Blockchain Extension for PostgreSQL Data Storage

Yash Madhwal
Center for Computational and Data-Intensive Science and Engineering, Skolkovo Institute of Science and technology, Moscow
Russia
yash.madhwal@skoltech.ru

Darkhan Nurlybay
Faculty of Computer Science, National Research University Higher School of Economics, Moscow
Russia
dnurlybay@edu.hse.ru

Yury Yanovich
Center for Computational and Data-Intensive Science and Engineering, Skolkovo Institute of Science and technology, Moscow
Russia
yury.yanovich@skoltech.ru

ABSTRACT

Blockchain is an emerging technology with the potential to resolve auditing issues. Implementing a new blockchain-related feature implies moving to a platform with another database or duplicating its parts in a blockchain system. Both ways are difficult to migrate and maintain. The alternative is to implement blockchain features within the existing database, including consensus mechanisms and specific data structures for audit needs. The paper describes and evaluates a database extension with blockchain-related structures, leaving consensus beyond the scope. We use an account-based prototype of cryptocurrency as a model example. The proposed extension allows provably checking transaction content and user balance without a full database lookup. The numerical experiments to study the overhead of the proposed extension are provided.

CCS CONCEPTS

• **Information systems** → **Data management systems**; • **Computer systems organization** → **Peer-to-peer architectures**.

KEYWORDS

Blockchain, Database, Merkle tree, Modified Merkle Patricia Trie

ACM Reference Format:
Yash Madhwal, Darkhan Nurlybay, and Yury Yanovich. 2021. Blockchain Extension for PostgreSQL Data Storage. In *2021 3rd Blockchain and Internet of Things Conference (BIOTC 2021) (BIOTC 2021), July 8–10, 2021, Ho Chi Minh City, Vietnam.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3475992.3476002

1 INTRODUCTION

A database is a collection of data or information that is stored electronically on computer devices. The majority of industrial companies manage and organize their data to improve efficiency. Relational databases [8] are databases that store and provides access to data points. The relational database helped organizations maintain databases and create information by combining the data tables, thus allowing users to understand the relationships between them. The sheer volume of stored information instigates data auditability problems [3, 4, 15], i.e., users without full access can send their queries but cannot verify the validity of the response, especially if they do not trust the system maintainers. Blockchain provides an immutable transaction log with specific tree structures [29] and solves the auditability problem.

Enabling new blockchain-related features for classical database systems implies moving to a platform with another database inside or duplicating it in a blockchain system. Such migration and maintenance are difficult. Although the blockchain has application in many areas, like the financial sector, state registries, supply chains, machine learning, Internet of things [2, 16, 19–21, 23, 24, 29, 31, 32, 34], its implementation in existing projects is inconvenient. The alternative is to implement blockchain features within the existing database. Researchers have already proposed several such solutions [25, 28, 35, 36]. But they only affect the consensus mechanism, while classical blockchain systems [5, 29] also provide specific data structures cryptographically provable query responses without looking at the entire data store. In the current research, we reproduce these data structures through a traditional database.

This paper describes and evaluates the proposed blockchain's data storage performance using the database example of an account-based prototype cryptocurrency so that the transaction and balance requests can support cryptographic proofs for the response's correctness. PostgreSQL is a database management system for implementation because it is popular, open-source, highly scalable, and compatible with different programming languages [12]. However, the implementation is not platform-specific, and the results are portable to other relational databases with SQL-like language support. The cryptocurrency is easy to understand and, historically, is the first example of the blockchain system. We emphasize that it is only a prototype as it is not secure as a cryptocurrency because of authorization absence. Nevertheless, it allows demonstrating the concepts.

2 RELATED WORK

2.1 Databases inside Blockchains

Blockchains are characterized by a consensus and a method to store data. Consensus algorithms are used to agree on a new batch of transactions to commit to the blockchain block. From the point of view of this paper, consensus represents a reliable system timestamp. Meaning, the economic [14, 18, 19, 29] fault-tolerant [7, 11, 13, 22], impossibility of corruption within the chosen model. However, the consensus mechanism is beyond the scope of the paper. Initially,

BIOTC 2021, July 8–10, 2021, Ho Chi Minh City, Vietnam

ACM ISBN 978-1-4503-8951-8/21/07…$15.00
https://doi.org/10.1145/3475992.3476002

databases were used as internal parts of blockchain solutions providing data storage. Let us point some of their features.

The Bitcoin white paper [29] introduces storing the transaction in the data directory associated with blocks and Merkle trees [27]. This data storage structure verifies if a given block's transaction is valid by comparing precalculated block's Merkle root with the recalculating the root of the Merkle tree starting from the given transaction.

The Ethereum wallet does not store the account balances directly in the blockchain, but only the root node hashes of the transaction prefix tree (trie), state trie, and receipts trie in the form of modified Merkle Patricia trie (hereafter, Patricia tree to be short). When a transaction occurs, the world state, which keeps the wallets' balance, gets updated, and only the root hash of this state tire is stored in the blockchain. Ethereum supports proofs for each account state and associated storage [17]. Since 2015, parallel to public blockchains, such as Bitcoin and Ethereum, private blockchain [6, 37] got their popularity for blockchain applications in many projects at the state and global business levels [30, 31].

2.2 Databases with Built-in Blockchains

Several distributed databases are designed to have blockchain as its part [25, 28, 35, 36]. They are characterized by an internal database, supported query language (SQL mostly standard [26]), and a consensus protocol. These systems timestamp database transactions between nodes in a decentralized way, which provides an incorruptible record of history and Bitcoin-like Merkle proofs for committed transactions. The existing databases with blockchain allow only to check the transaction's presence if the whole data storage is observable for the user. However, the blockchain will enable one to extend cryptographic proofs into arbitrary transaction-driven objects.

This paper considers a database for a prototype cryptocurrency. It shows how to support query responses with proofs for transaction entries and user balances, i.e., for more involved transaction-driven objects. The implementation is an extension of the popular database system using its standard instruments by adding extra tables and modifying queries within the supported language.

3 PREREQUISITE: MERKLE AND PATRICIA TREES

Merkle and Patricia Trees play a vital role in blockchain data storage organization, and they are building blocks for the proposed architecture in the paper. The structures include both benefits of

- **Trees:** operations on elements (appending a new element, getting an element) take $O(lnN)$ operations, where N is number of elements. Hereafter $O(x)$ is a *Big O* [10].
- **Hashes:** verification of the (data storage) copies.

It is time-consuming and computationally expensive to check the entirety of each part. That is why Merkle and Patricia Trees are used whenever a system wants to verify data. The reference to these two concepts from the blockchain point of view is provided in the current Section.

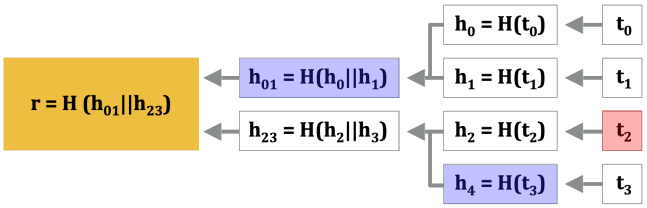


Figure 1: Merkle tree example, The data correctness of t_2 can be proved only with availability of data h_4 and h_{01}. Which can then be compared with root of the Merkel tree (r).

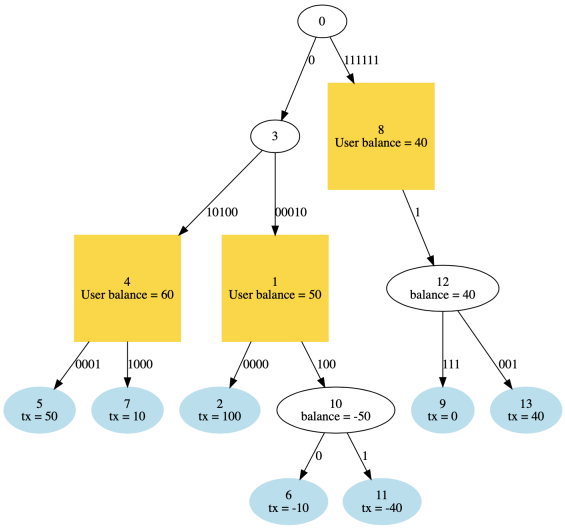


Figure 2: Example of the proposed Patricia structure

3.1 Merkle Tree

Merkle tree [27] is a binary tree built over a sequence of data pieces that aggregates hashes at each level till the root hash, storing information about the state of all data chunks. Root hash is used as a digital fingerprint of a block in the blockchain. Merkle trees store information in an effective way [33] such that verification takes $O(h) = O(ln(B))$, where B is the number of transactions per block, and h is the tree height. The Merkle trees are used in BitTorrent protocol [9], Interplanetary File System (IPFS), distributed Git Version Control System, etc.

The values in the tree nodes are filled in from the bottom up. If sibling nodes are odd in number at each level, then the last node in the sequence is duplicated and appended and is repeated recursively until one element is left. Therefore, it is convenient to use a simple recursive algorithm to build Merkle Tree.

3.2 Patricia Tree

The Ethereum project uses a modified version of the Merkel tree–modified Merkle Patricia trie (Patricia tree, MPT)–to store data about accounts, transactions, results of their execution, and other necessary data required for the system to function. MPT allows to

efficiently store key-value pairs and still provide verification of the stored data.

Every node has a hash value of its content. Key-value storage is used as the path on the MPT. Hex-prefix (HP) encoding is used on the trie data structure paths that differentiate between the type of nodes, i.e., extension or leaf nodes. Extension nodes have edges connected to a child node, and a node without a child node is called a leaf node. HP uses its encoding known as Recursive Lenght Prefix (RLP) for data serialization.

Patricia trie is entirely deterministic, i.e., with the same (key, value) bindings are guaranteed to be the same down to the last byte and therefore have the same root hash, provide the holy grail of efficiency for inserts, lookups, and deletes, and are much easier to understand and code than more complex comparison- based alternatives like red-black trees, where is the number of transactions.

4 PROTOTYPE IMPLEMENTATION

In our prototype cryptocurrency, we have users (unique strings) with non-negative balances and two types of transactions

- create a user with a given name and balance
- transfer some tokens from sender to receiver (without authorization).

The implementation is written in Python 3.7.0.

4.1 Implementation of Merkle Tree

Merkle tree is used to provide auditability for individual transactions. Like in blockchains, it is more effective (in terms of throughput expressed in transactions per second) to reach a consensus about a batch of transactions–block–instead of each one. So Merkle trees are computed to generate proof within blocks.

Merkle tree data structure stores information like transaction hash, the hash of user, position in the Merkle tree, and block id. For each block size, random transactions are generated and stored in the Merkle node. The hash of the path to reach a specific node is hashed and added in that particular node's Merkle node database.

4.2 Implementation of Patricia Tree

MPT is used to provide an auditability for user balance. We have a path to each user with at least one incoming transaction. Under it, we have an MPT subtree with transactions that changed the user's balance.

Consider a model cryptocurrency system that has the following types of transactions:

- **CREATE:** Creating an account for the user with or without initial balance.
- **SPEND:** Transfer of token from one account to another.

Like in Ethereum, each block of the blockchain contains the root hash of the user's state. With every transaction, the user's balance is updated. Thus, the new root hash value is stored in the next block of the blockchain. The leaves of the state tree contain a balance of the user, known as the balance node, and the user's key is used as a path to traverse to reach the balance node from the root. The balance node stores the root hash of transaction trie, i.e., incoming and outgoing transactions of a particular user. The transaction root is updated with any transaction taking place. For both sender and receiver, state root is updated and added to the blockchain. In Figure 2, We have three users $user_1$: 010100, $user_2$: 000010, $user_3$: 111111 with balances 60,50 and 40 respectively stored at balance node (marked in yellow). $User_1$ and $user_2$ have same prefix 0, therefore at node 3, the unique prefix is split into two nodes. $User_2$ has two separate outgoing transactions 1000 to $user_1$ of value 10 and 1001 to $user_3$ of value 40. Transaction trie also uses the same encoding like state tree where common transaction prefix is split.

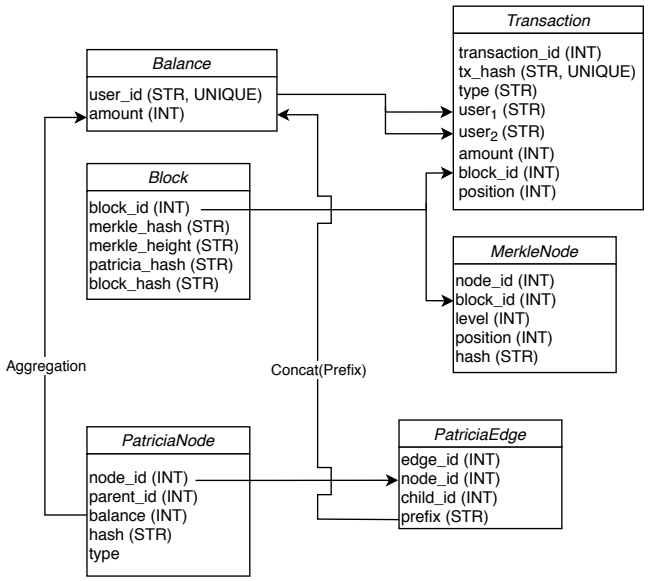


Figure 3: Proposed Database Structure

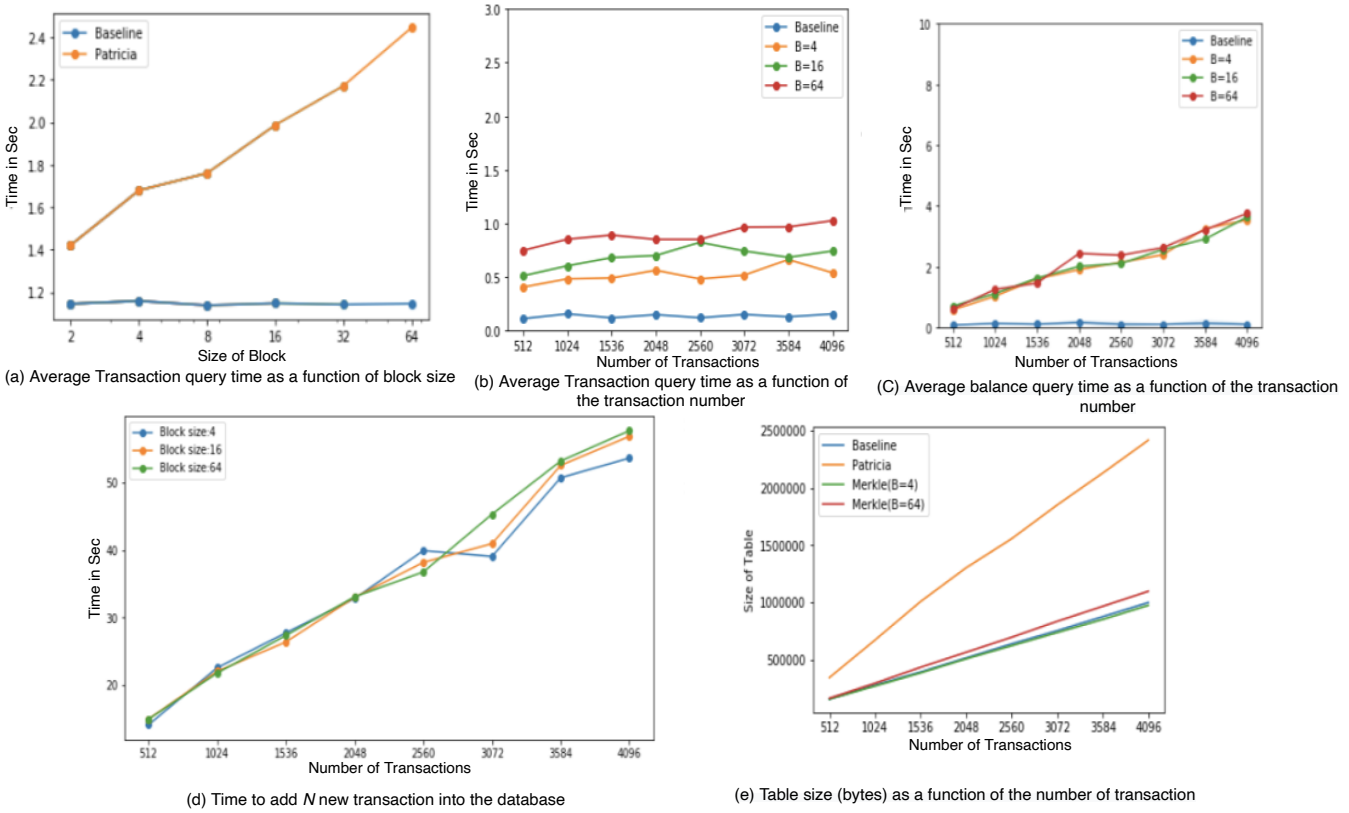
Note. *For ease of understanding, we use 0 and 1 as a string character for user address of length 6 and transaction hash of length 4 in Figure 2*

4.3 Data Storage

The data is stored in 6 tables. Figure 3 describes the data storage architecture of the proposed system. The transactions and users' balances are recorded in the transaction and balance tables, respectively. They are not specific to the blockchain. All newly committed transactions are grouped into blocks and stored in Merkle tree architecture. The precalculated information about Merkle tree nodes is stored in the *merklenode* table, and the block information is in block table. To support proofs for balances, we maintain a global Patricia tree. The precalculated information about its nodes and edges is stored in *patricianode* and *patriciaedge* tables, respectively.

In this model system, all transactions are processed in a fixed block size of the transaction that can be accommodated. The block size is the hyperparameter of the system. Each transaction block has its Merkle tree. In the database, we store Merkle tree nodes as a tuple, i.e., storing (the block number, the height of this node, the sequence number in this height, the hash value of this node). A hash is calculated by applying the hash function to the previous block's concatenation.

Each transaction is characterized by a user, transaction hash, and change of the balance, the prefix tree is updated as follows:



(a) Average Transaction query time as a function of block size

(b) Average Transaction query time as a function of the transaction number

(C) Average balance query time as a function of the transaction number

(d) Time to add N new transaction into the database

(e) Table size (bytes) as a function of the number of transaction

Figure 4: Performance results of processing different queries and memory consumption

- Find the vertex in the prefix tree corresponding to the user ID
- The vertex's balance is updated.
- After updating the user's vertex balance, the balance of all vertexes located below it and the transaction hash string is updated. In other words, the subtree of the node corresponding to each user contains information about all its transactions in the form of Merkel Patricia trie.
- Finally, recursion is performed to the root of the prefix tree. While recursing, the vertex hashes in the prefix tree are updated. The new hash of a vertex is obtained by applying the hash function to concatenate all the hashes of its child nodes.

5 EXPERIMENTS

To evaluate the performance of the prototype, we performed tests on a local machine. The database without any blocks and tree structures is used as a **baseline** algorithm, i.e., only two tables–transaction and balance (see Figure 4). The full blockchain-based data structure from Section 3 is used as a **proposed** solution. The experiments were performed on a computer with memory 16 GB 2400 MHz DDR4, Intel Core i9 running @2,3GHz.

- initialize empty database
- create U random users and commit a transaction with their initial balances
- generate and process N random one-to-one transfer transactions.

Note. *The initial balances equal* 10^6 *tokens, the amounts for transfers are uniform random integers from [1,100], and the number of transactions M always less than* 10^6*. So, the situation of too low user's balance to transfer is impossible during the computations. No user authorization is used during the experiment.*

We varied two parameters:

- the number of transfer transactions N
- the block size B (applicable only to the **proposed** system).

The following characteristics were measured

(A) Average transaction query time as function of block size
(B) Average transaction query time
(C) Average balance query time
(D) Time to add N new transactions into the database
(E) Database tables sizes.

The code to the repository is available at GitHub [1].

5.1 Transaction Request

Random existing transactions hashes were generated for the experiment. The average transaction query time as a function of the number of transactions N is shown in Figure 4(a). For **baseline**

system one need only to find the proper line in the transaction table. While for the **proposed** algorithm, one also needs to find and return the lines for the corresponding block's Merkle tree. The time is constant as a function of N, where the constant increases as $O(lnB)$ as shown in Figure 4(b).

5.2 Transaction Processing

The average time to add a new into the database is represented in Figure 4(d). The baseline algorithm add only a new line into the sorted transaction table and updates two lines in the sorted balance (sender's and receiver's) per transaction, i.e. we need $O(ln(U + T))$ operations, where $O(x)$ is the *Big O* [10]. The **proposed** system process transactions block-wise, i.e. batch-wise. And, in addition to transaction and balance tables update, per B transactions we need to generate and commit a Merkle tree with $O(B)$ for $O(B)$ operations elements and perform $O(B)$ Patricia trie updates for $O(ln(U+T))$ computation.

5.3 Balance Request

Random existing user names were generated for the experiment. The average balance query time as a function of the number of transactions N is shown in Figure 4(c). For **baseline** system one need only to find the proper line in the balance table.

5.4 Memory Usage

Memory usage per table as a function of the number of transactions is shown in Figure 4(e). All the tables sizes grow linearly as the number of their elements grows linearly. The Merkle structures for the bigger block size is also more significant due to the tree depth increase.

6 CONCLUSION

The paper presents and evaluates the performance of blockchain extension for data storage within the database. A particular example of an account-based prototype cryptocurrency is used. It is implemented using Python and PostgreSQL, and this approach is applicable for other database systems with SQL-like language. The prototype supports two types of queries with verifiable answers. All the supplementary data structures for proofs are precalculated during the insert and update events processing and are stored in the database. The numerical experiments show the expected overhead for the database blockchainization, which is acceptable in non-high-performance services. An addition of a consensus mechanism is needed to make blockchain on the top of the prototype.

Nevertheless, it is enough to estimate the computation and memory overhead for blockchainization. Additionally, a constructor of the necessary tree structure is required to implement practical interest. We consider the automated new queries addition, consensus protocol inclusion, stored procedures support as future work.

REFERENCES

[1] 2020. GitHub Repository. https://github.com/yashmadhwal/Blockchain-Extension-for-PostgreSQL-Data-Storage

[2] Naif Alzahrani and Nirupama Bulusu. 2018. Block-Supply Chain: A New Anti-Counterfeiting Supply Chain Using NFC and Blockchain. In *Proceedings of the 1st Workshop on Cryptocurrencies and Blockchains for Distributed Systems - CryBlock'18*. ACM Press, New York, New York, USA, 30–35. https://doi.org/10.1145/3211933.3211939

[3] Bitfury Group. 2016. On Blockchain Auditability. *bitfury.com* (2016), 1–40. https://bitfury.com/content/downloads/bitfury-white-paper-on-blockchain-auditability.pdf

[4] Eric A Brewer. 2000. Towards robust distributed systems (abstract). In *Proceedings of the nineteenth annual ACM symposium on Principles of distributed computing - PODC '00*. ACM Press, New York, USA, 7. https://doi.org/10.1145/343477.343502

[5] Vitalik Buterin. 2014. Ethereum White Paper: A Next Generation Smart Contract & Decentralized Application Platform. *Etherum* January (2014), 1–36. https://github.com/ethereum/wiki/wiki/White-Paper

[6] Christian Cachin. 2016. Architecture of the Hyperledger Blockchain Fabric. *IBM Research* July (2016).

[7] Miguel Castro and Barbara Liskov. 2002. Practical Byzantine fault tolerance and proactive recovery. *ACM Transactions on Computer Systems* 20, 4 (11 2002), 398–461. https://doi.org/10.1145/571637.571640

[8] E F Codd. 1970. A relational model of data for large shared data banks. *Commun. ACM* 13, 6 (6 1970), 377–387. https://doi.org/10.1145/362384.362685

[9] Bram Cohen. 2008. The BitTorrent Protocol Specification. (2008). http://www.bittorrent.org/beps/bep_0003.html

[10] Thomas H Cormen, Charles E Leiserson, and Ronald L Rivest. 2001. *Introduction to Algorithms , Second Edition*. 1184 pages. https://doi.org/10.2307/2583667

[11] Jean Dollimore, Tim Kindberg, and George Coulouris. 2005. *Distributed Systems: Concepts and Design*. Addison-Wesley. 944 pages.

[12] Joshua D. Drake and John C. Worsley. 2002. *Practical PostgreSQL*. O'Reilly Media, Inc. 640 pages.

[13] Cynthia Dwork, Nancy Lynch, and Larry Stockmeyer. 1988. Consensus in the presence of partial synchrony. *J. ACM* 35, 2 (4 1988), 288–323. https://doi.org/10.1145/42282.42283

[14] Cynthia Dwork and Moni Naor. 1992. Pricing via Processing or Combatting Junk Mail. In *Advances in Cryptology — CRYPTO' 92*. Springer Berlin Heidelberg, Berlin, Heidelberg, 139–147. https://doi.org/10.1007/3-540-48071-4{_}10

[15] Seth Gilbert and Nancy Lynch. 2002. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. *ACM SIGACT News* 33, 2 (6 2002), 51. https://doi.org/10.1145/564585.564601

[16] Tzipora Halevi, Fabrice Benhamouda, Angelo De Caro, Shai Halevi, Charanjit Jutla, Yacov Manevich, and Qi Zhang. 2019. Initial Public Offering (IPO) on Permissioned Blockchain Using Secure Multiparty Computation. In *2019 IEEE International Conference on Blockchain (Blockchain)*. IEEE, 91–98. https://doi.org/10.1109/Blockchain.2019.00021

[17] Simon Jentzsch and Christoph Jentzsch. 2018. EIP-1186: RPC-Method to get Merkle Proofs. https://github.com/ethereum/EIPs/issues/1186

[18] Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. 2017. Ouroboros: A Provably Secure Proof-of-Stake Blockchain Protocol. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 10401 LNCS. Springer, Cham, 357–388. https://doi.org/10.1007/978-3-319-63688-7{_}12

[19] Sunny King and Scott Nadal. 2012. PPCoin : Peer-to-Peer Crypto-Currency with Proof-of-Stake. *Self-published paper* (2012), 1–6. http://encryptopedia.org/ppcoin-proof-of-stake/

[20] Kari Korpela, Jukka Hallikas, and Tomi Dahlberg. 2017. Digital Supply Chain Transformation toward Blockchain Integration. In *Proceedings of the 50th Hawaii International Conference on System Sciences*. 4182–4191. https://doi.org/10.24251/HICSS.2017.506

[21] I. Kotsiuba, M. Nesterov, Y. Yanovich, I. Skarga-Bandurova, T. Biloborodova, and V. Zhygulin. 2019. Multi-Database Monitoring Tool for the E-Health Services. In *Proceedings - 2018 IEEE International Conference on Big Data, Big Data 2018*. https://doi.org/10.1109/BigData.2018.8622020

[22] Leslie Lamport, Robert Shostak, and Marshall Pease. 1982. The Byzantine Generals Problem. *ACM Transactions on Programming Languages and Systems* 4, 3 (7 1982), 382–401. https://doi.org/10.1145/357172.357176

[23] Yash Madhwal and Peter Panfilov. 2017. Blockchain And Supply Chain Management: Aircrafts' Parts' Business Case. In *Annals of DAAAM and Proceedings of the International DAAAM Symposium*. 1051–1056. https://doi.org/10.2507/28th.daaam.proceedings.146

[24] Nisha Malik, Priyadarsi Nanda, Xiangjian He, and RenPing Liu. 2019. Trust and Reputation in Vehicular Networks: A Smart Contract-Based Approach. In *2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*. IEEE, 34–41. https://doi.org/10.1109/TrustCom/BigDataSE.2019.00015

[25] Trent Mcconaghy, Rodolphe Marques, Andreas Müller, Dimitri De Jonghe, Troy Mcconaghy, Greg Mcmullen, Ryan Henderson, Sylvain Bellemare, and Alberto Granzotto. 2016. BigchainDB: A Scalable Blockchain Database. *BigchainDB* (2016), 1–65. http://www.noql.com

[26] Jim Melton. 1998. Database Language SQL. In *Handbook on Architectures of Information Systems*. Springer Berlin Heidelberg, Berlin, Heidelberg, 103–128. https://doi.org/10.1007/978-3-662-03526-9{_}5

[27] Ralph C. Merkle. 1988. A Digital Signature Based on a Conventional Encryption Function. In *Conference on the theory and application of cryptographic techniques*. Springer, Berlin, Heidelberg, 369–378. https://doi.org/10.1007/3-540-48184-2{_}32

[28] Muhammad Muzammal, Qiang Qu, and Bulat Nasrulin. 2019. Renovating blockchain with distributed databases: An open source system. *Future Generation Computer Systems* 90 (1 2019), 105–117. https://doi.org/10.1016/j.future.2018.07.042

[29] Satoshi Nakamoto. 2008. Bitcoin: A Peer-to-Peer Electronic Cash System. *www.bitcoin.org* (2008), 1–9. https://bitcoin.org/bitcoin.pdf

[30] Marc Pilkington. 2016. Blockchain Technology: Principles and Applications. In *Research Handbook on Digital Transformations*. Springer, 225 – 253. https://doi.org/10.4337/9781784717766.00019

[31] Qiuyun Shang and Allison Price. 2019. A Blockchain-Based Land Titling Project in the Republic of Georgia: Rebuilding Public Trust and Lessons for Future Pilot Projects. *Innovations: Technology, Governance, Globalization* 12, 3-4 (1 2019), 72–78.

[32] Melanie Swan. 2015. *Blueprint for a new economy*. Vol. 58. Cambridge University Press, Cambridge. 152 pages. https://doi.org/10.1017/CBO9781107415324.004

[33] Michael Szydlo. 2004. Merkle Tree Traversal in Log Space and Time. In *Advances in Cryptology - EUROCRYPT 2004*, Christian Cachin and Jan L Camenisch (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 541–554. https://doi.org/10.1007/978-3-540-24676-3{_}32

[34] Hien Thi Thu Truong, Miguel Almeida, Ghassan Karame, and Claudio Soriente. 2019. Towards Secure and Decentralized Sharing of IoT Data. In *2019 IEEE International Conference on Blockchain (Blockchain)*. IEEE, 176–183. https://doi.org/10.1109/Blockchain.2019.00031

[35] Manisha Verma and Debasis Ganguly. 2019. LiRME: Locally interpretable ranking model explanation. In *SIGIR 2019 - Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1281–1284. https://doi.org/10.1145/nnnnnnn.nnnnnnn

[36] Tommy van der Vorst. 2017. Catena: a distributed database based on a blockchain, accessible using SQL. https://github.com/pixelspark/catena

[37] Yury Yanovich, Igor Shiyanov, Timur Myaldzin, Ivan Prokhorov, Darya Korepanova, and Sergey Vorobyov. 2018. Blockchain-Based Supply Chain for Postage Stamps. *Informatics* 5, 4 (11 2018), 42. https://doi.org/10.3390/informatics5040042