Data Science and Engineering (2023) 8:196–219
https://doi.org/10.1007/s41019-023-00212-z

REVIEW/SURVEY PAPERS

A Survey on the Integration of Blockchains and Databases

Changhao Zhu[1] · Junzhe Li[1] · Ziyue Zhong[1] · Cong Yue[2] · Meihui Zhang[1]

Received: 12 January 2023 / Revised: 28 February 2023 / Accepted: 2 April 2023 / Published online: 24 April 2023

Abstract
The success of blockchain technology in cryptocurrencies reveals its potential in the data management field. Recently, there is a trend in the database community to integrate blockchains and traditional databases to obtain security, efficiency, and privacy from the two distinctive but related systems. In this survey, we discuss the use of blockchain technology in the data management field and focus on the fusion system of blockchains and databases. We first classify existing blockchain-related data management technologies by their locations on the blockchain-database spectrum. Based on the taxonomy, we discuss three types of fusion systems and analyze their design spaces and trade-offs. Then, by further investigating the typical systems and techniques of each type of fusion system and comparing the solutions, we provide insights of each fusion model. Finally, we outline the unsolved challenges and promising directions in this field and believe that fusion systems will take a more important role in data management tasks. We hope this survey can help both academia and industry to better understand the advantages and limitations of blockchain-related data management systems and develop fusion systems that meet various requirements in practice.

Keywords Blockchains · Databases · Data management · Introductory and survey

1 Introduction

Blockchain technology has come into people's view with the release of the Bitcoin white paper [1] in 2008. Since then, more and more cryptocurrencies and decentralized applications are adopting blockchain technology. Blockchain has taken the world by storm in the past decades.

The huge success of blockchain technology raises people's interest in applying it to the data management field. A blockchain is essentially a novel data management system, which is maintained by multiple participants (or nodes). Compared to traditional database systems, there may be some participants behaving unexpectedly, but blockchains hold some promising properties under such a circumstance to protect the integrity of data.

- *Decentralization.* There is no central node in a blockchain system and every node in the network holds a replica of the data. In this way, the blockchain eliminates the risks that come with a centralized storage schema in traditional databases, i.e., malicious or failed central storage may cause the loss of data.
- *Immutability.* Once data are appended to the blockchain and confirmed by the majority of the chain's participants, it can never be replaced or reversed as the records are linked one after another with hash values. This marks blockchains as different from regular databases, in which information can be easily edited or deleted.
- *Tamper-Proof.* When mining a new block, metadata of current system states and corresponding proofs are generated and distributed to the network with the replication of the block. Since the proof is guaranteed by cryptography methods, any tiny alteration to the data will lead to a failure of validation. If there is any conflict during block validation, the participant can immediately recognize that

✉ Meihui Zhang
meihui_zhang@bit.edu.cn

Changhao Zhu
zhuchanghao@bit.edu.cn

Junzhe Li
junzhe_li@bit.edu.cn

Ziyue Zhong
ziyue_zhong@bit.edu.cn

Cong Yue
yuecong@comp.nus.edu.sg

1 Beijing Institute of Technology, Beijing, China

2 National University of Singapore, Singapore, Singapore

the block has been tampered with, then he can refuse this block to protect the security of the data.

- *Provenance.* Since the immutability of blockchain, the only accepted way to modify what has already been on the chain is to create a new log and append it to the chain to declare the invalidity of previous data. This mechanism ensures that every modification of data entry can be recorded as a trail, from which one can clearly obtain the history status of the data.

Despite the strong guarantee in data security, blockchain is still far from an ideal data management system. It suffers from low performance, high resource consumption, and potential privacy concerns.

- *Performance.* With its underlying chain structure, blockchain has to process each transaction serially. Moreover, other participants validate the received block by replaying the transactions in it, which is also a sequential process. These two linear transaction processing steps have a significant impact on the blockchain's performance. It is reported that Bitcoin, as a representative blockchain system, only achieves a throughput of 7 transactions/second. In contrast, a commercial database system can easily process 2000 to 56,000 transactions in one second [2].
- *Resource Consumption.* On one hand, as the transactions go on, the append-only ledger consumes more and more storage, which will be a burden for devices with limited storage capacity such as smartphones or even personal computers. On the other hand, the mining procedure requires participants to compete with others to calculate a specific problem, while only one of them wins the right to append a block, which wastes massive energy and computing resources.
- *Privacy Issues.* Every participant in a blockchain network holds the full copy of data due to the verification need. However, this is at the cost of some privacy concerns. In real-world business applications, companies will never want collaborators or customers to access their sensitive information, while this goal can be easily achieved by leveraging views in databases.

Apparently, blockchain technology has its superiority and defect, and neither it nor a database can perfectly undertake all the requirements of modern data management tasks. Fortunately, blockchains and databases share so many similar technical concepts and solutions, making it possible to combine the strengths of security, efficiency, and privacy from both sides. For example, transactions in both systems result in state changes and should hold ACID properties to ensure their reliability. Smart contracts in blockchains are corresponding to stored procedures in databases, as they aggregate transactions. Moreover, both systems adopt indexes to satisfy various requirements, i.e., tamper-proof and verifiability for blockchains, and efficient query for databases.

We have noticed that there are massive works trying to integrate blockchain and database technologies to develop a fusion system that protects data integrity and processes transactions effectively at the same time. Though the integration of blockchains and databases has attracted more and more attention, there are few discussions about it. At present, most of the surveys about blockchains [3–9] concentrate on some specific domains, instead of a comprehensive study of the trend of fusion. We argue that drawing a whole picture is of vital importance, as it will better guide the database community to develop systems that fit various real-world needs.

Difference with Existing Works Existing surveys only focused on some specific aspects of blockchains in the data management field. For example, Wang et al. [6] investigate the storage and query techniques of blockchains, while the authors of [5, 8] focus on the applications in specific domains. Other aspects including system architecture [3], query processing [4], and sharding technique [7, 9, 10] have also been analyzed. There are also surveys [11, 12] trying to comprehensively analyze blockchains as a data management system; however, their goal is to dichotomize blockchains and databases, and to compare the two systems.

The trend of the fusion between blockchains and databases has also been noticed and analyzed in other works. Based on the comparisons between blockchains and distributed databases, Ruan et al. [13] discuss the fusion trend and some representative works. Recently, the authors of [14] conduct extensive experiments on some hybrid blockchain database systems and reveal the variety of design choices of such systems. However, due to the experimental limitations, only a few systems are studied. Thus, some most recent works are not covered and discussed.

Contributions In this paper, we conduct a comprehensive survey on the integration of blockchains and databases in the data management field. To sum up, we made the following contributions.

- We propose the blockchain-database spectrum, a framework to analyze the works about blockchains in the data management field, and recognize the trend of integrating blockchains and traditional databases. We further identify three typical models of the fusion, namely *database-oriented blockchains*, *blockchain-oriented databases*, and *hybrid systems*, and conduct a comprehensive comparison of the three types of systems in the design spaces and trade-offs.
- We review each of the representative systems of *database-oriented blockchains*, *blockchain-oriented databases*, and *hybrid systems*. Besides, we summarize and evaluate the techniques used in each model, which provides insights into each fusion model.

- Based on the exhaustive research and analysis of existing works, we discuss the limitations of existing methods for blockchain-related data management systems and provide future research directions.

The rest of this paper is organized as follows. First, the preliminaries are provided in Sect. 2, including a basic introduction to blockchains and databases and the blockchain-database spectrum. We also classify existing blockchain-related data management technologies by their coordinates on the blockchain-database spectrum and identify their design spaces in this section. We review the representative systems and techniques of *database-oriented blockchains*, *blockchain-oriented databases*, and *hybrid systems* in Sects. 3–5, respectively. Then, we compare these systems and provide challenges and opportunities in the blockchain-related data management field in Sect. 6. Finally, Sect. 7 concludes the paper.

2 Preliminaries

2.1 Backgrounds

We begin this section with some basic information about blockchains and databases to provide a primary impression of the two different but relevant technologies.

2.1.1 Blockchain

Blockchain is an innovative data storage and management technology that integrates a variety of established technologies, including high-performance data storage, peer-to-peer networks, cryptography, consensus protocols, etc. The concept of blockchain originated from Bitcoin, which is proposed by Satoshi Nakamoto [1], and most of the existing blockchain systems also follow the chain structure in Bitcoin. Taking Bitcoin as an example, the structure of a typical blockchain is shown in Fig. 1a. Blocks are connected in a linked list and new blocks can only be added at the end of the chain. Therefore, all nodes in the blockchain system store the blocks and the transactions in a consistent order. The block, the basic structural unit of the blockchain, consists of a block header containing metadata and a block body containing transaction data. Block header consists of block height, the previous block's hash value, timestamp, nonce, miner signature, and Merkle root, and block body can be viewed as a collection of transaction records consisting of multiple transactions. For example, the block body of the Bitcoin system contains a Merkle tree consisting of approximately 2,500 transaction records that have been hashed, each consisting of information such as transaction hash, inputs, outputs, timestamps, and fees. Blocks are connected by hash values. The hash value of each block is obtained by re-hashing the Merkle tree's root, the previous block's hash value, and other information. Any change in transaction data in a block will cause a change in the hash value of this block, which in turn will change all subsequent blocks along the chain. To sum up, blockchain incorporates the hash function in the chain structure, making data tampering infeasible in blockchain and enhancing data storage security.

For a clear understanding of the blockchain hierarchy, we abstract the blockchain into 5 layers in Fig. 1b.

- *Data layer*. To efficiently organize various data in the blockchain, the data layer contains elements such as data structure, transaction model, index data, state data, and persistent storage scheme.
- *Network layer*. To meet the communication between nodes in a decentralized blockchain network, the P2P protocol plays an important role in the network layer.

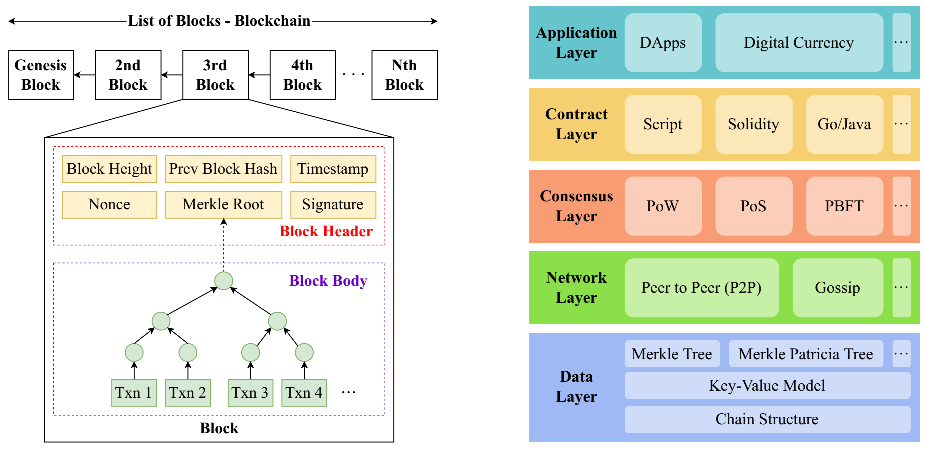

Fig. 1 Blockchain overview

(a) Chain-like structure of blocks **(b)** Architecture of a blockchain system

The content transmitted between nodes mainly consists of transaction data and block data.

- *Consensus layer*. Unlike centrally governed databases, blockchain uses a distributed consensus algorithm to ensure that nodes in the network that do not trust each other can agree on the same ledger. The use of consensus algorithms improves the blockchain's ability to cope with crash tolerance or Byzantine fault tolerance, giving the blockchain a higher level of security than traditional databases.
- *Contract layer*. Containing various scripts, algorithms and smart contracts, it is the foundation of blockchain programmability.
- *Application layer*. Users can easily develop new decentralized and cryptographically secure blockchain-based applications using the APIs provided by the blockchain.

Permissionless and Permissioned Blockchains can be broadly classified into two categories: permissionless blockchains and permissioned blockchains.

Permissionless blockchains are a type of blockchain in which anyone can participate in the network without any prior approval or authorization. It is often referred to as a public blockchain as the network is open to the public. Examples of popular permissionless blockchains include Bitcoin, Ethereum [15], etc. Ethereum abandons the UTXO transaction model proposed by Bitcoin in favor of the account/balance transaction model and extends the Merkle tree to the Merkle Patricia Trie (MPT) [16]. The Merkle roots originally stored in the block header are changed to three Merkle Patricia tree roots in Ethereum, corresponding to the world state tree, transaction tree, and receipt tree, respectively. Besides, the most important innovation of Ethereum is that it provides Turing-complete scripting languages Solidity and Serpent, and provides a sandbox environment Ethereum Virtual Machine (EVM) for users to write and run smart contracts. With programmability, built-in persistent state storage, and Turing completeness, smart contracts make it easy for developers to create their blockchain applications on the Ethereum platform, which marks the birth of the blockchain 2.0 era.

Permissioned blockchain are blockchains that require permission to join and participate in consensus. Hyperledger Fabric [17], a permissioned blockchain, has evolved into an enterprise-accessible distributed ledger technology platform. Fabric allows users to write smart contracts (also called chaincodes) in Go or Java. Therefore, it is the first platform that supports high-level programming languages for writing smart contracts. Unlike permissionless blockchains such as Bitcoin and Ethereum, Fabric adds an access mechanism where only authorized nodes can join the network and uses Raft, an efficient algorithm but cannot resist Byzantine behaviors. Fabric is innovative in that it uses a loosely coupled design that modularizes consensus algorithms, authentication, key management protocols, and cryptographic libraries, further meeting the diversity of enterprise needs for blockchain.

Innovations on Blockchain Blockchains have achieved great success and promoted many developments in different fields. However, traditional blockchain systems still suffer from problems of low throughput and high latency. There are several innovations in consensus algorithms and transaction concurrency to address these issues.

The consensus algorithm is one of the core technologies of blockchain, which describes how the peers reach an agreement on the state of the world. The efficiency of consensus algorithm impacts the performance of the entire blockchain system. Here, we introduce some BFT-based protocols. Castro et al. [18] propose PBFT consensus algorithm which reduces BFT's complexity from exponential to polynomial. To further optimize the decentralization level and performance scalability of blockchains, SBFT [19] reduces communication to linear with collectors and threshold signatures. FastBFT [20] designed a novel message aggregation technique, reducing message complexity from $O(n^2)$ to $O(n)$.

The purpose of concurrency control is to optimize transaction processing, which involves improving the efficiency of transaction validation, execution, and confirmation on blockchains. Take Hyperledger Fabric for instance, although it parallelly executes the transactions in the execution phase, the throughput cannot further improve, especially when there are high contentions among each transaction. To be more specific, though all the conflict transactions are simulated in the execution phase, only one of them can be eventually submitted in the final validation phase, and others have to be aborted. The solution is to reduce the abort rate. Fabric++ [21] uses reordering and early abortion to solve this problem. It obtains the read/write set of each transaction in the execute phase and recognizes the conflicting transactions with a dependency graph between the transactions within the same block. Then it reorders the transactions and early aborts the transactions that cannot be serialized. FabricSharp [22] optimizes the reordering mechanism to support inter-block transactions, which further promotes the commit rate and the performance in terms of throughput. FastFabric [23] extends the concurrency of Fabric by introducing a validation pipeline, which parallelizes as many validation steps as possible, including endorsement policy validation and syntactic verification. Finally, XOX Fabric [24] proposes a novel hybrid execution model consisting of a pre-order and a post-order execution step which makes a trade-off between minimal invalid transactions and maximal concurrent execution.

2.1.2 Database

Database technology has been developed for decades. Unlike blockchain, it supports features like ACID properties, complex queries, low transaction latency, high throughput, and scalability. Mainstream databases are divided into three categories: SQL databases, NoSQL databases, and NewSQL databases.

- *SQL databases*. As one of the most widely used databases supporting the relational model, SQL database (e.g., MySQL [25], Oracle [26]) is usually used to store structured data which is highly-organized and formatted. Therefore, SQL databases can comply with atomicity, consistency, isolation and persistence. In addition, SQL databases also have good support for transaction concurrency control and data privacy protection.
- *NoSQL databases*. To have better horizontal scalability, many databases abandon the relational model and support for SQL statements, replacing them with support for semi-structured and unstructured data. These databases are called NoSQL databases. Unlike relational databases, NoSQL databases have multiple types: key-value databases (e.g., LevelDB [27], BerkeleyDB [28], Redis [29]), column-oriented databases (e.g., Bigtable [30], Apache HBase [31]), document-oriented databases (e.g., MongoDB [32]), graph databases (e.g., Neo4j [33]), time series databases (e.g., InfluxDB [34]) and so on.
- *NewSQL databases*. A new type of database management system (DBMS) is designed to provide a NoSQL system's high scalability and performance while retaining the ACID transactional characteristics of a traditional relational database management system (RDBMS). NewSQL systems can use both relational and non-relational data models. The mainstream NewSQL systems include Google Cloud Spanner [35], CockroachDB [36], TiDB [37], and Amazon Aurora [38]. These NewSQL systems are all built with distributed architectures that provide high scalability and performance while retaining ACID transaction features and SQL query language support. Their emergence provides new options for addressing the needs of large-scale data processing and distributed systems.

In general, blockchains and databases are different data management technologies with different futures and application scenarios. Blockchains have the advantage of security for applications requiring security, while databases have the advantage of performance and usability for large-scale data processing and high concurrent access.

2.2 Blockchain-Database Spectrum

Though blockchains and databases are essentially designed for different goals, both systems have the capability to manage data. Along this point of view, we present our blockchain-database spectrum in Fig. 2 to compare them and find possible fusion directions.

In this framework, blockchains lie at the security end of the spectrum, while databases are at the other performance end. Besides both ends, there are also systems located in the middle parts of the blockchain-database spectrum. These systems are fusions of blockchains and databases to varying degrees and can be further classified into three major types, namely *database-oriented blockchains*, *blockchain-oriented databases*, and *hybrid systems*. As Fig. 2 depicts, the difference lies in the design considerations and trade-offs between performance and data security.

2.3 Fusion Systems

In this survey, we focus on the fusions systems, i.e., *database-oriented blockchains*, *blockchain-oriented databases*, and *hybrid systems*, which occupy the middle parts of the blockchain-database spectrum in Fig. 2. We will informally define these systems and briefly describe their design considerations in this section, while leaving the details to the rest of this survey. A high-level and coarse-grained comparison between pure blockchains, pure databases, and three types of fusion systems based on their locations in the blockchain-database spectrum is summarized in Table 1.

Database-Oriented Blockchains The *database-oriented blockchains* are at the blockchain side of the blockchain-database spectrum. Same as blockchains, *database-oriented blockchains* retain the essential chain-like structure of ledgers, which keeps track of data modifications and ensures data security. Besides the security, *database-oriented blockchains* also pursue features to provide a better experience in real-world practice just as databases do, such as easy-to-use APIs, higher throughput, lower resource consumption, and assurance of secret data's privacy. To sum up, *database-oriented blockchains* are a collection of systems that are built on top of blockchains and integrated with database features.

As it has been revealed in the spectrum, the most straightforward and widely-used solution is to equip the systems with mature techniques from databases, including sharding [7, 53–59], concurrency control [21, 49, 60–64], indexing [41,

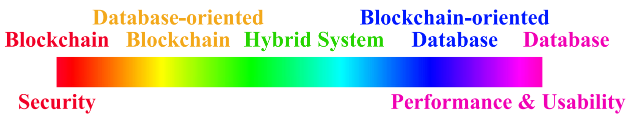

Fig. 2 Blockchain-database spectrum

Table 1 A high-level comparison of inherent features between data management technologies

System	Security			Performance and usability				
	Decentralization	Data security	Auditability	Performance	Ease of Use	Data Model	Privacy	Resource requirement
Blockchains	High	Very high	Every participant	Very low	Low	KV	No	Very high
Database-oriented Blockchains	High	Very high	Every participant	Low	Medium	KV [39, 40], Relational [41, 42], Document [43]	Low	High
Hybrid Systems	Partial	High	Every participant	Low	High	Relational [44], Document [45], Graph [46]	Low[a]	High
Blockchain-oriented databases	Partial	High	Authorized users	High	High	Relational [47–51], Column [52]	High	Normal
Databases	No	Basic	Authorized users	Very high	High	KV, Relational, Graph, Document	High	Normal

[a]No privacy for the on-chain metadata

62, 65–75], views [76, 77], and so on. There is another direction that modifies existing components in blockchains, such as consensus protocols [50, 78–81] and data processing layers [41, 82–85].

Blockchain-Oriented Databases Opposite to the *database-oriented blockchains*, the *blockchain-oriented databases* are closer to databases. Such systems pay more attention to processing performance and usually support more complicated data models such as relational. Some of them also support SQL-like interfaces, making them more convenient for application developers.

To achieve such a goal while keeping a basic security guarantee, *blockchain-oriented databases* are built upon an existing database instance, while learning lessons of hash chain from blockchains. That is, they usually contain a blockchain layer [50] or a middleware [48, 49] with blockchain features, and the chain-like relationships are either revealed by the internal fields or stored in a specific table. We regard such systems as general *blockchain-oriented databases* and introduce the technical details in Sect. 4.

We also notice that there is another way to build a database system that supports verifiable data processing, which results in the so-called ledger databases [86–89]. However, such systems adopt a different trust assumption with blockchains, i.e., there is usually a centralized service provider and require a trusted auditor to replay the log of transactions to detect if the server has tampered with the data. Hence, we exclude the ledger databases from the *blockchain-oriented databases*, and ignore them in the following of this paper.

Hybrid Systems Such systems locate around the very center of the spectrum, which means they reach a balance between security and performance. Note that this can be interpreted into two situations. The ideal one is to achieve decentralized data security as blockchains and high throughput as commercial databases at the same time. However, this is an unreachable target at present and no one has been recognized to provide a perfect solution to this problem. On the other hand, equally combining blockchains and databases into a single system is also a way to reach the balance [44–46]. This usually results in a middleware that connects a blockchain of metadata or logs, and a database of various forms of data. In this way, such systems ensure the security of metadata and the performance of data processing, which is at the cost of inheriting some defects from both sides. For example, the system may be redundant to include both instances, and the actual data stored in the database are usually not protected by the blockchain. We use *hybrid systems* to refer to the latter systems in the rest of this survey.

3 Database-Oriented Blockchains

The efforts to explore the data management possibility of blockchains have taken a long way. At the early stage of the exploration, many researchers try to adopt blockchains to real application scenarios, which leads to the earliest *database-oriented blockchains*. For example, MedRec [90] is an Ethereum-based decentralized system to process electronic medical records (EMR) and can be integrated with the existing EMR management systems. It utilized the data management ability of underlying Ethereum by proposing three dedicated smart contracts to contain metadata about the record ownership, permissions, and data integrity. Other works attempt to apply blockchain to other fields and manage corresponding data with the help of smart contracts, such as vehicles [91], cognitive radio (CRs) [92], IoT [40], cloud computing and services [93–96], decentralized privacy-preserving search [97], MOOC [98], and COVID-19 contact tracing [99].

However, the aforementioned systems just take exiguous steps toward databases in the spectrum. The successors propose prototype systems or protocols to manage general data with integrated database systems, which mainly focus on data integrity. Gaetani et al. [100] design a two-layer blockchain-based system in cloud computing environment. The first layer uses a lightweight distributed consensus protocol that ensures low latency and high throughput, and the second one is a PoW blockchain to ensure data integrity. Sui et al. [101] propose an encrypted data management system with mandatory access control, in which blockchain provides the integrity guarantee. Konashevych proposes a protocol [39] to design a cross-blockchain database that manages data on different chains and solves problems of immutability, as well as duplication of tokens as the result of hard forks.

Recently, researchers of *database-oriented blockchains* aim to equip pure blockchain systems with the ability to manage general data and reach the goal of high throughput, low resource consumption, easy-to-use APIs, and privacy of secret data. Such systems usually modify several components of blockchains, including: (1) index, (2) protocol, e.g., sharding and consensus, (3) API and data models, and (4) ledger arrangement, as Fig. 3 shows. Note that these technical routes do not necessarily correspond to the goals. They can either be combined to solve a single problem, or improve the system in various aspects individually. Moreover, many researchers try to improve multiple aspects in their single system and adopt many techniques. Thus, we review existing studies from the technical routes rather than the goals in this section.

3.1 Index

In databases, an index is a structure that sorts the specified values which aim to boost query processing and data updates. However, indexes in blockchains usually take an additional task to prove the integrity of data as an authenticated data structure (ADS) does. For example, Ethereum uses MPT to index the states of each account and protect the data. However, such an index has poor performance since it has to fetch data from LevelDB whenever it visits a node in the MPT. Thus, recent works try to develop indexes that fit the batch data in the blockchain environment, which improves the performance of the indexes of *database-oriented blockchains*. Besides the original data, researchers also index some metadata to support a broader range of queries. There are also works trying to add concurrency to the indexes that support parallel updates.

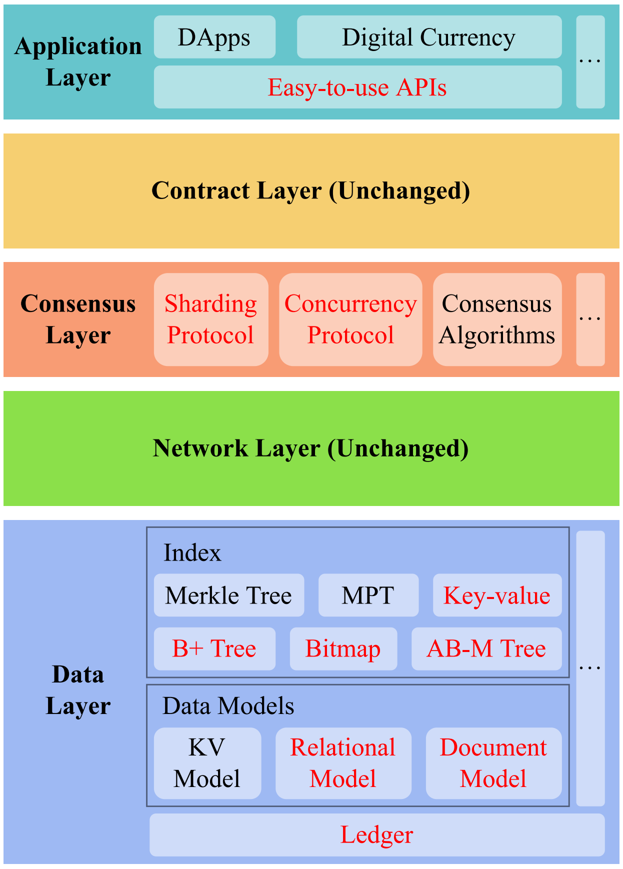

Fig. 3 Architecture of *database-oriented blockchains*

The efforts around the indexes are summarized in Table 2 and introduced in detail as follows.

3.1.1 Boosting Data Access

The authors of SEBDB [41] identified three basic operations in blockchains, namely: (1) fetching a block by a given block id, transaction id, or timestamp, (2) fetching tuples with the same transaction type, and (3) fetching transactions by given conditions. They designed a corresponding index structure for each operation to boost data access. For the first operation, a block-level B^+ tree with key (block_id, first_tx_id, ts) is built. In this way, given the query condition, one can go from the root down to the leaf node to get the location of the target block. A table-level bitmap index recording table distribution is built to solve the second scenario. Each bitmap refers to a table, and the i-th bit in a bitmap indicates whether block i contains transactions of that table or not. Layered indexes are designed to deal with the third operation, in which the first level consists of bitmaps or entries

Table 2 Summary of representative indexes in *database-oriented blockchains*. For the general notations, n refers to the number of blocks, N refers to the number of transactions, and e refers to the number of data entries

System	Index	Underpinning techniques	Index Level	Query time complexity	Space occupation	Supported query types[a]
SEBDB [41]	B^+ Tree Index	B^+ Tree	Block	$O(\log n)$	$O(n)$	SQL-like (condition), Provenance, On-chain Join, On-off Join
	Bitmap Index	Bitmap	Table	$O(1)$	$O(n)$	
	Layered Index	Bitmap, B^+ Tree	Transaction	$O(\log N)$	$O(bN)$[b]	
AuthQX [69, 70]	–	Merkle B Tree, SkipList	Entry	$O(\log e)$	$O(e)$	Range
SE-Chain [72]	AB-M tree	Balanced BST, Merkle Tree	Transaction	$O(N)$	$O(N)$	KV
Yan et al. [67]	B^+ Tree Based Index	B^+ Tree, Bitmap	Block	$O(\log n)$	$O(n)$	KV, Range
	Key-valueIndex	Key-value	Transaction	$O(1)$	$O(N)$	KV
ForkBase [73]	POS-tree	B^+ Tree, Merkle Tree	FNode	$O(\log m)$[c]	$O(m)$	KV
LineageChain [74, 75]	DASL	SkipList	State (Transaction)	$O(\log N)$	$O(N)$	KV, Provenance
vChain [65]	Intra-block Index	Merkle Tree	Transaction	$O(\log N)$	$O(N)$	Boolean, Range[f], Subscription
	Inter-block Index	SkipList	Block	$O(\log n)$	$O(n)$	
	IP-Tree	Prefix Tree, Inverted File	Grid Node	$O(l)$[d]	$O(k)$[e]	
vChain+ [68]	SWA Index	Sliding Window, Merkle Tree	Entry	$O(w)$[g]	$O(w)$	Range, Multi-dimensional, Combination
Zhu et al. [85]	GCA^2-tree	Merkle Tree	Block	$O(\log n)$	$O(n)$	Multi-dimensional Aggregation (Count, Max, Min, Average)
Feng et al. [62]	Merkle Forest	Merkle Tree	Entry	$O(t \log e)$[h]	$O(e)$	Merkle Multiproofs
Zhang et al. [102]	GEM^2-Tree	SMB Tree, Merkle B Tree	Entry	$O(\log e)$	$O(e)$	Range
Zhang et al. [103]	Suppressed $Merke^{inv}$ Index	Merkle B Tree	Entry	$O(\log e)$	$O(e)$	Keyword
	$Chameleon^{inv}$ Index	Chameleon Tree	Entry	$O(\log e)$	$O(e)$	
	$Chameleon^{inv*}$ Index	Chameleon Tree, Bloom filter	Entry	$O(\log e)$	$O(e)$	

[a]We omitted the "authenticated" or "verifiable" prefixes, since they are already guaranteed by the blockchains

[b] b is the number of buckets that describes the distribution of attribute's values among blocks

[c] m represents the number of FNodes, which is specific to the queries

[d] l represents the number of grid node layers, which is specific to the queries

[e] k represents the number of all the grid nodes, which is specific to the queries

[f] Only supports integers, fixed-point numbers, or other data types that can be transformed to set-valued attributes

[g] w is related to the user-defined sliding window size, thus the overhead is constant to a given sliding window size

[h] t is a user-defined parameter to control the layer of the Merkle Forest

that describe the distribution of attribute's values among blocks, while the second level is a B^+ tree for the attributes within the block.

AuthQX [69, 70] runs in an environment with TEEs, which provides an isolated memory for sensitive data and ensures secure computing in the hardware level. However, the limited memory of TEE hinders its wide application. To solve this problem, the authors developed a mechanism to organize data hierarchically in the untrusted and trusted memory and designed corresponding index structures. To be more specific, data in the untrusted memory are organized into Merkle B trees, while the frequently accessed internal nodes are cached in the TEE. The skip list maintained in trusted memory buffers newly attached block data. Once the capacity of the skip list reaches the threshold, the merge operation from the skip list to the MB tree will be started.

The replacement between hot and cold data follows LRU strategy. What's more, since the data cached in the TEE are authenticated, the update inside the TEE can be batched to improve the efficiency when the tree is frequently updated.

Each transaction of SE-Chain [72] is maintained in the AB-M tree (adaptive balanced Merkle tree), which combines the advantages of balanced binary trees (fast retrieval) and Merkle trees (fast verification). Specifically, an AB-M tree is divided into two layers, the lower one is a Merkle tree, and the upper one is a binary tree containing node hash information, so as to meet the verification requirements from leaf node to root node. There is a threshold T controlling the size of the Merkle tree, while the rest are arranged in the binary tree. When processing queries, the system first searches an approximate range from the top level according to the balanced binary tree search algorithm, and then traverses the Merkle tree to fetch the specified data.

Yan et al. [67] designed a dual-index to adapt the block form of data storage for their proposed construction engineering management system. Instead of traversing the entire blockchain, there is a B^+ tree-based index and a key-value index in the proposed system to accelerate the queries. The former one is arranged by the locations, and takes care of range queries. The internal nodes of the B^+ tree index contain a bitmap to indicate whether there is a file of specific types, which boosts the queries by the file types. The keys of the latter index are hashes of each file and the values are the corresponding transactions, which is responsible for the single queries.

3.1.2 Enriching Query Types

With the help of specifically designed indexes for basic blockchain operations, SEBDB [41] further supports SQL-like operations such as track-trace, on-chain join, and on-chain and off-chain join (on-off join). The roles of the proposed indexes are described as follows. First, a track-trace operation is to find who sends the transaction and which transaction is done. To support this, the layered indices on the operation sender and operation type are pre-created. Second, for the on-chain join, the table-level index can accelerate the searching process, and layered indexes on the join attribute can further optimize the performance. Third, similar to the on-chain join situation, the bitmap index and layered index on the join attribute also help the scan procedure of on-chain data.

The structures of most blockchains' index not only depend on the items stored in the index, but also on its update history. However, the authors of ForkBase [73] extracted the need for structurally-invariant reusable indexes (SIRI), whose structure is uniquely determined by the set of records. They further proposed a SIRI instance called POS-Tree. In a POS Tree, the data entries are sorted and arranged into a byte sequence. Then, different types of split functions are applied to the sequence recursively to create leaf nodes and internal nodes, which are modeled as FNodes in the POS-Tree. The FNodes are linked and protected as those in Merkle trees. In this way, the POS-Tree supports effective data deduplication of multi-version data, which enables fork semantics of blockchains that manage the conflicts.

LineageChain [74, 75] supports online forward provenance tracking, i.e., providing historical blockchain states in a tamper-evident manner to smart contracts while they are running. To achieve such a goal, LineageChain reorganizes the leaf nodes in the original Merkle tree into a Merkle DAG, to enhance the storage layer of blockchains, and provide efficient tracking and tamper evidence. Then, it indexes the Merkle DAG with a deterministic append-only skip list (DASL) to avoid searching from the head of DAG. The DASL leverages the append-only and non-random properties of blockchains to distinguish it from normal skip lists. Such a scheme enables fast and low-cost history data query, making it possible to track a specific value when a smart contract is running.

There are light nodes that only store block headers in a blockchain network, and they usually represent a user. It is important for them to verify the integrity of query results. Xu's team successively proposed systems to support authenticated queries for light nodes [65, 68]. These systems split the indexing and proving function of blockchain indexes and designed accumulator-based ADSs. In vChain [65], an accumulator-based ADS is proposed to aggregate any query attributes dynamically and support authenticated Boolean queries, while a Merkle tree-based intra-block index and a skip list-based inter-block index undertake the acceleration task. The authors also build an inverted prefix tree (IP-Tree) over subscription queries to efficiently handle a large number of subscription queries. By introducing a set accumulator-based ADS with sliding time window and building corresponding SWA index, vChain+ [68] supports authenticated queries on range, multi-dimensional, and the combination of these types. Zhu et al. proposed another accumulator-based ADS, namely GCA^2-tree [85], that supports verifiable multidimensional aggregate queries. To enrich authenticated query types in the hybrid-storage blockchain, Zhang et al. [102] propose a gas-efficient ADS, called GEM^2-tree, which supports authenticated queries. To further reduce gas cost due to storing intermediate data in GEM^2-tree and extend keyword search in hybrid-storage blockchain, Zhang et al. [103] design novel ADS schemes, such as suppressed Merkle^{inv} index, Chameleon^{inv} index, and Chameleon^{inv*} index.

3.1.3 Adding Concurrency Support

Fang et al. [62] focus on introducing concurrency to blockchains. Besides the concurrency in the transaction execution framework, they also upgrade the index to support parallel updates and validations. Specifically, they designed a Merkle Forest consisting of multiple sub-trees at a specified size of 2^N to increase the parallelism of generating multiproofs for data and verifying data integrity. It is a layered structure that the roots of the lowest level are the leaves of the upper level, and the root is computed recursively. In this way, the modification of low-level Merkle trees can be done in parallel, and the recursive updates of upper-level trees are batched to decrease the overhead. The validation of the multiproofs is conducted from bottom to top.

3.2 Protocol

In blockchains, protocols are a set of rules that allow participants to communicate and share data. Though the existing blockchain protocols ensure relatively secure communication, the full-replicated and serial nature lowers the whole system's performance, which hinders the further application of blockchains in the data management field. In this survey, we focus on two of the promising solutions, namely sharding and concurrency. In addition, the consensus algorithm is orthogonal with the two approaches and can be arbitrarily combined with them according to actual needs. We provide an overview of the surveyed works in Table 3.

3.2.1 Sharding

Sharding is originally a technique in databases to expand storage capacity and reach higher throughput. By sharding, the huge data is divided into multiple subsets and stored on different nodes, so that transactions on different nodes can be processed in parallel. Many *database-oriented blockchains* also benefit from such a method and improve the data processing capability. Elastico [104] is the first sharded blockchain that divides the network into multiple groups. Each group processes disjoint transaction data and runs PBFT consensus independently. Afterward, OmniLedger [105] introduced 2PC to the cross-shard transaction process to ensure the atomicity of cross-shard

Table 3 Summary of representative works about protocols in *database-oriented blockchains*, in which "A/B" refers to the Account/Balance model

System	Type	Threat model	Transaction model	Consensus protocol	Fault tolerance	Throughput[a]	Additional techniques
Elastico [104]	UTXO	Permissionless	BFT	PBFT	1/3 (Intra-shard) 1/4 (Total)	N/A	–
OmniLedger [105]	UTXO	Permissionless	BFT	ByzCoinX	1/3 (Intra-shard) 1/4 (Total)	13,000	2PC
RapidChain [106]	UTXO	Permissionless	BFT	PBFT+EC	1/2 (Intra-shard) 1/3 (Total)	7380	Erasure Coding
Monoxide [107]	A/B	Permissionless	BFT	Chu-ko-nu Mining	1/2	11,694.89	–
SlimChain [61]	A/B	Permissioned	CFT	Raft	N/A	1284	TEE, Serializable Snapshot Isolation, Optimistic Concurrency Control
		Permissionless	BFT	PoW	1/2	462	
BrokerChain [56]	A/B	Permissionless	BFT	PBFT	1/3 (Intra-shard) 1/4 (Total)	30,000	–
Meepo [58]	A/B	Permissioned	BFT, BFT/CFT[b]	*Any*	–	124,583.7	–
BFT-Store [53–55]	A/B	Permissionless	BFT	BFT+EC	1/r [c]	2100	Erasure coding
Section-Blockchain [57]	A/B	Permissionless	BFT	Proof of Storage	1/2	N/A	–
SChain [63]	A/B	Permissioned	BFT	PBFT	1/3	N/A	Deterministic Concurrency Control
PEPP [60]	A/B	Permissioned	BFT	PBFT	1/3	14,000	–
SEFrame [64]	A/B	Permissioned	BFT	PBFT	1/3	N/A	Optimistic Concurrency Control

[a]In tps (transactions per second), and each is the best throughput reported in the paper

[b] The inter-shard threat model is BFT, and the intra-shard one can be configured as either BFT or CFT according to the requirement

[c] r is configurable and specific to the BFT protocol it adopts

transactions; RapidChain [106], combined the PBFT protocol with erasure coding (EC) [108] to reduce the huge network traffic brought by PBFT; Monoxide [107] proposed Chu-ko-nu Mining to solve the problem that the computing power in the PoW is diluted after sharding, and enhance the security of the system. However, there are still problems to be solved, e.g., the efficiency of the cross-shard transaction process, and the storage issues.

Although the transactions within the same shards can be efficiently executed in sharded blockchains, the cross-shard transactions usually become the bottleneck. Authors of BrokerChain [56] pointed out that the cross-shard transactions can be reduced if the partition can be altered. Thus, they analyze the accounts that involve in the upcoming transactions and dynamically adjust to the optimized account distribution. However, such a scheme cannot completely avoid cross-shard transactions. To solve this problem, the accounts involved in the cross-shard transactions are virtually divided into several sub-accounts with part of the assets it holds, and distributed in different shards. Thus, cross-shard transactions can be divided into intra-shard and cross-shard sub-transactions. The former can be effectively handled, while the latter is taken care of by special broker accounts in both shards.

Meanwhile, Meepo [58] provides another solution to improve the efficiency of cross-shard transaction execution. It requires a consortium environment, i.e., every node belongs to a specific organization and each organization contains several data shards. Moreover, nodes within the same organization trust each other. Under such a scenario, the authors proposed a cross-shard protocol. To be more specific, several cross-epochs are inserted after processing intra-shard transactions to execute inter-shard transactions in the consensus. Each shard sends cross-calls that include necessary data to the remote shard and executes the inter-shard transactions on the target shard. This procedure is executed repeatedly until there are no more cross-calls generated, i.e., all cross-shard transactions have been processed. This indicates the cross-epochs of the current block have finished and the consensus of the next block goes on. In this way, cross-shard communication can be done according to the order of cross-calls, which reduces the contention of cross-shard transactions and improves efficiency. Meepo also provides atomic guarantees for cross-shard transactions in replay-epoch, which follows cross-epochs to remove any faulty transactions.

Since the data are supposed to be fully replicated in the primitive blockchain network, sharding should reduce the storage overhead of every single machine. BFT-Store [53–55] is a Byzantine fault-tolerant partition storage engine that equips Reed-Solomon (RS) [109], a widely-used EC. Specifically, assuming that there are n nodes in the network and the system can tolerate at most f faulty nodes, an RS engine is responsible to encode $n-2f$ original blocks into n chunks with $(n-2f, 2f)$-RS encoding. The choice of parameters is based on the fact that the BFT protocol can only ensure that $n-2f$ honest nodes commit blocks. Then each chunk is distributed to a node. In this way, the storage complexity of each block is reduced from $O(n)$ to $O(1)$. The read engine handles the read requests and responses with the target block. When the target block is local to the node, it is returned directly, otherwise the node sends a query request to the target node. If the request is not replied until timeout exceeds, the node broadcasts a decoding request to $n-f$ random nodes to obtain necessary chunks and returns the decoded target block. Other components include a recovery engine that recovers the data and a scale-out engine that coordinates the re-encoding process when a new node joins the system.

Section-Blockchain [57] also tries to reduce the storage overhead without compromising the security of the system via the sharding technique. It simply partitions the blocks into several blockchain fragments and the corresponding database snapshots that record the global system settings and account states at an exact moment. Then, the author designed an efficient protocol that helps participants to optimize the connection with each other to achieve a formatted network. With such a protocol, the author proved that the data are safe when the participants hold all the block headers of the mainchain, a subset of blockchain fragments and database snapshots, and a map table between fragments and snapshots. Since the map is much smaller than a whole block, Section-Blockchain reduces the storage overhead of each participant.

SlimChain [61] adopts a novel stateless scheme, in which the off-chain storage nodes store the ledger states and simulate smart contract execution, while the on-chain consensus nodes only maintain the short commitment of ledger states. Shardings in SlimChain aim to lower the overhead of the off-chain storage nodes. In particular, each off-chain node can choose to store partial or full states based on their storage capacities. The transactions are assigned to the nodes which hold the necessary data fragment. There's no need to worry about the cross-shard transactions that no node holds complete data, since the data can be authentically retrieved from other nodes with the TEE environment.

3.2.2 Concurrency

Many works have revealed that the serial execution of transactions is one of the bottlenecks that encumber the performance of blockchains, as it does not fully make use of the concurrency ability of modern multiprocessors. How to enable blockchains with concurrency to improve transaction execution efficiency is a hot topic in recent years, and the key lies in how to ensure that the results of concurrent schedules are the same in all nodes. As a typical blockchain system, Hyperledger Fabric [17] adopts a novel execute-order-validate (EOV) scheme, in which transactions are executed parallelly in the first stage while keeping serially in the latter two. Such a scheme inspires the design of succussing *database-oriented blockchains* to further improve the concurrency of the transactions. For example, as mentioned before, the stateless design of SlimChain [61] naturally supports the parallel execution of transactions in the off-chain storage nodes. To ensure the ACID property of these transactions, the authors introduce the concurrency control algorithms in the commit phase.

SChain [63] introduces concurrency to permissioned blockchain transactions from both intra- and inter-block. Since nodes from the same organization trust each other, transactions with the same block are assigned to different nodes and the results are shared in the organization. To maximize the concurrency while ensuring execution correctness, SChain pre-analyzes the potential conflicts by each transaction's read/write set and assigns conflicting transactions to the same executor. As for intra-block concurrency, SChain divides the block formation into five stages and overlaps the execution process of different blocks. In other words, it turns the original pipeline model into a transaction streaming pipeline model to make the most use of the computing resources.

A parallel execution engine, PEPP [60], is proposed for the consortium blockchain. It adopts a deterministic concurrency mechanism to obtain the predetermined serial order of parallel execution, and conducts parallel update operations on the state tree. The workflow to process a transaction includes three phases, namely ordering, execution, and finalization, and the PEEP involves the latter two. In the execution phase, a schedule layer is responsible to coordinates the parallel execution in a deterministic serial order. An ordered locking mechanism is used to eliminate the non-determinism without introducing additional network communication. In the finalization phase, the results of transactions are updated to a specially designed state tree that allows deferred commits and parallel updates. In this way, the workflow will not be blocked by the time-consuming tree update, and the performance of the entire system can be further improved.

Recently, new hardware are introduced to blockchain systems. It is also important to design suitable concurrency mechanisms for these systems. SEFrame [62, 64] proposes a concurrent execution mechanism based on SGX, an instance of TEE. As mentioned before, the SGX cannot hold the entire ledger in its memory due to hardware limitations, and the data swap between it and main memory should also be minimized because of the huge cost. SEFrame solves the problem and achieves concurrency both between nodes and within a single node. Specifically, the transactions are protected by the SGX in the execution phase, while the trusted results are replicated in the network. To enable inter-node concurrency, a batch of transactions is divided into several micro-batches and assigned to different nodes for execution. For intra-node concurrency, a pre-execution mechanism that fetches needed data in batches is proposed to minimize the burden on the SGX. After the pre-execution, the transactions of a micro-batch are executed in the SGX with a batching optimistic concurrency control (batching OCC) protocol.

3.3 Data Model

Existing blockchain platforms are far from convenient compared to traditional databases, as they lack the capability of modeling complex tasks in the real world. The cumbersome interfaces also prevent them from further use in business. To solve such a problem, many works aim to enable blockchains with complex semantics and easy-to-use APIs.

Since the relational model is widely used in business, many researchers and engineers try to implement relational semantics on *database-oriented blockchains*. SEBDB [41] adds relational data semantics into the blockchain platforms and supports SQL-like language as the data management interface. The block structure of SEBDB is re-designed to meet the requirement of relational semantics. Each transaction contains several system-level and user-defined application-level attributes, making it possible to maintain and store the schema as a regular relational table. FalconDB [84] is another system that explicitly supports the SQL data model. Unlike SEBDB, one FalconDB block body only consists of an arbitrary-sized transaction. And there are two attributes that record the validity time of the record to manage the history versions of data.

As for the cumbersome interfaces, BlockchainDB [83] exposes three straightforward key-value APIs, namely put, get, and verify, to the clients, while leaving the complicated primitive operations to a specifically designed storage layer. The storage layer undertakes the hideous works of reading and writing storage in a synchronized way, checking synchronization status, and computing write sets for further verification. EtherQL [43] provides two types of interfaces, namely API and REST to meet the different requirements of developers. Thus, application developers can directly use the encapsulated interfaces without fully understanding the low-level implementations. EthernityDB [82] integrates a lightweight database system with a MongoDB-like API into Ethereum by utilizing the smart contracts that are designed to process collections and documents in MongoDB.

SQL-Middleware [42] provides a different solution. Instead of modifying the underlying blockchain systems, it is a portable middleware that abstracts the blockchain into a SQL-based data management system. To be more specific, it maps each function of smart contracts into a table. When a smart contract is called, it records the structured data which is equivalent to inserting an item into the database.

3.4 Ledger

The ledger of a blockchain records either the account state or the operations on the data in plain text. As it has been introduced in Sect. 2.1, the ledger is distributed to all the nodes of the network. In such a scenario, the secret data of one participant are also leaked among the whole system. Thus, many works try to modify the ledger to keep the privacy of sensitive data.

The first solution is to encrypt the ledger. Adkins et al. [66] designed an end-to-end encrypted system, in which data is encoded rather than stored in plaintext directly. They proposed three types of encrypted multi-map that enable efficient query and modification (including add, update, and, especially, delete) operations. The first is a list-based encryption scheme (LSX) that makes use of an append-only data store. Each value of a label is linked by its address and only the last value is stored with the label. There is a flag indicating whether it has been deleted. Then, the structure is encrypted and stored. The other two multi-maps are arranged in two dimensions, namely the tree-based scheme (TRX) and the patch-based one (PAX). As for the TRX, the list is replaced by a binary tree in the consideration of search efficiency, while in the PAX, multiple operations are packed into a patch to further improve the efficiency.

Instead of cryptographic methods, LedgerView [76] adds access control views of traditional databases to permissioned blockchains. The permission control methods can be classified from two dimensions, namely encryption-based/hash-based, and irrevocable/revocable. For the encryption-based methods, sensitive data are encrypted and stored on-chain. For the irrevocable view, the owner first creates a unique symmetric encryption key K_i for each transaction T_i and encrypts them, denoting them as $enc(T_i, K_i)$. Then, it produces a new symmetric key K_V for the view and stores the view as a list $enc([tid_1, K_1, \dots, tid_n, K_n], K_V)$ on the blockchain. Next, the authenticated users receive the K_V from the owner and access corresponding transactions via K_V. For the revocable views, the lists of transactions and corresponding keys are stored separately. To revoke the user's access, the owner simply generates a new key for the view. However, when access is revoked under the revocable views, users may still have access to information they downloaded and stored locally, but they cannot access and download further information. The hash-based methods are similar to the encryption-based ones, the difference is that data are stored off-chain and the hashes are used to verify the integrity of retrieved data.

CAPER [77] is a novel permissioned system that supports both inter- and intra-application transactions. Instead of a linear structure, the ledger in CAPER is extended to a directed acyclic graph (DAG), in which the transactions and orders are represented as vertexes and edges, respectively. In such a scenario, each application only maintains the view from its perspective, in which the order between each transaction is determined in a linear formation, while the whole ledger can be combined by all the views virtually. As for the inter-application transactions, the hashes of data from other applications are included. In this way, public records are copied on all applications, while private records of one application can only be accessed by the application to ensure privacy.

3.5 Discussion

We make the following observations from the aforementioned representative *database-oriented blockchains*. First, techniques from traditional databases (e.g., sharding, indexing, and concurrency) benefit current *database-oriented blockchains* a lot since they have been examined and proved efficient in the past decades. It is still important to draw lessons from mature optimizing techniques. Second, there are also several *database-oriented blockchains* aim to improve those components unique to the blockchains, such as the chain-like ledger and the Byzantine resistance consensus protocol. Given the difference between blockchains and databases, such components play key roles in the functionality of *database-oriented blockchains*. Experiments show that corresponding improvements can greatly improve the performance of the system. Last, more and more *database-oriented blockchains* adopt multiple technical routes to enhance its functionality and improve performance. We can conclude that these technical routes can improve the system in various aspects from the previous part of this section. Thus, the combination of these techniques is a wise and promising way to develop further *database-oriented blockchains*.

In a word, the *database-oriented blockchains* satisfy various needs of modern data management, and the development and improvement of it are with a wide prospect.

4 Blockchain-Oriented Databases

The *blockchain-oriented databases* take off from the database end on the blockchain-database spectrum and aim to equip the efficient and easy-to-use data management system with blockchain-powered secure guarantee. They are usually extended from mature database systems and even the already-running database instances (which are also called legacy systems). The key point of designing the *blockchain-oriented databases* is to efficiently implement the algorithms and protocols of the blockchain and minimize the impact on the base system. There are two mainstream technical routes to satisfy the requirements, namely blockchain middleware, and blockchain layer. In Fig. 4, we abstract the general

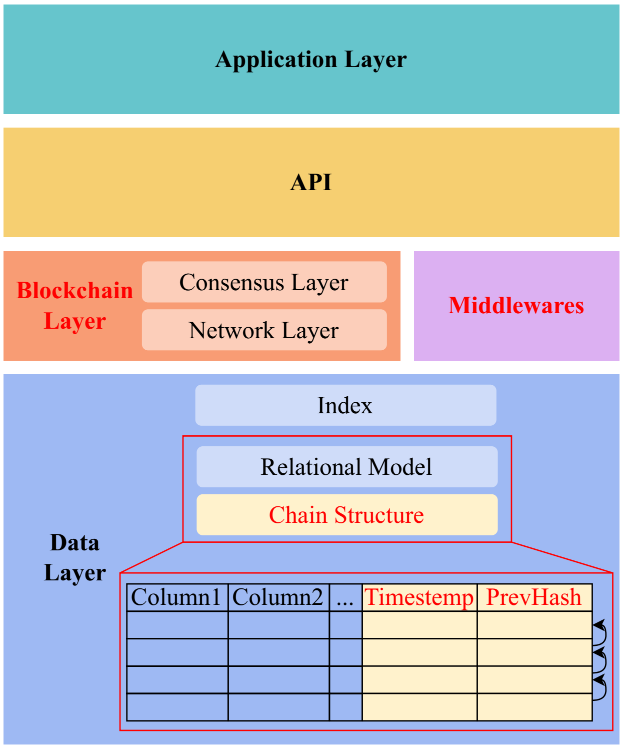

Fig. 4 Architecture of *blockchain-oriented databases*

architecture of the *blockchain-oriented databases* and highlight the mainly modified components.

4.1 Blockchain Middleware

Though it is mainstream to build a blockchain from the very beginning and add database features to it in the databases community, the attempt of leveraging the existing relational databases with rich features and transactional processing capabilities to build a blockchain is also been noticed. Nathan et al. [49] studied the feasibility of such an idea and proposed blockchain relational database. They first analyzed the similarities between blockchain requirements and database features, and then implement a blockchain on PostgreSQL, an open-sourced relational database. To be more specific, several middleware and components are designed. For example, the communication middleware is used to transfer transactions and blocks, the block processor handles the received blocks and replays the transactions in the commit phase, the built-in catalog tables store related metadata, and several shared memory data structures are used to process the transactions in different isolation levels. In this way, the traditional stand-alone databases are equipped with blockchain capabilities with the cost of only 4000 lines of C code.

Since then, more and more researchers develop various blockchain middleware to enable databases with security guarantees in different aspects. Lian et al leverage the immutability of blockchain ledgers to develop a tamper-proof detection middleware for relational databases, named TRDB [48]. In TRDB, the original data are stored and processed in the relational databases, while the hash digest of each entry is replicated among the blockchain for tamper detection. There are two additional considerations, namely data privacy and supporting relational semantics in blockchains. For the first one, AES is used to encrypt the original data to protect them from the transparency nature of blockchains. To solve the second problem, TRDB concatenates both rows and columns of each table to detect illegal insertion, deletion, and modification. What's more, TRDB caches the encrypted data with LRU strategy before it has been logged on chain to improve the data access efficiency. Thus, when querying the data, TRDB first intercepts the SQL request and executes it in the relational database, then it accesses the cache or the blockchain and subsequently compares the query results with the calculated digest. When a tamper is detected, TRDB warns of the misconduct behavior and returns the information to the user. Similar works include [110] and [111]. The former directly stores the raw data on the blockchain, while the latter concentrates on image data.

Beirami et al. [51] propose several additional built-in attributes for each relational table to simulate block headers in blockchains that support verifiable immutable transactions. The attributes include the transaction timestamp (i), a new table signature (sig_i), the previous table signature (sig_i'), the user public key($pubkey$), and a bit flag to indicate if the transaction is a deletion (del). The two signatures serve as the hash pointer of blockchain, while others are similar to the corresponding fields in block headers. Note that these attributes are calculated and inserted into the augmented tuple implicitly whenever the table is modified, which means they are transparent to the users. Given a transaction T, the system compares whether $enc(sig_t, pubkey) = hash(T, sig_{t-1})$ holds to verify its validity.

4.2 Blockchain Layer

Different from the simple additions of the middleware solutions, a blockchain layer means stepping inside the underlying databases and modifying the existing components. Though it may require more effort, such a solution allows researchers to adjust the inner workflow and improve the performance of the whole system.

Blockchain PG [47] adds the blockchain function to databases to ensure data integrity, and achieve the traceability of data through trace query. It is a permissioned system that requires a CA to prove the identity of clients. The core component of the system is PostgreSQL+ (derived from PostgreSQL), whose "blockchain layer" can be further divided into four sub-layers. The user layer provides interfaces to the clients, and verifies the user's identity with the public key; the query layer generates an optimal plan for authenticated queries to boost the query processing; the index layer provides authenticated indexes on data stored in the source layer, in which data are stored in an append-only behavior.

BigchainDB [112] is a commercial database-style distributed storage system that combines the key benefits of both distributed databases and blockchains. In fact, it is built on top of two existing RethinkDB [113] instances, namely `S` (stores an unordered set of transactions and serves as a backlog) and `C` (stores ordered list of blocks that forms a blockchain), and directly inherits the strength from databases. To enable BigchainDB with decentralized control and immutability, the authors built a blockchain layer that connects `S` and `C` with BigchainDB Consensus Algorithm (BCA) that is in charge of transaction assigning and voting.

HBasechainDB [52] adopts the same philosophy, yet it is built on Apache HBase [31], a column-based distributed database for big data. The workflow of transaction processing is quite similar to traditional blockchains such as Bitcoin – a transaction is first submitted to a transaction pool of a specific node, then the node verifies the transactions, packs all valid transactions in a block, and broadcasts to other nodes. When the majority votes for a block, it is considered to be valid and appended to the ledger, while the transactions in the invalid blocks are reassigned to other nodes randomly until it is included in the chain or removed from the system. Note that there is no need to worry about the competition between the miners and the resulting forks in practice, since HBase is of strong consistency, and the blocks are voted in the order of their timestamp. Due to the different data model, the authors designed 6 HBase tables, namely `backlog`, `block`, `hbasechaindb`, `toVote`, `vote`, and `reference`, in which `backLog` serves as the transaction pool and `reference` is an index table the maps transaction id and the content.

chainifyDB [50, 81] is a permissioned blockchain-like system built on heterogeneous database systems. The authors pointed out that blockchains and traditional DBMSs share considerable parts of their processing stack and "chainfied" existing databases by introducing a blockchain layer. To unify the heterogeneous underlying databases, a new transaction processing model called Whatever-Voting (WV) is proposed. It consists of two phases and only focuses on the results. In particular, each underlying database does whatever necessary to process the transactions and produces a digest of its behavior in the Whatever-phase, and each participant votes for the digest of W-phase in the Voting-phase. Only when an agreement is reached, the state changes are committed to a ledger by the individual organizations. chainifyDB instantiates the WV model by batching the proposed transactions and ordering them within a block. Then, the organizations use the WV model to reach a consensus on each block. For now, chainifyDB supports MySQL and PostgreSQL.

4.3 Discussion

In this section, we review representative works about the *blockchain-oriented databases* and identify two mainstream technical routes to implement it. We can conclude from the analysis that the hash-chain feature of blockchains, along with the multi-node consensus and backup mechanism, becomes important reinforcement of traditional databases' data integrity protection measures. The integration of blockchain features helps databases to further complete their functionality.

However, we also notice that the two technical routes have pros and cons. There exists a trade-off between flexibility and performance. In particular, building a blockchain middleware is easy and less intrusive. It can also bridge heterogeneous database instances with the same data model, which provides better portability and suits the inter-organization collaboration scenario. On the other hand, designing a blockchain layer for a specific database instance makes it possible to further optimize the components and provides higher performance. To sum up, the blockchain middleware is more friendly to the legacy systems, and the blockchain layer is more efficient. We further compare the works in Table 4.

5 Hybrid Systems

The *hybrid systems* locate in the center of the blockchain-database spectrum. Different from the other fusion systems which focus on either security or performance, they are equal combinations of blockchain and database, and reach a balance between the two aspects. Though it is indeed that the *hybrid systems* are less competitive than the other fusion systems in most scenarios, the balanced and comprehensive functionality enables them to cope with the basic secure data management tasks and focus on more complex requirements. In fact, many *hybrid systems* are designed to handle complicated problems in practical scenarios, such as graph data and the conflict situation between morals and laws.

The integration of the *hybrid systems* usually relies on middleware to connect the existing blockchain and database instances, and we present the abstract architecture of *hybrid systems* in Fig. 5.

5.1 Representative Systems

Instead of representing the relationship in abstract attributes, graph databases directly store and process the relationship

Table 4 Summary of representative *blockchain-oriented databases*

Category	System	Underlying databases	Performance[a]	Features and functionalities
Blockchain middleware	Blockchain relational Database [49]	PostgreSQL[b]	2500 tps (Throughput)	Transform with a trivial amount of code
	TRDB [48]	MySQL [b]	92.04% (Change in middleware execution efficiency compared to baseline)	Support tamper-proof detection
	Beirami et al. [51]	PostgreSQL[b]	50x (Improvement of throughput)	Support verifiable immutable transaction process
Blockchain layer	Blockchain PG [47]	PostgreSQL[b]	N/A	Transform from legacy systems
	BigchainDB [112]	RethinkDB	1,000,000 tps (Throughput)	Commercial system with high performance
	HBasechainDB [52]	HBase	5893 tps (Throughput)	Suitable for big data scenarios
	ChainifyDB [50, 81]	MySQL, PostgreSQL [b]	5000 tps (Throughput)	Support heterogeneous database systems

[a]We present the best result that is reported in the paper

[b]The authors claim that their systems are suitable for any relational database, here we present the database they used in the experiments

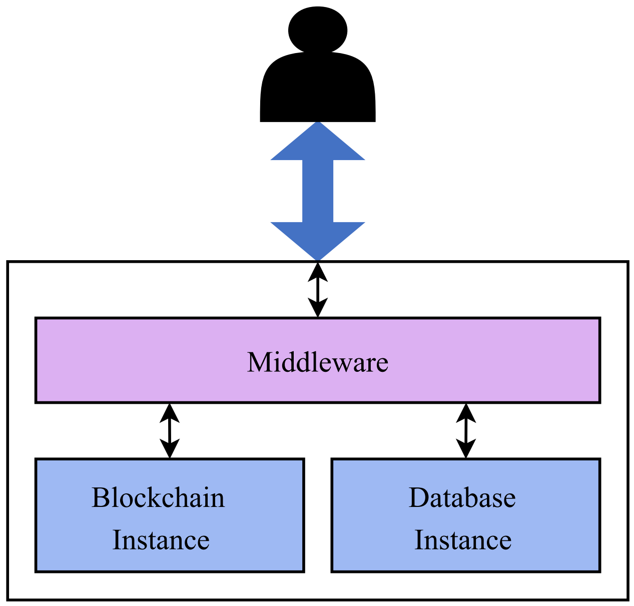

Fig. 5 Architecture of *hybrid systems*

of entities in the formation of vertexes and edges. The native graph storage and processing enable the graph databases with superior traversal performance; however, the plain KV model limits blockchains to processing such complicated data as the graph databases do. To enable the verifiable audit trail of data integrity and its modifications for information stored in a graph database, Ermolaev et al. [46] combine an Exonum [114] blockchain and Neo4j [33], the most popular graph database management system, into a single system. To be more specific, each Neo4j instance handles the data store and management, while the Exonum undertakes the role of verifiable operation log. Unlike blockchains that maintain data on the chain, the authors adopt a two-step solution that the blockchain first reaches an agreement on the modifications of graph data, and then each Neo4j instance executes the operations locally. In this way, the computation complexity and the durability of the consensus process can be greatly lowered.

In the situation of personal data management such as student data and medical records, there exists a conflict between personal privacy and public interests, i.e., stakeholders want to claim the ownership of their personal data and restrict third parties to access their data, while such a restriction may hinder the third parties to make use of these data in governance or innovation. Though blockchain and smart contract seems to be a promising solution to this problem, it is not practical since current blockchains cannot store and process such a massive amount of data. Bertram et al. [45] combine blockchain and databases to enable users to control the ownership of their private data. The system consists of three components. The core is a distributed database (e.g., BigchainDB [112]) that stores user data except identifying information. There is also a centralized MongoDB that stores user identity information. Finally, there is also a blockchain that links the two components. In particular, individual-specific smart contracts manage the map between the user identifier and his personal data. When a user decides to revoke access, he simply updates his identity in the centralized database but leaves the smart contract unchanged. Then, all third parties will lose track of this data entry while making use of the de-identified data.

The *hybrid systems* are also used to simultaneously manage data in blockchain and database platforms. This is from the observation that each platform has its solid advantage at the current stage. Thus, the most practical way is to build a combination system such that inherits both the resistance to data modification from blockchain and the query speed from the distributed databases. ChainSQL [115] is an implementation of the above scheme that ensures data integrity and fast query processing. It provides APIs that support operations in SQL and JSON format. In ChainSQL, the blockchain reaches the consensus of each transaction and stores the operations. After that, the transactions are forwarded to and executed in the database, and the actual data are also stored in the database. What's more, ChainSQL can also serve as a disaster recovery backup since the operations are logged on a trustworthy platform, i.e., the blockchain.

The authors of MOON [44] hold another view of simultaneous data management. They believe that neither of the platforms suits all types of data, thus they aim to partition the data to either blockchain or database, and expose unified interfaces to developers and final users. In general, MOON intercepts users' requests and redirects them to the platform that holds the data. When it comes to the situation that needs to process data from both platforms, MOON will retrieve data from the blockchain to a temporary table of the database and execute corresponding operations there to make full use of the mature data processing capability of databases. The authors also conduct a case study on clinical laboratory tests to find which platform is more suitable to hold different types of data, i.e., how to partition the data entries. They suggest that the data needed to change frequently should be stored in the database, while those persistent are more suitable to the blockchain.

5.2 Discussion

Table 5 summarizes the representative *hybrid systems*. Although there have been numerous studies and applications of other fusion systems, we can draw from the above analysis the unique value of *hybrid systems*. By directly integrating blockchain and database instances, the *hybrid systems* acquire a balanced and sufficient ability of integrity and fast data processing from both systems, which can satisfy the needs of most application scenarios. With such a solid foundation, we can further explore some complicated problems in secure data processing (e.g., data ownership management), which may be the most important application scenario of *hybrid systems*.

We also observe that the direct integration of several instances results in a bloated and redundant system that requires more resources than the *database-oriented blockchains* or *blockchain-oriented databases*. For example, the manipulation logs on the actual data are stored both as on-chain data on the blockchain instance and as WAL logs in the database instance. How to minimize the redundancy and make full use of the instances is a promising direction for *hybrid systems* in the future.

6 Discussion

6.1 Comparison

Based on the above analysis, we compare the three fusion systems within different dimensions in Table 6. We can draw the conclusion that each of the fusion systems has its unique advantages and suitable scenarios. In particular, *blockchain-oriented databases* suit the scenarios that value security most and want to improve the data processing capability, while *blockchain-oriented databases* are the best choice to satisfy the security needs of the efficiency-first applications. *Hybrid systems* provide a balance between data management capabilities and blockchain benefits, making them a viable option for many use cases. Thus, the three systems are of equal importance, since the three systems satisfy different urgent demands in the data management field.

We also observe the increasing trend in the research of *database-oriented blockchain*, which shows that people pay more attention to data security. Therefore, we suggest further studies of this aspect and expect a more competitive system based on the massive views in this field.

Table 5 Summary of representative *hybrid systems*

System	Blockchain	Database	Research field
Ermolaev et al. [46]	Exonum	Neo4j	Graph Data
Bertram et al. [45]	Etherum	BigchainDB[a](User Data), MongoDB (User Identity)	Data Ownership Management
ChainSQL [115]	Ripple	*Any*	Data Integration of Blockchain and Database
MOON [44]	BigchainDB[a]	PostgreSQL	Data Partition between Blockchain and Database

[a]The inconsistency comes from the underlying nature of fusion systems, depending on the authors' focus on its functionality

Table 6 A high-level comparison between three fusion systems

	Database-oriented Blockchains	Hybrid Systems	Blockchain-oriented Databases
Applicable Scenarios	Guarantee data security while Improving data processing capability	Acquire balanced properties from both sides and reduce modifications to legacy systems	Retain the data processing capability while enhancing security
Representative Systems	SEBDB [41], ForkBase [73], SlimChain [61], SE-Chain [72], BrokerChain [56]	Ermolev et al. [46], MOON [44]	TRDB [48], BigchainDB [112]
Specific Techniques	Index, Sharding, Concurrency, Data Model, Ledger	Customizable Middleware	Cryptography
Advantages	Decentralized, Data Security, Tamper-proof, Auditable	Low Coupling, Balanced Security-Performance Guarantee	High Performance, Easy-to-use, Privacy-friendly, Resource Efficient

6.2 Challenges and Future Works

Numerous cases have shown that there are strengths and weaknesses of blockchains and databases in the data management field, thus the integration of both systems to better undertake the task has become a promising solution in the database community. However, it is not an easy way. In this section, we present our observation on the research challenges and future opportunities in the integration of blockchains and databases, i.e., the fusion systems, from the aspects of performance, privacy, data description ability, new hardware, learning-based optimization, and application.

6.2.1 Performance

Performance is the most important feature of a data management system that is perceived by the users. Consequently, it is one of the most critical indicators to evaluate such a system. However, we have observed that there is still a huge performance gap between the fusion systems and mature commercial databases. This is due to the linear nature, one of the fundamental features of blockchain, hindering the transaction processing rate of blockchains and the succeeding fusion systems. To solve the problem, there are two parallel-but-associated targets, which affect two main operations (query and modification) in a data management system, respectively.

One is to build efficient indexes on the target data to accelerate data access. It is relatively easy for the off-chain part since the indexes of databases can achieve a satisfying performance. However, for the on-chain data, it is proved in practice that the design of new block data storage structure and corresponding indexes can effectively improve the query function and query performance. The index of on-chain data can either serve for real-time transaction verification [41, 67, 72] or improve the access efficiency of transaction history information [73–75]. In a word, a proper index of on-chain data provides not only higher real-time access efficiency in terms of performance, but also supports the traceability query based on historical data in terms of functionality.

The other is to improve the consensus mechanism, which is the key to ensuring the consistency of transaction execution among the participants, thus it has a great impact on the overall performance and application of blockchains and fusion systems. It is important to reach a balance between efficiency and consistency, but there are two main drawbacks and opportunities of current blockchain consensus mechanisms. First, the serial execution of transactions does not fully make use of the concurrency ability of modern multiprocessors, so it is a good idea to improve the concurrency of transactions [60, 63, 64]. What's more, the abort rate in a high contention environment also encumbers the ability to process transactions. Therefore, how to effectively eliminate the conflicts remains to be researched [21, 22].

6.2.2 Privacy

In the blockchain environment, all data and transactions need to be replicated to all nodes to obtain consensus. As a consequence, sensitive data may be accessed by unauthorized third parties, and cannot be managed in the blockchain environment. The fusion systems also face such a problem. The key lies in the access control of private data. However, directly applying access control methods for databases to the blockchain will result in the hash value of each block cannot correspond to the data obtained, so users cannot verify whether the data in the chain has been tampered with. This is a difficulty that remains to be solved.

Fortunately, with the help of database techniques, several promising solutions can be further studied. For example, the system can store sensitive data in the databases (off-chain) which have better support for data privacy [41], or learn lessons from database view to manage on-chain data with access control [76]. Other technique routes include cross-chain and cryptographic protection.

6.2.3 Data Description Ability

The development of the Internet applications has spawned a variety of data forms, such as graph data and document data. Many existing studies in blockchain and database fusion systems support and extend key-value model [39, 40] and relational model [41, 42, 44, 47–51]. However, supports for the new forms of data are not common in fusion systems, and the data description capability in blockchain fusion systems needs to be improved.

For example, graph databases are a rapidly evolving field with many active research directions. Graph mining and analysis involves extracting useful information and insights from large-scale graph data. By integrating graph databases with blockchains, transactional relationships can be analyzed in a secure and decentralized manner [116].

6.2.4 New Hardware

Recently, there is a notable development of various types of hardware related to blockchains. For example, the success of Bitcoin has led to the emergence of dedicated hardware such as Field Programmable Gate Array (FPGA) and GPU, which has greatly increased the efficiency of hash computing. In turn, how to make full use of these emerging hardware in the fusion systems to better manage data is an interesting topic for the database community. Here we present several observations.

A trusted execution environment (TEE) provides an isolated memory that resists outside corruption and ensures secure computing at the hardware level, which lowers the security assumption to a certain extent. Thus, there is an opportunity to improve other aspects of blockchains, especially in the terms of performance [62, 64, 69, 70]. GPU supports parallel computing of large amounts of data, and it is a promising idea to utilize its parallel processing capability in boosting the data processing of fusion systems [117].

6.2.5 Learning-Based Optimization

Machine learning has been extensively studied over the past decades. It simulates human learning behaviors with high computing power to acquire new knowledge or skills, and has been widely applicated in database optimizations such as cost estimation, join order selection, and end-to-end optimizer [118]. We believe that it will also gain huge success in the optimization of blockchain-database fusion systems.

For example, the data distribution in sharding blockchains can greatly affect the efficiency of data access. However, current sharding systems usually adopt a naive rule such as prefix/suffix-based, which may not suit the real data distribution. In this way, machine learning-based rules can capture the pattern and boost data access [119]. Other applications include misbehavior detection of nodes [120, 121] and vulnerability analysis of smart contracts [122, 123].

6.2.6 Domain-Specific Application

The collectively maintained and tamper-resistant public ledger of blockchain systems ensures the security and reliability of the data stored in a distributed network. In addition to general-purposed data management, blockchain-database fusion systems can also bring new solutions to many specific domains. We notice that more and more people combine their original business systems with blockchains to form a domain-specific fusion system in various fields. There is a trend that leverages blockchain characteristics to solve the drawbacks of the business system, and improve the shortcomings and limitations of the blockchain system itself. However, applications in various fields have also posed more challenges.

Take finance as an example. The processing capacity of the blockchains is not enough to replace the existing centralized trading system. Therefore, it is important to improve the consensus mechanism to adapt to high throughput financial transaction applications. As for the supply chain, it is necessary to equip the system with a traceability model that fits the industrial supply chain scenario to promote verifiable data sharing in supply chain management. Other applications such as intellectual property management, asset delivery, and medical data management, also have different requirements for the fusion system.

7 Conclusion

In this survey, we present the integrating trend of blockchains and traditional databases, and propose a blockchain-database spectrum to analyze the work related to the fusion systems in the field of data management. First, we classify the fusion systems into *database-oriented blockchains*, *blockchain-oriented databases*, and *hybrid systems*, and present a high-level comparison according to the different directions of their integration. Then, we review the representative fusion systems of *database-oriented blockchains*, *blockchain-oriented databases*, and *hybrid systems*. To be more specific, we review representative systems of *database-oriented blockchains* from index, protocol, data model, and ledger; we analyze blockchain middleware and blockchain layer scheme of *blockchain-oriented databases*; we also demonstrate the combination approaches and oriented research fields of different *hybrid systems*. Finally, we present a high-level comparison between the three fusion systems and our observations on the challenges and future work.

We believe that this survey demonstrates the current status and limitations of existing blockchain-related data management research and provides insight for researchers to conduct in-depth research in this area.

Acknowledgements Not applicable.

Author Contributions C.Z.: Conceptualization, Methodology, Literature search, Data analysis, Investigation, Writing - original draft; J.L.: Literature search, Data analysis, Investigation, Writing - original draft, Visualization; Z.Z.: Data analysis, Writing - review & editing; C.Y.: Writing - review & editing; M.Z.: Writing - review & editing, Funding acquisition, Supervision.

Funding This work is supported by National Natural Science Foundation of China (62072033).

Availability of Data and Materials Not applicable.

Declarations

Conflict of interest We declare that authors have no known competing interests or personal relationships that might be perceived to influence the discussion reported in this paper.

Ethics Approval Not applicable.

Consent to Participate Not applicable.

Consent for Publication Not applicable.

Code Availability Not applicable.

References

1. Nakamoto S (2008) Bitcoin: a peer-to-peer electronic cash system. Decentralized Business Review, 21260
2. Croman K, Decker C, Eyal I, Gencer AE, Juels A, Kosba A, Miller A, Saxena P, Shi E, Sirer EG, Song D, Wattenhofer R (2016) On scaling decentralized blockchains (a position paper), vol. 9604 LNCS. Christ Church, Barbados, pp 106–125. https://doi.org/10.1007/978-3-662-53357-4_8
3. Hong A, Sun C, Chen M (2020) A survey of distributed database systems based on blockchain. In: 2020 3rd International Conference on Smart BlockChain (SmartBlock). IEEE, Zhengzhou, China, pp 191–196. https://doi.org/10.1109/SmartBlock52591.2020.00042
4. Przytarski D, Stach C, Gritti C, Mitschang B (2021) Query processing in blockchain systems: current state and future challenges. Future Internet 14(1):1. https://doi.org/10.3390/fi14010001
5. Zhang J, Zhong S, Wang T, Chao H-C, Wang J (2020) Blockchain-based systems and applications: a survey. J Internet Technol 21(1):1–14. https://doi.org/10.3966/160792642020012101001
6. Wang Q-g, He P, Nie T-z, Shen D-r, Yu G (2018) Survey of data storage and query techniques in blockchain systems. Comput Sci 45(12):7. https://doi.org/10.11896/j.issn.1002-137X.2018.12.002
7. Zhang C-g, Zhang Y-f, Li X-h, Nie T-z, Yu G (2020) Survey of new blockchain techniques: Dag based blockchain and sharding based blockchain. Comput Sci 47(10):8. https://doi.org/10.11896/jsjkx.191000057
8. Qian W, Jin C, Shao Q, Zhou A (2018) Blockchain and sharing database. Big Data Res 4(1):10. https://doi.org/10.11959/j.issn.2096-0271.2018004
9. Huang HW, Kong W, Peng XW, Zheng ZB (2022) Survey on blockchain sharding technology. Comput Eng 48(6):10. https://doi.org/10.19678/j.issn.1000-3428.0063887
10. Yu G, Wang X, Yu K, Ni W, Zhang JA, Liu RP (2020) Survey: Sharding in blockchains. IEEE Access 8:14155–14181. https://doi.org/10.1109/ACCESS.2020.2965147
11. Yu G, Nie T-Z, Li X-H, Zhang Y-F, Shen D-R, Bao Y-B (2021) The challenge and prospect of distributed data management techniques in blockchain systems. Comput Sci 044(001):28–53. https://doi.org/10.11897/SP.J.1016.2021.00028
12. Zhang Z-W, Wang G-R, Xu J-L, Du X-Y (2020) Survey on data management in blockchain systems. Ruan Jian Xue Bao/J Softw 9:23. https://doi.org/10.13328/j.cnki.jos.006091
13. Ruan P, Dinh TTA, Loghin D, Zhang M, Chen G, Lin Q, Ooi BC (2021) Blockchains vs. distributed databases: Dichotomy and fusion. In: Proceedings of the 2021 international conference on management of data. SIGMOD '21. Association for Computing Machinery, New York, NY, USA, , pp 1504–1517. https://doi.org/10.1145/3448016.3452789
14. Ge Z, Loghin D, Ooi BC, Ruan P, Wang T (2022) Hybrid blockchain database systems: design and performance. Proc VLDB Endow 15(5):1092–1104. https://doi.org/10.14778/3510397.3510406
15. Buterin V et al (2014) A next-generation smart contract and decentralized application platform. White Paper 3(37):2-1
16. Buchmann J, Dahmen E, Schneider M (2008) Merkle tree traversal revisited. In: Post-quantum cryptography. Springer, Berlin, pp 63–78. https://doi.org/10.1007/978-3-540-88403-3_5
17. Androulaki E, Barger A, Bortnikov V, Cachin C, Christidis K, Caro AD, Enyeart D, Ferris C, Laventman G, Manevich Y, Muralidharan S, Murthy C, Nguyen B, Sethi M, Singh G, Smith K, Sorniotti A, Stathakopoulou C, Vukolic M, Cocco SW, Yellick J (2018) Hyperledger fabric: a distributed operating system for permissioned blockchains. In: Proceedings of the thirteenth EuroSys conference, EuroSys 2018, Porto, Portugal, April 23–26, 2018, pp. 30–13015. ACM, New York, NY, USA . https://doi.org/10.1145/3190508.3190538
18. Castro M, Liskov B et al (1999) Practical byzantine fault tolerance. In: OsDI, 99:173–186

19. Golan Gueta G, Abraham I, Grossman S, Malkhi D, Pinkas B, Reiter M, Seredinschi D-A, Tamir O, Tomescu A (2019) SBFT: A Scalable and Decentralized Trust Infrastructure. In: 2019 49th Annual IEEE/IFIP international conference on dependable systems and networks (DSN). IEEE, Portland, OR, USA, pp 568–580. https://doi.org/10.1109/DSN.2019.00063
20. Liu J, Li W, Karame GO, Asokan N (2019) Scalable byzantine consensus via hardware-assisted secret sharing. IEEE Trans Comput 68(1):139–151. https://doi.org/10.1109/TC.2018.2860009
21. Sharma A, Schuhknecht FM, Agrawal D, Dittrich J (2019) Blurring the Lines between Blockchains and Database Systems: The Case of Hyperledger Fabric. In: Proceedings of the 2019 international conference on management of data. ACM, Amsterdam, pp 105–122. https://doi.org/10.1145/3299869.3319883
22. Ruan P, Loghin D, Ta Q-T, Zhang M, Chen G, Ooi BC (2020) A transactional perspective on execute-order-validate blockchains. In: Proceedings of the 2020 ACM SIGMOD international conference on management of data. Association for Computing Machinery, New York, NY, USA, pp 543–557. https://doi.org/10.1145/3318464.3389693
23. Gorenflo C, Lee S, Golab L, Keshav S (2019) FastFabric: scaling hyperledger fabric to 20,000 transactions per second. In: 2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC). IEEE, Seoul, Korea (South), pp 455–463. https://doi.org/10.1109/BLOC.2019.8751452
24. Gorenflo C, Golab L, Keshav S (2020) XOX fabric: a hybrid approach to blockchain transaction execution. In: 2020 IEEE international conference on blockchain and cryptocurrency (ICBC). IEEE, Toronto, ON, Canada, pp 1–9. https://doi.org/10.1109/ICBC48266.2020.9169478
25. Matallah H, Belalem G, Bouamrane K (2021) Comparative study between the mysql relational database and the mongodb nosql database. Int J Softw Sci Comput Intell 13(3):38–63. https://doi.org/10.4018/IJSSCI.2021070104
26. Almeida F, Silva P, Araújo F (2019) Performance analysis and optimization techniques for oracle relational databases. Cybern Inf Technol 19(2):117–132. https://doi.org/10.2478/cait-2019-0019
27. Podgorelec B, Turkanović M, Šestak M (2020) A brief review of database solutions used within blockchain platforms. In: Prieto J, Pinto A, Das AK, Ferretti S (eds) Blockchain and Applications. Springer, Cham, pp 121–130. https://doi.org/10.1007/978-3-030-52535-4_13
28. Olson MA, Bostic K, Seltzer MI (1999) Berkeley DB. In: Proceedings of the FREENIX Track: 1999 USENIX annual technical conference, June 6–11, 1999, Monterey, California, USA, pp 183–191
29. Liu Q, Yuan H (2019) A high performance memory key-value database based on redis. J Comput 14(3):170–183. https://doi.org/10.17706/jcp.14.3.170-183
30. Chang F, Dean J, Ghemawat S, Hsieh WC, Wallach DA, Burrows M, Chandra T, Fikes A, Gruber RE (2008) Bigtable: a distributed storage system for structured data. ACM Trans Comput Syst 26(2):4–1426. https://doi.org/10.1145/1365815.1365816
31. Vora MN (2011) Hadoop-hbase for large-scale data, vol. 1. Harbin, China, pp 601–605. https://doi.org/10.1109/ICCSNT.2011.6182030
32. Chickerur S, Goudar A, Kinnerkar A (2015) Comparison of relational database with document-oriented database (mongodb) for big data applications. In: 2015 8th International conference on advanced software engineering and its applications (ASEA), pp 41–47. https://doi.org/10.1109/ASEA.2015.19
33. Guia J, Soares VG, Bernardino J (2017) Graph databases: Neo4j analysis, vol. 1. Porto, Portugal, pp 351–356 . https://doi.org/10.5220/0006356003510356
34. Ahmad K, Ansari M (2017) Hands-On InfluxDB, pp 341–354. https://doi.org/10.1201/9781315155579
35. Bacon DF, Bales N, Bruno N, Cooper BF, Dickinson A, Fikes A, Fraser C, Gubarev A, Joshi M, Kogan E, Lloyd A, Melnik S, Rao R, Shue D, Taylor C, van der Holst M, Woodford D (2017) Spanner: Becoming a sql system. In: Proceedings of the 2017 ACM international conference on management of data. Association for Computing Machinery, New York, NY, USA, pp 331–343. https://doi.org/10.1145/3035918.3056103
36. Taft R, Sharif I, Matei A, VanBenschoten N, Lewis J, Grieger T, Niemi K, Woods A, Birzin A, Poss R, Bardea P, Ranade A, Darnell B, Gruneir B, Jaffray J, Zhang L, Mattis P (2020) Cockroachdb: The resilient geo-distributed sql database. In: Proceedings of the 2020 ACM SIGMOD international conference on management of data, pp 1493–1509. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3318464.3386134
37. Huang D, Liu Q, Cui Q, Fang Z, Ma X, Xu F, Shen L, Tang L, Zhou Y, Huang M, Wei W, Liu C, Zhang J, Li J, Wu X, Song L, Sun R, Yu S, Zhao L, Cameron N, Pei L, Tang X (2020) Tidb: a raft-based htap database. Proc VLDB Endow 13(12):3072–3084 https://doi.org/10.14778/3415478.3415535
38. Verbitski A, Gupta A, Saha D, Brahmadesam M, Gupta K, Mittal R, Krishnamurthy S, Maurice S, Kharatishvili T, Bao X (2017) Amazon aurora: design considerations for high throughput cloud-native relational databases. In: Proceedings of the 2017 ACM international conference on management of data. Association for Computing Machinery, New York, NY, USA, pp 1041–1052. https://doi.org/10.1145/3035918.3056101
39. Konashevych O (2019) Cross-blockchain databases for governments: the technology for public registries and smart laws. SSRN Electron J. https://doi.org/10.2139/ssrn.3537258
40. Tseng L, Yao X, Otoum S, Aloqaily M, Jararweh Y (2020) Blockchain-based database in an IoT environment: challenges, opportunities, and analysis. Clust Comput 23(3):2151–2165. https://doi.org/10.1007/s10586-020-03138-7
41. Zhu Y, Zhang Z, Jin C, Zhou A, Yan Y (2019) SEBDB: semantics empowered BlockChain DataBase. In: 2019 IEEE 35th International conference on data engineering (ICDE). IEEE, Macao, Macao, pp 1820–1831. https://doi.org/10.1109/ICDE.2019.00198
42. Tong X, Tang H, Jiang N, Fan W, Gao Y, Deng S, Zhang Z, Jin C, Yang Y, Qin G (2021) SQL-middleware: enabling the blockchain with SQL. In: Jensen CS, Lim E-P, Yang D-N, Lee W-C, Tseng VS, Kalogeraki V, Huang J-W, Shen C-Y (eds) Database systems for advanced applications vol. 12683. Springer, Cham, , pp 622–626. https://doi.org/10.1007/978-3-030-73200-4_48
43. Li Y, Zheng K, Yan Y, Liu Q, Zhou X (2017) EtherQL: a query layer for blockchain system. In: Candan S, Chen L, Pedersen TB, Chang L, Hua W (eds) Database systems for advanced applications vol 10178. Springer, Cham, pp 556–567. https://doi.org/10.1007/978-3-319-55699-4_34
44. Carlos Marinho SS, Filho JSC, Moreira LO, Machado JC (2020) Using a Hybrid Approach to Data Management in Relational Database and Blockchain: A Case Study on The E-health Domain. In: 2020 IEEE international conference on software architecture companion (ICSA-C), pp 114–121. IEEE, Salvador, Brazil. https://doi.org/10.1109/ICSA-C50368.2020.00030
45. Bertram S, Georg C-P (2018) A privacy-preserving system for data ownership using blockchain and distributed databases. arXiv. arXiv:1810.11655
46. Ermolaev V, Klangberg I, Madhwal Y, Vapper S, Wels S, Yanovich Y (2020) Incorruptible auditing: blockchain-powered graph database management. In: 2020 IEEE international conference on blockchain and cryptocurrency (ICBC). IEEE, Toronto,

ON, Canada, , pp 1–3. https://doi.org/10.1109/ICBC48266.2020.9169431
47. Guo Q, Deng S, Cai L, Zhu Y, Zhang Z, Jin C (2020) Blockchain PG: Enabling Authenticated Query and Trace Query in Database. In: Wang, X., Zhang, R., Lee, Y.-K., Sun, L., Moon, Y.-S. (eds.) Web and big data, vol 12318. Springer, Cham, pp 529–534. https://doi.org/10.1007/978-3-030-60290-1_41
48. Lian J, Wang S, Xie Y (2021) TDRB: an efficient tamper-proof detection middleware for relational database based on blockchain technology. IEEE Access 9:66707–66722. https://doi.org/10.1109/ACCESS.2021.3076235
49. Nathan S, Govindarajan C, Saraf A, Sethi M, Jayachandran P (2019) Blockchain meets database: design and implementation of a blockchain relational database. Proc VLDB Endow 12(11):1539–1552. https://doi.org/10.14778/3342263.3342632
50. Schuhknecht FM, Sharma A, Dittrich J, Agrawal D (2021) chainifydb: How to get rid of your blockchain and use your DBMS instead. In: 11th Conference on innovative data systems research, CIDR 2021, virtual event, January 11–15, 2021, Online Proceedings
51. Beirami A, Zhu Y, Pu K (2019) Trusted relational databases with blockchain: design and optimization. Procedia Comput Sci 155:137–144. https://doi.org/10.1016/j.procs.2019.08.022
52. Sahoo MS, Baruah PK (2018) HBasechainDB: a scalable blockchain framework on hadoop ecosystem. In: Yokota R, Wu W (eds.) Supercomputing frontiers vol 10776. Springer, Cham, pp 18–29. https://doi.org/10.1007/978-3-319-69953-0_2
53. Qi X, Zhang Z, Jin C, Zhou A (2021) A reliable storage partition for permissioned blockchain. IEEE Trans Knowl Data Eng 33(1):14–27. https://doi.org/10.1109/TKDE.2020.3012668
54. Qi X, Zhang Z, Jin C, Zhou A (2020) BFT-Store: Storage Partition for Permissioned Blockchain via Erasure Coding. In: 2020 IEEE 36th international conference on data engineering (ICDE). IEEE, Dallas, TX, USA, pp 1926–1929. https://doi.org/10.1109/ICDE48307.2020.00205
55. Qi X, Chen Z, Zhang Z, Jin C, Zhou A, Zhuo H, Xu Q (2021) A Byzantine Fault tolerant storage for permissioned blockchain. In: Proceedings of the 2021 international conference on management of data. ACM, Virtual Event China, pp 2770–2774. https://doi.org/10.1145/3448016.3452744
56. Huang H, Peng X, Zhan J, Zhang S, Lin Y, Zheng Z, Guo S (2022) BrokerChain: a cross-shard blockchain protocol for account/balance-based state sharding. In: IEEE INFOCOM 2022—IEEE conference on computer communications. IEEE, London, UK, pp 968–1977. https://doi.org/10.1109/INFOCOM48880.2022.9796859
57. Xu Y (2018) Section-Blockchain: a storage reduced blockchain protocol, the foundation of an autotrophic decentralized storage architecture. In: 2018 23rd international conference on engineering of complex computer systems (ICECCS). IEEE, Melbourne, VIC, pp 115–125. https://doi.org/10.1109/ICECCS2018.2018.00020
58. Zheng P, Xu Q, Zheng Z, Zhou Z, Yan Y, Zhang H (2021) Meepo: sharded consortium blockchain. In: 2021 IEEE 37th international conference on data engineering (ICDE). IEEE, Chania, Greece, pp 1847–1852. https://doi.org/10.1109/ICDE51399.2021.00165
59. Al-Bassam M, Sonnino A, Bano S, Hrycyszyn D, Danezis G (2018) Chainspace: A Sharded Smart Contracts Platform. In: Proceedings 2018 network and distributed system security symposium. Internet Society, San Diego, CA. https://doi.org/10.14722/ndss.2018.23241
60. Chen Z, Qi X, Du X, Zhang Z, Jin C (2021) PEEP: A Parallel Execution Engine for Permissioned Blockchain Systems. In: Jensen, C.S., Lim, E.-P., Yang, D.-N., Lee, W.-C., Tseng, V.S., Kalogeraki, V., Huang, J.-W., Shen, C.-Y. (eds.) Database Systems for Advanced Applications vol. 12683, pp. 341–357. Springer International Publishing, Cham. https://doi.org/10.1007/978-3-030-73200-4_24
61. Xu C, Zhang C, Xu J, Pei J (2021) SlimChain: scaling blockchain transactions through off-chain storage and parallel processing. Proc VLDB Endow 14(11):2314–2326. https://doi.org/10.14778/3476249.3476283
62. Fang M, Zhang Z, Jin C, Zhou A (2021) High-performance smart contracts concurrent execution for permissioned blockchain using SGX. In: 2021 IEEE 37th international conference on data engineering (ICDE). IEEE, Chania, Greece, pp 1907–1912. https://doi.org/10.1109/ICDE51399.2021.00175
63. Chen Z, Zhuo H, Xu Q, Qi X, Zhu C, Zhang Z, Jin C, Zhou A, Yan Y, Zhang H (2021) SChain: a scalable consortium blockchain exploiting intra- and inter-block concurrency. Proc VLDB Endow 14(12):2799–2802. https://doi.org/10.14778/3476311.3476348
64. Fang M, Zhou X, Zhang Z, Jin C, Zhou A (2022) SEFrame: an SGX-enhanced smart contract execution framework for permissioned blockchain. In: 2022 IEEE 38th international conference on data engineering (ICDE). IEEE, Kuala Lumpur, Malaysia, pp 3166–3169. https://doi.org/10.1109/ICDE53745.2022.00289
65. Xu C, Zhang C, Xu J (2019) vChain: enabling verifiable boolean range queries over blockchain databases. In: Proceedings of the 2019 international conference on management of data. ACM, Amsterdam Netherlands, pp 141–158. https://doi.org/10.1145/3299869.3300083
66. Adkins D, Agarwal A, Kamara S, Moataz T (2020) Encrypted blockchain databases. In: Proceedings of the 2nd ACM Conference on Advances in Financial Technologies. ACM, New York, NY, USA, pp 241–254. https://doi.org/10.1145/3419614.3423266
67. Yan D, Jia X, Shu J, Yu R (2021) A blockchain-based database system for decentralized information management. In: 2021 IEEE global communications conference (GLOBECOM). IEEE, Madrid, Spain, pp 1–6. https://doi.org/10.1109/GLOBECOM46510.2021.9685695
68. Wang H, Xu C, Zhang C, Xu J, Peng Z, Pei J (2022) vChain+: optimizing verifiable blockchain boolean range queries. In: 2022 IEEE 38th international conference on data engineering (ICDE). IEEE, Kuala Lumpur, Malaysia, pp 1927–1940. https://doi.org/10.1109/ICDE53745.2022.00190
69. Pang S, Shao Q, Zhang Z, Jin C (2020) AuthQX: enabling authenticated query over blockchain via Intel SGX. In: Nah Y, Cui B, Lee S-W, Yu JX, Moon Y-S, Whang SE (eds) Database systems for advanced applications vol 12114. Springer, Cham, pp 727–731. https://doi.org/10.1007/978-3-030-59419-0_45
70. Shao Q, Pang S, Zhang Z, Jing C (2020) Authenticated range query using SGX for blockchain light clients. In: Nah Y, Cui B, Lee S-W, Yu JX, Moon Y-S, Whang SE (eds) Database systems for advanced applications vol. 12114. Springer, Cham, , pp 306–321. https://doi.org/10.1007/978-3-030-59419-0_19
71. Shao Q, Zhang Z, Jin C, Zhou A (2021) Trusted sliding-window aggregation over blockchains. In: 2021 IEEE 27th international conference on parallel and distributed systems (ICPADS). IEEE, Beijing, China, pp 257–265. https://doi.org/10.1109/ICPADS53394.2021.00038
72. Jia D-Y, Xin J-C, Wang Z-Q, Lei H, Wang G-R (2021) SE-chain: a scalable storage and efficient retrieval model for blockchain. J Comput Sci Technol 36(3):693–706. https://doi.org/10.1007/s11390-020-0158-2
73. Wang S, Dinh TTA, Lin Q, Xie Z, Zhang M, Cai Q, Chen G, Ooi BC, Ruan P (2018) Forkbase: an efficient storage engine for blockchain and forkable applications. Proc VLDB Endow 11(10):1137–1150. https://doi.org/10.14778/3231751.3231762

74. Ruan P, Chen G, Dinh TTA, Lin Q, Ooi BC, Zhang M (2019) Fine-grained, secure and efficient data provenance on blockchain systems. Proc VLDB Endow 12(9):975–988. https://doi.org/10.14778/3329772.3329775
75. Ruan P, Dinh TTA, Lin Q, Zhang M, Chen G, Ooi BC (2021) LineageChain: a fine-grained, secure and efficient data provenance system for blockchains. VLDB J 30(1):3–24. https://doi.org/10.1007/s00778-020-00646-1
76. Ruan P, Kanza Y, Ooi BC, Srivastava D (2022) LedgerView: access-control views on hyperledger fabric. In: Proceedings of the 2022 international conference on management of data. ACM, Philadelphia PA USA, pp 2218–2231. https://doi.org/10.1145/3514221.3526046
77. Amiri MJ, Agrawal D, Abbadi AE (2019) CAPER: a cross-application permissioned blockchain. Proc VLDB Endow 12(11):1385–1398. https://doi.org/10.14778/3342263.3342275
78. Gupta S, Hellings J, Sadoghi M (2020) RCC: resilient concurrent consensus for high-throughput secure transaction processing. arXiv. arXiv:1911.00837
79. Kant K, Pandey S, Shanker U (2022) A journey from commit processing in distributed databases to consensus in blockchain. In: 2022 IEEE 38th international conference on data engineering (ICDE). IEEE, Kuala Lumpur, Malaysia, pp 3236–3240. https://doi.org/10.1109/ICDE53745.2022.00306
80. Dang H, Dinh TTA, Loghin D, Chang E-C, Lin Q, Ooi BC (2019) Towards scaling blockchain systems via sharding. In: Proceedings of the 2019 international conference on management of data. ACM, Amsterdam Netherlands, pp 123–140. https://doi.org/10.1145/3299869.3319889
81. Schuhknecht FM, Sharma A, Dittrich J, Agrawal D (2019) Chainifydb: how to blockchainify any data management system. CoRR arXiv.1912:04820
82. Helmer S, Roggia M, Ioini NE, Pahl C (2018) EthernityDB: integrating database functionality into a blockchain. In: Benczúr A, Thalheim B, Horváth T, Chiusano S, Cerquitelli T, Sidló C, Revesz PZ (eds) New trends in databases and information systems vol 909. Springer, Cham, pp 37–44. https://doi.org/10.1007/978-3-030-00063-9_5
83. El-Hindi M, Binnig C, Arasu A, Kossmann D, Ramamurthy R (2019) BlockchainDB: a shared database on blockchains. Proc VLDB Endow 12(11):1597–1609. https://doi.org/10.14778/3342263.3342636
84. Peng Y, Du M, Li F, Cheng R, Song D (2020) FalconDB: blockchain-based collaborative database. In: Proceedings of the 2020 ACM SIGMOD international conference on management of data. ACM, Portland OR USA, pp 637–652. https://doi.org/10.1145/3318464.3380594
85. Zhu Y, Zhang Z, Jin C, Zhou A (2020) Enabling generic verifiable aggregate query on blockchain systems. In: 2020 IEEE 26th international conference on parallel and distributed systems (ICPADS). IEEE, Hong Kong, pp 456–465. https://doi.org/10.1109/ICPADS51040.2020.00066
86. Amazon: Amazon Quantum Ledger Database (2019). https://aws.amazon.com/qldb/
87. Yang X, Zhang Y, Wang S, Yu B, Li F, Li Y, Yan W (2020) Ledgerdb: a centralized ledger database for universal audit and verification. Proc VLDB Endow 13(12):3138–3151. https://doi.org/10.14778/3415478.3415540
88. Zhang M, Xie Z, Yue C, Zhong Z (2020) Spitz: A verifiable database system. Proc VLDB Endow 13(12):3449–3460. https://doi.org/10.14778/3415478.3415567
89. Yue C, Dinh TTA, Xie Z, Zhang M, Chen G, Ooi BC, Xiao X (2022) Glassdb: practical verifiable ledger database through transparency. CoRR arXiv:abs/2207.00944
90. Azaria A, Ekblaw A, Vieira T, Lippman A (2016) MedRec: using blockchain for medical data access and permission management. In: 2016 2nd International conference on open and big data (OBD). IEEE, Vienna, Austria, pp 25–30. https://doi.org/10.1109/OBD.2016.11
91. Aswathy SV, Lakshmy KV (2019) BVD: a blockchain based vehicle database system. In: Thampi SM, Madria S, Wang G, Rawat DB, Alcaraz Calero JM (eds) Security in computing and communications, vol 969. Springer, Singapore, pp 220–230. https://doi.org/10.1007/978-981-13-5826-5_16
92. Kotobi K, Bilen SG (2018) Secure blockchains for dynamic spectrum access: a decentralized database in moving cognitive radio networks enhances security and user access. IEEE Veh Technol Mag 13(1):32–39. https://doi.org/10.1109/MVT.2017.2740458
93. Vainshtein Y, Gudes E (2021) Use of blockchain for ensuring data integrity in cloud databases. In: Dolev S, Margalit O, Pinkas B, Schwarzmann A (eds) Cyber security cryptography and machine learning, vol 12716. Springer, Cham, pp 325–335. https://doi.org/10.1007/978-3-030-78086-9_25
94. Mashatan A, Lemieux V, Lee SHM, Szufel P, Roberts Z (2021) Usurping double-ending fraud in real estate transactions via blockchain technology. J Database Manag 32(1):27–48. https://doi.org/10.4018/JDM.2021010102
95. Hao K, Xin J, Wang Z, Cao K, Wang G (2019) Blockchain-based outsourced storage schema in untrusted environment. IEEE Access 7:122707–122721. https://doi.org/10.1109/ACCESS.2019.2938578
96. Zheng W, Zheng Z, Chen X, Dai K, Li P, Chen R (2019) NutBaaS: a blockchain-as-a-service platform. IEEE Access 7:134422–134433. https://doi.org/10.1109/ACCESS.2019.2941905
97. Hu S, Cai C, Wang Q, Wang C, Luo X, Ren K (2018) Searching an encrypted cloud meets blockchain: a decentralized, reliable and fair realization. In: IEEE INFOCOM 2018—IEEE conference on computer communications. IEEE, Honolulu, HI, pp 792–800. https://doi.org/10.1109/INFOCOM.2018.8485890
98. Li D, Han D, Zheng Z, Weng T-H, Li H, Liu H, Castiglione A, Li K-C (2022) MOOCsChain: a blockchain-based secure storage and sharing scheme for MOOCs learning. Comput Standards Interfaces 81:103597. https://doi.org/10.1016/j.csi.2021.103597
99. Pandey D, Agrawal N, Jhanwar MP (2020) Covidbloc: a blockchain powered exposure database for contact tracing. IACR Cryptol. ePrint Arch, 1543
100. Ravishankar B, Kulkarni P, Vishnudas MV (2020) Blockchain-based database to ensure data integrity in cloud computing environments. In: 2020 International conference on mainstreaming block chain implementation (ICOMBI). IEEE, Bengaluru, India, pp 1–4. https://doi.org/10.23919/ICOMBI48604.2020.9203500
101. Sui Z, Lai S, Zuo C, Yuan X, Liu JK, Qian H (2019) An encrypted database with enforced access control and blockchain validation. In: Guo F, Huang X, Yung M (eds) Information security and cryptology, vol 11449. Springer, Cham, pp 260–273. https://doi.org/10.1007/978-3-030-14234-6_14
102. Zhang C, Xu C, Xu J, Tang Y, Choi B (2019) GEMˆ2-Tree: A Gas-Efficient Structure for Authenticated Range Queries in Blockchain. In: 2019 IEEE 35th international conference on data engineering (ICDE). IEEE, Macao, Macao, pp 842–853. https://doi.org/10.1109/ICDE.2019.00080
103. Zhang C, Xu C, Wang H, Xu J, Choi B (2021) Authenticated keyword search in scalable hybrid-storage blockchains. In: 2021 IEEE 37th international conference on data engineering (ICDE). IEEE, Chania, Greece, pp 996–1007. https://doi.org/10.1109/ICDE51399.2021.00091
104. Luu L, Narayanan V, Zheng C, Baweja K, Gilbert S, Saxena P (2016) A secure sharding protocol for open blockchains. In:

Proceedings of the 2016 ACM SIGSAC conference on computer and communications security. Association for Computing Machinery, New York, NY, USA, pp 17–30. https://doi.org/10.1145/2976749.2978389

105. Kokoris-Kogias E, Jovanovic P, Gasser L, Gailly N, Syta E, Ford B (2018) Omniledger: a secure, scale-out, decentralized ledger via sharding, vol 2018. San Francisco, CA, United states, pp 583–598. https://doi.org/10.1109/SP.2018.000-5
106. Zamani M, Movahedi M, Raykova M (2018) Rapidchain: Scaling blockchain via full sharding. In: Proceedings of the 2018 ACM SIGSAC conference on computer and communications security. Association for Computing Machinery, New York, NY, USA, pp 931–948. https://doi.org/10.1145/3243734.3243853
107. Wang J, Wang H (2019) Monoxide: scale out blockchains with asynchronous consensus zones. In: 16th USENIX Symposium on networked systems design and implementation (NSDI 19), pp 95–112
108. Hamming RW (1950) Error detecting and error correcting codes. Bell Syst Tech J 29(2):147–160. https://doi.org/10.1002/j.1538-7305.1950.tb00463.x
109. Reed IS, Solomon G (1960) Polynomial codes over certain finite fields. J Soc Ind Appl Math 8(2):300–304. https://doi.org/10.1137/0108018
110. Rani K, Sharma C (2019) Tampering detection of distributed databases using blockchain technology. In: 2019 Twelfth international conference on contemporary computing (IC3). IEEE, Noida, India, pp 1–4. https://doi.org/10.1109/IC3.2019.8844938
111. Jnoub N, Klas W (2019) Detection of tampered images using blockchain technology. In: 2019 IEEE international conference on blockchain and cryptocurrency (ICBC), pp 70–73. https://doi.org/10.1109/BLOC.2019.8751300
112. Trent M, Rodolphe M, Andreas M, Dimitri DJ, Troy M, Greg M, Ryan H, Sylvain B, Alberto G. Bigchaindb: a scalable blockchain database (draft)
113. RethinkDB: RethinkDB. https://rethinkdb.com/
114. Yanovich Y, Ivashchenko I, Ostrovsky A, Shevchenko A, Sidorov A (2018) Exonum: byzantine fault tolerant protocol for blockchains. bitfury.com, pp 1–36
115. Muzammal M, Qu Q, Nasrulin B (2019) Renovating blockchain with distributed databases: an open source system. Futur Gener Comput Syst 90:105–117. https://doi.org/10.1016/j.future.2018.07.042
116. Tsoulias K, Palaiokrassas G, Fragkos G, Litke A, Varvarigou TA (2020) A graph model based blockchain implementation for increasing performance and security in decentralized ledger systems. IEEE Access 8:130952–130965. https://doi.org/10.1109/ACCESS.2020.3006383
117. Iliakis K, Koliogeorgi K, Litke A, Varvarigou T, Soudris D (2022) GPU accelerated blockchain over key-value database transactions. IET Blockchain 2(1):1–12. https://doi.org/10.1049/blc2.12011
118. Zhou X, Chai C, Li G, Sun J (2022) Database meets artificial intelligence: a survey. IEEE Trans Knowl Data Eng 34(3):1096–1116. https://doi.org/10.1109/TKDE.2020.2994641
119. Jia D, Xin J, Wang Z, Wang G (2021) Optimized data storage method for sharding-based blockchain. IEEE Access 9:67890–67900. https://doi.org/10.1109/ACCESS.2021.3077650
120. Jia D, Xin J, Wang Z, Guo W, Wang G (2020) An Optimized Data Distribution Model for ElasticChain to Support blockchain scalable storage. In: Cao J, Vong CM, Miche Y, Lendasse A (eds) Proceedings of ELM 2018, vol 11. Springer, Cham, pp 76–85. https://doi.org/10.1007/978-3-030-23307-5_9
121. Partheeban P, Kavitha V (2022) Blockchain based cloud service security architecture with distributed machine learning for smart device traffic record transaction. Concurr Comput Pract Exp 34(3). https://doi.org/10.1002/cpe.6583
122. Zhou Q, Zheng K, Zhang K, Hou L, Wang X (2022) Vulnerability analysis of smart contract for blockchain-based iot applications: a machine learning approach. IEEE Internet Things J 9(24):24695–24707. https://doi.org/10.1109/JIOT.2022.3196269
123. Shakya S, Mukherjee A, Halder R, Maiti A, Chaturvedi A (2022) Smartmixmodel: machine learning-based vulnerability detection of solidity smart contracts. In: 2022 IEEE international conference on blockchain (Blockchain), pp 37–44. https://doi.org/10.1109/Blockchain55522.2022.00016