Multidimensional Analysis of Blockchain Data Using an ETL-based Approach

Giorgio Camozzi, Felix Härer, and Hans-Georg Fill

University of Fribourg
Digitalization and Information Systems Group
Fribourg, Switzerland
{giorgio.camozzi, felix.haerer, hans-georg.fill}@unifr.ch

Abstract. *In this paper, a multidimensional model and an ETL workflow for blockchain data analysis are proposed. The workflow makes use of state-of-the-art open source tools and traditional data warehousing techniques to implement an extensible and efficient solution for the extraction, transformation, loading, and querying of data. Two scripts are implemented that aim at streamlining the ETL process, therefore simplifying the replication of the workflow to carry out a data analysis. Moreover, an exemplary use case demonstrating the analytical potential of the multidimensional model is presented and evaluated. Finally, strengths and limitations of this approach as well as the potential of future research are shown.*

Keywords: Blockchain, Ethereum, Token, ETL

1 Introduction and Motivation

In 2008, an individual, or group of people, going by the pseudonym of Satoshi Nakamoto, published a paper outlining a peer-to-peer electronic cash system named Bitcoin [1]. It was the first application of blockchain technology and gave birth to a plethora of new systems of its kind [2]. Five years later, Vitalik Buterin proposed Ethereum[1], a blockchain system that would allow the decentralized deployment and execution of programs [3]. The notion of executing programs in a decentralized and trustless fashion widened the horizons on the possibilities of blockchains and their applications. In recent years, the use of blockchains has been observed in various domains from Internet of Things[2] to food supply chains[3].

With the rise in popularity and adoption of blockchain technology, the amount of data stored on these distributed ledgers keeps increasing. The Bitcoin blockchain, for example, stored about 270 GB of data as of April 2020 according to Blockchain.com[4]. One year later, it amounted to around 336 GB. As all information in permissionless blockchains is publicly available, it is well suited for data analysis [4].

[1] See `https://ethereum.org/en/`(accessed 19-09-2021)
[2] See `https://iotex.io/`(accessed 20-09-2021)
[3] See `https://origintrail.io/`(accessed 20-09-2021)
[4] See `https://www.blockchain.com/charts/blocks-size` (accessed 19-09-2021)

According to Galici et al. [5], analyzing blockchains can provide insights into the use of the technology. They claim it can help in analyzing the usage and adoption of blockchains, possibly detecting instances of criminal uses of the platforms as well as predictive analytics about cryptocurrencies, among further statistical evaluations.

Possible analyses include the analysis of transactions during a specific time frame with the account balances of various accounts in the network. Secondly, an analysis utilizing publicly available data can be carried out for determining the metrics of transaction data of particular tokens. One example is the velocity of a specific token, i.e., the average duration until a coin is exchanged. This can provide insight into the usage and volatility of tokens and tokenized assets. Thirdly, a further useful analysis can be made to look at the token economy of a smart contract and understand, e.g., its adoption, trends and performance. This can be interesting from the point of view of an end-user as well as for a company or developer behind a smart contract.

In this paper we will describe a design artifact in the form of an approach and a multidimensional database schema for the Ethereum blockchain for conducting multidimensional analyses of blockchain data using a design science research methodology [6]. For demonstration purposes, we will conduct a token analysis of ResearchCoin[5] with a focus on available transaction and block data. The goal of this exemplary analysis is to show the practicality of the approach as well as its strengths and weaknesses.

Applying established methods in data warehouse design to blockchain data allows for great flexibility stemming from the inherent adaptability of the workflow, e.g., through the ability of changing the granularity of the analysis as needed. The potential applications of analyses of blockchain data are wide ranging, and to the best of our knowledge, neither structured or automated approaches, nor tools for an analysis of blockchains using multidimensional models exist. Therefore, we are going to present a novel solution to address this research gap.

As a foundation we used state-of-the-art tools specific to Ethereum such as Ethereum ETL[6] and OpenEthereum[7] for the synchronization and extraction of data. In addition, we developed additional scripting-tools based on Python that enabled us to add further dimensions and systematically automate the use of Ethereum ETL, thus achieving continuous and unattended transformations and loading of data. By publicly providing the code and a thorough documentation, we aim to motivate users to extend and further adapt the tools and methods to tailor them to their specific needs.

2 Research Methodology

The research and experiments presented in this paper were conducted following the design science research methodology proposed by Peffers et al. [6]. A representation of these steps is depicted in Figure 1. In a first step, the problem was identified as being a lack of efficient, flexible and automated data analysis methods for blockchain data using

[5] `https://medium.com/researchhub/announcing-researchcoin-bc075d4a3235` (accessed 02-22-2021)

[6] See `https://github.com/blockchain-etlethereum-etl/` (accessed 12-09-2021)

[7] See `https://openethereum.github.io/` (accessed 19-09-2021)

multidimensional models. In a second step, the objective of our research was defined. This is the creation of an efficient and flexible, automated and continuous process for the extraction of blockchain data and its storage in a data store to enable user-defined queries for data analysis purposes. Step three consisted of designing a database schema according to established methods in data warehouse design and developing the solution, which was done by extending and combining readily available open source solutions with our own implementation. The solution is developed for the Ethereum blockchain specifically. A demonstration of the working prototype was done in a fourth step by way of an exemplary token analysis on ResearchCoin[8] as well as through general testing to ensure correctness and integrity of the data. In a fifth step, we evaluated the solution and discussed its advantages and disadvantages as well as limitations and possible improvements. Finally in step six, the solution was communicated by sharing the source code with replication steps on GitHub [7], and through this paper to the scientific community.

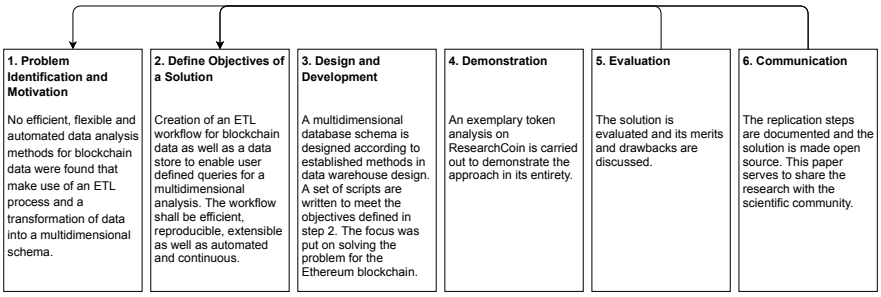

Figure 1. Adapted version of the Design Science Research methodology proposed by Peffers et al. [6].

3 Related Work

Ali and Wrembel studied state of the art techniques and various advancements in conceptual and logical model design for ETL workflows [8]. They benchmarked these existing approaches and identified their limitations. They concluded that there are several techniques in use for the conceptual and logical modeling of ETL workflows. Still, as they state in the paper, there is no standard procedure in use. As such, the authors say that it is hard to compare and contrast different approaches. They make an appeal for a standardized model for ETL workflows. They also highlight the importance of a framework assisting the ETL developer in the task of modeling, and in the research on performance enhancing techniques.

As there is great interest today in data analysis, many alike have proposed techniques to analyze blockchains. Pinna et al. used Petri Nets to model transactions and addresses in Bitcoin [9]. They modeled addresses as places and transactions as transitions. The

[8] See `https://etherscan.io/token/0xd101dcc414f310268c37eeb4cd376ccfa507f571` (accessed 12-09-2021)

experiments were carried out using the first 1.5 years of Bitcoin blockchain data. With their model, they were for example able to cluster individual addresses to groups of addresses belonging to the same entity. As they state in the paper, Petri Nets naturally lend themselves to the modeling of Bitcoin data and provide a great opportunity for blockchain data analysis.

XBlock-ETH was proposed by Zheng et al. [4]. It is a framework created for the analysis of blocks, traces and receipts on Ethereum. They were able to use it to benchmark the Ethereum network's performance as well as to make predictions on the behavior of gas prices. In their paper, further applications for their framework are stated, such as the analysis of smart contracts for fraud detection or of cryptocurrency prices.

Galici et al. used traditional data warehousing techniques to implement an efficient workflow for data extraction, transformation, and loading for the Bitcoin blockchain [5]. The data target was a relational database, enabling user-specific queries on Bitcoin addresses and transactions. They were able to analyze the data and, e.g., study changes in user behavior on the Bitcoin network.

Medvedev and the D5 team developed Ethereum ETL[9], an open source tool for the extraction, transformation and loading of Ethereum blockchain data. Supported are the extraction from a Ethereum node software with transformation and loading into a database. It is limited in scope to specific attributes used by the node software, without syntactic transformations for deriving fine-grained dimension attributes and without schema transformations that allow for multidimensional analysis. Ethereum ETL will be applied for extraction into a data staging area and streaming into a relational database.

Further works exist in the areas of block explorers [10] and visual analysis [11, 12]. In these approaches, an analysis of the dimensions address, transaction, and block is possible and, in some cases, visualized. However, the multidimensional analysis of arbitrary dimensions with data aggregation, e.g., for smart contract attributes or for custom dimensions of specific smart contracts, is not supported.

4 An ETL-based Data Analysis Process for Blockchain Data

A blockchain consists of a ledger that is distributed among peers in a peer-to-peer network [13]. Blocks, containing a list of transactions, a timestamp and a pointer to the preceding block among further attributes, are added to the blockchain by miners [14] and create a chain-like data-structure. A key property of blockchains is the near-immutable nature of the ledger [1], achieved by including a hash of a block in the succeeding block.

4.1 Data Structure of the Ethereum Blockchain

The Ethereum blockchain is a decentralized, quasi Turing-complete system [14] that was first introduced by Vitalik Buterin in 2013 [3]. At its core lies the Ethereum Virtual Machine (EVM), a global singleton run by all peers in the network, capable of executing smart contracts [14]. Smart contracts are immutable programs with an associated address that run in a distributed manner [14]. A contract's address can be used as target of

[9] See `https://github.com/blockchain-etl` (accessed 12-09-2021)

an Ether transaction to make function calls to the smart contract. The execution of a smart contract requires the user to pay a fee called gas that generally rises with the computational complexity of a program and the utilization of the network. This fee is paid in Ether, which is the currency the whole network is based upon. Smart contracts adhering to a specified standard are also known as tokens. The two prevailing standards are ERC20 for fungible tokens and ERC721 for non-fungible tokens (NFTs), defining functions to be implemented in the smart contract's code. Tokens have been used among many other things, as digital currency, e.g., the Basic Attention Token[10], or to determine ownership of a digital good. An example for this is the Twitter CEO's first ever Tweet[11], who's ownership was sold for nearly three million US dollars.

According to the specifications described in [15], a block in the Ethereum blockchain contains among other information, the hash of the parent block, the number of all ancestor blocks, the limit of gas that can be used per block, the amount of gas that was used in total by all transactions in the block and the timestamp when the block was mined. Each block also contains a list of transactions that have been mined.

Some of the main data contained in transactions is the price for gas in Wei dependent on the complexity of the transaction, the maximum amount of gas that should be used for a transaction, the receiver of the transaction and the amount of Ether to be sent [15].

4.2 Schema and Multidimensional Analysis

The first step in the design and development of the approach is the conceptualization of a data schema suitable for analyzing attributes over multiple dimensions. Depending on which entities are to be extracted, specific tables must be set up for the extraction process. As we limited our research to transactions and blocks, two corresponding tables were implemented. Their design is derived from the specifications of Ethereum ETL[12], the tool used for the extraction. Further entities can also be extracted, though we will disregard this fact for the purpose of this paper. The goal here is to generate a multidimensional database schema starting from two-dimensional data. This is achieved through a sequence of transformation steps.

The database schema for the multidimensional analysis consists of extraction, transformation and loading tables as well as the final star schema. Each step in the process requires multiple dimensions, each dimension containing attributes either directly carried over from the extraction process or derived from other values.

The core of the workflow is based on a star schema. It consists of six tables and represents the interface between the user and the data. Figure 2 shows a visualization of it. From the dimensions and its attributes, data aggregations of individual dimensions and arbitrary combinations of dimensions can be computed. E.g., commonly used dimensions such as blocks and transactions (see Section 3) in addition to, e.g, addresses, gas usage, or smart contract methods in relation to timespans or accounts. At the heart of the star schema lies the fact table. It serves the purpose of connecting all dimensions at a

[10] See `https://basicattentiontoken.org/` (accessed 11-09-2021)

[11] See `https://v.cent.co/tweet/20` (accessed 11-09-2021)

[12] See `https://github.com/blockchain-etl/ethereum-etl-postgres/tree/master/schema` (accessed 30-03-2021)

transaction level. It is also possible to set the grain to be at the block level, though the decision was made to have more fine-grained information in the star schema. Surrogate keys are used both for the transaction and block dimensions. Natural keys are used for the remaining dimensions. They are well suited, while still providing valuable information, such as the date and time, without the need to join further dimension tables. Though unusual in data warehousing, the decision was taken to have two distinct dimensions for date and time. This is done to keep space requirements as low as possible. The account dimension has two references, one from the `account_to` and one from the `account_from` attribute.

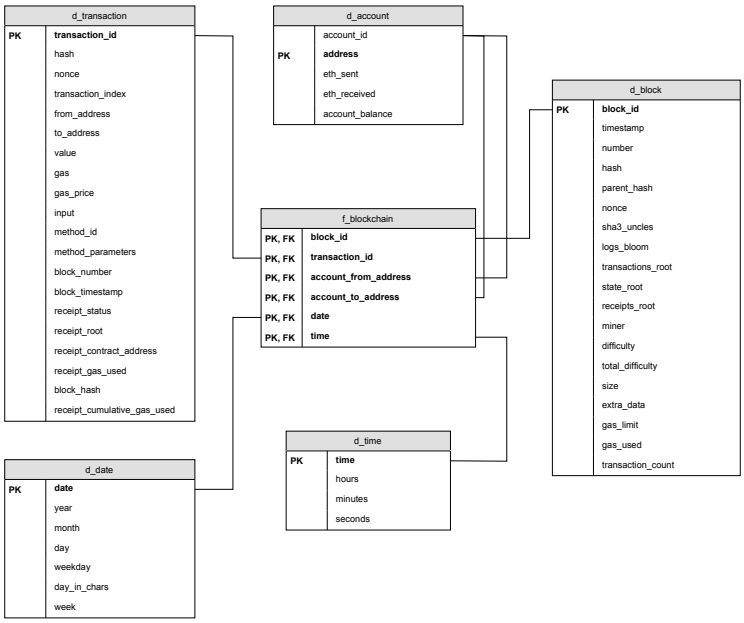

Figure 2. Visualization of the star schema with its fact table and five dimensions storing the blockchain data.

Two extraction dimensions `e_d_transactions` and `e_d_blocks` are created afterwards, compatible to the schema used by Ethereum ETL. This tool is executed for extraction from an Ethereum node, streaming data into the database as the blockchain is updated. Data is extracted into the two tables, only taking into account new data not existing in the schema beforehand.

In the next phase, the transformation and the values in the multiple dimensions of the star schema are computed. Further information on concrete transformations can be found in Section 4.4. Figure 3 shows the tables used in this step, using the dimensions of the final star schema with two additional temporary tables for calculating account balances.

The loading process consists of simply inserting the data in the transformation tables into the corresponding dimension table in the star schema.

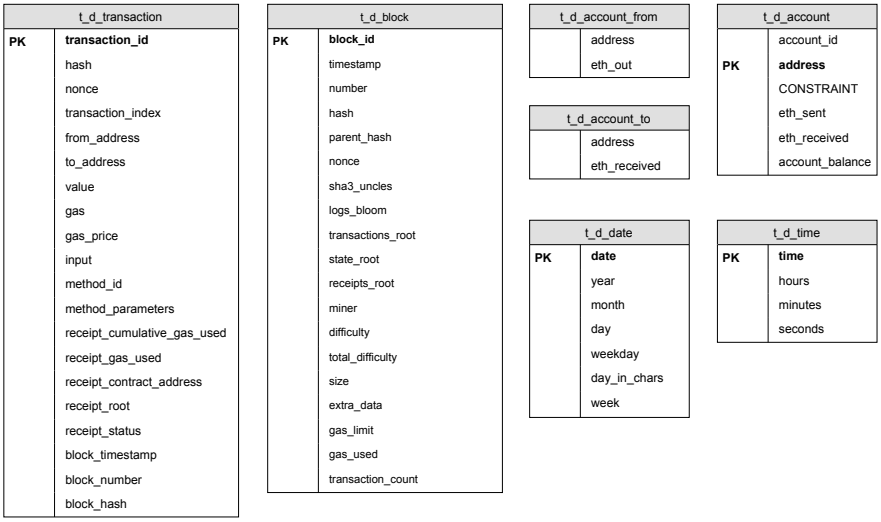

Figure 3. Visualization of the transformation dimensions used in the second ETL process step.

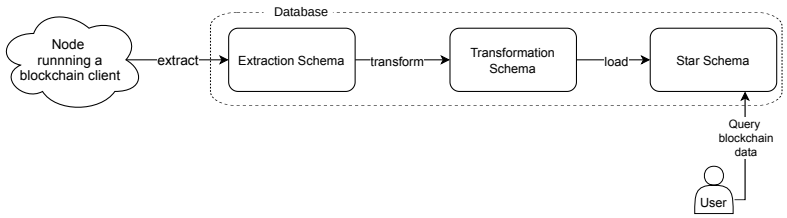

Figure 4. Data analysis setup of a node client synchronizing blockchain data, with the data being extracted, transformed and loaded in a database. A user carries out queries on the final star schema.

4.3 Extraction

The extraction phase is the initial step of the ETL process, concerned with the collection of raw data from one or more data sources.

The extraction process requires access to an Ethereum client. There exist multiple implementations of Ethereum clients[13], though for compatibility reasons we use the OpenEthereum client[14]. Furthermore, the workflow makes use of Ethereum ETL, an open source tool developed by [16]. This tool consists of Python scripts that extract transaction and block data from an OpenEthereum node, among other things. A wrapper around Ethereum ETL [7] was created for the purpose of facilitating the usage of the tool. Once started, the script requires the user to provide information about the target database (database location and credentials), the location of an OpenEthereum node, as

[13] See `https://ethereum.org/en/developers/docs/nodes-and-clients/#clients` (accessed 19-09-2021)

[14] See `https://openethereum.github.io/`(accessed 19-09-2021)

well as the data to be extracted (e.g. the starting block number and entities of interest). The corresponding Ethereum ETL *stream* command is then generated and Ethereum ETL is launched.

4.4 Transformation

The goal of the transformation phase is twofold: transforming data into multiple transactions, and carrying out syntactic and possibly semantic transformations where data harmonization is required.

During the transformation the two-dimensional schema that is the one with transaction and block tables is transformed into a multidimensional schema with, in our case, five analysis dimensions and one fact table for the metrics of interest in the particular case. The goal here is to create new data from already available data. This is done through a series of syntactic and semantic transformation steps consisting of aggregating values – e.g. calculating the amount of Ether sent by one account, deducing information – e.g. the day of the week from a timestamp, splitting attributes – e.g. separating the method id from the parameters of an input, or cleaning up data – e.g. assigning the zero hex address to all `NULL to_address` attributes. In our case, only syntactic transformations were needed. Worth noting are the account tables, which can be used to calculate account balances without the need for receipts. Here, two temporary tables are created where a row with an `address` and an `eth_out` attribute is inserted in the `account_from` table for each individual transaction from a specific address. The amount of outgoing Ether is computed by multiplying the gas used by the transaction and the gas price for the transaction. The obtained value is then added to the Ether sent by the account for the specific transaction and multiplied by minus one. This is to simplify calculations later on. Analogously, the `account_to` table is also populated, the incoming Ether here is simply the amount that was sent from the sender, as the receiver does not carry any gas costs. Thereafter, the `t_d_account` table is populated.

4.5 Loading

The final phase consists in loading the transformed data and make all data available for analysis along the dimensions defined before.

The process of loading the data into the star schema is automated and happens once all transformations have been done. From each transformation table the data is simply inserted as is into the corresponding dimension. Once the data has been inserted into the dimensions, the fact table is populated with one primary key per dimension – with the exception of the account dimension which is doubly referenced – and a composite key of the primary keys is created. Additionally, we carried out a verification step querying the first and last block of each month that was extracted in order to make sure that everything has been loaded correctly.

Updating the database at a later stage requires the whole process to be repeated. First, the OpenEthereum node must be synchronized to the desired block number, then the extraction using the wrapper script has to be carried out again, which automatically starts from the last synchronized block. Finally, the ETL script is run to transform and load

the new data into the existing star schema. This whole process is incremental, which means only new blocks are added to the star schema and the ones already inserted are not modified or updated in any way.

4.6 Queries

The benefit of multidimensional analyses is the ad-hoc construction of queries along the dimensions. A star schema enables a multitude of queries to be easily constructed. In multidimensional models, it is possible to slice and dice data, while still keeping the complexity of queries low compared to, e.g., a normalized schema. This is due to the fact that less transformation steps are usually required, as they have already been carried out during the ETL process.

Possible queries include, but are not limited to queries on the account holders, their transactions and balances during a specific time frame. Furthermore, it is possible to analyze the token economy of a specific smart contract – if applicable to the specific blockchain at hand – by, e.g., analyzing its usage in a network. This can be valuable for a business providing a smart contract, but also for any third party to gain insights into the usage of a specific token. Moreover, analyses of the volatility of a token can be done by, e.g., measuring the duration from the time a token is received until it is exchanged again.

5 Demonstration with the ResearchCoin Use Case

For the demonstration of our artifact we selected ResearchCoin tokens. Therefore, we focused on Ethereum blockchain data for the month of June 2021 and made use of the fact table in conjunction with the transaction dimension. Already with just one dimension and the fact table, we were able to retrieve valuable information on the use of ResearchCoin tokens. If necessary, this workflow could be further optimized, e.g., by making use of indexes, material views and aggregating intermediate results.

5.1 Purpose of ResearchCoin

ResearchCoin (RSC) is an Ethereum based ERC20 token that is used as a monetary incentive for contributing to ResearchHub[15]. The latter, crowd-sources the curation of scientific articles and tries to improve their accessibility by removing paywalls. The number of RSC received for a specific contribution depends on a number of factors, for example, the number of upvotes of a shared article[16]. There also exists a reputation metric for contributors, which decreases with increasing downvotes and has a direct impact on the privileges of the contributor. Furthermore, ResearchHub is set up as a Decentralized Autonomous Organization (DAO) and RSC holders have one vote in the DAO per token they possess. RSC also serves as a means to fund research proposals and to reward researchers for content a user deems to be valuable.

[15] See `https://www.researchhub.com/about` (accessed 15-10-2021)

[16] See `https://www.notion.so/ResearchCoin-21d1af8428824915a4d1f7c0b6b77cb4` (accessed 15-10-2021)

5.2 Data Analysis of ResearchCoin

All of the experiments were conducted on blocks from the month of June 2021. The first queries that were carried out are basic analyses, e.g., the total transactions made to the RSC smart contract address. This specific query showed that only 29 transactions were made to the smart contract during the month of June 2021. It must be noted that only RSC that is withdrawn from ResearchHub is actually stored on-chain. All Research Coin transactions to contributors are stored off-chain and it is thus not possible to determine how many RSCs people are currently holding from this data.

In a next step, we look in more detail at the reasons for the transactions made. First, the method IDs occurring in transactions to the smart contract are identified. Method IDs are assigned to functions written with a high-level language such as Solidity and assigned IDs when compiling them to bytes code. The use of method IDs is (1) for calling functions using only byte code through the so-called ABI (application binary interface) and (2) for addressing and actually carrying out the call within the Ethereum virtual machine (EVM). We see that only two distinct methods have actually been called, one is the ERC-20 transfer method with the ID 0xa9059cbb and the other is the ERC-20 approve method with the ID 0x095ea7b3.

Looking at the transfer method in detail by querying the hash of a function with the method and its parameters, we can find out how many ResearchCoins were transferred to which Ethereum account. Detailed queries and results can be found on GitHub[17].

Furthermore, it is possible to track transactions made to token exchanges. For exchanges between RSC and other coins or tokens to occur, the account holder sends an approve transaction to the RSC contract with the exchange smart contract address as spender parameter. Data about all of these kinds of transactions can be retrieved by means of a simple query, which can be found on GitHub[18]. Secondly, the account holder sends a transaction to the exchange smart contract, any time after the approval transaction. This transaction is the token exchange with parameters such as `amountIn`, the address where to send the exchanged tokens to, and a path which is an array of multiple addresses of ERC-20 smart contracts. Each pair of addresses specifies an exchange.

6 Evaluation and Discussion

In the following two subsections, we will first lay out various technical details, before wrapping up the evaluation with a discussion of the benefits and limitations of the proposed approach.

6.1 Technical Evaluation

Tests were done on a machine running Ubuntu 20.04, PostgreSQL Server 12, OpenEthereum v3.2.6, Ethereum ETL 1.6.0, Ethereum Data Analysis Scripts v1.0.0 and Python 3.8.

[17] See `https://github.com/grgcmz/eth-data-analysis/blob/master/RSCAnalysis.md`

[18] See `https://github.com/grgcmz/eth-data-analysis/blob/master/RSCAnalysis.md#erc-20-functions-and-parameters`

To set up the OpenEthereum node, the command shown in the documentation[19] was used. This command starts a full node on the machine it is run. In a full node all transactions are verified from the block zero. Unlike in an archive node, the state trie is pruned, i.e., not all states are stored on disk. This reduces disk space requirements, while still keeping most benefits that come from verifying all transactions.

The size on disk to store the blockchain as of the end of June 2021 amounts to around 680 GB. The synchronization took about 12 weeks running on a machine with an AMD 3700x processor, 32 GB of RAM and an NVMe SSD. The PostgreSQL database size for the extracted block and transaction data from 01-01-2021 to 30-06-2021 amounts to 209 GB. This database consists of the tables for Ethereum ETL, the extraction, transformation and loading tables as well as five dimension tables and one fact table.

As the experiments were conducted on data from the month of June 2021, the extraction process was started using `ethereum_etl_wrapper` from block 11565019. The transformation was again done by restricting the blocks to be transformed, which can be accomplished from within `etl_postgres.py`. It must be noted that restricting the extracted and transformed blocks will invalidate account balances, as these are calculated using the complete transaction history.

6.2 Benefits and Limitations

In light of the limitations of existing approaches regarding a flexible and automated data analysis (see Section 2), especially considering the use of arbitrary dimensions and aggregations (see Section 3), we identified the following benefits and limitations.

One of the advantages of having an ETL workflow and modeling data using a star schema is the flexibility gained in the possible queries and the choice of data source. Through the addition of further dimensions and transformation steps, a multitude of analytical needs can be covered. A further advantage of this approach is the ability to carry out transformations on the data. This leads to faster and simpler queries, as some information, such as account balances, does not have to be computed during the query. Furthermore, a core strength of this approach is the fact that the principles described in this paper are applicable to any type of blockchain based network, as they are quite generic and can be adapted as needed. As of the time of writing, the Blockchain ETL project provides ETL scripts for Ethereum, Bitcoin, Litecoin, Dash, Zcash, Dogecoin and Bitcoin Cash. It must be noted that these blockchains are not directly compatible with the scripts described in this paper.

The approach has some limitations. The first one being that it does not allow for fully continuous data streaming into the star schema. OpenEthereum and Ethereum ETL (and therefore also the Ethereum ETL wrapper) can be run continuously. The ETL process is what holds back the system in this regard. The script etl_postgres.py was designed with user interaction in mind, and as such requires input from the user. However, changing the program with hard coded decisions for a specific database is trivial. Scheduling a script to run in specific time intervals would then allow for a nearly continuous stream of data.

[19] See `https://github.com/grgcmz/eth-data-analysis/blob/master/AnalysisSetup.md#openethereum-configuration`

A further limitation of this system, is the fact that it does not support traces. Using traces would allow us to, e.g., distinguish between internal and external transactions. As of now, all account balances are calculated without the use of traces. This means that they are not completely accurate and must be calculated starting from block zero. For this same reason, we cannot distinguish between the creation of a smart contract, a call to it, or its deletion.

Moreover, the process is very time and resource intensive. Our experiments were carried out on the blocks mined in June 2021. Still, the synchronization has to be started from block zero, if verifying all transactions is a necessity. This entails storing roughly 680 GB of data to synchronize the node up to June 2021 as well as 340 GB more to store the extracted data in a PostgreSQL database.

On the other hand, it is possible to speed up the synchronization process and greatly reduce the size of the blockchain by starting from a snapshot of another peer without verifying all transactions before. This implies trust in a third party though, which goes against the core principles of trustless networks.

Overall, the proposed workflow is very promising. Although it was only demonstrated on the Ethereum blockchain, there is nothing in the way of adapting the process for many more blockchains. The sheer volume of data to store and process, as well as the time and resources required, might just be the Achilles heel of this approach to blockchain data analysis. Still, there are ways to partially work around these limitations.

7 Conclusion and Outlook

The overarching goal of our research was to develop a flexible and extensible database schema and ETL process to enable an analysis of blockchain data. We focused on the Ethereum blockchain for this research. OpenEthereum was used to synchronize a full node locally and Ethereum ETL for the data extraction. Furthermore, we designed a multidimensional model using a star schema, and developed two scripts to facilitate the ETL process. Finally, we evaluated the approach on the use case of ResearchCoin. All code is open source to encourage further development of the workflow and provide a contribution to the research field of blockchain data analytics.

Future research will entail solving the limitations described in the previous section, as well as expanding the capability and applicability of the proposed workflow. The additional support for further blockchains will be subject of future research and experimentation. Nonetheless, the proposed workflow lays important foundations to build upon and hopefully catalyses further research and development of workflows making use of ETL processes and multidimensional models in the field of blockchain data analysis.

8 Acknowledgements

This work was supported by the Swiss National Science Foundation project Domain-Specific Conceptual Modeling for Distributed Ledger Technologies [196889].

References

1. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system. Cryptography Mailing list at https://metzdowd.com (10 2008)
2. Dixon, M.F., Gel, Y., Kantarcioglu, M., Akcora, C.G.: Blockchain data analytics. IEEE Intelligent Systems (12 2018)
3. Buterin, V.: Ethereum: A next-generation smartcontract and decentralized application platform. GitHub repository 1 (2013), https://github.com/ethereum/wiki/wiki/White-Paper
4. Zheng, P., Zheng, Z., Wu, J., Dai, H.N.: Xblock-eth: Extracting and exploring blockchain data from ethereum. IEEE Open Journal of the Computer Society 1, 95 – 106 (2020)
5. Galici, R., Ordile, L., Marchesi, M., Pinna, A., Tonelli, R.: Applying the etl process to blockchain data. prospect and findings. Information 11, 204 (04 2020)
6. Peffers, K., Tuunanen, T., Rothenberger, M.A., Chatterjee, S.: A design science research methodology for information systems research. Journal of Management Information Systems 24(3), 45–77 (2007), https://doi.org/10.2753/MIS0742-1222240302
7. Camozzi, G.: grgcmz/eth-data-analysis (May 2021), https://doi.org/10.5281/zenodo.4756655
8. Ali, S., Wrembel, R.: From conceptual design to performance optimization of etl workflows: current state of research and open problems. The VLDB Journal 26 (09 2017), https://doi.org/10.1007/s00778-017-0477-2
9. Pinna, A., Tonelli, R., Orrú, M., Marchesi, M.: A petri nets model for blockchain analysis. The Computer Journal 61(9), 1374–1388 (2018)
10. Zhong, Z., Wei, S., Xu, Y., Zhao, Y., Zhou, F., Luo, F., Shi, R.: Silkviser: A visual explorer of blockchain-based cryptocurrency transaction data. In: 2020 IEEE Conference on Visual Analytics Science and Technology (VAST). pp. 95–106 (2020)
11. Härer, F., Fill, H.G.: A comparison of approaches for visualizing blockchains and smart contracts. In: 22nd International Legal Informatics Symposium/22. Internationales Rechtsinformatik Symposion (IRIS 2019). pp. 527–537. Editions Weblaw (2019)
12. Tovanich, N., Heulot, N., Fekete, J.D., Isenberg, P.: Visualization of blockchain data: A systematic review. IEEE Transactions on Visualization and Computer Graphics 27(7), 3135–3152 (2021)
13. Bashir, I.: Mastering blockchain. Packt Publishing Ltd (2017)
14. Antonopoulos, A.M., Wood, G.: Mastering Ethereum: Building Smart Contracts and DApps. O'Reilly Media (2019)
15. Wood, G.: Ethereum: A secure decentralised generalised transaction ledger. https://github.com/ethereum/yellowpaper (2017)
16. Medvedev, E., the D5 team: Ethereum etl (2018), https://github.com/blockchain-etl/ethereum-etl, last accessed 17 April 2021