A Unified Data Model for Blockchains

João Vicente Meyer[1], **Ronaldo dos Santos Mello**[1]

[1] PPGCC-INE - Universidade Federal de Santa Catarina (UFSC)
Florianópolis – SC – Brasil

Abstract. *The popularity of blockchains has been steadily rising since its inception in 2009. Its original focus was the creation of a distributed trustless system for a digital currency. Bitcoin was the first widely used blockchain. However, many more have been further created: Ethereum, Solana, Hyperledger Fabric, to name a few. Due to it, blockchain networks and its properties are being scrutinised by data scientists. This topic is known as blockchain analysis. However, analysis´ tasks are hard to be accomplished over multiple blockchains. Even though most have similar concepts, each one has its own data modeled differently. As a consequence, scientists end up needing to do rework in order to apply similar analytics and algorithms in more than one blockchain network. In fact, blockchain data modeling is an open issue. This master's thesis aims at proposing an unified model for blockchain data. For the best of our knowledge, there is no similar work in the literature.*

1. General information

- **Level:** Masters degree
- **Student:** João Vicente Meyer
- **Advisor:** Ronaldo dos Santos Mello
- **Admission:** November 2020
- **Qualification Exam:** March 2022
- **Defense Forecast:** November 2022
- **Completed Steps:** All mandatory course credits (2020 - 2021), Qualification Exam (2022), Bibliographic Review, Problem Statement.
- **Future Steps:** (a) Identification of entities and its attributes in popular blockchains; (b) proposal of a unified data model for blockchain data based on the schema integration of existing approaches; (d) proposal of a middleware for integrated access to several blockchain networks; (c) Thesis writing and defense; results publication.

2. Context and motivation

With the popularity of blockchain technologies, the world has seen the imagination of developers flourish with its many applications. Blockchain originated as a way to decentralise finances, with Bitcoin being the biggest and most successful example. However, with the advent of smart contract-based blockchains, like Ethereum, we have witnessed the creation of decentralised autonomous organisations (DAO), like Aragon[1], Maker[2] and The DAO[3], as well as decentralised exchanges, such as Uniswap[4], and full virtual worlds,

[1] https://aragon.org
[2] https://makerdao.com
[3] https://github.com/blockchainsllc/DAO
[4] https://uniswap.org

like Decentraland[5].

Unfortunately, new technologies come with new and innovative ways to trick their users. The pseudo anonymity given by blockchains make them useful for illegal activities, most commonly ransomware and ponzi schemes. There are already many studies describing techniques for fraud detection [Chen et al. 2018], and de-anonymisation [Nick 2015, Turner et al. 2020]. However, most studies focus on specific algorithms crafted for a single blockchain network. Although different blockchain implementations have different inner workings, the data they generate and the transactions between accounts are very similar. Still, when applying some analysis, researches ended up needing to write specific code to analyse each different blockchain. Which is very unpractical, error prone and time consuming.

Additionally, the research topic of blockchain analysis[6] [Akcora et al. 2018, Vo et al. 2018], shows us that (old) new problems, such as fast data querying and management, arise from the volume of data generated by blockchain networks, where data is often *churned* in order to extract extra information from it. Some examples of applications are: (1) Detecting accounts used by the same user; (2) Detecting account behaviours; (3) Account clustering; (4) Wealth movements; and others.

Some surveys about blockchain technology have already identified that the problem of a missing unified model exists [Przytarski et al. 2021, Huang et al. 2021]. However, no study has been found regarding a unified data model for blockchain data; and no study about the entities and the attributes that compose most blockchain networks. From this motivation and the lack of previous studies on the subject, this work intends to propose a unified data model for blockchain data. A model that can be used to represent the main entities and attributes of multiple blockchains, allowing also an integrated access layer over them.

The rest of this paper is organized as follows. Next section discusses related work. Section 4 gives an overview of our proposal, and Section 5 is dedicated to future works.

3. Related work

There are already studies that try to model, although indirectly, blockchain data. Some create easy-to-use interfaces, such as HTTP REST interfaces [Li et al. 2017, Pratama and Mutijarsa 2018] while others provide SQL interfaces that execute on top of native blockchain clients [Linoy et al. 2019, Trihinas 2019]. Finally, many studies decide to simply copy the data in order to make it accessible through popular database management systems and its interfaces [Peng et al. 2019, Bartoletti et al. 2017, Bartoletti et al. 2017, Han et al. 2019, Zhou et al. 2019].

All approaches, at some stage, end up applying some mapping on the original data. However, even though they provide some nifty, easier and useful ways to query the underlying blockchain data, no study details its data modeling approach or go further to specify an unified data model. Also, unfortunately, not all works define the entities explicitly, *i.e.*, blockchain data models are presented in a tangential way. The authors do make use of data models, but no discussion about how they defined the models is made.

[5] `https://decentraland.org`

[6] Sometimes referenced as "blockchain analytics".

For some of them, we had to search for the source code, if available, and extract the considered entities.

Besides the difficulties expressed, we were already able to identify similarities between the studies. All studies, for example, make use of the transaction entity. This is due to the fact that the transaction contains the core information of the distributed ledger, which usually is an account sending assets/coins to another account. Some studies also consider smart contracts as an entity [Bartoletti et al. 2017, Bragagnolo et al. 2018, Han et al. 2019, Li et al. 2017]. This is the case of some blockchains, like Ethereum and Hyperledger Fabric.

Despite the previous discussion of data modeling issues, we also point out other interesting features we found in the works. Most of them agree that blockchain querying is not optimal. Some of them propose solutions that make use of readily available DBMSs, such as MongoDB, PostgreSQL and MySQL, which run besides the original blockchain and duplicate their data in a structured manner. A data migration from LevelDB, a key-value storage solution, to some DBMSs, like MongoDB and PostgreSQL, is supported by some works [Han et al. 2019, Li et al. 2017, Peng et al. 2019, Pratama and Mutijarsa 2018, Zhou et al. 2019]. However, the authors do not propose a conversion process between the source and target data models of the DBMSs. Instead, they simply store the data format returned by the tools they use, like a third party API (*e.g.*, Etherscan[7]) or native blockchain clients (*e.g.*, Ethereum's geth[8]). This process could be improved by creating suitable data models for the entities and its relationships in order to make better use of the DBMS logical structure.

Also, the consideration of popular DBMSs sounds like a good approach to make data easily accessible by interested parties. Of course, the chosen DBMS should suit the information to be retrieved. For example, if only data regarding specific blocks, transactions and accounts is sufficient, without any aggregation, a key-value store is suitable. If aggregations are needed, like comparing transactions executed in a specific time window, document and relational DBMSs can be considered. Additionally, if more complex information is desired, like following the flow of transactions as funds are transferred between multiple accounts, a graph-based storage solution is better suited. So, depending on the use case, a specific DBMS can be chosen.

4. Proposal

In order to resolve the problem described previously, we intend to define a unified data model that encompasses different blockchains and its main components, such as blocks, transactions, accounts, and possibly others. Also, with the spawn of blockchains with different *structural formats*, like IOTA (a graph-based one), we intend that our data model be able to encompass the data generated by those as well. With a good model in hands, the next step would be to evaluate it by creating a tool to extract, transform and load (ETL) blockchain data and save it into a middleware system with a common query interface in order to evaluate the expressiveness of the model.

This work development is divided in four main parts: (a) *Research*, where an exploratory research will be executed, by analysing the most widely used/studied

[7] `https://etherscan.io/apis`
[8] `https://geth.ethereum.org/`

blockchains, such as Ethereum, Bitcoin and Hyperledger Fabric; (b) *Analysis*, where we identify the most common entities and attributes of each blockchain implementation studied previously; (c) *Merging*, where we will merge the resulting entities of the previous step in a single, unified data model; (d) *Evaluation*, where we will evaluate the expressiveness of our data model by checking if we can access all the relevant data when compared to the original data models.

Items (a) and (b) are already under way. So far we have already defined the models used by two different blockchain implementations: Ethereum and Nano. As shown in Figure 1, we have already identified that both blockchains have similar components, although in different forms. For example, in Figure 1 (a), the Ethereum blockchain has many transactions per block, while Nano, Figure 1 (b), has only one. Also, the state of the account entity in Ethereum is a separate entity, while in Nano, the account state is included in each transaction.

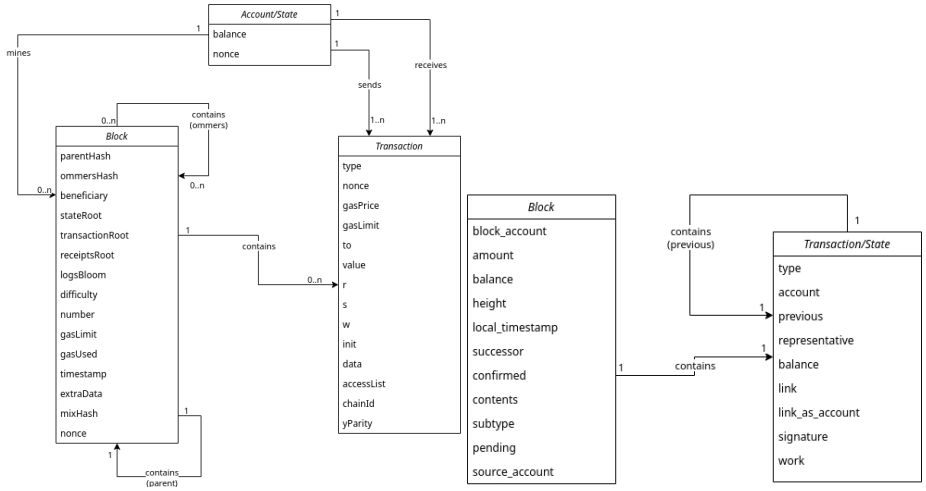

Figure 1. Class diagram showing the entities, attributes and relationships for Ethereum (a) and Nano (b) blockchains.

We also notice that many of the attributes on each implementation are similar, as presented in Table 1. Some of the attributes are very easy to identify, like “parentHash” and “previous”, where both of them point to the previous block in the chain.

However, some attributes can be misleading. For example, even though both ledgers have the attribute “type” in its transactions, they have different meanings. For Nano, the transaction type is used to identify different kinds of transactions, like “send” and “receive”. Instead, ”type” identifies which version of transaction is being used in Ethereum.

Also, there are many attributes that do not have a mapping to other blockchains. Table 2 shows some examples. Ethereum has an attribute called “gas”, and other related attributes, that refer to its smart contract nature. Gas is used to measure the *cost* of executing computations in the network. Nano, however, does not have smart contracts or anything similar, so this type of data does not exists. Yet, Nano has the attribute “representative”, which refers to other accounts which can *act* in the behalf of some account.

Ethereum	Nano	Comments
Block/parentHash	Transaction/previous	Hash of the previous block in the chain
Account/balance	Transaction/balance	Current balance of the account
Block/timestamp	Block/local_timestamp	Timestamp of when the block/transaction was mined
Transaction/from	Transaction/source_account	Account that originated the transaction (sender)
Transaction/r,s	Transaction/signature	Cryptographic signature of the creator of the transaction (sender)
Block/nonce	Transaction/work	Proof of work

Table 1. Table showing similar attributes in both blockchains.

This behaviour and consequently, the attribute, do not exist in Ethereum.

Ethereum	Nano	Comments
gasPrice/gasUsed/gasLimit		Price/amount/limit of gas used in transactions
	representative	Account that can act in behalf of other account
	link	Linking between different account chains

Table 2. Examples of attributes exclusive to a blockchain.

From this research and analysis on existing data models for blockchain networks, we are now working on an merging strategy for them. By now, we intend to use the *bottom-up hierarchical clustering* technique [Morzy et al. 1999], as shown in Figure 2. It consists in the affinity analysis of entities from pairs of blockchain data models, A and B in the example, which will be unified by similarity in order to produce a new data model, AB. This data model AB will then be merged to another data model C in order to produce ABC. The process is repeated until all data models be analyzed and a single, universal data model be produced.

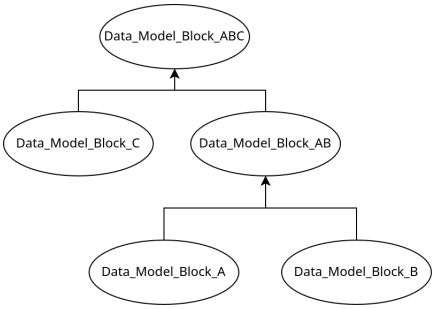

Figure 2. The *bottom-up hierarchical clustering* technique. Adapted from [Mello 2002].

The evaluation will be done as follows. After the universal model has been defined, we will load the data, using an ETL tool, into some DBMS. This way, we can query

the data using common querying interfaces, like SQL. In order to do so, we intend to create a middleware layer, that will provide a common query interface to query data from multiple blockchains. Figure 3 shows a top level view of the proposed system. Blockchain data is loaded into some DBMS and later queried, using our middleware as an entry point.

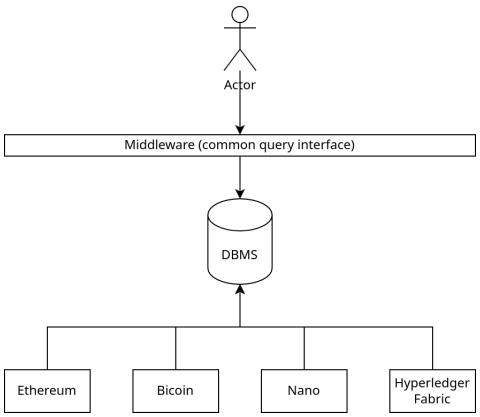

Figure 3. High level view of the proposed integration strategy using a middleware and DBMS.

This way, we will be able to evaluate if we can perform the same kind of analysis in the blockchain data as if we were using its native clients. For example, we want to be able to query blocks by its hash, a feature available in all blockchains, we want to be able to query the transactions of some block, and query account balances, and so on. The precise ways to perform this evaluation are still being discussed.

Finally, we also want to make our implementation work to query data in multiple blockchains at the same time. With the popularisation of blockchain *bridges*, *i.e.*, ways of connecting and passing data between different blockchains, we want to be able to follow information from one blockchain to another. For example, we would like to follow an user's assets from Ethereum to some other blockchain, like Arbitrum[9] or Polygon[10]. For this, we will probably need a graph-based analysis of the data.

5. Future works

The next steps of this research work is to perform analysis over more blockchain networks, most probably Bitcoin, IOTA and Hyperledger, in order to have a good set of entities that represent the main building components of blockchain networks. Finally, we will need to establish ways to measure the expressiveness of the final proposed model. One possibility is to create a set of queries that are already possible to answer using native blockchain clients, and a possible set of queries that are hard/impractical to get answers to. Another interesting metric would be the analysis of the time it takes to perform such queries, when the data is loaded in a DBMS, for example, when comparing to the native clients and our implemented middleware.

References

Akcora, C. G. et al. (2018). Blockchain Data Analytics. *Intelligent Informatics*, page 4.

[9] `https://arbitrum.io/`
[10] `https://polygon.technology/`

Bartoletti, M. et al. (2017). A General Framework for Blockchain Analytics. In *Workshop on Scalable and Resilient Infrastructures for Distributed Ledgers*, pages 1–6.

Bragagnolo, S. et al. (2018). Ethereum Query Language. In *International Workshop on Emerging Trends in Software Engineering for Blockchain*, pages 1–8.

Chen, W. et al. (2018). Detecting Ponzi Schemes on Ethereum: Towards Healthier Blockchain Technology. In *World Wide Web conference*, pages 1409–1418.

Han, J. et al. (2019). Enabling SQL-query Processing for Ethereum-based Blockchain Systems. In *Int. Conference on Web Intelligence, Mining and Semantics*, pages 1–7.

Huang, H. et al. (2021). A Survey of State-of-the-art on Blockchains: Theories, Modelings, and Tools. *ACM Computing Surveys*, 54(2):1–42.

Li, Y. et al. (2017). EtherQL: A Query Layer for Blockchain System. In *International Conference on Database Systems for Advanced Applications*, pages 556–567. Springer.

Linoy, S. et al. (2019). Scalable Privacy-preserving Query Processing over Ethereum Blockchain. In *International Conference on Blockchain*, pages 398–404. IEEE.

Mello, R. d. S. (2002). *Uma Abordagem Bottom-up para a Integração Semântica de Esquemas XML.* PhD thesis, Universidade Federal do Rio Grande do Sul.

Morzy, T., Wojciechowski, M., and Zakrzewicz, M. (1999). Pattern-oriented Hierarchical Clustering. In *East European Conference on Advances in Databases and Information Systems*, pages 179–190. Springer.

Nick, J. D. (2015). Data-driven De-anonymization in Bitcoin. Master's thesis, ETH-Zürich.

Peng, Z. et al. (2019). VQL: Providing Query Efficiency and Data Authenticity in Blockchain Systems. In *International Conference on Data Engineering Workshops*, pages 1–6. IEEE.

Pratama, F. A. and Mutijarsa, K. (2018). Query Support for Data Processing and Analysis on Ethereum Blockchain. In *Int. Symposium on Electronics and Smart Devices (ISESD)*, pages 1–5. IEEE.

Przytarski, D., Stach, C., Gritti, C., and Mitschang, B. (2021). Query Processing in Blockchain Systems: Current State and Future Challenges. *Future Internet*, 14(1):1.

Trihinas, D. (2019). Datachain: A Query Framework for Blockchains. In *International Conference on Management of Digital EcoSystems*, pages 134–141.

Turner, A. B., McCombie, S., and Uhlmann, A. J. (2020). Analysis Techniques for Illicit Bitcoin Transactions. *Frontiers Comput. Sci.*, 2:600596.

Vo, H. T., Kundu, A., and Mohania, M. K. (2018). Research Directions in Blockchain Data Management and Analytics. In *Extending Database Technology (EDBT)*, pages 445–448.

Zhou, E. et al. (2019). Ledgerdata Refiner: A Powerful Ledger Data Query Platform for Hyperledger Fabric. In *International Conference on Internet of Things: Systems, Management and Security (IOTSMS)*, pages 433–440. IEEE.