Blockchain: Research and Applications 4 (2023) 100116

Contents lists available at ScienceDirect

Blockchain: Research and Applications

journal homepage: www.journals.elsevier.com/blockchain-research-and-applications

Review Article

Databases fit for blockchain technology: A complete overview

Jovan Kalajdjieski [a,**], Mayank Raikwar [b,*], Nino Arsov [a], Goran Velinov [a], Danilo Gligoroski [b]

[a] Faculty of Computer Science and Engineering, Ss. Cyril and Methodius University, 1000 Skopje, North Macedonia
[b] Norwegian University of Science and Technology (NTNU), NO-7491 Trondheim, Norway

ARTICLE INFO

Keywords:
Blockchain
Databases
Transactions
ACID
Analytics
Translytical

ABSTRACT

Efficient data storage and query processing systems play a vital role in many different research areas. Blockchain technology and distributed ledgers attract massive attention and trigger multiple projects in various industries. Nevertheless, blockchain still lacks the features of a Database Management System (DBMS or simply databases), such as high throughput, low latency, and high capacity. For that purpose, there have been many proposed approaches for handling data storage and query processing solutions in the blockchain. This paper presents a complete overview of many different DBMS types and how these systems can be used to implement, enhance, and further improve blockchain technology. More concretely, we give an overview of 10 transactional, an extensive overview of 14 analytical, 9 hybrids, i.e., translytical, and 13 blockchain DBMSs. We explain how database technology has influenced the development of blockchain technology by unlocking different features, such as Atomicity, Consistency, Isolation, and Durability (ACID), transaction consistency, rich queries, real-time analysis, and low latency. Using a relaxation approach analogous to the one used to prove the Consistency, Availability, Partition tolerance (CAP)-theorem, we postulate a "Decentralization, Consistency, and Scalability (DCS)-satisfiability conjecture" and give concrete strategies for achieving the relaxed DCS conditions. We also provide an overview of the different DBMSs, emphasizing their architecture, storage manager, query processing, and implementation.

1. Introduction

Cryptocurrency has recently attracted immense attention from the industry and academia. The first cryptocurrency, Bitcoin [1], has been quite successful so far, with its capital market reaching 922 billion USD in December 2021 [2]. The initial acceptance of Bitcoin was triggered by its usage as a token of value that provided an alternative to government-issued currencies. However, after Bitcoin, many cryptocurrencies were designed that brought a paradigm shift in the financial market. As a result, the financial market of cryptocurrencies is growing rapidly and is currently worth around 2 trillion USD [2]. In the meantime, several new financial forms based on blockchain have been introduced, such as the "Decentralized Finances (DeFi)" with a market capitalization of 140 billion USD and "Collectibles & Non-Fungible Tokens (NFTs)" with a market capitalization of 58 billion USD (as of December 2021 [2]).

Blockchain was first proposed in 2008 and implemented in 2009 [1]. It is viewed by some authors as a disruptive technology [3] that has ultimately changed the market, the financial market in particular, and has triggered a new way of thinking across many industries. Blockchain can be interpreted as a public, distributed ledger for peer-to-peer (P2P) transactions executed securely and stored in an immutable way. Blockchain stores transactions in a growing chain of blocks. Its characteristics are decentralization, persistence, anonymity, and auditability. It can work in a decentralized environment enabled by integrating several core technologies, such as cryptographic hash functions, digital signatures (based on asymmetric cryptography), and a distributed consensus mechanism [4]. Blockchain transactions occur in a decentralized fashion. In a decentralized network, there is no point of single failure, and thus, the network becomes secure and tamper-proof. In contrast, a centralized network is much less secure since a failing central server or database will result in the failure of the entire network. The absence of a central

* Corresponding author.
** Corresponding author.
E-mail addresses: jovan.kalajdzhieski@finki.ukim.mk (J. Kalajdjieski), mayank.raikwar@ntnu.no (M. Raikwar), nino.arsov@gmail.com (N. Arsov), goran.velinov@finki.ukim.mk (G. Velinov), danilog@ntnu.no (D. Gligoroski).

https://doi.org/10.1016/j.bcra.2022.100116
Received 2 February 2022; Received in revised form 18 November 2022; Accepted 28 November 2022

authority increases the computational cost of the transactions, but the overhead costs decrease. The P2P transfer of value between the parties does not rely on an external or third-party authority, eliminating the need for the parties to know each other. The consensus makes most of the forgery attempts impossible because every transaction is validated, reducing the risk of tampering. Other than that, blockchain networks are much more accessible: anyone with a computer and an internet connection can join the network. Although blockchain has enormous potential in terms of efficiency, it still suffers from challenging shortcomings. Some of them comprise, but are not limited to, scalability, energy consumption, privacy leakage, and selfish mining. With the recent advances in blockchain technology, however, these challenges have been addressed by appropriate solutions [5].

Apart from its application to finance, blockchain can also be used in the Internet of Things (IoT), public and social services, reputation systems, or act as a service providing security and privacy. Blockchain can also be viewed as a database (DB) because it equips a storage mechanism [6]. After the emergence of Bitcoin, questions arose about the difference between blockchain and databases and whether blockchain can be used as a database mechanism [7]. While blockchain does provide a storage mechanism, it has, in fact, many critical differences with traditional databases, chiefly its decentralization, cryptographic security using chained hashes, no administration control, immutability, and freedom to transfer without the permission of any central authority [8]. Many enterprise applications have switched to blockchain storage to provide a more secure implementation and involve less trust among the parties in the industry.

As many enterprises are focusing [9–11] on decentralized cloud storage, the question that arises is to what extent blockchain provides a better storage solution than decentralized cloud storage or just a traditional database. Not only does blockchain pertain to the features mentioned above, but it also offers better capabilities, user privacy, and security using cryptographic primitives. On the other hand, blockchain lacks some of the features provided by traditional databases, such as rich queries. The efforts to include these DB functionalities in the blockchain are continuously growing and progressing.

If blockchain storage is measured by the traditional DB criteria, it does not provide promising results. It has a throughput of just a few transactions per second (tps), a 10-min latency before a single confirmed write (for Bitcoin), and a capacity of just a few dozen gigabytes. In blockchain, adding nodes to the network increases traffic but also reduces performance. On the other hand, a modern distributed database can have a throughput exceeding 1 million tps, several terabytes of capacity, and a latency of only a fraction of a second. Finally, adding nodes improves the throughput and capacity of the system [6].

Analogously, one approach to transforming blockchain into an efficient storage mechanism is by blending traditional databases with existing blockchain technologies, which is currently being done by many platforms. The goal is to produce an enhanced storage mechanism with high throughput, scalability, low latency, and a greater extent of security since this mechanism would be built on top of the existing blockchain technologies.

Another approach is to use the so-called blockchain databases that have been evolving extensively in recent years [6,12–17]. These databases have their own consensus mechanism for the network parties' joint agreement on a data block. They are disturbed and offer support for complex data types, rich query structures, and ACID-compliant transactions [18]. Furthermore, they provide low latency, fast scalability, and cloud hosting. Hence, leveraging blockchain and database functionality together makes these databases a promising alternative in many enterprise solutions. Moreover, many of the blockchain platforms are now integrated with some databases. The existing databases, which are used in some of the blockchain systems, either have master-slave or P2P architecture through which the replication of transaction logs and states [19] takes place among the nodes. As the database can be SQL/relational or post-relational, selecting the database for use in blockchain should be based on many critical points, such as database structure and query processing speed.

The development of databases integrated with blockchain features is an active research topic. This paper analyzes the blockchain capabilities and the fit for blockchain of modern database solutions. In general, blockchain capabilities are enabled by either adding a blockchain layer on top of the existing database layer or, vice versa, by implementing blockchain functionalities within the database layer and the architectural model. First, we sublimate seven baselines, namely, decentralization, consistency, scalability, immutability, low latency, high throughput, and sharding, that add up to the fir of database systems for blockchain. Then, we describe and analyze 10 transactional, 14 analytical, 9 hybrid, and 13 blockchain database systems concerning these baselines. This work is the first of its kind and aims to provide a summarized overview that will help choose the database management system that best fits their needs.

The rest of the paper is organized as follows. In Section 2, we provide an overview of the CAP and DCS theorems and how the two mutually influence each other. We also analyze the main characteristics of the types of databases and present a table that summarizes the database systems' fit for blockchain according to the properties, capabilities, and features needed for the blockchain. The different types of blockchain databases, split into three categories—transactional, analytical, and hybrid (i.e., translytical)—are described in Section 3. In Section 4, we provide a detailed explanation of how database and blockchain technology mutually influence and improve each other. We continue with the challenges of joining these two technologies in Section 4.2 and provide a summary of databases adapted for blockchain. Section 5 presents a brief overview of the databases and the DCS theorem. Finally, Section 6 concludes the paper with future research directions.

2. DCS: the CAP theorem for blockchain

CAP refers to Consistency, Availability, and Partition tolerance. CAP was introduced 20 years ago by Brewer [18] as a principle or conjecture, and two years later, it was proven in the asynchronous network model as a CAP theorem by Gilbert and Lynch in Ref. [20]. In the same paper, they proved similar impossibility results for a partially synchronous network model. Additionally, by weakening the consistency conditions, they showed that it is possible to achieve all three properties in the so-called t-Connected Consistency model. The CAP theorem is a fundamental theorem for defining the transaction properties of a distributed system. A distributed system is defined as a set of computing nodes connected over a network that communicate with one another to perform some tasks.

In more detail, the CAP theorem identifies the three specific system properties for any distributed/decentralized system. These properties are **C**onsistency, **A**vailability, and **P**artition tolerance.

- **Consistency**—Any read in the distributed system gives the latest write on each computing node.
- **Availability**—A client always receives a response at any point of time irrespective of whether the read is the latest write, which means a request for some data is always available.
- **Partition tolerance**—In the case of a partition between nodes in the distributed system (when a subset of nodes fail to operate), the system should still be functioning.

The CAP theorem states that it is possible to achieve two of these three properties as guaranteed features in a distributed system, but it is impossible to accomplish all three features simultaneously. In practice, a distributed system always needs to be partition tolerant, thus leaving us to choose one property from Consistency or Availability. Hence, there is a trade-off between consistency and availability. After the influence of CAP in the NoSQL movement [21], it became a big debate on the trade-offs in database systems to achieve strong consistency guarantees for the database systems [22].

The CAP theorem has also influenced the blockchain realm (see, for

example, Ref. [23]). In the blockchain context, CAP properties mean: 1) Consistency: All nodes in blockchain have an identical ledger with the most recent update; 2) Availability: A transaction generated at any point of time in the network will be accepted and available in a block of the blockchain; 3) Partition tolerance: Even if a part of nodes in the blockchain P2P network fails to operate, the network can still perform the normal operations. Partition tolerance corresponds to decentralization in the blockchain context. A CAP theorem in the blockchain context refers to that a widely accepted blockchain is hard to exist without all three properties.

If we pick Availability over Consistency, any reads are not guaranteed to be up-to-date, and we call the system AP. However, if we choose Consistency over Availability, the system called CP would be unavailable at the time of partition and might disrupt the consensus. Thus, in blockchain systems, both properties are desirable. Although blockchain does not always require strong consistency, eventual consistency can serve the purpose and can be achieved through consensus. For example, in the case of Bitcoin, the longest chain method brings eventual consistency, but there are no fixed methods to achieve eventual consistency, which leaves this topic for debate. Fig. 1 shows the different database systems according to the CAP theorem.

PACELC Theorem: The Partition, Availability, Consistency, Latency, and Consistency (PACELC) theorem [30] is grounded on the CAP model. It is best described by the following statement: "If Partition (P) happened, then choose between Availability (A) or Consistency (C), Else (E) choose between Latency (L) or Consistency (C)." It is considered a more pertinent approach to the design of distributed systems (since blockchain by itself is a concrete type of distributed system) and has had a more significant influence than the CAP theorem. The generally perceived influence of CAP is rather overestimated [30]. At its core, PACELC extends to Latency in addition to Consistency, Availability, and Partition tolerance by adding a logical exclusion between the combinations of the four concepts based on whether or not a network partition has occurred. PACELC also introduces a new pertinent trade-off—between consistency and latency—that has influenced the design of modern distributed database systems. As soon as the system replicates data, provided no network partition has happened, based on how the system handles reads, the consistency-latency trade-off subsumes: (1) synchronous replication, entailing consistency over latency (the master node waits until all updates have made it to the replicas, ensuring consistency to the detriment of latency), (2a) asynchronous replication with routing through the master node, entailing consistency over latency (routing the read requests through the master node ensures consistent read results, but a busy, failed, or a distant master node increases the latency of the reads), (2b) asynchronous replication with reads served from any node, entailing latency over consistency (different nodes have different versions of the requested data item), and (3) synchronous and asynchronous replication (a combination of (1), (2a), and (2b) that allocates a subset of nodes to which data are replicated synchronously, while the rest of the nodes are updated asynchronously; if reads are routed to at least one synchronously updated node, then consistency is retained at the expense of latency, otherwise, the reads have a low latency, but likely inconsistent results).

DCS Theorem: The Decentralization, Consistency, and Scalability (DCS) theorem [31,32] was proposed as an analogy to the CAP theorem for blockchain. The DCS theorem states that a blockchain system can simultaneously satisfy at most two out of the three properties: decentralization, consistency, and scalability. Although the PACLEC theorem is built on top of the CAP theorem for real systems, it is easier to illustrate the DCS theorem analogous to CAP. Due to the fact that both theorems state that a distributed system can satisfy at most two out of three properties, it is also easier to map DCS properties with CAP properties. The DCS properties are defined as follows:

- **Decentralization**—There is no trusted entity controlling the network, hence no single point of failure. Blockchains are inherently decentralized, but in the DCS triangle, we are considering the case of complete decentralization. In the case of complete decentralization, any node can join the network and participate as a validator.
- **Consistency**—The blockchain nodes will read the same data at the same time. A query for the blockchain data on any blockchain node should fetch the same result. The consistency in blockchain should prevent double-spending and should be brought from the consensus algorithm used.
- **Scalability**—The performance of blockchain should increase with an increase in the number of peers and the number of allocated computational resources. The system's throughput and capacity should be high, and latency should be low.

Similar to CAP, we can also categorize the blockchain systems in DCS as DC, CS, and DS systems in terms of trade-offs between the DCS properties. Fig. 2 depicts the various systems, according to the DCS theorem. Following, we define these systems in detail:

- **DC Systems.** The main focus of these systems is to provide a consistent blockchain state in a decentralized network environment. Most of the cryptocurrencies, such as Bitcoin [1] and Ethereum [33], fall into the category of DC systems. Many of these systems have Proof of Work (PoW) as their consensus mechanism, where each node stores the full data of the blockchain. Furthermore, nodes (miners) in these systems are incentivized to increase their chances of successfully mining a block using ASIC (Application-Specific Integrated Circuit) farms. Therefore, over time, these systems do not scale.

Achieving higher performance negatively impacts the consistency of these systems. Therefore, to achieve eventual consistency, these systems employ some techniques such as the longest chain rule in Bitcoin or the GHOST protocol [36] in Ethereum as the branch selection rule.

- **CS Systems.** All the permissioned blockchains do not have complete decentralization; hence, they can be regarded as CS systems if they achieve a planetary scale. These systems have a certain degree of decentralization, as not every node can be a validator that takes part in consensus. The nodes in these systems have distributed computing resources, and to join the consensus, a node can become a validator by a fair voting mechanism among the existing federation of nodes.

Permissioned blockchains such as Hyperledger Fabric [37] have scalability bottlenecks due to the use of communication-intensive protocols, such as the Practical Byzantine Fault Tolerance (PBFT) protocol [38]. A decent amount of work [39,40] has been done to scale the

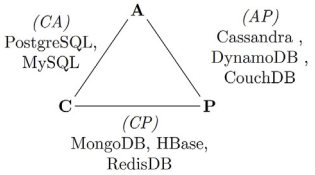

Fig. 1. CAP (Consistency, Availability, Partition tolerance) triangle for database systems [24–29].

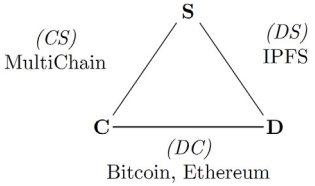

Fig. 2. DCS (Decentralization, Consistency, Scalability) triangle for blockchain systems [1, 33-35].

Hyperledger Fabric. Furthermore, a recent work [41] provides a novel method to practically improve the scalability of permissioned blockchains.

- **DS Systems.** These systems are decentralized and provide good scalability with compromise on consistency. A node of this system does not try to store the whole network; instead, it is limited by the node's capacity. The Interplanetary File System (IPFS) [35] is a DS system that does not provide consistency, as different parts of the data are distributed to different nodes. It is like a mashup of Git and BitTorrent.

Although consistency prevents double-spending, DS systems can still be useful in many applications that do not store tokens of values, such as storing data and assets. The IPFS can have eventual consistency by employing Conflict-free Replicated Data Type (CRDT) [42].

Conjecture 1. (DCS-satisfiability). There exists a well-balanced and relaxed set of requirements for DCS properties such that a blockchain system can have all three properties satisfied.

While for the CAP theorem, the relaxation of the requirements was achieved by introducing the *t*-connected consistency model [20], and we can apply a similar relaxation approach to prove the above conjecture. Slepak and Petrova [31] showed that a decentralized consensus system cannot have all three properties simultaneously. They also presented two methods to get around the DCS triangle. We use a similar idea to present a set of requirements or methods so that a blockchain system can have all three properties. Following, we present requirements/methods for each of the above-defined three systems to get around the DCS triangle. Note that we describe only a few methods for the DCS-satisfiability conjecture; we do not exhaust all the possible requirements/methods.

The following discussion presents general methods to get around the DCS triangle, especially those more aligned with blockchain systems. These methods are applied to make blockchain systems achieve DCS properties and can also be employed in real systems. Nevertheless, for databases, a brief description is provided in Section 4.1. Section 4.1 presents many examples of different databases and how these databases can achieve all three DCS properties simultaneously.

- **DC Systems → DCS**: DC systems can achieve scalability by a few methods. We describe the following two methods for DC systems to scale.

1. **Using Layer-2 Solutions** DC systems can scale by employing layer-2 scaling protocols [43]. Layer-2 solutions handle the transaction in an off-chain manner and hence provide scalability. Although there are many layer-2 solutions, some of the solutions tend toward centralization. Therefore, finding a decentralized layer-2 solution is necessary in order to make a DC system scalable. These decentralized layer-2 solutions (e.g., Lightning Network [44]) can be categorized as DS systems. Thus, combining a DC system with a DS system makes the whole system DCS, as depicted in Fig. 3.

2. **Using Sharding** DC systems can scale using sharding [45]. Sharding is a technique where the transactions are split among a group of nodes called shards. These shards work in parallel to improve the throughput and hence achieve better scalability. These shards should have a certain degree of transparency to trust the other shards and participate in consensus. A shard can be considered a small DC system. Hence, the parallel working of shards corresponds to combining multiple DC systems that make the whole system DCS, as depicted in Fig. 4.

- **CS Systems → DCS**: CS systems can have decentralization by having interoperability among multiple CS systems. In this sense, a single CS system does not have full control of the system. Interoperability in blockchain systems can be achieved using cross-chain methods such as cross-chain communication protocols [46] or using cross-chain interoperability using transactions [47]. Cross-chain methods basically combine multiple CS systems and make a DCS system, as depicted in Fig. 5.

- **DS Systems → DCS**: DS systems can achieve consistency by using safe and verifiable smart contracts [48,49], together with making the blockchain attack-resilient and handling the forks. Verifiable smart contracts help to maintain the consistency of blockchain data. In addition to such smart contracts, preventing blockchain attacks, e.g., double-spending attacks, and by handling the event of forks in blockchain can lead DS systems towards greater consistency. Validation in such systems is simpler, usually by only checking hashes, and there is no strict order of transactions to maintain. However, there is implicit ordering based on the hashed objects. There are three levels of consistency: no consistency, strong eventual consistency, and CAP consistency. Data replication methods can also be used to achieve a certain level of consistency in DS systems, hence making the DS system DCS. One example is CRDT [42], which ensures consistency and data integrity when there are conflicts between different data updates.

DS systems, such as IPFS, use CRDT, which guarantees strong eventual consistency. CRDT is a code-versioning system with the addition of no-merge conflict. However, CRDT does not solve all problems, and double-spend attacks persist in DS systems. Nevertheless, Ref. [50] addresses the double-spend attack by building a swarm coin on top of a CRDT that makes the attack expensive. The DS system can hence become DCS.

3. Databases fit for blockchain and blockchain databases

The databases summarized in Table 2 show how individual systems comply with the DCS theorem and our conjecture. Here, we describe and analyze in detail 10 transactional, 14 analytical, 9 hybrid (also called "translytical"), and 13 blockchain database solutions in terms of the functionalities they provide out of the box for implementing blockchain. Each of the three types has its advantages with respect to the others when implementing or storing a blockchain. Most well-known solutions offer functionalities that support blockchain out of the box, such as the Oracle Database and Microsoft SQL Server, while others, such as PostgreSQL and MySQL, can also be leveraged for blockchain, although they do not provide application-ready functionalities. But there are also emerging technologies that are good candidates because of their creative architectural models and features that aim to meet the demand of contemporary applications. Before describing the transactional, analytical, hybrid, and blockchain database systems, we emphasize decentralized and distributed systems.

Decentralized systems do not have a central node. Instead, the system

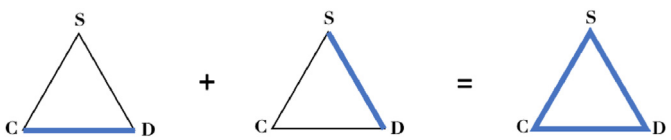

Fig. 3. DC System to DCS using Layer-2 solutions.

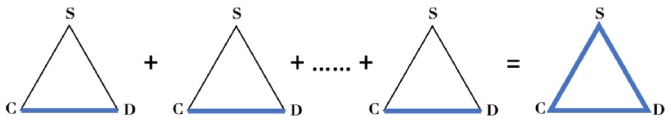

Fig. 4. DC System to DCS using sharding.

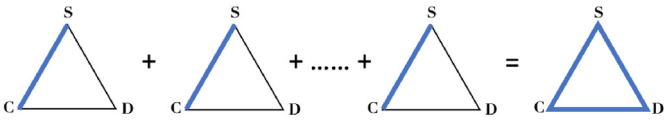

Fig. 5. CS System to DCS using cross-chain methods.

has multiple central nodes that store the copy of resources. In a decentralized system, each node of the system makes its own decision. The final behavior of the system is the aggregate decision of the individual nodes. This ensures that no single node has complete system information. Decentralized systems are more tolerant of faults. This is because when one or more central nodes fail, the other nodes still keep the system running.

Distributed systems are similar to decentralized systems, but the decision-making can be performed in a centralized or decentralized manner. In a distributed system, nodes are physically separated but communicate and coordinate by passing messages. In a distributed system, nodes have equal access to data, though the privileges can be enabled when required. These systems promote resource sharing and have better fault-tolerant and scalability.

3.1. Transactional databases

One can store a blockchain in a traditional SQL database. This kind of storage allows fast access to blockchain data and an efficient mechanism for analyzing blockchain data by complex queries. Thus, SQL adds many capabilities for the blockchain and makes it easier to work on blockchain data using SQL tools and libraries. Below, we give an extensive analysis of the usage of different SQL databases for blockchain and their characteristics.

AergoLite[1] allows us to have a replicated SQLite database secured by a private and lightweight blockchain. Each app has a local replica of the database. New database transactions are distributed to all the peers, and once they reach a consensus on the order of execution, all the nodes execute the transactions. As the execution order of the transactions is the same, all the nodes have the same resulting database content. AergoLite uses special blockchain technology focused on resource-constrained devices. The consensus protocol uses a Verifiable Random Function (VRF) to determine which node will produce the next block, and the nodes cannot discover which node is selected ahead of time, which makes it safe against Denial of Service (DoS) attacks. AergoLite uses absolute finality. Once the nodes reach a consensus on a new block and confirm the transactions, there is no way back. Only the last block is required to check the blockchain and the database state integrity; therefore, the nodes do not need to keep and verify all the history of blocks and transactions. It is also possible to set up some nodes to keep all the history for audit reasons. It also uses a hash of the database state.

CovenantSQL (CQL) is a decentralized, trusted, and GDPR-compliant database with blockchain features built on top of SQLite that makes the database immutable. It is a Byzantine Fault Tolerance (BFT) relational database built on SQLite. It can be used as a low-cost Database as a Service (DBaaS). CQL has a layered architecture consisting of a global consensus Layer, SQL consensus Layer, and datastore Layer. It supports two consensus algorithms: Delegated Proof of Stake (DPoS)[2] and Byzantine Fault Tolerant RAFT (BFT-RAFT) [51]. CovenantSQL provides an infrastructure to facilitate and build decentralized apps with full SQL support. It can also be integrated into asset management and IoT solutions.

Dqlite ("distributed SQLite") is an open-source, fast, disk-backed database with in-memory options. It supports SQLite transactions, and it uses the C-RAFT consensus, which is an optimized RAFT [52] with high performance, ultra-low latency, and fault tolerance. In case one or more machines fail, consensus ensures data persistence as long as the majority survives. Hence, it is best suited for fault-tolerant IoT and Edge devices. Dqlite has not been used in blockchain, but it can be helpful in blockchain implementation.

Microsoft SQL Server [53] is a proprietary Relational Database Management System (RDBMS). As a mainly centralized system, it offers a wide variety of native data types and supports a wide variety of transaction processing, business intelligence, and analytical applications in corporate IT environments. **Azure Blockchain Workbench** offers an SQL database suitable for blockchain. It is built on top of Azure SQL [54], the intelligent, scalable, and distributed relational database cloud service using the Microsoft SQL Server Engine. Azure Blockchain Workbench delivers data from distributed ledgers to an off-chain SQL Database, offering a messaging API and a REST API for abstraction. The REST API allows users to create and manage blockchain applications and workflows within a blockchain consortium, with full support for blockchain transactions. It also offers high availability, tuning, backups, and execution of complex SQL queries and is mainly suited to build private blockchains. Its PACELC categorization is PC/EC, which ensures consistency in all cases.

MySQL [55] and its community-developed fork **MariaDB** is an open-source relational database system with advanced replication and clustering features. In MySQL, the biggest challenge in implementing the blockchain life-cycle is its distribution and replication across multiple instances. In terms of PACELC, it is a consistency-favoring system to the detriment of availability under network partition and latency under normal operation. **OurSQL**[3] is a standalone server connected to a MySQL database. It is a combination of blockchain and MySQL. OurSQL can be used for private blockchain applications. It supports PoW-type consensus algorithms. It is mostly suited to build private blockchains.

Oracle Database 21c [56] is a proprietary, scalable, reliable, and secure RDBMS built around the relational database framework. Oracle Database 21c resolves concurrency issues by using various types of locks and a multi-version consistency model. To manage the multi-version consistency model, it creates a read-consistent set of data when a table is queried (read) and simultaneously updated (written), thus falling into the consistency-favoring PC/EC category of the PACELC design principle due to its ACID property.

PostgreSQL [57] is an open-source enterprise RDBMS. It has a wide variety of native data types and supports user-defined objects, which can be beneficial for defining blockchain assets in a blockchain system. It is highly modular and extensible, and supports isolation on different levels. PostgreSQL has been used to create a so-called blockchain relational database, where the replicas are managed by different organizations that do not trust each other [58]. PostgreSQL is a PC/EC system that favors consistency over availability/latency in terms of the PACELC principles and supports both immediate and eventual consistency. The latter is used for scaling reasons, and upon a write, all blockchain nodes will be in sync sooner or later, while, in the meantime, the write might be reflected on a blockchain node or still operating on another blockchain node. In contrast, immediate consistency asserts that every read operation will return the value stored by the latest write in the whole system, regardless of the accessed blockchain node.

RQLITE is an open-source, lightweight, fault-tolerant, and distributed relational database. RQLITE uses SQLite databases and RAFT [52] as the consensus mechanism. It allows the dynamic creation of a cluster of nodes and provides node-to-node encryption. Therefore, RQLITE appears to be a potential candidate to be used in blockchain systems, especially lightweight blockchain solutions.

[1] https://aergolite.aergo.io/.
[2] https://en.bitcoinwiki.org/wiki/DPoS.
[3] https://en.bitcoinwiki.org/wiki/DPoS, http://oursql.org, https://covenantsql.io, https://dqlite.io, http://www.rqlite.com/.

SQLite [59] is an embedded, non-client-server, ACID-compliant relational database system. It is suitable to be embedded as a local database in the blockchain nodes. To implement blockchain in SQLite, it is first necessary to implement the basic crypto, blockchain, and database operations. Then, it is essential to implement the networking interface that will render SQLite as a P2P network. Finally, key management operations will allow entities to sign someone else's key as the most important process. However, automatic key management is still a challenging functionality. Usually, there will be no cryptocurrency layer, and the metadata for each block in the chain will be stored in the database. The consensus rules for accepting new blocks would have to be adapted in a database fashion.

VoltDB [60] is an open-source, in-memory distributed database that focuses specifically on fast data. Increasing the number of nodes in a VoltDB cluster both increases throughput (by increasing the number of simultaneous queues in operation) and increases the data capacity (by increasing the number of partitions used for each table). The new version of VoltDB V8 adds many blockchain capabilities. It provides SQL support for the traversal of blockchain records with recursive Common Table Expressions (CTEs). Using CTE, implementing a blockchain in VoltDB provides high throughput and low latency and allows complex queries. Ultimately, VoltDB is a PC/EC system in terms of the PACELC design principles, favoring consistency over availability under network partition and latency under normal operation.

3.2. Analytical databases

This section gives an extensive overview of 14 analytical database engines, including column stores and analytical systems used for data warehousing. The most popular and commonly known system is the Apache Hadoop framework, whose blockchain-related features are analyzed in detail in the next section.

Apache Hadoop [61] is a framework that provides distributed storage and allows the processing of large data sets across clusters of commodity hardware. It delivers a high-availability service by automatically redirecting jobs to healthy nodes when a node failure occurs, consolidated by data replication. Unlike traditional relational databases, it supports unstructured data, such as text, images, and videos.

There has been much research done on integrating the Hadoop system with blockchain technology, such as Refs. [62–64]. The substantial and diverse Hadoop ecosystem contains six prominent modules suitable for storing and analyzing blockchain data: **HBase**, **HBasechainDB**, **Spark**, **Hive**, **Pig**, and **Kudu**.

HBase [65] is a scalable, distributed, column-oriented non-relational database management system. It is tuned for storing massive data sets by leveraging master-slave replica architecture. It provides a fault-tolerant mechanism for storing sparse data sets, which are common in big data use cases. It is well suited for real-time data processing or random read/write access to large volumes of data. It ensures consistency over availability/latency and is thus a PC/EC system in terms of PACELC.

Spark [66] is a data processing framework that can quickly perform processing tasks on very large data sets by distributing the processing tasks to multiple nodes, either on its own or in combination with other distributed computing tools. Its integration with HBase enables developers to perform high-performance analytic tasks.

Kudu [67] is a free and open-source column-oriented storage system developed for Apache Hadoop. It supports low-latency random access millisecond-scale access to individual rows together with great analytical access patterns. It aims to integrate HDFS and HBase. Kudu merges the upsides of HBase and Parquet[4], being as fast as HBase at ingesting data and almost as quick as Parquet when it comes to analytical queries. As required, Apache Kudu uses the RAFT consensus algorithm to scale up or down horizontally. It also has an update-in-place feature. It can scale up to tens of cores per node and even benefit from SIMD operations for data-parallel computation. Kudu has already been used for Ethereum blockchain analytics[5], integrated with Hadoop.

Apache Cassandra [27] is one of the most popular NoSQL databases developed by Facebook. Due to the excellent throughput of massive writes, many large enterprises like Netflix, Instagram, GitHub, and eBay use Cassandra. It is a fully decentralized system and provides excellent performance, durability, and fault tolerance without compromising availability. In addition, Cassandra supports lightweight transactions to establish consistency. The wide-column storage architecture, the Paxos Consensus, and the similarity to DynamoDB make it a good fit for blockchain. It is a definite candidate for blockchain storage solutions, but, to this date, no concrete implementations exist.

In terms of PACELC, Apache Cassandra is a PA/EL system, which means that if partition occurs, it favors availability to the detriment of consistency, and otherwise, its normal operation favors low latency over consistency.

BigQuery is an enterprise, scalable data warehouse that solves the massive data sets problem by enabling super-fast SQL queries using the processing power of Google's infrastructure. It is a cloud-based big data analytics service provided by Google. Because of the high performance offered by this service, it is most suitable for blockchain data analytics. One such approach is explained in Ref. [68], where BigQuery is used to query an Ethereum network. Also, six cryptocurrency blockchain data sets are released for comparative analyses.

ClickHouse[6] is an open-source column-oriented DBMS. Because of its high-performance characteristics, ClickHouse is a viable storage solution for blockchain and analytics performed on blockchain data, as explained in Ref. [69].

CouchDB [29] is a key-value database by Apache that provides rich querying capabilities similar to MongoDB. For instance, CouchDB supports content-based JSON queries, which in turn makes it appropriate to be used in Hyperledger Fabric. In Hyperledger Fabric [37], CouchDB acts as a state database for storing chaincode-processed transaction data as key-value pairs. It supports rich queries against chaincode data. Hyperledger Composer also uses CouchDB by converting SQL queries into the aforementioned content-based JSON queries supported by CouchDB. Another benefit of CouchDB enabling it to be leveraged for blockchain is its scalable clustering ability. The cluster can run on an arbitrary number of nodes. CouchDB supports multiple levels of availability, making it an 'eventual-consistency' system, sacrificing immediate consistency for huge performance improvements as data grow. Thus, CouchDB is a PA/EL system in terms of PACELC, but its adjustable availability makes it possible to have a PC/EC system.

CrateDB[7] is a distributed database that integrates a fully searchable document-oriented data store based on a shared-nothing architecture. It offers the scalability and flexibility typically associated with a NoSQL database and is designed to run on inexpensive commodity machines. Its shared-nothing architecture provides greater deployment flexibility in heterogeneous environments. To date, there has been no research done on the integration of CrateDB in a blockchain system, but based on its high performance, it could be a possible solution.

DynamoDB [28] is a fully managed NoSQL database service that provides fast and predictable performance with seamless scalability offered by Amazon Web Services (AWS). It provides key-value and document models and delivers single-digit millisecond performance at any scale. It is a multi-master, durable database with built-in security and in-memory caching. DynamoDB's encryption at rest eliminates the

[4] https://parquet.apache.org/.

[5] https://www.phdata.io/blog/hadoop-meets-blockchain-trust-your-big-data/.

[6] https://clickhouse.tech/.

[7] https://cbrate.io/.

operational burden and complexity involved in protecting sensitive data. DynamoDB has already been employed in many different blockchain systems [70,71], where it has been used on the same level as the blockchain system, serving only for storage and validation purposes. However, with the AWS modules for blockchain support, DynamoDB's usage is extended even further in blockchain solutions. DynamoDB adheres to PA/EL, sacrificing consistency for the sake of availability when network partition occurs, or low latency under normal operation. Although DynamoDB's fit for blockchain is obscure, one such approach is explained in Ref. [72], where it has been used as a building block for private Ethereum networks.

Elasticsearch [73] is a distributed open-source search and analytics engine for textual, numerical, geospatial, structured, and unstructured data. It stores data as JSON documents and provides Elasticsearch queries for querying the data. It provides a quorum-based consensus algorithm and the primary-backup approach. Elasticsearch's fit for blockchain is related to the Elastic Stack (ELK Stack) built on top of it, a set of open-source tools for data ingestion, enrichment, storage, analysis, and visualization. The ELK Stack is very commonly used for observability on a blockchain, enabling users to easily monitor their blockchain systems. One such approach is the integration of the Elastic Stack as a monitoring tool for the Hyperledger Fabric framework, as explained in Ref. [74].

ScyllaDB[8] is an open-source distributed NoSQL column-oriented data store. Its shard-based design makes it compatible with Apache Cassandra while achieving significantly higher throughput and lower latency. At each node, it handles data in parallel by sharding it into multiple subsets and assigning each shard to a different CPU core. The cores do not share shards but rather communicate when they need data from another core. Although there is still no implementation using ScyllaDB as a storage solution in blockchain projects, this database has been used in the IOTA project[9], which is a Tangle that falls under the same top-level category as blockchain-distributed ledgers. ScyllaDB has been used as the distributed storage solution because it provides fault tolerance, data consistency, fast and efficient data queries, and other features essential for distributed ledger technologies [75].

TiesDB [76] is a public, decentralized, and distributed database. The ties Network is a deep modification of the Cassandra database. Because of its flexibility regarding the underlying NoSQL database, TiesDB inherits most of its features. It adds BFT, while most NoSQL databases lack BFT. It supports sharding, smart contracts, and incentive schemes. It can be used to build decentralized applications providing fast data retrieval. Interestingly, TiesDB uses blockchain technology to enhance the architecture of the database [77]. However, it does not store data on the blockchain unit due to its complexity, which cannot ensure high-speed data operations over large volumes of storage. Nevertheless, TiesDB implements blockchain technology to build a decentralized management center that stores mostly passive and read-only key meta-information that does not affect the operational speed of the database.

Voldemort[10] is a distributed key-value store used by LinkedIn for highly scalable storage. Voldemort cannot satisfy arbitrary relations and is not ACID-compliant. It is rather a large, distributed, and persistent hash table. Its fit for blockchain originates from the built-in fault-tolerant Paxos-style consensus algorithm and in-memory caching. It has high performance, following the PA/EL PACELC principle for ensuring high availability/low latency when replication occurs, and, as such, is a promising solution for blockchain systems.

3.3. Hybrid (translytical) databases

This section presents an overview of nine hybrid databases.

Aerospike [78] is an open-source distributed key-value in-memory NoSQL database with excellent scaling capabilities. It supports flexible data schemas, high availability and scalability, and ACID transactions. It also supports SSD storage persistence. Because of its flash-optimized and SSD storage layer and architecture, it provides high performance and almost the best massive write performance on the market. Thus, it has a big potential for use in blockchain applications.

Azure CosmosDB [79] is Microsoft's globally distributed, fully decentralized, multi-model database. It supports key-value, graph, and document data models. The key features of CosmosDB are its huge storage capability, high throughput, low latency, and high scalability. It delivers high availability and consistency with comprehensive Service Level Agreements (SLAs). It offers multi-master replication across various regional distributions. From a PACELC perspective, it follows PA/EL, favoring availability under network partition or low latency under normal operation at the expense of consistency in both cases. However, CosmosDB supports multi-level consistency tuning, making it possible to follow PC/EC, which essentially follows the latency-consistency trade-off introduced in Ref. [30]. Many enterprises can benefit from building a decentralized blockchain application using CosmosDB.

CockroachDB [80] is a scalable and distributed key-value database that is highly adaptive and open-source. It supports strongly consistent ACID semantics and horizontal scalability. It also uses the RAFT consensus protocol for replica synchronization and uses an SQL API for querying and structuring the data. CockroachDB is currently being used by the company Tierion to implement its blockchain. CockroachDB provides a variety of benefits that ease the process of implementing blockchain. In short, they use Merkle Trees to build resistant hashes starting from the leaf nodes of the tree and use a specific rewarding system to reward healthy blockchain nodes.

Greenplum Database [81] is a Massively Parallel Processing (MPP) database server with an architecture specially designed to manage large-scale analytic data warehouses and business intelligence workloads. MPP (also known as a shared-nothing architecture) refers to systems with two or more processors cooperating to carry out an operation, each processor with its own memory, operating system, and disks. Greenplum uses this high-performance system architecture to distribute the load of multi-terabyte data warehouses and can use all of a system's resources in parallel to process a query. The Greenplum Database is based on the PostgreSQL open-source technology, where several PostgreSQL instances are organized into a distributed, shared-nothing database management system. It inherits most of the PostgreSQL features, such as SQL and transactions.

InfinityDB[11] is an embedded NoSQL database engine, a hierarchically sorted key-value store based on a B + Tree. InfinityDB supports a rich data representation space. ItemSpace is the lowest-level data model containing a set of items. The basic data model can be used to define huge spare arrays, matrices, any mixture of trees, graphs, key/value maps, key/value maps, or user-defined structures. It is a high-performance, multi-core, flexible, and maintenance-free database. InfinityDB is appropriate for embedded hardware platforms, text indexing engines, distributed industrial data collection systems, and heterogeneous data environments. **InfinityDB Encrypted**[12] is identical to InfinityDB but encrypts 100% of the database 100% of the time; hence, it is fit for the blockchain.

MongoDB [24] is the fastest-growing document-based database in the market. The distributed architecture of MongoDB makes it an ideal platform for building blockchain databases. MongoDB offers data model flexibility, high scalability, robust security, complex queries, and SQL capabilities. Due to its powerful technological features, it is used by many leading enterprises nowadays. The MongoDB Enterprise edition supports encryption, auditing, sharding, and access control. From a PACELC perspective, MongoDB is a PA/EC system, ensuring high availability at

[8] https://www.scylladb.com/.
[9] https://www.iota.org/.
[10] https://www.project-voldemort.com/voldemort/.
[11] https://boilerbay.com/.
[12] https://boilerbay.com/infinitydb-encrypted/.

the expense of consistency when partition occurs, while under normal operation, it ensures consistency, making data retrieval latency a second priority. It implements a consensus protocol and leader election and also offers SQL capabilities.

OrientDB [82] was developed as a direct response to polyglot persistence. It is a multi-model open-source NoSQL DBMS that combines the power of graphs with document, key/value, reactive, object-oriented, and geospatial models into a single scalable, high-performance operational database. It also provides support for ACID transactions. The multi-model flexibility allows it to store data in different formats and makes it a very promising storage system for blockchain.

Redis [26] is an open-source, in-memory data structure store that is widely used as a database, cache, or message broker. It supports data structures such as strings, hashes, lists, sets, sorted sets with range queries, bitmaps, hyperloglogs, geospatial indexes with radius queries, and streams. Redis has built-in replication, Lua scripting, Least Recently Used (LRU) eviction, transactions, and different levels of on-disk persistence and provides high availability via Redis Sentinel and automatic partitioning with Redis Cluster. Because of its in-memory data set, it provides outstanding performance. Apart from its usage as a storage system, Redis has been used as a message broker for many blockchain systems [83,84].

TiDB [85] ("Titanium DB") is an open-source and distributed SQL database with strong consistency and high availability. It supports Hybrid Transactional and Analytical Processing workloads. It has a modular design containing three components for cluster coordination, replicating key-value stores, and scheduling SQL queries. It uses the RAFT consensus protocol for replica consistency.

3.4. Blockchain databases

This section presents an overview of 13 blockchain databases.

BigchainDB [6] is like a database possessing blockchain characteristics, hence providing decentralization and immutability. It claims to solve the transaction speed and storage problem. BigchainDB works by offering an API on top of the underlying database. It was initially built upon the RethinkDB [86] cluster, and it provides rich query functionalities, high throughput, and low latency, hence achieving high performance. It enables enterprises to design decentralized applications by deploying proof-of-concept on them. Recently, it collaborated with MongoDB, and from version 2.0, it employs Tendermint consensus [87] over a set of independent MongoDB instances, each owned by different organizations. BigchainDB supports storing multiple assets of various types, e.g., an intellectual property right, a data set.

BlockchainDB [88] implements a database layer on top of an existing blockchain system. The database layer provides an abstraction called a shared table and stores all data in its storage layer. The shared tables are partitioned (i.e., sharded), which allows each shard to be implemented as a separate blockchain network. The shards are only replicated to a limited number of peers. To obtain higher speeds, not all peers store all data locally. In BlockchainDB, a peer can be either deployed as a full peer that hosts a database and a replica of at least one shard or as a thin peer that only connects to other remote peers to access data in a shard. In this way, a peer participates in a network and allows clients to read/write data into a shared database. Having thin peers enables parties with only limited resources to participate in a BlockchainDB network and access the shared tables.

ChainifyDB [17] adds a blockchain-characteristic layer on top of a standard database layer, and the standard database is plugged into the ChainfyDB network to synchronize the database records. Hence, it allows enterprises to build decentralized cutting-edge blockchain applications on top of their database systems. ChainifyDB implements the WLC model (Whatever-Ledger Consensus), a novel processing method. The core idea is not to seek consensus on what actions should be taken but to seek consensus on the effect of the actions after they have been performed. One of the core features of ChainifyDB is to upgrade existing infrastructures consisting of several DBMSs. However, the challenge is that the existing infrastructures can be highly heterogeneous, i.e., every participant could potentially run a different DBMS where each system can interpret a specific transaction differently. As a result, the effects across participants may differ.

EthernityDB [89] integrates database functionalities in the Ethereum blockchain [33]. It modularizes the Ethereum smart contracts and deploys the database functionalities in it. EthernityDB uses MongoDB as the database for coupling with the Ethereum blockchain and provides an interface to perform rich queries on deployed smart contracts using the Solidity language.

FalconDB [15] is a shared database consisting of multiple servers with verification interfaces accessible to clients. FalconDB stores the digests for query/update authentications on a blockchain. Using blockchain as a consensus platform and a distributed ledger, FalconDB is able to work without any trust in each other. Meanwhile, FalconDB requires only minimal storage cost on each client and provides anywhere-available, real-time, and concurrent access to the database. As a result, FalconDB overcomes the disadvantages of needing expensive hardware or having low performance and enables individual users to participate in the collaboration with high efficiency, low storage cost, and blockchain-level security guarantees.

FlureeDB [90] is a scalable blockchain database. It consolidates blockchain with the document and graph databases to support a broad range of industrial use cases. It provides rich access capability directly inside the database. It offers transactions, sharding, censorship resistance, privacy, and cloud hosting. It uses composite consensus that enables multiple databases to be queried as a single database. Each block of blockchain in FlureeDB represents a unique time moment, and this feature is called "time-travel". FlureeDB comes with key features, e.g., multiple sharding, cloud hosting, and low latency.

HBasechainDB [91] is a scalable big data store based on the concept of blockchain. It adds the blockchain characteristics of immutability and decentralization to the HBase database. HBasechainDB can be used as a tamper-proof, decentralized, and distributed big data store. It can also be used for Hadoop-enabled existing ecosystems for efficient immutable data storage.

LedgerDB [14] is a centralized ledger database built on Alibaba Cloud, with tamper-evidence and non-repudiation features similar to blockchain. LedgerDB provides strong auditability by adopting a two-way peg protocol called TSA, which prevents malicious behaviors from users and service providers. LedgerDB supports verifiable data removals demanded by many real-world applications, which are able to remove obsolete records for storage savings and hide records for regulatory purposes without compromising their verifiability. The authors claimed that LedgerDB has much higher throughput compared to blockchains, and based on their experimental evaluation, LedgerDB's throughput is 80 $\times$ higher than state-of-the-art permissioned blockchains (i.e., Hyperledger Fabric). They also claimed that many blockchain customers on Alibaba Cloud have switched to LedgerDB for its high throughput, low latency, strong auditability, and ease of use.

Modex BCDB [92] is a blockchain database that applies Proof of Stake (PoS) consensus and supports multiple databases, including SQL and NoSQL databases. It provides a blockchain layer between a database and a client application. It also has a plug-and-play approach for organizations to develop blockchain-enabled software systems. It supports multiple blockchain frameworks, such as Hyperledger Sawtoothlake and Tendermint, as a blockchain layer. It also supports multiple database systems, such as MongoDB and MySQL.

The **Oracle Blockchain Platform** is a comprehensive and distributed ledger cloud and on-premise platform that enables the building of blockchain networks and the support for many types of tables, including blockchain tables. The platform uses cryptographic hashing for tamper-resistance, and digitally-signed updates are mandatory in transactions. It offers a distributed ledger replicated across organizations, multi-signature and decentralized trust, and smart contracts across multiple

nodes. A key blockchain feature of the platform is that it is interoperable with other Hyperledger Fabric blockchain nodes.

The blockchain table in Oracle Database 19c/21c is a simpler concept but offers higher throughput for insert transactions and an SQL or PL/SQL programming model. It is a tamper-proof, insert-only table with an associated table-level and row-level retention period, where rows are organized into chains, with each row containing a hash of the data contained in the row, and the hash of the previous row data.

Postchain [93] is the first consortium database, as claimed by the company ChromaWay. It combines the features of a mature distributed database and blockchain. A blockchain solution can be implemented using Postchain and SQL. Postchain is BFT-enabled, secure, flexible, scalable, and has powerful features to manage integrity, validation, and data independence, along with the inherited traits from the underlying standard database. Postchain applies Proof of Authority (PoA) consensus to maintain the data in a network of nodes. Postchain transactions are defined using SQL, and a network of validator nodes handles these transactions. These transactions are communicated by using encryption and signed messages. The nodes from the validator set validate the proof and synchronize their databases.

ProvenDB [94] adds the blockchain characteristic layer on top of the MongoDB database. Hence, it leverages enterprises to build decentralized cutting-edge blockchain applications on top of their database systems. It is compatible with MongoDB and stores multiple versions of the database, which are cryptographically provable, immutable, and tamper-proof. It solves the performance dilemma by anchoring several thousands of database transactions to the blockchain in a single operation. ProvenDB delivers a temper-resistant secure digital data store where all sorts of sensitive data can be stored, such as legal and financial records.

QLDB Amazon QLDB [95] is a ledger database that abstracts many blockchain features. It renders a tamper-resistant, transparent, and cryptographically verifiable ledger of transactions. To implement a blockchain using QLDB, it is necessary to set up an entire blockchain network with multiple nodes and implement verification procedures for each transaction in the blockchain.

QLDB lacks decentralization and hence does not follow any consensus algorithm. Therefore, it best suits enterprises that do not require any consensus and still want to have the immutability of their data. QLDB also supports SQL queries, leveraging SQL developers to run complex queries on the stored data.

Veritas [16] is a new abstraction for trusted data sharing in the cloud by adding trust and auditability to existing database management systems. To be able to add these capabilities to existing database systems, the authors proposed the implementation of shared, verifiable database tables. These tables integrate the tables from the blockchain database directly into the databases of the nodes—as if they were to share a single, common instance of this table. The authors also proposed using the Caesar Consensus to support tamper-evident collaboration across mutually untrusted entities.

4. Databases and blockchain: Mutual influence and development

The primary motivation behind this paper was to investigate how databases and blockchain can be coupled together and benefit each other. Blockchain and databases can both achieve many functionalities and features, adding to each other. If we frame blockchain as a database that provides a storage mechanism, then we can analyze how it differs from existing database systems.

4.1. The key features

The following are the key points where blockchain and database differ in their properties, but both can leverage and enhance the characteristics of each other.

- Traditional blockchain throughput decreases when the processing capacity of nodes participating in the blockchain increases. Yet, in the case of the distributed database, the throughput increases when the nodes increase. Hence, throughput can be enhanced.
- The latency of transactions in the blockchain is usually high compared to the latency in the database. Thus, the latency can potentially be lowered using a database.
- Transactions in blockchain require serializable isolation, which can be achieved by consensus algorithms, providing strong consistency, and other mechanisms—in databases, there is a well-understood mechanism called two-phase locking and concurrency control [96]. However, new blockchain databases, such as BlockchainDB [88], based on MongoDB [24], are beginning to offer new transaction mechanisms based on blockchain. Blockchain transactions should be ACID-compliant [18], which is an inbuilt property of the database.
- Most blockchain platforms do not support complex queries in their historical data. These queries are needed in many applications to retrieve the desired information. The complex query feature is available in most databases, but the provenance queries on historical data can be supported by the use of Multi-Version Concurrency Control [97].
- The decentralization feature of blockchain has rewired most of the financial systems and industries over the last decade. However, decentralization is not available in the traditional distributed database. With the advent of new blockchain-style databases, decentralization is now possible and has promising potential for many applications.
- The transaction replication feature of blockchain has had a significant influence on the development of new distributed database systems. Blockchains replicate an ordered log of transactions (or ledger), while distributed databases replicate the ordered log of read and write operations on top of the storage. This paradigm has been adopted in so-called 'NewSQL' distributed databases, which aim to retain ACIDity and attain vertical scalability in addition to horizontal scalability through the adoption of various blockchain-inspired consensus algorithms [98].
- One of the other excellent features of blockchain is the immutability or tamper-resistance of transactions. Tamper-resistance can be achieved in database systems by mechanisms that disallow deletes and updates in the database.
- Blockchain allows the creation and movement of digital assets, which is not permitted in a classical database. However, a blockchain-style distributed database can have this feature as a built-in feature.

In Table 1, we give a summary of this mutual influence and the entangled development of databases and blockchain.

Describing the similarities and differences between blockchain and databases, one should not forget about the CAP [20] theorem for a database system that can help make the right choice of database to suit the blockchain needs. Therefore, in the next section, we discuss the CAP theorem modeled for the blockchain.

The entangled development of databases and blockchain is most evident in distributed databases. There is a clear distinction between blockchain and distributed databases in four aspects: replication, concurrency, storage, and sharding [98]. Table 1 gives a slightly different comparison to the one provided by the authors of Ref. [98] and provides information on how databases and blockchain domains influence each other. For instance, the low transaction latency. In recent years, the development of distributed databases fully supporting decentralization has been accelerated by different consensus algorithms, such as RAFT and Paxos variations. In terms of decentralization, Aerospike [78] is an interesting example of a distributed database (a flash-optimized in-memory distributed database) that fully supports high-speed distributed cluster formation based on a Paxos gossip algorithm to ensure agreement on a critical shared state. In terms of the movement of digital assets, its design allows for shards of the data to move from one server to another in

Table 1
A summary of the mutual influence and the entangled development of databases and blockchain.

Feature	Database domain	Influence direction	Blockchain domain
High throughput and scalability	✓ (in distributed databases)	→	To be implemented
Transactions latency	Low	→	High
Serializable isolation	Alternatives to two-phase locking	←	✓
ACID properties	✓	→	Hyperledger Fabric [37] due to CouchDB [29]
Complex queries on the historic data	✓	→	Techniques such as VQL [99]
Decentralization	New, blockchain-style databases	←	✓
Replication	✓ (via a trusted transaction manager, a main replica, and consensus: Paxos, PBFT, RAFT)	←	✓ (primary-backup and chained replication) [98]
Immutability (tamper-resistance)	Mechanisms that prevents deletes and record updates' history	←	✓
Movement of digital assets	New, blockchain-style distributed databases	←	✓
CAP (Consistency, Availability, Partition tolerance)	✓	→	DCS (Decentralization, Consistency, Scalability)

order to handle nodes coming in and out of the cluster while ensuring sufficient replication for horizontal scaling besides the inherent vertical scaling. While Aerospike can support tens of millions of transactions per second using DRAM and millions using Flash storage, cutting-edge distributed blockchain databases like BigchainDB [6], which add blockchain properties on top, introduce a much higher transaction latency and sacrifice vertical scalability. In general, such properties are still in a very premature phase of development in the blockchain domain.

Replication is tightly coupled with decentralization. Replication in the blockchain is implemented on the transaction level. In contrast, distributed databases traditionally maintain a trusted entity that manages the replication of the read/write operations so that the nodes are unaware of the transaction logic. Without the presence of such an entity, blockchains replicate the entire transaction whose execution can then be replayed by each participant. There are two types of replication approaches in blockchains: primary-backup, in which a primary replica synchronizes its state with backup replicas, and a chained, state-based replication approach. The first approach has already been adopted by many distributed databases, such as Apache Cassandra, a widely used open-source database management system. The second approach, chained replication, is more complex and uses consensus protocols on top of a state model to implement and maintain an ordered shared log of transactions. The consensus protocols help the replicas agree on the ordered log [98]. Protocols such as Paxos, RAFT, PBFT, and other non-blocking protocols have increasingly become popular for adoption in distributed database systems. With the rise of cloud-based distributed applications, horizontally and vertically scalable graph databases such as Neo4j [100] and Dgraph [101] have become increasingly popular. Neo4j has adopted chained replication with a non-blocking consensus algorithm, while Dgraph uses RAFT. Aerospike [78] is another example of an in-memory and flash-optimized distributed database that adopted the Paxos protocol to attain extreme degrees of vertical and horizontal scalability.

From CAP and PACELC to DCS in blockchain. As stated in Section 2 and the last row in Table 1, in the blockchain domain, the CAP theorem is replaced by the DCS theorem. In this part, we analyze several blockchain systems, such as BigchainDB, BlockchainDB, Blockchain Relational Database, and ChainifyDB, with respect to the DCS theorem and the DCS-Satisfiability Conjecture given in Section 2. In distributed systems, the more applicable PACELC theorem implies that to balance out the trade-off between latency and consistency when data are replicated, the hybrid combination of synchronous and asynchronous replication could reduce the intensity of the trade-off, providing satisfactory levels of both latency and consistency. Furthermore, the relaxed DCS requirements have been adopted in the design of some of the systems of concern in order to balance out the trade-off between the remaining two properties when one of decentralization, consistency, or scalability is desired.

Scalability (DC → DCS). The most critical factors affecting scalability are the consensus algorithm and the replication strategy. When scalability is desired in a blockchain, a trade-off between decentralization and consistency arises.

The crucial advantage of BlockchainDB is to avoid replicating the data to all peers, thus avoiding the high overhead of a blockchain consensus. Instead, shared tables are partitioned (i.e., sharded). This means that each shard is implemented as a separate blockchain network. The shards are only replicated to a limited number of peers, allowing better trade performance and trust characteristics depending on the application's requirements.

ChainifyDB creates a network of possible heterogeneous DBMSs with different transaction processing systems, where each DBMS keeps an actual replica of the database. If a database deviates for any reason, there is a robust recovery mechanism. The system uses the powerful features of well-established DBMSs, such as SQL, the relational model, and high transaction processing performance. Also, ChainifyDB provides parallel transaction execution based on the strategy when a block of transactions is received by the execute-subphase, and the system first identifies all existing conflict dependencies between transactions. This allows for the formation of mini-batches of transactions that can be executed safely in parallel, as they have no conflicting dependencies.

For a blockchain relational database, some improvements and optimizations should be implemented in the communication layer to ensure the system's scalability.

BigchainDB inherits characteristics of modern distributed databases: linear scaling in throughput and capacity with the number of nodes, a full-featured NoSQL query language, efficient querying, and permissioning. Scalable capacity means that legally binding contracts and certificates may be stored directly on the blockchain database.

Decentralization (CS → DCS). As soon as decentralization is desired in a blockchain system, then a trade-off between consistency and scalability arises.

BlockchainDB assumes that the parties who want to share data are previously authenticated and known to each other, similar to permissioned blockchains. However, parties do not need to trust each other since they may have contrary goals. BlockchainDB inherits several security characteristics and guarantees the underlying blockchain systems, like peer authentication using public/private keys, replay protection, and several tolerable malicious nodes.

ChainifyDB is a flexible permissioned blockchain system. It provides encrypted communication among the databases plugged into the network through an end-to-end encryption mechanism. Moreover, ChainifyDB is seamlessly invasive when plugged into a database, does not require any tool for its maintenance, and presents a web front-end to perform that.

The blockchain relational database is suitable for building a private and permissioned network of organizations that are known to one another but may be mutually distrustful. A new organization must be permissioned to join the decentralized network.

BigchainDB can be used in a fully decentralized setting or as a mild extension of a traditional centralized computing context. In cases where decentralizing just storage brings the majority of benefits or where

scalability needs are more significant than the capabilities of existing decentralized processing technologies. In these cases, BigchainDB provides a bridge to eventual fully-decentralized systems.

Consistency (DS → DCS). As soon as consistency is desired, a trade-off between decentralization and scalability arises. Some systems support tuned consistency levels, while others use custom methods such as snapshot isolation.

BlockchainDB provides different consistency levels on top of blockchains. A sequential consistency guarantees a client to see new values; it has a much higher latency than eventual consistency since it forces a client to wait until pending writes for a key are committed. In contrast, eventual consistency has a significantly lower latency but does not provide any guarantee of the staleness of retrieved values. An eventual consistency with bounded staleness allows clients to trade off staleness for improved read latency. Moreover, it allows online and off-chain verification methods.

In ChainifyDB, the WLC model implies that organizations cannot commit to the effects of their actions if consensus on the effects of specific actions cannot be reached. The core idea behind the WLC model is not to seek consensus on what actions should be taken but to seek consensus on the effect of the actions after they have been performed. This allows the authors to drop assumptions on the concrete transaction processing behavior of the organizations. It also allows them to detect any external tampering with the data. The system supports using arbitrary vote-based mechanisms or consensus mechanisms, such as Paxos and PBFT, depending on the required guarantees. Also, it enables the consensus policy parameter to be set, which defines the number of organizations that must agree to transactions.

In Blockchain Relational Database, the system uses the SSI (Serializable Snapshot Isolation) method to ensure the serializability of the transactions. Also, it provides two approaches to building a consistently replicated ledger across untrusted nodes starting from a relational database: order-then-execute and execute-order-in-parallel. While the first approach is more straightforward and requires fewer modifications to the relational database, the second approach has the potential to achieve better performance. SSI, if directly applied, does not guarantee serializability and consistency across nodes for the execute-order-in-parallel approach.

BigchainDB has the blockchain benefits of decentralized control, immutability, and creation and transfer of assets. The decentralized control is via a federation of nodes with voting permissions. The voting operates at a layer above the DB's built-in consensus. Immutability is achieved via several mechanisms: shard replication, reversion of disallowed updates or deletes, regular database backups, and cryptographic signing of all transactions, blocks and votes.

4.2. Databases adapted for blockchain technology

This section summarizes the availability of blockchain features in database systems and the extent to which transactional, analytical, and hybrid database systems offer blockchain features, characteristics, and capabilities. We used seven baselines—decentralization, consistency, scalability, immutability, low latency, high throughput, and sharding—to analyze and rate the capabilities of a database system to implement blockchain. Table 2 lists all the database systems for which there is enough blockchain-related information available. The right-most column shows the database system's overall rating regarding its fit for blockchain. Some of the database systems described in Section 3 are not included in the table because of scarce information regarding their fit for blockchain. However, some of them are new on the market and possess the potential to implement blockchain.

Our methodology in Table 2 categorizes 29 database management systems into three levels of fitness for blockchain: low, medium, and high. Systems that satisfy all seven baselines, namely decentralization, consistency, scalability, immutability, low latency, high throughput, and sharding, are considered the most suitable for implementing blockchain.

Data Model. The data model plays a significant role when implementing blockchain. The databases analyzed in Table 2 have different models: relational databases, key-value stores, document-oriented databases, graph databases, and wide-column stores. The majority of the systems most suitable for implementing blockchain are relational, but document-oriented key-value or column stores slightly outnumber them. Each model has its own advantages, such as offering SQL capabilities, while graph databases offer modern query language support, such as GraphQL. In general, this means that NoSQL and NewSQL databases are as capable as relational databases when it comes to blockchain fit. However, in general, Table 2 shows that one-third of all the databases are relational, while the rest are document-oriented. Graph databases are the least frequent.

Consensus. The consensus algorithm also plays an important role. The most suitable databases for implementing blockchain use all kinds of different consensus algorithms, and there is no dominant one. These databases use consensus algorithms such as Zen Discover, RAFT, ZAB (ZooKeeper Atomic BoardCast) Consensus, PBFT, BFT-RAFT, Two-Phase Commit, and others. The most widely used consensus is RAFT or its variations.

Decentralization. Almost all of the databases in Table 2 are decentralized. This is one of the key properties required in blockchain. A third of the databases that do not support decentralization implement a relational data model.

Consistency. Most of the databases that cannot guarantee consistency are relational. They are ChainifyDB, Dqlite, OurSQL, and TiesDB. The rest of the databases have the full capabilities of consistency, and only a few have eventual consistency or configurable consistency.

Scalability. Scalability can be either horizontal or vertical. Horizontal scalability enables the distributed database to easily manage huge amounts of replicated data with high availability. In contrast, vertical scalability allows the databases to handle large amounts of transactions in a short period of time. As shown in Table 1, replication approaches from blockchain have been adopted in distributed databases to attain vertical scalability.

Immutability. Immutability is another key feature of blockchain. Traditional databases do not always feature immutability, and since they are tightly coupled with blockchain, databases that do not support immutability cannot be regarded as highly fit for implementing blockchain. Table 2 shows 13 such databases, i.e., more than a third of all 29 databases.

Low latency. Most of the databases feature low latency, but there is a significant number of databases for which such conclusions cannot be drawn. This is also an important feature required to implement blockchain, and these databases are thus not considered highly fit for blockchain.

High throughput. High throughput is correlated with low latency. Table 2 shows the correlation clearly. Database systems that do not have high throughput are considered either a low or a medium fit for blockchain. The absence of low latency and high throughput decreases their suitability for blockchain. High throughput is usually attained by vertical scalability.

Sharding. Sharding is an important baseline for implementing blockchain. There are no highly fit databases that do not support sharding since it has a direct influence on horizontal and vertical scalability. One of the biggest problems in blockchain is scalability, so sharding is a basic prerequisite when implementing blockchain.

The comparison matrix indicates that 9 out of 29 studied database management systems possess properties that render them highly suitable for implementing blockchain. The majority of these systems are document-oriented, some being key-value stores or wide column stores, while only a third are traditional relational database systems. Moreover, these three systems have been commercially successful over the last few decades: Microsoft's SQL Server, Oracle Database, and PostgreSQL; it is well known that PostgreSQL is an open-source project that generally follows the Oracle Database's development life-cycle and inherently

Table 2
Comparison matrix for different systems.

System	Data Model	Consensus	Decentral-ization	Consistency	Scalability	Immutability	Low Latency	High Throughput	Sharding	Blockchain Fit
BigchainDB	Document oriented	Tendermint [87]	✓	✓	✓	✓	✓	✓	✓	High
BlockchainDB	Key-Value	Underlying Blockchain Consensus	✓	✓	–	✓	ą	ą	✓	Medium
Cassandra	Wide Column store, Key-Value	Paxos [102] Consensus	✓	✓*	✓	ą	✓	✓	✓–	Medium
ChainifyDB	Relational	Whatever ledger	✓	–	✓	✓	–	✓	–	Medium
ClickHouse	Column oriented	ZAB [103] Consensus	✓	✓	✓	–	✓	✓	–	Medium
CockroachDB	Key-Value	RAFT Consensus	ą	✓	✓	ą	✓	–	✓	Medium
CosmosDB	Key-Value, Document oriented, Graph	No Consensus	ą	✓**	✓	ą	✓	✓	✓	Medium
CouchDB	Key-Value	No Consensus	ą	✓*	✓	ą	✓	✓	✓–	Medium
CovenantSQL	Relational	DPOS[1], BFT-RAFT [51]	✓	✓	–	✓	–	–	–	Medium
CrateDB	Document oriented	Zen Discovery Consensus [104]	✓	✓*	✓	–	✓	✓	✓	Medium
Dqlite	Relational	C-RAFT [52] Consensus	ą	–	–	ą	✓	✓	–	Low
DynamoDB	Document oriented Key-Value	Paxos [102] Consensus	✓	✓	✓	✓	✓	✓	✓	High
Elasticsearch	Document oriented	Zen Discovery Consensus	✓	✓*	✓	✓	✓	✓	✓	High
FlureeDB	Document oriented, Graph	PBFT, RAFT	✓	✓	✓	✓	✓	✓	✓	High
HBasechainDB	Wide Column store	No, but uses Blockchain	✓	✓	✓	✓	✓	✓	✓	High
Microsoft	Relational	IBFT Consensus	✓	✓	✓	✓	✓	✓	✓	High
MongoDB	Document oriented	RAFT Based	ą	✓	✓	ą	–	ą	✓	Medium
Oracle Database	Relational	RAFT Consensus	✓	✓	✓	✓	✓	✓	✓	High
OurSQL	Relational	PoW type Consensus	✓	–	–	✓	–	–	–	Low
Postchain	Relational	BFT Based	✓	✓	✓	✓	–	–	–	Medium
PostgreSQL	Relational	Two-Phase Commit	✓	✓	✓	✓	✓	✓	✓	High
ProvenDB	Document oriented	Not Mentioned	✓	✓	–	✓	–	ą	–	Medium
QLDB	Document oriented	No Consensus	ą	✓	✓	✓	–	–	–	Medium
Rqlite	Relational	RAFT Consensus	ą	✓**	–	ą	–	–	–	Medium
ScyllaDB	Wide Column store Key-Value	RAFT Consensus	✓	✓	✓	✓	✓	✓	✓	High
TiesDB	Document oriented	BFT Based	✓	–	–	ą	–	–	✓	Low
TitaniumDB	Key-Value	RAFT Consensus	ą	✓	✓	ą	–	–	✓	Low
VoldemortDB	Key-Value	Paxos [102] Consensus	✓	✓*	✓	–	✓	✓	–	Medium
VoltDB	Relational	No Consensus	ą	✓	✓	ą	✓	✓	✓	Medium

Here, '✓' indicates that the feature is present, 'ą' indicates that the feature is not present in the corresponding system, '✓*' represents eventual consistency, '✓**' represents configurable consistency, '✓–' represents that the database has its own sharding method, '–' represents inconclusive data.

implements most of its features. Other widely used solutions, such as MySQL, MariaDB, and SQLite, have still not reached the required maturity level. However, they have 'chainified' variants (forks) with concrete implementations of blockchain, listed and discussed in more detail in Table 3. The consensus algorithm does not have a significant influence on the systems' fit because most of them use well-studied and

Table 3
Original databases adapted for the blockchain ecosystem, along with the different features of the chainified database.

Category	Original DB	Chainfied DB	Maturity	Blockchain Type	Architecture Type	Replication and Architecture	Implementation & Development	Concurrency Control	Query Processing
OLTP	MySQL	OurSQL	Prototype	Private	Out-of-the Database	A blockchain replication tool on top of relational database	Golang, SQL	MVCC, 2 PL	Tuple-at-a-time model
OLTP	SQLite	Aergolite	Framework	Private	Out-of-the Database Blockchains	Replicated SQLite database secured by a private and lightweight blockchain	C/C++, SQL	2 PL	Tuple-at-a-time model
OLTP	SQLite	CovenantSQL	Framework	Public	Out-of-the Blockchain	Decentralized SQL database management system derived from SQLite and built on top of a blockchain	Python, Golang, Java, JavaScript, SQL	2 PL	Tuple-at-a-time model
OLTP	*	Postchain	Framework	Private & Public	Permissioned Blockchains	Combines the features of a mature distributed database and blockchain	Java, NodeJS, PostgreSQL	/	/
OLTP	PostgreSQL	Blockchain relational DB	Research prototype	Permissioned	Out-of-the Database	Decentralized replicated relational share nothing DB with blockchain properties	C++, SQL	MVCC, 2 PC	Tuple-at-a-time model
OLAP	CouchDB	Hyperledger Fabric	Framework	Private	Permissioned Blockchains	State database for storing chaincode processed transaction data as key-value pairs	Erlang, JavaScript, C, C++	MVCC	/
OLAP	HBase	HBasechainDB	Research prototype	/	NoSQL Databases	Scalable big data store based on the concept of Blockchain	Java	MVCC	Tuple-at-a-time model
OLAP	TiesDB	TiesDB	In use	Private	Permissionless Blockchains	TiesDB uses blockchain technology to enhance their security	Solidify, NodeJS	/	/
HTAP	CockroachDB	Tierion	Prototype	/	NewSQL Databases	Company turning blockhain into global platform for veryfying data	SQL, NodeJS	MVCC	Tuple-at-a-time, Vector-at-a-time model
HTAP	FlureeDB	FlureeDB	In use	Private & Public	Out-of-the Database Blockchains	Scalable graph database which provides the benefits of blockchain technologies, such as immutability, replayability and fault tolerance	GraphQL, SPARQL, FlureeQL	/	/
HTAP	MongoDB	MongoDB Atlas	In use	/	NoSQL Databases	Enterprise grade blockchain database on top of MongoDB	SQL, JavaScript, NodeJS, Python	2 PL	Tuple-at-a-time model
HTAP	MongoDB	BigchainDB	In use	Private & Public	Out-of-the Database Blockchains	Blockchain characteristics on top of distributed database	RazorSQL, Java, Python, JavaScript, NodeJS	2 PL	Tuple-at-a-time model
HTAP	QLDB	QLDB	In use	/	Quantum Ledger Databases	Purpose-built ledger database that provides a complete and cryptographically verifiable history of all changes made to the application data	PartiQL, Java, .NET, NodeJS, Go, Python	Optimistic CC	/

Here, '*' in the column Original DB means that any database can be used, while '/' means that there is no information for that property. OLTP (Online Transaction Processing), OLAP (Online Transaction Processing), HTAP (Hybrid Transactional/Analytical Processing)

proven consensus algorithms.

The absence of specific properties renders the systems unfit for blockchain. One such property is sharding, which maintains consistency. The support for sharding while data consistency is not maintained categorically places the systems into the low-fit category. The absence of immutability prevents the systems from being considered a good fit for implementing blockchain.

Table 2 shows that even well-known commercial solutions, such as the Oracle Database and Microsoft SQL Server, are suitable for implementing blockchain. Other database systems that stand out are BigchainDB, BlockchainDB, DynamoDB, Elasticsearch, FlureeDB, HBasechainDB, and ScyllaDB.

While many databases are fit to be adapted for usage in blockchain technology, there are still challenges when trying to adapt databases into the blockchain ecosystem. One of the challenges is scaling blockchains by optimizing the consensus protocols, which are addressed by the authors in Refs. [106,107]. In addition, the authors in Ref. [107] explained that all the different stages that a transaction goes through before it is considered committed to the blockchain could also be a potential bottleneck and be subject to future optimizations.

However, there is a very close similarity between the flow of a transaction in a distributed database and the transaction flow in the blockchain system. In fact, the most significant difference is the consensus protocol. Because of the close similarity between the database technologies and the blockchain systems, the authors in Ref. [107] proposed four different approaches inspired by the database design that would improve blockchain performances, as listed below. We also add replication to the list due to the closely entangled replication approaches in blockchain and distributed databases:

- **Decoupling the layers and optimizing them individually.** This approach would decouple the storage, execution engine, and consensus layer from each other and proceed to optimize and scale them independently.
- **Embracing new hardware primitives.** This approach would take hardware optimized for data processing systems, as explained in Refs. [108–110].
- **Sharding.** This approach would use the sharding technique of distributed databases while still maintaining consistency.
- **Supporting declarative language.** This approach would use a set of high-level operations that can be composed in a declarative manner, making it easy to define complex smart contracts.
- **Replication.** Some decentralized database systems use non-standard and optimized data replication strategies, slightly sacrificing data availability for the sake of reducing the replication bottleneck. Usually, this is done by reducing the number of transactions replicated using the shared logs. This can be done by replicating to a smaller group of trusted replicas (managers) that periodically synchronize with the main manager or replica while maintaining a relatively satisfying data availability.

While all these approaches for optimizing blockchain by adapting database concepts are only defined theoretically, many databases are adapted and optimized for usage in the blockchain ecosystem. Advances in sharding and replication have contributed significantly to improving the performance and scalability of databases. The different blockchain databases, their original database, their general category, their maturity, their blockchain type, architecture type, their replication and architecture details, the supported implementation and development languages, the concurrency control mechanism, and the query processing mechanism are given in Table 3. The general category is divided into 1) Online Transaction Processing (OLTP), 2) Online Analytical Processing (OLAP), and 3) Hybrid Transactional/Analytical Processing (HTAP).

The databases in Table 3 are categorized into six categories derived from the type of database architecture. The table provides the information currently available for each database across the literature and the internet. Out-of-the-database blockchains are a hybrid design approach that starts with a database and then adds blockchain features to it. Out-of-the-blockchain databases are another hybrid approach that starts with a blockchain-like system and adds database features on top of it [98]. NewSQL databases are a category of databases following a design approach that starts with NoSQL databases in order to provide the scalability they offer but, at the same time, maintain the ACID properties of traditional RDBMS.

The concurrency control mechanisms given in Table 3 are used to ensure the correctness of the results of concurrent operations or transactions in the database. Major methods used in databases are optimistic timestamp ordering and pessimistic Two-Phase Locking (2 PL), as well as Two-Phase Commit (2 PC). However, Multi-Version Concurrency Control (MVCC) is most widely used today since it is considered optimal for mixed workloads and has been used for accelerating analytical processing [111]. Query processing is generally categorized as Tuple-at-a-time, Operator-at-a-time, or Vector-at-a-time. Tuple-at-a-time is used by most database management systems, and the query plan is executed such that each operator calls **next** on their child to get the next tuple to process. Operator-at-a-time is ideal for in-memory OLTP database engines because it materializes the entire output of the parent operator, resulting in a reduced number of function calls and tuples per operator. The Vector-at-a-time approach processes query plans such that each operator calls **next** on their child to obtain the next vector (batch) of data to process.

PostgreSQL has a chainified variant that is still a research prototype: it is a decentralized and replicated relational database with blockchain properties that implements a permissioned blockchain and uses the Postgres storage manager. PostgreSQL was chainified by implementing two new components, i.e., background workers: communication middleware to communicate with other nodes and block a processor to process blocks. It uses additional shared memory data structures and blockchain-related catalog tables. Currently, PostgreSQL's blockchain relational database supports blockchain features such as immutability, decentralization, and data replication but is still a research prototype.

OurSQL is a prototype for a blockchain replication tool, implemented on top of the MySQL relational database, and implements a private blockchain. MySQL is chainified into an OurSQL by adding a replication middleware acting as a blockchain replication tool on top of MySQL. It uses PoW consensus.

CouchDB's blockchain counterpart is Hyperledger Fabric, which is a state database for storing chaincode processed transaction data as key-value pairs. It implements a private blockchain, uses a B-tree-based storage engine, and supports transient and persistent replication.

There are two chainified variations of SQLite. The first, Aergolite, is a replicated SQLite database secured by a private and lightweight blockchain. As a framework, Aergolite was implemented using a bottom-up approach as a replicated transactional database with blockchain on top of the base database. It uses VRF-based consensus and implements absolute finality, eliminating the need for nodes to keep and verify all the history of transactions and blocks. The second, CovenantSQL, is a decentralized SQL database management system derived from SQLite and built on top of a public blockchain. As a framework, it was implemented using a top-down approach in which the database functionalities are added on top of the blockchain.

Postchain is a natively chainified database. It combines the features of a mature distributed database and public and private blockchains, along with the redundancy of replicated data. Its core defines interfaces that help with code organization and enable interoperability between different modules; its base provides tools for building custom blockchains, particularly enterprise private/federated blockchains. Here, GTX defines a Generic Transaction Format, and Postchain uses the EBFT (efficient BFT) consensus protocol inspired by Castro's PBFT but works on the block level and restricts concurrency as much as possible.

ChainifyDB is a blockchain-characteristic layer on top of a database layer, implementing permissioned blockchain. It is a natively chainified

database that upgrades existing infrastructures consisting of several database management systems by adding a layer of blockchain functionalities on top of them.

TiesDB is a fully chainifed database that is actively in use and uses blockchain technology to enhance security, and permissions are stored in a blockchain. Nodes and users are registered in the blockchain, and node owners make security deposits to protect the network from malicious behavior. Mutual settlements between users and nodes are also performed via blockchain.

HBase is a scalable big data store based on the concept of blockchain. It uses an HDFS storage system and was chainified in a research prototype called HBasechainDB that adds the blockchain characteristics of immutability and decentralization to the HBase database.

CockroachDB is chainified in a system called Tierion that aims to turn blockchain into a global platform for verifying data. This system is still only a prototype.

FlureeDB is a scalable graph database that provides the benefits of blockchain technologies, such as immutability, replayability, and fault tolerance. It implements both private and public blockchains and is an actively used enterprise blockchain-based database solution that combines blockchain's security, immutability, decentralization, and distributed ledger capabilities with a feature-rich graph-style database.

QLDB is a purpose-built ledger database that provides a complete and cryptographic verifiable history of all changes made to the application data. In particular, it is a fully managed ledger database that provides a transparent, immutable, and cryptographic verifiable transaction log.

MongoDB has two active chainified variations: MongoDB Atlas and BigchainDB. MongoDB Atlas is an enterprise-grade blockchain database implemented on top of MongoDB. On the one hand, MongoDB Atlas is a blockchain-enabled MongoDB that wraps the core database (MongoDB) and implements the three blockchain characteristics of decentralization, immutability, and assets. On the other hand, BigchainDB implements blockchain characteristics on top of a distributed database. It is a big data distributed database that adds the following blockchain characteristics on top of MongoDB: decentralized control, immutability, and the transfer of digital assets.

Enterprise cloud service providers have also begun offering blockchain capabilities to their customers to comply with the recent trends of moving to the cloud. Enterprise-grade cloud-hosted systems offer fully integrated blockchain solutions. IBM supports Hyperledger Fabric nodes and advertises its blockchain solution as enabling trusted data exchange and workflow automation with a distributed ledger. Amazon's AWS offers a full enterprise blockchain solution and provides partnership-based support for all major blockchain protocols, including Hyperledger Sawtooth, Ethereum, Quorum, Kadena ScalableBFT, and others. Other cloud service providers, such as Microsoft Azure and HP, also offer blockchain solutions. One aspect that has a stake in most cloud-based systems is the lack of decentralization. They are inherently scalable, and most of them provide the means to ensure consistency (immediate or eventual). However, they are not decentralized. At the time of writing, there have been signs of cloud decentralization initiatives by leveraging blockchain. By contrast, blockchain systems in the most widely used cloud platforms are not decentralized.

All the different chainified databases have advantages and disadvantages in their architecture and implementation and should be used based on the use case. However, many chainified databases take a database layer and put the blockchain characteristics on top of the database layer, whether a relational or distributed database is used. On the other hand, some databases do not add the blockchain characteristics on top of the storage system but rather incorporate characteristics of the blockchain directly in the database, as shown by FlureeDB. While some of the systems are still prototypes, their implementations imply that merging blockchain and database technologies provides the systems with extraordinary performances.

5. Discussion

This section first presents a discussion about advancements in blockchain databases and distributed file storage systems. Further, it discusses the implementation and challenges in adding database-related functionality in blockchains and adding blockchain-related functionality in databases. Finally, a short description with some future research directions about DCS is presented.

Many of the blockchain databases discussed in Section 3 have SQL capabilities. Nevertheless, a few of the databases are only NoSQL databases, which are of different types, such as key-value, document, or graph stores. These NoSQL databases are fast and do not require any fixed table schema. SQL databases cannot be truly distributed due to the restrictions of the CAP theorem [112]; hence, most of the blockchain databases are NoSQL. To make the distributed database, the NoSQL database sacrifices consistency over availability but reaches eventual consistency.

As discussed above, the databases in blockchain are either SQL or NoSQL; therefore, a new blockchain database application platform can be designed to support both SQL and NoSQL databases. It can have the decentralized, distributed, and audibility features of the blockchain, quick query processing, and a well-designed data structure of the distributed databases. An open-source system, CHAINSQL [113], supports both SQL and NoSQL databases and integrates the database system with the blockchain network. It is equipped with a mechanism for auditable transaction logs and means for safe and cost-effective recovery backup. In addition, it has a consensus mechanism supporting high throughput and fast validation time. Therefore, although many available database systems can be used in blockchain, the construction of new blockchain database systems supporting SQL and NoSQL capabilities would be an interesting direction to work.

Although blockchain employs traditional databases for storage, which are discussed in Section 3, there are other storage systems used in different systems, such as file storage or cloud storage systems. It would be interesting to study the similarities and relations of blockchain databases with other storage systems.

Speaking about storage systems, Decentralized File Storage (DFS) systems have been showing rapid growth recently. DFS is a P2P file storage system that allows the sharing of files on different systems. The concept is based on BitTorrent and a distributed hash table[13]. The new IPFS [35] is a decentralized and secure solution to store files in a distributed way. It gives high throughput and low latency. However, there are a few limitations in IPFS that can be lifted using the decentralized cloud file storage systems. These storage systems are similar to cloud storage but differ in storing the file on users' systems rather than in cloud data servers. These storage systems are fast, highly reliable, and have a huge capacity for data storage. There are many projects on decentralized cloud file storage systems such as Storj [9], Sia [11], and Ethereum Swarm[14]. An extensive analysis of the available DFS systems can be an interesting research direction to study their co-relation with traditional databases, their applicability in blockchain, and their pros and cons.

Adding database-related functionalities to blockchain systems. There have been many advancements to provide efficient database-related functionalities in the blockchain. The umbrella of these functionalities is enormous and can be studied and scrutinized rigorously. Verifiable efficient data query capabilities, such as those provided by vChain [114] and VQL [99], can be further studied and applied in blockchain. Researching the best suitable databases for decentralized apps (dApps), such as OrbitDB [115], can be an interesting area to explore. Furthermore, choosing an appropriate database for a blockchain project is a challenging problem. Many factors need to be considered while deciding on the perfect fit for a blockchain project. The major

[13] https://en.wikipedia.org/wiki/Distributed_hash_table.
[14] https://ethersphere.github.io/swarm-home/.

factors to consider while choosing the best fit for the database involve security, performance, fault tolerance, cost support, and query capabilities. Therefore, another interesting direction of work can be to explore all possible factors involved in making the right choice of database for the blockchain project.

Adding blockchain-related functionalities to database systems. To implement a blockchain functionality-enabled database, there must be some factors and challenges that are needed to be considered and addressed. These challenges are handling the low throughput and high latency of the blockchain database, addressing the scalability issue when more nodes are added to the network, and enabling querying capabilities in the blockchain database. Next, while designing a blockchain database is the parties involved in the decentralized network, whether it is inside an enterprise or a consortium of individuals or companies. Furthermore, the database can be set to operational and non-operational, wherein in an operational mode, a client can directly access the query response from the database; however, in a non-operational mode, intermediaries are set to provide a response for the client queries. All these things should be considered while designing a blockchain database. Therefore, a research direction would be to investigate all the major factors to be considered while designing a blockchain database.

Discussion on the DCS theorem. Regarding the DCS Theorem, Section 2 gives a brief overview of DCS; however, a lot more research is required to get around the DCS triangle. In Section 2, we presented a few ways to make the systems like DCS, but it would be interesting to explore more ways to provide all three properties. More research is required to dig into the available methods that can help achieve the DCS system or new methods or requirements that can be constructed or presented to achieve the goal. It would be interesting to argue about the degree of consistency, such as eventual consistency or full consistency. Eventual consistency means that all the nodes see the same data eventually (e.g., in Bitcoin), and full consistency means that all the nodes see the same data all the time. A similar discussion goes for the degree of decentralization, such as server-based or server-free decentralization. Server-based decentralization means that a node can become a validator only when it is voted by an existing federation; however, in server-free decentralization, any node having enough computation power can become a validator in the system. More discussion and research are required to define and assess the degree of DCS properties in the blockchain system.

6. Conclusions

The last decade was a decade of intense interchanged and mutually influenced the development of database and blockchain technologies. This paper is the first systematized and comprehensive study of these development trends. We provided a tabular rating of many database systems concerning their blockchain compatibility based on seven baseline criteria: decentralization, consistency, scalability, immutability, low latency, high throughput, and sharding. We provided a detailed summary of traditional databases that are being used or can be used in the design of blockchain platforms or applications. Further, we presented a detailed explanation of different decentralized solutions that use underlying traditional databases to deliver blockchain-enabled solutions. We also discussed the DCS theorem, an analogy for the CAP theorem for traditional databases, and postulated an analogous DCS-satisfiability conjecture.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

All authors approved the version of the manuscript to be published. This work is a product of collaboration among the authors.

References

[1] S. Nakamoto, Bitcoin: a peer-to-peer electronic cash system, URL, https://bitcoin.org/bitcoin.pdf, 2019. (Accessed 16 December 2021).
[2] CoinMarketCap, Total market capitalization. https://coinmarketcap.com, 2021. (Accessed 16 December 2021).
[3] M. Nofer, P. Gomber, O. Hinz, D. Schiereck, Blockchain, Bus. Inf. Syst. Eng. 59 (2017) 183–187, https://doi.org/10.1007/s12599-017-0467-3.
[4] M. Raikwar, D. Gligoroski, K. Kralevska, SoK of used cryptography in blockchain, IEEE Access 7 (2019) 148550–148575, https://doi.org/10.1109/ACCESS.2019.2946983.
[5] Z. Zheng, S. Xie, H.-N. Dai, X. Chen, H. Wang, Blockchain challenges and opportunities: a survey, Int. J. Web Grid Serv. 14 (2018) 352–375, https://doi.org/10.1504/IJWGS.2018.10016848.
[6] T. McConaghy, R. Marques, A. Müller, et al., BigchainDB: a scalable blockchain database. https://git.berlin/bigchaindb/site/raw/commit/b2d98401b65175f0fe0c169932ddca0b98a456a6/_src/whitepaper/bigchaindb-whitepaper.pdf, 2016. (Accessed 16 December 2021).
[7] M.J.M. Chowdhury, A. Colman, M.A. Kabir, J. Han, P. Sarda, Blockchain versus database: a critical analysis, in: Proceedings of the 2018 17th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/12th IEEE International Conference on Big Data Science and Engineering (TrustCom/BigDataSE), IEEE, 2018, pp. 1348–1353, https://doi.org/10.1109/TrustCom/BigDataSE.2018.00186.
[8] M. Raikwar, D. Gligoroski, G. Velinov, Trends in development of databases and blockchain, in: Proceedings of the 2020 Seventh International Conference on Software Defined Systems (SDS), IEEE, 2020, pp. 177–182, https://doi.org/10.1109/SDS49854.2020.9143893.
[9] S. Wilkinson, T. Boshevski, J. Brandoff, V. Buterin, Storj A Peer-To-Peer Cloud Storage Network, 2014. URL, https://www.storj.io/storj2014.pdf. (Accessed 16 December 2021).
[10] Y. Psaras, D. Dias, The InterPlanetary file system and the filecoin network, in: Proceedings of the 2020 50th Annual IEEE-IFIP International Conference on Dependable Systems and Networks-Supplemental Volume (DSN-S), IEEE, 2020, https://doi.org/10.1109/DSN-S50200.2020.00043, 80–80.
[11] D. Vorick, L. Champine, Sia: simple decentralized storage, URL, https://blockchainlab.com/pdf/whitepaper3.pdf, 2014 (Accessed 16 December 2021).
[12] D. Loghin, The anatomy of blockchain database systems, URL, http://sites.computer.org/debull/A22june/p48.pdf, 2022 (Accessed 16 November 2022).
[13] Z. Ge, D. Loghin, B.C. Ooi, P. Ruan, T. Wang, Hybrid blockchain database systems: design and performance, Proc. VLDB Endowm. 15 (5) (2022) 1092–1104, https://doi.org/10.14778/3510397.3510406.
[14] X. Yang, Y. Zhang, S. Wang, B. Yu, F. Li, Y. Li, W. Yan, LedgerDB: a centralized ledger database for universal audit and verification, Proc. VLDB Endowm. 13 (12) (2020) 3138–3151, https://doi.org/10.14778/3415478.3415540.
[15] Y. Peng, M. Du, F. Li, R. Cheng, D. Song, FalconDB: blockchain-based collaborative database, in: Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data, 2020, pp. 637–652, https://doi.org/10.1145/3318464.3380594.
[16] L. Allen, P. Antonopoulos, A. Arasu, J. Gehrke, J. Hammer, J. Hunter, R. Kaushik, D. Kossmann, J. Lee, R. Ramamurthy, et al., Veritas: shared verifiable databases and tables in the cloud, in: Proceedings of the 9th Biennial Conference on Innovative Data Systems Research (CIDR), CIDRDB, 2019.
[17] F.M. Schuhknecht, A. Sharma, J. Dittrich, D. Agrawal, ChainifyDB: How to Blockchainify Any Data Management System, 2019 arXiv:1912.04820.
[18] E.A. Brewer, Towards robust distributed systems, in: Proceedings of the Nineteenth Annual ACM Symposium on Principles of Distributed Computing, ACM, 2000, https://doi.org/10.1145/343477.343502.
[19] M. Wiesmann, F. Pedone, A. Schiper, B. Kemme, G. Alonso, Understanding replication in databases and distributed systems, in: Proceedings of the 20th IEEE International Conference on Distributed Computing Systems, IEEE, 2000, pp. 464–474, https://doi.org/10.1109/ICDCS.2000.840959.
[20] S. Gilbert, N. Lynch, Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services, ACM SIGACT News 33 (2) (2002) 51–59, https://doi.org/10.1145/564585.564601.
[21] W. Vogels, Eventually consistent, Commun. ACM 52 (1) (2009) 40–44, https://doi.org/10.1145/1435417.1435432.
[22] E. Brewer, CAP twelve years later: how the "rules" have changed, Computer 45 (2) (2012) 23–29, https://doi.org/10.1109/MC.2012.37.
[23] I. Weber, V. Gramoli, A. Ponomarev, M. Staples, R. Holz, A.B. Tran, P. Rimba, On availability for blockchain-based systems, in: Proceedings of 2017 IEEE 36th Symposium on Reliable Distributed Systems (SRDS), IEEE, 2017, pp. 64–73, https://doi.org/10.1109/SRDS.2017.15.
[24] MangoDB, Building enterprise-grade blockchain databases with MongoDB, White paper URL, https://www.ashnik.com/whitepaper/mongodb/building-enterprise-grade-blockchain-databases-with-mongodb/. (Accessed 16 December 2021).
[25] Mehul Nalin Vora, Hadoop-HBase for large-scale data, in: Proceedings of 2011 International Conference on Computer Science and Network Technology, vol. 1, IEEE, 2011, pp. 601–605, https://doi.org/10.1109/ICCSNT.2011.6182030.
[26] J.L. Carlson, Redis in Action, Manning Shelter Island, Shelter Island, NY, USA, 2013.

[27] A. Lakshman, P. Malik, Cassandra: a decentralized structured storage system, SIGOPS Oper. Syst. Rev. 44 (2) (2010) 35–40, https://doi.org/10.1145/1773912.1773922.

[28] S. Sivasubramanian, Amazon dynamoDB: a seamlessly scalable non-relational database service, in: Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data, ACM, 2012, pp. 729–730, https://doi.org/10.1145/2213836.2213945.

[29] J.C. Anderson, J. Lehnardt, N. Slater, CouchDB: the Definitive Guide: Time to Relax, O'Reilly Media, Inc., Sebastopol, CA, USA, 2010.

[30] D. Abadi, Consistency tradeoffs in modern distributed database system design: CAP is only part of the story, Computer 45 (2) (2012) 37–42, https://doi.org/10.1109/MC.2012.33.

[31] G. Slepak, A. Petrova, The DCS Theorem, 2018 arXiv preprint arXiv:1801.04335.

[32] K. Zhang, H.-A. Jacobsen, Towards dependable, scalable, and pervasive distributed ledgers with blockchains, in: Proceedings of 2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS), IEEE, 2018, pp. 1337–1346, https://doi.org/10.1109/ICDCS.2018.00134.

[33] G. Wood, Ethereum: a secure decentralised generalised transaction ledger, Yellow Paper, 2014. URL, https://ethereum.github.io/yellowpaper/paper.pdf.

[34] J. Benet, IPFS - Content Addressed, Versioned, P2P File System, 2014 arXiv:1407.3561.

[35] MultiChain, URL, https://www.multichain.com/, 2015. (Accessed 16 December 2021).

[36] Y. Sompolinsky, A. Zohar, Secure high-rate transaction processing in bitcoin, in: R. Böhme, T. Okamoto (Eds.), Financial Cryptography and Data Security, Springer, Berlin, Heidelberg, 2015, pp. 507–527, https://doi.org/10.1007/978-3-662-47854-7_32.

[37] E. Androulaki, A. Barger, V. Bortnikov, et al., Hyperledger fabric: a distributed operating system for permissioned blockchains, in: Proceedings of the Thirteenth EuroSys Conference, ACM, 2018, pp. 1–15, https://doi.org/10.1145/3190508.3190538.

[38] M. Castro, B. Liskov, et al., Practical byzantine fault tolerance, in: Proceedings of the Third Symposium on Operating Systems Design and Implementation, vol. 99, USENIX Association, 1999, pp. 173–186.

[39] C. Gorenflo, S. Lee, L. Golab, S. Keshav, FastFabric: scaling hyperledger fabric to 20 000 transactions per second, Int. J. Netw. Manag. 30 (5) (2020), e2099, https://doi.org/10.1002/nem.2099.

[40] J. Sousa, A. Bessani, M. Vukolic, A byzantine fault-tolerant ordering service for the hyperledger fabric blockchain platform, in: Proceedings of the 2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), IEEE, 2018, pp. 51–58, https://doi.org/10.1109/DSN.2018.00018.

[41] G.A. Marson, S. Andreina, L. Alluminio, K. Munichev, G. Karame, Mitosis: practically scaling permissioned blockchains, in: Proceedings of Annual Computer Security Applications Conference, ACM, 2021, pp. 773–783, https://doi.org/10.1145/3485832.3485915.

[42] M. Shapiro, N. Preguiça, C. Baquero, M. Zawirski, Conflict-free replicated data types, in: X. Défago, F. Petit, V. Villain (Eds.), Stabilization, Safety, and Security of Distributed Systems, Springer, Berlin, Heidelberg, 2011, pp. 386–400, https://doi.org/10.1007/978-3-642-24550-3_29.

[43] L. Gudgeon, P. Moreno-Sanchez, S. Roos, P. McCorry, A. Gervais, Sok: layer-two blockchain protocols, in: J. Bonneau, N. Heninger (Eds.), Financial Cryptography and Data Security, Springer, Cham, 2020, pp. 201–226, https://doi.org/10.1007/978-3-030-51280-4_12.

[44] J. Poon, T. Dryja, The bitcoin lightning network: scalable off-chain instant payments, https://www.bitcoinlightning.com/wp-content/uploads/2018/03/lightning-network-paper.pdf, 2016. (Accessed 16 December 2021).

[45] G. Wang, Z.J. Shi, M. Nixon, S. Han, Sok: sharding on blockchain, in: Proceedings of the 1st ACM Conference on Advances in Financial Technologies, ACM, 2019, pp. 41–61, https://doi.org/10.1145/3318041.3355457.

[46] H. Wang, Y. Cen, X. Li, Blockchain router: a cross-chain communication protocol, in: Proceedings of the 6th International Conference on Informatics, Environment, Energy and Applications, ACM, 2017, pp. 94–97, https://doi.org/10.1145/3070617.3070634.

[47] B. Pillai, K. Biswas, V. Muthukkumarasamy, Cross-chain interoperability among blockchain-based systems using transactions, Knowl. Eng. Rev. 35 (2020) E23, https://doi.org/10.1017/S0269888920000314.

[48] S. Benbernou, M. Ouziri, Smart contract satisfiability checking for blockchain consistency, in: M. Aiello, A. Bouguettaya, D.A. Tamburri, W.-J. van den Heuvel (Eds.), Next-Gen Digital Services. A Retrospective and Roadmap for Service Computing of the Future, Springer, Cham, 2021, pp. 264–272, https://doi.org/10.1007/978-3-030-73203-5_20.

[49] T. Osterland, T. Rose, Correctness of smart contracts for consistency enforcement, ERCIM News 110 (2017) (June 2017) 30.

[50] V. Grishchenko, Distributed digital currency as an RDT, URL, https://gritzko.gitbooks.io/swarm-the-protocol/content/coin.html, 2011. (Accessed 16 December 2021).

[51] J. Clow, Z. Jiang, A Byzantine Fault tolerant raft, URL, https://www.scs.stanford.edu/17au-cs244b/labs/projects/clow_jiang.pdf, 2017. (Accessed 16 December 2021).

[52] D. Ongaro, J. Ousterhout, In search of an understandable consensus algorithm, in: Proceedings of the 2014 USENIX Annual Technical Conference, USENIX Association, 2014, pp. 305–319.

[53] K. Delaney, Inside Microsoft SQL Server 2000, 3rd edn., Microsoft Press, Redmond, WA, USA, 2000.

[54] J. Lee, G. Malcolm, A. Matthews, R. Negrin, Z. Owens, D. Robinson, Overview of Microsoft SQL azure database, Microsoft Technical Whitepaper URL, http://old.geninf.com/softwaredevelopment/azure_sql/sql_whitepaper.pdf.

[55] MariaDB Foundation, MariaDB 10.5.0 now available, URL, https://mariadb.org/mariadb-10-5-0-now-available/, 2019.

[56] Oracle Database 21c, 2022. URL, https://docs.oracle.com/en/database/index.html (Accessed 16 November 2022).

[57] Postgresql V14, 2022. URL, https://www.postgresql.org (Accessed 16 November 2022).

[58] S. Nathan, C. Govindarajan, A. Saraf, M. Sethi, P. Jayachandran, Blockchain Meets Database: Design and Implementation of a Blockchain Relational Database, 2019 arXiv:1903.01919.

[59] Hipp, Wyrick & Company, Inc., SQLite, URL, https://www.sqlite.org/, 2020 (Accessed 16 December 2021).

[60] M. Stonebraker, A. Weisberg, The VoltDB main memory DBMS, IEEE Data Eng. Bull. 36 (2) (2013) 21–27.

[61] K. Shvachko, H. Kuang, S. Radia, R. Chansler, The Hadoop distributed file system, in: Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST), IEEE, 2010, pp. 1–10, https://doi.org/10.1109/MSST.2010.5496972.

[62] L.K. Yeddu, S. Yeddu, Hadoop: bitcoin-BlockChain-A new era needed in distributed computing, Int. J. Comput. Appl. 154 (6) (2016) 13–19.

[63] M.A. Krishnan, C.G. Shankar, S.A. Raj, A. Ragavan, Peer to peer file sharing by blockchain using IoT, Int. J. Sci. Res. Sci. Eng. Technol. 3 (2) (2017) 91–94.

[64] D.S. Kumar, M.A. Rahman, Simplified HDFS architecture with blockchain distribution of metadata, Int. J. Appl. Eng. Res. 12 (21) (2017) 11374–11382.

[65] The Apache Software Foundation, Welcome to Apache HBase™, URL, https://hbase.apache.org/, 2020 (Accessed 16 December 2021).

[66] M. Zaharia, R.S. Xin, P. Wendell, T. Das, M. Armbrust, A. Dave, X. Meng, J. Rosen, S. Venkataraman, M.J. Franklin, et al., Apache spark: a unified engine for big data processing, Commun. ACM 59 (11) (2016) 56–65, https://doi.org/10.1145/2934664.

[67] T. Lipcon, D. Alves, D. Burkert, J.-D. Cryans, A. Dembo, M. Percy, S. Rus, D. Wang, M. Bertozzi, C.P. McCabe, et al., Kudu: storage for fast analytics on fast data, URL, https://kudu.apache.org/kudu.pdf, 2015 (Accessed 16 December 2021).

[68] Building hybrid blockchain/cloud applications with Ethereum and Google Cloud, URL, https://cloud.google.com/blog/products/data-analytics/building-hybrid-blockchain-cloud-applications-with-ethereum-and-google-cloud, 2019 (Accessed 16 December 2021).

[69] Analysis of the blockchain, or why the mixer broke?, URL, https://sudonull.com/post/14859-Analysis-of-the-blockchain-or-why-the-mixer-broke, 2018 (Accessed 16 December 2021).

[70] M. Chanson, A. Bogner, F. Wortmann, E. Fleisch, Blockchain as a privacy enabler: an odometer fraud prevention system, in: Proceedings of the 2017 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2017 ACM International Symposium on Wearable Computers, ACM, 2017, pp. 13–16, https://doi.org/10.1145/3123024.3123078.

[71] R.C. Celiz, Y.E. De La Cruz, D.M. Sanchez, Cloud model for purchase management in health sector of Peru based on iot and blockchain, in: Proceedings of the 2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), IEEE, 2018, pp. 328–334, https://doi.org/10.1109/IEMCON.2018.8615063.

[72] Get Started with Blockchain Using the New AWS Blockchain Templates, 2018. URL, https://aws.amazon.com/blogs/aws/get-started-with-blockchain-using-the-new-aws-blockchain-templates/ (Accessed 16 December 2021).

[73] C. Gormley, Z. Tong, Elasticsearch: the Definitive Guide: a Distributed Real-Time Search and Analytics Engine, O'Reilly Media, Inc., Sebastopol, CA, USA, 2015.

[74] Observability on Blockchain and the Hyperledger Project, 2019. URL, https://www.elastic.co/blog/observability-on-blockchain-and-the-hyperledger-project (Accessed 16 December 2021).

[75] IOTA, Using Scylla for Distributed Storage of the Tangle, 2020. URL, https://www.scylladb.com/2020/08/13/iota-using-scylla-for-distributed-storage-of-the-tangle/ (Accessed 16 December 2021).

[76] D. K. Anton Filatov, TIES.NETWORK, Database Management System, Technical description, Yellow paper (2017).

[77] How the Ties.DB Database Uses Blockchain Technology, 2018. URL, https://insidebigdata.com/2018/03/04/ties-db-database-uses-blockchain-technology/ (Accessed 16 December 2021).

[78] V. Srinivasan, B. Bulkowski, W.-L. Chu, S. Sayyaparaju, A. Gooding, R. Iyer, A. Shinde, T. Lopatic, Aerospike: architecture of a real-time operational dbms, Proc. VLDB Endowm. 9 (13) (2016) 1389–1400, https://doi.org/10.14778/3007263.3007276.

[79] J.R. Guay Paz, Introduction to Azure Cosmos DB, Apress, Berkeley, CA, 2018, pp. 1–23, https://doi.org/10.1007/978-1-4842-3351-1_1.

[80] CockroachDB, An Evolution of the Database, 2017. URL, https://www.cockroachlabs.com. (Accessed 16 December 2021).

[81] F.M. Waas, Beyond conventional data warehousing—massively parallel data processing with greenplum database, in: M. Castellanos, U. Dayal, T. Sellis (Eds.), Business Intelligence for the Real-Time Enterprise, Springer, Berlin, Heidelberg, 2008, pp. 89–96, https://doi.org/10.1007/978-3-642-03422-0_7.

[82] C. Tesoriero, Getting Started with orientDB, Packt Publishing Ltd, Birmingham, UK, 2013.

[83] A. Stanciu, Blockchain based distributed control system for edge computing, in: Proceedings of the 2017 21st International Conference on Control Systems and Computer Science (CSCS), IEEE, 2017, pp. 667–671, https://doi.org/10.1109/CSCS.2017.102.

[84] M.U. Wasim, A.A. Ibrahim, P. Bouvry, T. Limba, Law as a service (LaaS): enabling legal protection over a blockchain network, in: Proceedings of the 2017 14th International Conference on Smart Cities: Improving Quality of Life Using ICT & IoT (HONET-ICT), IEEE, 2017, pp. 110–114, https://doi.org/10.1109/HONET.2017.8102214.
[85] TiDB, SQL at scale. https://pingcap.com/en/, 2019. (Accessed 16 December 2021).
[86] L. Walsh, V. Akhmechet, M. Glukhovsky, Rethinkdb-rethinking database storage, URL, https://pdfs.semanticscholar.org/3cdf/b12ceee0f82a08b352cead0bf791477dca98.pdf, 2009. (Accessed 16 December 2021).
[87] J. Kwon, Tendermint: consensus without mining, 2014 URL, https://tendermint.com. (Accessed 16 December 2021).
[88] M. El-Hindi, M. Heyden, C. Binnig, R. Ramamurthy, A. Arasu, D. Kossmann, BlockchainDB - towards a shared database on blockchains, in: Proceedings of the 2019 International Conference on Management of Data, ACM, 2019, pp. 1905–1908, https://doi.org/10.1145/3299869.3320237.
[89] S. Helmer, M. Roggia, N.E. Ioini, C. Pahl, EthernityDB – integrating database functionality into a blockchain, in: A. Benczúr, B. Thalheim, T. Horváth, S. Chiusano, T. Cerquitelli, C. Sidló, P.Z. Revesz (Eds.), New Trends in Databases and Information Systems, Springer, Cham, 2018, pp. 37–44, https://doi.org/10.1007/978-3-030-00063-9_5.
[90] FlureeDB, URL, https://flur.ee, 2017. (Accessed 16 December 2021).
[91] M.S. Sahoo, P.K. Baruah, HBasechainDB – a scalable blockchain framework on Hadoop ecosystem, in: R. Yokota, W. Wu (Eds.), Supercomputing Frontiers, Springer, Cham, 2018, pp. 18–29, https://doi.org/10.1007/978-3-319-69953-0_2.
[92] Modex, Multiple databases support. https://modex.tech/wp-content/uploads/2020/10/Modex_Whitepaper.pdf, 2020. (Accessed 13 December 2021).
[93] J.M. Graglia, C. Mellon, Blockchain and property in 2018: at the end of the beginning, innovations: technology, governance, Globalizations 12 (1–2) (2018) 90–116, https://doi.org/10.1162/inov_a_00270.
[94] ProvenDB Team, ProvenDB, Trust Your Data, Yellow Paper. 2019.
[95] Amazon quantum ledger database, URL, https://aws.amazon.com/qldb/, 2019. (Accessed 16 December 2021).
[96] H.T. Kung, J.T. Robinson, On optimistic methods for concurrency control, ACM Trans. Database Syst. 6 (2) (1981) 213–226, https://doi.org/10.1145/319566.319567. ISSN 0362-5915.
[97] P.A. Bernstein, N. Goodman, Multiversion concurrency control—theory and algorithms, ACM Trans. Database Syst. 8 (4) (1983) 465–483, https://doi.org/10.1145/319996.319998.
[98] P. Ruan, T.T.A. Dinh, D. Loghin, M. Zhang, G. Chen, Q. Lin, B.C. Ooi, Blockchains vs. Distributed databases: dichotomy and fusion, in: Proceedings of the 2021 International Conference on Management of Data, ACM, 2021, pp. 1504–1517, https://doi.org/10.1145/3448016.3452789.
[99] Z. Peng, H. Wu, B. Xiao, S. Guo, VQL, Providing query efficiency and data authenticity in blockchain systems, in: Proceedings of the IEEE 35th International Conference on Data Engineering Workshops, ICDEW), 2019, pp. 1–6, https://doi.org/10.1109/ICDEW.2019.00-44.
[100] The Neo4j graph data platform, URL, https://neo4j.com/, 2022. (Accessed 16 November 2022).
[101] Dgraph database overview, URL, https://dgraph.io/docs/dgraph-overview/, 2022. (Accessed 16 November 2022).
[102] L. Lamport, The part-time parliament, ACM Trans. Comput. Syst. 16 (2) (1998) 133–169.
[103] A. Medeiros, ZooKeeper's Atomic Broadcast Protocol: Theory and Practice, Tech. Rep, 2012. Technical report, https://zdq0394.github.io/papers/zab.pdf (Accessed 16 December 2021).
[104] A. Paro, Elasticsearch 7.0 Cookbook: over 100 Recipes for Fast, Scalable, and Reliable Search for Your Enterprise, Packt Publishing Ltd, Birmingham, UK, 2019.
[105] L. Luu, V. Narayanan, C. Zheng, K. Baweja, S. Gilbert, P. Saxena, A secure sharding protocol for open blockchains, in: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, 2016, pp. 17–30, https://doi.org/10.1145/2976749.2978389.
[106] E.K. Kogias, P. Jovanovic, N. Gailly, I. Khoffi, L. Gasser, B. Ford, Enhancing bitcoin security and performance with strong consistency via collective signing, in: Proceedings of the 25th USENIX Security Symposium, USENIX Association, 2016, pp. 279–296.
[107] T.T.A. Dinh, R. Liu, M. Zhang, G. Chen, B.C. Ooi, J. Wang, Untangling blockchain: a data processing view of blockchain systems, IEEE Trans. Knowl. Data Eng. 30 (7) (2018) 1366–1385, https://doi.org/10.1109/TKDE.2017.2781227.
[108] K.-L. Tan, Q. Cai, B.C. Ooi, W.-F. Wong, C. Yao, H. Zhang, In-memory databases: challenges and opportunities from software and hardware perspectives, ACM SIGMOD Record 44 (2) (2015) 35–40, https://doi.org/10.1145/2814710.2814717.
[109] H. Zhang, G. Chen, B.C. Ooi, K.-L. Tan, M. Zhang, In-memory big data management and processing: a survey, IEEE Trans. Knowl. Data Eng. 27 (7) (2015) 1920–1948, https://doi.org/10.1109/TKDE.2015.2427795.
[110] A. Dragojević, D. Narayanan, E.B. Nightingale, M. Renzelmann, A. Shamis, A. Badam, M. Castro, No compromises: distributed transactions with consistency, availability, and performance, in: Proceedings of the 25th Symposium on Operating Systems Principles, ACM, 2015, pp. 54–70, https://doi.org/10.1145/2815400.2815425.
[111] A. Sharma, F.M. Schuhknecht, J. Dittrich, Accelerating analytical processing in mvcc using fine-granular high-frequency virtual snapshotting, in: Proceedings of the 2018 International Conference on Management of Data, ACM, 2018, pp. 245–258, https://doi.org/10.1145/3183713.3196904.
[112] E. Brewer, A certain freedom: thoughts on the CAP theorem, in: Proceedings of the 29th ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing, PODC '10, ACM, New York, NY, USA, 2010, https://doi.org/10.1145/1835698.1835701, 335–335.
[113] M. Muzammal, Q. Qu, B. Nasrulin, Renovating blockchain with distributed databases: an open source system, Future Generat. Comput. Syst. 90 (117) (2019) 105, https://doi.org/10.1016/j.future.2018.07.042. ISSN 0167-739X.
[114] C. Xu, C. Zhang, J. Xu, vChain: enabling verifiable boolean range queries over blockchain databases, in: Proceedings of the 2019 International Conference on Management of Data, 2019, pp. 141–158, https://doi.org/10.1145/3299869.3300083.
[115] OrbitDB, Peer-to-Peer databases for the decentralized web, URL, https://orbitdb.org, 2015. (Accessed 16 December 2021).