





Data Management Challenges in Blockchain-Based Applications

Stanly Wilson , Kwabena Adu-Duodu , Yinhao Li , and Ellis Solaiman , Newcastle University, Newcastle Upon Tyne, NE1 7RU, U.K.

Omer Rana , Cardiff University, Cardiff, CF24 4AG, U.K.

Schahram Dustdar , Technische Universitat Wien, 1040, Vienna, Austria

Rajiv Ranjan , Newcastle University, Newcastle Upon Tyne, NE1 7RU, U.K.

Effective data management is crucial to ensure the security, integrity, and efficiency of blockchain systems. This study proposes a detailed data management taxonomy specifically designed for blockchain technology. The taxonomy provides a structured framework to categorize and address various aspects of data management in blockchain networks. It covers essential aspects, such as data flow, data storage, access control, and querying. It provides a systematic approach to understanding and addressing the unique challenges associated with managing data in blockchain systems. The framework enables informed decision making, promotes effective strategies, and helps to leverage the full potential of blockchain technology.

Blockchain technology's fast growth and acceptance have created a need for efficient data management strategies in this decentralized and immutable environment. As blockchain networks store vast amounts of data across multiple nodes, it is crucial to manage and organize these data effectively to ensure the system's integrity, security, and usability. Therefore, developing a comprehensive data management taxonomy is essential to tackle the unique challenges of data management in the blockchain.

Blockchain data management presents a few challenges, and we list four here:

- › **Scalability:** Storing and managing large volumes of data can be a challenge for blockchain networks. As the amount of data in the blockchain increases, so do the storage and processing requirements, which can affect the network's performance and efficiency.¹
- › **Privacy and confidentiality:** Although blockchain technology ensures transparency and immutability of data, it can lead to privacy and confidentiality

issues. This is because all members of the blockchain can view the stored data, which is a concern for organizations that need to store sensitive information.²

- › **Access control:** This is related to the second challenge. The task of safeguarding sensitive data while maintaining transparency is complicated due to the decentralized and immutable nature of blockchains. Balancing decentralization and trustlessness with controlled access and data privacy is a challenge.
- › **Querying:** Retrieving data stored on the blockchain can be complex, especially while dealing with large datasets. Blockchain has limited options for queries, and traditional query languages and their operations are not available on the blockchain. This creates the need for other approaches to query data on the blockchain.³

Keeping these challenges in mind, we bring a data management taxonomy for blockchain, which provides a structured framework that classifies and categorizes various aspects of data management specific to the blockchain. It offers a systematic approach to understanding, organizing, and addressing the challenges associated with storing, securing, retrieving, and analyzing data in a

blockchain network. This taxonomy serves as a guide to navigate the intricacies of blockchain data management and implement effective strategies.

The contributions of this article are as follows:

- › It provides detailed data management on blockchain, enabling a better understanding of the data landscape and its management requirements.
- › It identifies data management techniques and technologies required for blockchains.

The remaining sections are as follows. The following section describes the motivation and scenario. The "Taxonomy" section elaborates on the proposed taxonomy. The "Future Directions" section discusses the future works. Finally, the last section concludes the article.

MOTIVATION AND SCENARIO

In today's complex business networks, secure, trustworthy, and transparent data management is crucial. Traditional approaches relying on centralized servers often lack trust and verifiability due to data manipulations. As the number of stakeholders increases, different levels of data access are necessary, and systems must identify user actions. In industries where traceability and authenticity are crucial, technology is needed to connect stakeholders and ensure data integrity. Businesses previously had to add security, immutability, and transparency features externally, but blockchain technology integrates them intrinsically. Blockchain technology can enhance collaboration, accountability, and confidence through its decentralized and immutable ecosystem.

Consider, for example, supply chains: each is a complex matrix of manufacturers, retailers, consumers, and recyclers. As their networks grow more complex, the need for real-time transparency, traceability, and validation of product authenticity becomes more important and challenging.

- › *Data integrity*: Ensuring the integrity of data is challenging because parties may deny transactions, making it hard to hold them accountable.
- › *Access control*: With multiple parties involved, it becomes complex to regulate who can view, modify, or access specific data, leading to unauthorized access and data manipulations.
- › *Data privacy*: Protecting confidential data while providing necessary access is challenging and risks exposure.
- › *Data retrieval*: Retrieving accurate historical transaction records efficiently is challenging and time-consuming, which can impact decision-making and auditing processes.

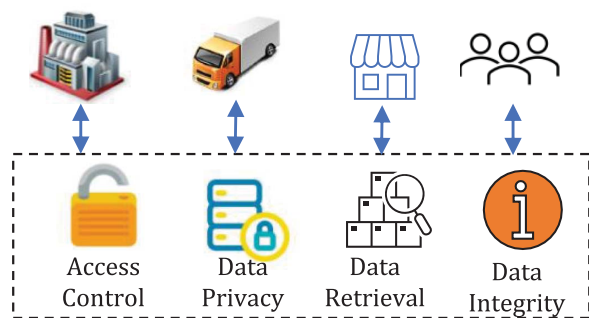


FIGURE 1. Data management requirements.

Figure 1 depicts the challenges of the scenario and requirements. These challenges constitute a trust deficit in the overall process. Blockchain technology can bring operational fluidity and trust reinforcement. In the next section, we delve into the data management taxonomy that outlines how blockchain supports modern business requirements.

TAXONOMY

Data management in blockchain refers to the processes and procedures that are either in place or required to efficiently store, organize, secure, and retrieve data on the blockchain. The decentralized nature of blockchain means that data are stored and processed on multiple nodes of the blockchain network, and all the nodes of the blockchain must have the same copy of all the consistent and accurate ledgers. For many reasons, data management is vital in ensuring data integrity and security, scalability, and efficiency in storing and retrieving data and making it available at a given time. Understanding how blockchain works and manages data can assist organizations and developers in making informed decisions and implementing effective strategies to leverage blockchain data. The proposed taxonomy focuses on four main aspects: data flow, storage, access control, and querying. The data management taxonomy for blockchain is given in Figure 2.

Data Flow

Blockchain has an immutability feature that permanently stores information. However, as a data storage system, it also needs to be able to store new information and change its state. To comprehend this concept, it is important to understand the two types of data that are stored on the blockchain: immutable/static and mutable/dynamic. Table 1 summarizes a few related works on data.

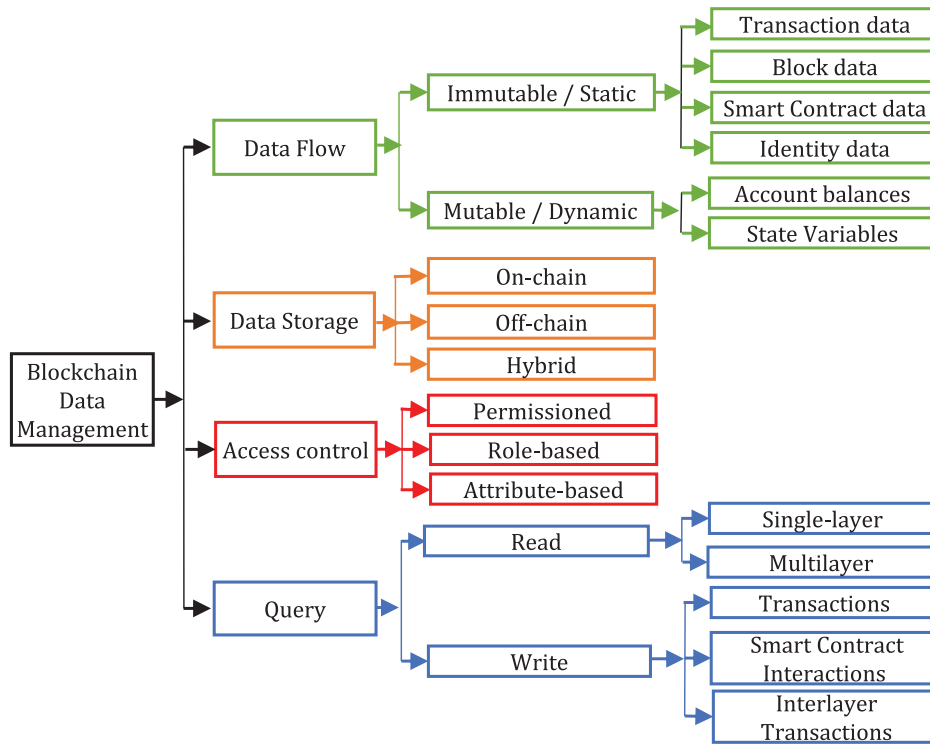


FIGURE 2. Data management taxonomy for blockchain.

Immutable/Static

This refers to the data on the blockchain that cannot be deleted/modified. One of the distinguishing features of blockchain is the immutability of data. Once the information is recorded on the blockchain, it is permanently stored. Any attempt to modify the information would invalidate the entire chain and can be traced back to the changed content. The aspect of immutability is achieved through various cryptographic mechanisms, such as asymmetric cryptography, hashing, and the specific consensus protocol on the blockchain. The immutable nature makes blockchain a secure platform on which to store data resistant to tampering or manipulation, building up trust and transparency.^{4,7}

Transaction Data

This is the information stored as part of a valid transaction. A transaction refers to an operation that changes the current state of information stored on the blockchain by adding something new or modifying its current state. It can typically involve transferring assets or tokens from one account to another, changing or updating information by executing smart contracts, or recording any data on the blockchain. Every valid transaction that modifies or changes the current state is recorded on the blockchain and is immutable once stored on the blockchain. Transaction data may typically contain the sender’s address, receiver’s address, assets, token information that gets transferred, transaction fees, timestamps, and digital signatures. Not all transactions become part of the blockchain. Only transactions approved by other nodes in the network or transactions that meet the requirements of the specific consensus algorithm become part of the blockchain. Every valid transaction will have a transaction hash, using which the transaction information can be retrieved and verified at a given point in time by those who have access to the network. More importantly, blockchain stores the transaction data chronologically and uses them in the consensus process to decide upon the latest information.⁵

TABLE 1. Blockchain platforms and their structure.

Work	Platform
Antonopoulos ⁴	Bitcoin
Zheng et al. ⁵	Ethereum
Androulaki et al. ⁶	Hyperledger
Paik et al. ²	Multiple blockchain platforms
Hafid et al. ¹	Multiple blockchain platforms

Block Data

Blockchain derives its name from the way it stores data in blocks. Every block in a blockchain contains several transactions, which are validated by other nodes and satisfy the consensus protocol. There is no fixed number of transactions for a block, but depending on the blockchain platform, the number of transactions in a block varies.¹ A block may contain the hash of the present block, the previous block's hash, several transactions, a timestamp indicating the creation of the block, gas expenses of block creation, and other related metadata. Each block is linked to the previous block, making a permanent and unalterable record of all of the transactions, making the information stored secure and reliable. A block can be retrieved either using its block number in the long chain of blocks or using its unique hash. The block data are designed to be secure, immutable, and transparent to all of the network nodes, ensuring data integrity and preventing fraudulent behavior. Like transaction data, blocks are also stored chronologically. The latest block will remain at the top of the blockchain stack.⁴

While discussing block data, it is apt to make note of distributed ledger technology (DLT). This is a digital system that provides a synchronized and decentralized view of a shared ledger or database to multiple participants. This eliminates the need for a central authority and allows for collective maintenance and validation of the ledger. Through cryptographic techniques and consensus algorithms, DLT ensures transparency, immutability, and security.⁸ Blockchain is one example of DLT. There are alternative methods for managing transactions besides storing them on blocks, like directed acyclic graph, which is used in IoT^a and Hashgraphs,^b to mention a few.²

Smart Contract Data

This refers to the data stored and processed in the smart contract on the blockchain. A smart contract is a self-executing code agreed upon by the participating entities that can automate complex operations and business logic on the network. Smart contract data may include contract code, state variables, mutually agreed terms of the contract, and any kind of data stored on the blockchain by the smart contract. All smart contract data are immutable and stored on the blockchain.⁹ The smart contract's programming language and structure may vary depending on the blockchain platform. An example of smart contract

language would be Solidity, used in Ethereum and Ethereum Virtual Machine (EVM)-based platforms.⁷

Identity Data

Every user/node has a unique address used to identify an account on a blockchain, and most of the transaction requires a sender's and receiver's addresses. The unique account address acts as the public name/identifier for the specific account. Blockchain uses the identity data, such as the account address, to authorize and authenticate the users to access and perform various operations (access controls), such as transferring assets or tokens or fetching any other information stored on the blockchain. While performing transactions, several digital signatures are also produced to prove the integrity of the transactions, which are verified using the private key of the node/user. The digital signatures verify that a specific user initiated the specific transaction and that the information is not tampered with.⁷ It is not only the users who have a unique address, but all the smart contracts stored on the blockchain have a unique address. Using the address, the authorized users can access the smart contract to automate their processes and bring trust to their business.¹⁰ The identity data are very critical for blockchain to ensure security and privacy, and they are immutable information stored on a blockchain.

Mutable/Dynamic

Mutable data refers to the information generated over a period that can be changeable or modifiable. They can be changed after they are initially set. On the blockchain, there cannot be a delete or update operation in its strict sense. Any information stored is immutable, and any updates to it will form another transaction, which will be stored on the blockchain. Hence, the actual value and modified value are available on the blockchain, but their current state may be different, like the balance of an account may be different at a given time due to the transactions. In smart contracts, state variables are mutable and can be updated by executing transactions that call functions within the contract.

Account Balances

Every user on the blockchain has an account and is required to have some tokens or coins specific to the blockchain to perform transactions. Since every transaction on a blockchain has a fee attached (whether big or small), it is mandatory to have some balance before making any transaction. Account balances are dynamic since they can change their state over a period, either by transferring to another account or spending as a part of processing a transaction.²

^a<https://www.iota.org/>

^b<https://hedera.com/>

State Variables

These are dynamic in nature since they change depending on the context or nature of a transaction. It may be the basic details required for a transaction or the information stored in a smart contract. Smart contracts can contain a variety of data variables, which may change their values because of a specific transaction. The input values in a smart contract may be the output of another completed transaction or a fresh value inserted by the user. Each time a smart contract performs a write transaction, it changes some variables or states of the smart contract. The current state of the blockchain contains the latest information on these variables.^{10,11}

Data Storage

Blockchain, as a decentralized, distributed ledger, stores data in a secure and immutable manner. The decentralized storing of data makes it difficult for intruders or malicious entities to manipulate or corrupt the data stored on the blockchain. Using various cryptographic mechanisms, blockchain protects stored data from being tampered with or modified without proper authentication. Though blockchain offers an innovative way to store data, storing any information in its present architecture is unsuitable. For example, storing large datasets, images, or media files on the blockchain directly may not be advisable due to the high cost of storing information on the chain and related transaction fees. Blockchain data are transparent, and every node has access to the data; it may not be a good idea to store sensitive information on the blockchain, which may breach privacy. Such situations necessitate different ways of storing information on the blockchain.² The data storage can be divided into three, depending on how applications and organizations store the information. Table 2 summarizes a few use cases that use different storage models.

On-Chain Data

This refers to the data stored on the blockchain directly and available to all of the network nodes. The on-chain data may be transaction and block data, smart contract data, or any other data that are directly stored on the blockchain and are transparent in nature. By their very nature, they are immutable, decentralized, and publicly available. Storing anything on the blockchain requires some cost, which may vary depending on the data size. Hence, it is not suitable to store large amounts of data on chain. It is ideal to store a minimal amount of data on chain. Since the data are available and transparent to all of the network nodes, blockchain is not ideal for storing sensitive data, like personal and health information and business secrets.

TABLE 2. Storage models and use cases.

Work	Storage	Application	Platform
Longo et al. ¹²	Hybrid	General Supply Chain	UnicalCoin
Liu et al. ¹³	Off chain	Data sharing	Ethereum
Wu et al. ¹⁴	Hybrid	Food Supply Chain	Hyperledger Fabric
Bocek et al. ¹⁵	Hybrid	—	Modum

While developing the business logic using smart contracts, one must take utmost care with what would be stored on the blockchain and optimize the size of data stored on chain.

Off-Chain Storage

Off-chain storage refers to storing data outside of the blockchain in a separate database or storage and storing only its reference on chain. Off-chain storage is not as secure or transparent as on-chain storage, and it may be vulnerable to manipulations. Depending on the way of storing, off-chain storage can be centralized and distributed. Centralized refers to storing information in some central databases from which the data can be retrieved. Distributed refers to storage that stores data in a decentralized manner. Examples of distributed storages are Interplanetary File System,^c BigchainDB,^d and OrbitDB.^e

Hybrid

As the name suggests, this combines on-chain and off-chain storage solutions and shares the benefits of both. An example of hybrid storage can be storing the critical data that require higher levels of security, immutability, and visibility to other nodes on chain. In contrast, the less important/critical data can be stored off chain to reduce the storage cost and improve overall performance. Hybrid storage solutions may be ideal for Internet-of-Things-related applications, health care, supply chains, or any organizations that would like to make their critical data available to the outside world while storing their sensitive data within themselves.

Access Controls

Access control mechanisms are necessary for the security and reliability of most applications. In a

^c<https://ipfs.tech/>

^d<https://www.bigchaindb.com/>

^e<https://github.com/orbitdb>

TABLE 3. Access controls and use cases.

Work	Platform	Application	Types
Valentin et al. ¹⁷	Ethereum	IoT	RBAC
Zaidi et al. ¹⁸	Ethereum	IoT	ABAC
Cruz et al. ¹⁹	Ethereum	General	RBAC
Yang et al. ²⁰	—	Cloud	RBAC and ABAC

ABAC: attribute-based access control; IoT: Internet of Things; RBAC: role-based access control.

blockchain, this refers to the mechanisms or a collection of policies that regulate who can access the network resources or their data. Access controls are essential in blockchain to ensure the security and integrity of data as well as to prevent unauthorized access to data and their modifications. They help to ensure compliance with privacy laws and regulations by controlling who can access sensitive information on the blockchain and prevent fraudulent or illegal behavior. Access controls promote mutual confidence among the participants of the network by providing trust in the data and its users. Rouhanie and Deters¹⁶ discusses the access control mechanisms in blockchain, and we elaborate on the most commonly used ones. Table 3 summarizes works dealing with access controls in different domains.

Permissioned Access Control

Blockchains can be either permissionless or permissioned. In permissionless blockchains, anyone can join and leave anytime, and its membership is open to anyone. All of the members of the network have the same rights to access data and create transactions, while, in permissioned blockchains, the membership is restricted. Permissioned access control refers to the restrictions provided by the blockchain to permit only selected users to be part of the network. By restricting the membership to only selected users, the network ensures privacy and confidentiality. The information stored on the blockchain will be available only to the members of the network and not to the outside world, as in the case of permissionless blockchains. This is a general access control system that a permissioned blockchain can provide by default.

Role-Based Access Control (RBAC)

RBAC restricts access to resources and information based on the roles and responsibilities of the users in an organization. The users are assigned roles to which responsibilities are attached, and, depending on the responsibilities, specific permissions may be associated that determine what actions or resources they are allowed to use or access. RBAC can be implemented

using smart contracts, which define various roles for the users and associated permissions. Smart contracts can enforce access control automatically based on predefined criteria. Using RBAC, blockchain provides granular control over the users and ensures that only authorized entities can create a transaction, view transaction details, and modify existing information.

Attribute-Based Access Control (ABAC)

ABAC uses characteristics or attributes of users to permit access to the network. It evaluates the attributes of users against a set of predefined policies or criteria. ABAC is becoming more popular, as it provides more flexible ways of providing access control not just based on the user roles but based on attributes such as the user's location, the time of the day, or even the content of a transaction. ABAC can be implemented using smart contracts that evaluate the relevant attributes of the user to determine whether they are authenticated to access the information. ABAC differs from RBAC based on its approaches to evaluating users. RBAC uses the roles and responsibilities of the users, while ABAC uses attributes for authentication. ABAC provides more fine-grained access control and can be dynamic and adaptable in its approach.

Query

In the context of a blockchain, querying is the process of retrieving information stored as transactions on single or multiple blocks. The querying may be done for different purposes and may depend on specific use cases. On a blockchain, information may be fetched for provenance. Provenance is the ability to track and trace the origin and authenticity of information stored on transactions spread over a period.²¹ Efficient querying can bring better data analysis and visualizations. The article by Li et al.¹¹ claims to be the first to investigate the aspect of querying in blockchain, and it identified the limited possibilities the blockchain could provide for querying. The framework provided by them has a SQL-like query layer for retrieving information from the blockchain. A few works^{22,23,24,25} used

authenticated data structures (ADSs) along with the blockchain to enhance the querying. They store either the full or partial information in the ADS, and a reference of the same is stored on the blockchain. By doing so, they try to leverage the benefits of traditional database queries on the blockchain. Przytarski et al.³ systematically study existing query mechanisms available in blockchain and discusses various challenges. All of these works deal with querying in single-layer blockchains, and querying in multilayer blockchains is still under research.

There are different ways in which queries can retrieve information from the blockchain. First, a node client connected with a blockchain can retrieve information stored using smart contracts. Here, the smart contracts communicate with the chain directly; depending on their access permissions, they may return the requested information. With full nodes (nodes with complete replicas), the smart contract can retrieve information without relying on any third party. The second method is using block explorers like Etherscan,^f and this method is available for Ethereum public blockchains. Using any browser, one can access them, and they provide various information regarding accounts, tokens, blocks, and transactions in a well-structured manner. The third method uses application programming interfaces, like Web3,^g which are provided by the specific blockchain platforms that can communicate the blockchain using various endpoints and retrieve information.² The queries discussed in this section are a combination of the methods mentioned. Table 4 summarizes a few works on queries. There are of two types of queries: read and write. Any query operation will have either of them or a combination of both.

Read Queries

These retrieve information stored on the chain without making any changes to data. Even though these queries do not alter the current data, they still count as a transaction on most of the blockchain and necessitate a gas fee to be paid to motivate the nodes to provide accurate data. Such operations can be simple as well as complex. We divide these operations into single layer and multilayer. In this section, single-layer query operations refer to the read operations in one blockchain (single layer), while multilayer read queries deal with retrieving information from multiple blockchains. Figure 3 shows categories of read queries.

The read operations may return a single attribute value (account balances of a specific account) or multiple attributes. We can divide these queries under four

^f<https://etherscan.io/>

^g<https://web3js.readthedocs.io/en/v1.8.2/>

TABLE 4. Blockchain queries.

Work	On Chain/ Off Chain	Query Types
EtherQL ¹¹	Yes	Standard, range, aggregate, and top K
FalconDB ²²	Yes	Standard, historical, range, and delta
SEBDB ²⁵	Yes	Standard, historical, range, and join
VQL ²³	Yes	Range, account, transaction, and block

categories: Boolean, standard, range, and historical queries. Single-layer Boolean queries return Boolean results (*yes/no* or *true/false*) for one or more attributes. Let us say a node needs to verify if a transaction is genuine. It can do so by checking if the transaction is recorded in a blockchain. The output will confirm whether the transaction exists in the blockchain and is authentic. Blockchain access control involves using Boolean queries to determine whether certain operations are permitted. The result of these queries determines the level of permissions granted. A single-layer standard query refers to a query that retrieves simple results from the blockchain that involve single or multiple attributes/parameters. The information it retrieves may be a specific transaction or block details, account details, or some information stored on a smart contract.

Single-layer range queries bring values between some specific range or interval. All of the attributes or some attributes can have ranges specified in the query. The retrieved information may be a single result or a collection of results. Single-layer historical queries have relevance in blockchain transactions. The data in the blockchain are immutable, and every transaction is always available on the blockchain. Hence, the historical information of a specific product and its provenance on blockchain in various states can be traced and verified using blockchain transaction details. Historical queries can also provide the ownership details of the specific product in its journey. Queries of these kinds are not directly available but depend on the smart contract's logic deployed, like various mappings or using the transaction and block hashes.

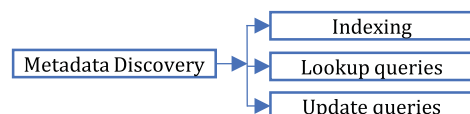


FIGURE 3. Read queries.

Multilayer read queries are meant to fetch data from multiple blockchains comprising multiple attributes/parameters. They need to query data that satisfy multiple conditions across multiple layers of blockchains. Such queries are often complex and may involve different access control mechanisms while dealing with different blockchains. These queries may be dependent or independent: dependent if the results from one chain act as input for the query in another chain and independent if they can be executed independently. Similar to single layer, we can categorize multilayer queries into four types. Multilayer Boolean queries involve querying based on multiple criteria on different blockchains in which each condition may return a Boolean result. Situations of sending some amount from one chain to another require verifying whether an account has sufficient balance in one chain and checking whether the receiver exists in another chain. Queries of this sort may be part of bigger operations, and depending on the Boolean result, further operations can be carried out. In the case of the example mentioned, the decision would be either to send the amount or not.

Multilayer standard queries contain multiple attributes and return results from multiple blockchains. The information they retrieve may be independent or dependent. Examples of independent results may be returning the account balances of two accounts in two blockchains. Such a query just returns the values and is independent of others. Examples of dependent results can be retrieving transaction details of an account in the first chain that has transferred some amount to an account in a second chain, along with proof for that transaction from both chains. Such a query may require having information from the first chain, which goes as input to the second chain. Multilayer range queries retrieve data depending on multiple criteria, simultaneously enabling more precise and accurate data retrieval. Multilayer historical queries support provenance queries in multilayer blockchains. They are quite helpful in tracking the asset journey in a multilayer blockchain.

Write Queries

Write queries in blockchain refer to the process of adding or updating data. They are used to initiate transactions that create new records or modify existing ones, ultimately updating the state of the blockchain. They need to undergo several valuations to ensure the integrity and security of the blockchain. While performing a write query, they may be accompanied by read queries that mostly do the validation processes. Any write operation is a new blockchain transaction requiring a transaction fee.

Transactions

These are the basic building blocks of a blockchain. Anything that changes the current state in a blockchain can be called a transaction, and a transaction is an immutable record stored on a blockchain. They may or may not be accompanied by a read operation. In situations like transferring amounts or tokens or moving anything on chain, some verification may be required, like whether the execution account has the right to perform such operations, whether the account has sufficient funds to transfer, or whether the token it tries to transfer belongs to the smart contract account or not. It necessitates a read operation before writing. An example of a transaction that does not require a read operation would be creating a new account or asset on the blockchain, as there is no prior state to be verified.

Smart Contract Interactions

These are the write operations that modify the data stored on a smart contract. Smart contracts can store various data types and perform multiple write operations depending on the design. They can write data to the blockchain as a stand-alone data structure, like mapping; generate tokens and transfer them between accounts; perform payments when conditions are met; or enforce access control. A smart contract can also change data stored on another smart contract if it has sufficient permissions to carry it out.

Interlayer Transactions

This refers to the operations that transfer tokens, assets, or data between different blockchains. Several methods can empower this process, and a two-way peg is one of the most common ways of moving between different layers.²⁶ It makes use of sidechains to achieve this process. To illustrate the example, consider two blockchains: a mainchain and a sidechain. Suppose a client in the mainchain wants to transfer tokens to another. In that case, it has to send those tokens to some designated nodes on the mainchain, which lock the tokens in the mainchain and generate corresponding tokens in the sidechains. The client can use the tokens created by the sidechains to transact with the target blockchains. As the client uses the tokens generated by the sidechains, the locking nodes destroy the respective tokens locked in the main chain. This process is referred to as a two-way peg.

FUTURE DIRECTIONS

Data Flow

Future research should prioritize the development of standards for block sizes across different DLTs. This is

because variations in block size can have a significant impact on scalability and efficiency. In addition, ensuring interoperability among diverse DLTs is crucial for seamless cross-chain data sharing. More research is required to develop new methods to enable secure cross-chain communication and data sharing. Smart contracts play a crucial role in automating blockchain systems, and one promising area of research is to improve their capabilities. There are many smart contract languages and multiple virtual environments to compile them. It is necessary to develop standardized and cross-compatible smart contract languages that can function effectively across various DLT platforms. Such research can address the challenges related to differences in data types and semantics between different smart contract languages.

Data Storage

As blockchain networks continue to expand, scalability issues related to on-chain data storage become more significant. Future research can explore innovative approaches, such as sharding and layer-2 solutions, to address these challenges. Off-chain storage solutions provide flexibility but can introduce risks of data tampering. Further research is needed to develop cryptographic techniques and protocols that ensure the immutability of off-chain stored data. Another promising direction is the seamless integration of decentralized storage solutions with blockchain networks. Such integration could enable smart contracts to access and interact with larger datasets and external resources efficiently.

Access Control

Exploring the possibility of self-adaptive access control is a valuable endeavor. Such systems can assign or revoke roles based on real-time conditions or behaviors. However, further research is necessary to investigate the implementation of these self-adaptive systems. In addition, developing methods and standards for interoperable access control across multiple blockchains would enable the seamless exchange of information between ecosystems while adhering to consistent policies.

Query

Blockchain interoperability is becoming increasingly important. To facilitate cross-chain transactions and data sharing, it is important to develop query mechanisms that can update data across multiple interconnected blockchains. This has a lot of potential to improve the functionality and usefulness of blockchain technology. Future research in querying should explore ways to leverage metadata for more efficient and

context-aware query processing. This involves developing techniques to generate queries based on available metadata dynamically, improving query optimization, and enhancing the precision of search results. Metadata-driven query engines can significantly improve data discovery and retrieval in complex blockchain environments.

CONCLUSION

The taxonomy offers a well-organized framework for comprehending, categorizing, and dealing with different aspects of data management within the context of blockchain technology. It provides a standardized way of discussing data management concerns that are unique to blockchain technology. The taxonomy encompasses dimensions such as data flow, storage options, access control mechanisms, and data retrieval, covering the essential aspects of managing data in blockchain networks. Researchers can gain valuable insights into the complexities of blockchain data management and develop effective strategies for implementing solutions in blockchain systems by using the taxonomy. Furthermore, the taxonomy serves as a guide to navigating the complexities of blockchain data management, enabling users to make informed decisions regarding data storage, access control, privacy measures, and scalability solutions. As the field of blockchain data management continues to evolve and move forward, the proposed taxonomy will adapt and expand to incorporate emerging trends, technologies, and best practices.

ACKNOWLEDGMENT

This work was supported in part by the Engineering and Physical Sciences Research Council "Digital Economy" program: EP/V042521/1 and EP/V042017/1.

REFERENCES

1. A. Hafid, A. S. Hafid, and M. Samih, "Scaling blockchains: A comprehensive survey," *IEEE Access*, vol. 8, pp. 125,244–125,262, Jul. 2020, doi: [10.1109/ACCESS.2020.3007251](https://doi.org/10.1109/ACCESS.2020.3007251).
2. H. Y. Paik, X. Xu, H. M. N. D. Bandara, S. U. Lee, and S. K. Lo, "Analysis of data management in blockchain-based systems: From architecture to governance," *IEEE Access*, vol. 7, pp. 186,091–186,107, Dec. 2019, doi: [10.1109/ACCESS.2019.2961404](https://doi.org/10.1109/ACCESS.2019.2961404).
3. D. Przytarski, C. Stach, C. Gritti, and B. Mitschang, "Query processing in blockchain systems: Current state and future challenges," *Future Internet*, vol. 14, no. 1, Dec. 2021, Art. no. 1, doi: [10.3390/fi14010001](https://doi.org/10.3390/fi14010001).
4. A. M. Antonopoulos, *Mastering Bitcoin: Unlocking Digital Cryptocurrencies*. Sebastopol, CA, USA: O'Reilly Media, Inc., 2014.

5. Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, "An overview of blockchain technology: Architecture, consensus, and future trends," in *Proc. IEEE Int. Congr. Big Data (BigData Congr.)*, 2017, pp. 557–564, doi: [10.1109/BigDataCongress.2017.85](https://doi.org/10.1109/BigDataCongress.2017.85).
6. E. Androulaki et al., "Hyperledger fabric: A distributed operating system for permissioned blockchains," in *Proc. 13th EuroSys Conf. (EuroSys)*, 2018, pp. 1–15, doi: [10.1145/3190508.3190538](https://doi.org/10.1145/3190508.3190538).
7. T. T. A. Dinh, R. Liu, M. Zhang, G. Chen, B. C. Ooi, and J. Wang, "Untangling blockchain: A data processing view of blockchain systems," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 7, pp. 1366–1385, Jul. 2018, doi: [10.1109/TKDE.2017.2781227](https://doi.org/10.1109/TKDE.2017.2781227).
8. A. Sunyaev, "Distributed ledger technology," in *Internet Computing*. Cham, Switzerland: Springer-Verlag, 2020, pp. 265–299.
9. P. Mukherjee and C. Pradhan, "Blockchain 1.0 to blockchain 4.0—The evolutionary transformation of blockchain technology," in *Blockchain Technology: Applications and Challenges*, S. K. Panda, A. K. Jena, S. K. Swain, and S. C. Satapathy, Eds. Cham, Switzerland: Springer, 2021, pp. 29–49.
10. S. N. Khan, F. Loukil, C. Ghedira-Guegan, E. Benkhalifa, and A. Bani-Hani, "Blockchain smart contracts: Applications, challenges, and future trends," *Peer-to-Peer Netw. Appl.*, vol. 14, no. 5, pp. 2901–2925, Apr. 2021, doi: [10.1007/s12083-021-01127-0](https://doi.org/10.1007/s12083-021-01127-0).
11. Y. Li, K. Zheng, Y. Yan, Q. Liu, and X. Zhou, "EtherQL: A query layer for blockchain system," in *Database Systems for Advanced Applications*, S. Candan, L. Chen, T. Pedersen, L. Chang, and W. Hua, Eds. Cham, Switzerland: Springer, 2017, pp. 556–567.
12. F. Longo, L. Nicoletti, A. Padovano, G. d'Atri, and M. Forte, "Blockchain-enabled supply chain: An experimental study," *Comput. Ind. Eng.*, vol. 136, pp. 57–69, Oct. 2019, doi: [10.1016/j.cie.2019.07.026](https://doi.org/10.1016/j.cie.2019.07.026).
13. S. Liu, J. Wu, and C. Long, "IoT meets blockchain: Parallel distributed architecture for data storage and sharing," in *Proc. IEEE Int. Conf. Internet Things IEEE Green Comput. Commun. IEEE Cyber, Phys. Soc. Comput. IEEE Smart Data*, 2018, pp. 1355–1360, doi: [10.1109/Cybermatics_2018.2018.00233](https://doi.org/10.1109/Cybermatics_2018.2018.00233).
14. H. Wu et al., "Data management in supply chain using blockchain: Challenges and a case study," in *Proc. 28th Int. Conf. Comput. Commun. Netw.*, 2019, pp. 1–8, doi: [10.1109/ICCCN.2019.8846964](https://doi.org/10.1109/ICCCN.2019.8846964).
15. T. Bocek, B. B. Rodrigues, T. Strasser, and B. Stiller, "Blockchains everywhere - A use-case of blockchains in the pharma supply-chain," in *Proc. IFIP/IEEE Symp. Integr. Netw. Service Manage.*, 2017, pp. 772–777, doi: [10.23919/INM.2017.7987376](https://doi.org/10.23919/INM.2017.7987376).
16. S. Rouhani and R. Deters, "Blockchain based access control systems: State of the art and challenges," in *Proc. IEEE/WIC/ACM Int. Conf. Web Intell.*, 2019, pp. 423–428.
17. M. Valentin, C. Pahl, N. E. Ioini, and H. R. Barzegar, "A blockchain-based access and management system for IoT devices," in *Proc. 8th Int. Conf. Internet Things, Syst., Manage. Secur.*, 2021, pp. 1–8, doi: [10.1109/IOTSMS53705.2021.9704951](https://doi.org/10.1109/IOTSMS53705.2021.9704951).
18. S. Y. A. Zaidi et al., "An attribute-based access control for IoT using blockchain and smart contracts," *Sustainability*, vol. 13, no. 19, Sep. 2021, Art. no. 10556, doi: [10.3390/su131910556](https://doi.org/10.3390/su131910556).
19. J. P. Cruz, Y. Kaji, and N. Yanai, "RBAC-SC: Role-based access control using smart contract," *IEEE Access*, vol. 6, pp. 12,240–12,251, Mar. 2018, doi: [10.1109/ACCESS.2018.2812844](https://doi.org/10.1109/ACCESS.2018.2812844).
20. C. Yang, L. Tan, N. Shi, B. Xu, Y. Cao, and K. Yu, "AuthPrivacyChain: A blockchain-based access control framework with privacy protection in cloud," *IEEE Access*, vol. 8, pp. 70,604–70,615, Apr. 2020, doi: [10.1109/ACCESS.2020.2985762](https://doi.org/10.1109/ACCESS.2020.2985762).
21. R. Neisse, G. Steri, and I. Nai-Fovino, "A blockchain-based approach for data accountability and provenance tracking," in *Proc. 12th Int. Conf. Availability, Rel. Secur.*, 2017, pp. 1–10, doi: [10.1145/3098954.3098958](https://doi.org/10.1145/3098954.3098958).
22. Y. Peng, M. Du, F. Li, R. Cheng, and D. Song, "FalconDB: Blockchain-based collaborative database," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2020, pp. 637–652, doi: [10.1145/3318464.3380594](https://doi.org/10.1145/3318464.3380594).
23. H. Wu, Z. Peng, S. Guo, Y. Yang, and B. Xiao, "VQL: Efficient and verifiable cloud query services for blockchain systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 6, pp. 1393–1406, Jun. 2022, doi: [10.1109/TPDS.2021.3113873](https://doi.org/10.1109/TPDS.2021.3113873).
24. C. Zhang, C. Xu, J. Xu, Y. Tang, and B. Choi, "GEM2-tree: A gas-efficient structure for authenticated range queries in blockchain," in *Proc. IEEE 35th Int. Conf. Data Eng.*, 2019, pp. 842–853, doi: [10.1109/ICDE.2019.00080](https://doi.org/10.1109/ICDE.2019.00080).
25. Y. Zhu, Z. Zhang, C. Jin, A. Zhou, and Y. Yan, "SEBDB: Semantics empowered blockchain database," in *Proc. IEEE 35th Int. Conf. Data Eng.*, 2019, pp. 1820–1831, doi: [10.1109/ICDE.2019.00198](https://doi.org/10.1109/ICDE.2019.00198).
26. R. Belchior, A. Vasconcelos, S. Guerreiro, and M. Correia, "A survey on blockchain interoperability: Past, present, and future trends," *ACM Comput. Surv.*, vol. 54, no. 8, pp. 1–41, 2022, doi: [10.1145/3471140](https://doi.org/10.1145/3471140).

STANLY WILSON is a Ph.D. student at the School of Computing at Newcastle University, Newcastle Upon Tyne, NE1 7RU, U.K. Contact him at s.w.palathingal2@newcastle.ac.uk.

KWABENA ADU-DUODU is a Ph.D. student at the School of Computing at Newcastle University, Newcastle Upon Tyne, NE1 7RU, U.K. Contact him at k.adu-duodu2@newcastle.ac.uk.

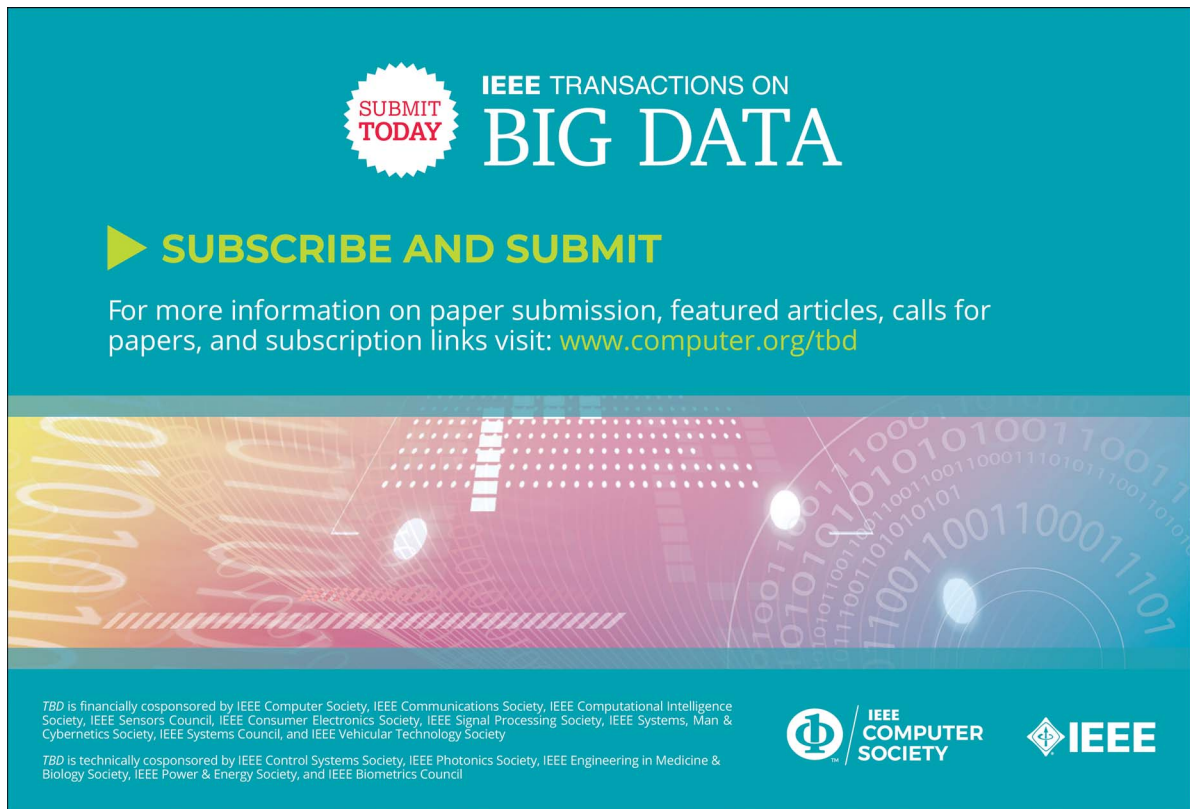
YINHAO LI is a lecturer with the School of Computing at Newcastle University, Newcastle Upon Tyne, NE1 7RU, U.K. Contact him at yinhao.li@newcastle.ac.uk.

ELLIS SOLAIMAN is a senior lecturer in the School of Computing at Newcastle University, Newcastle Upon Tyne, NE1 7RU, U.K. Contact him at ellis.solaiman@newcastle.ac.uk.

OMER RANA is a professor of performance engineering at Cardiff University, Cardiff, CF24 4AG, U.K. Contact him at ranaof@cardiff.ac.uk.

SCHAHRAM DUSTDAR is a full professor of computer science, heading the Research Division of Distributed Systems at Technische Universitat Wien, 1040, Vienna, Austria. Contact him at dustdar@dsg.tuwien.ac.at.

RAJIV RANJAN is a chair professor of computing science and the Internet of Things at Newcastle University, Newcastle Upon Tyne, NE1 7RU, U.K. Contact him at raj.ranjan@ncl.ac.uk.



SUBMIT TODAY IEEE TRANSACTIONS ON **BIG DATA**

SUBSCRIBE AND SUBMIT

For more information on paper submission, featured articles, calls for papers, and subscription links visit: www.computer.org/tbd

TBD is financially cosponsored by IEEE Computer Society, IEEE Communications Society, IEEE Computational Intelligence Society, IEEE Sensors Council, IEEE Consumer Electronics Society, IEEE Signal Processing Society, IEEE Systems, Man & Cybernetics Society, IEEE Systems Council, and IEEE Vehicular Technology Society

TBD is technically cosponsored by IEEE Control Systems Society, IEEE Photonics Society, IEEE Engineering in Medicine & Biology Society, IEEE Power & Energy Society, and IEEE Biometrics Council

IEEE COMPUTER SOCIETY IEEE