

UMT Artificial Intelligence Review (UMT-AIR)

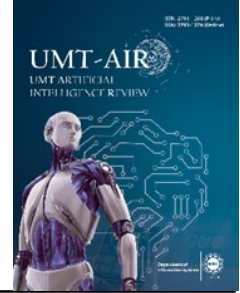
Volume 2 Issue 1, Spring 2022

ISSN(P): 2791-1276 ISSN(E): 2791-1268

Homepage: <https://journals.umt.edu.pk/index.php/UMT-AIR>



Article QR



Title: Role of GraphDB in FinTech, Blockchain Ledgers

Author (s): Hassan Kaleem¹, Sundas Rukhsar¹, Waqar Ahmad², Hafiz Ali Haris³


Affiliation (s): ¹9 Frances Street Crewe, England
²Electronic Government Authority RAK, UAE
³Communication & Works Department, GOP, Pakistan

DOI: <https://doi.org/10.32350.umt-air.21.03>

History: Received: March 10, 2022, Revised: April 25, 2022, Accepted: June 10, 2022

Citation: H. Kaleem, S. Rukhsar, W. Ahmad, and H. A. Haris, "Role of GraphDB in FinTech, blockchain ledgers," *UMT Artif. Intell. Rev.*, vol. 2, no. 1, pp. 00–00, 2022, doi: <https://doi.org/10.32350.umt-air.21.03>

Copyright: © The Authors

Licensing:  This article is open access and is distributed under the terms of [Creative Commons Attribution 4.0 International License](https://creativecommons.org/licenses/by/4.0/)

Conflict of Interest: Author(s) declared no conflict of interest



A publication of

Department of Information System, Dr. Hasan Murad School of Management
University of Management and Technology, Lahore, Pakistan

Role of GraphDB in FinTech, Blockchain Ledgers

Hassan Kaleem^{1*}, Sundas Rukhsar¹, Waqar Ahmad², Hafiz Ali Haris³

¹ Frances Street Crewe, England

² Electronic Government Authority RAK

³ Communication & Works Department, GOP, Pakistan

Abstract— GrpahDB stores data in nodes and edges, nodes represent entities and edges represents the relationship between entities. The role of GraphDB in the blockchain is described as blockchain uses blocks and these blocks are connected through hashcode to store the data. In cipher language, hash is the irreversible conversion of data which makes it impossible to decrypt. Blockchain also uses proof of work system, in which data is entered only if maximum people allows verifies it. And once anything entered into ledger, it cannot be altered or deleted. The paper has provided how hashing & indexing, query processing, transaction management, data management and data distribution is done for GraphDB into ledger, with previously done work and libraries to build and manage GraphDB blockchain.

Index Terms—GraphDB in FinTech, Neo4j, Hashing & Indexing, Query Processing, Transaction Management, Data Management, Data Distribution

I. Introduction

Blockchain is a peer to peer network that uses digital ledger system that is maintained by different computer in a network. It creates block and these blocks are linked together through hash function [1], in cryptographic world Cipher is an algorithm which encrypts and decrypts the data but Hash Code is an irreversible conversion data, which makes blockchain network more secure. Because once you have entered the data into blockchain ledger which is the database of blockchain you can't alter or delete the data from it. Blockcahin is used for financial transactions between two parties without the involvement of any third party for example banks. Blockchain was first developed by Satoshi Nakamoto [2]. GraphDB [3] is a type of NoSql database which uses nodes and edges to store its data in the database. Nodes and Edges represents the object and its properties. Relational database [4] is not flexible because once you have created a relational database

* Corresponding Author: Hassanrao.hr@gmail.com

it's unless you changed it by yourself which requires extra efforts and time and it's not a good practice. GraphDB solves this problem because it's easy to add new attributes and relationships in the database. But when we develop GraphDB in blockchain few problems are related with the security of Ledger [5]. First, there was bitcoin [2] and then came Ethereum [6] which is much more programmable than bitcoin. But both of these use their own currency as a medium of exchange but Hyperledger Fabric [7] does not own its currency, since bitcoin and Ethereum are open networks anybody can connect with this network, Hyperledger Fabric is a private network which can only be accessed by the people who are the part of the network. And it is used

for developing blockchain-based applications that can be used within a private organization. The research provides a discussion on the motivation and benefits of the techniques adopted in recent 2017-2022 GraphDB models for Hashing & Indexing, Query Processing, Transaction Management, Data Management and Data Distribution for Blockchain.

II. Background Material

In this section, the first thing get to know the graph database. GraphDB [3] is a type of NoSql database which uses nodes and edges to store its data in the database. Nodes and Edges represent the object and its properties.

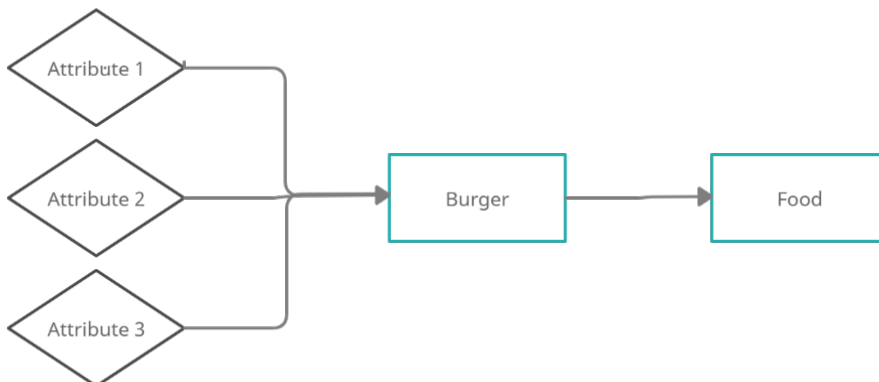


Fig. 1. Nodes

A. GraphDB

The importance of storing the data in GraphDB is explained in the following research [8], [9]. The GraphDB consists of two elements, first edge and second node. Nodes and Edges represents the object and its properties. Many systems are using GraphDB now, twitter is great example of GraphDB [10]. For example, in Figure 1 “burger” and “food” nodes would have the relationship, here’s each node has different attributes. There are different models for GraphDB [11] such as,

AllergoGraph provides special features for analysis on social network. DEX which is based on Java libraries for management of temporary graph. DEX ensures the good performance for vary large scale database. HyperGraphDB, it is useful for modeling the data like Artificial Intelligence and Bio-Informatics. InfiniteGraph, it is useful for analysis in business, social and government intelligence. Neo4j is an open source GraphDB, which uses node, edges and its attributes to store data. Sones is a GraphDB that has its own graph query language.

III. GrpahDB in Blockchain

Blockchain is a peer to peer network that uses digital ledger

system that is maintained by different computer in a network. The goal of this research is to discuss the motivation and benefits of the techniques adopted in recent 2017-2022 GraphDB models for Hashing & Indexing, Query Processing, Transaction Management and Data Management. Study showed [12] that 256 GB of Bitcoin’s data will require $258 \times 0.33 = 86$ GB of storage [13]. Thus, it will result in high performance.

A. Hasing & Indexing

Indexing is way of sorting numbers of records in different fields. And Hashing is used to retrieve them using a shorter hashed key. Hashing & Indexing is used in different contexts in GrpahDB [14]. Due to existence of property values in GraphDB, there are two kinds of graph indexes, structure based and value based. They occur in GraphDB in different forms for example full text querying support. Some search engines like Apache Lucene used in GraphDB as index backend.

In Value-Based Indexing the research presents indexing with three different Graph Database Management Systems (GDBMS). The Cypher language of Neo4j allows indexing for one or multiple for all nodes, but only for nodes

who has given labels [15]. Sparksee uses compressed bitmap indexes and B+Trees to store nodes and edges and with their properties. For faster query processing titan supports two different kinds of indexing mechanism, node-centric and graph indexing. In node-centric indexing, these are local index structure which are built on nodes individually. The graph indexes allows enhanced retrieval of nodes and edges from their properties on selection conditions. Some GDBMS are multi-model which supports graph data model, key value and document for example OrientDB. It uses indexing algorithms like SB-Tree [16]. All of these value based indexes, Neo4j's efficacy is less than OrientDB on some of its nodes. So it is recommended for large number of nodes to use OrientDB [17].

In Structure-Based Indexing the structure-based indexing is used to index and extract structural properties of GraphDB, generally at the time of insertion and in response to a query. Many methods take path for example SPath algorithm [18], which focus on a path-based technique for indexing for local nodes in G in order to transform a query graph into a set of a shortest path for query processing. Srinivasa distinguishes [19] three type of

indexes based on structure, which are path-based, index-based and spectral methods. Spectral methods uses the concept of spectral graph theory. But no Index structure supports all kind of substructure features. Researchers' [20] purposes a Lindex, a kind of graph index which allows indexes to subgraphs contained in GraphDB. Similarly feature-based graph index techniques can be found in [21]. In [18], the researchers introduced two indexing techniques, structure-aware indexing and attribute-aware indexing to process graph matching for property graph.

Index Graph patterns are a pointer based data structure that stores for the reference of graph patterns. Several number of graph patterns are found on different GraphDBs. Index graph have different values for patterns that is based on the kind of data which is stored in GraphDB, and use cases are included in these graph patterns. One of the most popular graph pattern, defined as

$GP = (V_p; E_p)$, where $V = \{v_1; v_2; v_3\}$ and $E = \{(v_1; v_2); (v_2; v_3); (v_3; v_1)\}$, is called a triangle. In Cipher triangle can be described in a few different ways, for example

(n1) - [r1] - (n2) - [r2] - (n3) - [r3] - (n1)

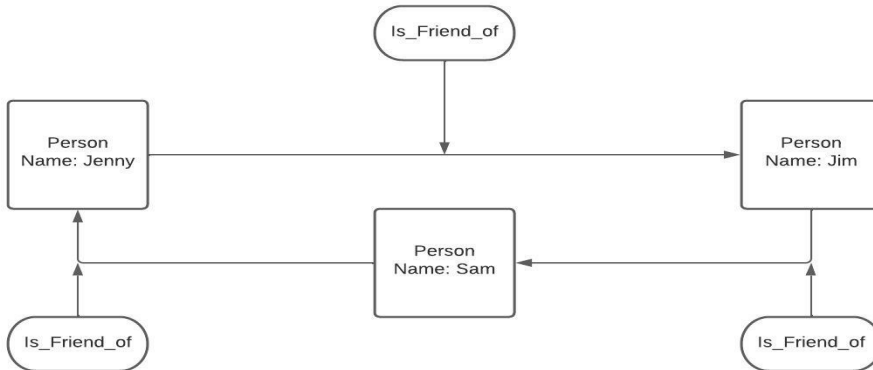


Fig. 2. Ciper triangle

Figure 2 shows a social group. To retrieve that kind of a pattern is easy for Neo4j. But the problem arises when we focus on structural features of the graph and require all the triangle who has friendship features in them. In that kind of case structural-based indexing is more suitable.

B. Query Processing

Neo4j [3] is an open source NoSql language. It stores the data in nodes and edges format. Nodes are the entities in GraphDB that store attributes. For fast query processing results Neo4j offers multiple libraries [22]. With almost zero coding, these algorithms can reveal the hidden patterns in GraphDB. These features makes Neo4j an ideal tool for analyzing and visualizing the data in blockchain ledgers. Moreover, Neo4j offers multiple graph models to support

the user’s requirements. Cypher [23] a Declarative Query Language (Dql) similar to Structured Query Language (Sql), but Dql is optimized for GraphDB. And now this language is used by SAP HANA [24]. RedisGraph [25] and Open-Cypher [26], Cypher Queries are available at Neo4j [27].

The following example of Cypher Query [12], carried out to return all the number of tokens that have been sent on “Address” with transactions. These transactions include the chain extended from a block of hash.

```

MATCH (b:Block{hash:Hash}) -
[:CHILD_OF*0...] -> () - [K] -
(t:Transactions) - [:TO] ->
(u:Users{hex:'Address'})
RETURN SUM(t.amount)
  
```

Unlike the classical structure of the blockchain, these GraphDB

models allows bidirectional pass over of entities in the requested path. With the help of Neo4j, we can analyze and evaluate just by looking at the execution plan of the query. In case of any Casper's [28] event, Casper's usefulness is based on its capacity to detect and penalize bad validators who took advantage of any vulnerability. We reward network nodes to track and report those in our Casper-like protocol. Criminals by providing cash incentives in the event of a well-executed slashing. Furthermore, any evidence in support of a rule any node may detect and recover from a breach since they are all connected.

```
MATCH (v1:Vote), (v2:Vote)
WHERE v1.r_from = v2.r_from
AND v1.target_height =
v2.target_height AND ID(v1) <
ID(v2) RETURN v1.r_from, v1, v2
```

Returns all separate pairs of votes, v1 and v2, submitted by the same validator with targets at the same height.

```
MATCH (v1:Vote), (v2:Vote)
WHERE v1.r_from = v2.r_from
AND v1.target_height >
v2.target_height AND
v1.source_height <
v2.source_height AND NOT
ID(v1) = ID(v2) RETURN
v1.r_from, v1, v2
```

Returns all separate pairs of votes; v1 and v2, sent by the same validator, where one vote is inside the span of the other.

The above queries, are not for a specific branch, these are for the whole blockchain tree.

C. Transaction Management

Cypher is a query language that based on patterns, and cypher is specially designed to recognize these patterns. Common keywords in cypher are, MATCH, WHERE and RETURN. These keywords are similar to Sql but with a slight difference, although they are very similar. There are 4 properties of Transaction Management by using Neo4j GraphDB. ACID are described below,

1. Atomicity: in case of any transaction failure, data is left unchanged.
2. Consistency: all the transaction will leave the database in a consistent state.
3. Isolation: similar to locking technique, during the transaction data cannot be accessed by any other operations.
4. Durability: The database management system can recover the results of a committed transaction.

Transaction Management in Neo4j [29], code in java is available at Github [30]. According to Neo4j, there are 7 steps to manage a transaction in GraphDB.

1. Interaction cycle ensures that it fulfil ACID properties or not, for example begin a transaction, performing a database operation and commit or roll back in transaction.
2. Isolation levels, it means that if the transaction is under process it is not available for other processes until it's completely committed.
3. Default Locking behavior. It is similar to relational database.
4. Deadlock. Neo4j can detect any before they even happen and throw an exception, Deadlock code is available at Github [31].
5. Delete Semantics, Neo4j provide semantics that every data node should have a start node and end note. It means that if we try to delete a node that still has a relationship with another node, it will throw an exception.
6. Creating unique Nodes, certain level of uniqueness is required in nodes. To ensure this there are two techniques to ensure the techniques, single thread and get or create.
7. Transaction Event, it notified in case of any transaction event in GraphDB through a Transaction Event Listener [32].

D. Data Management

Decentralized nature of blockchain allows data to be accessed within the organization easily. Following figure describes how data is managed in GraphDB.

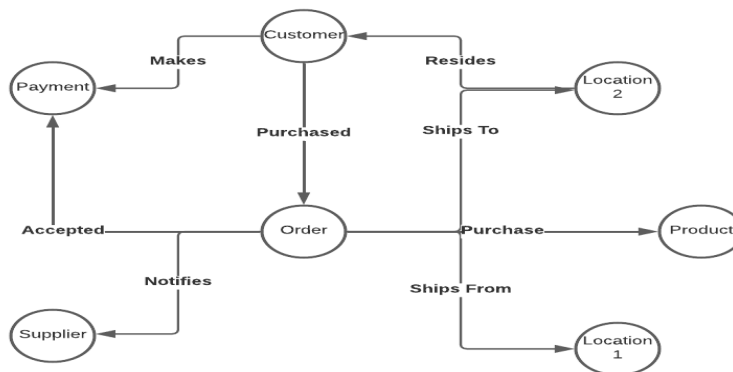


Fig. 3. Data management

Benefits of Blockchain for Data Management

1. **Data Security:** blockchain uses hash values which is irreversible conversion of data, it provide maximum level of security as compared to any encryption algorithms.
2. **Data Quality:** data is stored in edges and if any data is entered into the ledger, it is analyzed and cross-checked before it stored in the database.
3. **Data Traceability:** data can be traced through tokens and as blockchain is linear, an historical chain of events can be followed easily.
4. **Real-Time Data Analysis:** due to blockchain nature, businesses

can easily notice any kind of irregularities as they occur and they can be viewed.

5. **Data Sharing:** blockchain nature allows data to be shared within the organization easily and permissions can be set to whom can read or edit the data.

Following are the studies and tools to build and manage the data in ledger [33]-[36].

E. Data Distribution

According to Jim Webber chief scientist of Neo4j states that, Neo4j is going distributed in GraphDB [37]. For scalability Neo4j has a limit. With the launch of 4.0 Neo4j is going to be distributed GraphDB.

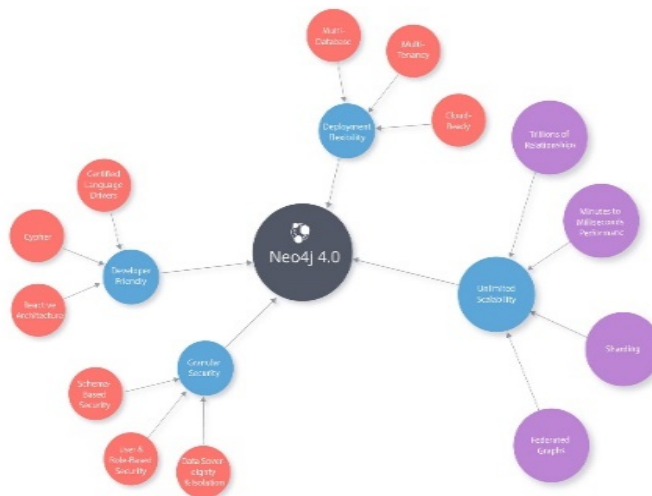


Fig. 4. Data distribution [38]

With Neo4j, Webber says the customer has to decide how to split-up or share their data into a distributed environment, or sub-GraphDB. But in a Fabric server, they can be queried as a single entity. Neo4j is following MongoDB [39] model for this. In this model, customers can split their data as per their business function and geographical region requirement.

IV. Conclusion

The study highlighted the features of GraphDB in financial technologies. GraphDB stores data in nodes and edges. Nodes stores data entities and edges stores relationship between entities. The study has provided Hashing and Indexing techniques for GrpahDB in blockchain. There are two types of Hashing, the first is value-based which uses Cypher query language, and the second structure based which uses SPath algorithm. The study then presented how queries are performed in GraphDB blockchain. For query processing, Neo4j has provided multiple libraries, unlike traditional databases GrpahDB allows bidirectional passes between entities. In the transaction management section, the cypher query language is used to recognize patterns, Java code is available for

this. Data Management section describes how data is stored and managed in database. This study also provided solution in how to build and manage the data in GrpahDB blockchain.

References

- [1] H. Stevens, "Hans Peter Luhn and the birth of the hashing algorithm," *IEEE Spectr.*, vol. 55, no. 2, pp. 44-49, Feb. 2018, doi: <https://doi.org/10.1109/MSPE.C.2018.8278136>
- [2] Satoshi Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Decen. Bus. Rev.*, Art. no. 21260," 2008.
- [3] Neo4j, "The graph data platform for today's intelligent applications," Neo4j. <https://neo4j.com/> (accessed Jan. 23, 2022).
- [4] M. A. Rodriguez and P. Neubauer, "Constructions from dots and lines," *Bull. Am. Soc. Inf. Sci. Technol.*, vol. 36, no. 6, pp. 35-41, 2010, doi: <https://doi.org/10.1002/bult.2010.1720360610>
- [5] M. Shkoukani and A. M. Altamimi, "Graph Database Security: Blockchain Solution and Open Challenges," *Int. J. Simul.-Sys. Sci. Technol.*, vol. 21, no.1, pp. 1-7, 2020, doi:

- <https://doi.org/10.5013/IJSSST.a.21.01.09>
- [6] V. Buterin, "Ethereum white paper," *GitHub Repos.*, no. 1, pp. 22-23, 2013.
- [7] AWS, "What is hyperledger fabric?," AWS. <https://www.ibm.com/topics/hyperledger> (accessed Jan. 23, 2022).
- [8] S. Medhi and H. K. Baruah, "Relational database and graph database: A comparative analysis," *J. Process Manag. New Technol.*, vol. 5, no. 2, pp. 1-9, 2017, doi: <https://doi.org/10.5937/joupro-man5-13553>
- [9] A. Martínez Porras, R. A. Mora Rodríguez, D. Alvarado González, G. López Herrera, and S. Quirós Barrantes, "A Comparison between a relational database and a graph database in the context of a Personalized Cancer Treatment Application," *Alberto Mendelzon Int. Workshop Found. Data Manag.*, vol. 1644, 2016.
- [10] H. Huang and Z. Dong, "Research on architecture and query performance based on distributed graph database Neo4j," in *2013 3rd Int. Conf. Consumer Elec. Commun. Net.*, Nov. 2013, pp. 533-536.
- [11] R. Angles, "A comparison of current graph database models," in *2012 IEEE 28th Int. Conf. Data Eng. Workshops*, pp. 171-177, April 1-5, 2012, doi: <https://doi.org/10.1109/ICDEW.2012.31>
- [12] Y. Tang et al., "Graph database-based knowledge graph storage and query for power equipment management," in *2020 12th IEEE PES Asia-Pacific Power Energy Eng. Conf. (APPEEC)*, Sep. 2020, pp. 1-5. doi: <https://doi.org/10.1109/APPEEC48164.2020.9220382>
- [13] Blockchain, "Blocks-size," *Blockchain.com*. <https://www.blockchain.com/e/charts/blocks-size> (accessed Feb. 13, 2022).
- [14] J. Pokorný, M. Valenta, and J. Ramba, "Graph Patterns Indexes: their Storage and Retrieval," in *Proc. 20th Int. Conf. Inf. Integ. Web-based Appl. Serv.*, New York, USA, Nov. 2018, pp. 221-225. doi: <https://doi.org/10.1145/3282373.3282374>
- [15] J. Pokorný and M. Troup, "Indexing Patterns in Graph

- Databases," in *Proc. 7th Int. Conf. Data Sci. Technol. Appl.*, 2018, pp. 313-321, doi: <https://doi.org/10.5220/0006826903130321>
- [16] P. E. O’Neil, “TheSB-tree an index-sequential structure for high-performance sequential access,” *Acta Inform.*, vol. 29, no. 3, pp. 241–265, Mar. 1992, doi: <https://doi.org/10.1007/BF01185680>
- [17] S. A. T. Mpinda, L. C. Ferreira, M. X. Ribeiro, and M. T. P. Santos, “Evaluation of graph databases performance through indexing techniques,” *Int. J. Artif. Intell. Appl.*, vol. 6, no. 5, pp. 87–98, Sep. 2015, doi: <https://doi.org/10.5121/ijaia.2015.6506>
- [18] D. Yuan and P. Mitra, "Lindex: A lattice-based index for graph databases," *VLDB J.* vol. 22, pp. 229–252, 2013, doi: <https://doi.org/10.1007/s00778-012-0284-8>
- [19] S. Srinivasa, "Data, storage and index models for graph databases," in *Graph data management: techniques and applications*, S. Sakr, E. Pardede, Eds., India, IGI Global, 2012, pp. 47-70, doi: <https://doi.org/10.4018/978-1-61350-053-8.ch003>
- [20] X. Yan and J. Han, "Graph Indexing," in *Managing and mining graph data. Advances in database systems*, C. Aggarwal and H. Wang, Eds., Boston, MA, Springer, 2010, doi: https://doi.org/10.1007/978-1-4419-6045-0_5
- [21] H. Dinari, “A Survey on Graph Queries Processing: Techniques and Methods,” *Int. J. Comput. Netw. Inf. Secur.*, vol. 9, no. 4, pp. 48–56, Apr. 2017, doi: <https://doi.org/10.5815/ijcnis.2017.04.06>
- [22] M. Needham and A. E. Hodler, *Graph Algorithms: Practical Examples in Apache Spark and Neo4j*. USA: O’Reilly Media, 2019.
- [23] O. Panzarino, *Learning Cypher*. Birmingham, UK: Packt Publishing Ltd, 2014.
- [24] SAP HANA loud, "In-Memory Database,” *SAP*. <https://www.sap.com/products/hana.html> (accessed Feb. 13, 2022).
- [25] Redis, “Fast graph processing powered by linear algebra and matrix multiplication,” redis.com.

- <https://redis.com/modules/redis-graph/> (accessed Feb. 13, 2022).
- [26] OpenCypher, "What is openCypher?," Opersypher.org. <http://opencypher.org/> (accessed Feb. 13, 2022).
- [27] G. Walker, "How to import the bitcoin blockchain into Neo4j [Community Post]," *Neo4j Graph Data Platform*. <https://neo4j.com/blog/import-bitcoin-blockchain-neo4j/> (accessed Feb. 13, 2022).
- [28] V. Buterin and V. Griffith, "Casper the Friendly Finality Gadget," *ArXiv171009437 Cs*, Jan. 2019. [Online]. Available: <http://arxiv.org/abs/1710.09437>
- [29] Neo4j, "Transaction management - Java Reference," *Neo4j Graph Database Platform*. <https://neo4j.com/docs/java-reference/4.4/transaction-management/> (accessed Feb. 13, 2022).
- [30] R. H. Kaleem, "RaoHassanKaleem/Transaction-Management-of-GraphDB-in-Blockchain-Java-Code," Github. <https://github.com/RaoHassanKaleem/Transaction-Management-of-GraphDB-in-Blockchain-Java-Code> (accessed Feb. 13, 2022).
- [31] Neo4j, "About Neo4j documentation," Neo4j. <https://github.com/neo4j/neo4j-documentation/blob/68a550e7890a9f5328ff2ec4191cb764c57b0500/kernel/src/test/java/examples/DeadlockDocTest.java> (accessed Feb. 14, 2022).
- [32] Neo4j, "Interface TransactionEventListener<T>," Neo4j. <https://neo4j.com/docs/java-reference/4.4/javadocs/org/neo4j/graphdb/event/TransactionEventListener.html> (accessed Feb. 14, 2022).
- [33] MongoDB, "Blockchains, ledgers, and databases: A guide to navigate the confusion," *MongoDB*. <https://www.mongodb.com/collateral/blockchains-ledgers-and-databases-a-guide-to-navigate-the-confusion> (accessed Feb. 14, 2022).
- [34] R. Wattenhofer, *The science of the blockchain*, Inverted Forest Publishing, 2016.
- [35] IDC, "Blockchain — A data management, integration, and integrity disruptor?," *IDC: The premier global market*

- intelligence company*.
<https://www.idc.com/getdoc.jsp?containerId=US42074217>
(accessed Feb. 14, 2022).
- [36] DLACM, “Graph Chain – A distributed database with explicit semantics and chained rdf graphs,” DLACM Digital Library.
<https://dl.acm.org/doi/fullHtml/10.1145/3184558.3191554>
- [37] Neo4j, “Neo4j going distributed with graph database,” *Neo4j Graph Data Platform*.
<https://neo4j.com/news/neo4j-going-distributed-with-graph-database/> (accessed Feb. 20, 2022).
- [38] Datanami, “Neo4j going distributed with graph database,” *Datanami*,
<https://www.datanami.com/2020/02/04/neo4j-going-distributed-with-graph-database/> (accessed Feb. 20, 2022).
- [39] MongoDB, “MongoDB: The application data platform,” *MongoDB*.
<https://www.mongodb.com>
(accessed Feb. 20, 2022).