INSTITUT FÜR INFORMATIK
DER LUDWIG–MAXIMILIANS–UNIVERSITÄT MÜNCHEN

Bachelor's Thesis

Technology Comparison for long-term Retention of Contract Data

Luca Geiger

INSTITUT FÜR INFORMATIK
DER LUDWIG–MAXIMILIANS–UNIVERSITÄT MÜNCHEN

Bachelor's Thesis

Technology Comparison for long-term Retention of Contract Data

Luca Geiger

Aufgabensteller:	Prof. Dr. Dieter Kranzlmüller
Betreuer:	Maximilian Höb Peter Blenninger
Abgabetermin:	9. Januar 2020

Hiermit versichere ich, dass ich die vorliegende Bachelorarbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

München, den 9. Januar 2020

. . . . . .
(Unterschrift des Kandidaten)

Abstract

With the emergent development of blockchain technology, the number and variety of applications increases. Businesses, such as life insurance companies, are considering using it as their storage solution. It has potential benefits over the traditional option of relational databases like immutable data. For life insurance companies a significant part of IT projects is data migration. In the worst case, contract durations can be up to 120 years. This thesis helps deciding if blockchain is a suitable candidate for long retention of contracts. Mandatory technology features of data storage and native data migration will be collected. Based on these, blockchain technology will be compared with relational databases management systems. To select the most suitable blockchain system, several approaches are compared regarding their data model. Hyperledger Fabric determined to be the best choice. Since the comparison focused on the use case of this thesis relational databases show clear advantages over the blockchain technology Hyperledger Fabric.

Technologievergleich zur Langzeitspeicherung von Vertragsdaten

Abstract

Mit der fortschreitenden Entwicklung der Blockchain-Technologie steigt die Anzahl und Vielfalt der Anwendungen. Unternehmen, wie z.B. Lebensversicherungen, erwägen ihren Einsatz als Speicherlösung. Diese Technologie hat potentielle Vorteile gegenüber der traditionellen Wahl von relationalen Datenbanken wie z.B. der Unveränderlichkeit von Daten. Für Lebensversicherungsunternehmen ist ein wesentlicher Teil der IT-Projekte die Datenmigration. Im schlimmsten Fall können die Vertragslaufzeiten bis zu 120 Jahre betragen. Diese Arbeit hilft bei der Entscheidung, ob Blockchain ein geeigneter Kandidat als eine Langzeitspeicherlösung ist. Es werden relevante Anforderungen an die Technologien anhand der Fähigkeiten zur Datenspeicherung und der nativen Datenmigration erhoben. Darauf aufbauend wird Blockchain-Technologie mit relationalen Datenbankmanagementsystemen verglichen. Zur Auswahl des am besten geeigneten Blockchain-Systems werden mehrere Ansätze hinsichtlich ihres Datenmodells verglichen. Hyperledger Fabric ermittelt sich als die beste Wahl. Da der Vergleich auf den Anwendungsfall dieser Arbeit fokussiert ist, zeigen relationale Datenbanken deutliche Vorteile gegenüber der Blockchain-Technologie Hyperledger Fabric.

Contents

1 Introduction

More and more people develop an active interest in the topic of cryptocurrencies like Bitcoin. Many speculate with it to make profit almost like speculating on the stock exchange. The possession of cryptocurrencies is especially helpful for people living in countries with an unstable national currency, due to its independence from it.

The increasing curiosity in cryptocurrencies has attracted the attention of enterprises to its underlaying technology which is blockchain. Companies tend to use state-of-the-art technology to comply with highest information security standards. These want to find out whether blockchain can be considered as one of these. Systems that use blockchain could bring increased security over using other storage approaches. It is designed to perform in an adversial environment. This means that the network contains potentially malicious participants. The data structure blockchain implies that it is tamper-evident as each block contains a cryptographic hash generated about a subset of its payload and the hash of its predecessor block in the chain. To generate the hash, mostly a computationally intensive problem has to be solved. To make a change to a block, hence the data, every subsequent block needs to be updated to make it unnoticeable. Everyone can check if the integrity of the blockchain has been violated. Changes are noticeable as the hashes of the subsequent blocks would have to be recalculated.

Another benefit of blockchain is that the data can be made immutable[1] or tamper-proof. The blockchain is distributed across a network of nodes. Every node has a copy of it. A consensus algorithm controls concurrent changes and determines which changes to the blockchain are valid.

Use cases build upon the utilization of the blockchain as an immutable transaction log. Some applications have already been found, for example the tracking of physical or intangible data like monetary transactions, votes, health data, patents etcetera.[ØUJ17] Although the number of possible applications seems to be limited by the structure of the blockchain, as it mainly serves a as ledger, companies as well as researchers are still investigating the viability of the technology.

One research area investigates the viability of blockchain technology as a database. Companies have great interest in finding out whether blockchain still is just a hype or brings actual advantages over other storage technologies. Traditionally relational databases, or sometimes NoSQL databases, are chosen for this purpose.

A reason why relational databases are used is because it is a mature technology and mostly intuitive to use. It brings a lot of features amongst the creation, reading, updating and deleting of data. It is made to store structured data in form of tables but also has the option to store binary files or data formatted in XML. It also provides complex database queries and performance. There exists a variety of relational database management systems to satisfy many different needs e.g. sophisticated query mechanisms or high fault tolerance.

[1] Immutability in this context means that it may take several years for attackers to get access to the data.

Within the scope of this work relational database management systems are compared to blockchain technology by their data storing and data migration capabilities. One of the many different blockchain systems has to be utilized for the comparison. For this, a concept for storing data in a blockchain environment is developed. The outcome is that Hyperledger Fabric is the most suitable one for this use case which is a burial insurance. Hence requirements are implied which are the basis of the technology comparison. The burial insurance is a product of a use case life insurance company. One of the challenges of life insurance companies in general is the long lifetime of contracts. Some contracts last up to 120 years. Therefore one of the biggest part of IT projects is data migrations. These have to be performed every 8-10 years according to the company. The reasons for migrations for example are changes to business processes. This is also a consequence of legislative changes, e.g. the EU GDPR in 2018.[EUd] Other reasons for data migrations are changes to the hardware or operating system, major releases or even that the software vendor stops the support of its products.

In this work it is presented that relational database management systems are better than Hyperledger Fabric for this use case. The drawback of HLF is that it cannot entirely delete data and that it does not provide native options to migrate data. Both of these are crucial features in the context of long enduring contracts. RDBMS is more mature in contrast to the relatively new blockchain systems like HLF. Since the environment running the storage technology does not contain malicious users, the ability to operate on such is not an advantage over RDBMS.

The structure of this thesis is as follows:

Chapter 2 compares similar technology comparisons. It shows that there is no published work dealing with the objective as this work. In chapter 3 a requirements analysis is used to collect features that each of the technologies shall and should have. The storage technologies are presented in chapter 4. In chapter 5 the actual comparison will be conducted based on the requirements and evaluated. By this, trade-offs of each technology in the scope of a burial insurance contract will be presented. Chapter 6 presents a conclusion and gives an outlook for the future.

2 Related Work

This chapter will evaluate published work dealing with the objective of comparing the storage technologies of relational databases and blockchain. In section 2.1 three research papers will be presented that involve this kind of technology comparison. In section 2.2 the related work will be evaluated.

2.1 Cross-Technology Comparison

2.1.1 Comparison of MySQL and the Ethereum Blockchain

The authors in *Data Management: Relational vs Blockchain Databases*[CMC19] compare the Ethereum blockchain[1] with the relational database management system of MySQL. For the technology comparison four aspects, which are evaluated by conducting an experiment of an energy trading scenario, are investigated. The aspects are *defining data structures* to represent buyers and sellers in the trading scenario, *deploying these data structures* for each of the technologies, *how the data storage works* and *how to traverse data*. The authors iterate over the aspects which are to be compared in the context of the energy trading scenario for each of the technologies. For this, two backend databases are implemented.

The iteration for MySQL is straightforward as every of the four aspects is fully supported. The energy trading scenario is modeled via the Entity-Relationship (ER) model and implemented by using SQL commands. The described iteration is also applied to the Ethereum blockchain. It is not straightforward because smart contracts are used as a data structure. They need to be designed and deployed to the blockchain. For this purpose the authors developed a single-page application. This application sends transactions to a test network for Ethereum which replicates the main network. The execution of transaction of the 'real' Ethereum network involves costs wherefore the test network is used because it provisions free funds to accounts. The transactions are used to modify and add entries to the smart contracts, which are similar to the tables in a relational database for this purpose. Querying the blockchain is difficult because it does not provide a sophisticated query mechanism to traverse data.

The authors then discuss technological limitations of relational databases and blockchains which shows trade-offs for each of the technologies in this scenario.

2.1.2 Comparison of Blockchain, Relational Databases, and Google Sheets

The authors of *Blockchain: business' next new "It" technology—a comparison of blockchain, relational databases, and Google Sheets*[MA19] investigate the applicability of blockchain, relational databases and Google Sheets[2] to meeting the business needs of an accounting organization. The technologies are evaluated using a set of criteria which is essential to

[1] `https://www.ethereum.org/` (last access: 06.01.2020)
[2] `https://www.google.com/intl/en/sheets/about/` (last access: 06.01.2020)

most business operations. The comparison comprises performance, use cases, privacy and security, access to use and more. Hence, capabilities and shortfalls of each technology are identified.

2.1.3 Blockchain Relational Database

In the paper *Blockchain Meets Database: Design and Implementation of a Blockchain Relational Database*[NGS+19] a decentralized replicated relational databases with blockchain features ('target state') is designed and implemented. The starting point is the identification of similarities or similar concepts and features between relational database and blockchain. Thereof a list of needed improvements to get to the target state is concluded. These enhancing features are implemented using PostgreSQL[3]. For this, the authors develop new components or modify existing ones. In the evaluation the focus lays on performance, measured in terms of throughput and latency, and cost which is measured in terms of resource utilization like CPU, memory, disk and network bandwidth.

2.2 Result

The paper described in subsection 2.1.2 compares different aspects of blockchain, relational databases and Google Sheets but not explicitly the data models of the technologies. However, the paper provides capabilities and shortcomings of relational databases and blockchain in a business context. These are taking into consideration for the evaluation in this work.

Within the scope of the paper described in subsection 2.1.3 the focus on the design of a relational database enhanced with blockchain features. The authors pursue the objective to develop an implementation using PostgreSQL and evaluate it by analyzing performance and cost metrics. Because the focus does not lay on an actual (data model) comparison, the paper is not taken into consideration for the concept development in this thesis.

There is a similar comparison approach as used in this work which is described in 2.1.1. The difference is that the authors of that paper develop an implementation. For that, exclusively MySQL and Ethereum is used. The authors do not refer to or compare other distributed ledger technologies. The data model of the technologies is a main focus of the comparison. In contrast, in this thesis a differentiation between three blockchain systems is made (see 4.2) based on the used data model.

In summary it can be said that at the time of this research there is no published work dealing with the aim of investigating the viability of RDBMS and blockchain systems for the use case of storing a long enduring (burial) insurance contract.

[3] `https://www.postgresql.org/` (last access: 06.01.2020)

3 Requirements Analysis

The output of this chapter is a list of all relevant requirements (see 3.5). A requirement describes an aspect which each of the introduced technology (see 4) needs to have or should have in the scope of a burial insurance contract, depending on its importance. The importance is discussed in chapter 5. They are gathered from conducted expert interviews with employees of the use case company. The use case is described in section 3.0.1. The analysis in this chapter is non-exhaustive meaning that not all possible requirements are considered.

A method to classify requirements is to distinguish between functional and non-functional requirements.[CNYM12, p. 6] However, it is not used because some requirements do not fit the assignment or it is not clear. Another method is implied by the international standard for software product quality ISO 25010 which presents eight different subdivisions for software quality.[ISO11] Although it provides perspectives on how to structure the requirements, the method of grouping them by the source they are taken from is used because it fits best for this work. The sources are general terms and therefore allows a top-down view to the requirements. The sources are the contract with the business model (see 3.1), information technology security and data privacy objectives (see 3.3), legal requirements or restrictions (see 3.2), the usability (see 3.5.4) and technological changes (see 3.4). The latter are changes which occur during the long time a contract is "alive" in the context of the infrastructure in which the storage system is executed.

3.0.1 Use Case Description

The use case is a burial insurance. It is a type of of funeral expense life insurance policy covering the cost of funeral or cremation expenses. These are several thousand euros. The paid-in capital is paid into an insurance policy so that it cannot be misused. For example a 50 year old man takes out this insurance. His life expectancy is about 85 years. Therefore the contract runs for 35 years. The policy holder, which for instance is the insured person himself, pays in a premium every month until he dies or the contract is withdrawn/revoked. When the insured person dies the contract terminates and the beneficiary receives the payout.

3.1 Contract Model

The reason for choosing a burial insurance is its relatively low complexity. The complexity of an insurance contract mostly increases by the number of inputs, involved roles and amount of involved business processes. The process after concluding a burial insurance contract between the life insurance company and a person is mapped to the contract model. It is presumed that the referred data already has been digitalized[1].

[1]Note that, life insurance companies have to store the actual contracts for a certain amount of time due to, for example, tax laws or insurance contract laws.

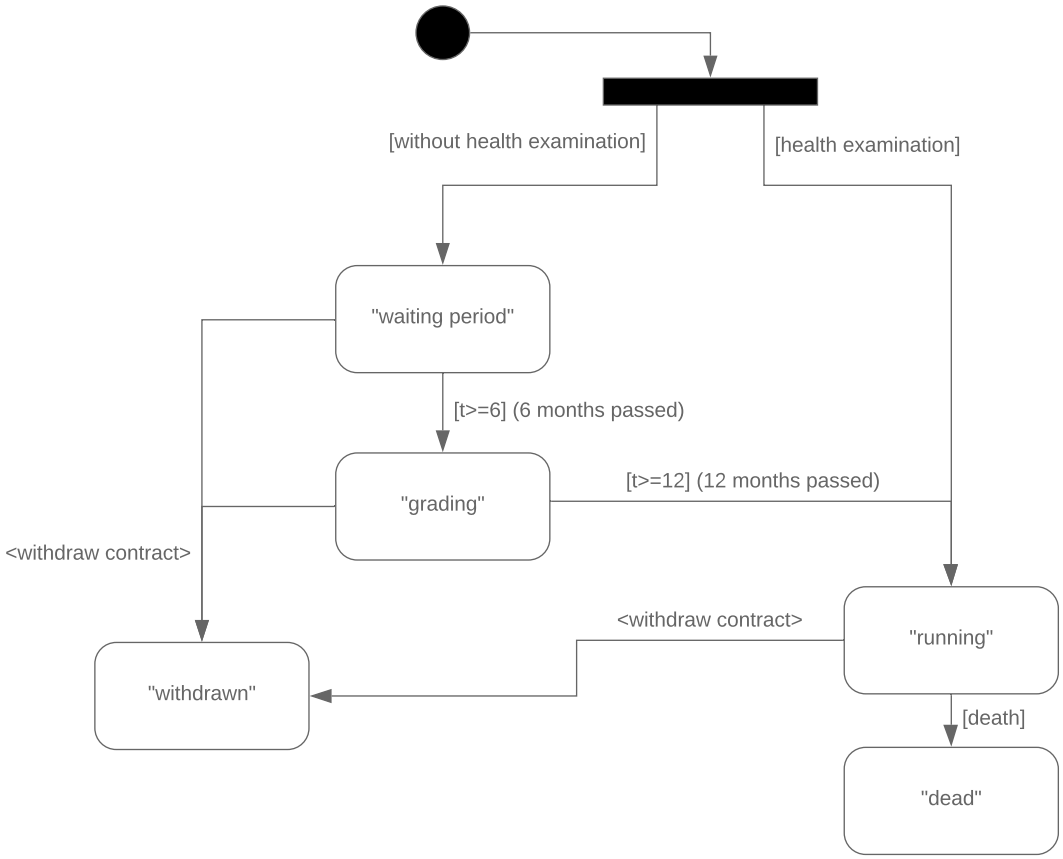

Figure 3.1: State model overview of the burial insurance contract model

3.1.1 Contract Life Cycle

To understand the next two subsections, the life cycle of the burial insurance contract is explained. Therefore a state model is defined. The model is illustrated in figure 3.1. A contract is within a state[2] at any time. They are represented by the rounded rectangles.

Before a person can conclude this type of contract, he or she has the option to make a health examination. If the examination is chosen, the contract will be in the running state. If not, it is in the waiting period state. This distinction is made to take the risk of the person to die shortly after concluding the insurance into account. If the insured person dies within the waiting period, the paid-in capital minus a administrative fee is paid out. The waiting period in this case lasts six months and the state after six months is the grading state. The remaining in these states depends on the age of the insured person. The calculation in the grading state works as follows: if the insured person dies after seven months, 6/12 of the payout will be paid. If he dies in the eighth month, 7/12 of the payout will be paid etc. This continues until one year after concluding the contract, or six month after leaving the waiting period, because then the full payout (12/12) is paid. After twelve months the contract is in the running state (without health examination). The next state would be the dead state if

[2]This is a (simplified) mapping of the real-world contract model.

the insured person dies. The contract can be withdrawn at any time[3]. If this happens, the contract is canceled and a guaranteed amount (guarantee values), calculated according to the already paid-in capital, is paid out to the policyholder. The withdrawn and dead states are final states.

3.1.2 Accounting Transactions

During the lifetime of a contract accounting transactions (AT) occur. They are events defined by their real-world trigger events. A distinction is made between scheduled and event based accounting transactions. Scheduled AT mostly are triggered by a certain amount of time which has passed. Event based AT mostly involve interaction of the insured person or policy holder. An event could be that the insured person moves to another city. AT initiate one or more business processes which indicate what actions or rather changes to data have to be performed. Only a subset of all accounting transactions are considered in this use case.

The AT are state transitions in the state model (see 3.1) Scheduled AT are recognized by arrows labeled with text inside angle brackets ("< ... >"). The labels refer to actions performed by an human in the real world, like the withdrawal of the contract. Event based AT are recognized by arrows labeled with conditions inside square brackets ("[...]"). The black rectangle represents a mutual exclusive condition, meaning that only one outgoing transition is possible at a time. The term $t >= 6$ is a temporal condition where t is the time in months.

3.1.3 Input Data

The input data refers to the data needed to conclude contracts. Customers can choose the type of insurance and additional options. Information about the insured person has to be specified to and saved by the life insurance company. This includes personal data like the date of birth, the address etcetera, information about the chosen insurance with selected options, administrative data and data to help making the contract machine readable needs to be stored. The input data is subdivided to contract and process data by the purpose of the data[4].

Process Data

The process data is needed for the verification of the contract a customer wants to conclude. For example it includes information like whether the to be insured person is politically exposed or the date of receipt/acknowledgment. This information does not change during the lifetime of the contract.

Non-Technical Contract Data

In contrast, contract data may change over time. It is subdivided to non-technical and technical contract data. The non-technical contract data contains organizational/administrative

[3]To revoke or withdraw the contract has consequences like paying a administrative fee for example.

[4]Note that such input data is individual for each company. There can be added more data to the contract depending on, for example, which business process is involved. Therefore, a very simplified version of the contract is used.

information, i.e. the insurance ID, tax information or the bank account reference of the policy holder. This type of data is not used for calculations.

Technical Contract Data

The technical contract data is relevant for programs which process the data or use it for calculations. The table 3.1.3 shows a listing of relevant data. These elements are also referred to as 'keys'. They have values for example the key contract ID has the value 1234.

Technical contract data
• fixed: – Contract ID (unique identifier) – Product (in this case the burial insurance) – Start date – Insured capital (payout) – Date of birth (to calculate the age) • temporal: – Version (to distinguish between different stored version of the same contract) – State (state model (see 3.1.2)) • calculated: – Contributed capital (capital which has been paid up to a certain time) – Guarantee values – Premium (monthly rate)

Table 3.1: Contract data composition

Note: Fixed means that the value of the key does not change once it is created. In contrast, calculated and temporal data changes over time. The latter refers to the contract model. It is not contained in the real world contract. Machine readability is supported by temporal data and the contract ID. Calculated means that the value depends on some other values, for example date of birth.

3.2 Legislation and Regulation

In general, the applying laws for contracts, information technology and security depend on the country in which a company is located. A distinction is made between guidelines/recommendations which are useful aspects to consider and restrictions which shall be applied to stay in compliance with the law. If not, legal penalties follow by the government or according institutions.

For the research of law requirements an expert interview was conducted. The issue of how secure a system is is not as important as the issue of the kind of data stored within a certain context. To show an example: Microsoft Azure Cloud or Amazon Web Services are operated by multi-billion dollar companies. They have the resources to keep the IT security level extremely high compared to small companies with their in-house IT. For smaller organizations it is relatively expensive to be certified, for example to ISO 27001. To fulfill requirements, the organizational overhead exceeds the value for the company.

Table 3.2 contains noteworthy laws or rather paragraphs (the types of laws are within the parentheses) for this work. They require that access to customer data is protected. The expert also named the "Versicherungsaufsichtliche Anforderungen an die IT (VAIT)" which show special requirements for life insurance companies. These requirements are not a law but should be explicitly considered in this context. They are provided by the Federal Financial Supervisory Authority (BaFin)[5] in Germany. Due to time reasons they are not detailed further in the work.

European Union
• EU-GDPR (Data Protection Act)[EUd] – the EU-GDPR is mainly aimed at the protection of natural persons regarding to the processing of personal data – Article 17 Right to erasure ('right to be forgotten'): "The data subject shall have the right to obtain from the controller the erasure of personal data concerning him or her without undue delay and the controller shall have the obligation to erase personal data without undue delay."[EUd]
Germany
• Strafgesetzbuch StGB (Criminal Code)[Str] – §203 Violation of private secrets: (excerpt of) (1) Anyone who unauthorizedly discloses a foreign secret, namely a secret belonging to the personal sphere of life or an industrial or business secret, which has been entrusted or otherwise become known to him as a member of a private life insurance company, shall be punished with imprisonment for up to one year or with a fine.

Table 3.2: Listing of relevant law paragraphs

3.3 Information Security

The aspects of information security are preservation of confidentiality, integrity and availability of information, authenticity, accountability, non-repudiation, and reliability.[ISO16] The prior motivation of achieving a 'high' or rather appropriate level of information security is to be in compliance with the law because violations of the law may result in fines[6] or imprisonment.

The ISO/IEC 27000 (abbreviated as 27K) is an international standard for information technology security and data privacy. The standard "maintains an expert committee dedicated to the development of international management systems standards for information security, otherwise known as the Information Security Management System (ISMS) family of standards. [...] [It] is intended to assist organizations of all types and sizes to implement and operate an ISMS".[ISO16] The standard also provides a variety of security controls and

[5] `https://www.bafin.de/DE/DieBaFin/diebafin_node.html` (last access: 06.01.2020)

[6] EU-GDPR Article 83(6): "Non-compliance with an order by the supervisory authority as referred to in Article 58(2) shall, in accordance with paragraph 2 of this Article, be subject to administrative fines up to 20 000 000 EUR, or in the case of an undertaking, up to 4% of the total worldwide annual turnover of the preceding financial year, whichever is higher"[EUd]

measures. These should cover every facet of a company that has something to do with information security including human resources, organizational processes, physical environment security, local/wide area network with access restrictions etcetera.

In Germany, the Federal Office for Information Security (BSI) provides a national guideline for information technology security and data privacy. Its objective is similar to ISO/IEC 27K: "The aim of IT-Grundschutz is to achieve an appropriate security level for all types of information of an organisation."[7] The BSI also provides hands-on examples to apply security controls which is helpful besides ISO 27K's rather theoretical descriptions of security controls. They also published a table which maps the security controls from ISO 27001 and 27002 to their 'IT-Grundschutz'.[Inf18]

Note that there exist more than the two mentioned standards. These are not detailed any further, mainly because they affect infrastructure or management aspects more than data storage technologies.

3.4 Data Migration

Due to the long contract durations, data migrations must be carried out approximately every 8-10 years. Data migration means moving a subset of the data or all of it from the 'old' (source) to a 'new' or newer version of the same storage system (target). The data needs to be migrated together with its change history. One of the ways of proceeding a data migration is presented in the white paper by Bloor Research.[How07] The reasons for migrations for example are changes to business processes which could also imply changes to the contract model. These are invoked by regulatory changes, such as the EU-GDPR. Another reason to do a data migration is the breaking of encryption algorithms or the keys used for encryption, if such is applied to data. This could be possible within the next 35 years as a result of increasing computing power or new technologies. More possible changes include modifications to the hardware or operating system, the consolidation of multiple data preserving software systems, the lack of vendor support, major releases[8] etc.[BXW19].

In this work, data migration refers to the exporting of data out of a storage system and importing it to the a newer version of the same system. The migration quality is especially important to the use case company meaning that the process needs to be lossless. The data with its structure must be preserved implying the integrity of data is not corrupted. That includes the change history/transaction log and (if existent) data types, constraints and built in mechanisms like stored procedures. The reason why only migrations from the same system to a newer version of that system (not vice versa) are considered is that cross-system migrations mostly are not lossless. That is due to the fact that internal data structures differ between systems although they use a similar storage technology (i.e. RDBMS). The migration from a newer to an older version of the same system could result in losses to data integrity as versions are not necessarily backwards compatible.

[7] `https://www.bsi.bund.de/EN/Topics/ITGrundschutz/ITGrundschutzHome/itgrundschutzhome_node.html` (last access: 06.01.2020)

[8] "Major Release means a new version of Software that includes changes to the architecture and/or adds new features and functionality in addition to the original functional characteristics of the preceding software release." `https://www.lawinsider.com/dictionary/major-release` (last access: 06.01.2020)

3.5 Resulting Requirements

In this section all relevant requirements are listed. They will be distinguished by where they are taken from. Hence, the following subsections indicate the 'source' of the requirements. The titles of the subsections of the subsections are identifiers to enable the referencing in chapter 5. The sections 3.5.1 and 3.5.2 are requirements for the 'pure' data storage and the ones from section 3.5.3 are requirements related to data migration. The last section (3.5.4) is not considered in the comparison in section 5.1 but it will be annotated in section 5.2.

3.5.1 Contract: Input Data

In this part the requirements coming from the data (see 3.1.3) of contract model are listed.

RQ1-1 Store structured data

The storage technology needs to have the ability to store structured/machine readable data. Structured data has a composition which is enforced on a atomic level. Unstructured data does not have a conceptual definition, for example like a textual document.[Weg04] The definition used for this work differs because the semantic does not necessarily need to be defined by data types. Machine readable means that the data is formatted in a way that it is possible to use it for programs. These formats, like XML (Extensible Markup Language) or JSON (JavaScript Object Notation), are supported by a variety of parsers and libraries. For example data formatted as a PDF (Portable Document Format) usually is not machine readable.

RQ1-2 Queries

There needs to be an option query the stored data. Users need data wherefore a data processing layer between the users and the database exists. Within this layers programs query the data and preprocess it for a certain type of user.

Time variance[9] of data is not included in the comparison.

RQ1-3 Unique identifiers

There needs to be a mechanism to ensure that some entries are unique.

RQ1-4 Modify (contract) keys

Sometimes keys need to be modified or deleted as the contract model evolves over time. Keys refer to the contract elements (see 3.1.3).

RQ1-5 Data types

Values of keys have a data type like 'natural number' or a generic string value.

[9] This is important when saving for example addresses of persons. Following scenario: a customer enters into a insurance contract on day X. A few years later, on day Y, he moves to another town and changes his address. Later, on day Z, he wants to know the status of his insurance for the date before he moved, this means between day X and. If the life insurance company for example saved personal information about the customer 'next to'/combined with information about his contract, the company would query invalid information about the residence of the customer, namely the address he lived at before day Y.

RQ1-6 Constraints

The technology has the option to add constraints to data. Constraints are like rules or restrictions applied to data. An example is: a number may never be below a certain value (i.e number >= threshold). Constraints that are necessary for maintaining integrity conditions[10] are excluded from these.

3.5.2 Contract: State Model

These requirements directly result from the state model (see 3.1).

RQ2-1 Create contracts

There shall be the possibility to create a instance of a contract. The term 'instance' refers to the object instantiation of a class in the world of Object Oriented Programming.

RQ2-2 Delete contracts

There shall be the possibility to delete the instance of a contract. There is a special interest due to regulations (see 3.2). The LV1871 has a internal definition (status: October 2019) of the term 'delete'. Deleting data means the (1) irreversible elimination of data or (2) the permanent removal of the link or linkability of data or (3) irreversible anonymization of data. Anonymization means changing personal data in such a way that this data can no longer be assigned to a person or only with disproportionate effort. This means that any identifying characteristics (e.g. address data, date of birth and other characteristics that enable identification of the person) are removed without replacement.

RQ2-3 Modify (contract) values

This requirement represents a mapping of accounting transactions (see 3.1.2). The technology shall offer the possibility to modify values of keys (see 3.1.3).

3.5.3 Technological Changes

RQ3-1 Data export

The storage technology natively has the option to export all of its data including the format or rather scheme of the data. Natively means without the usage of third party programs.

RQ3-2 Data import

The storage technology natively has the option to import data which has a supported format.

RQ3-3 Lossless data structure migration

The exporting (see RQ3-1) and importing (see RQ3-2) of data needs to be lossless. This means that the data and its structure must be preserved implying the integrity of data is not corrupted. That includes (if existent) data types 3.5.1, constraints 3.5.1 and built in mechanisms 3.5.1.

[10]This refers to referential integrity of data in RDBMS for example (see 4.3.4).

RQ3-4 Lossless migration of transaction history

There needs to be an option to migrate the change history/transaction log without losing the integrity of data.

3.5.4 Usability

This aspect is an individual consideration of the use case company. The term usability refers to integrability and maintainability. As the users mainly are developers and the IT of the use case company, it is beneficial for them to have a technology which is well known, standardized and has an active community online[11]. The more members the technology use the more help is available to solve occurring problems. What helps users a lot to understand and work with a technology and is a well defined and documented API (Application Programming Interface), especially if employees leave the company and other people with less experience have to deal with the technology. Integrability includes the number of prerequisites, in terms of knowledge and software development environments, which are necessary to run the storage technology. This could also determine the degree of interoperability as software environments are not supported by every operating system.

[11] For example how many users use this technology at Stack Overflow: `https://stackoverflow.com/`

4 Technologies

In this chapter the technologies for storing data are presented which are blockchain technology and relational databases. An output of this chapter will be information about their specifications and characteristics to present the basic functionality. The main focus is on the 'pure' data storage because business logic is outsourced to the data processing programs. This includes creating, reading, updating and deleting data (CRUD). Data migration capabilities will be mentioned briefly as they belong more to the database management system than the database itself. Advanced features like stored procedures in RDBMS are not considered in the technology comparison (see 5.1) because they differ between RDBMS. In section 4.1 the technology 'blockchain' with its components will be presented and defined. It will be differentiated between three blockchain systems based on the used data model in section 4.1.2. The data models or rather the approach of storing data will be compared to each other in 4.2. This determines which system is best suited to store contracts or rather contract data. The definition of relational databases and the underlaying relational model is presented in section 4.3. The data storage functionality for relational databases is detailed in section 4.3.4 which is about SQL.

4.1 Distributed Ledger Technology

The concept of DLT has already existed before blockchain technology. The authors in *Distributed Ledger Technology Systems: A Conceptual Framework*[RGG+18] show different definitions of the term DLT selected from a range of papers or institutions which deal with the subject blockchain or DLT. They propose a formal definition including only the essential requirements[1]

As a type of distributed system, a "DLT system is a system of electronic records that

i. enables a network of independent participants to establish a consensus around
ii. the authoritative[2] ordering of cryptographically-validated ('signed') transactions[3]. These records are made
iii. persistent by replicating the data across multiple nodes[4], and
iv. tamper-evident[5] by linking them by cryptographic hashes.
v. The shared result of the reconciliation/consensus process - the 'ledger' - serves as the authoritative version for these records"[RGG+18].

[1] The definitions in the footnotes of the upcoming definitions are copied from the just mentioned paper.
[2] Is used to designate the set of records that all network participants agree upon, and which are not subject to subsequent alteration without consensus.
[3] A change to the records in the system.
[4] A network participant communicating with peers over a shared communication channel.
[5] Refers to the the ability of participants to easily detect non-consensual changes to records.

4.1.1 Blockchain

Blockchain technology is a subset of DLT technology that uses a data structure called blockchain[RGG+18]. In this work only DLT systems that use blockchain technology are considered. The terms DLT system and blockchain system are interchangeable.

The 'blockchain' itself is a ledger. It contains signed transactions which are grouped to blocks. These blocks are cryptographically linked. Every block comprises a block header and block data (mostly transactions). The block header always has the following entries: a timestamp, the hash generated over a subset of its block data[6] (mostly the transactions) and the hash of the previous block header in the blockchain. Depending on the protocol and consensus method, the header can also contain additional entries. An example: the 'nonce' entry which is needed if the consensus method Proof-of-Work (PoW) is used.[YMRS18]

The ledger is distributed to nodes in a network of nodes. Every node holds its own individual instance of the ledger. Because it is a network including multiple nodes which all are holding their own version of the ledger, it may be that there is no 'final' or correct ledger at a given time. This imperfect state exists because, for example, clients like wallets[7] want to send or rather broadcast transactions (i.e. monetary transaction) to the blockchain network. These transactions imply a modification of the ledger. The protocol rules indicate how valid transactions and blocks shall look like. The set of actions to add transactions to the ledger is called the consensus method. For example the consensus method PoW is used in the Bitcoin blockchain[8]. There are specific nodes (in case of PoW called 'miners') in the network whose task is to collect proposed and yet unconfirmed transactions. The task of these nodes is to check the validity of transactions. Transactions that are in compliance with the protocol are valid and can be confirmed. The transactions are then signed and bundled to blocks which are then added to the blockchain. If multiple transaction collecting nodes do their task simultaneously they may have conflicting individual instances, called journals. For this the protocol needs a rule for conflict resolution. For instance, the Bitcoin blockchain "resolves a temporary split caused by two competing valid blocks at equal height by choosing the block on the branch that carries most cumulative PoW (longest-chain rule)"[RGG+18]. The goal is to keep the journals in sync. This leads to "a convergence towards a single accepted set of authoritative records"[RGG+18] - the ledger.

It is tamper-evident meaning that everyone can check if the integrity of the blockchain has been violated by checking the integrity of the hashes. Often the ledger is referred to as immutable or tamper-proof due to the fact that it is distributed to multiple nodes. This property can only be maintained as long as there is no entity that has a superior computing power to produce records faster than the rest of the network. The '51%' attack is an example for this. The records of 'honest' nodes could be replaced by records of the entity having the majority of computing power due to conflict resolution rules.[RGG+18] It is more accurate to say that the blockchain is resistant and not immune to unwanted tampering of data.

[6] `https://hackernoon.com/merkle-trees-181cb4bc30b4` (last access: 06.01.2020)

[7] A wallet is a device, program or service. It stores public and or private keys to track the ownership of cryptocurrencies.

[8] `https://bitcoin.org/en/how-it-works` (last access: 06.01.2020)

4.1.2 Differentiation of Blockchain Systems

There is a variation of blockchain systems with different design decisions. This leads to the choice of certain protocols and configurations. The authors of *Distributed Ledger Technology Systems: A Conceptual Framework*[RGG+18] propose a framework to compare these systems on hierarchical system levels. The top level categories are protocol, network and data layer.

The protocol layer refers to the governance of the system. This term for example covers how collective decisions are made and who has permission to submit votes. The network layer refers to the blockchain systems architecture. It includes how the network access is regulated, which nodes are allowed to process transactions etc. The data layer refers to the type of data stored in the blockchain. The authors distinguish between four types. Bitcoin is a *endogenous* reference. This type only exists within the boundaries of the system. In contrast *exogenous* data exclusively exists outside the boundaries. An example are assets like real estate wherefore a connection to the real world is required. The type of data referred in the context of this work is called *hybrid*. This type of data exclusively exists within the boundary of the system (for example a key-value representation of a contract) but may has references to real world entities (the burial insurance contracts). The fourth type of data is called *self-referential*. The authors define it as pieces of code that do not have references to endogenous nor exogenous variables. They may have internal or external variables.

Smart Contracts

Smart contracts belong to the category self-referential. These can be regarded as stored procedures which are invoked by certain events like an incoming transaction or an occurred condition. "The inputs, outputs and states affected by the smart contract execution are agreed on by every node."[DLZ+18] There are built-in and user defined smart contracts. The first ones implement parts of the blockchain protocol and define the processes like transaction validation. User defined smart contracts can specify arbitrary computations. Ethereum smart contracts are written in Solidity which is Turing complete code executed on the virtual machine of Ethereum.[DLZ+18] Only this type of smart contracts is referred to in this work.

4.2 Evaluation of Blockchain Systems

Because the comparison (see chapter 5) only considers the storage of data, the data model is the most important component. The authors in *Untangling blockchain: A data processing view of blockchain systems*[DLZ+18] compare blockchain systems, amongst other criteria, by the used data model. It is differentiated between three different kinds of data models: transaction-based (Bitcoin), account-based (Ethereum) and key-value (Hyperledger Fabric[9] (HLF)). The terms inside the parentheses are DLT systems that implement the corresponding data model. They are taken as explanatory examples for this work. All three of them are open-source projects which use blockchain technology.

[9] `https://hyperledger-fabric.readthedocs.io/en/latest/` (last access: 06.01.2020)

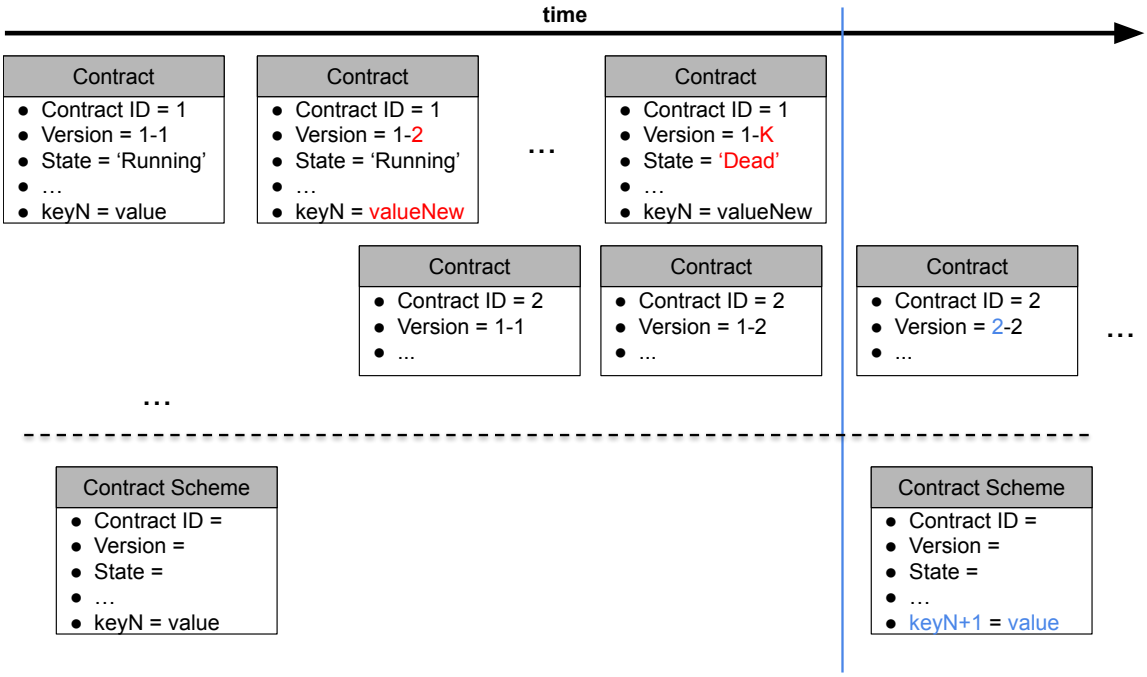

Figure 4.1: Contract data represented as key-value pairs

4.2.1 Desired Concept

Because blockchain systems natively are not databases, a concept of how to store contracts needs to be developed. Therefore the elements of the contract are transformed to a key-value representation of the contract. It is illustrated in figure 4.1.

In order to establish the basis for the comparison the blockchain system data scheme needs the following functionality. Contracts can be created and have a contract scheme. The vertical blue line in figure 4.1 represents a contract scheme modification. This means that keys of the contract are modified. Every contract after the scheme modification has the updated scheme. Values of keys need to be modifiable. This is represented by the red colored text. The version key is used to distinguish between different versions of the same contract. It works as follows: the first digit refers to the contract scheme, the second refers to the contract version. The version changes every time an element of the contract is modified. The system must also be able to delete contracts. All these requirements come from chapter 3 (see 3.5.1 and 3.5.2). The deletion is a special challenge when using blockchain systems because the transaction log is immutable.

In the following three subsections the three data models will be presented. It will be decided whether it can be used for this purpose or not. If not the reason for that decision will be presented.

4.2.2 Transaction-based Blockchain Systems

This model often is used to save transactions of tokens between addresses. A token could be a cryptocurrency and addresses could be a public key of an entity like a person. Therefore to get the current state of, for example, an 'account' balance, the whole blockchain has to

be traversed to gather only the transactions which have references to the address of that 'account'. Although the concept in 4.2.1 could be implemented, the transaction-based data model is not recommended for the use case of storing contracts.

One of the reasons is that the protocol needs to be adapted to a key-value store or similar[10]. When using the reference client Bitcoin Core, every change request needs to be proposed to a Github repository[11] as a so called Bitcoin Improvement Proposal. These work similar to Request For Comments[12]. The enforcement of a change is virtually impossible in public blockchains like the one from Bitcoin as the proposals require community consensus. An option to bypass this is to use a different software client. Bitcoin has no formal protocol specification as clients implement the rules. Bitcoin Core is the 'official' client but there are more competing clients users can choose from. Any of these clients could implement rule changes which then would be enforced by users running that particular client.[RGG+18]

Another contra argument is that making changes is expensive or only partially possible with limitations. By propagating blockchains as an immutable ledger it never was the intention to modify the ledger. In the following argument a change does not mean an actual change to the data but make it seem like a change was made. The effect of this is like using views in RDBMS. An option to make changes is to create a new block and using version tags. Figure 4.2[13] helps to understand the following. You cannot make changes to a block when it is verified except rehashing every block after the modified block, which is expensive in terms of computation power (for example if PoW is used). By using version tags, the newest version of a contract can always be determined.[YMRS18] The problem here is that multiple versions of contracts coexist. Therefore only the newest version of the contract is valid. The newest one is the one which is located closest to the last appended block of the blockchain. It needs to be ensured that only the last appended version to the blockchain can be accessed to users.

Thereby another problem comes up. Natively blockchain systems do not provide a data querying mechanism, or not as advanced as RDBMS do.[CMC19] To solve the problem that only the newest version of a contract shall be accessible is to use asymmetric encryption like RSA.[RSA78] The values of the keys inside a block are encrypted with a public key and can be decrypted using the corresponding private key. So if a newer version of a contract exists, a new block is created including the changed values. It will be encrypted using a new created key pair. The key of the older version will be deleted or thrown away so that no one can access it. However this needs a key management system to maintain which is additional expensive overhead using it for this purpose.

4.2.3 Account-based Blockchain Systems

The account-based[14] data model uses addresses as accounts. This model makes it possible to save data without traversing the whole transaction log first.

[10] Another way to store data on the Bitcoin blockchain is to encode it as Bitcoin addresses. The authors of *Blockchain: business' next new "It" technology* stated that someone stored an image of Nelson Mandela using this method.[MA19] However, there is no source indicated for a proof in the paper.

[11] `https://github.com/bitcoin/bips` (last access: 2.1.20)

[12] `https://www.rfc-editor.org/` (last access: 06.01.2020)

[13] This figure is taken from `https://medium.com/cybermiles/diving-into-ethereums-world-state-c893102030ed` (last access: 06.01.2020)

[14] All of the information in the following is taken from *Ethereum: A secure decentralised generalised transaction ledger*[W+14] unless another source is referenced.

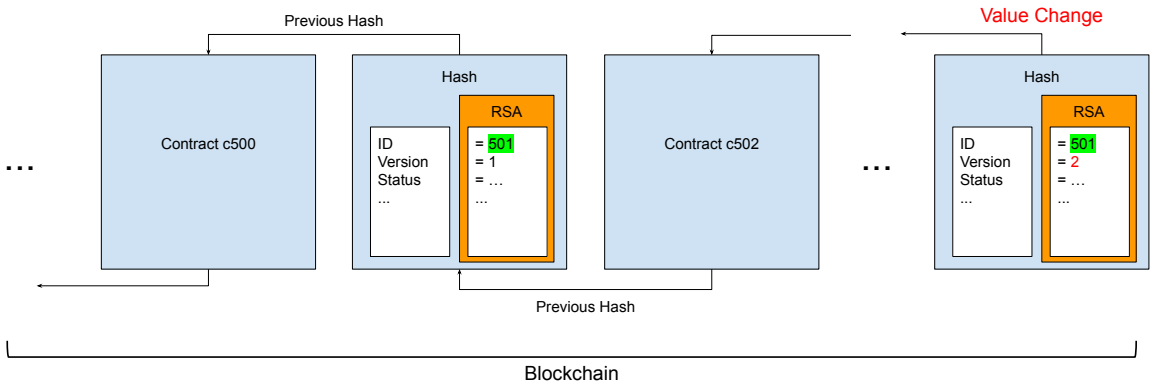

Figure 4.2: Concept of saving data using the transaction based data model

Relevant technical details and part of the architecture of Ethereum is presented first. Their explanations and relevance for this work will be shown in forthcoming paragraphs.

Figure 4.3[15] illustrates a subset of entries inside a Ethereum block header and corresponding components. Every block header has the entries State Root and Transactions Root. They are 256-bit Keccak[16] hashes of the corresponding root node of a modified Merkle Patricia tree[17]. The arrows in the figure represent the references to the state and transaction trie. "The root node of this structure is cryptographically dependent on all internal data". [W+14] The Merkle Patricia Tree stores data. The data is stored as key-value bindings. The operations insert, lookup and delete can be executed in $O(log\ n)$[18].

Every block has its own transaction trie. This is to keep track of executed changes to data associated within this block. In contrast, there is only one global state trie, the world state. It is constantly updated. "The state trie contains a key and value pair for every account which exists on the Ethereum network"[19]. The key is a 160-bit identifier, an Ethereum account address. The value is the corresponding account state, encoded by the Recursive-Length Prefix (RLP)[20] encoding method.

It is differentiated between two types of accounts: external and contract accounts. External accounts are controlled by private keys. These can send transactions to contract accounts (containing so called 'smart contracts'[21]). The contract accounts have code[22] and an internal storage for persistent smart contract data. By receiving messages, i.e. from an external account, the code of the contract account will be executed with the submitted

[15] Based on `https://medium.com/cybermiles/diving-into-ethereums-world-state-c893102030ed` (last access: 06.01.2020).
[16] `https://keccak.team/keccak.html` (last access: 06.01.2020)
[17] It is interchangeable with the term 'trie'.
[18] `https://github.com/ethereum/wiki/wiki/Patricia-Tree` (last access: 06.01.2020)
[19] `https://medium.com/cybermiles/diving-into-ethereums-world-state-c893102030ed` (last access: 06.01.2020)
[20] `https://github.com/ethereum/wiki/wiki/RLP` (last access: 06.01.2020)
[21] `https://pegasys.tech/ethereum-explained-merkle-trees-world-state-transactions-and-more/` (last access: 06.01.2020)
[22] `https://github.com/ethereum/wiki/wiki/White-Paper` (last access: 06.01.2020)

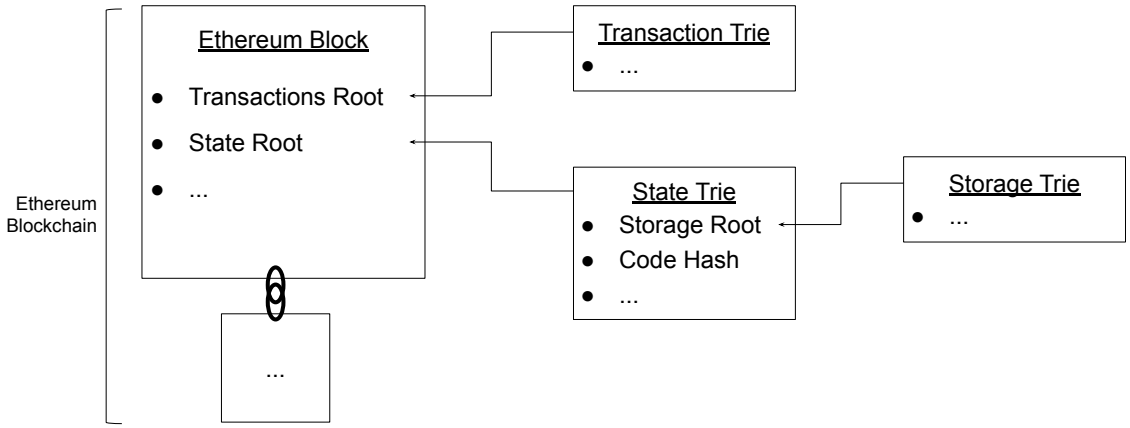

Figure 4.3: Ethereum block architecture

transaction parameters. Contract accounts then can also send messages to contracts. Every performed action (i.e. sending of messages to other contract accounts) needs to be paid for by the initiator of the transaction.

Every account in the state trie has its own storage trie. The storage trie is used for the smart contract data[18]. Every account (state), which is in the state trie, has the entry Storage Root which is a 256-bit Keccak hash of the root node of the storage trie. This is illustrated in figure 4.3. Amongst other entries, every account state has the so called *Code Hash* entry. This is a hash of the executable code of the account. This implies that the code is immutable[23].

In the following it is explained how to actually store data via smart contract. Smart contracts have variables and functions, i.e. setter functions to change values of variables. A mapping of real world contracts to smart contracts would be possible, like presented in 4.2.1 as keys (i.e. contract ID) represent variables of the smart contract.

However, there is a problem: smart contracts are immutable: it is not possible to change functions or variables (names of the variables/the keys), also it is not possible to add or remove these. But the values of variables can be changed, as mentioned before. And that means to change code, also for example to fix bugs or for refractoring, a new contract account has to be created[21] while the old one coexists.

The paper *Data Management: Relational vs Blockchain Databases*[CMC19] shows how to save data to the Ethereum blockchain and proves that it is possible. The authors state a problem that "blockchains don't have any sophisticated data query mechanism to interrogate the data that is stored in them".[CMC19] The Ethereum blockchain does not provide queries like SQL does. Therefore the filtering of data needs to be done in data returning functions of smart contracts. If there is a need for new queries either new smart contracts have to

[23]The entries Storage Root and Code Hash of external accounts are empty because they do not have any code

be created, because they are immutable. Another option is to transfer the filtering to the data processing programs. In this case it may be unknown how much data is going to be retrieved. This is because every event in which the address of the smart contract occurs will be returned (the authors used the Web3 framework). Another problem related to querying data would be: how to manage all the parallel existing smart contracts on the (Ethereum) blockchain? It would need a key management system to keep track of the accounts that are currently in the blockchain. There also shall be a way to mark smart contracts as deleted, because once appended to the blockchain, it is nearly impossible to revert this action. The authors mention more considerations when using Ethereum for example the lacking data formats of Solidity[24] and the costs (transaction fees).

4.2.4 Key-value-based Blockchain Systems

The last one of the three introduced data models is the key-value-based data model. As an explanatory example, the enterprise blockchain solution Hyperledger Fabric (HLF) was chosen. The following information is provided by the documentation of HLF[9].

Architecture

HLF is to be considered separately and may not entirely match the given definition in section 4.1.1. The reason is that the system is modular. Parts of the architecture are interchangeable for specific needs like algorithms for identities, encryption, consensus etc. There are different types of nodes in the network. They fulfill certain roles for example transaction processing with an endorsement policy. This policy defines the type of voting procedure for instance a two-thirds majority could be implemented. A privacy feature of HLF is to allow groups of participants to create separate ledgers of transactions in one network (channels). This is achieved by using access restrictions with a Public Key Infrastructure.

Data Layer

The ledger is a combination of the world state and an immutable blockchain or rather transaction log history. The world state is an actual database (levelDB[25] or couchDB[26]) For each channel of which a participant is member of, a copy of the ledger is maintained.

To store data in the ledger of a channel, user defined smart contracts (called chaincode) are used. Chaincode is encoded logic[27] invoked by specific transactions on the channel. These transactions can be sent by an external application. The result of chaincode invocations are key-value writes (changes to objects). These changes are broadcast to the network, processed and applied to the ledger. Therefore objects can be defined as key-value representations or modified. Values have an implicit data type like string or number. They always have an unique identifier (ID). The structure of objects (data scheme) can also be modified, which is the equivalent to modifying keys in the concept (see 4.2.1).

The ledger can contain different versions of the same object. Therefore modifications to objects can be tracked because these are saved as transactions to the blockchain while the

[24] The programming language for smart contracts in Ethereum
[25] `https://github.com/google/leveldb` (last access: 06.01.2020)
[26] `http://docs.couchdb.org/en/stable/intro/index.html` (last access: 06.01.2020)
[27] Chaincode can be written in Go, node.js, or Java.

world state contains the latest version of each object. So there is no need to traverse the whole blockchain to get the current state of the ledger.

In order to execute queries, transactions are sent by an application to the network which invokes chaincode. It interacts with the database component of the ledger to receive current data to send it back to the application. The ledger history can also be queried if older versions of data objects are wanted. Most queries are key based, for example by addressing the ID of objects. By using queries inside chaincode for example, restrictions to values of keys can be made (like constraints in SQL).

4.2.5 Summary

The following refers to the explanatory examples: Bitcoin, Ethereum and Hyperledger Fabric. Therefore the made statements are not general for all DLT system using the corresponding data model:

As discussed, DLT systems using a transaction-based data model do not suit for the use case of storing contracts. To generalize it, these systems should not be used as a database like relational databases because you cannot make changes to data inside the blockchain. To query data the whole blockchain has to be traversed which is too much overhead. Storing data comes along with limitations or is not even possible.

DLT systems using the account-based data model are 'more viable' for the use case of storing data than the transaction-based model but still have limitations or restrictions. Actual (states of) data can be persistently saved using smart contracts. Data can be modified which is a benefit over transaction-based DLT systems. But it is not possible to change a smart contract, more precise the structure of it because it is immutable. Another disadvantage is querying of data. There is no sophisticated query mechanism like RDBMS have. The filtering of data is not flexible, in contrast to SQL. For using account-based DLT systems a key management system is necessary to manage account addresses. An open question is: how scalable is this approach with accounts as addresses.

Public blockchains should not be used as historical records, especially not for unencrypted records. The problem mentioned in *NISTIR 8202 Blockchain Technology Overview*[YMRS18] is called the 'Blockchain Death'. As well as traditional centralized systems, blockchain networks will be taken down one day. Due to its decentralized nature it could be that it never shuts down entirely, meaning that there could always be remaining blockchain nodes. The less publishing nodes are left the higher gets the chance for a malicious user to overpower the few publishing nodes to modify the blockchain.

The technology of blockchain was designed as a transaction log performing in an adversial environment enabling entities that do not necessarily trust each other to reach consensus over a shared set of data without relying on a central authority.[RGG+18] However, for this use case, its advantages do not outweigh its shortcomings related to the failure to comply with the desired concept (see 4.2.1) and lacking query mechanisms.

Therefore for this use case the key-value approach is taken, more precise HLF. Every feature of the desired concept can be achieved. Additionally the network access can be restricted

to keep control over the users. Thus, HLF is not affected by the problem of potentially dying public blockchains. This technology will be used for the comparison with a relational database in chapter 5.

4.3 Relational Databases

In this section a general definition of relational database management systems (RDMBS) is given. Most of the information is taken from the book *Datenbanksysteme: Eine Einführung.*[KE15]

4.3.1 Definition

Generally, a database system consists of the database management system and the database itself. The DBMS consists of a set of data and the programs that are necessary for processing data. DBMS base on a data model which provides the infrastructure to map parts of the real world to the model. In the case of RDBMS the model is based on the relational model according to Edgar F. Codd.[Cod70] The model describes data objects, called data scheme. For this purpose the DBMS provides a Data Definition Language (DDL) and a Data Manipulation Language (DML) to define operations on the data objects. The DML consists of a Query Language and the actual data manipulation language to insert, modify or delete data. The scheme doesn't change very frequently over time unlike the saved data. There are several RDBMS on the market. Some are open-source like MySQL[28], some are proprietary software like IBM DB2[29]. They provide almost the same basic functionality because they all use SQL (see 4.3.4). The feature differences are for example internal procedures like triggers, or security features like different access control models. Another aspect in which they differ is performance due to internal data structuring using indexing.

4.3.2 Abstraction of Data

A distinction is made between three levels of abstractions of a database system: the physical, logical and views level. The physical level includes data structures and eventually index structures. This level is mandatory for the performance of the database system. The logical level is responsible for the data scheme. The function of the views level is to provide different, tailored views for different users or applications. By this, only necessary data is shown and also data privacy can be ensured. The benefit of having such a distinction is to have data independence. Theoretically, if a modification on a certain level is made, the other levels or rather the users should not be affected by the change. To achieve this, the interfaces between the layers need to be well defined. Usually only physical data independence can be ensured by today's database systems because only little conceptual changes are feasible to the database scheme. A change to the logical layer also impacts the views level because the views rely on the data scheme.

The focus lays on the logical layer, especially on the data model.

[28] `https://www.mysql.com/de/` (last access: 3.1.20)
[29] `https://www.ibm.com/products/db2-database` (last access: 3.1.20)

4.3.3 Logical Data Model

More specific DBMS for example are relational DBMS (RDBMS). They are always referred to in this work when speaking of DBMS. The data model is based on relations.

In general, there are formal models to operate on relations and to describe queries. Mostly relational algebra and calculus are used. They serve as a theoretical basis for query languages like SQL. The relevant parts of the mathematical algebra are operands (relations or rather tables) and operations (on the tables).

The purpose of all this is to map real-world entities/objects and their relations to each other to the data model of databases. In the case of RDBMS the method of achieving this mostly is to use the Entity-Relation Model (ERM). In this case the objects are contracts. The mapping of the data objects to the relations looks as follows: the rows of a table represent objects and therefore the current database state. Tables represent the data scheme of the database and serve as a container for objects. Columns of a specific table describe properties of a data object[30].

4.3.4 SQL

As mentioned the RDBMS has a DDL, DML and query language. In this work SQL (Structured Query Language) is used which is a combination of a these three languages. Most RDBMS provide a set of SQL commands which are executed for example in a shell, some provide web interfaces to manage the database. SQL can be embedded to a host language like Java. That increases usability and maintenance because the data processing including business logic can be separated from the database. In the following concepts of SQL are specified. They serve as a proof that the requirements are fulfilled in the technology comparison in section 5.1.

Definition of data schemes

SQL provides commands to create tables (data schemes) and to alter and delete them. When adding columns to tables, the system fills the yet missing columns of objects (rows) with null values. The specification of tables or rather their data scheme includes data types of columns and constraints for them which describe details of objects. Supported are three fundamental data types: numbers, strings and a type for data. The main purpose of RDBMS is to store structured data. For other applications some SQL implementations provide additional types for binary or XML files. To distinguish between objects in a table, every object has at least one unique attribute, co called 'key'. By this objects can be referred to across tables ('foreign key relationship'). The deletion of tables could lead to dangling references as some refer to not existing tables then ('referential integrity').

Elementary Data Manipulation

Elementary data manipulation refers to the adding, modifying and deletion of rows or rather objects in tables. When having foreign key relationships the deletion of objects could lead to dangling references as some refer to the void then.

[30] It should be noted that the meta information, or data scheme, which describes the structure of the data is not the same as the database state containing the current valid saved data itself.

Queries

SQL provides a sophisticated query mechanism to traverse data from the database. This enables it to return whole tables or only one specific row of a table. The query mechanism includes for example aggregation functions like sums or averages and conditional queries. To increase usability for users and information security SQL provides views. They are virtual tailored tables which only consist of certain parts of tables. These views can be used as a pre-query/macros to make complex database queries more comprehensible.

4.3.5 Data Migration Tools

Usually the option of exporting data to import it to a newer version of the same product is available. For example the RDBMS Oracle additionally provides an option to migrate data from a third-party RDBMS[31]. The number of data migration tools and the quality of the migration (see 3.5.3) depends on the used RDBMS.

[31] `https://docs.oracle.com/en/database/oracle/sql-developer/19.2/rptug/migrating-third-party-databases.html` (last access: 3.1.20)

5 Evaluation

To determine the best candidate for the use case, the technologies RDBMS and HLF are compared with each other. The basis for this are the requirements from chapter 3. In section 5.1 it is iterated through the list of requirements (see 3.5) to check whether each of the technologies is capable of meeting them. For this purpose, the requirements have weightings which reflect their importance. The result of the comparison is a numeric score which will be evaluated in section 5.2.

5.1 Technology Comparison

In table 5.2 a decision matrix is presented. In that table features about creating, reading, updating and deleting data (CRUD) are listed. Table 5.3 also presents a decision matrix with features about data migration. The outcome of these tables is a numeric score. The higher the score, the better the technology fits the use case. Hence weightings ('importance') are assigned to requirements/features for this use case. What they mean is explained in table 5.1.

Weighting	Explanation
10	A must-have feature which is especially crucial to the use case company.
7	This is basic functionality of what the technology shall have and/or mandatory for the use case company.
6	The functionality may be outsourced to, for instance, a migration project because it is not needed frequently. Due to the fact that migration projects should be minimized, the feature still is mandatory.
1	These are should-haves or nice-to-haves because functionality could be outsourced to data processing programs.

Table 5.1: Explanation of weightings for the technology comparison

The score is the sum of the products of the weightings and multipliers (0, 0.5 or 1). If a requirement is fulfilled, the multiplier is 1. This is indicated by the checkmarks in tables 5.2 and 5.3. How the technologies fulfill these requirements will not be explained again because it already was explained in section 4.2.4 and 4.3. If the requirement is not fulfilled, the multiplier is 0. The following are motives which sets the multiplier to 0.5: the fulfillment is only a proof-of-concept[1] or possible under certain conditions. Bringing the requirements in a decision matrix gives an overview of how well each of the technology fits the use case.

[1]This refers to a (adapted) definition given here `https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator` (last access: 3.1.20)

The requirements are separated into two tables. Table 5.2 contains CRUD requirements and table 5.3 contains the ones depending on the database management system. The output of each table is a score. The overall score is the sum of each table score.

Table 5.2 will be evaluated in section 5.1.1 and table 5.3 in 5.1.2.

Reference		Requirement	Weighting	RDBMS	HLF (DLT)
3.5.1	RQ1-1	Store structured data	7	✓	✓
	RQ1-2	Queries	7	✓	✓
	RQ1-3	Unique identifiers	7	✓	✓
	RQ1-4	Modify (contract) keys	6	Conditional	✓
	RQ1-5	Data types	1	✓	✓
	RQ1-6	Constraints	1	✓	✓
3.5.2	RQ2-1	Create contracts	7	✓	✓
	RQ2-2	Delete contracts	10	Conditional	No
	RQ2-3	Modify (contract) values	7	✓	✓
			Score:	**45/53**	**43/53**

Table 5.2: Technology comparison based on CRUD features

Reference		Requirement	Weighting	RDBMS	HLF (DLT)
3.5.3	RQ3-1	Data export	7	✓	Proof-of-concept
	RQ3-2	Data import	7	✓	Proof-of-concept
	RQ3-3	Lossless data structure migration	10	Conditional	Proof-of-concept
	RQ3-4	Lossless migration of transaction history	10	Conditional	Proof-of-concept
			Score:	**24/34**	**17/34**

Table 5.3: Technology comparison based on migration capabilities

5.1.1 Data Storage Capabilities

The resulting score in table 5.2 for RDBMS is 45/53 and for HLF 43/53. Both scores will be evaluated in the following.

The upcoming explains, why the RDBMS fulfills the requirements of modifying contract keys (RQ1-4) and deletion of contracts (RQ2-2) only under certain conditions. The RDBMS has a specific disadvantage in contrast to HLF in terms of deleting entries in tables or entire tables. The aspect also includes modifying names of entries and tables. This is the result of foreign key relationships. If an entry has a reference to another entry of another table and would be deleted, dangling references occur (see 4.3.4). To address this problem, SQL provides multiple constraints (i.e. 'ON DELETE' and 'ON UPDATE') which will reject the deletion or modification of keys which have foreign key relationships and return a error message. Another way to react to potential violations of referential integrity is to add con-

straints ('SET DEFAULT' and 'SET NULL') which set the key to its default value or a null value.[KE15]

HLF does not fulfill the delete contracts requirement (RQ2-2). It depends on the definition of deleting data given in section 3.5.2. Transactions in the blockchain cannot be deleted after applying them. As HLF has a world state database which consists of the current state of the transaction log, the illusion of deleting (and also modifying) data can be created (see 4.2.4). Objects have multiple versions. Therefore deleted objects do not appear in the database but still exist in the transaction log. The solution could be to encrypt affected values with encryption keys or anonymize the data. However, the encryption requires key management for encryption keys and anonymization is difficult to achieve as contracts need to be assignable to a natural person for administrative reasons. The requirement of deleting contracts could be accomplished, under the condition of allowing the existence of data in the transaction log and restricting access to it. Nonetheless, this does not meet the definition of permanently removing the link or linkability of data.

To sum it up, RDBMS is better than HLF in the CRUD based comparison. Its shortcoming of modifying and deleting can be fixed by using SQL constraints. HLF is not able to delete data according to the definition given in 3.5.2 wherefore it has a lower score.

5.1.2 Data Migration Capabilities

In this subsection table 5.3 is evaluated which refers to the definition given in 3.4. The score for RDBMS is 24/34 and for DLT 17/34.

RDBMS usually have the option of exporting data to import it to a newer version of the same product. For this purpose natively a migration assistant guides users through this process. The quality of data migration is different from system to system. It might also differ from version to version of the same system. It is not possible make a general statement saying that every RDBMS has the option to execute a lossless (see 3.4) data migration. Nevertheless there are systems which meet the given definition of data migration.[2]

HLF was published in 2015. As it is relatively new, its maturity is not as advanced as from RDBMS. Therefore HLF does not have native options to perform a data migration. The paper *Patterns for Blockchain Migration*[BXW19] deals with data migration patterns for blockchain. The authors present multiple methods to export data from a source blockchain system and import it to a target system. One of the export methods is Snapshotting. This gets a snapshot of the global state on the source system including smart contracts. A method to export and import data is Transaction Replay which replays a selected number of transactions or all of them on the target system to recreate the state and the transaction history. For this use case, a combination of both methods is needed. The whole state including the smart contracts and the transaction history must be preserved. Because the paper only presents a proof-of-concept and not a implementation, it cannot be guaranteed that it will be working for HLF, especially without third party tools. However this could change in the future.

[2]The use case company stated DB2 (at least the currently used version) is able to perform data migrations according to the here given definition.

The better candidate in this category is RDBMS because it has native options to perform a data migration.

5.2 Result

In section 5.1.1 and 5.1.2 it was detailed why HLF is not as capable for this use case as RDBMS. One of the main reasons is that HLF cannot delete data according to the given definition in 3.5.2 as the transaction log is immutable. In contrast, the RDBMS can delete data, but care must be taken that the referential integrity is not violated. The other reason is that HLF is not as mature as RDBMS. One of the consequences is that there is no native option to migrate data. That is a crucial feature for the use case company as contracts endure up to 120 years in the worst case. However, if HLF gets the native option in the future to perform data migrations it then depends on the involved effort which technology is better. It is connected with the question of how often a migration must be carried out. The following trade-off must be assessed: For example one system has to be migrated every 10 years with a relatively small effort and in contrast the other system has to be migrated only every 30 years but with a relatively large effort. The monetary and human resources required (and available), which obviously differs between companies as well as use cases, would play a decisive role. At the moment HLF is inferior to the RDBMS in terms of native data migration options.

In section 3.5.4 the aspect of usability was presented. It was not taken into comparison between RDBMS and HLF. The reason for this is that usability has to be defined precisely to measure and evaluate it. It may also be subjective in some sub-aspects. That would have gone beyond the scope of this work. Nonetheless one comparable, measurable point should be annotated. It is, as mentioned already, that HLF or blockchain technology is not as mature as RDBMS. This means that blockchain technology is not standardized in opposition to SQL.[KE15] Experimenting is still needed to learn the technology better.[ØUJ17]

The final and overall score for RDBMS is 69/87 and for HLF 60/87. This makes RDBMS the best candidate for this use case.

6 Conclusion and Outlook

The use case life insurance company wanted to know if relational database management systems (RDBMS), which they are currently using, could be replaced by blockchain technology for the use case of a burial insurance. This work has shown that it is not replaceable by this relatively new technology. At least not at the moment of writing this thesis.

The method to prove that it is not as viable as RDBMS is to gather necessary features to run the services of the company (in scope of the use case). As the use case is a burial insurance a contract model including a state was developed. In this context, especially laws had to be considered. The contract models and regulations implied requirements and therefore necessary features. The blockchain technology needs to fulfill these requirements so that all necessary services, which are supported by the currently used technology, can be delivered on an adequate level.

As there exist multiple blockchain systems for a variety of use cases, it had to be determined which blockchain system would come into consideration for storing data of contracts with a lifetime up to 120 years.

Hence, blockchain systems were evaluated by their data model. There are transaction-based, account-based and key-value-based blockchain systems. For that purpose explanatory implementations for each of the data models were chosen to make it more comprehensible. The key-value-based data model, which is implemented in Hyperledger Fabric (HLF), is most suitable for this use case. HLF does not have a public blockchain, as it turned out to be problematic when dealing with personal data, especially due to data privacy laws. The main reason for choosing HLF is that it is able to track states of contracts within a world state database as it is main component of its architecture. An advantage over the other two data models, or rather their implementations, is the advanced and native query mechanism of HLF. This one is a mandatory feature for the use case company as the business processes rely on database queries.

The chosen blockchain system then was compared to RDBMS based on the requirements gathered beforehand and evaluated. The reasons of why blockchain systems are not the best candidate for saving long enduring contracts were presented. For HLF it was shown that data cannot be deleted. Although this works within the scope of its database, the data still exists in its transaction log (blockchain). Another reason the blockchain system is inferior to RDBMS is that it does not provide native options to migrate data. This is a crucial feature because the data might be saved for up to 120 years. There are strong doubts that a technology, or that version of it, lasts this long wherefore data needs to be migrated to newer systems and technologies. In this work it was also discovered that the ability of blockchain to perform in an adversial environment is not a benefit over RDBMS as in this case participants are trusted.

The search for native options of data migration when using HLF showed that not much research has been done in this area. There are concepts and patterns for migration in the scope of blockchain technology but there are no implementations including proofs. RDBMS is more mature than the relatively new blockchain systems like HLF. Therefore, the technology

needs more research and experimentation to serve as a storage system for (life insurance) companies on an adequate level like RDBMS.

Thus, a future research project could be the development of a prototype. This could present a proof, e.g. for HLF, of how to migrate data in a similar quality as RDBMS does. The quality obviously has to be defined. Another work in the future could be about the 'layer above' the data storage (database management systems). This is most responsible for the preservation of information security including confidentiality and integrity. It also offers the possibility to extend the database with programmatic logic. For RDBMS this refers to stored procedures and for the blockchain systems it refers to smart contracts. For this use case the database management systems were not investigated, because there was no need to use them as the business logic is outsourced to the data processing programs. However, smart contracts seem to be a powerful tool compared to the limited possibilities stored procedures in RDBMS provide.

As more research is invested in this technology, it could become more viable as a storage option in the future. Even if the work has come to the conclusion that blockchain technology is currently not suitable for storing contracts, the findings are still valuable to companies with similar use cases that have considered using the technology. These results may support their decision process.

List of Figures

Bibliography

[BXW19] Bandara, HMN ; Xu, Xiwei ; Weber, Ingo: Patterns for Blockchain Migration. In: *arXiv preprint arXiv:1906.00239* (2019)

[CMC19] Chitti, Phani ; Murkin, Jordan ; Chitchyan, Ruzanna: Data Management: Relational vs Blockchain Databases. In: *International Conference on Advanced Information Systems Engineering* Springer, 2019, S. 189–200

[CNYM12] Chung, Lawrence ; Nixon, Brian A. ; Yu, Eric ; Mylopoulos, John: *Non-functional requirements in software engineering*. Bd. 5. Springer Science & Business Media, 2012

[Cod70] Codd, Edgar F.: A relational model of data for large shared data banks. In: *Communications of the ACM* 13 (1970), Nr. 6, S. 377–387

[DLZ+18] Dinh, Tien Tuan A. ; Liu, Rui ; Zhang, Meihui ; Chen, Gang ; Ooi, Beng C. ; Wang, Ji: Untangling blockchain: A data processing view of blockchain systems. In: *IEEE Transactions on Knowledge and Data Engineering* 30 (2018), Nr. 7, S. 1366–1385

[EUd] European Commission: *EU General Data Protection Regulation*. `https://eur-lex.europa.eu/legal-content/EN/TXT/?qid=1575809341428&uri=CELEX:32016R0679`

[How07] Howard, Philip: Data Migration. In: *A White Paper by Bloor Research* (2007), S. 1–15

[Inf18] Informationstechnik, Bundesamt für Sicherheit in d.: *Zuordnungstabelle Zuordnung ISO/IEC27001 sowie ISO/IEC27002 zum modernisierten IT-Grundschutz*. Apr 2018

[ISO11] ISO/IEC: ISO/IEC 25010 System and Software Quality Models. ISO/IEC, 2011. – Forschungsbericht

[ISO16] International Organization for Standardization: Information technology – Security techniques – Information security management systems – Overview and vocabulary. Geneva, CH, Februar 2016. – Standard

[KE15] Kemper, Alfons ; Eickler, André: *Datenbanksysteme: Eine Einführung*. 10. Berlin : De Gruyter Oldenbourg, 2015 (De Gruyter Studium). – ISBN 978–3–11–044375–2

[MA19] McAliney, Peter J. ; Ang, Ban: Blockchain: business' next new "It" technology—a comparison of blockchain, relational databases, and Google Sheets. In: *International Journal of Disclosure and Governance* (2019), Aug.

http://dx.doi.org/10.1057/s41310-019-00064-y. – DOI 10.1057/s41310–019–00064–y. – ISSN 1746–6539

[NGS+19] Nathan, Senthil ; Govindarajan, Chander ; Saraf, Adarsh ; Sethi, Manish ; Jayachandran, Praveen: Blockchain Meets Database: Design and Implementation of a Blockchain Relational Database. In: *arXiv preprint arXiv:1903.01919* (2019)

[ØUJ17] Ølnes, Svein ; Ubacht, Jolien ; Janssen, Marijn: *Blockchain in government: Benefits and implications of distributed ledger technology for information sharing*. 2017

[RGG+18] Rauchs, Michel ; Glidden, Andrew ; Gordon, Brian ; Pieters, Gina C. ; Recanatini, Martino ; Rostand, Francois ; Vagneur, Kathryn ; Zhang, Bryan Z.: Distributed ledger technology systems: a conceptual framework. (2018)

[RSA78] Rivest, Ronald L. ; Shamir, Adi ; Adleman, Leonard: A method for obtaining digital signatures and public-key cryptosystems. In: *Communications of the ACM* 21 (1978), Nr. 2, S. 120–126

[Str] *Strafgesetzbuch (StGB).* https://www.gesetze-im-internet.de/stgb/index.html

[W+14] Wood, Gavin u. a.: Ethereum: A secure decentralised generalised transaction ledger. In: *Ethereum project yellow paper* 151 (2014), Nr. 2014, S. 1–32

[Weg04] Weglarz, Geoffrey: Two Worlds Data-Unstructured and Structured. In: *DM REVIEW* 14 (2004), S. 19–23

[YMRS18] Yaga, Dylan ; Mell, P ; Roby, N ; Scarfone, K: NISTIR 8202 Blockchain Technology Overview. In: *Retrieved from National Institute of Standards and Technology, US Department of Commerce* (2018)