Dynamic Data Quality for Static Blockchains

Alan G. Labouseur
School of Computer Science and Mathematics
Marist College
Poughkeepsie, NY USA
Alan.Labouseur@Marist.edu

Carolyn C. Matheus
School of Computer Science and Mathematics
Marist College
Poughkeepsie, NY USA
Carolyn.Matheus@Marist.edu

***Abstract*—Blockchain's popularity has changed the way people think about data access, storage, and retrieval. Because of this, many classic data management challenges are imbued with renewed significance. One such challenge is the issue of Dynamic Data Quality. As time passes, data changes in content and structure and thus becomes dynamic. Data quality, therefore, also becomes dynamic because it is an aggregate characteristic of the changing content and changing structure of data itself. But blockchain is a static structure. The friction between static blockchains and Dynamic Data Quality give rise to new research opportunities, which the authors address in this paper.**

***Index Terms*—Blockchain, Dynamic Data Quality, Graphs**

I. Introduction

Ever since its introduction in Satoshi Nakamoto's Bitcoin white paper [1] in November 2008, developers have been trying to use the distributed ledger technology known as *blockchain* to disrupt traditional relational and graph databases. Many systems have been built in the past decade in areas as diverse as finance (e.g., *Quorum* [2] from JPMorgan Chase & Co., and *Circle* [3] backed in part by Goldman Sachs), the Internet of Things (e.g., the *Vecap* [4] smart home IoT security platform), healthcare (e.g., the *Medicalchain* [5] telemedicine platform), and supply chain management (e.g., *TradeLens* [6] from Maersk and IBM). The popularity of blockchain is changing the way people think about data access, storage, and retrieval as they connect the physical world with the digital. Because of this, many classic data management challenges are imbued with renewed significance as new research opportunities emerge for managing data in a blockchain.

One such challenge is the issue of *Dynamic Data Quality*. We live in an evolving world. As time passes, data changes in content and structure, and thus becomes dynamic. Data quality, therefore, also becomes dynamic because it is an aggregate characteristic of the changing content and changing structure of data itself. However, by its essential nature blockchain is a structure for static data. It is append-only, meaning that while data can be added to a blockchain, existing data is preserved in permanent stasis and cannot be altered without invalidating that block and all subsequent blocks. The friction between static blockchains and Dynamic Data Quality gives rise to new research opportunities. For one:

How can we align Dynamic Data Quality with a static structure like blockchain?

The remainder of this paper is organized as follows: In Section II we define *essential blockchain* to avoid getting mired in trivialities found among Bitcoin, Ethereum, Hyperledger, and other blockchain implementations. In Section III we summarize some existing data management issues in Dynamic Data Quality and, in our first contribution, document new challenges of supporting Dynamic Data Quality in static environments like blockchain. Section IV contains our second contribution: a graph-based approach to addressing the challenge of Dynamic Data Quality in static environments like blockchain. Finally, Section V concludes with what we have learned so far and suggests possible research directions.

II. Essential Blockchain

In his famous essay, "No Silver Bullet" [7], Frederick P. Brooks Jr. addresses some of the difficulties inherent in software development and offers advice on conquering its complexity. Brooks takes inspiration from the mother of all sciences, philosophy (specifically, Aristotle), in defining two terms to discuss software complexity:

essence Difficulties inherent in the nature of software.
accidents Difficulties that attend its production but are not inherent.

With those ideas in mind, we strip away accidents to avoid implementation-specific trivialities and develop *essential blockchain*. First, we define its structural parts.

Definition 1. *Transaction – a container for arbitrary data.*

Definition 2. *Block – a container for one or more transactions.*

Definition 3. *Blockchain – an ordered, append-only container for one or more blocks where the i^{th} block b_i depends on the prior block b_{i-1} to confirm b_i's permanent stasis where $i \geq 1$.*

Second, given the above definitions, and after examining many blockchain use cases in varied areas such as finance, IoT, healthcare, and supply chain, we find that blockchain is more than a data structure, it is also a consensus network of peer instances of that data structure. Now we have its essence:

Definition 4. *Essential Blockchain – a peer-to-peer network of Blockchain instances cooperating for consensus.*

III. Problems

Many general challenges in Dynamic Data Quality stem from the concept of *fitness for use*, a foundational idea in data quality research [8]. Some specific challenges come from issues of how available and retrievable data are relative to methods for accessing and storing it. We are challenged to handle a dizzying range of data types found in varying time frames of differing granularity from diverse sources in our evolving and streaming world.

Other challenges involve the ease of manipulating data into varying representations to address our changing needs for its use. For example, if we want to measure influence we would like our data as a graph, but if we want to slice and dice our data into segments we would prefer it as relational tables.

These problems of Dynamic Data Quality are currently being explored in the context of traditional graph and relational systems [9]. An emerging challenge is addressing the dynamic nature of evolving data and our changing needs for its use in a blockchain environment whose essential nature is one of append-only data in permanent stasis (Definition 3). Numerous data quality dimensions have been identified [10], [11] and are ripe research directions in the blockchain context. To get things started, we examine *Accessibility* and *Representation* here.

A. Accessibility

Accessibility refers to the extent to which data are available and easily retrievable in both detail and aggregate form. This includes the extent to which data are formatted and represented in a way that is easily retrievable for a desired task, as well as the time lapse between request, retrieval, and delivery. Batini et al. [12] propose using response or delivery time as a metric for accessibility.

Query performance is often used as a proxy for response or delivery time to measure *accessibility*. In traditional relational and graph systems this is addressed by optimizing queries and using indexes or summaries. That is a problem for blockchain because you cannot generally **query** a blockchain in the common sense of the word.[1] Rather, you must **crawl** from the most recent block backwards towards the Genesis block, searching for the target data. Without structures and metadata to support log-time search functions, the only option is to conduct a linear search, which is more expensive in terms of resources like CPU cycles and time.

B. Representation

Representation refers to the extent to which data are concisely represented, well organized, and well formatted for extracting meaningful information [10].

In traditional systems, we have flexibility to change the underlying format of our data to align with our dynamic *fitness for use* needs. For example, data may be initially captured and stored in JSON format but later transformed to a graph for influence queries or to relational tables for creating segmentations based on common attributes. This is a problem for blockchain because its essential static nature does not permit us the flexibility to change its underlying format to suit our dynamic needs.

[1]The term "query" as used here implies expressing target data or aggregates in formal expressions, parsing those expressions, generating a query plan or operator network to satisfy those expressions, optimizing that plan or operator network, and executing it to retrieve the target data or compute the specified aggregates.

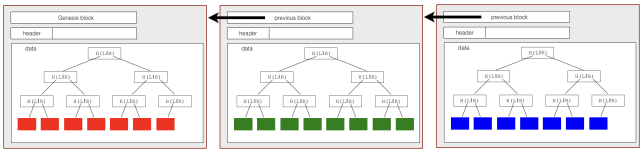

Fig. 1. Tiny blockchain example with transactions in red, green, and blue.

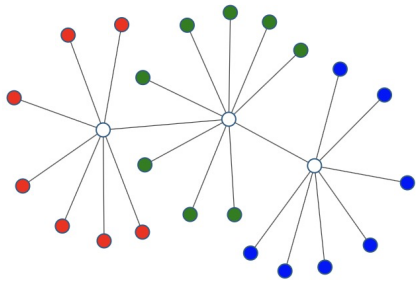

Fig. 2. The (tiny) blockchain in Figure 1 expressed **as a graph** (with transactions in red, green, and blue).

IV. Solutions

Problems of *accessibility* (e.g., querying and aggregating) and *representation* (e.g., changing the underlying structure as our needs evolve and supporting concise representation) stem from misalignment between these data quality dimensions and the essential static and linear nature of blockchain. We can align Dynamic Data Quality with a static structure like blockchain by using graphs.

A. Blockchain is Graph-like

Intuitively, blockchain is naturally graph-like and shares many characteristics of a graph. As shown in Figure 1, the chain of blocks is essentially a linked list, which is a special case of a graph. Transactions within each block are usually stored in the leaf nodes of a hierarchy (typically a Merkle tree), which is also a special case of a graph. It is therefore natural to think of blockchain in terms of graph structures like the one shown in Figure 2. It is also natural to embrace graph databases (especially distributed graph databases) and the rich field of graph analytics as tools for resolving the misalignment between Dynamic Data Quality dimensions and static blockchains.

Using these tools, we can model a blockchain using a graph $G(V, E)$, where V is a set of vertices that represent blocks or transactions within blocks, and E is a set of edges that represent hash pointers between blocks or descendents in a tree. This graph, like the one in Figure 2, is a *shapshot* of a blockchain.

Algorithm 1: Generating a graph from a blockchain API

```
new graph;
lastBlockId ← null;
blockCount ← api/status?q=getBlockCount;
for i ← blockCount-1 down to 0 do
    hash_i ← /api/block-index/i;
    block ← /api/block/hash_i;
    thisBlockId ← "block" ‖ i;
    add vertex thisBlockId;
    transactions[]_i ← /api/txs/?block=hash_i;
    foreach tx in transactions[]_i do
        thisTxId ← thisBlockId ‖ "tx" ‖ tx.id;
        add vertex thisTxId;
        add edge thisTxId – thisBlockId;
    end
    if lastBlockId ≠ null then
        add edge lastBlockId – thisBlockId;
    end
    lastBlockId ← thisBlockId;
end
```

B. Blockchain as a Graph

To address the needs of any particular industry or use case we would define specific attributes to capture in our snapshots relative to their *fitness for use* in that domain. For example, in a healthcare blockchain we may focus on patient data with regard to prescription drugs and their prescribers, allowing us to compute connectivity metrics like clustering coefficient; in a finance blockchain we may focus on asset ownership and trading with an eye towards uncovering suspicious influence through metrics like betweenness centrality and PageRank.

Algorithm 1 is a recipe for generating a graph from a blockchain using a representative `API` from BlockExplorer [13] and the graph language GSQL supported by G* Studio [14]. Executing Algorithm 1 on the tiny blockchain example in Figure 1 results in the graph in Figure 2.

C. Graph Operations to Improve Dynamic Data Quality

Once we have a graph containing details of our target industry or use case, we can write, optimize, and execute queries to gain insight about various dimensions of Dynamic Data Quality like *accessibility*. Many graph systems support high-level queries for top-k vertices by degree, clustering coefficient, PageRank, and betweenness centrality. They also support the ability to quickly and easily compute simple aggregates like `count` and `max` as well as more complex aggregates like average degree distribution and network diameter. These details and aggregates are far more accessible in a graph database than in a blockchain.

We can also improve *representation* because we are free to transform our graphs into JSON documents or relational tables as our *fitness for use* needs evolve. We could, for example, write graph queries that output SQL to create relational tables or output oddly formatted ASCII to make JSON documents.

Generating graph summaries for blockchain is a promising research direction because summaries support query efficiency and aid in visualization, both important for concise *representation*. Graph summaries also support *accessibility* as another form of aggregation. Liu, et. al [15] cover many graph summarization techniques, most of which are applicable here.

V. Conclusions and Research Directions

We proposed using graph systems to remove friction between Dynamic Data Quality and static blockchains by promoting their alignment through distributed graph storage and queries. We supported this by defining essential blockchain, showing the similarities of essential blockchain to graphs, and pointing out how graph queries and summaries can be used to enhance measures of *accessibility* and *representation*.

Our research directions include experimentally exploring these techniques with large-scale data sets using graphs generated from Ethereum or Hyperledger-based blockchains, and working with our own research blockchain[2] to investigate enhancing block structures to better support summarization and log-time search functions. Modeling blockchain evolution as a series of graph snapshots to support incremental summarization is another promising research direction.

References

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," November 2008, https://bitcoin.org/bitcoin.pdf Accessed 2019-01-01.

[2] JPMorgan Chase & Co., "Quorum: Enterprise-ready distributed ledger and smart contract platform," https://www.jpmorgan.com/global/Quorum, last Accessed: 2019-01-01.

[3] Circle Internet Financial Limited, "Circle: The new shape of money," https://www.circle.com, last Accessed: 2019-01-01.

[4] Vecap GmbH, "Vecap smarthome IoT security," https://vecap.io, last Accessed: 2019-01-01.

[5] Medicalchain SA, "Medicalchain telemedicine platform," https://medicalchain.com, last Accessed: 2019-01-01.

[6] Maersk and IBM, "Tradelens, a blockchain shipping solution," https://www.tradelens.com, last Accessed: 2019-01-01.

[7] F. P. Brooks, Jr., "No silver bullet essence and accidents of software engineering," *Computer*, vol. 20, no. 4, pp. 10–19, Apr. 1987.

[8] S. E. Madnick, R. Y. Wang, Y. W. Lee, and H. Zhu, "Overview and framework for data and information quality research," *J. Data and Information Quality*, vol. 1, no. 1, Jun. 2009.

[9] A. G. Labouseur and C. C. Matheus, "An introduction to dynamic data quality challenges," *J. Data and Information Quality*, vol. 8, no. 2, pp. 6:1–6:3, Jan. 2017.

[10] L. L. Pipino, Y. W. Lee, and R. Y. Wang, "Data quality assessment," *Commun. ACM*, vol. 45, no. 4, pp. 211–218, Apr. 2002.

[11] Y. Wand and R. Y. Wang, "Anchoring data quality dimensions in ontological foundations," *Commun. ACM*, vol. 39, no. 11, pp. 86–95, November 1996.

[12] C. Batini, C. Cappiello, C. Francalanci, and A. Maurino, "Methodologies for data quality assessment and improvement," *ACM Comput. Surv.*, vol. 41, no. 3, pp. 16:1–16:52, Jul. 2009.

[13] BlockExplorer, "BlockExplorer API," https://blockexplorer.com/api-ref, last Accessed: 2019-01-09.

[14] A. Labouseur, S. Crumlish, C. Graves, M. J. Iori, G. Miller, and T. J. Wojnar, "G* studio: An adventure in graph databases, distributed systems, and software development," *ACM Inroads*, vol. 7, no. 2, pp. 58–66, May 2016.

[15] Y. Liu, T. Safavi, A. Dighe, and D. Koutra, "Graph summarization methods and applications: A survey," *ACM Comput. Surv.*, vol. 51, no. 3, pp. 62:1–62:34, Jun. 2018.

[2]Our research blockchain is available on GitHub at https://github.com/Marist-Innovation-Lab/blockchain. All are welcome to it.