future internet

MDPI

Article

Query Processing in Blockchain Systems: Current State and Future Challenges

Dennis Przytarski [1,*], Christoph Stach [1], Clémentine Gritti [2] and Bernhard Mitschang [1]

[1] Institute for Parallel and Distributed Systems, University of Stuttgart, Universitätsstraße 38, 70569 Stuttgart, Germany; Christoph.Stach@ipvs.uni-stuttgart.de (C.S.); Bernhard.Mitschang@ipvs.uni-stuttgart.de (B.M.)
[2] Department of Computer Science and Software Engineering, University of Canterbury, Christchurch 8041, New Zealand; clementine.gritti@canterbury.ac.nz
* Correspondence: dennis.przytarski@ipvs.uni-stuttgart.de; Tel.: +49-711-68588-235

Abstract: When, in 2008, Satoshi Nakamoto envisioned the first distributed database management system that relied on cryptographically secured chain of blocks to store data in an immutable and tamper-resistant manner, his primary use case was the introduction of a digital currency. Owing to this use case, the blockchain system was geared towards efficient storage of data, whereas the processing of complex queries, such as provenance analyses of data history, is out of focus. The increasing use of Internet of Things technologies and the resulting digitization in many domains, however, have led to a plethora of novel use cases for a secure digital ledger. For instance, in the healthcare sector, blockchain systems are used for the secure storage and sharing of electronic health records, while the food industry applies such systems to enable a reliable food-chain traceability, e. g., to prove compliance with cold chains. In these application domains, however, querying the current state is not sufficient — comprehensive history queries are required instead. Due to these altered usage modes involving more complex query types, it is questionable whether today's blockchain systems are prepared for this type of usage and whether such queries can be processed efficiently by them. In our paper, we therefore investigate novel use cases for blockchain systems and elicit their requirements towards a data store in terms of query capabilities. We reflect the state of the art in terms of query support in blockchain systems and assess whether it is capable of meeting the requirements of such more sophisticated use cases. As a result, we identify future research challenges with regard to query processing in blockchain systems.

Keywords: blockchain systems; query processing; data models; data structures; block structures

check for updates

Citation: Przytarski, D.; Stach, C.; Gritti, C.; Mitschang, B. Query Processing in Blockchain Systems: Current State and Future Challenges. *Future Internet* **2022**, *14*, 1. https://doi.org/10.3390/fi14010001

Academic Editor: Luis Javier García Villalba

Received: 17 November 2021
Accepted: 13 December 2021
Published: 21 December 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Digitization fostered by the evolution of the Internet of Things (IoT) has made data one of the most important commodity in both business and private environments [1]. Data became the backbone for a variety of new data-driven application areas such as digital health [2], food supply chain [3], or the production of goods in Industry 4.0 [4]. All of these use cases have in common that they are permanently dependent on demand-driven data provisioning — i. e., the data generated and provided by several data producers must be made available to all data consumers in the required quality and quantity [5]. For this purpose, database systems are often used, as they significantly facilitate the management and provision of data [6].

However, due to the fact that data are nowadays highly valuable, they became attractive targets for cybercriminals who exploit these data in order to harm the involved parties. There is a wide variety of attack types, e. g., tampering with the data in these databases [7]. Detecting data tampering is nearly impossible without additional security measures, consequently being one of the most serious attacks to defend. Considering the worst-case scenario, where data are minimally tampered with, at stages that hardly arouse

This is the author's version of the work. It is posted at https://opencms.uni-stuttgart.de/fak5/ipvs/departments/as/publications/stachch/fi_22_blockchains.pdf for your personal use. Not for redistribution. The definitive version was published in *In: Giuli, D. and Hudson-Smith, A. and García Villalba, L. J. (Eds.) Future Internet, Volume 14, Number 1*, pp. 1–31, 2022, doi: 10.3390/fi14010001.

suspicion, can cause long-lasting damage to an organization [8]. These attacks are usually performed either by outsiders such as hackers, who are unaffiliated with the organization itself, or by malicious insiders such as untrustworthy database or system administrators. In traditional database systems, data are unprotected against this attack vector because they lack the necessary data integrity checks in the sense of ensuring that stored data are still in the same state as it was once inserted. Therefore, the recent emphasis lies on hardening these databases against data tampering [9].

Another problem is that attackers do not even have to attack the data itself to harm the involved parties. It is already enough to attempt to affect the availability of the data [10]. Denial-of-service attacks cause the database or the server on which it is running to become unreachable by flooding it with fake requests. While the server is occupied with processing the malicious requests, there are no more resources available for processing the legitimate requests, which is why they do not receive any feedback and thus, the data are no longer available to them. Another focus with regard to cybersecurity is, therefore, on strengthening availability to be prepared for the failure of resources [11].

Blockchains offer a solution to these two problems. Firstly, it is immutable and tamper-resistant, thus protected against data tampering. Secondly, it is decentralized and thus protected against denial-of-service attacks [12]. Yet, when Satoshi Nakamoto envisioned the first blockchain system for his digital currency *Bitcoin* [13], his priority was to solve the double spending problem, since there is no actual physical relinquishment in a digital currency. Therefore, many of the conveniences of traditional data management systems, such as a powerful query engine, are missing, i. e., they are much less convenient to use in terms of query language and query processing [14].

In this context, blockchains in particular offer many interesting additional use cases for queries due to their internal data management. In a blockchain, data are only appended, which results in the construction of a data log (i. e., blockchain data history) where different revisions of data coexist. This enables the possibility to query the data history for provenance analyses, unlike with a traditional database where data are modified in-place, which means that there is no natively existing data log to query [15]. The existence of this blockchain data history, however, means that applications are forced to store data externally to a blockchain and in many cases also need to perform additional query processing mostly local to the application.

This is why we investigate the necessary query capabilities for blockchain data histories. To this end, we provide three contributions in this paper:

1. Based on use cases from different application domains, we derive common types of usage of blockchain technologies in terms of types of data and queries.
2. For these types of data and queries, we investigate how they can be implemented in blockchain systems and how they can be supported by the available data history.
3. We explore the state of the art regarding query processing in blockchains and identify future research challenges.

By means of these three contributions, we identify open research gaps that need to be solved in order to enable efficient query processing in blockchain systems.

The remainder of this paper is structured as follows: We open by outlining the fundamentals of blockchain technologies in Section 2, with respect to their relevance in the context of this paper. In particular, our goal is to highlight the conceptual and architectural differences between blockchains and traditional database systems that are responsible for the challenges regarding efficient query processing. We then identify five emerging application domains in Section 3 where blockchains are becoming prevalent for data management. Based on a literature review, we identify types of data and queries that are relevant in these application domains. In Section 4, we generalize these types of data into two object types that must be distinguished when querying blockchains. Then, in Section 5, we determine for these two object types which query capabilities are required in blockchains to be efficiently usable in the application domains. In Section 6, we present the state of the art in research and discuss to which extent it provides these required query

capabilities. Subsequently, we identify future research challenges in Section 7 and conclude this paper in Section 8.

2. Fundamentals of Blockchain Technology

Before we can delve into queries to a blockchain system, we need to address a few fundamentals of blockchain technology that have an impact on query processing. Even though it is often referred to as "the blockchain", a blockchain is actually a modular assembly of different components. In general terms, a blockchain is a ledger of sequential blocks that contain arbitrary information. This ledger is managed by a network of computers. That is, the distinctive feature of the blockchain is not what can be done with it — i. e., the secure management of data — but rather how this can be accomplished in a decentralized manner on a trustless infrastructure. For this purpose, well-established technologies from different fields of information technology are used in a blockchain. A blockchain architecture therefore has a modular structure, consisting of at least three layers: ❶ Storage, ❷ Network, and ❸ Consensus. Each layer is freely configurable to the respective requirements from a variety of technology variants, with all their advantages and disadvantages [16]. Figure 1 shows this modular architecture. In the following, we discuss these layers in detail.

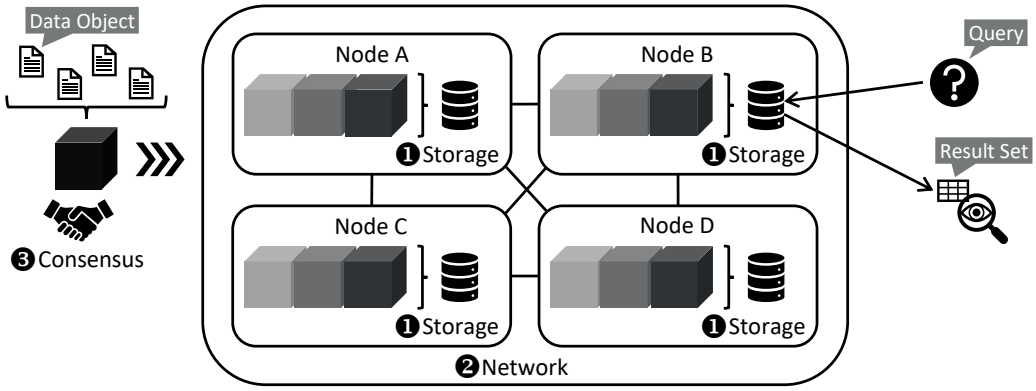


Figure 1. Simplified architecture of a blockchain system with its three layers: ❶ Storage, ❷ Network, and ❸ Consensus.

A blockchain is a list of blocks that are singly linked backwards using cryptographic signatures, with each block containing data. Backward linking is accomplished by including a header in a block that contains the hash value of its predecessor in addition to the actual payload data. A block cannot be modified subsequently, i. e., it is immutable. In particular, data and even entire blocks cannot be deleted retroactively due to this structure. In other words, a blockchain is an append-only data store. When new data are to be added to the blockchain, a new block must be created for this purpose, which is then appended to an existing blockchain [17].

There are many ways to manage a blockchain ❶. Usually, the data in a block are stored in a data structure that enables efficient verification of its integrity (e. g., *Merkle trees* [18], *Modified Merkle Patricia trees* [19]), and the blocks themselves are stored as a log-like structure on a storage device, with derived information stored in a state database for ease of access. The log is therefore mainly used to rebuild or verify the state database in case of problems [20].

Since data are never deleted from a blockchain, a blockchain automatically maintains a native data history. In contrast, a traditional database system must either manually implement the data history at the application layer (e. g., by implementing triggers to populate an audit trail table) or utilize specialized features like plugins for data history support (e. g., *Oracle Flashback Technology* (see https://www.oracle.com/database/technologies/high-availability/flashback.html; accessed on 15 December 2021) for *Oracle Databases* (see https://www.oracle.com/database/; accessed on 15 December 2021)) [21].

A blockchain is represented by multiple blockchain instances hosted on separate nodes in a distributed manner ❷. This replication approach increases availability and reliability [22].

In order to add new data to a blockchain, a new block must be created and announced to all nodes to become part of all blockchain instances. This distribution feature, however, leads to the possible situation where there could be competing blocks that are linked to the same predecessor and therefore cannot both be appended to the blockchain. To solve this, all nodes have agreed on a consensus mechanism ❸. This ensures that the network agrees on the next state of the blockchain, i. e., which block will be appended next to the blockchain. The consensus mechanism also defines, if a blockchain is permissionless or public, i. e., everyone can maintain a node, or if a blockchain is permissioned or private, i. e., only invited entities can maintain a node [23].

Permissionless. Consensus is typically achieved through communication (e. g., voting quorums). In a permissionless blockchain, however, the participants are unknown, so it is not even known how many are participating at all. Here, communication is replaced by computation. It requires that enough work has been put into the creation of a new block so that it can be appended to a permissionless blockchain, e. g., *Proof-of-Work* [24]. This ensures that only one participant generates a new block in a given period of time on average.

Permissioned. In a permissioned blockchain, the participants are known, and their number may be limited so that consensus can be reached through communication. This type of consensus is more lightweight and efficient. In most cases, participants do not trust each other, so a central database system as an alternative solution is not an option.

In summary, a blockchain has the following three key properties:

I. It is **immutable**: Once a block is created, it is final. It cannot be modified subsequently, not even the link to its predecessor. The blockchain is an append-only data store. A new block can only be appended to an existing blockchain.
II. It is **tamper-resistant**: The data of a block are stored in authenticated data structures. These data structures are capable of verifying the integrity of their content. Tampering with their content gets therefore detected.
III. It is **decentralized**: Each node in a blockchain network manages its own instance of the blockchain. Thus, there is no single point of failure or attack. A consensus mechanism ensures that all nodes append the same, new block to the blockchain.

Although blockchains are becoming more and more popular as a secure and trusted data store, they differ significantly from traditional databases because of their completely different focus. While traditional databases are based on client-server architectures, blockchains are managed by a network of peer nodes, each of which holds a redundant copy of the full blockchain data. By eliminating the central management entity that has full control over the data store (and thus the data), trust is built — even if there is no trust among the participants of the network — but the management and communication overhead increases significantly. Besides this transparency, blockchains also create additional trust due to the immutability of the data and their tamper-resistance. These two properties are inherently guaranteed by the design of the blockchain, i. e., by organizing the data into blocks, all of which are linked via the cryptographic hashes in their headers. These blocks have no semantic meaning — they only reflect the chronological sequence in which the data are inserted into the blockchain. Data within a block can be entirely heterogeneous. There is no partitioning of the data into semantically associated tables or a strict schema for describing the data, as is the case for traditional databases. Meanwhile, traditional databases do not have comparable inherent security mechanisms. Yet, these security features are obtained in blockchains by the fact that they are append-only data stores, i. e., data cannot be subsequently deleted or modified. An update to an existing data record must be realized as a new entry, e. g., as a newer version of the complete

data record or as an addition entry containing only the changes. As a result, blockchains cannot provide full CRUD support (create, read, update, and delete). However, due to the append-only structure of blockchains, they provide a complete data history in addition to the current state of the stored data, whereas traditional databases usually only contain the latest snapshot of the data. Depending on the chosen consensus mechanism, it may take some time until a data record is actually included in the blockchain, whereas there is no such delay in traditional database systems [25,26]. For all these conceptual and architectural reasons, query performance is also much higher for traditional databases in terms of throughput, the key efficiency metric for data stores [27,28]. Table 1 summarizes the main differences between traditional databases and blockchains that have an impact on their query capabilities.

Table 1. Main differences between traditional databases and blockchains.

Property	Traditional Database	Blockchain
Architecture	The traditional database model assumes that there is a central trustable administrator for the entire database. On that account, the database is hosted on a server and subordinated clients have to send their queries to this server.	The blockchain model assumes a network of equal nodes. Each node hosts its own instance of the entire blockchain. Although each node can execute queries independently, the network must agree on which is the valid state of the blockchain.
Replication	Even though traditional databases can use replication techniques internally, e. g., to prevent failure of physical storage media, externally there is only one database instance.	In a blockchain, there is full replication of all data on all nodes, i. e., the failure of a single node does not affect the availability of the data.
Validation	Traditional databases only ensure that if the database was in a consistent state before a write operation, it is also consistent after that operation. In addition, it is ensured that no side effects can occur when several users operate on the database.	Two types of validation take place in blockchains: a) The nodes in the network agree in a consensus feature on what the valid state of the blockchain is, i. e., what data are part of the blockchain. b) Users can verify the integrity of the data due to the tamper-resistance.
Structuring	Traditional databases organize data into tables, each with its own schema.	Blockchains organize data into blocks that have no semantic meaning.
Operations	Traditional databases provide full CRUD support.	Blockchains support only read and write (add new data) operations.
History	Traditional databases contain the latest snapshot of the data only.	Blockchains provide the complete data history.
Insertion	Inserted data are immediately available in a traditional database.	Due to the consensus mechanism, data are inserted with a time delay.
Performance	Traditional databases are geared towards a high data throughput.	The data throughput is significantly low due to the consensus.

Unlike in traditional database systems, data do not necessarily have to be stored in a blockchain. To this end, there are basically two approaches [29]. In the first approach called "on-chain", the actual data are stored within a blockchain. In the second approach, called "off-chain", the actual data are still stored in a traditional database system, but the information required to verify the actual data is stored on the blockchain. However, the verification overhead is significantly greater than with the first approach. Hybrid approaches are also possible, e. g., data are stored partly in a blockchain and partly in a traditional database system with their verification information on a blockchain.

Overall, the public verification of the data in a blockchain is a fundamental characteristic of blockchain technology. This transparency enables every node to check the integrity of the data in a blockchain, thus creating trust in the stored data. The focus on blockchain

technology is on security, unlike traditional database systems, which focus on performance (i. e., transaction throughput). Additionally, blockchain technology provides protection against attackers, whether from hackers or malicious insiders, as well as protection against a single point of failure or attack, as data are replicated by many nodes, hopefully located around the world.

3. Application Domains Identified through Literature Review

As shown in the previous section, blockchains are technically very different from traditional databases, yet blockchains can in principle be used just like traditional databases — as a data store. Due to their decentralization, immutability, and tamper-resistance, blockchains offer additional security features that traditional data stores lack. At the time of this writing, many entities like companies, governments, and startups are evaluating the applicability of blockchain technology in their domains. As a result, further use cases utilizing blockchain technology in addition to a cryptocurrency have emerged over the course of time. According to Lo et al. [30], the use of blockchain technology is particularly beneficial when one or more of the following requirements are present:

- There is a need for establishing a trustworthy foundation between several parties without having to involve external authorities (e. g., notaries).
- There is a need for a single view of the truth (e. g., when different companies have to share data).
- There is a need for greater auditability by stakeholders through transparency (i. e., all published data are visible to every participant in the blockchain network) and provenance (i. e., the full history of data is available).
- There is a need for data being immutable (i. e., already stored data cannot be subsequently modified or deleted) and tamper-resistant (i. e., preventing an attacker from manipulating stored data).

From our literature review, we have identified five main application domains where one or more of the aforementioned requirements are present, and blockchain technology could therefore be a suitable technical design choice. These domains are *health data management* (see Section 3.1), *financial accounting* (see Section 3.2), *registries* (see Section 3.3), *food supply chains* (see Section 3.4), and *e-voting* (see Section 3.5). From these application domains, we derive typical types of data and types of queries in order to determine whether today's blockchain technology provides comprehensive query capabilities of the data history of a blockchain. These application domains are just a few selected examples that seemed particularly relevant in the context of our work. There are many other application domains that have similar query requirements, e. g., in the domains of *Smart Grids* [31,32], *digital rights management* [33,34], or *Smart Traffic* [35,36].

The main findings regarding the requirements for the query engine resulting from these use cases are summarized at the end of this section (see Section 3.6).

3.1. Health Data Management

In the health sector, digitization of many processes can significantly facilitate the lives of patients and physicians [37]. To this end, data in the form of patient records, e. g., electronic health records [38], must be shared and extended reliably and trustworthy among physicians. For example, a primary care physician prepares a medical record, and then refers the patient to a specialist, who adds their diagnosis. In addition, due to the *Quantified Self Movement* [39], people started to monitor themselves using IoT technologies, e. g., blood glucose measurements via continuous glucose monitoring [40] or heart rates via a smartwatch [41]. All these measured data are gathered in a central hub (e. g., a smartphone) and linked to compose a personal health profile [42]. By adding these personal health profiles to the patient records, physicians have access to even more health-related data which helps them to make a more accurate diagnosis.

The use of blockchain technology is suitable in such a use case because it allows decentralized data sharing. With a blockchain, a hospital can provide a data infrastructure

through which physicians can share patient data with each other in a simple manner [43]. Moreover, the inherent immutability and tamper-resistance characteristics of a blockchain ensure data security, which is mandatory for medical data. This is particularly important due to the increasing threat of cyberattacks in healthcare [44]. By enabling patients to participate in the blockchain, they are empowered to provide additional health-related data on their own [45].

Especially when sensitive data such as health data are stored in a blockchain, it is obvious that data privacy protection measures have to be applied. This, however, contradicts the fundamental principles of a blockchain, according to which every participant has full access to all data. To this end, Peng et al. [46] present an approach in which data are stored tamper-resistant in a blockchain, but in which queries are processed in a privacy-preserving manner, and in which the result sets do not allow further inference about the data subjects.

There are multiple examples in literature in which blockchains are used to manage and share health data, e. g., De Aguiar et al. [47], Hasselgren et al. [48], Khatoon [49], Przytarski et al. [50], and Tanwar et al. [51].

Based on this research, we can conclude that there are two types of data in health data management:

- The data entered by physicians are usually documents, e. g., diagnosis and treatment plans, that are modified over time.
- The data entered by patients are usually measurements carried out by medical IoT devices that are only valid at a specific point in time.

In the context of health data management, queries regarding the current health status of an individual patient, information on disease progression over a given period of time, as well as aggregate measurement data are particularly relevant. Typical queries therefore include, but are not limited to:

- Retrieve all diagnoses of a specific patient from a given date.
- Retrieve the latest diagnosis of a specific patient where changes to the document are highlighted.
- Aggregate the measurements of a specific patient over a given period.

3.2. Financial Accounting

Today's accounting is still based on the double-entry system that was described in a treatise written by Luca Pacioli over 500 years ago [52]. The double-entry system has two sides known as debit and credit. Each financial record is entered into an account on both sides where the entry on the credit side is a corresponding and opposite entry of the debit side. The books are considered trustworthy if and only if the sum of the debits equals the sum of the credits [53]. Since a company is accountable to multiple parties — e. g., owners and investors — it is necessary to publish financial statements regularly. This implies that financial data must be shared with these shareholders, but also with tax advisors and financial authorities. The exchange of data is usually carried out via the error-prone import and export functionality of accounting software. As financial records must be immutable by law — i. e., they must not be tampered with retrospectively — such a modus operandi entails a considerable threat potential [54].

Since blockchain technology has already proven to be a backbone for cryptocurrencies, they also seem suitable for financial accounting. Accounts for any kind of assets, liabilities, equity, revenue, and expenses are established [55]. As all transactions between these accounts are transparent to all participants of the blockchain and no party has sole control over the blockchain due to its decentralized and distributed design, it can be considered a trusted single view of truth. Moreover, due to the immutability of financial records, a blockchain-based financial accounting is almost immune to tampering [56].

There are multiple examples in literature in which blockchains are used to support accounting, e. g., Faccia et al. [57], Gökten and Özdoğan [58], Schmitz and Leoni [59], Sveistrup Søgaard [60], and Zhang et al. [61].

Based on this research, we can conclude that there is only one type of data in financial accounting:

- The data entered by companies and tax advisors are financial records that are only valid at a specific point in time.

In the context of financial accounting, queries regarding the aggregated characteristic values over a given period of time or queries that support an accounting report are particularly relevant. Typical queries therefore include, but are not limited to:

- List all financial records for a given period (e. g., usually for a day, week, month, quarter, or year).
- Generate an accounting report by aggregating the financial records grouped by accounts for a given period.

3.3. Registries

A registry is an authoritative data source of records, usually maintained by a government agency. For instance, a land registry specifies who is permitted to use land, for how long, and on which conditions. Although the registry is maintained by a central authority, several other parties have to have access to the data in order to enable an economic and healthy business environment for the sale and purchase of property [62]. Only a few countries maintain a functioning land registry, which is still often based on paper-based documents, leaving them vulnerable to loss, misuse, or corruption. As a result, delays in ownership transfer or tampering with the land register are possible and bound to happen on a regular basis [63]. Another problem is that some registries exist duplicated in siloed entities so that this fragmentation might cause possible data conflicts and therefore, no single view of truth [64].

It is obvious that the use of blockchain technology can also provide a solution to all of these problems. On the one hand, blockchain technology ensures that documents are available to all participants almost immediately after they have been added to the blockchain. This eliminates unnecessary delays in processing that occur when paper-based documents are shipped. As a result, all participants always have the latest state of a document at their disposal and conflicting copies of one and the same document cannot exist [65]. On the other hand, the use of blockchains reliably prevents the forgery of documents due to the characteristics of a blockchain, i. e., its immutability and tamper-resistance. Since no central authority can gain full control over the blockchain, corruption is also not a problem as long as the majority of the participants are honest [66]. Obviously, it must be ensured that insights from the documents are not made public. However, this can be achieved by means of access policies and tailored permissions restricting the access of individual parties to the data. Such an approach is acceptable in terms of fraud protection as long as the blockchain itself is still governed by multiple entities [67].

This benefit is also demonstrated by many research papers for other registries, e. g., Benarous et al. [68], Rosado et al. [69], Sahai and Pandey [70], Shinde et al. [71], and Singh Yadav and Singh Kushwaha [72].

Based on this research, we can conclude that there is only one type of data in registries:

- The data entered into registries are usually documents (i. e., semi-structured data) that are modified over time. Typically, the latest state of a document is of importance, but in cases of conflicts, its history is also required (e. g., in court).

In the context of registries, queries regarding the latest of a certain document (as well as its history) are particularly relevant. Moreover, a data subject can be part of multiple registries, e. g., one registry containing all house owners and one containing all vehicle owners. In order to determine all properties of a certain data subject, a join between all available registries is required. Typical queries therefore include, but are not limited to:

- Retrieve the latest state of a specific document.
- Retrieve the latest state and a prior state of a specific document to highlight changes in the latest state.

- Join two or more registries on a certain attribute to get a holistic view of all stored documents.

3.4. Food Supply Chains

A supply chain is a network of entities in such sectors as agriculture and manufacturing ranging from producers, who produce a product or service, to the final consumer. In such supply chains, not only the physical exchange of the goods is important, but also the exchange of information about these goods. This information must be available to the participants of supply chain management in order to be able to ensure comprehensive quality control [73]. In the food industry, for example, meat products must maintain a cold chain in order to avoid endangering consumers' health [74]. This means, the temperature of the meat products has to be permanently monitored and fully documented during transport from the slaughterhouse to a retail store [75]. In order to exclude human errors, IoT technologies can be used for the metering and documentation [76].

While the use of IoT technologies can prevent unintentional measurement errors, it is also necessary to prevent tampering with the documents retrospectively, e. g., to guarantee that a breach of the cold chain is recognizable. Although the captured data must not be edited subsequently, it has to be possible to modify the accompanying documents to the meat products nevertheless, e. g., if additional entries are made during customs inspections or when the goods are handed over to the next supply chain entity [77]. The use of a blockchain to establish an immutable and decentralized data store for this data therefore makes sense. Besides eliminating the risk of fraud, the transparent data sharing capabilities of the blockchain also increase consumer confidence in the quality assurance of food products, as they can verify it in a tamper-proof manner [78].

There are multiple examples in literature in which blockchains are used to store proofs and certificates regarding food supply chains, e. g., Duan et al. [79], Köhler and Pizzol [80], Kayikci et al. [81], Shahid et al. [82], and Zhang et al. [83].

Based on this research, we can conclude that there are two types of data in food supply chains:

- The data generated by IoT devices are events and thus, only valid at a specific point in time (e. g., temperature or location).
- There may exist accompanying documents (i. e., semi-structured data) to the goods that are modified over time (e. g., during customs inspections).

In the context of food supply chains, queries that provide an aggregated overview of all captured data, as well as comprehensive querying of all documented data related to the transport, are particularly relevant. Typical queries, therefore, include, but are not limited to:

- Aggregate the events by specific attributes for a given period.
- Retrieve the latest state of an accompanying document for a given transport.
- Retrieve the latest state and a prior state of a specific document to highlight changes in the latest state.

3.5. E-Voting

Electronic voting systems (known as e-voting) are a means of strengthening democratic processes. By digitizing the election process, not only is bureaucracy reduced, but people can cast their votes much more efficiently. This is an advantage especially for elderly voters or voters with a disability, as e-voting enables them to participate in the election without having to leave home and rely on the help of others [84]. While in the past, mostly technical difficulties impeded the introduction of e-voting, in today's fully connected world, it is rather a matter of security concerns [85]. To this end, the transmission of votes must be trustworthy and secure [86], and the secrecy of the ballot has to be respected [87].

However, one of the most important confidence-building measures is to ensure full transparency in e-voting and election results. This means, all voters must be able to verify

that every vote is counted and that ballots are not manipulated retroactively. The use of blockchains is therefore particularly suitable to manage the votes. First of all, the community decides by consensus which data are included in the blockchain, i. e., which votes are valid. Storing votes in a blockchain ensures that they are immutable, and tampering can be detected immediately. In addition, blockchains provide great transparency because each participant in the blockchain network keeps a complete copy of the blockchain — and thus all of the data — on their node [88]. Furthermore, the decentralized nature of blockchains ensures availability, as they are less susceptible to denial-of-service attacks than a centralized approach [89].

There are multiple examples in literature in which blockchains are used to support secure and transparent e-voting, e. g., Hanifatunnisa and Rahardjo [90], Hjálmarsson et al. [91], Kshetri and Voas [92], Ruparel et al. [93], and Wang et al. [94].

Based on this research, we can conclude that there is only one type of data in e-voting:

- The votes are stored in the blockchain as independent records. Once a vote has been cast, it must not be subsequently altered or deleted. Without any loss of generality, we assume that some kind of verification of whether a ballot is valid takes place before the votes are entered into the blockchain. Therefore, no extensions to the stored data are required.

In the context of e-voting, statistical queries that aggregate the stored data are particularly relevant. Typical queries therefore include, but are not limited to:

- Determine the final result of an election.
- Determine the voting behavior of different groups of voters.
- Determine which shifts of voters happened compared to the last election.

3.6. Lessons Learned

Derived from the presented application domains, we conclude that there are two different types of data that are entered into a blockchain. We outline their characteristics in Table 2. The first type of data entered into a blockchain is only valid at a specific point in time, which we call a *constant object*. Constant objects are, in other words, just events, such as those known from complex event processing [95]. However, there is a peculiarity in dealing with the timestamp of a constant object. This is because the timestamp can be dependent on the block in which the object is stored (i. e., an object with a *block-dependent timestamp*), or dependent on the object, because the object itself provides a timestamp attribute that must be used rather than the timestamp of the block (i. e., an object with an *object-dependent timestamp*). The second type of data entered into a blockchain is modified over time, which we call an *expandable object*. As the modifications are scattered over many blocks, they must first be combined in order to be used further. Therefore, expandable objects have only block-dependent timestamps. We use the term "object" to describe a set of attributes, i. e., data in the form of a set of key-value pairs, so-called fields. Although the concept of objects is mainly known in the paradigm of object orientation, this data model does not restrict us to the use of object-oriented data stores. These objects can also be represented in other data models such as *JSON documents*, *RDF triples* (i. e., mapping the fields of an object to individual triples), or *XML instances*. Listing 1 shows an object named `obj1` with three attributes and their values in those three representations. We discuss those object types further in Section 4.

Table 2. The two types of objects which are relevant in the context of blockchains.

Type of Data	Main Property	Timestamp
Constant Object	This type of data is only valid at a specific point in time. Thus, it is final, i. e., its fields (key-value pairs) and corresponding values are constant and do not change over time.	block-dependent
		object-dependent
Expandable Object	This type of data can grow and shrink over time, i. e., in the future, new fields may be added, values of existing fields may be modified, or existing fields may be removed. Any state of the object can be restored by exploiting the history feature of the blockchain.	block-dependent

An object represented in the form of a JSON document named `obj1`.

```
{
  "attr1": "val1",
  "attr2": "val2",
  "attr3": "val3"
}
```

An object represented in the form of RDF triples.

```
<obj1, attr1, "val1">
<obj1, attr2, "val2">
<obj1, attr3, "val3">
```

An object represented in the form of an XML instance.

```
<obj1>
  <attr1>val1</attr1>
  <attr2>val2</attr2>
  <attr3>val3</attr3>
</obj1>
```

Listing 1. An object with three attributes and their values represented as a JSON document, RDF triples, and an XML instance.

Furthermore, from the presented use cases, we derive eight query capabilities that an efficient query engine for blockchain systems has to support in order to be usable in the given application domains. These required capabilities are *projection*, *selection*, *sorting*, *aggregation*, *grouping*, and *joins*. These operators are well-known from the *relational algebra*, on which the query languages of many traditional database systems are based.

Projection means selecting specific attributes from objects that are included in the result set, i. e., if an object has several attributes, only a specific subset of them is returned. For instance, a physician requires a projection operator to query specifically blood pressure measurements from an electronic health record, which also includes other medical data such as blood glucose measurements or dietary studies. Selection means eliminating objects from the result set, i. e., an object is only included in the result set, if its attribute values meet a given condition. For instance, a physician requires a selection operator to query for female patients (i. e., patients whose attribute "gender" is set to "female"). Sorting means to sort the objects in the result set in ascending or descending order, based on the values of the attributes of the objects. For instance, in financial accounting, it is necessary to sort the accounting items in order to present them according to the date they were registered. Aggregation means to compute a single value from a set of values with the help of an aggregate function, such as average, maximum/minimum, or sum. For instance, in financial accounting, an aggregation is required to compare the total sum of income with the total sum of expenses in the end. Grouping means to partition objects into groups of objects, based on the values of their attribute. For instance, land registries have to group the landowners based on the county their property is assigned to. Usually, an aggregation is then applied on these groups, e. g., to determine how much real estate tax each county receives. Joining means to combine data from multiple sources into a joint result set. While in traditional database systems joins are applied to different tables within the same database, in blockchains there is no such semantically structuring construct like a table.

Therefore, joins have to be applied to different blockchains. This, however, raises further technical issues, see Sections 5 and 7. Nevertheless, there are use cases in which joins have to be supported by blockchain systems. For instance, if there are different registries, e. g., a land register and a car register, each stored in its own blockchain. In order to query all possessions of a data subject, a join on all of these blockchains is necessary.

In addition to these six basic query operators, which are also known from traditional database systems, blockchains have special requirements towards query capabilities due to the two different object types that have to be handled by them. Firstly, there are *temporal queries* when dealing with constant objects. In temporal queries, the temporal relationship of the data plays a key role. These time references can be obtained from two different sources: On the one hand, each block has its own inherent timestamp. Since new blocks can only be added at the end of the blockchain, the sequence of the blocks implicitly reflects the chronological order in which they were created. This timestamp is used for block-dependent objects for temporal queries. However, it is possible that this timestamp deviates substantially from the time at which a data object was captured, since data initially remain in a data pool until a consensus is reached, and they are added to a new block. Therefore, for object-dependent objects, where the time of capturing the data is crucial, an additional individual timestamp for each object is needed. For instance, in the e-voting context it is necessary to query only valid votes, i. e., only ballots that were submitted neither too early nor too late have to be considered. Secondly, there are state-based queries when dealing with expandable objects. Such objects are initially added to the blockchain and then changes are made by means of transactions (e. g., to change certain attribute values, and add or remove some attributes) which are also stored in the blockchain. In a state-based query, the complete change history up to a specific point in time must therefore first be retrieved from the blockchain in order to assemble the expandable object. For instance, in the food supply chain it must be possible to query the status of a food product at any time between production and sale, e. g., in order to monitor the cold chain.

Table 3 provides an overview of these six basic operators as well as the two blockchain-specific query capabilities. More details on these query options are provided in Section 5.

Table 3. Overview of the six basic query operators (white rows) and two blockchain-specific query capabilities (gray rows) derived from the presented application domains.

Query Capability	Main Property
Projection	It is possible to specify which fields (i. e., key-value pairs) are included in the result set.
Selection	It is possible to specify which objects are included in the result set.
Sorting	It is possible to sort the result set by given fields.
Aggregation	It is possible to aggregate the values of certain fields using functions.
Grouping	It is possible to group given fields.
Join	It is possible to join different blockchains.
Temporal Queries	It is possible to query constant objects based on a timestamp. While for block-dependent objects there is an inherent timestamp given by the block they are stored in, object-dependent objects have their individual timestamp, which is specified in their attributes.
State-based Queries	It is possible to query expandable objects. Expandable objects can be scattered over multiple blocks, meaning that a state-based query must first find all pieces and compose them.

4. Object Types in Blockchains

From the presented application domains in Section 3, we derive two object types that are relevant in the context of blockchains, namely constant objects and expandable objects. Their main properties are summarized in Table 2. In the following, we elaborate on these

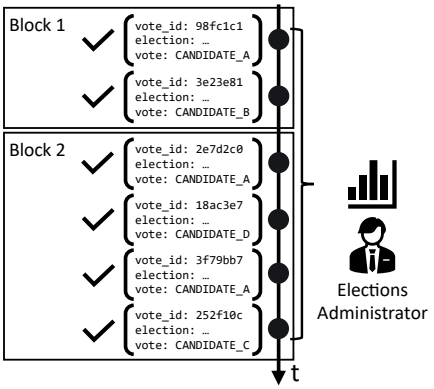
two object types and describe why they need to be considered in particular when managing data in blockchains.

As described in Section 2, blockchains are append-only data stores where blocks are appended to an existing blockchain. Furthermore, blocks cannot be modified subsequently, so the data within a block are immutable. If changes to the data must occur, there are two options. Either the complete object with all its fields is recreated or only a change history is kept. This means that there are two different forms of use. Either, an object *lives* until a new version of it is added to the blockchain or the entire change history of an object must be searched in the blockchain and applied to the *genesis object*, i. e., the original version of the object. These two forms of use are reflected by the following two object types:

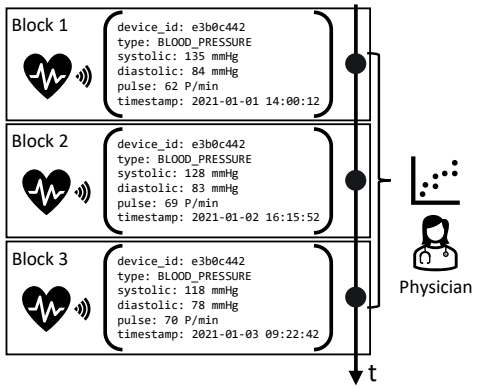
Constant Object. A constant object is final. This means that once the object is added to a block, its fields do not change. Constant objects occur over time and are valid at a specific point in time. In other words, constant objects are events, i. e., actions or occurrences that happened at a specific point in time.

Expandable Object. An expandable object is never final. This means that over time, the fields of this object are modified, new fields are added, or existing fields are removed. In other words, expandable objects are documents that get modified over time.

Constant objects are, for example, votes in e-voting (see Figure 2a) or blood pressure measurements from medical IoT devices in health data management (see Figure 2b).



(**a**) Votes during an election, represented as constant objects with block-dependent timestamps, which are aggregated by an election administrator.



(**b**) Blood pressure measurements from medical IoT devices, represented as constant objects with object-dependent timestamps, which are analyzed by a physician.

Figure 2. Two different use cases utilizing constant objects with (**a**) block-dependent timestamps and (**b**) object-dependent timestamps.

In e-voting, votes are created by voters during elections. These votes are only valid once they are successfully added to the blockchain. A vote does not contain its own timestamp attribute, because in this case, only the timestamp of the block is relevant. An election official can query and aggregate these votes to derive valuable information about an election. For these queries, it is relevant in which block a vote is included.

In health data management, a medical IoT device performs blood pressure measurements at certain time intervals. These measurements are either added to the blockchain individually or in batches. A measurement contains, among other attributes, a timestamp that records the time of the measurement. A physician can query and aggregate these measurements so that valuable information can be derived for the patient. For these queries, however, it is not relevant in which block the measurement is included, but at which time it was performed (nota bene: Due to the delayed insertion of data into the blockchain, not only the timestamp of a measurement can significantly differ from the timestamp of the

block it is stored in, but also the chronological order in which measurements are captured can differ from the order within the blocks.).

Thus, in the first example, the timestamp of the block is relevant, but in the second example, the timestamp of the object is relevant. For this reason, we introduce the following notion for timestamps on objects:

Block-Dependent. In this case, the object depends on the timestamp of the block it was included in. Each block has its own timestamp, i. e., the time at which it was created. Here, the timestamp of a block acts as a global timestamp for all its payload data, superseding possible timestamp attributes of objects, thus all objects in a block have the same timestamp.

Object-Dependent. In this case, the object has its own timestamp attribute. Additionally, it is not relevant in which block this object was included. During query processing, the timestamps of these objects must be considered instead of the timestamp of a block. However, this entails new challenges. In a blockchain architecture, there is no guarantee that the objects are sorted by the timestamp attribute of the objects. As a result, when searching for an object with a specific timestamp, it can only be assumed that the object was created earlier than the block that includes it. Thus, the lower search bound is set by the timestamp of the object, however, no statement can be made about the upper search bound.

Whether an object has a block-dependent or an object-dependent timestamp is determined by its further usage. In our e-voting example, the action is to cast a vote and this is considered to be performed once it is correctly added to the blockchain. In our health data management example, the action is a blood pressure measurement carried out by an IoT medical device, which is considered to be performed once the measurement is successfully completed. This action is completely independent of the creation of a block for a blockchain.

Expandable objects are, for example, documents in land registries (see Figure 3). An expandable object consists of a genesis object (i. e., the source object) as well as modifications to the object that are scattered over numerous blocks. As a result, it has as many states (i. e., document revisions) as how many blocks exist that include fields of this object.

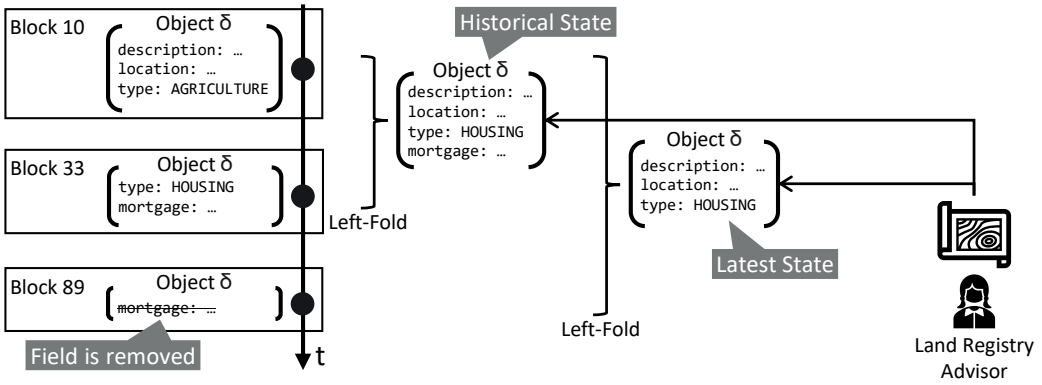


Figure 3. A land registry document, represented as expandable object, which is modified over time. Different states of the document can be retrieved, i. e., the latest state and all historical states.

In land registries, land documents are inserted, modified, and deleted over time. When a land document is modified, it means that fields of the document are modified, new fields are added, or existing fields are removed. The result of a modification is a new state of the expandable object. Thus, each block that include a modification of an expandable object represents a different state of this very object. A land registry advisor can query these land documents at any available state. For this, the requested state of the document has to be computed.

To compute a state of an expandable object, all fields from the previous and the requested block must be combined. This is done by recursively recombining the fields from

the first block that includes fields of the object until the requested state — this approach is also called *left-folding*.

In our land registry example, the object first appeared in Block 10, the so-called genesis object. After that, there have been two modifications to it, namely in Block 33 and Block 89. This means that there are three states for this object, all of which can be queried. Querying its state in Block 10 is simple, since no modifications have taken place yet. Querying its state in Block 33 requires its assembly by combining the fields from Block 10 with the modifications in Block 33. The same procedure is used for querying its state in Block 89, although an additional combination step has to be performed then.

Furthermore, the timestamps of expandable objects are only block-dependent, i. e., the block defines the corresponding timestamp for these objects.

5. Query Capabilities for Blockchain Technology

As discussed in Section 4, the different object types have a significant impact on how data can be queried in a blockchain. Therefore, in this section, we adapt a potential query language to the object types using the query capabilities listed in Table 3 and elaborate on possible issues that need to be considered when implementing a query engine.

In blockchain technology, writing data is a completely different process compared to traditional database systems. This is due to the consensus mechanism used to add new data to the blockchain (see Section 2). Therefore, we only consider non-modifying query techniques, i. e., read queries as they have no persistent effects. Nevertheless, data can still be added to a blockchain by creating a new block that includes the new data and propagating it via the given consensus mechanism.

A query engine consists of a frontend and a backend. The frontend is responsible for transforming a query written in a defined query language into an intermediate representation. The backend is responsible for processing this intermediate representation and computing the result of that query.

The use cases shown in Section 3 require comprehensive query capabilities such as aggregations or joins. For the complete breakdown of required capabilities, see Table 3. We consider a query engine to be powerful, if it supports a query language with at least the same power as a `SELECT` statement from the declarative query language SQL — just like in traditional database systems. Current blockchain systems, however, have native but naive query interfaces [96]. Moreover, their query languages and the efficiency of query processing is severely limited [97]. Since descriptive query languages have proven themselves in practice also for object-oriented database systems [98], we describe the required queries in SQL. SQL provides an expressive query language [99], however, SQL is just one example that can easily be replaced by any other declarative query language. In particular, we focus on the `SELECT` statement, since this is used for the read queries. However, the `SELECT` statement cannot be simply adopted, but has to be modified to support the different object types.

In relational database systems, the SQL `SELECT` statement is the most common option to query a database. Within this `SELECT` statement, there are various clauses intended for, e. g., selecting, aggregating, or sorting. Table 4 shows these various clauses and maps them to the respective query capability along with a mapping to the blockchain domain.

For almost all of these clauses, a relatively straightforward mapping to the blockchain domain can be found. However, the `JOIN` command represents an exception. Since blockchains have no logical internal structuring (nota bene: The blocks in which the data are organized have no semantic meaning regarding the data. They only represent the chronological order in which the data were added to the blockchain.) (e. g., in semantically and schematically homogeneous tables), a `JOIN` gets a different and new meaning in this context. As illustrated in the example of the registries (see Section 3.3), it happens in practice that data from a single data subject are contained in several different blockchains. To collect and combine all information, a `JOIN` across multiple blockchains is required. However, as outlined in Section 2, blockchains do not have a uniform structure. Thus, it

must be resolved how a `JOIN` can be realized despite the highly diverse technologies that are involved in this case.

While these SQL clauses are sufficient to cover all six basic query operators (see Table 3), the inclusion of novel blockchain-specific object types (see Section 4) represent a significant deviation from SQL. Due to these object types, additional query capabilities — alongside with extensions to the query language — are needed in blockchain systems.

Table 4. The various clauses of an SQL statement and their mapping to the blockchain domain.

Query Capability	Relational Data Model	Blockchain Domain
Projection	`SELECT <columns>` An SQL statement starts with the projections, a list of columns to include in the final result set.	`SELECT <attributes>` Instead of columns, attributes of objects are specified.
Join	`FROM <table> JOIN <other tables>` This clause indicates the table from which to retrieve the data. `JOIN` subclauses enable the joining of additional tables.	`FROM <blockchain> JOIN <other blockchains>` Instead of a table, the blockchain is specified. If there is only one blockchain given, then the clause might be omitted. If there is more than one blockchain given, `JOIN` subclauses are required.
Selection	`WHERE <comparison predicates on columns>` This clause eliminates all rows from the result set where a comparison predicate does not evaluate to true.	`WHERE <comparison predicates on attributes>` Instead of rows, objects are eliminated.
Grouping & *Aggregation*	`GROUP BY <columns>` This clause groups values of one or more columns in conjunction with aggregation functions in the projection on those columns.	`GROUP BY <attributes>` Instead of columns, attributes of objects are specified.
	`HAVING <comparison predicates on groups>` This clause eliminates all groups of returned rows to only those whose comparison predicate does not evaluate to true.	`HAVING <comparison predicates on groups>` Instead of rows, objects are returned.
Sorting	`ORDER BY <columns>` This clause indicates the columns to use to sort the result set including the sort direction.	`ORDER BY <attributes>` Instead of columns, attributes of objects are specified.

Constant objects are self-contained, which means that, considered individually, they do not provide valuable information in most cases. Thus, it is suitable to consider several of these objects at the same time. This can be done, for example, either in the form of an aggregation or viewing the data as time series to track any trends. In order to support this, a start and end point are required. However, the range queries differ here in whether the objects have block-dependent or object-dependent timestamps. For objects with block-dependent timestamps, the timestamp of a block is relevant, therefore, it must be possible to specify two block numbers. Thus, it must be possible to search between block N_1 and block N_2. To apply this to SQL, the `SELECT` start clause could be adjusted as follows:

Block Range. `SELECT <attributes> BETWEEN BLOCK` N_1 `AND` N_2
(where N_1 and N_2 of type `Integer` and $N_1 \leq N_2$)
A *block range* is necessary when a blockchain stores constant objects with block-dependent timestamps.

The situation is different for objects with object-dependent timestamps. Here, the order in which the data was added to the blockchain is irrelevant, it only matters when the data was originally generated. Therefore, it is necessary to search via the timestamp of the objects. This means that only objects created between timestamp T_1 and T_2 are searched. To apply this to SQL, the `SELECT` start clause could be adjusted as follows:

Timestamp Range. `SELECT <attributes> BETWEEN TIMESTAMP` T_1 `AND` T_2
(where T_1 and T_2 of type `DateTime` (e. g., ISO 8601 [100]) and $T_1 \leq T_2$)
A *timestamp range* is necessary when a blockchain stores constant objects with object-dependent timestamps.

Even though the two queries look very similar, they are internally very different from each other. Since the block range corresponds to the structure of the blockchain, such a query can be supported very efficiently. The timestamp range, however, requires all blocks between the block with timestamp T_1 (nota bene: Even if it is not clear when an object is added to the blockchain, the insertion time (i. e., the timestamp of a block) can in no case precede the timestamp of the object (i. e., the time of capturing).) and now to be searched, since the timestamp of the actual block where the object has been included is greater than the timestamp of the object itself.

Expandable objects have fields that are scattered over one or more blocks. These objects must be assembled before they can be processed to compute the result of a query. It is obvious that the states of all processed objects must be at the same *height* (In this context, the term "height" is used to describe the block number within a blockchain up to which all required objects have to be assembled.) to prevent the processing of incompatible states of data. Therefore, it is necessary to specify a block number N up to which block the objects are being assembled (nota bene: A lower bound is not required in this case, since it is always necessary to start with the genesis object and apply all modifications from there.). To support this, the `SELECT` start clause could be adjusted as follows:

Block Number. `SELECT <attributes> ASOF BLOCK` N
(where N of type `Integer`)
A *block number* is necessary when a blockchain stores expandable objects.

This way, all required query capabilities for all object types can be represented in a declarative query language. This shows how powerful a declarative query language is. However, the query language is just the frontend of a query engine.

The actual issues arise when the backend of a query engine is considered, as it accesses the underlying data structures to compute the result of a query. We identified the following eight issues that need to be addressed:

(A) `JOIN` operators as provided by traditional database systems, do not need to be considered here, as there is no demand for this functionality in practice. Unlike traditional database systems, blockchain typically store data on a single topic only. An internal structuring into separate tables, each with its own schema, is therefore not necessary in blockchain systems. Consequently, joins cannot be performed within the data set of a single blockchain. However, there are use cases that require a join between data sets held in different blockchains. For instance, Blockchain X contains health data that are captured self-reliant by patients as part of the Quantified Self Movement, while Blockchain Y contains clinical data of these patients captured by hospital staff as part of health checks. In order to get a comprehensive view of a patient's health situation or history, physicians need to be able to join the data from these two blockchains. Since each blockchain system has its unique technical architecture regarding its storage, network, and consensus (see Section 2), such a join represents a substantial technical challenge.

(B) Unlike a relational table, where all data are applied to a table schema, blockchain objects have no common well-defined schema. Here, the structuring of the objects

is done solely at the application level. That is, each application stores its objects in its own predefined schema. However, when several applications share a blockchain to store their data, multiple schemas are simultaneously present in that blockchain. Therefore, the question is how this inhomogeneity affects query processing?

(C) Data read from a blockchain should always be verified to detect any tampering. However, data could also be stored externally to a blockchain in a database system with better query capabilities, but without verification capabilities. Therefore, the question here is how, and when does the verification of the data take place? During query processing, if possible, or as an additional step by verifying an externally computed result against the blockchain?

(D) Database systems utilize index structures to facilitate query processing. Can such structures also be used for query processing in blockchains? If so, how could these look like for constant objects and/or expandable objects? Is it possible to verify the data in these index structures?

(E) Blockchains lack an internal structuring that has a semantic meaning. While the segmentation into blocks is beneficial for some queries — think of queries for expandable objects, for instance, where the state up to a specific point in time is required, which can be easily realized via a query on the block number — this complicates queries on the timestamp of a constant object, for instance, since all blocks created at this timestamp or later have to be traversed for this purpose.

(F) The query processing of constant objects and expandable objects is very different. Can these objects be technically processed simultaneously in a blockchain? If so, does it make sense from a query language perspective?

(G) The query processing of constant objects with object-dependent timestamps is more complex than that of constant objects with block-dependent timestamps. Can these objects be technically processed simultaneously in a blockchain? If so, does it make sense from a query language perspective?

(H) The query processing of expandable objects is significantly more complex than that of constant objects with block-dependent timestamps, since for each object it is first necessary to determine which attributes it has, and in which blocks they are located.

6. Overview of the State of the Art

While blockchain technology was initially developed for cryptocurrencies, for which it is sufficient to query the current account balance, the new use cases identified from the application domains in Section 3 introduce different types of objects (see Section 4) that require comprehensive query capabilities (see Section 5). Since there is not a standard for blockchain systems, but rather a modular design that can be freely configured from a variety of technology variants (see Section 2), there are various blockchain systems, each targeting a different goal. As a result, the query capabilities of these systems are quite different. In this section, we therefore first consider the state of technology (see Section 6.1) and then the state of research (see Section 6.2) in the field of query processing in blockchains.

6.1. State of Technology

The currently most popular blockchain system *Hyperledger Fabric* [101] manages a ledger that consists of a blockchain and a database that holds the current *world state*. The world state represents the latest state of a blockchain and is stored in an additional NoSQL database. Hyperledger Fabric uses *CouchDB* (see https://couchdb.apache.org; accessed on 15 December 2021) to this end. Despite the fact that a blockchain maintains a native data history, however, there are only limited interfaces to access this data history (e. g., through Fabric SDK (see https://hyperledger-fabric.readthedocs.io/en/release-2.2/fabric-sdks.html; accessed on 15 December 2021) or smart contracts). It is possible to execute comprehensive queries against the CouchDB, which manages the latest state.

However, the result of a query is not cross-checked against the blockchain, so there is a possibility of reading tampered data.

In such blockchain systems, there is no efficient technique to query the underlying data structure, i. e., the data history of the blockchain. A solution to overcome this limitation is therefore to duplicate the data of the blockchain (or even just the current state) into a separate database with support for a powerful query engine, while sacrificing the built-in technique of the blockchain to verify the integrity of data while computing the result of a query. If, thus, information must be directly extracted from the data history of the blockchain, expert knowledge and self-developed tools are required to extract this information.

Furthermore, there are systems that have features from blockchains and databases. They are called hybrid systems and there are two alternative approaches. The first approach is to start with a blockchain system and then enhance it with database features. The second approach is to start with a database system and then enhance it with blockchain features. Ruan et al. [102] compare six of such hybrid systems and came to the following findings:

Blockchain Systems Enhanced with Database Features. These systems use blockchains as an integrity-protected data store and utilize a separate database layer on top of it. Within the network, storage operations are replicated (e. g., a block containing transactions) rather than individual transactions. Examples are:

- *BlockchainDB* [103] provides a key-value database layer on top of a blockchain, which provides a simple `get`/`put` interface as well as an additional `verify` method for data verification.
- *FalconDB* [104] provides a "traditional" database layer with temporal attributes on top of a blockchain. It relies on smart contracts for querying as there is an incentive model that each node remains honest.
- *Veritas* [105] provides a verifiable database layer on top of a blockchain.

Database Systems Enhanced with Blockchain Features. These systems use ordinary database systems and utilize transaction-based replication. Within the network, each node manages its own database instance and executes globally ordered transactions (achieved through a consensus mechanism) on it. Examples are:

- *BigchainDB* [106] provides a blockchain layer on top of a *MongoDB* (see https://www.mongodb.com; accessed on 15 December 2021) database. As all blocks, transactions, and metadata are stored in it, the full query power of MongoDB can be used to query data.
- *Blockchain Relational Database* [107] integrates a blockchain layer into a relational database management system, namely *PostgreSQL* (see https://www.postgresql.org; accessed on 15 December 2021). PostgreSQL was chosen because it keeps all versions of a row. Usually, relational database systems update data in-place and maintain a rollback log.
- *ChainifyDB* [108] provides a blockchain layer on top of arbitrary database management systems that are SQL-99 [109] compliant. It uses a new processing model that reaches consensus on the effects, i. e., database states and snapshots.

We conclude that in both approaches, the system *can* provide query capabilities that are mostly as powerful as the query engines of the applied database systems (i. e., document-oriented databases and relational databases). However, these underlying traditional database systems provide no support for block range queries, timestamp range queries, and block number queries, as required in modern blockchain use cases. In addition, each approach has its own disadvantage.

The disadvantage of blockchain systems enhanced with database features is generally that data in the database are decoupled from the data in the blockchain so that verifying the results of a query is an additional step that can become expensive. Depending on how the data of the blockchain are stored in the database, queries are possible either only on the

latest state or also on the history. FalconDB uses *MySQL* (see https://www.mysql.com ; accessed on 15 December 2021), which provides a relational data model, that they extended by temporal attributes to support SQL queries on the history.

The disadvantage of database systems enhanced with blockchain features is generally that the database system itself might not use tamper-resistant data structures so that tampered data is detectable. There are techniques to overcome this such as querying multiple nodes in the blockchain network and comparing the result or re-executing the transactions from the blockchain to detect incorrect data. However, these techniques are cumbersome and can also become expensive.

6.2. State of Research

Given the problems with the State of Technology, there is also a variety of research. These can be divided into four research directions:

Improvements to the Frontend Query Capabilities. *Ethereum* [110] is a popular public blockchain that supports smart contracts, which are programs with code (i. e., functions) and data (i. e., states) that run on the blockchain. It uses the key-value database LevelDB as persistent storage. Han et al. [111] extend the Ethereum-based blockchain system *quorum* by an embedded relational database system *SQLite* (see https://www.sqlite.org/index.html; accessed on 15 December 2021) next to *LevelDB* (see https://github.com/google/leveldb; accessed on 15 December 2021) enabling SQL `SELECT` queries. In this system, the data of smart contract transactions are stored in the SQLite database instead of the LevelDB database. Smart contract transactions can use SQL queries (e. g., range or conditional queries), which are then executed by the relational database system. However, there are some open questions, e. g.,

- What happens, if the data in the relational database SQLite is tampered with?
- Smart contracts in Ethereum only have access to the latest state of their data. Is this also the case here?

The research work of Tong et al. [112] also focuses on providing SQL support in blockchains systems. However, they take a different approach. They introduce an SQL middleware, which encapsulates RPC-based (remote procedure call) interfaces of blockchain systems as SQL interfaces to facilitate SQL queries on the blockchain data, just like the aforementioned approach, where blockchain systems are enhanced with database features. Furthermore, Li et al. [113] present a data query layer called *EtherQL*, which enables a set of useful analytical queries such as range and top-k queries on the blockchain Ethereum.

Efficiency Improvements in Query Processing. Bragagnolo et al. [114] use the parallelization technique Map/Reduce to extract and analyze information from a blockchain, in their case from the Ethereum blockchain. Here, a master node instructs different jobs to worker nodes, each of which extracts data from the Ethereum blockchain and writes them to a relational database. After that, queries can be made to the relational database to obtain information from the Ethereum blockchain.

Xu et al. [115] present an accumulator-based authenticated data structure that allows aggregation over arbitrary attributes. This enables lightweight users, i. e., users who have only the block headers locally stored, to have service providers storing the full blockchain to execute boolean range queries, while allowing them to verify the integrity of the results.

Xing et al. [116] present a subchain index structure for the transaction chain. Here, the transaction chain is divided into subchains and different subchains are linked with hash pointers. The goal is to shorten the query path for queries on historical transactions.

Jia et al. [117] present the *AB-M tree* structure as a storage structure for transactions, which combines the advantages of *balanced binary trees* (fast data retrieval) and Merkle trees (fast data verification). Instead of storing transactions in an ordinary Merkle tree within a block, they are now stored in an AB-M tree. This provides faster transaction retrieval, but at the same time guarantees the integrity of the transactions.

Peng et al. [118] and, based on this, Wu et al. [119] present a middleware layer called *Verifiable Query Layer* (*VQL*). It extracts information about the blocks, their transactions, and possible balances from an underlying blockchain system and stores these data reorganized in one or more databases so that queries can be answered more efficiently. Then, a cryptographic hash value for each generated database is computed and stored in a blockchain, preferably in the underlying blockchain system. Whenever data is queried through the middle layer, the integrity of the queried database can be verified by comparing the hash value in the blockchain with the hash value computed by the user.

Tailored Blockchain Optimizations for Specific Use Cases. As IoT technologies capture growing volumes of time series data, there is an emerging need to comprehensively analyze it in an efficient manner. While there are approaches to verify the authenticity of the sources of this IoT data [120] and subsequently provide these time series data to third parties on a demand-driven basis [121], it is also necessary to ensure that the data cannot be tampered with when it is stored and managed.

Wortner et al. [122] therefore investigate particularly for time series data how these can be managed in blockchain systems and how in particular their timestamps, which play a key role in subsequent analyses, can be protected against tampering. In this context, however, the focus is solely on the storage of the data. An efficient processing of queries or let alone an analysis of the blockchain data is completely out of scope. This is being researched by Dhanush et al. [123]. In their approach, however, the time series data must first be completely extracted from the blockchain and then stored and analyzed in a special time series database (e. g., *InfluxDB* (see https://www.influxdata.com/products/influxdb/; accessed on 15 December 2021)) for which there are tailored analysis tools and dashboards (e. g., *Grafana* (see https://grafana.com; accessed on 15 December 2021)). This causes a large overhead, because there are no efficient ways to restrict the amount of data in such a way that only those data are read that are relevant for the analysis. Since the amount of data in the blockchain is continuously growing due to the append-only nature of the blockchain, this overhead is also constantly increasing. Another problem with this approach is the fact that once the data has been extracted, there is no longer any protection against tampering. This completely undermines the main reason why the data was stored in the blockchain in the first place.

Yu et al. [124] therefore propose a novel blockchain storage architecture specifically for time series data. In their approach, they introduce an index structure for blockchains enabling an efficient access to the blocks and transactions in conjunction with a time series database for managing the time series data. The system decides for incoming queries whether they should be processed by the blockchain or the time series data and then forwards them accordingly. This approach reduces the overhead significantly, because on the one hand, time series databases are highly optimized to process time series queries. On the other hand, time series data are not immutable so that the data volume can be reduced as needed by deleting data that is no longer needed. However, this also represents the key weak point of this approach — the data in the time series databases are not protected against tampering or deletion.

Yet, there are research approaches towards tailored index structures especially for time series data in blockchains. Studies show that the performance of time series queries in blockchain systems can be increased significantly by such indices [125]. This could also improve the throughput of, for example, timestamp range queries (see Section 5).

Similar research approaches can be found for other specialized data and query types, such as index structures for location data in order to support efficient spatial queries, e. g., the work by Nurgaliev et al. [126].

Verifiable Queries and Database Systems. With verifiable queries, a user is able to verify the integrity of the result of a query. This ensures that the data and the execution have not been tampered with. For this purpose, a new class of database systems has emerged, the so-called verifiable database systems.

Zhang et al. [127] propose such a verifiable database system called *vSQL*, which supports arbitrary SQL queries. Here, a user is able to outsource a relational database to an untrusted server and has only to store a hash value locally. Then, the user can send SQL queries to that untrusted server and verify the integrity of the result. This verification is done by an interactive protocol, which utilizes interactive proofs.

Zhang et al. [128] propose another verifiable database system, which is called *Spitz*. It builds on top of *Forkbase* [129], which is a distributed multi-version storage engine utilizing the key-value data model, and maintains multiple index structures to facilitate verifiable query processing. The verification of the result of a query is done by comparing the hash value, which must be computed by using the proofs included in the result, with a previously locally stored hash value.

Zhou et al. [130] propose an SGX-based verifiable database system called *VeriDB*, which uses a trusted execution environment called *Intel SGX* (see https://www.intel.com/content/www/us/en/architecture-and-technology/software-guard-extensions.html; accessed on 15 December 2021) where data are isolated and encrypted in memory [131]. VeriDB hosts the query engine supporting SQL queries within an Intel SGX enclave, with the actual data residing in untrusted memory. The verification of the data is performed during the query processing by the query engine using a verifiable storage layer.

Table 5 summarizes the main findings regarding the characteristics and features of these six research directions in the field of query processing in blockchains.

Table 5. Summary of key findings regarding the state of the art.

Research Direction	Characteristics and Features
Blockchain Systems Enhanced with Database Features	A database layer is built on top of a blockchain system that is used as an integrity-protected data store. The database layer provides an interface for querying data efficiently. However, verifying the results of a query is an additional step, which increases the overhead significantly.
Database Systems Enhanced with Blockchain Features	A blockchain layer is built on top of a traditional database system. Data are queried directly from the database system. However, these database systems are not designed to detect tampered data during query processing.
Improvements to the Frontend Query Capabilities	Existing public blockchain systems such as Ethereum are internally modified or extended with a query layer to support familiar query languages such as SQL. However, queries regarding the data history are expensive.
Efficiency Improvements in Query Processing	Various techniques such as the parallelization of data processing or novel data structures enable more efficient querying of blockchain data. However, in order for query engines to benefit from this, they have first to be adapted accordingly.
Tailored Blockchain Optimizations for Specific Use Cases	Tailored index structures for blockchain systems increase the performance of specific types of queries such as time series queries. However, they are designed specifically for a certain use case, i. e., the blockchain system loses some of its universality.
Verifiable Queries and Database Systems	Verifiable queries are enabled over novel or existing database systems. However, these approaches do not necessarily require blockchain systems to be involved.

Blockchains were conceptually not developed to compete with traditional database systems in terms of data and query throughput. However, due to their inherent security features, they are increasingly used for managing important data. Considering the current state of technology, however, blockchains are still at the very beginning as far as query

capabilities are concerned. Either one has to live with the native but naive query interfaces or the data processing takes place in a connected database system, which partially eliminates or at least reduces the security features. Therefore, there is a large body of research that aims to improve query capabilities in terms of usability, power, and performance. However, as our assessment of the state of research has shown, there are still many open research questions to be answered. In the following section, we elaborate on these open research questions.

7. Future Research Challenges

In this section, we elaborate on future research challenges based on the issues identified in Section 5 that need to be solved in order to enable an efficient query processing in blockchain systems. We group these issues into four categories of challenges, namely research challenges regarding *data models*, *data structures*, *block structures*, and *query processing*.

7.1. Data Models

In order to realize a `JOIN` operator for blockchains, a query engine has to be able to access, read, and process the common data stock of all involved blockchain systems. Since different blockchain systems have a highly heterogeneous technological infrastructure, a generic and standardized data model is needed that can be applied on all of these systems (see Issue A). Furthermore, for constant objects with object-dependent timestamps, it is useful to assign these timestamps a special status in the data model in order to access them more easily and in a standardized manner. To enable comparability of objects, it is worth considering introducing a type system, so that it is ensured that when comparing attributes of multiple objects with the same identifier, they are of the same type (see Issue B). Therefore, the first challenge is to create a standard for an expressive data model for blockchains. With such a data model, it must be possible to represent arbitrary kinds of data for any given use case. Triples, for example, have demonstrated their suitability in the context of RDF stores and could also be a beneficial approach for a blockchain data model.

7.2. Data Structures

In order to process queries on blockchain systems efficiently, state-of-the-art solutions operate a traditional database system in parallel to the actual blockchain. This database presents the current world state, i. e., the current value of the attributes of the objects stored in the blockchain. However, since these database systems cannot check the integrity of the data as required, an additional verification step is needed to check the results against the blockchain (see Issue C). To eliminate this verification step, it is necessary to come up with novel data structures, e. g., by combining search data structures with authenticated data structures such as *Merkle B-Trees* [132]. Such data structures are applied in current blockchain systems such as Ethereum. However, these structures are primarily used to facilitate the verification of transactions. A full-fledged support for comprehensive queries, as required by emerging use cases, is not provided by these structures. Therefore, the second challenge is to investigate how data structures can be designed that store generic data in a verifiable manner while providing fast access to the stored data.

7.3. Block Structures

There is some flexibility in organizing the data within a block. Data can either be physically clustered or added to useful data structures that allow efficient access to that data (see Issue D). It is also possible to construct index structures outside a block, but this would again require an additional verification step to check the results of a query against the blockchain. Thus, it is necessary to consider how the data are stored within a block. For example, different versions of the data can be stored within a block, each optimized for a certain type of query [133], similar to a triplestore with an *RDF3X engine* [134]. A lot of related work is concerned with the support of efficient spatio-temporal queries by adding special index structures to blockchain systems. Similar efforts are also needed

for other types of data that are relevant in emerging application domains for blockchains (see Issue E). For example, expandable objects require special index structures in order to assemble them more efficiently. This can be done by storing pointers to their previous state, which simplifies left-folding. Similarly, constant objects also require index structures so that their history can be queried efficiently. Therefore, the third challenge is to investigate how the structure of a block could be designed to efficiently support different types of queries.

7.4. Query Processing

In query processing, the question is whether technically both object types (constant and expandable) can be supported at the same time (see Issue F). Even if this is technically possible, it could be contradictory from the perspective of the query language. The same question arises whether constant objects with object-dependent timestamps should be stored together with block-dependent timestamps (see Issue G). Another difficulty concerns the expandable objects, since their fields might be scattered over multiple blocks (see Issue H). During query processing, it is first necessary to locate the blocks that include the fields of the requested object, and then to assemble them by left-folding. Therefore, the fourth challenge is to investigate how query processing should be performed for each object type in order to efficiently compute the result of a query. Additionally, it is also necessary to investigate how a user can be supported in such a way that they can adequately formulate their queries.

The four research challenges mentioned above generally apply to any current blockchain system due to the conceptual design of blockchains. However, we expect that two factors will make these challenges even more difficult in the future, namely new blockchain architectures and legal restrictions.

7.5. New Blockchain Architectures

The fundamental architecture of a blockchain, as presented in Section 2, is constantly evolving. One trend that can be observed in this context is the so-called *sharding*. Sharding is introduced to address the typically low scalability of blockchains [135]. With blockchain sharding, the blockchain data is horizontally partitioned into shards where each shard is managed by a subset of the nodes in a network. One strategy in this regard can be to keep thematically related data in a common partition in order to create homogeneous partitions. A quite similar approach is known from traditional databases when a *snowflake schema* is applied. That is, data is divided among several tables in accordance with a specific dimension [136]. This makes queries regarding a certain topic highly efficient, since only a part of the data needs to be processed. However, the number of necessary joins increases if a comprehensive view on the entire data set is required. The same issue arises with sharding. As discussed in Issue A, blockchain systems are not designed to support joins efficiently. Moreover, the nodes that belong to an associated shard can only validate data they store. Therefore, when a join is made, the validation results from different shards must first be merged. For this reason, the data structures and block structures as well as the query processing must be adapted so that even complex `JOIN` operators can be executed efficiently.

Another emerging trend are the so-called *atomic cross-chain swaps*. Here, multiple parties exchange assets across multiple blockchains. Initially, this function was introduced so that different cryptocurrencies can be traded [137]. However, the exchanged assets are technically not limited to cryptocurrencies. That is, using cross-chain swaps, it is also possible to transfer data from one blockchain system to another [138]. Similar to sharding, this allows to create thematically homogeneous blockchains. Each blockchain provider would then only include data that corresponds to its respective topic. If necessary, external content can be imported from another blockchain via cross-chain swaps. Of course, this also results in the same challenges as with sharding, namely the high number of joins required to obtain a comprehensive view on the entire data set. Unlike sharding, where all partitions have at least the same technical foundation, cross-chain swap requires a wide variety of

blockchain systems to interoperate in order to support cross-chain join operations. Thus, the data structures and block structures must also be created in a cross-blockchain manner.

7.6. Legal Restrictions

As illustrated in Section 3.1, blockchains are becoming increasingly popular for storing sensitive data, such as health data. However, such private data are protected by data protection laws, such as the *General Data Protection Regulation (GDPR)* [139]. Although blockchains are ideal for the secure storage and distribution of sensitive data in terms of immutability and tamper-resistance, they are fundamentally in conflict with data privacy principles [140]. Special categories of personal data, such as health data, however, are subject to a particularly high degree of protection — here data subjects must be granted full control over their data. To this end, comprehensive adjustments to a blockchain are necessary [141]. In particular, the *right to be forgotten* is in conflict with the immutability of a blockchain, and the *right to restriction of processing* contradicts the fully decentralized distribution of data to nodes that manage them autonomously. Moreover, it is impossible for data subjects to exercise their right to data minimization against individual data processors, since the data are tamper-resistant available in a blockchain [142].

However, such adjustments to make a blockchain GDPR-compliant also have a significant impact on query processing in blockchains. These implications concern two aspects in particular. On the one hand, due to the right to be forgotten `DELETE` statements are required. In the context of blockchains, however, this is technically difficult not only due to immutability, but also because of expandable objects. If such an object has to be deleted, initially all components of the object have to be found. These components can be distributed arbitrarily over all blocks of the blockchain. To support `DELETE` statements efficiently, data structures and block structures are required that exceed auxiliary structures found in current blockchains systems significantly. On the other hand, the access control in blockchain systems must be considerably refined in order to grant data subjects the legally guaranteed control over their data. Data subjects must be able to make fine-grained decisions about who should have access to which data. As a consequence, queries regarding the change history of objects become much more complex in particular. If a user has restricted access to some of the changes, only, it must be resolved how a history query can be executed in this case without having to process the restricted data. Expandable objects constitute a special challenge in this respect as well, since they can only be queried and assembled if all components can be accessed. If this cannot be guaranteed due to access restrictions, the data models and also the query processing itself have to be revised.

8. Conclusions

Blockchains are considered the new go-to technology in many application domains to store data in an immutable and tamper-resistant manner while ensuring high availability. A blockchain, however, is rather a conceptual design than a specific embodiment of a technology. Therefore, there are different implementations of a blockchain, each with their respective advantages and disadvantages. To support query capabilities on blockchain data, there are currently two prevalent approaches:

The first approach is to store all data in the blockchain and then execute the queries on it. The advantages of this approach are that the data history is fully available, and the data are protected by being immutable and tamper-resistant. The disadvantage of this approach is that query processing requires sequential traversal of the blocks, since there are no index structures to improve the efficiency of query processing.

The second approach is to operate a database in parallel to the blockchain. This database maintains the world state. This way, SQL-like queries can be executed efficiently, which is this approach's advantage. Its disadvantage is that such a database does not provide the data history. As a consequence, temporal queries and state-based queries are not or at least insufficiently supported. Furthermore, the authenticity of this data is not guaranteed by the blockchain. To this end, an additional verification step is required.

Therefore, to unlock the full potential of the blockchain technology (i. e., security and data history combined with comprehensive query capabilities), many research efforts are still needed (e. g., in terms of developing new index and data access structures for blockchains). In particular, we identified four categories of current research challenges in this regard: data models, data structures, block structures, and query processing.

In summary, the importance of blockchain systems as a secure data store is undeniable for a digitized society. However, there are still many research questions to be addressed before blockchains can compete with traditional database systems in terms of query capabilities and efficiency.

Author Contributions: Conceptualization, C.G., D.P. and C.S.; methodology, D.P. and C.S.; software, D.P.; validation, C.G., D.P. and C.S.; formal analysis, D.P. and C.S.; investigation, D.P. and C.S.; resources, B.M.; data curation, D.P. and C.S.; writing — original draft preparation, D.P. and C.S.; writing — review and editing, C.G., B.M., D.P. and C.S.; visualization, D.P. and C.S.; supervision, B.M. and C.S.; project administration, B.M. and D.P.; funding acquisition, B.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by German Federal Ministry of Education and Research (BMBF) as part of the Software Campus program grant number 01IS17051.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: We thank the anonymous reviewers for their valuable comments and suggestions which helped us to improve the content and presentation of this paper.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Faroukhi, A.Z.; El Alaoui, I.; Gahi, Y.; Amine, A. Big data monetization throughout Big Data Value Chain: A comprehensive review. *J. Big Data* **2020**, *7*, 3:1–3:22.
2. Wiens, J.; Price, W.N.; Sjoding, M.W. Diagnosing bias in data-driven algorithms for healthcare. *Nat. Med.* **2020**, *26*, 25–26.
3. Saetta, S.; Caldarelli, V. How to increase the sustainability of the agri-food supply chain through innovations in 4.0 perspective: A first case study analysis. *Procedia Manuf.* **2020**, *42*, 333–336.
4. Ren, L.; Meng, Z.; Wang, X.; Zhang, L.; Yang, L.T. A Data-Driven Approach of Product Quality Prediction for Complex Production Systems. *IEEE Trans. Ind. Inform.* **2021**, *17*, 6457–6465.
5. Stach, C.; Bräcker, J.; Eichler, R.; Giebler, C.; Mitschang, B. Demand-Driven Data Provisioning in Data Lakes: BARENTS — A Tailorable Data Preparation Zone. In Proceedings of the 23rd International Conference on Information Integration and Web Intelligence, iiWAS '21, Linz, Austria, 29 November–1 December 2021; pp. 191–202.
6. Dièbea, B.; Rodrigues, J.J.; Diallo, O.; Ndoye, E.H.M.; Korotaev, V.V. Data management techniques for Internet of Things. *Mech. Syst. Signal Process.* **2020**, *138*, 106564:1–106564:19.
7. Pavlou, K.E.; Snodgrass, R.T. Forensic Analysis of Database Tampering. *ACM Trans. Database Syst.* **2008**, *33*, 1–47.
8. Iqbal, M.; Matulevičius, R. Blockchain as a Countermeasure Solution for Security Threats of Healthcare Applications. In Proceedings of the 19th Business Process Management Conference, BPM '21, Rome, Italy, 6–10 September 2021; pp. 67–84.
9. Chopade, R.; Pachghare, V. Data Tamper Detection from NoSQL Database in Forensic Environment. *J. Cyber Secur. Mobil.* **2021**, *10*, 421–450.
10. Nwosu, A.U.; Goyal, S.B.; Bedi, P. Blockchain Transforming Cyber-Attacks: Healthcare Industry. In Proceedings of the 11th International Conference on Innovations in Bio-Inspired Computing and Applications, IBICA '20, online, 16–18 December 2020; pp. 258–266.
11. Maity, M.; Tolooie, A.; Sinha, A.K.; Tiwari, M.K. Stochastic batch dispersion model to optimize traceability and enhance transparency using Blockchain. *Comput. Ind. Eng.* **2021**, *154*, 107134:1–107134:12.
12. Ge, C.; Liu, Z.; Fang, L. A blockchain based decentralized data security mechanism for the Internet of Things. *J. Parallel Distrib. Comput.* **2020**, *141*, 1–9.
13. Nakamoto, S. *Bitcoin: A Peer-to-Peer Electronic Cash System*; White Paper, Bitcoin; 2008.
14. Zhu, Y.; Zhang, Z.; Jin, C.; Zhou, A.; Qin, G.; Yang, Y. Towards Rich Query Blockchain Database. In Proceedings of the 29th ACM International Conference on Information & Knowledge Management, CIKM '20, Ireland (Virtual Event), 19–23 October 2020; pp. 3497–3500.

15. Ruan, P.; Anh Dinh, T.T.; Lin, Q.; Zhang, M.; Chen, G.; Chin Ooi, B. Revealing Every Story of Data in Blockchain Systems. *ACM SIGMOD Rec.* **2020**, *49*, 70–77.
16. Hellwig, D.; Karlic, G.; Huchzermeier, A. *Build Your Own Blockchain: A Practical Guide to Distributed Ledger Technology*; Management for Professionals; Springer Nature: Cham 2020.
17. Krishnan, S.; Balas, V.E.; Golden Julie, E.; Robinson, Y.H.; Balaji, S.; Kumar, R. (Eds.) *Handbook of Research on Blockchain Technology*; Academic Press: London, San Diego, Cambridge, Oxford 2020.
18. Merkle, R.C. A Digital Signature Based on a Conventional Encryption Function. In Proceedings of the 7th Conference on Advances in Cryptology, CRYPTO '87, Santa Barbara, California, USA, 16–20 August 1987; pp. 369–378.
19. Vujičić, D.; Jagodić, D.; Ranđić, S. Blockchain technology, bitcoin, and Ethereum: A brief overview. In Proceedings of the 2018 17th International Symposium INFOTEH-JAHORINA, INFOTEH '18, East Sarajevo, Bosnia and Herzegovina, 21–23 March 2018; pp. 1–6.
20. Yue, C.; Xie, Z.; Zhang, M.; Chen, G.; Ooi, B.C.; Wang, S.; Xiao, X. Analysis of Indexing Structures for Immutable Data. In Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data, SIGMOD '20, Portland, OR, USA, 14–19 June 2020; pp. 925–935.
21. Ruoti, S.; Kaiser, B.; Yerukhimovich, A.; Clark, J.; Cunningham, R. Blockchain Technology: What is It Good For? *Commun. ACM* **2019**, *63*, 46–53.
22. Nofer, M.; Gomber, P.; Hinz, O.; Schiereck, D. Blockchain. *Bus. Inf. Syst. Eng.* **2017**, *59*, 183–187.
23. Muzammal, M.; Qu, Q.; Nasrulin, B. Renovating blockchain with distributed databases: An open source system. *Future Gener. Comput. Syst.* **2019**, *90*, 105–117.
24. Dwork, C.; Naor, M. Pricing via Processing or Combatting Junk Mail. In Proceedings of the 12th Annual International Cryptology Conference, CRYPTO '92, Santa Barbara, California, USA, 16–20 August 1992; pp. 139–147.
25. Chowdhury, M.J.M.; Colman, A.; Kabir, M.A.; Han, J.; Sarda, P. 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 '18, New York, NY, USA, 1–3 August 2018; pp. 1348–1353.
26. Rehmani, M.H. Blockchain Technology and Database Management System. In *Blockchain Systems and Communication Networks: From Concepts to Implementation*; Springer International Publishing: Cham 2021; Chapter 2, pp. 15–22.
27. Chen, S.; Zhang, J.; Shi, R.; Yan, J.; Ke, Q. A Comparative Testing on Performance of Blockchain and Relational Database: Foundation for Applying Smart Technology into Current Business Systems. In Proceedings of the 6th International Conference on Distributed, Ambient, and Pervasive Interactions, DAPI '18, Las Vegas, NV, USA, 15–20 July 2018; pp. 21–34.
28. Ozdayi, M.S.; Kantarcioglu, M.; Malin, B. Leveraging blockchain for immutable logging and querying across multiple sites. *BMC Med. Genom.* **2020**, *13*, 82–88.
29. Eberhardt, J.; Tai, S. On or Off the Blockchain? Insights on Off-Chaining Computation and Data. In Proceedings of the 6th IFIP WG 2.14 European Conference on Service-Oriented and Cloud Computing, ESOCC '17, Oslo, Norway, 27–29 September 2017; pp. 3–15.
30. Lo, S.K.; Xu, X.; Chiam, Y.K.; Lu, Q. Evaluating Suitability of Applying Blockchain. In Proceedings of the 2017 22nd International Conference on Engineering of Complex Computer Systems, ICECCS '17, Fukuoka, Japan, 5–8 November 2017; pp. 158–161.
31. Alladi, T.; Chamola, V.; Rodrigues, J.J.P.C.; Kozlov, S.A. Blockchain in Smart Grids: A Review on Different Use Cases. *Sensors* **2019**, *19*, 4862.
32. Mollah, M.B.; Zhao, J.; Niyato, D.; Lam, K.Y.; Zhang, X.; Ghias, A.M.Y.M.; Koh, L.H.; Yang, L. Blockchain for Future Smart Grid: A Comprehensive Survey. *IEEE Internet Things J.* **2021**, *8*, 18–43.
33. Gaber, T.; Ahmed, A.; Mostafa, A. PrivDRM: A Privacy-Preserving Secure Digital Right Management System. In Proceedings of the Evaluation and Assessment in Software Engineering, EASE '20, Trondheim, Norway, 15–17 April 2020; pp. 481–486.
34. Hei, Y.; Liu, J.; Feng, H.; Li, D.; Liu, Y.; Wu, Q. Making MA-ABE fully accountable: A blockchain-based approach for secure digital right management. *Comput. Netw.* **2021**, *191*, 108029:1–108029:12.
35. Ren, Q.; Man, K.L.; Li, M.; Gao, B. Using Blockchain to Enhance and Optimize IoT-based Intelligent Traffic System. In Proceedings of the 2019 International Conference on Platform Technology and Service, PlatCon '19, Jeju, South Korea, 28–30 January 2019; pp. 1–4.
36. Wang, Q.; Ji, T.; Guo, Y.; Yu, L.; Chen, X.; Li, P. TrafficChain: A Blockchain-Based Secure and Privacy-Preserving Traffic Map. *IEEE Access* **2020**, *8*, 60598–60612.
37. Holderried, M.; Hoeper, A.; Holderried, F.; Heyne, N.; Nadalin, S.; Unger, O.; Ernst, C.; Guthoff, M. Attitude and potential benefits of modern information and communication technology use and telemedicine in cross-sectoral solid organ transplant care. *Sci. Rep.* **2021**, *11*, 9037:1–9037:9.
38. Hörbst, A.; Ammenwerth, E. Electronic health records: A systematic review on quality requirements. *Methods Inf. Med.* **2010**, *49*, 320–336.
39. Lupton, D. *The Quantified Self*; Polity Press: Malden, MA 2016.
40. Heinemann, L. Continuous Glucose Monitoring (CGM) or Blood Glucose Monitoring (BGM): Interactions and Implications. *J. Diabetes Sci. Technol.* **2018**, *12*, 873–879.
41. Isakadze, N.; Martin, S.S. How useful is the smartwatch ECG? *Trends Cardiovasc. Med.* **2020**, *30*, 442–448.

42. Stach, C.; Steimle, F.; Franco da Silva, A.C. TIROL: The Extensible Interconnectivity Layer for mHealth Applications. In Proceedings of the 23rd International Conference on Information and Software Technologies, ICIST '17, Druskininkai, Lithuania, 12–14 October 2017; pp. 190–202.
43. Pham, H.L.; Tran, T.H.; Nakashima, Y. A Secure Remote Healthcare System for Hospital Using Blockchain Smart Contract. In Proceedings of the 2018 IEEE Globecom Workshops, GC Wkshps '18, Abu Dhabi, United Arab Emirates, 9–13 December 2018; pp. 1–6.
44. Spanakis, e. g.; Bonomi, S.; Sfakianakis, S.; Santucci, G.; Lenti, S.; Sorella, M.; Tanasache, F.D.; Palleschi, A.; Ciccotelli, C.; Sakkalis, V.; Magalini, S. Cyber-attacks and threats for healthcare — A multi-layer thread analysis. In Proceedings of the 2020 42nd Annual International Conference of the IEEE Engineering in Medicine Biology Society, EMBC '20, Montreal, QC, Canada, 20–24 July 2020; pp. 5705–5708.
45. Ball, M.J.; Smith, C.; Bakalar, R.S. Personal health records: Empowering consumers. *J. Healthc. Inf. Manag.* **2007**, *21*, 76–86.
46. Peng, Z.; Xu, C.; Wang, H.; Huang, J.; Xu, J.; Chu, X. P^{2}B-Trace: Privacy-Preserving Blockchain-Based Contact Tracing to Combat Pandemics. In Proceedings of the 2021 International Conference on Management of Data, SIGMOD/PODS '21, China (Virtual Event), 20–25 June 2021; pp. 2389–2393.
47. De Aguiar, E.J.; Faiçal, B.S.; Krishnamachari, B.; Ueyama, J. A Survey of Blockchain-Based Strategies for Healthcare. *ACM Comput. Surv.* **2020**, *53*, 27:1–27:27.
48. Hasselgren, A.; Kralevska, K.; Gligoroski, D.; Pedersen, S.A.; Faxvaag, A. Blockchain in healthcare and health sciences — A scoping review. *Int. J. Med. Inform.* **2020**, *134*, 104040:1–104040:10.
49. Khatoon, A. A Blockchain-Based Smart Contract System for Healthcare Management. *Electronics* **2020**, *9*, 94.
50. Przytarski, D.; Stach, C.; Gritti, C.; Mitschang, B. A Blueprint for a Trustworthy Health Data Platform Encompassing IoT and Blockchain Technologies. In Proceedings of the ISCA 29th International Conference on Software Engineering and Data Engineering, SEDE '20, Las Vegas, NV, USA (Virtual Event), 19–21 October 2020; pp. 56–65.
51. Tanwar, S.; Parekh, K.; Evans, R. Blockchain-based electronic healthcare record system for healthcare 4.0 applications. *J. Inf. Secur. Appl.* **2020**, *50*, 102407:1–102407:13.
52. Smith, F. The influence of Amatino Manucci and Luca Pacioli. *BSHM Bull. J. Br. Soc. Hist. Math.* **2008**, *23*, 143–156.
53. Ellerman, D.P. The Mathematics of Double Entry Bookkeeping. *Math. Mag.* **1985**, *58*, 226–233.
54. Brandon, D. The BLOCKCHAIN: The Future of Business Information Systems? *Int. J. Acad. Bus. World* **2016**, *10*, 33–40.
55. Beck, R.; Avital, M.; Rossi, M.; Thatcher, J.B. Blockchain Technology in Business and Information Systems Research. *Bus. Inf. Syst. Eng.* **2017**, *59*, 381–384.
56. Carlin, T. Blockchain and the Journey Beyond Double Entry. *Aust. Account. Rev.* **2019**, *29*, 305–311.
57. Faccia, A.; Moşteanu, N.R.; Leonardo, L.P. Blockchain Hash, the Missing Axis of the Accounts to Settle the Triple Entry Bookkeeping System. In Proceedings of the 2020 12th International Conference on Information Management and Engineering, ICIME '20, Amsterdam, Netherlands, 16–18 September 2020; pp. 18–23.
58. Gökten, S.; Özdoğan, B. The Doors Are Opening for the New Pedigree: A Futuristic View for the Effects of Blockchain Technology on Accounting Applications. In *Digital Business Strategies in Blockchain Ecosystems: Transformational Design and Future of Global Business*; Hacioglu, U., Ed.; Springer International Publishing: Cham 2020; pp. 425–438.
59. Schmitz, J.; Leoni, G. Accounting and Auditing at the Time of Blockchain Technology: A Research Agenda. *Aust. Account. Rev.* **2019**, *29*, 331–342.
60. Sveistrup Søgaard, J. A blockchain-enabled platform for VAT settlement. *Int. J. Account. Inf. Syst.* **2021**, *40*, 100502:1–100502:18.
61. Zhang, Y.; Xiong, F.; Xie, Y.; Fan, X.; Gu, H. The Impact of Artificial Intelligence and Blockchain on the Accounting Profession. *IEEE Access* **2020**, *8*, 110461–110477.
62. Femenia-Ribera, C.; Mora-Navarro, G.; Martinez-Llario, J.C. Advances in the Coordination between the Cadastre and Land Registry. *Land* **2021**, *10*, 81.
63. Panda, S.K.; Mohammad, G.B.; Nandan Mohanty, S.; Sahoo, S. Smart contract-based land registry system to reduce frauds and time delay. *Secur. Priv.* **2021**, *4*, e172:1–e172:21.
64. Zhang, P.; Schmidt, D.C.; White, J.; Lenz, G. Blockchain Technology Use Cases in Healthcare. In *Advances in Computers*; Raj, P., Deka, G.C., Eds.; Elsevier: Amsterdam 2018; Chapter 1, pp. 1–41.
65. Peiró, N.N.; Martinez García, E.J. Blockchain and Land Registration Systems. *Eur. Prop. Law J.* **2017**, *6*, 296–320.
66. Vos, J. *Blockchain-Based Land Registry: Panacea, Illusion or Something in Between*; ELRA Annual Publication 7; European Land Registry Association: Brussels 2017.
67. Dabbagh, M.; Choo, K.K.R.; Beheshti, A.; Tahir, M.; Safa, N.S. A survey of empirical performance evaluation of permissioned blockchain platforms: Challenges and opportunities. *Comput. Secur.* **2021**, *100*, 102078:1–102078:13.
68. Benarous, L.; Kadri, B.; Bouridane, A.; Benkhelifa, E. Blockchain-based forgery resilient vehicle registration system. In *Transactions on Emerging Telecommunications Technologies*; John Wiley & Sons: Hoboken 2021; pp. 1–18.
69. Rosado, T.; Vasconcelos, A.; Correia, M. A Blockchain Use Case for Car Registration. In *Essentials of Blockchain Technology*; Li, K.C., Chen, X., Jiang, H., Bertino, E., Eds.; Chapman & Hall/CRC: New York 2019; Chapter 10, pp. 205–234.
70. Sahai, A.; Pandey, R. Smart Contract Definition for Land Registry in Blockchain. In Proceedings of the 2020 IEEE 9th International Conference on Communication Systems and Network Technologies, CSNT '20, Gwalior, India, 2020, 10–12 April 2020; pp. 230–235.

71. Shinde, D.; Padekar, S.; Raut, S.; Wasay, A.; Sambhare, S.S. Land Registry Using Blockchain — A Survey of existing systems and proposing a feasible solution. In Proceedings of the 2019 5th International Conference On Computing, Communication, Control And Automation, ICCUBEA '19, Pune, India, 19–21 September 2019; pp. 1–6.
72. Singh Yadav, A.; Singh Kushwaha, D. Query Optimization in a Blockchain-Based Land Registry Management System. *Ingénierie Systèmes D'Inf.* **2021**, *26*, 13–21.
73. Shah, R.; Meyer Goldstein, S.; Ward, P.T. Aligning supply chain management characteristics and interorganizational information system types: An exploratory study. *IEEE Trans. Eng. Manag.* **2002**, *49*, 282–292.
74. Nastasijević, I.; Lakićević, B.; Petrović, Z. Cold chain management in meat storage, distribution and retail: A review. *IOP Conf. Ser. Earth Environ. Sci.* **2017**, *85*, 012022:1–012022:10.
75. Tian, F. A supply chain traceability system for food safety based on HACCP, blockchain & Internet of things. In Proceedings of the 2017 International Conference on Service Systems and Service Management, ICSSSM '17, Dalian, China, 16–18 June 2017; pp. 1–6.
76. Stach, C.; Gritti, C.; Przytarski, D.; Mitschang, B. Trustworthy, Secure, and Privacy-aware Food Monitoring Enabled by Blockchains and the IoT. In Proceedings of the 2020 IEEE International Conference on Pervasive Computing and Communications Workshops, PerCom '20, Austin, TX, USA, 23–27 March 2020; pp. 50:1–50:4.
77. Fan, Y.; de Kleuver, C.; de Leeuw, S.; Behdani, B. Trading off cost, emission, and quality in cold chain design: A simulation approach. *Comput. Ind. Eng.* **2021**, *158*, 107442:1–107442:16.
78. Menon, K.N.; Thomas, K.; Thomas, J.; Titus, D.J.; James, D. ColdBlocks: Quality Assurance in Cold Chain Networks Using Blockchain and IoT. In Proceedings of the 2nd International Conference on Emerging Technologies in Data Mining and Information Security, IEMIS '20, Kolkata, India, 2–4 July 2020; pp. 781–789.
79. Duan, J.; Zhang, C.; Gong, Y.; Brown, S.; Li, Z. A Content-Analysis Based Literature Review in Blockchain Adoption within Food Supply Chain. *Int. J. Environ. Res. Public Health* **2020**, *17*, 1784.
80. Köhler, S.; Pizzol, M. Technology assessment of blockchain-based technologies in the food supply chain. *J. Clean. Prod.* **2020**, *269*, 122193:1–122193:10.
81. Kayikci, Y.; Subramanian, N.; Dora, M.; Singh Bhatia, M. Food supply chain in the era of Industry 4.0: Blockchain technology implementation opportunities and impediments from the perspective of people, process, performance, and technology. In *Production Planning & Control*; Taylor & Francis: Zug / Saint Helier 2020; pp. 1–21.
82. Shahid, A.; Almogren, A.; Javaid, N.; Al-Zahrani, F.A.; Zuair, M.; Alam, M. Blockchain-Based Agri-Food Supply Chain: A Complete Solution. *IEEE Access* **2020**, *8*, 69230–69243.
83. Zhang, X.; Sun, P.; Xu, J.; Wang, X.; Yu, J.; Zhao, Z.; Dong, Y. Blockchain-Based Safety Management System for the Grain Supply Chain. *IEEE Access* **2020**, *8*, 36398–36410.
84. Gritzalis, D.A. Principles and requirements for a secure e-voting system. *Comput. Secur.* **2002**, *21*, 539–556.
85. Gibson, J.P.; Krimmer, R.; Teague, V.; Pomares, J. A review of E-voting: The past, present and future. *Ann. Telecommun.* **2016**, *71*, 279–286.
86. Boyd, C.; Gjøsteen, K.; Gritti, C.; Haines, T. A Blind Coupon Mechanism Enabling Veto Voting over Unreliable Networks. In Proceedings of the 20th International Conference on Cryptology in India, INDOCRYPT '19, Hyderabad, India, 15–18 December 2019; pp. 250–270.
87. Haines, T.; Gritti, C. Improvements in Everlasting Privacy: Efficient and Secure Zero Knowledge Proofs. In Proceedings of the 4th International Joint Conference on Electronic Voting, E-Vote-ID '19, Bregenz, Austria, 1–4 October 2019; pp. 116–133.
88. Moura, T.; Gomes, A. Blockchain Voting and Its Effects on Election Transparency and Voter Confidence. In Proceedings of the 18th Annual International Conference on Digital Government Research, dg.o '17, Staten Island, NY, USA, 7–9 June 2017; pp. 574–575.
89. Wani, S.; Imthiyas, M.; Almohamedh, H.; Alhamed, K.M.; Almotairi, S.; Gulzar, Y. Distributed Denial of Service (DDoS) Mitigation Using Blockchain–A Comprehensive Insight. *Symmetry* **2021**, *13*, 227.
90. Hanifatunnisa, R.; Rahardjo, B. Blockchain based e-voting recording system design. In Proceedings of the 2017 11th International Conference on Telecommunication Systems Services and Applications, TSSA '17, Lombok, Indonesia, 26–27 October 2017; pp. 1–6.
91. Hjálmarsson, F.Þ.; Hreiðarsson, G.K.; Hamdaqa, M.; Hjálmtýsson, G. Blockchain-Based E-Voting System. In Proceedings of the 2018 IEEE 11th International Conference on Cloud Computing, CLOUD '18, San Francisco, CA, USA, 2–7 July 2018; pp. 983–986.
92. Kshetri, N.; Voas, J. Blockchain-Enabled E-Voting. *IEEE Softw.* **2018**, *35*, 95–99.
93. Ruparel, H.; Hosatti, S.; Shirole, M.; Bhirud, S. Secure Voting for Democratic Elections: A Blockchain-Based Approach. In Proceedings of the 2020 International Conference on Communication, Computing and Electronics Systems, ICCCES '20, Coimbatore, India, 21–22 October 2020; pp. 615–628.
94. Wang, B.; Sun, J.; He, Y.; Pang, D.; Lu, N. Large-scale Election Based On Blockchain. *Procedia Comput. Sci.* **2018**, *129*, 234–237.
95. Buchmann, A.; Koldehofe, B. Complex Event Processing. *IT-Inf. Technol.* **2009**, *51*, 241–242.
96. Pratama, F.A.; Mutijarsa, K. Query Support for Data Processing and Analysis on Ethereum Blockchain. In Proceedings of the 2018 International Symposium on Electronics and Smart Devices, ISESD '18, Bandung, Indonesia, 23–24 October 2018; pp. 1–5.
97. Zhu, Y.; Zhang, Z.; Jin, C.; Zhou, A.; Yan, Y. SEBDB: Semantics Empowered BlockChain DataBase. In Proceedings of the 2019 IEEE 35th International Conference on Data Engineering, ICDE '19, Macao, China, 8–11 April 2019; pp. 1820–1831.

98. Heuer, A.; Scholl, M.H. Principles of Object-Oriented Query Languages. In *Datenbanksysteme in Büro, Technik und Wissenschaft*; Appelrath, H.J., Ed.; Springer: Berlin/Heidelberg, Germany, 1991; pp. 178–197.
99. Libkin, L. Expressive power of SQL. *Theor. Comput. Sci.* **2003**, *296*, 379–404.
100. Klyne, G.; Newman, C. *Date and Time on the Internet: Timestamps*; Standards Track RFC 3339, July; IETF — Network Working Group: Reston / Geneva 2002.
101. Androulaki, E.; Barger, A.; Bortnikov, V.; Cachin, C.; Christidis, K.; De Caro, A.; Enyeart, D.; Ferris, C.; Laventman, G.; Manevich, Y.; et al. Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains. In Proceedings of the Thirteenth EuroSys Conference, EuroSys '18, Porto, Portugal, 23–26 April 2018; pp. 30:1–30:15.
102. Ruan, P.; Dinh, T.T.A.; Loghin, D.; Zhang, M.; Chen, G.; Lin, Q.; Ooi, B.C. Blockchains vs. Distributed Databases: Dichotomy and Fusion. In Proceedings of the 2021 International Conference on Management of Data, SIGMOD/PODS '21, China (Virtual Event), 20–25 June 2021; pp. 1504–1517.
103. El-Hindi, M.; Binnig, C.; Arasu, A.; Kossmann, D.; Ramamurthy, R. BlockchainDB: A Shared Database on Blockchains. *Proc. VLDB Endow.* **2019**, *12*, 1597–1609.
104. Peng, Y.; Du, M.; Li, F.; Cheng, R.; Song, D. FalconDB: Blockchain-Based Collaborative Database. In Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data, SIGMOD '20, Portland, OR, USA, 14–19 June 2020; pp. 637–652.
105. Allen, L.; Antonopoulos, P.; Arasu, A.; Gehrke, J.; Hammer, J.; Hunter, J.; Kaushik, R.; Kossmann, D.; Lee, J.; Ramamurthy, R.; Setty, S.; Szymaszek, J.; van Renen, A.; Venkatesan, R. Veritas: Shared Verifiable Databases and Tables in the Cloud. In Proceedings of the 9th Biennial Conference on Innovative Data Systems Research, CIDR '19, Asilomar, CA, USA, 13–16 January 2019; pp. 111:1–111:9.
106. BigchainDB GmbH. *BigchainDB 2.0: The Blockchain Database*; White Paper; BigchainDB GmbH: Berlin 2018.
107. Nathan, S.; Govindarajan, C.; Saraf, A.; Sethi, M.; Jayachandran, P. Blockchain Meets Database: Design and Implementation of a Blockchain Relational Database. *Proc. VLDB Endow.* **2019**, *12*, 1539–1552.
108. Schuhknecht, F.M.; Sharma, A.; Dittrich, J.; Agrawal, D. chainifyDB: How to get rid of your Blockchain and use your DBMS instead. In Proceedings of the 11th Annual Conference on Innovative Data Systems Research, CIDR '21, online, 11–15 January 2021; pp. 4:1–4:10.
109. Eisenberg, A.; Melton, J. SQL: 1999, Formerly Known as SQL3. *ACM SIGMOD Rec.* **1999**, *28*, 131–138.
110. Wood, G. *Ethereum: A Secure Decentralised Generalised Transaction Ledger*; Ethereum Yellow Paper Berlin Version 888949c, Ethereum Project; 2021.
111. Han, J.; Kim, H.; Eom, H.; Coignard, J.; Wu, K.; Son, Y. Enabling SQL-Query Processing for Ethereum-based Blockchain Systems. In Proceedings of the 9th International Conference on Web Intelligence, Mining and Semantics, WIMS '19, Seoul, Republic of Korea, 26–28 June 2019; pp. 9:1–9:7.
112. Tong, X.; Tang, H.; Jiang, N.; Fan, W.; Gao, Y.; Deng, S.; Zhang, Z.; Jin, C.; Yang, Y.; Qin, G. SQL-Middleware: Enabling the Blockchain with SQL. In Proceedings of the 26th International Conference on Database Systems for Advanced Applications, DASFAA '21, Taipei, Taiwan, 11–14 April 2021; pp. 622–626.
113. Li, Y.; Zheng, K.; Yan, Y.; Liu, Q.; Zhou, X. EtherQL: A Query Layer for Blockchain System. In Proceedings of the 22nd International Conference on Database Systems for Advanced Applications, DASFAA '17, Suzhou, China, 27–30 March 2017; pp. 556–567.
114. Bragagnolo, S.; Marra, M.; Polito, G.; Gonzalez Boix, E. Towards Scalable Blockchain Analysis. In Proceedings of the 2019 IEEE/ACM 2nd International Workshop on Emerging Trends in Software Engineering for Blockchain, WETSEB '19, Montreal, QC, Canada, 27 May 2019; pp. 1–7.
115. Xu, C.; Zhang, C.; Xu, J. vChain: Enabling Verifiable Boolean Range Queries over Blockchain Databases. In Proceedings of the 2019 International Conference on Management of Data, SIGMOD '19, Amsterdam, Netherlands, 30 June–5 July 2019; pp. 141–158.
116. Xing, X.; Chen, Y.; Li, T.; Xin, Y.; Sun, H. A blockchain index structure based on subchain query. *J. Cloud Comput.* **2021**, *10*, 52:1–52:11.
117. Jia, D.Y.; Xin, J.C.; Wang, Z.Q.; Lei, H.; Wang, G.R. SE-Chain: A Scalable Storage and Efficient Retrieval Model for Blockchain. *J. Comput. Sci. Technol.* **2021**, *36*, 693–706.
118. Peng, Z.; Wu, H.; Xiao, B.; Guo, S. VQL: Providing Query Efficiency and Data Authenticity in Blockchain Systems. In Proceedings of the 2019 IEEE 35th International Conference on Data Engineering Workshops, ICDEW 19, Macao, China, 8–12 April 2019; pp. 1–6.
119. Wu, H.; Peng, Z.; Guo, S.; Yang, Y.; Xiao, B. VQL: Efficient and Verifiable Cloud Query Services for Blockchain Systems. *IEEE Trans. Parallel Distrib. Syst.* **2022**, *33*, 1393–1406.
120. Gritti, C.; Önen, M.; Molva, R. Privacy-Preserving Delegable Authentication in the Internet of Things. In Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing, SAC '19, Limassol, Cyprus, 8–12 April 2019; pp. 861–869.
121. Stach, C.; Bräcker, J.; Eichler, R.; Giebler, C.; Gritti, C. How to Provide High-Utility Time Series Data in a Privacy-Aware Manner: A VAULT to Manage Time Series Data. *Int. J. Adv. Secur.* **2020**, *13*, 88–108.
122. Wortner, P.; Schubotz, M.; Breitinger, C.; Leible, S.; Gipp, B. Securing the Integrity of Time Series Data in Open Science Projects using Blockchain-based Trusted Timestamping. In Proceedings of the Workshop on Web Archiving and Digital Libraries held in conjunction with the 18th ACM/IEEE Joint Conference on Digital Libraries, WADL '19, Champaign, Illinois, 2 June 2019; pp. 2:1–2:3.

123. Dhanush, G.A.; Raj, K.S.; Kumar, P. Blockchain Aided Predictive Time Series Analysis in Supply Chain System. In Proceedings of the 2021 2nd International Conference on Electrical and Electronics Engineering, ICEEE '21, NCR New Delhi, India, 2–3 January 2021; pp. 913–925.
124. Yu, Z.; Cai, Y.; Hong, W. A Storage Architecture of Blockchain for Time-Series Data. In Proceedings of the 2019 2nd International Conference on Hot Information-Centric Networking, HotICN '19, Chongqing, China, 13–15 December 2019; pp. 90–91.
125. Qu, Q.; Nurgaliev, I.; Muzammal, M.; Jensen, C.S.; Fan, J. On spatio-temporal blockchain query processing. *Future Gener. Comput. Syst.* **2019**, *98*, 208–218.
126. Nurgaliev, I.; Muzammal, M.; Qu, Q. Enabling Blockchain for Efficient Spatio-Temporal Query Processing. In Proceedings of the 19th International Conference on Web Information Systems Engineering, WISE '18, Dubai, United Arab Emirates, 12–15 November 2018; pp. 36–51.
127. Zhang, Y.; Genkin, D.; Katz, J.; Papadopoulos, D.; Papamanthou, C. vSQL: Verifying Arbitrary SQL Queries over Dynamic Outsourced Databases. In Proceedings of the 2017 IEEE Symposium on Security and Privacy, SP '17, San Jose, CA, USA, 22–26 May 2017; pp. 863–880.
128. Zhang, M.; Xie, Z.; Yue, C.; Zhong, Z. Spitz: A Verifiable Database System. *Proc. VLDB Endow.* **2020**, *13*, 3449–3460.
129. Wang, S.; Dinh, T.T.A.; Lin, Q.; Xie, Z.; Zhang, M.; Cai, Q.; Chen, G.; Ooi, B.C.; Ruan, P. Forkbase: An Efficient Storage Engine for Blockchain and Forkable Applications. *Proc. VLDB Endow.* **2018**, *11*, 1137–1150.
130. Zhou, W.; Cai, Y.; Peng, Y.; Wang, S.; Ma, K.; Li, F. VeriDB: An SGX-Based Verifiable Database. In Proceedings of the 2021 International Conference on Management of Data, SIGMOD/PODS '21, China (Virtual Event), 20–25 June 2021; pp. 2182–2194.
131. McKeen, F.; Alexandrovich, I.; Anati, I.; Caspi, D.; Johnson, S.; Leslie-Hurd, R.; Rozas, C. Intel® Software Guard Extensions (Intel® SGX) Support for Dynamic Memory Management Inside an Enclave. In Proceedings of the Hardware and Architectural Support for Security and Privacy 2016, HASP '16, Seoul, Republic of Korea, 18 June 2016; pp. 1–9.
132. Li, F.; Hadjieleftheriou, M.; Kollios, G.; Reyzin, L. Dynamic Authenticated Index Structures for Outsourced Databases. In Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data, SIGMOD '06, Chicago, IL, USA, 27–29 June 2006; pp. 121–132.
133. Przytarski, D. Using Triples as the Data Model for Blockchain Systems. In Proceedings of the Blockchain enabled Semantic Web Workshop and Contextualized Knowledge Graphs Workshop co-located with the 18th International Semantic Web Conference, BlockSW/CKG@ISWC '19, Auckland, New Zealand, 26–30 October 2019; pp. 1–2.
134. Neumann, T.; Weikum, G. The RDF-3X engine for scalable management of RDF data. *VLDB J.* **2010**, *19*, 91–113.
135. Dang, H.; Dinh, T.T.A.; Loghin, D.; Chang, E.C.; Lin, Q.; Ooi, B.C. Towards Scaling Blockchain Systems via Sharding. In Proceedings of the 2019 International Conference on Management of Data, SIGMOD '19, Amsterdam, Netherlands, 30 June–5 July 2019; pp. 123–140.
136. Levene, M.; Loizou, G. Why is the snowflake schema a good data warehouse design? *Inf. Syst.* **2003**, *28*, 225–240.
137. Herlihy, M. Atomic Cross-Chain Swaps. In Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing, PODC '18, Egham, United Kingdom, 23–27 July 2018; pp. 245–254.
138. Xu, J.; Ackerer, D.; Dubovitskaya, A. A Game-Theoretic Analysis of Cross-Chain Atomic Swaps with HTLCs. In Proceedings of the 2021 IEEE 41st International Conference on Distributed Computing Systems, ICDCS '21, Washington, DC, USA, 7–10 July 2021; pp. 584–594.
139. European Parliament and Council of the European Union. Regulation on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (Data Protection Directive). Legislative Acts L119. *Off. J. Eur. Union* **2016**.
140. Hofman, D.; Lemieux, V.L.; Joo, A.; Alves Batista, D. "The margin between the edge of the world and infinite possibility": Blockchain, GDPR and information governances. *Rec. Manag. J.* **2019**, *29*, 240–257.
141. Shi, S.; He, D.; Li, L.; Kumar, N.; Khan, M.K.; Choo, K.K.R. Applications of blockchain in ensuring the security and privacy of electronic health record systems: A survey. *Comput. Secur.* **2020**, *97*, 101966:1–101966:20.
142. Tatar, U.; Gokce, Y.; Nussbaum, B. Law versus technology: Blockchain, GDPR, and tough tradeoffs. *Comput. Law Secur. Rev.* **2020**, *38*, 105454:1–105454:11.