A COMPARATIVE STUDY ON BLOCKCHAIN DATA MANAGEMENT SYSTEMS: BIGCHAINDB VS FALCONDB

Abrar Alotaibi[12], Sarah Alissa[12] and Salahadin Mohammed [1]

[1] Department of Information and Computer Science, King Fahd University of Petroleum and Minerals
Dhahran, Saudi Arabia
[2] Computer Science department, College of Computer Science and Information Technology,
Imam Abdulrahman bin Faisal University
Dammam, Saudi Arabia

Abstract
The widespread usage of blockchain technology in cryptocurrencies has led to the adoption of the blockchain concept in data storage management systems for secure and effective data storage and management. Several innovative studies have proposed solutions that integrate blockchain with distributed databases. In this article, we review current blockchain databases, then focus on two well-known blockchain databases—BigchainDB and FalconDB—to illustrate their architecture and design aspects in more detail. BigchainDB is a distributed database that integrates blockchain properties to enhance immutability and decentralization as well as a high transaction rate, low latency, and accurate queries. Its architecture consists of three layers: the transaction layer, consensus layer, and data model layer. FalconDB, on the other hand, is a shared database that allows multiple clients to collaborate on the database securely and efficiently, even if they have limited resources. It has two layers: the authentication layer and the consensus layer, which are used with client requests and results. Finally, a comparison is made between the two blockchain databases, revealing that they share some characteristics such as immutability, low latency, permission, horizontal scalability, decentralization, and the same consensus protocol. However, they vary in terms of database type, concurrency mechanism, replication model, cost, and the usage of smart contracts.

Keywords:
Blockchain Data Management, BigchainDB, FalconDB, Distributed Databases, Decentralization.

1. Introduction

A blockchain system's efficiency is determined not only by its cryptographic security, which relies on asymmetric cryptography and distributed consensus mechanisms but also by its capacity to deliver effective data management solutions. Blockchain technology is defined by its decentralization, persistency, anonymity, and auditability. Blockchain systems can be classified into three types based on their chain structures: standard blockchain, hybrid blockchain, and DAG-based blockchain [1]. Standard blockchain systems, such as Bitcoin [2], store all confirmed transactions in an ever-expanding list of blocks, known as a chain. Any device can participate in the blockchain network as a node and maintain a full copy of the chain. On the other hand, hybrid blockchain systems like HyperLedger [3] operate with multiple chains in the network, with each node having access to specific chain data only. These individual chains work together to share information and enable transactions across chains. DAG-based blockchain systems, such as DagCoin [4], use a Directed-Acyclic-Graph (DAG) as the primary data structure for handling transactions, replacing the traditional block list. In DAG structures, each node represents a transaction and is linked to at least two other nodes in the DAG, based on node selection algorithms [5].

Despite their many benefits and potential uses, blockchain systems encounter several issues that must be resolved to ensure their widespread adoption and efficiency. One of the main challenges blockchain systems face is scalability. Scalability is a major issue for blockchain systems, as factors like low throughput, high data load, and inefficient query engines significantly restrict their deployment [6].

Innovative solutions such as BigchainDB [7] and FalconDB [8] have been developed to address the scalability challenges faced by standard blockchain technology. BigchainDB is a distributed database that combines the benefits of traditional databases with the features of blockchain technology, offering high transaction throughput, low latency, and powerful query capabilities. It maintains the decentralized, immutable, and cryptographic aspects of a blockchain while providing a scalable and efficient data management solution. Similarly, FalconDB is another emerging technology that strives to enhance the scalability and performance of blockchain systems. By incorporating advanced data structures and storage mechanisms, FalconDB aims to optimize transaction throughput and data access while maintaining the core

Manuscript received May 5, 2023
Manuscript revised May 20, 2023
https://doi.org/10.22937/IJCSNS.2023.23.5.15

principles of decentralization and security found in blockchain technology. Both BigchainDB and FalconDB represent promising approaches to overcoming the inherent limitations of traditional blockchain systems, paving the way for more efficient and scalable blockchain-based applications.

Blockchain data management has emerged as an area of research that addresses the security and scalability issues, exploring aspects such as blockchain architecture, data structure, and storage engine. Considering these advancements in blockchain data management, our research has been designed to offer a thorough insight into this domain. This work is organized as follows. Section 2 offers a literature review of related works. Sections 3 and 4 respectively examine BigchainDB architecture and design as well as the design aspects of FalconDB. Section 5 presents a comparison between BigchainDB and FalconDB. Finally, Section 6 includes the conclusion from this work.

2. Review of Related Work

The blockchain data management system can be broken down into four components: blockchain-assisted databases, architecture, data structure, and storage engine. Blockchain architecture pertains to the overall organization of the network, with three primary designs: standard, hybrid, and DAG-based blockchains, each with distinct ledger maintenance structures [9]. Blockchain data structures dictate how transactions are organized within the network, either through mining processes (e.g., Proof-of-Work or Proof-of-Stake) in standard and hybrid blockchains or via direct attachment as individual nodes in DAG-based systems. The storage engine manages and stores blockchain data on nodes, often utilizing a combination of databases and file systems to maintain data integrity [1].

Various research has proposed solution to the enhance data management of blockchain, and multiple data management system has been implemented [5], [10]–[15]. The researchers aim to find the optimal data management system for blockchain. This section presents some of the relevant works.

BlockchainDB was presented by El-Hindi et al. [10], employing blockchains as a storage layer while adding a database layer on top to incorporate traditional data management techniques, such as sharding. It offers a standardized key/value-based query interface, simplifying blockchain adoption for data sharing. This not only enhances blockchain performance and scalability for data sharing but also reduces implementation complexity. However, the added database layer may increase storage resource consumption.

Block-secure was introduced by Li et al. [11] as a blockchain-based security architecture for distributed cloud storage. Users can split their files into encrypted data chunks and randomly upload them to P2P network nodes with available storage capacity. A customized genetic algorithm was used to allocate file block replicas between users and datacentres in the distributed cloud storage environment. However, this method is only suitable for static data environments and increases data dispersion, making queries more difficult.

Chen et al. [12] proposed an enhanced P2P file database using IPFS blockchain and content service providers to address low-throughput issues. They introduce a zigzag-based storage model, using Bitcoin as the underlying blockchain. However, once the service provider node completes verification, the blockchain system's centralization and immutability are compromised, affecting overall security.

Blockstack was developed by AliNSF et al. [13], a blockchain-based naming and storage system that separates control and data planes, storing minimal metadata on the blockchain and using external data stores for bulk storage.

Do et al. [14] presented a blockchain-based system for secure data storage with keyword search capabilities. The system enables clients to upload encrypted data, distributes content to cloud nodes, and uses cryptographic techniques to ensure data availability. Data owners can grant search permissions, and the system supports private keyword searches over encrypted datasets.

Wang et al. [15] proposed a decentralized storage and sharing framework combining an interplanetary file system, Ethereum blockchain, and attribute-based encryption. Smart contracts on Ethereum implement keyword search on ciphertext, addressing issues with traditional cloud storage systems. Although this approach is novel, it consumes considerable storage resources, impacting system scalability.

ChainifyDB was proposed by F. M. Schuhknecht et al.[16] which is a blockchain solution that is implemented as a blockchain layer added on top of the standard database layer and connected within a network. This helps developers create decentralized blockchain applications and keep their existing database system as an underlying layer. However, this may raise an issue related to the heterogeneous infrastructures of the participant so they may interpret a particular transaction differently.

L. Allen et al.[17] introduced Veritas which is a verifiable database that is implemented on top of the blockchain and it uses shared tables. The database and tables use the regular features of relational databases and tables. It provides verification mechanisms that allow the users to check for the accuracy of query responses and results and it is consistent with the database content, the

only verifiable property is the immutable logs. Nevertheless, this approach focuses on cloud infrastructure [16].

S. Nathan et al. [18] designed the Blockchain Relational Database, which leverages the feature of existing relational databases (e.g. PostgreSQL) and implements a decentralized system with database replicas. It allows different users that do not have trust in each other to manage transactions like a new company. This approach is still a prototype and needs improvements and optimizations to support scalability [19].

3. BigchainDB

BigchainDB is a blockchain database that merges the advantages of distributed databases and blockchain technology. McConaghy et al. [7] introduced it in their paper titled "BigchainDB: A Scalable Blockchain Database". It delivers decentralization, immutability, and inherent support for cryptographic assets while maintaining the high throughput and low latency typical of distributed databases.

BigchainDB features a three-tier architecture shown in Figure 1, comprising the transaction layer, consensus layer, and data model layer:

1) Transaction layer: This layer is responsible for the generation, signing, and validation of transactions. BigchainDB transactions are formatted as JSON documents containing inputs, outputs, and metadata. Inputs and outputs specify the conditions for asset ownership and transfer, while metadata includes supplementary information related to the transaction. Transactions are cryptographically signed by the sender to ensure authenticity and prevent tampering.
2) Consensus layer: BigchainDB achieves consensus through a voting-based Byzantine fault tolerance algorithm known as Federated Byzantine Agreement (FBA). In this method, each network node has a group of other nodes, called its quorum slice, to which it sends its vote. Nodes reach consensus when a predefined threshold of nodes shares the same vote. The FBA consensus guarantees transaction consistency and finality within the network.
3) Data model layer: BigchainDB employs a distributed database called RethinkDB to store its data. RethinkDB is a NoSQL database that offers horizontal scalability, high availability, and real-time synchronization features. In BigchainDB, the data model is composed of blocks, transactions, and assets. Blocks serve as transaction containers, and each block possesses a unique ID, a timestamp, and a list of transaction IDs. Assets represent digital or physical items tracked by the blockchain and are created and managed via transactions.

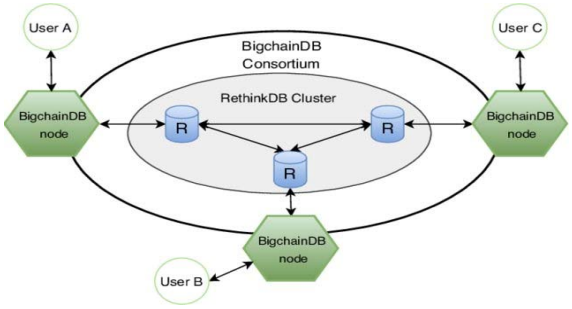

Figure 1 BigchainDB Architecture [20]

BigchainDB is a distinctive blockchain solution that merges the benefits of distributed databases and blockchain technology, resulting in a highly scalable and versatile system. It exerts high scalability due to its utilization of RethinkDB a distributed database to deliver high throughput and low latency, allowing for horizontal scaling and support for millions of transactions per second. Also, it supports complex queries on assets, metadata, and transaction history, facilitating advanced data retrieval and analysis.

Transactions in BigchainDB are cryptographically signed and stored in blocks, guaranteeing that the records are tamper-resistant and traceable, a crucial aspect of blockchain systems. By employing the FBA consensus algorithm, BigchainDB ensures decentralization. It accommodates various deployment models, including public, private, and consortium blockchains, making it adaptable for diverse use cases and organizations. BigchainDB natively supports cryptographic assets, enabling secure asset creation, transfer, and management through transactions.

Nevertheless, BigchainDB has limitations, one such is the limited decentralization compared to public blockchains, its FBA consensus algorithm is more centralized than public blockchain consensus algorithms like Proof-of-Work (PoW) or Proof-of-Stake (PoS). This makes BigchainDB better suited for consortium or private blockchains rather than fully decentralized public networks.

Another drawback of BigchainDB is the potential single point of failure, while BigchainDB relies on a distributed database for scalability, the underlying database (RethinkDB) can still become a single point of failure if not configured properly or distributed across multiple nodes. BigchainDB's performance and features are closely tied to the underlying database (RethinkDB). If the database encounters issues or limitations, it may impact the overall functionality and performance of BigchainDB.

A Security concern of BigchainDB is the FBA consensus algorithm offers a different security model compared to PoW or PoS, and its resilience against adversarial attacks may vary. Additionally, since BigchainDB is relatively new,

it may not have undergone the same level of thorough testing as more established blockchain systems.

4. FalconDB

FalconDB is a blockchain-based collaborative database, that was introduced by Y. Peng et al. [8] in their paper titled "FalconDB: Blockchain-based Collaborative Database". It allows different clients to access the shared database securely and efficiently even when they have minimal hardware resources. FalconDB leverages database servers with client-accessible verification interfaces and maintain the digests for query/update authentications on a blockchain.

FalconDB is composed of two distinct types of entities: a set of servers that maintain the database's content and a set of clients that can query and modify the database without saving it, which is presented in Figure 2.

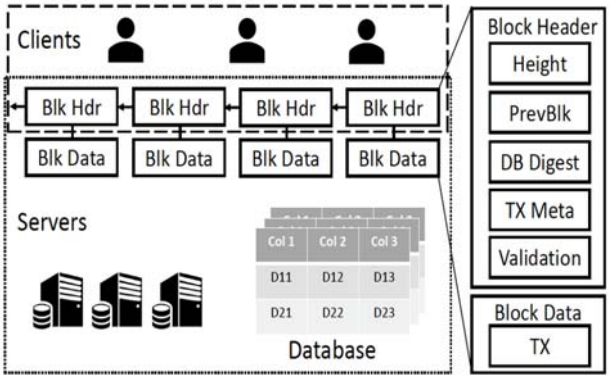

Figure 2: FalconDB System Overview[8]

- Server nodes: are responsible for storing the system database and the whole blockchain data. In addition to responding to client queries and updates, validating newly added blockchain blocks, and producing proofs for query results. These server nodes are hosted on several entities, and they will receive financial rewards for their services. Each server's database is stored in a persistent structure that keeps track of all previous versions. Additionally, an Authenticated data structure (ADS) is built from the database content, which enables the clients to receive authenticated queries and updates.
- Client nodes: are the nodes that can access the database collaboratively. Based on their privileges, they can read, and write part of the database, however, they cannot maintain the database content locally on their machines. To gain access to the database, a client node must send a request to one of the server nodes, then they can utilize the Authenticated data structures (ADS) for authenticating the results of query or update requests. Not all the clients' nodes can connect to the blockchain network to validate a block, it depends on the configuration some clients can only passively pull the most recent blocks without participating in the blockchain consensus.

The servers and the clients' nodes collaboratively maintain the blockchain network. Every block in the database contains only one transaction that involves one or more reads or writes instructions to the database. The blockchain transaction in FalconDB is similar to the database transaction. The content of the block header is an ADS digest from the same version of the database used when the transaction in the block has been executed and the block is committed. Therefore, the client users can use these digests in verifying whether their requests were executed successfully.

Before joining consensus, all the server nodes and client nodes have to make a deposit to a smart contract provided by Ethereum. Using this smart contract, a privilege function could be implemented and agreed on by all the nodes within a network. The function specifies dynamically for each client, the type of update operations they can perform. When a client became suspicious or compromised, the other nodes may withdraw all of its access rights to the database.

The underlying architecture of FalconDB is separated into two layers. The first is the authentication layer and the second is the consensus layer [21].

1) Consensus layer: Let's consider the client sent a query to the server node, when the server finish executing the client request, it will send the result immediately. Based on the consensus protocol agreed on by all blockchain nodes, the new block should be committed in the blockchain. Then each client can pull the latest digest.
2) Authentication layer: Through this layer, the client can send an authentication request to the server node to validate the query execution as illustrated in Figure 3.

FalconDB was implemented with the aim to guarantees the following security properties and performance properties:

The Security properties are:

- Immutability: modification on the database that is committed to a blockchain is immutable, which means that, according to our threat model, it cannot be altered or denied/discredited.

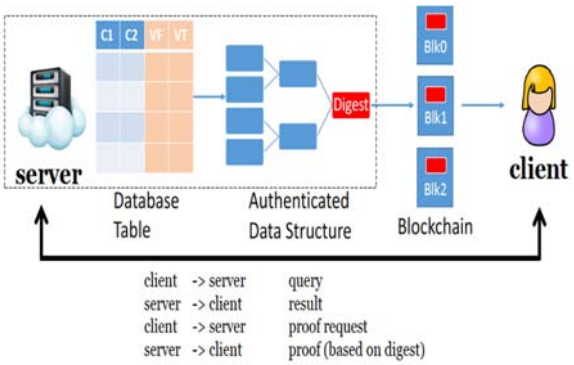

Figure 3: Simplified query workflow of FalconDB [8]

- Transparency: All the operations done on the database are transparent to every node in the network. So, the users can retrieve for any record all its previous versions along with the metadata of previous updates such as the update time and the user who perform it.
- Data integrity: It allows the client to ensure that the database content is changed based on the committed transaction and unauthorized updates on the database are prohibited.
- Query correctness: The client can verify the correctness of a query result sent from the server by checking that it is equivalent with the latest database version and that nothing is modified or omitted.

The Performance properties are:

- Highly expressive: the FalconDB supports several types of queries, including lookup, range, and aggregation queries.
- Low cost: FalconDB aimed to reduce the cost on the client, so any client who has few resources still can collaborate in the database.
- Performance: FalconDB intends to make the database transaction offered with low latency and high throughput although it has a complex consensus and authentication in comparison to a simple external database.

5. BigchainDB Comparison Between BigchainDB and FalconDB

A comparative analysis between BigchainDB and FalconDB is summarized in Table 1. The resultant data was extracted from the published papers of BigchainDB [7], FalconDB [8], and further supplemented with data collected from the work presented in [22], [21], [19], and [23]. The table illustrates the commonalities and differences shared by these two databases.

Both BigchainDB and FalconDB exhibit key attributes such as immutability, low latency, permission-based access, horizontal scalability, decentralization, and utilize the Byzantine Fault Tolerance (BFT) consensus protocol. However, notable differences are observed in their approach toward supporting blockchain technology, their choice of database type, concurrency mechanism, replication model, and in the implementation of smart contracts.

Table 1: Comparison Between BigchainDB and FalconDB

Properties	BigchainDB	FalconDB
Immutability	Yes	Yes
Low Latency	Yes	Yes
Decentralized	Yes	Yes
Permission	Yes	Yes
Scalability	Support Horizontal scaling	Support Horizontal scaling
Consensus protocol	Byzantine Fault Tolerance (BFT)	Byzantine Fault Tolerance (BFT)
Blockchain technique	Blockchain characteristics on top of distributed database	Ethereum as its blockchain layer
Database type	Document-oriented database	Traditional DB
Concurrency	Blockchain Pipelining	Optimistic Concurrency Control (OCC)
Replication Model	Transaction-based replication	Storage-based replication
Throughput (TPS)	<100	2k
Cost	High	Low
Smart contract	Not supported	Supported

BigchainDB amalgamates blockchain with a distributed database, whereas FalconDB incorporates Ethereum as its blockchain layer. As for the type of database, BigchainDB employs a document-oriented database, such as RethinkDB or MongoDB, in contrast to FalconDB, which opts for traditional relational databases like MySQL. In terms of the concurrency mechanism, BigchainDB leverages blockchain pipelining, while FalconDB employs optimistic concurrency control. Both databases endorse replication; however, BigchainDB adopts transaction-based replication, whereas FalconDB implements storage-based replication.

A crucial difference lies in throughput and cost. FalconDB exhibits a higher throughput compared to BigchainDB, making it capable of processing a larger number of transactions per unit of time. In contrast, FalconDB is more

cost-effective than BigchainDB in terms of resource requirements. Lastly, it is noteworthy that only FalconDB incorporates the use of smart contracts within its system, a feature absent in BigchainDB.

6. Conclusions

The domain of blockchain data management systems has swiftly expanded, prompting a growing research frontier that captivates both academia and industry. Numerous studies have proposed innovations aimed at improving the data management and storage aspects of blockchain technology. In this context, this work undertakes a review of contemporary blockchain data management systems, focusing specifically on two prominent blockchain databases: BigchainDB and FalconDB.

BigchainDB combines the distinctive traits of blockchain technology with those of a decentralized distributed database. Consequently, it offers a range of benefits, including immutability, scalability, high throughput, and low latency. Alternatively, FalconDB assimilates a public platform as its blockchain layer and superimposes a shared traditional database. This architectural configuration facilitates secure and efficient access to the shared database for clients with limited resources.

A comparative analysis of the two aforementioned blockchain databases revealed both commonalities and differences. Both BigchainDB and FalconDB support key attributes such as immutability, low latency, permission, horizontal scalability, decentralization, and the Byzantine Fault Tolerance (BFT) consensus protocol. However, discrepancies are evident, primarily in the aspect of throughput, where FalconDB (2k) exhibits superiority over BigchainDB (<100). High throughput in a database system allows for the processing of a larger number of transactions or requests per unit of time, leading to improved overall system performance, better resource utilization, and enhanced user experience due to faster response times. Other variations pertain to the type of database, concurrency mechanism, replication model, cost, and the utilization of smart contracts.

Acknowledgment

The authors would like to thank King Fahd University of Petroleum and Minerals and Imam Abdulrahman Bin Faisal University, Saudi Arabia, for supporting this work.

References

[1] X. Fan, B. Niu, and Z. Liu, “Scalable blockchain storage systems: research progress and models,” *Computing*, vol. 104, no. 6, pp. 1497–1524, 2022, doi: 10.1007/s00607-022-01063-8.

[2] S. Nakamoto, “Bitcoin: A Peer-to-Peer Electronic Cash System,” *Cryptography Mailing list at https://metzdowd.com*, Mar. 2009.

[3] C. Cachin and others, “Architecture of the hyperledger blockchain fabric,” in *Workshop on distributed cryptocurrencies and consensus ledgers*, 2016, vol. 310, no. 4, pp. 1–4.

[4] S. D. Lerner, “DagCoin: a cryptocurrency without blocks.” 2015.

[5] Q. Wei, B. Li, W. Chang, Z. Jia, Z. Shen, and Z. Shao, “A Survey of Blockchain Data Management Systems,” *ACM Trans. Embed. Comput. Syst.*, vol. 21, no. 3, May 2022, doi: 10.1145/3502741.

[6] I.-C. Lin and T.-C. Liao, “A survey of blockchain security issues and challenges.,” *Int. J. Netw. Secur.*, vol. 19, no. 5, pp. 653–659, 2017.

[7] T. McConaghy *et al.*, “Bigchaindb: a scalable blockchain database,” *white paper, BigChainDB*, 2016.

[8] Y. Peng, M. Du, F. Li, R. Cheng, and D. Song, “FalconDB: Blockchain-based Collaborative Database,” *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 637–652, 2020, doi: 10.1145/3318464.3380594.

[9] H. Huang, W. Kong, S. Zhou, Z. Zheng, and S. Guo, “A Survey of State-of-the-Art on Blockchains: Theories, Modelings, and Tools,” *ArXiv*, vol. abs/2007.0, 2020.

[10] M. El-Hindi, C. Binnig, A. Arasu, D. Kossmann, and R. Ramamurthy, “BlockchainDB: a shared database on blockchains,” *Proceedings of the VLDB Endowment*, vol. 12, pp. 1597–1609, Jul. 2019, doi: 10.14778/3342263.3342636.

[11] J. Li, J. Wu, and L. Chen, “Block-secure: Blockchain based scheme for secure P2P cloud storage,” *Information Sciences*, vol. 465, pp. 219–231, 2018, doi: https://doi.org/10.1016/j.ins.2018.06.071.

[12] Y. Chen, H. Li, K. Li, and J. Zhang, “An improved P2P file system scheme based on IPFS and Blockchain,” in *2017 IEEE International Conference on Big Data (Big Data)*, 2017, pp. 2652–2657. doi: 10.1109/BigData.2017.8258226.

[13] M. Ali, J. Nelson, R. Shea, and M. J. Freedman, “Blockstack: A Global Naming and Storage System Secured by Blockchains,” in *Proceedings of the 2016 USENIX Conference on Usenix Annual Technical Conference*, 2016, pp. 181–194.

[14] H. G. Do and W. K. Ng, “Blockchain-Based System for Secure Data Storage with Private Keyword Search,” in *2017 IEEE World Congress on Services (SERVICES)*, 2017, pp. 90–93. doi: 10.1109/SERVICES.2017.23.

[15] S. Wang, Y. Zhang, and Y. Zhang, "A Blockchain-Based Framework for Data Sharing With Fine-Grained Access Control in Decentralized Storage Systems," *IEEE Access*, vol. PP, p. 1, Jun. 2018, doi: 10.1109/ACCESS.2018.2851611.

[16] F. M. Schuhknecht, A. Sharma, J. Dittrich, and D. Agrawal, "ChainifyDB: How to Blockchainify any Data Management System," pp. 1–31, 2019, [Online]. Available: http://arxiv.org/abs/1912.04820

[17] L. Allen *et al.*, "Veritas: Shared verifiable databases and tables in the cloud," *CIDR 2019 - 9th Biennial Conference on Innovative Data Systems Research*, 2019.

[18] S. Nathan, C. Govindarajan, A. Saraf, M. Sethi, and P. Jayachandran, "Blockchain meets database: Design and implementation of a blockchain relational database," *Proceedings of the VLDB Endowment*, vol. 12, no. 11, pp. 1539–1552, 2018, doi: 10.14778/3342263.3342632.

[19] J. Kalajdjieski, M. Raikwar, N. Arsov, G. Velinov, and D. Gligoroski, "Databases fit for blockchain technology: A complete overview," *Blockchain: Research and Applications*, p. 100116, 2022, doi: 10.1016/j.bcra.2022.100116.

[20] A. Kalogeropoulos, "A Reference Architecture for Blockchain-based Resource-intensive Computations managed by Smart Contracts," 2018.

[21] D. L. Fekete and A. Kiss, "A survey of ledger technology-based databases," *Future Internet*, vol. 13, no. 8, 2021, doi: 10.3390/fi13080197.

[22] S. Lupaiescu, P. Cioata, C. E. Turcu, O. Gherman, C. O. Turcu, and G. Paslaru, "Centralized vs. Decentralized: Performance Comparison between BigchainDB and Amazon QLDB," *Applied Sciences (Switzerland)*, vol. 13, no. 1, 2023, doi: 10.3390/app13010499.

[23] P. Ruan *et al.*, "Blockchains vs. Distributed Databases: Dichotomy and Fusion," *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 1504–1517, 2021, doi: 10.1145/3448016.3452789.

Abrar Alotaibi is a PhD candidate the Information and Computer Science Department at King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia. She received her master's degree in computer science in 2021, and her bachelor's degree from Imam Abdulrahman Bin Faisal University, Dammam, Saudi Arabia, in 2015. She holds a Lecturer position at the College of Computer Science and Information Technology in the Computer Science Department at Imam Abdulrahman Bin Faisal University. Her research interests are Image Processing, natural language processing, artificial intelligence and deep learning. She is a member of IEEE.

Sarah Alissa is a PhD candidate in the Information and Computer Science Department at King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia. She received her master's degree in computer science from University of Southampton, Southampton, United Kingdom in 2019, and her bachelor's degree from Imam Abdulrahman Bin Faisal University, Dammam, Saudi Arabia, in 2014. She holds a Lecturer position at the College of Computer Science and Information Technology in the Computer Science Department at Imam Abdulrahman Bin Faisal University. Her research interests are data analysis, natural language processing, and artificial intelligence.

SALAHADIN MOHAMMED received B.S. and M.S. degrees in computer science from King Fahd University of Petroleum and Minerals (KFUPM) and Ph.D. degree in Computer Science from the Computer Science Department, Monash University, Melbourne, Australia. He is currently an Assistant Professor with the Department of Information and Computer Science, KFUPM. He has more than 30 years of experience in industry and academia involving project management, software development, database administration, Unix administration, research, teaching, supervision, curriculum design, program assessment and quality assurance in higher education. He is a certified Oracle Database Administrator. He has more than 50 journal and conference publications. His research area is on machine learning, data mining and analytics, databases, data warehousing and big data.