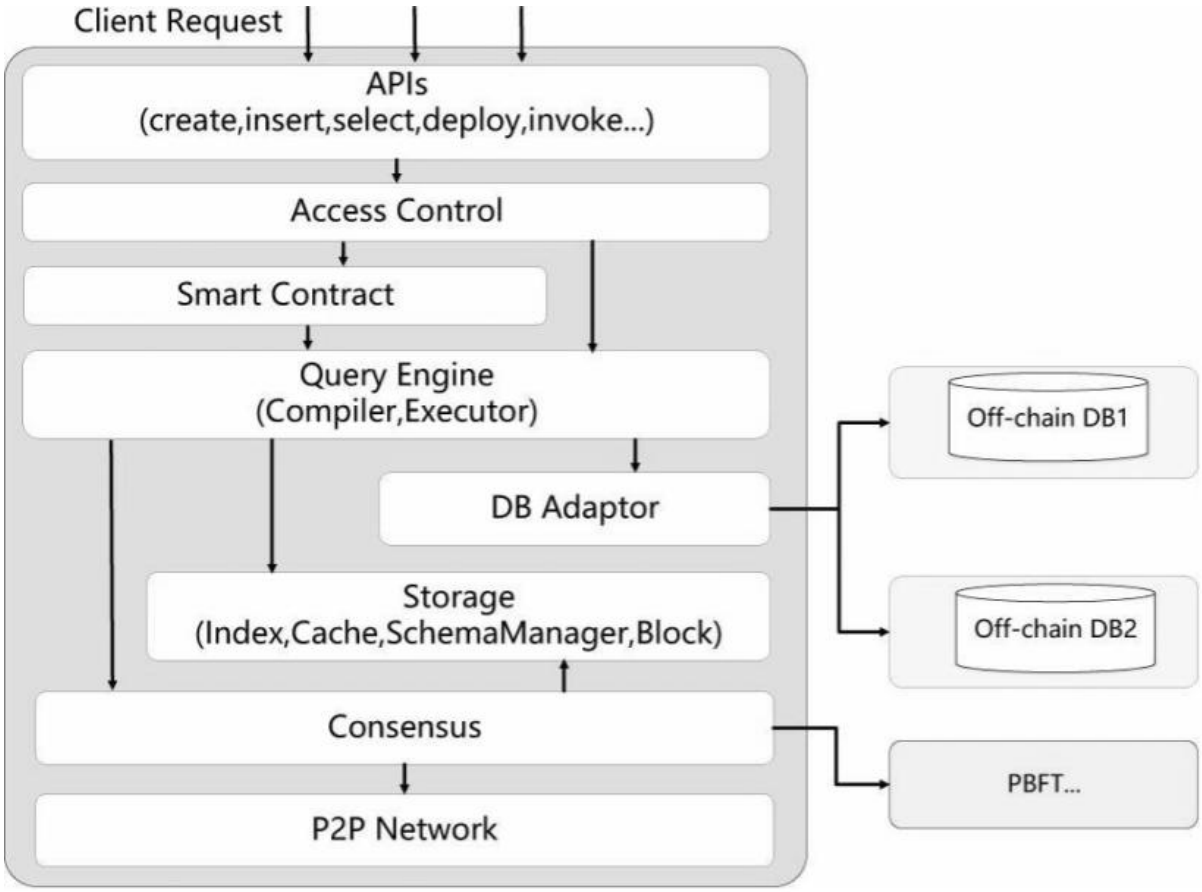


Figure 10 – Architecture of SEBDB - From Zhu et *al.* 2019

one by one in a block with multiple tables. There are indexing mechanisms to speed up data access in three operations. The first uses the Block-level B+-tree Index to get a block from its ID, transaction ID, or timestamp. The second uses Table-level Bitmap Index and consists of getting tuples belonging to the same type of transaction. The third is to obtain transactions under some condition and uses the Layered Index.

Although it has several similar points to the MOON, there are some notable differences between the two approaches. First, there is no database replication in SEBDB because, as the relational model is not part of the network, each node has an RDB with a different data scheme. Consequently, the RDBs schemas are not the SEBDB's focus, as it does not know it. Moreover, SEBDB does not discuss data mapping between the Blockchain and relational models. Finally, the way to index data is different because the MOON uses it in database relations, and the SEBDB uses index files.

Table 6 – Comparison of related works - Own authorship

Work	Data Partition	Use of SQL-like	Blockchain and RDB simultaneously	Well-defined RDB Schema	Blockchain indexing
(HEARN, 2016)		✓	✓		
(BARTOLETTI *et al.*, 2017)		✓			
(BRAGAGNOLO *et al.*, 2018)		✓			✓
(MUZAMMAL *et al.*, 2019)		✓	✓	✓	
(ZHU *et al.*, 2019)	✓	✓	✓		✓
MOON	✓	✓	✓	✓	✓

3.6 Comparison among the related work

Table 6 makes a comparative summary of the five studies presented with the approach proposed in this research. For each work, there is a check if it uses: i) data partitioning between Blockchain and RDB; ii) languages like SQL or related; iii) Blockchain and database concomitantly, that is, there are both infrastructures ready for use, iv) well-defined RDB schema, and v) indexing data in a Blockchain. The third comparison point excludes i) approaches that do not use relational databases but use Blockchain with some characteristics of the relational model, and ii) works that present only extract data from one model for use in another. Finally, it important to note that the choice of criteria is by the main topics covered in this research.

The work presented in this section propose approaches to Blockchain data management with features of the relational model or with the use of RDB itself. None of the work discussed has all the characteristics of the MOON, motivating its creation. SEBDB is more similar to ours. However, the significant difference between both is that, in SEBDB, the data scheme used in RDBs is not known, and each RDB of each node may have a scheme. The purpose of the MOON is to define a single data scheme for all nodes in the entire application. In this way, relational data is essential for the functioning of applications, not just complementary data.

Unlike what was proposed by Hearn (2016), Bartoletti et *al.* (2017), Bragagnolo et *al.* (2018), and Muzammal et *al.* (2019), the MOON proposes the hybrid use of RDB and Blockchain. This feature is advantageous to MOON because it is a general-purpose approach, supporting applications that want store part of their data in Blockchains and others in RDBs, taking advantage of each model's benefits. For example, it makes it faster to insert data that does not need to be in Blockchain, as it will not be necessary to make a consensus for the data to be stored in RDB. However, it is worth considering that applications for specific domains can be optimized for a well-defined and restricted purpose.

3.7 Conclusion

The main work related to the theme of this research were presented in this Chapter. There is a description of each work's main characteristics and its convergent and divergent points in relation to the MOON. The main topics discussed were the partitioning of data between RDB and Blockchain, indexing data in Blockchain, and whether the work has a well-defined relational data scheme.

4 MOON: AN APPROACH TO DATA MANAGEMENT ON RELATIONAL DATABASE AND BLOCKCHAIN

This chapter details the approach proposed in this research, named *An approach to data **M**anagement on relati**O**nal database and bl**O**ckchai**N*** (MOON). The MOON explanation includes relevant aspects, such as architecture, user profiles, algorithms, and workflows. Moreover, this chapter considers the opportunities and limitations identified in the related work discussed in chapter 3.

4.1 MOON Overview

The MOON proposes to manage data in a hybrid form, partitioning it between RDB and Blockchain. Furthermore, the SQL language is used by client applications to communicate to the MOON. Consequently, the client sends insert, update, select, and delete operations to MOON, regardless of whether the data is in a Blockchain or RDB. However, delete operations on Blockchain data are refused because its properties ensure no data removal. Moreover, although it may be a relevant topic, exclusions are not in this research's scope.

Figure 11 briefly shows the context of the MOON. First, data-driven client applications send requests to the MOON using the SQL language. Next, MOON communicates with the Blockchain or the RDB to gets the data requested by the client. Finally, MOON returns it to

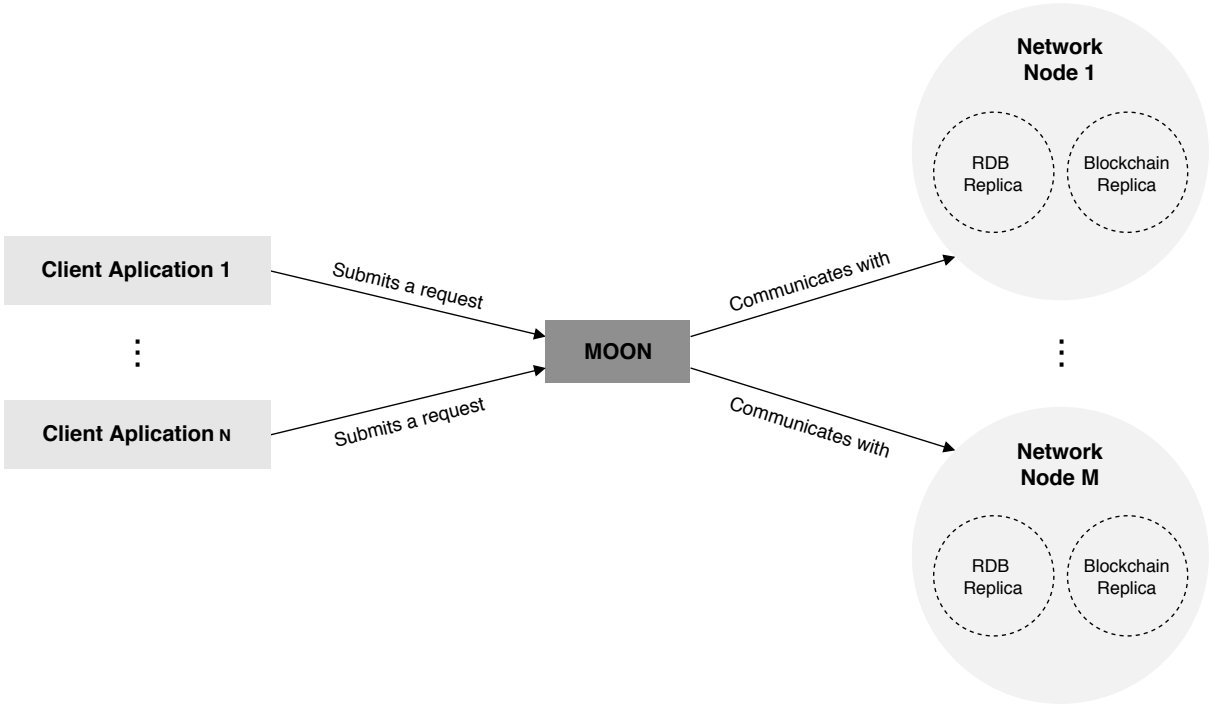


Figure 11 – MOON Overview - Own authorship

the client application. Therefore, the client application communicates in the same way as if it requested data from a traditional RDB. It is important to note that there can exist several nodes with Blockchain and RDB replicas, as well as there can be many client applications.

Adopting SQL as a communication interface between the MOON and its clients has some benefits. Firstly, SQL is well-documented and widely used by developers. Hence, using it avoids the need to learn a new and unconsolidated communication method. Moreover, this language's adoption supports transparency, as the client does not communicate differently for each persistence model used, being MOON responsible for providing this abstraction. Consequently, stakeholders may interact with the system and do not know how and where data is stored. Finally, applications that migrate their data from the relational model to the MOON do not need to do considerable refactorings in their implementations of data access.

4.2 MOON User Profiles

The MOON design considers three profiles of users: a final user, a developer, and a configuration user. The first consumes data through applications, such as a doctor that gets exam results on his mobile application. The second sends requests using an API and needs knowledge about the data schema. However, a developer does not need to know where each data item is, as the MOON provides data access using only SQL.

The configuration user is responsible for configuring and installing the RDB replicas and Blockchain network. Therefore, this user understands how to create and configure the Blockchain and database environments. It is also the configuration user's responsibility to set up the schema definition and configuration files in Figure 13.

It is important to note that these profiles are not necessarily different personas. For example, a Database administrator is an application developer and a configuration user because he uses the API, configures the environment, and knows the data schema. Moreover, this same person can be a customer of the application, consequently, a final user.

4.3 MOON Data Partitioning

The MOON data partitioning is vertical, and its granularity is of entities. Therefore, there is the storage of each entity in one of the two forms of persistence, Blockchain or RDB. An entity is one part of the entire data representing something to model, like tables in the Relational

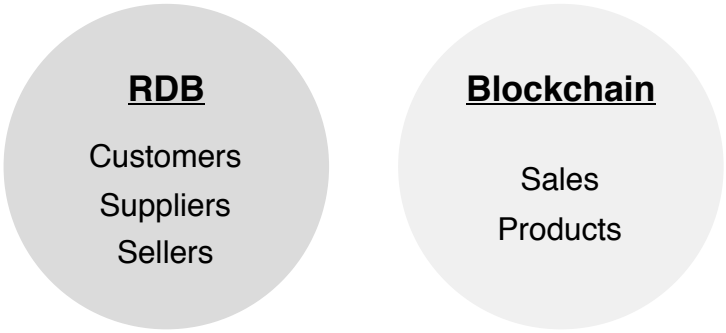


Figure 12 – Example of a possible data partitioning in the e-commerce domain - Own authorship

Model. For example, in the e-commerce domain, there may be entities of customers, suppliers, sellers, sales, and products. There may be the distribution of these entities among the RDB and the Blockchain. Therefore, some of these entities would be in Blockchain, while others would be in RDB.

The configuration user specifies which model of persistence each entity use. To determine the partitioning, one must analyze the application business rules. Data inherent of Blockchain require security, integrity, and no third party. Data best suited to RDBs is mutable and frequently gives better performance. Figure 12 shows an example of a possible partitioning in a hypothetical scenario in the e-commerce domain mentioned in the previous paragraph.

4.4 MOON Architecture

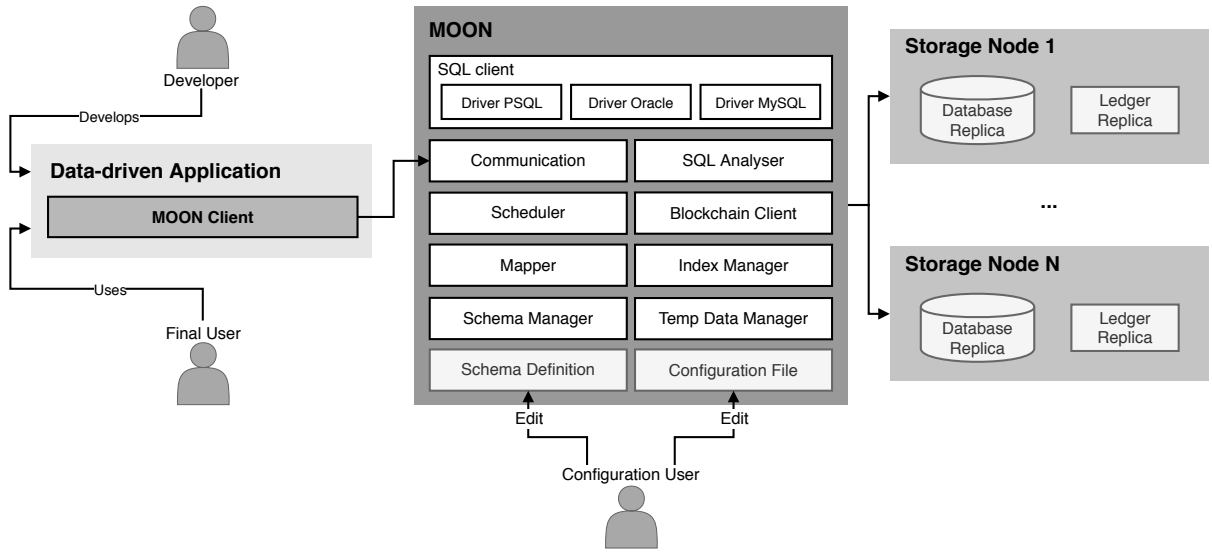


Figure 13 – MOON Architecture - Own authorship

In the architecture presented in Figure 13, the MOON has nine modules:

– **Communication Module**: Receives queries from the MOON Client and forwards them

to the Scheduler Module.

- **Scheduler Module**: Orders the incoming transactions and forwards them to the SQL Client or Blockchain Client. To decide which one to forward, it communicates with the SQL Analyser and Schema Manager modules.
- **Mapping Module**: Maps Relational to Blockchain and vice versa. It makes it possible to execute SQL Queries in Blockchain and the conversion of Blockchain data to the Relational Model, providing transparency to the client application.
- **SQL Client**: Sends received requests to the RDB and has drivers from many DBMSs, such as Oracle and PostgreSQL.
- **Blockchain Client**: Retrieves and stores data on Blockchain.
- **Index Manager**: Retrieves and stores Blockchain index entries.
- **SQL Analyser**: Identifies relevant information on received queries, such as the involved entities and the type of request: select, insert, update, or delete.
- **Temp Data Manager**: Manages temporary data in RDB, which happens when a SQL query has joins and conditionals on updates. There is a detailed explanation about it in sections 4.5 and 4.7.
- **Schema Manager**: Manages the data schema information. It provides information about an entity's persistence model, its attributes, keys, and restrictions, to the other modules. This module takes this information from the Schema Definition File when the MOON is started.

There is the use of two JSON files: the Configuration File and the Schema Definition File. The first has the necessary information to execute the entire system correctly: i) the DBMSs and Blockchain infrastructure used; ii) the addresses of the storage nodes. The Schema Definition File has the following information for each entity: i) its name; ii) its data and types; iii) the persistence model used, and iv) primary keys and foreign keys. Figure 14 shows an example of Schema Definition File.

As shown in Figure 13, the MOON architecture includes a Client Module, which stores the keys used in Blockchain and the credentials of the database. It is used in client applications as a library, receiving SQL queries and sending them to the Communication Module, along with database credentials and Blockchain keys. Hence, the communication is transparent to the developer or the final user, without the need for inserting credentials and keys whenever accessing data.

```
{
    "entities": [
        {
            "name":"entity_a",
            "data":[
                {"name":"id_a", "type":"integer"},
                {"name":"label_a_1", "type":"varchar"},
                {"name":"label_a_2", "type":"integer"},
                {"name":"label_a_3", "type":"varchar"}
            ],
            "persistence":"blockchain",
            "primary_key":[{"name":"id_a"}],
            "foreign_key":[{"name":"label_a_1", "ext_ref_name":"entity_name", "fields":"id"}]
        },
        {
            "name":"entity_b",
            "data":[
                {"name":"id_b", "type":"integer"},
                {"name":"label_b_1", "type":"varchar"},
                {"name":"label_b_2", "type":"integer"},
                {"name":"label_b_3", "type":"varchar"}
            ],
            "persistence":"blockchain",
            "primary_key":[{"name":"id_b"}],
            "foreign_key":[{"name":"label_b_1", "ext_ref_name":"entity_name", "fields":"id"}]
        }
    ]
}
```

Figure 14 – Example of a Schema Definition File - Own authorship

4.5 Mapping

The use of SQL introduces a relevant point. It is imperative to map the SQL requests to Blockchain because the Blockchains design does not support relational data storage. Hence, there is a change in the structure of the data received on request to store it in Blockchains, maintaining its semantics. Therefore, one of the MOON purposes is to transform each tuple of the relational model to text in JSON and then store it in Blockchain. However, when retrieving Blockchain data, the MOON does the opposite work, transforming JSON data to the tuple format.

Figure 15 shows an example of mapping for retrieving data in an e-health domain application. It illustrates how MOON works for selects on Blockchain. First, a client sends a select request to the MOON, which is represented by Arrow 1. Next, Arrow 2 denotes the MOON sending a request to the Blockchain using its API. Next, the MOON receives the requested data in JSON format and convert it to tuple format, being the arrows 3 and 4, respectively. It is important to note that the response returned by the MOON to clients is always in the format of tuples, making the interaction the same as a traditional database.

When the MOON receives a join between relational and Blockchain entities, a temporary table is used to store Blockchain data. It is worth noting that these tables are virtual, not materialized, decreasing the overhead. Therefore, MOON obtains data from the Blockchain, mapping it to the relational model, and then stores it in the temporary table.

Figure 16 illustrates how the join between entities that are in Blockchain and RDB

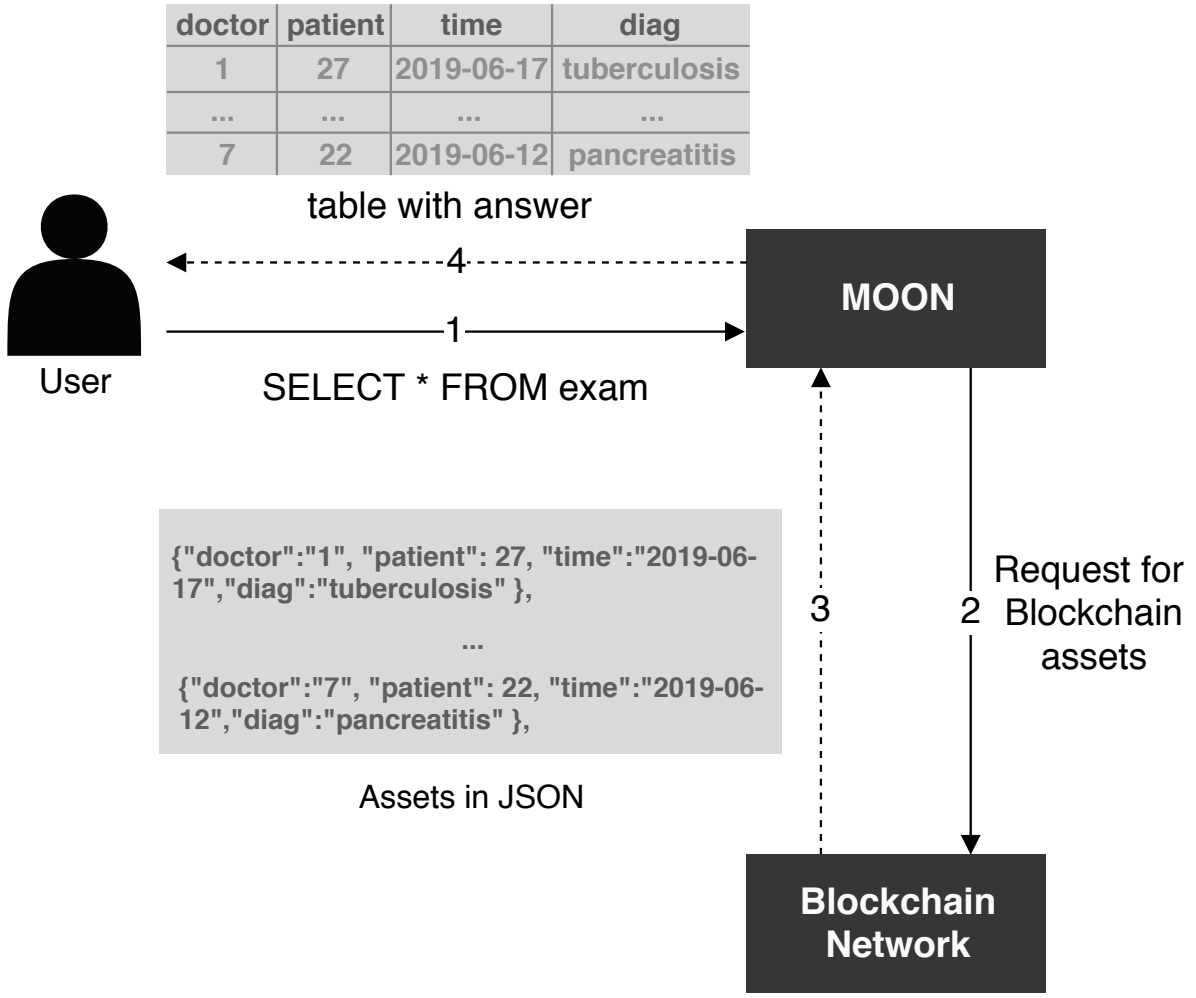


Figure 15 – Mapping from Blockchain to the relational model - Own authorship

works. First, similar to Figure 15, the MOON retrieves Blockchain data. Next, the MOON maps the data to create and store it in a temporary table in the RDB, using it to make joins. Then, the database returns a response in tuple format to the MOON, which forwards it to the client. This strategy is motivated by the use of efficient join implementations of DBMSs, although it introduces a delay in creating temporary tables.

4.6 Indexing

Indexes are needed to resolve discrepancies between Blockchain and RDB. In Blockchains, the record identifier is its hash, while in RDBs other forms of data identification are used, such as numeric or text, adopted as the primary key. Consequently, indexing Blockchain data is necessary to match the identifier defined by the Configuration User and the hash identifier used in the Blockchain network.

Blockchains generally provide only sequential searches or direct data access, using a

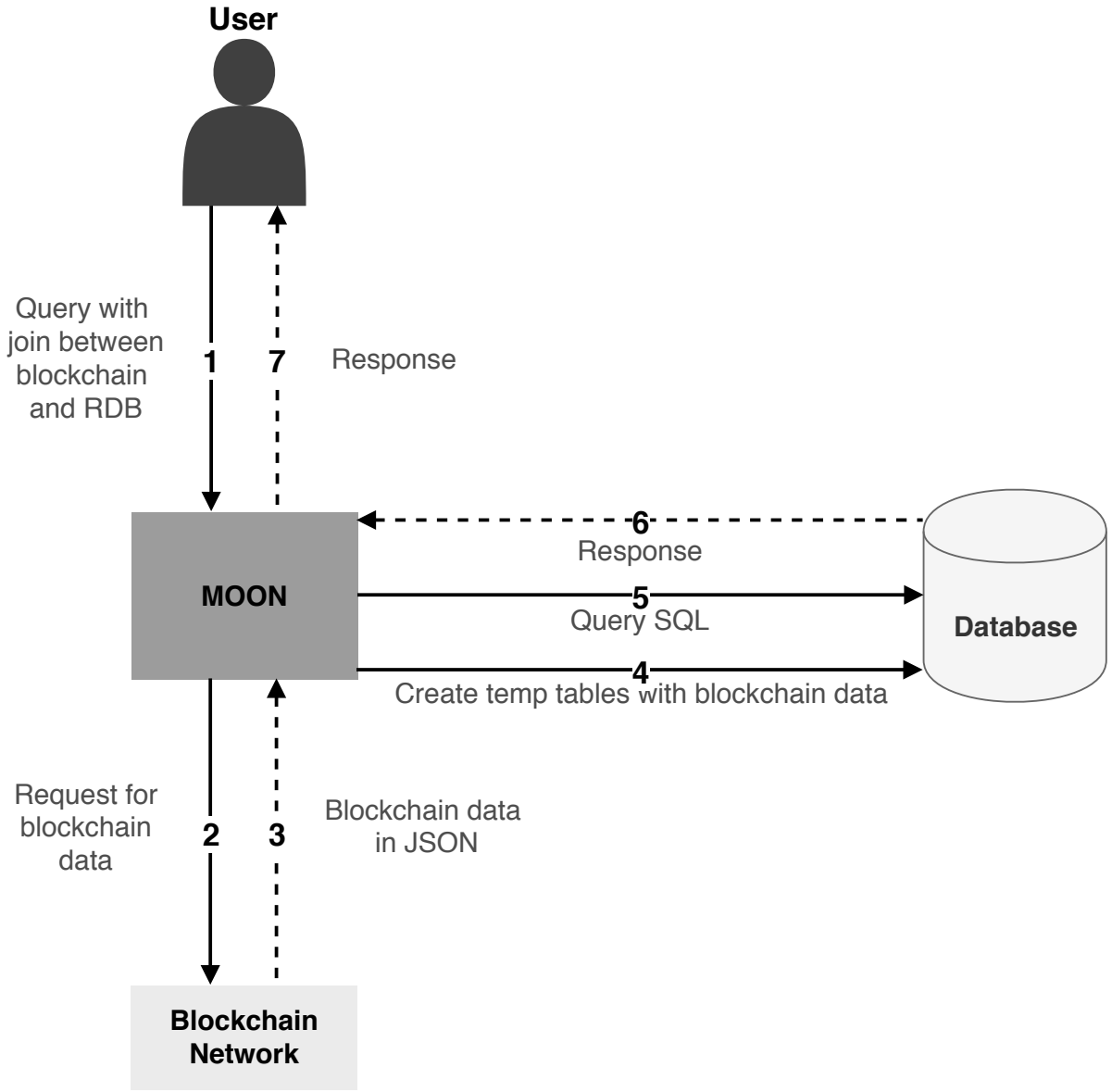


Figure 16 – Data retrieval with join between Blockchain data and database data - Own authorship

hash identifier. Inspecting block by block sequentially to find specific data is costly. Therefore, indexing Blockchain data, it is possible to check the requested data, get its index entry, and then quickly retrieve it from the Blockchain network. Consequently, it makes faster retrieving specific data from Blockchain, although it increases a small delay in inserts.

The logical separation of data is another reason for indexing Blockchain data. In contrast to databases, Blockchains do not divide data logically in, for example, clients, orders, and products, in an e-commerce domain. It probably slows down the data retrieval on Blockchain, as it is necessary to verify all blocks' data or index entries to find specific data, including that in entities not related to the requested data. Therefore, MOON creates one index structure for each entity, separating them logically. That is, each index structure has only entries from one entity.

Consequently, retrieving specific entity data becomes faster as it is not necessary to access all data and select requested ones.

The structure of the MOON's Blockchain index uses the relational model. Therefore, each entity on Blockchain matches one index relation in the RDB, where each tuple is an index entry. Moreover, each index relation has the following information: i) the id of the data, defined as the primary key on Schema Definition File; ii) the data hash on Blockchain, and iii) the height of the block where is the data. Utilizing the third, data recovery is faster, as several requested data can be in the same block. Moreover, using indexing management in RDBs gives benefits from the efficient implementations provided by DBMSs, for example, concurrency control and optimized data retrieving.

Indexing Blockchain data on databases introduces a security point. Someone accessing the database can tamper with index entries so that some queries do not return all the data they should. Therefore, although the data is in the Blockchain, it may not be returned if its index entry was removed. Hence, the MOON implements a function to investigate discrepancies among index entries and Blockchain. It does not get Blockchain data using index entries, but querying each block and checking if all data are in the index entries. If there are inconsistencies, the function returns it.

Figure 17 shows the main flow of Blockchain data retrieval using indexes, without considering queries with conditionals. The dashed arrows correspond to returns of calls. First, the Communication Module receives a request (1) and sends it to the Scheduler (2). Following, the Scheduler interacts with the SQL Analyser (3), identifying the involved entity and the request type, and with the Schema Manager (4), getting the persistence model for the entity. Next, the Scheduler sends the request to the Blockchain Client (5), which recovers the hashes of the requested entity data to the Index Manager (6). It is important to note that the Index Manager obtains hashes in the database (7). Later, the Blockchain Client gets the data from Blockchain Network (8) and formats the response calling the Mapper Module (9). Finally, the Blockchain Client Module returns the response to the Scheduling, which returns it to the Communication Module. These returns have been omitted, simplifying the Figure.

4.7 Algorithms

This section describes the algorithms for the main features of the MOON. There is a explanation of how the approach works, considering its core topics: i) receiving requests

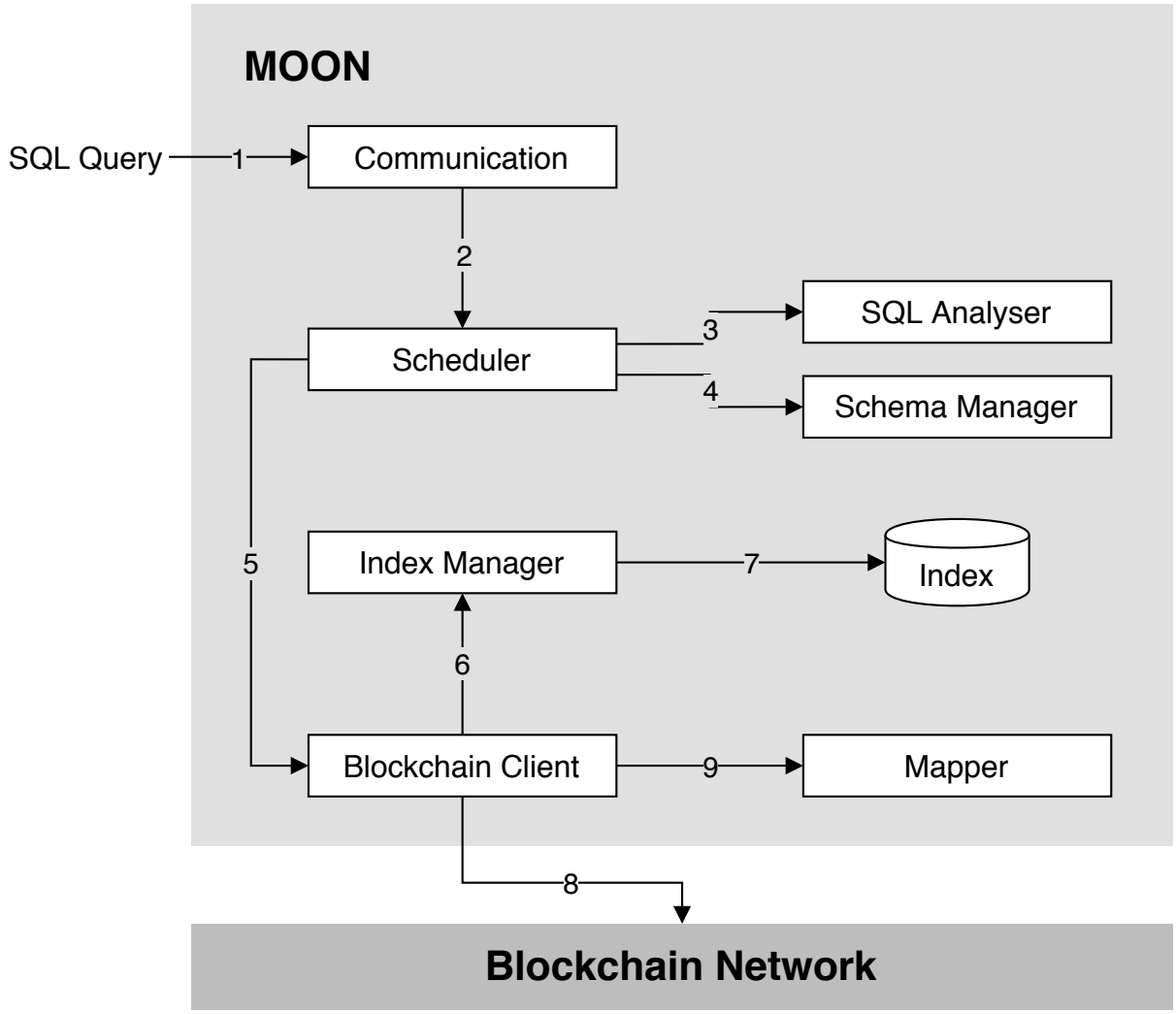


Figure 17 – Step by step of a select request in MOON for Blockchain data - Own authorship

(Algorithm 1); ii) handling requests for data that are only in the RDB (Algorithm 2); iii) handling of inserts, selects and updates in data that are only in the Blockchain (Algorithms 3, 4, and 5, respectively), and iv) processing of joins in data that are in Blockchain and RDB (Algorithm 6).

Algorithm 1 shows how the MOON handles requests according to the context. Initially, the MOON waits for requests and attends one by one. Next, using the SQL Analyser and Schema Manager modules, the MOON recognizes which entities are involved in the request and their persistence models. Then, if all the entities involved use the relational model, the request is sent to ClientSQL. Oppositely, it is sent to the Blockchain Client module. Regardless of the client module chosen, the information about the involved entities, and the operation type are sent along with the request. Finally, CLientSQL or Blockchain Client returns the response, which is returned to the client. If an exception occurs during execution, the client receives an error.

The Algorithm 2 shows how the SQL Client Module works. There is the use of

Algoritmo 1: MOON Receiving Requests

```
Input: Request request, int request_type
1:  wait for request
2:  list involved_entities ← SQLAnalyser.check_involved_entities(request);
3:  int entities_location ← SchemaManager.check_persistence_model(involved_entities);
4:  int request_type ← SQLAnalyser.get_request_type(request);
5:  if entities_location == DATABASE_ONLY then
6:      if request_type == SELECT then
7:          tuples result ← ClientSQL.solve(request, request_type);
8:          return result;
9:      else
10:         int affected_rows ← ClientSQL.solve(request, request_type);
11:         return affected_rows;
12:     end if
13: else if entities_location in [BLOCKCHAIN_ONLY, BOTH] then
14:     if request_type == SELECT then
15:         tuples result ← ClientBlockchain.solve(request, request_type, involved_entities);
16:         return result;
17:     else
18:         int affected_rows ← ClientBlockchain.solve(request, request_type,
            involved_entities);
19:         return affected_rows;
20:     end if
21: end if
```

this module whenever the requests demand entities only in the RDB. First, it reads the RDB credentials from the request DBMSs to open a new connection: i)password; ii)username; iii)address; and iv)database name. Next, if the request type is select, the SQL Client Module sends it to the DBMSs, receives the requested tuples, and returns it. Otherwise, if the query is not select, it receives the number of affected rows and forwards it back to the client. Finally, if an error occurs during execution, it is returned.

Algoritmo 2: MOON Handling Requests From Entities in the RDB

```
Input: Request request, int request_type
1:  DbInfo db_credentials ← get_db_info(request);
2:  if request_type == SELECT then
3:      tuples result_tuples ← read_dbms(request, db_credentials);
4:      return result_tuples;
5:  else
6:      int affected_rows ← write_dbms(request, db_credentials);
7:      return affected_rows;
8:  end if
```

Algorithm 3 shows how the MOON handles inserts for entities in Blockchain. First, the Client Module gets the credentials from request to open a new connection with the blockchain network: i) the node address; ii) the public key, and iii) the private key. Next, using the Mapper module, it obtains two lists, one with the values in the insert query and another with the attributes' names. Later, there is the verifying of the two lists, checking if all the mandatory attributes are there and if their values are adequate. If they are correct, an empty JSON object receives the values, attributes, and the entity name. The MOON stores this object in the Blockchain and gets the following data: i) if the insertion was successful; ii) the hash of the new Blockchain data; iii) the number of affected data, and iv) the block height that the data was inserted. Finally, the Index Manager Module creates new index entries for the new data, and the client receives the number of inserted data.

Algorithm 4 shows how MOON handles select queries from entities in Blockchain. Like the Algorithm 3, there is an obtaining of credentials, and a new connection is opened with a Blockchain node. Next, the Blockchain Client module uses the SQL Analyser module to check if the select query has conditionals. If it does not, the MOON gets all data from each requested entity, selects of the requested columns, maps the data to tuples, and returns it.

Algorithm 4 includes handling selects with conditionals that use the Data Temp Manager module. First, there is the creation of a temporary table for each Blockchain entity in the request. Next, the Blockchain data is retrieved and inserted in the temporary tables, so the same data in Blockchain is temporarily in the RDB. It is worth noting that the Blockchain retrieval data uses indexing, represented by the method *get_data_from_entity*, in the Index Manager Module. Finally, the query is executed by DBMSs, and the MOON returns it to the client.

As shown in Algorithm 5, updates of data in Blockchain starts getting the query that selects the data that will be updated. The Mapper module extracts the WHERE clause from the original request to create this select query. If the WHERE clause does not exist, the query returned by Mapper is a select without the WHERE clause, selecting all entity data. Later, the sequence of steps performed is like Algorithm 4, creating a temporary table. However, the data in the temporary table has an attribute with its hash on the Blockchain network. Next, the MOON sends the update query to the RDB, gets the updated data, maps it to JSON, and sends it to the Blockchain. It is important to note that the updated data are new inserts that reference their previous ones. Finally, there is an update of the Blockchain indices.

Algorithm 6 shows how the MOON handles queries that require RDB and Blockchain

Algoritmo 3: MOON handling Insert Requests in Entities in Blockchain

Input: Request *request*, str *entity*;

```
1:  BcInfo bc_info ← get_bc_info(request)
2:  list values_received ← Mapper.get_values(request);
3:  list attributes_names ← Mapper.get_attributes(request);
4:  json json_file ← JSON empty file;
5:  if attributes_names is in SchemaManager.get_attributes(entity) then
6:    if values_received are valid then
7:      int count ← 0;
8:      json_file.insert("entity": entity);
9:      while count < length(values_received) do
10:       json_file.insert(attributes_names[count]:values_received[count]);
11:       count++;
12:     end while
13:     ResponseWrapper result_bc ← send_to_blockchain(json_file,
        bc_info.get_pub_key(), bc_info.get_priv_key(), bc_info.get_address())
14:     bool success ← result_bc.get_success();
15:     str id_data_bc ← result_bc.get_id_data_bc();
16:     int height ← result_bc.get_block_height();
17:     int number_affected_data ← result_bc.get_number_affected_data();
18:     if success then
19:       IndexManager.store_new_index_entry(id_data_bc, json_file, height);
20:       return number_affected_data;
21:     else
22:       return ERROR;
23:     end if
24:   else
25:     return ERROR;
26:   end if
27: else
28:   return ERROR;
29: end if
```

entities. The algorithm shows the Client Blockchain Module flow that starts getting the database credentials and Blockchain keys. First, there is the check which entities are on the Blockchain, which is needed to create temporary tables. Later, for each table, SQL queries for creating a temporary table and inserting data are obtained. Next, the Data Temp Manager Module executes inserts and the query in the DBMSs. Finally, the Blockchain Client receives the response, and it is forwarded between the modules until that reaches the client.

Algoritmo 4: MOON handling Select Requests in Entities in Blockchain

```
Input: Request request, list involved_entities;
1:  BcInfo bc_info ← get_bc_info(request)
2:  bool has_conditionals ← SQLAnalyser.check_has_conditionals(request);
3:  if not has_conditionals then
4:      list data_requested ← empty list;
5:      for each entity in involved_entities do
6:          data_requested.insert(get_data_to_entity(entity, bc_info.get_pub_key(),
            bc_info.get_priv_key(), bc_info.get_address()));
7:      end for
8:      list selected_columns ← select_requested_columns(request, data_requested);
9:      return Mapper.json_to_relational(selected_columns);
10: else
11:     DbInfo db_credentials ← get_credentials(request);
12:     list sql_list ← empty list;
13:     for each entity in involved_entities do
14:         str sql_create_temp_table ←
            DataTempManager.generate_sql_to_create_temp_table(entity);
15:         sql_list.insert(sql_create_temp_table);
16:         list entity_index_list ← IndexMagager.get_index(entity);
17:         data_entity ← get_bc_data(entity_index_list);
18:         sql_list.insert(DataTempManager.generate_insert_temp_data(data_entity, entity));
19:     end for
20:     DataTempManager.execute_queries(sql_list,db_credentials);
21:     tuples response ← DataTempManager.execute_queries(request,db_credentials);
22:     return response;
23: end if
```

4.8 Conclusion

This chapter presented an approach to managing data in a hybrid way, between RDB and Blockchain. It receives SQL requests regardless of the persistence model used for a data entity. The three user profiles of the approach were defined: i) Developer, ii) Final User, and iii) Configuration User. There was a discussion about the nine modules and the two files that make up the approach, explaining its main features. Moreover, there was an explanation about how the MOON mapping inserts relational data in Blockchain and converts Blockchain data to the relational model. An important aspect of blockchain data recovery is indexing, which was designed to logically separate data and make data searches more agile. At the end of the chapter, the algorithms detail MOON's features and how it behaves in different situations.

Algoritmo 5: MOON Handling Update Requests in Entities in Blockchain

Input: Request *request*, list *entity*;

```
1: BcInfo bc_info ← get_bc_info(request);
2: DbInfo db_credentials ← get_db_info(request);
3: str select_request ← Mapper.generate_select_from_update_request(request);
4: list list_data_selected ← ClientBC.execute(select_request, bc_info);
5: str sql_create_temp_table ←
   DataTempManager.generate_sql_to_create_temp_table(request);
6: str sql_inserts_temp ←
   DataTempManager.generate_sql_to_insert_temp_table(list_data_selected);
7: DataTempManager.execute(sql_create_temp_table);
8: DataTempManager.execute(sql_inserts_temp);
9: DataTempManager.execute(request);
10: list data_updated ← Mapper.to_bc_format(DataTempManager.get_data_entity(entity));
11: ResponseWrapper result_bc ← send_to_blockchain(data_updated,
    bc_info.get_pub_key(), bc_info.get_priv_key(), bc_info.get_address())
12: bool success ← result_bc.get_success();
13: str id_data_bc ← result_bc.get_id_data_bc();
14: int height ← result_bc.get_block_height();
15: int number_affected_data ← result_bc.get_number_affected_data();
16: if success then
17:     IndexManager.update_index_entry(id_data_bc, data_updated, height);
18:     return number_affected_data;
19: else
20:     return ERROR;
21: end if
```

Algoritmo 6: Handling Select Requests with Blockchain and RDB Entities

Input: Request *request*, list *involved_entities*;

```
1: BcInfo bc_info ← get_bc_info(request);
2: DbInfo db_credentials ← get_db_info(request);
3: list entities_to_create_temp_data ← SchemaManager.get_bc_entities(involved_entities);
4: list sql_to_execute ← empty list;
5: for entity in entities_to_create_temp_data do
6:     sql_to_execute.insert(DataTempManager.generate_sql_to_create_temp_table(entity));
7:     data_entity ← get_bc_data(IndexMagager.get_index(entity));
8:     sql_to_execute.insert(DataTempManager.generate_insert_temp_data(data_entity,
       entity));
9: end for
10: DataTempManager.execute(sql_to_execute);
11: tuples response ← DataTempManager.execute(request);
12: return response;
```

5 EXPERIMENTAL EVALUATION

This chapter explains the experimental evaluations carried out to validate the MOON. Section 5.1 presents the use of the approach in the context of e-health, while Section 5.2 exposes the experiments carried out using two benchmarks known to the scientific community.

5.1 Using Real Data in an E-health Domain

This experimental evaluation uses an application in the e-health domain, in the context of clinical laboratory tests. It was used a real dataset with consolidated data from blood samples: i) glucose; ii) insulin; iii) leptin; iv) adiponectin; v) resistin; vi) MCP-1. Originally developed to assess breast cancer cases, the dataset is from a hospital located in Coimbra, Portugal, and has data collected between 2009 and 2013 (PATRÍCIO *et al.*, 2018).

As shown in Figure 19, the application has four entities: i) Patient; ii) Doctor; iii) Laboratory worker; iv) Exam. Table 7 defines the requirements for the application. The Patient entity has patient data, which can be changed, such as name, address, and gender. The doctor entity has the personal and professional data about a doctor, for example, name and specialty. Moreover, the doctor requests exams and sees the results. Laboratory workers are responsible for quality control and release exams, signing for the exams, and can be biologists and biomedical, for example. One Laboratory Worker cannot refute that he issued the result of an exam and cannot modify the exam result after its publication.

Table 7 – Application Requirements - Own authorship

ID	Description	Details
R1	Manage Patients	Insert, Select, Update, and Delete Patients
R2	Manage Doctors	Insert, Select, Update, and Delete Doctors
R3	Manage Laboratory Workers	Insert, Select, Update, and Delete Laboratory Workers
R4	Request Exam	A Doctor requests an exam for a patient
R5	View the Exam Result	The doctor see the results
R6	Issue an Exam Result	A laboratory worker issues the result of an exam

Using the MOON, it is necessary to partition the entities, identifying which ones are more suitable to the relational model and which are more related to Blockchain. The entities Patient, Doctor, and Laboratory Worker need to be in RDB since they receive inserts, updates, and deletes, according to requirements R1, R2, and R3 in Table 7. Nevertheless, the entity Exam needs to be in Blockchain, satisfying the requirements R4, R5, and R6.

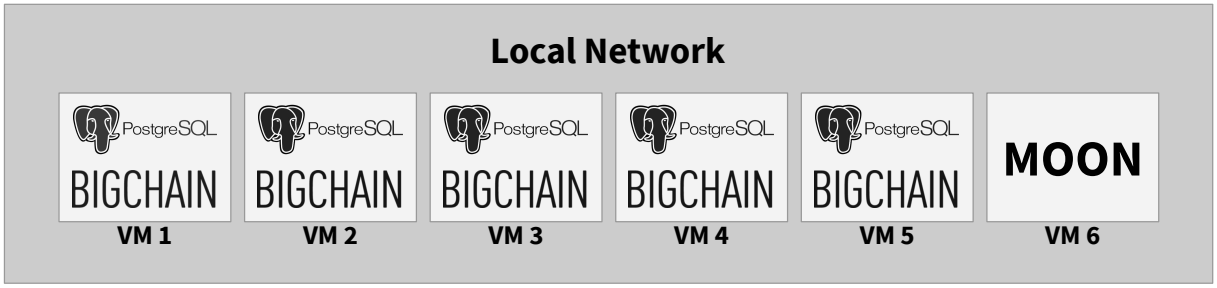


Figure 18 – Experiment setup - Own authorship

To this evaluation, there are three scenarios with the same data schema. First, the four entities in Figure 19 are in the RDB, without using Blockchain and MOON. In the second scenario, there is the use of the MOON, partitioning data between RDB and Blockchain, as discussed in the last paragraph. Third, Blockchain stores all data from the four entities, without using the MOON and RDB. Hence, the third approach uses a sequential search, checking block-to-block to find specific data. Updates get data sequentially, select the relevant ones, change them, and send an insert with the changed data to the Blockchain.

As shown in Table 8, the workload consisted of using six varieties of queries, which represents 4156 executed totally. The character $ followed by a number indicates the different values utilized throughout the experiment. It is relevant to highlight that inserts for laboratory workers and patients were performed, but it is not in Table 8 because their context is very similar to Q5, with much closer results. Finally, two metrics were used to compare the three scenarios: i) the mean of response times; and ii) the correctness, that is, the number of responses without errors and inconsistencies.

As seen in Figure 18, there was used six Virtual Machines (VMs) virtualized from a server with processor Intel(R) Xeon(R) E5645 @2.40GHz. Moreover, the experiments were performed in a local network, which does not introduce a significant network delay. Having DBMSs and a Blockchain node installed, five VMs composed the storage nodes network, each of which had 1GB RAM, 20GB of storage, and Ubuntu 18.04 LTS. The sixth VM had the same previous settings and was used to host the MOON.

Some technologies supported the evaluation. For managing relational data, there was a use of the PostgreSQL DBMSs since it is a well-documented technology. The replication model used was master-master, with the strong consistency model. The Blockchain infrastructure used was BigchainDB[1] because i) it has a well-documented API; ii) it is robust general-purpose and open-source; iii) it is flexible, accepting many data formats, such as JSON. Furthermore,

[1] BigchainDB. Available at: https://www.bigchaindb.com/. Accessed: March 27, 2020.

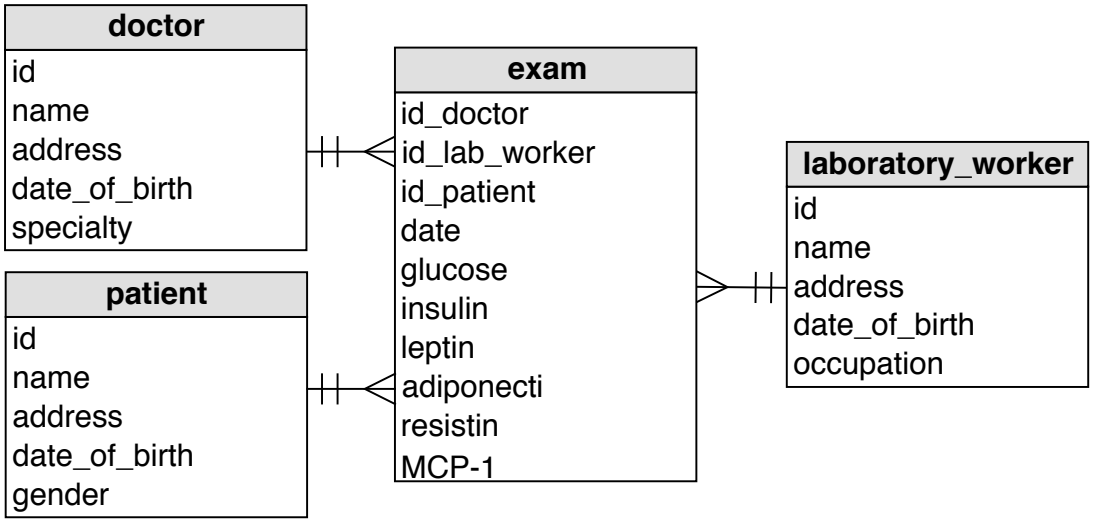


Figure 19 – Data schema used in the case study - Own authorship

Table 8 – E-health used queries - Own authorship

ID	Query
Q1	SELECT * FROM exam
Q2	SELECT * FROM exam, doctor WHERE exam.id_doctor = doctor.id
Q3	UPDATE laboratory_worker SET name = $1 WHERE id = $2
Q4	UPDATE exam SET glucose = $1, insulin = $2, leptin = $3, adiponectin = $4, resistin = $5, MCP-1 = $6 WHERE id_doctor = $7 AND id_lab_worker = $8 AND id_patient = $9 AND date = $10
Q5	INSERT INTO doctor VALUES ($1, $2, $3, $4, $5)
Q6	INSERT INTO exam VALUES ($1, $2, $3, $4, $5, $6, $7, $8, $9, $10)

the Mimesis framework created fictitious personal data of doctors, patients, and laboratory workers, as it was anonymized of real data. Finally, the MOON was implemented in the Python programming language, providing integration with BigchainDB and PostgreSQL.

5.1.1 Results

The results of Q1, Q2, Q3, Q4, Q5, and Q6 are presented by the Figures 20, 21, 22, 23, 24, and 25, respectively. For each query, there are results for the three scenarios previously presented: i) data stored only in the RDB (Scenario 1); ii) data managed using MOON (Scenario 2); iii) data stored only in the Blockchain (Scenario 3).

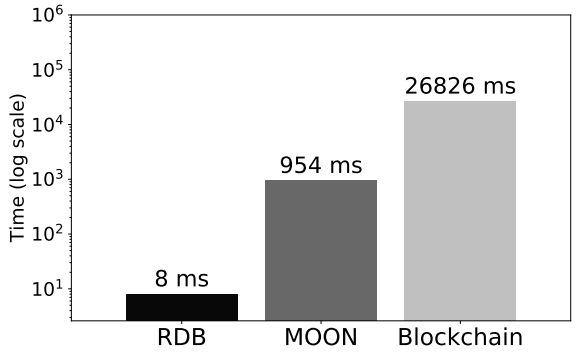


Figure 20 – Q1 Results - Own authorship

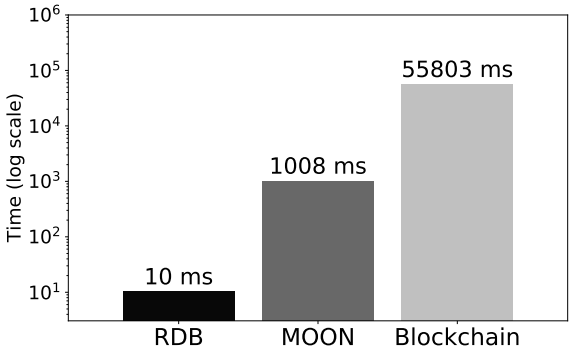


Figure 21 – Q2 Results - Own authorship

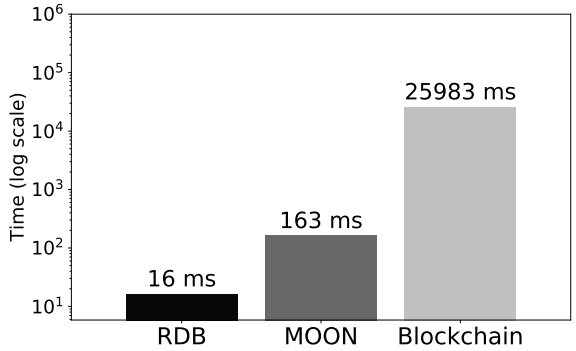


Figure 22 – Q3 Results - Own authorship

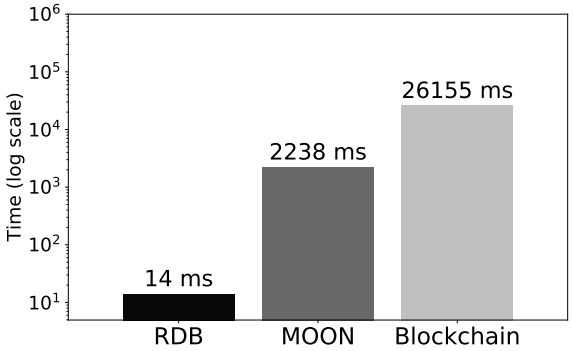


Figure 23 – Q4 Results - Own authorship

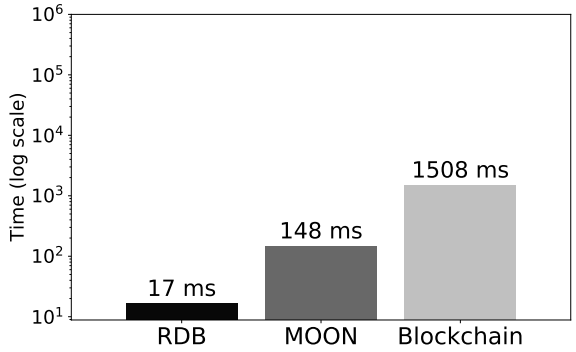


Figure 24 – Q5 Results - Own authorship

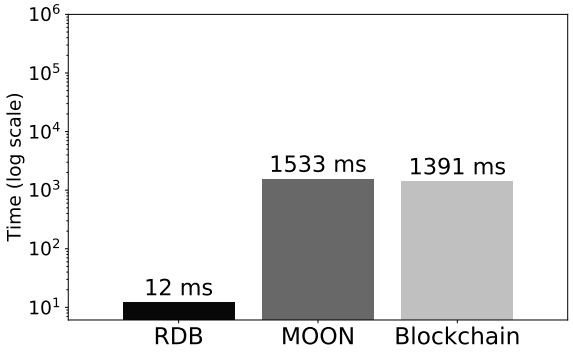


Figure 25 – Q6 Results - Own authorship

The MOON responses confirm its correctness. It returned the same data instance obtained if all data were stored in the traditional RDB, as expected. Comparing MOON responses with the RDB responses is appropriate since RDB is a validated technology. Hence, it attests that the MOON architecture works correctly, including the mapping and indexing models.

The results for Q1 executions attest that the MOON indexing model for Blockchain data reduces search time on Blockchain. In Scenario 3, there was a verifying of all Blockchain records sequentially to get data from the Exam entity, which includes the data of Laboratory Worker, Patient, and Doctor. The MOON was faster because it got the Exam index entries and then requested only the Exam data from Blockchain, avoiding unnecessary requests to Blockchain. It is important to note that the most time-consuming step of the query executed by

the MOON is to get the data in Blockchain, representing more than half the time of the select operation. Therefore, Scenario 2 was about 28 times quicker than Scenario 3, considering their response times. On the other hand, as expected, RDB was considerably faster than the MOON since it did not retrieve Blockchain data.

Regarding Q2 results, Scenario 2 response time represents around 2% of the Scenario 3 response time. As already mentioned, DBMSs data retrieval is more efficient than getting the same data on Blockchain. Consequently, Scenario 2 was faster since only Exam data was retrieved from Blockchain while the Doctor data is on RDB. Furthermore, there was a use of Blockchain data indexing on Scenario 2 to get the data of Exam, while Scenario 3 used sequential search, as in Q1. Therefore, although using an RDB temporary table to execute the WHERE clause increases MOON response time, it is faster than checking data sequentially, as the DBMSs implementation is optimized.

The results of Q3 presents a situation that is a beneficial use of the MOON. In Scenario 2, the MOON checked that the Laboratory Worker entity was in the relational model. Next, the MOON forwarded the request to the RDB, which returned the response efficiently. Scenario 2 has only these two steps more than Scenario 1, making its response times around 150 milliseconds slower. Moreover, Scenario 3 was the slowest since searched data sequentially in the Blockchain, selected the requested data, changed it, and then stored it to the Blockchain as new data. Therefore, using the MOON is beneficial if there is a need for only part of the entities to be on the Blockchain. It ensures a satisfactory performance for data that is on RDB and integrates it with Blockchain networks.

The Q4 results confirm the analysis of Q2 results, which showed that Scenario 3 was slower than Scenario 2, in similar circumstances. Indexing Blockchain data, getting the data from Exam, and inserting it into a temporary table to run the WHERE clause are faster than get Blockchain data sequentially. Moreover, as expected, Scenario 1 was the most efficient since it does not guarantee Blockchain features, which may be advantageous in some situations.

The results of Q5 and Q6 suggest that the use of MOON does not add large overhead in inserts operations. In Q5, MOON verified that the Doctor entity was in the relational model and sent the received request to the RDB, being approximately 132 milliseconds slower than Scenario 1. In Q6, MOON checked the persistence model for the entity Exam, performed the mapping of the request to JSON, sent it to the Blockchain, and stored an index entry for this data, which was 141 milliseconds slower than Scenario 3.

Scenario 2 was the slowest for Q5 and Q6 executions than the persistence models used, as expected. The MOON adds steps to store data from RDB or Blockchain. For example, using the API to send data to the Blockchain is the same in scenarios 2 and 3. Still, Scenario 3 was faster because it did not make data mapping, and there was no verification of data location. Furthermore, in this evaluation, the data insertion on Blockchains was relatively fast because i) the used network had a tiny number of nodes, and ii) the BigchainDB consensus is based on votes, so it is not expensive.

5.2 Using Benchmarks

This section presents the experimental evaluation carried out with benchmarks known and validated by the scientific community. The benchmark selection process prioritized those who use more than one table and are inserted in a context that justifies the use of MOON. Furthermore, the setup configuration of VMs was the same used in section 5.1, with six machines. Moreover, the used technologies are the same as section 5.1, which are PostgreSQL and BigchainDB.

Subsection 5.2.1 explains the evaluation carried out with the Twitter benchmark, while the subsection 5.2.2 describes the experiments made using the Voter benchmark.

5.2.1 Twitter

The Twitter benchmark is based on the popular social network that has the same name. Twitter allows users to send and view personal updates from other users, which are known as tweet and consists of texts of up to 280 characters. Moreover, a user needs to follow another user to see their tweets, like a subscription. The Twitter benchmark data schema is shown in Figure 26, having five entities: i) user_profiles; ii) followers; iii) follows; iv) tweets, and v) added_tweets (DIFALLAH *et al.*, 2013).

There is a partitioning developed for the use of MOON in this experimental evaluation. The entities tweets, followers, follows, and added_tweets were stored in RDB, while the user_profiles entity was in Blockchain. It is worth noting that there is no correct or incorrect data partitioning. It depends on the business rules. Hence, the evaluation data partition purposes guarantee the use of Blockchain properties only to the tweets entity.

As can be seen in Table 9, some Twitter benchmark queries were used in this evaluation, measuring their response times and correctness. The process of obtaining queries

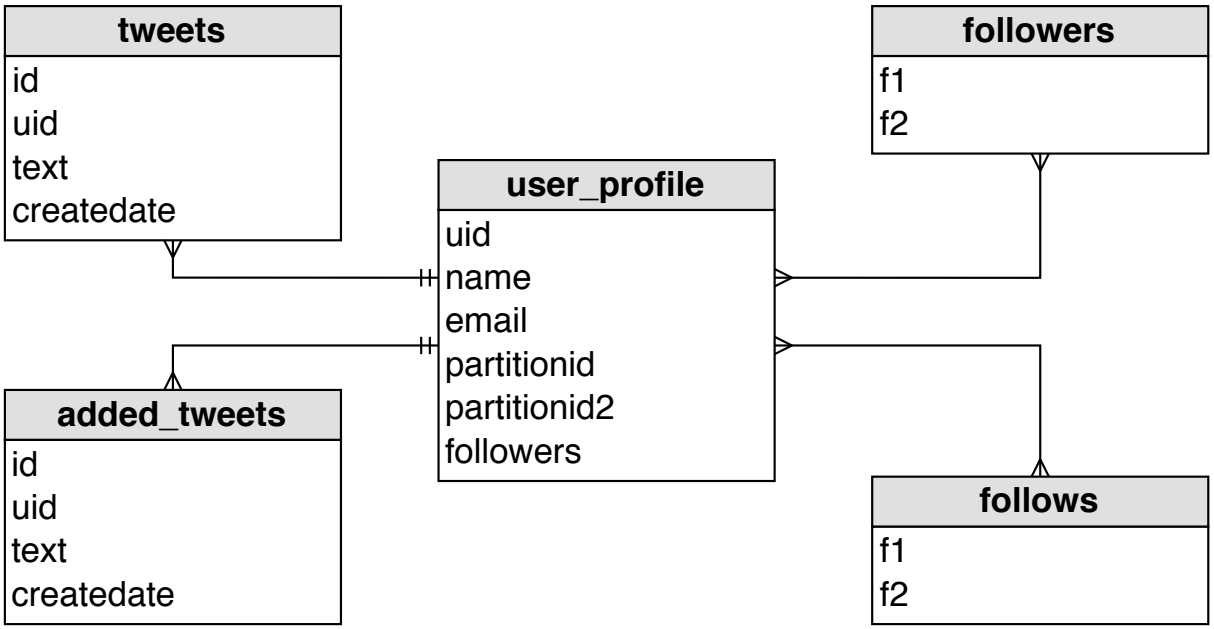


Figure 26 – Twitter data schema - Own authorship

Table 9 – Twitter used queries - Own authorship

ID	Query
Q7	INSERT INTO user_profiles VALUES ($1, $2, $3, $4, $5, $6)
Q8	INSERT INTO tweets VALUES ($1, $2, $3, $4)
Q9	SELECT uid, name FROM user_profiles WHERE uid IN ($1, $2, $3, $4, $5, $6, $7, $8, $9, $10, $11, $12, $13, $14, $15, $16, $17, $18, $19, $20)
Q10	SELECT * FROM tweets WHERE uid = $1 LIMIT 10

occurred in the following form: i) executing a workload in an RDB; ii) getting the RDB log; iii) handling the log file to get only the executed queries, and iv) keeping only the chosen queries for the evaluation. It is important to note that the use of all queries provided by the benchmark is not possible due to time limitations, as each execution of Blockchain queries uses programming code. Hence, implementing all queries is unfeasible. Moreover, as in section 5.1, all inserts of all entities were executed, but one of each persistence model is mentioned.

In total, in carrying out this evaluation, 32327 queries were sent. This amount corresponds to the queries selected after the execution of the benchmark configured with a scale factor of 1. Moreover, one important to note that the queries in table 9 do not have specific values. The characters $ followed by a sequential numerical value replaces the real values used in each query, indicating that there was the execution of the same query with different values.

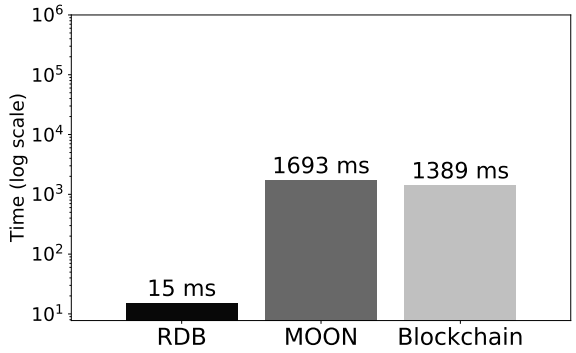


Figure 27 – Q7 Results - Own authorship

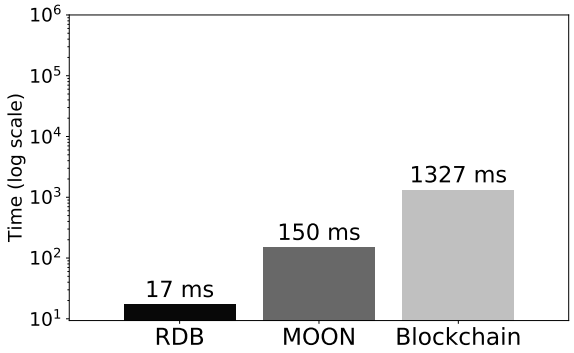


Figure 28 – Q8 Results - Own authorship

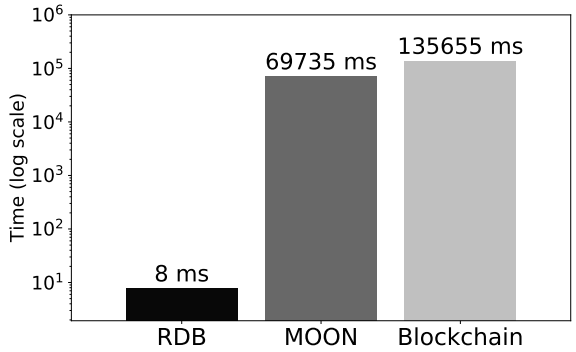


Figure 29 – Q9 Results - Own authorship

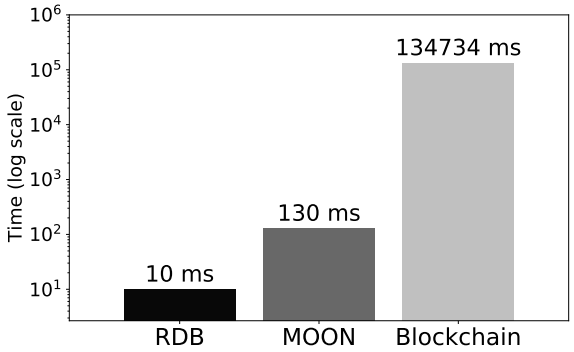


Figure 30 – Q10 Results - Own authorship

5.2.1.1 Results

As expected, the MOON returned the correct answers. It implies that its responses were the same as the results returned in Scenario 1, in which the data was stored exclusively in RDBs. These results were also the same as in Scenario 3, in which the data were stored only in Blockchains. As in the experiments in subsection 5.1, Scenario 2 represents the use of the MOON. Moreover, the results of Q7, Q8, Q9, and Q10 are in Figures 27, 28, 29, and 30, respectively.

Regarding the Q7 results, MOON has a longer response time than RDBs and Blockchains. As MOON uses the Blockchain, it performs the same steps as scenario 3, which consists of sending data to the Blockchain network. However, the MOON does additional actions: i) detects the user_profiles entity in Blockchain, ii) generates a JSON data format from the received query, and iii) saves an index entry for each data stored in the Blockchain. Although the difference compared to scenario 3 is around 300 milliseconds, using MOON may not be the best choice when obtaining the shortest response time for insertions is the most important to the application.

The results of Q8 exposes that the MOON is substantially faster than scenario 3. If it is possible to store a part of the data in Blockchain and keep another part in RDBs, executing

inserts using MOON gets response times not far to those of RDBs. Moreover, the difference between the response times of Scenarios 2 and 3 was approximately 1500 milliseconds, which is a considerable difference. On the other hand, the difference between the response times for scenarios 1 and 2 was 133 milliseconds, as MOON identifies where the data is and sends it to the RDB, which is responsible for storing it. The MOON's advantage over scenario 1 is that using MOON allows part of the data to be stored in Blockchain.

Q9 results present a case in which Scenario 2 is considerably advantageous to Scenario 3, although it is not satisfactory in some contexts. The MOON has a response time that is half the response time of Scenario 3. As there are thousands of data in the Blockchain, it is expensive to retrieve each one to check if it relates to a specific entity and then filter if its id value is in a list. The MOON is advantageous because it uses index entries to obtain only those records for the specific entity. Hence, it inserts the data into a temporary RDB table, which filters the data. Finally, it is worth noting that the RDB is substantially faster, more advantageous when performance is more relevant than ensuring Blockchain properties.

Regarding the Q10 results, MOON uses RDB, which makes it substantially faster than scenario 3, although it does not guarantee Blockchain properties. It is advantageous if the entity's tweets data are more related to RDB and do not need aspects such as immutability, auditability, and other features associated with Blockchains. As the volume of data used in this assessment is bigger, this implies a wide difference between the response times for scenarios 2 and 3.

5.2.2 Voter

The voter benchmark bases on the software used to register votes for a Japanese and Canadian television talent show. Users submit their votes for a candidate during the program, and the system performs transactions to store each candidate's votes. It is important to note that each user has a limited number of votes, and the system refuses votes after the user exceeds that limit. Moreover, there is a periodical execution of a transaction to calculate the total votes momentarily. The Voter benchmark data schema is presented in figure 31, which has three entities: i) contestants; ii) area_code_state; and iii) votes (DIFALLAH *et al.*, 2013).

Similarly to subsection 5.2.1, there was a selection of some Voter benchmark queries to using it in this evaluation. The use of these queries made it possible to measure the response times and the proposed approach's correctness. As discussed previously, using all the workload

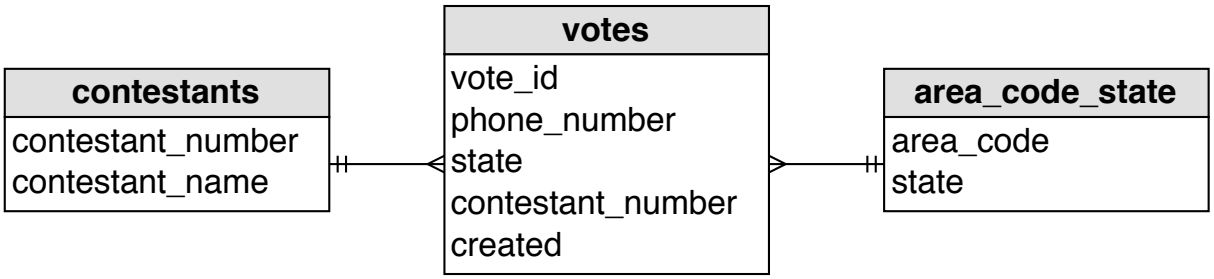


Figure 31 – Voter data schema - Own authorship

Table 10 – Voter used queries - Own authorship

ID	Query
Q11	INSERT INTO AREA_CODE_STATE VALUES ($1, $2)
Q12	INSERT INTO VOTES VALUES ($1, $2, $3, $4, $5)
Q13	SELECT state FROM AREA_CODE_STATE WHERE area_code = $1
Q14	SELECT COUNT(*) FROM VOTES WHERE phone_number = $1

provided by the benchmark is not feasible. The process of obtaining queries occurred in the same way as described in Subsection 5.2.1.

A data partitioning was needed to evaluate the MOON using the voter. Contestants and area_code_state entities were in RDB, while the entity votes storage was in Blockchain. One notice that this partitioning is in a context where it seems appropriate. Nonetheless, there may be other contexts, with different business rules, changing the necessary partitioning. Thus, considering the partitioning done in this evaluation, there is a need to guarantee Blockchain properties only for the votes entity, while the others are more related to RDBs.

Altogether, this evaluation used 494 queries. It is related to the queries obtained after running the benchmark configured with a scale factor of 1. It is important to note that this workload is quantitatively different from that used in the Twitter evaluation, making it possible to observe potential distinct behaviors. Furthermore, as in the previous experiments, the characters $ followed by a sequential numeric value in Table 10 means that each query was used with different values.

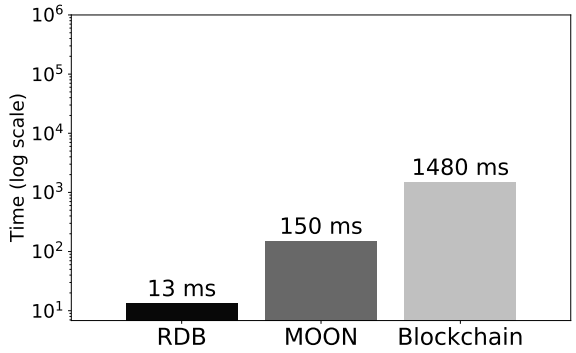


Figure 32 – Q11 Results - Own authorship

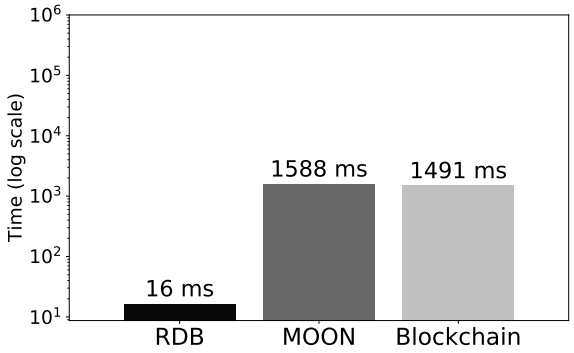


Figure 33 – Q12 Results - Own authorship

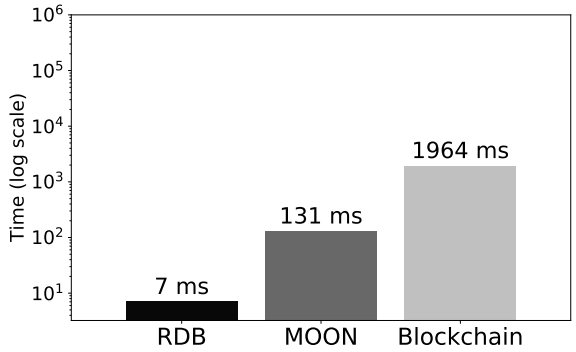


Figure 34 – Q13 Results - Own authorship

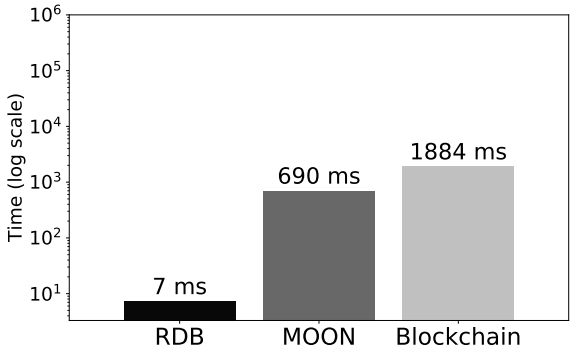


Figure 35 – Q14 Results - Own authorship

5.2.2.1 Results

Figures 32, 33, 34, and 35, show the results of queries Q11, Q12, Q13, and Q14, respectively. As in previous experiments, the results obtained demonstrate that the MOON returned the correct responses. Moreover, the Q11 and Q12 results corroborate the analysis made about the Q5, Q6, Q7, and Q8, which are comparable to them. Hence, it is attested that to insertions using the relational model, the response times of the MOON are close to the response times of RDBs, although it is still longer, presenting a small delay. Similarly, MOON has slightly longer response times than scenario 3 for data insertions.

The response times for Scenario 2 in Q13 indicate the benefit of partitioning data using MOON. As the AREA_CODE_STATE entity is more pertinent to the relational model and does not need to guarantee Blockchain properties, its response times are close to those obtained by using only RDBs, in Scenario 1. The additional delay that MOON imposes to the use of only RDBs is recognizing the data model used by the AREA_CODE_STATE entity. After that, the MOON sends the request to the RDBMS, which handles it and sends the result back. Next, there is forwarding of the response to the user. Unlike Scenario 2, Scenario 3 performs the sequential search previously mentioned in all Blockchain data, making the process slower, although it guarantees Blockchain properties to AREA_CODE_STATE data.

The results of Q14 confirm that MOON has response times higher than Scenario 1, but lower than Scenario 3. Specifically, in this result, MOON obtained results no too far from those obtained by Scenario 1 due to the tiny amount of data used. The number of blocks used to store data was substantially less than other evaluations, optimizing the recovery time of data in blocks. As already mentioned, getting blocks is the most expensive part that MOON performs when it select queries.

5.3 Conclusion

This chapter presented the experimental evaluation carried out to validate the MOON. Experiments were carried out using real data from the e-health domain and two benchmarks already validated by the scientific community, Twitter and voter. The experiments also compared three approaches: i) data only in RDB, ii) data partitioned using MOON, and ii) data only in Blockchain. The metrics used for evaluation were correctness and response time. The results showed that the MOON answered the requests correctly, in the same way as the traditional database.

The RDB was the fastest in all scenarios, as expected. Also, MOON indexing has made the data recovery more agile than traditional blockchains. Furthermore, creating temporary tables in RDB to resolve queries with conditionals or more than one entity involved was faster than Blockchain's sequential search. Moreover, when using RDB, MOON added little delay to using only RDB. Finally, MOON was only slower than using the just Blockchain to insert data into Blockchain, as expected.

6 CONCLUSION AND FUTURE WORK

This chapter concludes this document, highlighting the most important aspects of this research. Moreover, there is a discussion of future work, showing the new challenges in extending this work.

6.1 Conclusion

This document presented the MOON, an approach that manages partitioned data between Blockchain and RDB, which receives SQL requests. Answering the research question RQ1, in Section 1.4, it has some advantages: i) allows transparent data management in two different persistence models, which have different communication interfaces; ii) takes advantage of relevant Blockchain properties, such as immutability and no third parties; and iii) uses RDB characteristics, which are data mutability, complex queries, and fast query processing. Moreover, the data partitioning carried out for the evaluation demonstrates how this process occurs, pondering each persistence model's advantages, which explains the RQ2. Finally, solving the RQ3, the research described a mapping model between RDB and Blockchain, and an architecture model generic enough to support applications from different domains.

There was an experimental evaluation using the MOON with data of different domains to validate the proposed approach. MOON has been compared with two other approaches: storing data in RDB only and using Blockchain as the unique form of storage. Moreover, the experimental evaluation consisted of two parts: using real data and using benchmarks. The first simulated an application that manages blood exams using six queries, which supported to analyze the behavior of MOON in different scenarios. The second part of the evaluation used two benchmarks validated by the scientific community. From each benchmark, four queries were used, examining different situations.

The evaluation results revealed that the MOON returns the right responses to the clients, that is, the same data instance obtained if all data were just in a traditional RDB. Furthermore, the proposed approach was slower about the obtained response times than RDB, which may be admissible in any application domains since it provides some advantages. Moreover, except for inserts, MOON is faster than using data stored only in Blockchains with sequential access to data. Furthermore, it was proved that the MOON provides relational and blockchain features to the data according to the defined partitioning, confirming the research hypothesis.

6.2 Future Work

Future work opportunities arise from this research. One is examining how the MOON performs in large-scale networks. The real networks may have a huge number of nodes, such as hundreds, and test with it its important to evaluate if there any substantial overhead of communication or processing.

The use of other workloads brings higher reliability and attests to the robustness of the MOON. As the proposed approach was developed to be of general-purpose, there are plans to evaluate the solution in other datasets and benchmarks, generating different workloads. Some options to consider are i) using datasets with real and open data from different domains, such as legal, insurance, and credit card applications, and ii) use benchmarks validated by the scientific community, such as SmallBank (DIFALLAH *et al.*, 2013).

The next experiments using MOON should use large volumes of data to evaluate its scalability. MOON possibly does not have satisfactory response times for some applications that use a large volume of data, although it is still faster than using only Blockchain utilizing sequential access. It is mainly due to the need to get data from multiple blocks on the Blockchain network. One possible solution for this is by caching Blockchain data. However, it should be carefully assessed to analyze how much it makes immutability violations possible and compromises auditability.

The proposed approach does not implement smart contracts. It is not in this research's scope because its focus is on partitioning and mapping between RDB and Blockchain. However, using smart contracts is a relevant point to explore since it can provide relevant features to the MOON, such as triggers and queries to external data. One opportunity is to adopt the Procedural Language extensions to SQL (PL/SQL) as the language used in smart contracts, which considers the expertise of those who have consolidated knowledge of RDB and their technologies.

To solve select operations, the MOON uses temporary tables in RDB, taking the support of its optimized implementations. Nevertheless, there are probably other ways to make it more efficient, as the approach presented in this research has significantly longer response times than those presented by RDBs. Therefore, one relevant future work is to evaluate other strategies that make faster selects in the MOON. Obtaining response times closer to RDBs would be a significant benefit to the use of the MOON.

A limitation of the MOON is that it supports only simple queries. Nested queries are not supported, which may be useful for applications in contexts where more complex queries are

required. Another unsupported functionality is the use of queries with the reserved word JOIN, so joins must be made using the WHERE clause and conditionals. Therefore, another possibility for future work is to examine strategies that support the use of these features, making the MOON more robust.

Another important future work is to prove that the MOON meets the General Law on the Protection of Personal Data, approved by the Brazilian Congress in 2018 (BRASIL, 2018). Furthermore, it is appropriate to investigate the best strategies to provide data privacy, informational self-determination, and the inviolability of confidence, honor, and image. Hence, ensuring this is not only complying with the law but also providing security and robustness.

BIBLIOGRAPHY

ANTONOPOULOS, A. M. **Mastering Bitcoin: Programming the open blockchain**. [*S. l.*]: " O'Reilly Media, Inc.", 2017.

AZARIA, A.; EKBLAW, A.; VIEIRA, T.; LIPPMAN, A. Medrec: Using blockchain for medical data access and permission management. In: IEEE. **2016 2nd International Conference on Open and Big Data (OBD)**. [*S. l.*], 2016. p. 25–30.

BARTOLETTI, M.; LANDE, S.; POMPIANU, L.; BRACCIALI, A. A general framework for blockchain analytics. In: **Proceedings of the 1st Workshop on Scalable and Resilient Infrastructures for Distributed Ledgers**. NY, USA: ACM, 2017. (SERIAL '17), p. 7:1–7:6. ISBN 978-1-4503-5173-7. Available at: http://doi.acm.org/10.1145/3152824.3152831. Access on: 15 sep. 2018.

BASHIR, I. **Mastering Blockchain: Distributed ledger technology, decentralization, and smart contracts explained, 2nd Edition**. Packt Publishing, 2018. ISBN 9781788838672. Available at: https://books.google.com.br/books?id=3ZlUDwAAQBAJ. Access on: 02 jan. 2018.

BRAGA, A.; DAHAB, R. Introdução à criptografia para programadores: Evitando maus usos da criptografia em sistemas de software. In: ____. SBC, 2015. cap. 1, p. 1–50. Available at: https://siaiap34.univali.br/sbseg2015/anais/LivroMinicursosSBSeg2015.pdf. Access on: 10 may. 2018.

BRAGA, A. M.; MARINO, F. C. H.; SANTOS, R. R. dos. Segurança de aplicações blockchain além das criptomoedas. In: ____. SBC, 2017. cap. 3, p. 100–148. Available at: https://sbseg2017.redes.unb.br/wp-content/uploads/2017/04/20171107-SBSeg2017-Livro_de_Minicursos.pdf. Access on: 10 may. 2018.

BRAGAGNOLO, S.; ROCHA, H.; DENKER, M.; DUCASSE, S. Ethereum query language. In: **Proceedings of the 1st International Workshop on Emerging Trends in Software Engineering for Blockchain**. NY, USA: ACM, 2018. (WETSEB '18), p. 1–8. ISBN 978-1-4503-5726-5. Available at: http://doi.acm.org/10.1145/3194113.3194114. Access on: 11 jan. 2019.

BRASIL. Lei n. 13.079, de 14 de agosto de 2018. lei geral de proteção de dados pessoais (lgpd). **Diário Oficial da União**, 2018.

CACHIN, C. Architecture of the hyperledger blockchain fabric. In: **Workshop on distributed cryptocurrencies and consensus ledgers**. [*S. l.*: *s. n.*], 2016. v. 310, p. 4.

CAHILL, D. G.; BAUR, D. G.; LIU, Z. F.; YANG, J. W. I am a blockchain too: How does the market respond to companies' interest in blockchain? **Journal of Banking & Finance**, Elsevier, p. 105740, 2020.

CASADO-VARA, R.; CORCHADO, J. Distributed e-health wide-world accounting ledger via blockchain. **Journal of Intelligent & Fuzzy Systems**, IOS Press, v. 36, n. 3, p. 2381–2386, 2019.

CHALAEMWONGWAN, N.; KURUTACH, W. State of the art and challenges facing consensus protocols on blockchain. In: **2018 International Conference on Information Networking (ICOIN)**. [*S. l.*: *s. n.*], 2018. p. 957–962.

CHICARINO, V. L. R.; JESUS, E. F.; ALBUQUERQUE, C. V. N.; ROCHA, A. A. de A. Uso de blockchain para privacidade e segurança em internet das coisas. In: ____. SBC, 2017. cap. 4, p. 149–199. Available at: https://sbseg2017.redes.unb.br/wp-content/uploads/2017/04/20171107-SBSeg2017-Livro_de_Minicursos.pdf. Access on: 17 feb. 2018.

CHOWDHURY, M. J. M.; COLMAN, A.; KABIR, M. A.; HAN, J.; SARDA, P. Blockchain versus database: A critical analysis. In: IEEE. **2018 17th TrustCom / 12th BigDataSE**. [*S. l.*], 2018. p. 1348–1353.

CROSBY, M.; PATTANAYAK, P.; VERMA, S.; KALYANARAMAN, V. *et al.* Blockchain technology: Beyond bitcoin. **Applied Innovation**, v. 2, n. 6-10, p. 71, 2016.

DATE, C. **An Introduction to Database Systems**. Addison-Wesley Publishing Company, 1995. (Addison-Wesley systems programming series). ISBN 9780201543292. Available at: https://books.google.com.br/books?id=2xBRAAAAMAAJ. Access on: 14 may. 2018.

DIFALLAH, D. E.; PAVLO, A.; CURINO, C.; CUDRE-MAUROUX, P. Oltp-bench: An extensible testbed for benchmarking relational databases. **Proceedings of the VLDB Endowment**, VLDB Endowment, v. 7, n. 4, p. 277–288, 2013.

Dinh, T. T. A.; Liu, R.; Zhang, M.; Chen, G.; Ooi, B. C.; Wang, J. Untangling blockchain: A data processing view of blockchain systems. **IEEE Transactions on Knowledge and Data Engineering**, v. 30, n. 7, p. 1366–1385, July 2018. ISSN 1041-4347.

DRESCHER, D. **Blockchain basics**. [*S. l.*]: Springer, 2017. v. 276.

EF, C. A relational model of data for large shared data banks. **Communications of the ACM**, v. 13, n. 6, p. 377–387, 1970.

ELMASRI, R.; NAVATHE, S. **Fundamentals of database systems**. [*S. l.*]: Pearson, 2015.

EYSENBACH, G. What is e-health? **J Med Internet Res**, v. 3, n. 2, p. e20, Jun 2001. ISSN 1438-8871. Available at: https://doi.org/10.2196/jmir.3.2.e20. Access on: 12 may. 2018.

GATTESCHI, V.; LAMBERTI, F.; DEMARTINI, C. Blockchain technology use cases. In: **Advanced Applications of Blockchain Technology**. [*S. l.*]: Springer, 2020. p. 91–114.

GONCZOL, P.; KATSIKOULI, P.; HERSKIND, L.; DRAGONI, N. Blockchain implementations and use cases for supply chains-a survey. **IEEE Access**, IEEE, v. 8, p. 11856–11871, 2020.

GREVE, F. Protocolos fundamentais para o desenvolvimento de aplicações robustas. In: ____. [*S. l.*]: SBC, 2005. cap. 5, p. 330–398.

GREVE, F.; SAMPAIO, L.; ABIJAUDE, J.; COUTINHO, A.; VALCY Ítalo; QUEIROZ, S. Blockchain e a revolução do consenso sob demanda. In: ____. SBC, 2018. cap. 5, p. 1–52. Available at: http://portaldeconteudo.sbc.org.br/index.php/minicursossbrc. Access on: 12 feb. 2018.

HEARN, M. Corda: A distributed ledger. **Corda Technical White Paper**, 2016.

HUH, S.; CHO, S.; KIM, S. Managing iot devices using blockchain platform. In: IEEE. **2017 19th international conference on advanced communication technology (ICACT)**. [*S. l.*], 2017. p. 464–467.

KASSAB, M.; DEFRANCO, J.; MALAS, T.; DESTEFANIS, G.; NETO, V. G. Exploring research in blockchain for healthcare and a roadmap for the future. **IEEE Transactions on Emerging Topics in Computing**, PP, 08 2019.

KING, S.; NADAL, S. **Peer-to-peer crypto-currency with proof-of-stake**. 2012. Self-Published Paper. Available at: https://www.researchgate.net/publication/265116876_PPCoin_Peer-to-Peer_Crypto-Currency_with_Proof-of-Stake. Access on: 22 fev. 2018.

KUO, T.-T.; KIM, H.-E.; OHNO-MACHADO, L. Blockchain distributed ledger technologies for biomedical and health care applications. **Journal of the American Medical Informatics Association**, Oxford University Press, v. 24, n. 6, p. 1211–1220, 2017.

LARIMER, D. Delegated proof-of-stake (dpos). **Bitshare whitepaper**, 2014.

LI, X.; JIANG, P.; CHEN, T.; LUO, X.; WEN, Q. A survey on the security of blockchain systems. **Future Generation Computer Systems**, Elsevier, 2017.

MAENHAUT, P.; MOENS, H.; ONGENAE, V.; TURCK, F. D. Design and evaluation of a hierarchical multi-tenant data management framework for cloud applications. In: **2015 IFIP/IEEE IM**. [*S. l.*: *s. n.*], 2015. p. 1208–1213. ISSN 1573-0077.

MARINHO, C.; MOREIRA, L. O.; MACHADO, J. Moon: An approach to data management on relational database and blockchain. In: **SBBD 2019 - WTDBD** (). Fortaleza, Ceará: [*S. n.*], 2019. Available at: https://sbbd.org.br/2019/wp-content/uploads/sites/6/2020/01/Proceddings.pdf. Access on: 20 apr. 2020.

MARINHO, C. S. S.; MOREIRA, L.; COUTINHO, E.; FILHO, J. S. C.; SOUSA, F.; MACHADO, J. Labareda: A predictive and elastic load balancing service for cloud-replicated databases. v. 9, 06 2018.

MELCHIORRE, M. G.; LAMURA, G.; BARBABELLA, F.; CONSORTIUM, I. ehealth for people with multimorbidity: Results from the icare4eu project and insights from the “10 e’s” by gunther eysenbach. **PloS one**, Public Library of Science San Francisco, CA USA, v. 13, n. 11, p. e0207292, 2018.

METTLER, M. Blockchain technology in healthcare: The revolution starts here. In: IEEE. **2016 IEEE 18th international conference on e-health networking, applications and services (Healthcom)**. [*S. l.*], 2016. p. 1–3.

MINGXIAO, D.; XIAOFENG, M.; ZHE, Z.; XIANGWEI, W.; QIJUN, C. A review on consensus algorithm of blockchain. In: IEEE. **2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)**. [*S. l.*], 2017. p. 2567–2572.

MOREIRA, L. O.; MARINHO, C. S. S.; NETO, M. M.; COUTINHO, E. F.; SOUZA, J. N. de; MACHADO, J. C. Oportunidades de pesquisa para o uso de infraestruturas blockchain na gestão de dados distribuídos. **Revista Sistemas e Mídias Digitais (RSMD)**, v. 4, n. 1, abril 2019. ISSN 2525-9555. Available at: http://revistasmd.virtual.ufc.br/arquivos/volume-4/numero-1/rsmd-v4-n1-p7.pdf. Access on: 19 apr. 2020.

MUZAMMAL, M.; QU, Q.; NASRULIN, B. Renovating blockchain with distributed databases: An open source system. **Future Generation Computer Systems**, v. 90, p. 105 – 117, 2019. ISSN 0167-739X. Available at: http://www.sciencedirect.com/science/article/pii/S0167739X18308732. Access on: 15 fev. 2020.

NAKAMOTO, S. Bitcoin: A peer-to-peer electronic cash system. Working Paper, 2008.

NARAYANAN, A.; BONNEAU, J.; FELTEN, E.; MILLER, A.; GOLDFEDER, S. **Bitcoin and Cryptocurrency Technologies**. [*S. l.*]: Princeton University Press, 2016.

NOFER, M.; GOMBER, P.; HINZ, O.; SCHIERECK, D. Blockchain. **Business & Information Systems Engineering**, v. 59, n. 3, p. 183–187, Jun 2017. ISSN 1867-0202. Available at: https://doi.org/10.1007/s12599-017-0467-3. Access on: 28 may. 2019.

PATRÍCIO, M.; PEREIRA, J.; CRISÓSTOMO, J.; MATAFOME, P.; GOMES, M.; SEIÇA, R.; CARAMELO, F. Using resistin, glucose, age and bmi to predict the presence of breast cancer. **BMC Cancer**, v. 18, n. 1, p. 29, Jan 2018. ISSN 1471-2407. Available at: https://doi.org/10.1186/s12885-017-3877-1. Access on: 10 jul. 2019.

PAVLO, A.; ASLETT, M. What's really new with newsql? **ACM Sigmod Record**, ACM, v. 45, n. 2, p. 45–55, 2016.

RAMAKRISHNAN, R.; GEHRKE, J. **Database management systems**. [*S. l.*]: McGraw Hill, 2003.

RAY, P. P.; DASH, D.; SALAH, K.; KUMAR, N. Blockchain for iot-based healthcare: Background, consensus, platforms, and use cases. **IEEE Systems Journal**, IEEE, 2020.

ROGAWAY, P.; SHRIMPTON, T. Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. In: SPRINGER. **International workshop on fast software encryption**. [*S. l.*], 2004. p. 371–388.

ROSS, J.; STEVENSON, F.; LAU, R.; MURRAY, E. Factors that influence the implementation of e-health: a systematic review of systematic reviews (an update). **Implementation Science**, v. 11, n. 1, p. 146, Oct 2016. ISSN 1748-5908. Available at: https://doi.org/10.1186/s13012-016-0510-7. Access on: 17 aug. 2019.

RUAN, P.; CHEN, G.; DINH, T. T. A.; LIN, Q.; LOGHIN, D.; OOI, B. C.; ZHANG, M. **Blockchains and Distributed Databases: a Twin Study**. 2019.

SAMANIEGO, M.; DETERS, R. Blockchain as a service for iot. In: IEEE. **2016 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)**. [*S. l.*], 2016. p. 433–436.

SIKORSKI, J. J.; HAUGHTON, J.; KRAFT, M. Blockchain technology in the chemical industry: Machine-to-machine electricity market. **Applied Energy**, Elsevier, v. 195, p. 234–246, 2017.

SWAN, M. **Blockchain: Blueprint for a new economy**. [*S. l.*]: " O'Reilly Media, Inc.", 2015.

SZABO, N. Smart contracts. **Unpublished manuscript**, 1994.

VASIN, P. **BlackCoin's Proof-of-Stake Protocol v2**. 2014. Self-Published Paper. Available at: https://blackcoin.org/blackcoin-pos-protocol-v2-whitepaper.pdf. Access on: 27 fev. 2018.

VO, H. T.; KUNDU, A.; MOHANIA, M. K. Research directions in blockchain data management and analytics. In: **EDBT**. [*S. l.*: *s. n.*], 2018. p. 445–448.

WENTING, L.; SÉBASTIEN, A.; JENS-MATTHIAS, B.; GHASSAN, K. Securing proof-of-stake blockchain protocols. In: GARCIA-ALFARO, J.; NAVARRO-ARRIBAS, G.; HARTENSTEIN, H.; HERRERA-JOANCOMARTÍ, J. (Ed.). **Data Privacy Management, Cryptocurrencies and Blockchain Technology**. Cham: Springer International Publishing, 2017. p. 297–315. ISBN 978-3-319-67816-0.

WOOD, G. Ethereum: A secure decentralised generalised transaction ledger. **Ethereum project yellow paper**, v. 151, p. 1–32, 2014.

WÜST, K.; GERVAIS, A. Do you need a blockchain? In: IEEE. **2018 Crypto Valley Conference on Blockchain Technology (CVCBT)**. [*S. l.*], 2018. p. 45–54.

XIAO, Y.; ZHANG, N.; LOU, W.; HOU, Y. T. A survey of distributed consensus protocols for blockchain networks. **IEEE Communications Surveys & Tutorials**, IEEE, 2020.

XIE, S.; ZHENG, Z.; CHEN, W.; WU, J.; DAI, H.-N.; IMRAN, M. Blockchain for cloud exchange: A survey. **Computers & Electrical Engineering**, Elsevier, v. 81, p. 106526, 2020.

XU, X.; PAUTASSO, C.; ZHU, L.; GRAMOLI, V.; PONOMAREV, A.; TRAN, A. B.; CHEN, S. The blockchain as a software connector. In: **2016 13th Working IEEE/IFIP Conference on Software Architecture (WICSA)**. [*S. l.*: *s. n.*], 2016. p. 182–191.

XU, X.; WEBER, I.; STAPLES, M.; ZHU, L.; BOSCH, J.; BASS, L.; PAUTASSO, C.; RIMBA, P. A taxonomy of blockchain-based systems for architecture design. In: IEEE. **2017 IEEE International Conference on Software Architecture (ICSA)**. [*S. l.*], 2017. p. 243–252.

YAGA, D.; MELL, P. Nistir 8202: Blockchain technology overview. In: ____. [*S. l.*]: National Institute of Standards and Technology, 2018. cap. 1, p. 1–68.

YLI-HUUMO, J.; KO, D.; CHOI, S.; PARK, S.; SMOLANDER, K. Where is current research on blockchain technology?—a systematic review. **PloS one**, Public Library of Science, v. 11, n. 10, p. e0163477, 2016.

ZAMFIR, V. **Introducing Casper "the Friendly Ghost"**. 2015. Available at: https://blog.ethereum.org/2015/08/01/introducing-casper-friendly-ghost/. Access on: 27 fev. 2018.

ZHANG, P.; BOULOS, M. N. K. Blockchain solutions for healthcare. In: **Precision Medicine for Investigators, Practitioners and Providers**. [*S. l.*]: Elsevier, 2020. p. 519–524.

Zhang, P.; Walker, M. A.; White, J.; Schmidt, D. C.; Lenz, G. Metrics for assessing blockchain-based healthcare decentralized apps. In: **2017 IEEE 19th International Conference on e-Health Networking, Applications and Services (Healthcom)**. [*S. l.*: *s. n.*], 2017. p. 1–4.

ZHENG, Z.; XIE, S.; DAI, H.; CHEN, X.; WANG, H. An overview of blockchain technology: Architecture, consensus, and future trends. In: **2017 IEEE International Congress on Big Data (BigData Congress)**. [*S. l.*: *s. n.*], 2017. p. 557–564.

ZHENG, Z.; XIE, S.; DAI, H.-N.; CHEN, X.; WANG, H. Blockchain challenges and opportunities: A survey. **International Journal of Web and Grid Services**, Inderscience Publishers (IEL), v. 14, n. 4, p. 352–375, 2018.

ZHU, Y.; ZHANG, Z.; JIN, C.; ZHOU, A.; YAN, Y. Sebdb: Semantics empowered blockchain database. In: **2019 IEEE 35th International Conference on Data Engineering (ICDE)**. [*S. l.*: *s. n.*], 2019. p. 1820–1831.

ZYSKIND, G.; NATHAN, O.; PENTLAND, A. Decentralizing privacy: Using blockchain to protect personal data. In: **2015 IEEE Security and Privacy Workshops**. [*S. l.*: *s. n.*], 2015. p. 180–184.

ÖZSU, M. T.; VALDURIEZ, P. **Principles of Distributed Database Systems, Third Edition**. 3. ed. Springer-Verlag New York, 2011. ISBN 1441988335,9781441988331. Available at: http://gen.lib.rus.ec/book/index.php?md5=7A0F200338289C2F7CBE5B7E165FA510. Access on: 25 fev. 2018.