SQL-Middleware: Enabling the Blockchain with SQL

Xing Tong[1], Haibo Tang[1], Nan Jiang[1], Wei Fan[1], Yichen Gao[1], Sijia Deng[1], Zhao Zhang[1(✉)], Cheqing Jin[1], Yingjie Yang[2], and Gang Qin[2]

[1] East China Normal University, Shanghai, China
{xtong,haibtang,njiang,wfan,ycgao,sjdeng}@stu.ecnu.edu.cn,
{zhzhang,cqjin}@dase.ecnu.edu.cn
[2] Ouyeel International Co., Ltd., Shanghai, China
{yangyingjie,qingang}@ouyeel.com

Abstract. With the development of blockchain, blockchain has a broad prospect as a new type of data management system. However, limited to the data modeling method of blockchain, the usability of blockchain is restricted; In addition, every blockchain system has its own native but naive interfaces, when developing based on the different blockchain systems, which will leads to low development efficiency and high development costs. In this study, we construct a SQL-Middleware for blockchain system to solve these problems. The SQL-Middleware first performs relational modeling of blockchain data, mapping the blockchain data into a relational table; On the basis of modeling the blockchain data, SQL-Middleware encapsulates a set of SQL interfaces for blockchain system, thus realizing the unification of interface access methods of different blockchain systems. At last, we implement the SQL-Middleware based on the open source blockchain system CITA. Demonstration shows that the SQL-Middleware greatly improves the data management capabilities of blockchain and simplifies the blockchain access steps.

Keywords: Blockchain · Middleware · Data modeling · SQL

1 Introduction

Blockchain, as a distributed ledger formed by multi-party consensus, can build a credible interactive platform for multiple parties who do not trust each other. As a new type of data management system, blockchain has many drawbacks: First, the expressive capability of blockchain data is weak [1], all data is modeled in a unified transaction format, which limits the potential value of data; Second, from the perspective of evolution of the data management system: from SQL to NoSQL, and then NewSQL, the continuous changes in data management methods have made us aware of the irreplaceable role of SQL in data management, however, generally, blockchain can be classified as a NoSQL system and only

Shanghai Engineering Research Center of Big Data Management.

C. S. Jensen et al. (Eds.): DASFAA 2021, LNCS 12683, pp. 622–626, 2021.
https://doi.org/10.1007/978-3-030-73200-4_48

supports RPC-based naive interfaces. These features restrict the evolution of the blockchain to be an excellent emerging data management system.

In addition, it is worth mentioning that we observe that transactions are structured data, which provides a basis for relational modeling. Based on this observation, we design a middleware called SQL-Middleware for blockchain system to abstract blockchain system as a relational data management system. The middleware we designed has nothing to do with the blockchain consensus and execution process. The reason for adopting the form of middleware is that middleware can transplant to other systems easily and not just limited to a specific blockchain system [2]. However, there are two challenges when building SQL-Middleware for blockchain system: *i*) How to convert a monotonous transaction-based model into different models towards different scenarios; *ii*) How to provide developers with efficient query interfaces. To solve these challenges, we use the **Data** in the transaction as an application-related field and encapsulate the application data into **Data** uniformly, and then we establish a schema for **Data** so that the data in the blockchain system has rich semantic models; Then, we implement part of SQL interfaces for blockchain system, application developers can interact with blockchain system through these SQL interfaces; Finally, we implement a **Terminal** based on the SQL-Middleware which we implemented.

In related research works, there are existing research works, but these works cannot be well compatible with different blockchain systems [1], and cannot support efficient query operations [2,3] or will bring relatively large storage overhead. On the contrary, SQL-Middleware solves these problems very well.

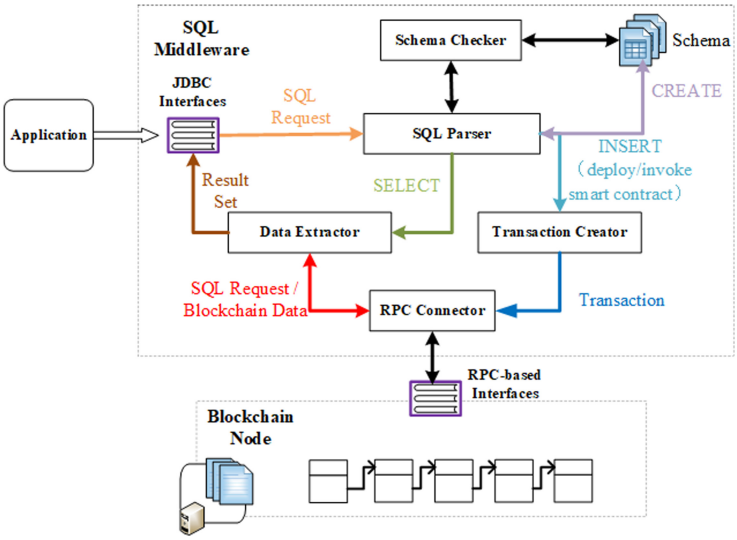

Fig. 1. System overview.

2 System Overview and Key Designs

The overall architecture of the system is shown in Fig. 1. Using SQL-Middleware, we abstract the underlying blockchain system into a SQL-based data management system. On the one hand, the SQL-Middleware model the blockchain system as a relational data management system, which enhance the expressive ability of blockchain data. On the other hand, SQL-Middleware encapsulates the RPC-based interfaces provided by the blockchain system into SQL interfaces, which makes the blockchain system behave like a database. The modules of SQL-Middleware include: *i*) JDBC interface module, applications can transmit SQL statements and result set through this module; *ii*) SQL parser module, which is responsible for parsing the SQL statements transmitted by the application and checking the validity of SQL statements based on locally maintained schema information; *iii*) Transaction Creator and Data Extractor, which are responsible for constructing transactions based on the results of SQL parser, and querying the required data based on the obtained blockchain data; *iv*) RPC Connector module, this module directly connects with the blockchain node by RPC, sends the constructed transaction to the blockchain node for consensus, and pulls the on-chain data from blockchain node.

2.1 Data Modeling and SQL Interfaces

Taking the popular blockchain systems as examples (e.g., Ethereum or CITA all transactions have unified structure. In transactions that used for invoking smart contracts, all parameters needed to call the contract are included in **Data** (Transaction.**Data** = {param1, param2, et al.}), that is, all application-related data is included in **Data**, resulting in a monotonous blockchain transaction model, which limits the data expression capability of blockchain data. To solve this problem, we parse the **Data** in transaction and model it into a table, and each parameter in **Data** is mapped into a field of table. Specifically, we map each function in the smart contract that can be called externally into a table. When smart contract is called, it is equivalent inserting an item into the database.

SQL query is an efficient query method, through SQL query interfaces, complicated query logic can be supported, which is an important support in data mining. However, as far as we can tell, only RPC-based naive interfaces are provided in blockchain system. Compared with SQL, native interfaces of blockchain only support simpler query logic and cannot support rich and efficient query operations. In order to solve this problem, we use the RPC-based interfaces provided by the blockchain system to encapsulate a set of SQL interfaces. We mainly implement three interfaces: CREATE, SELECT, and INSERT. The functions of these SQL statements that supported by SQL-Middleware are shown in Table 1.

3 Demonstrations Details

We develope a SQL-Middleware in C++ using CITA as a case. Based on SQL-Middleware, we implement a **Terminal**. Using this terminal, developers can per-

Table 1. Supported SQL statements and their functions.

SQL	Function	Processing flow
CREATE	Create schemas for on-chain data	Establish schema info to model on-chain data and use schema info to describe on-chain data, then persist it in schema file
INSERT	Insert data into the blockchain	Extract the field info in the INSERT statement, construct it into a transaction format and forward it to blockchain node
SELECT	Read on-chain data that meets the requirements	Pull data on the blockchain node and extract required data according to the schema information

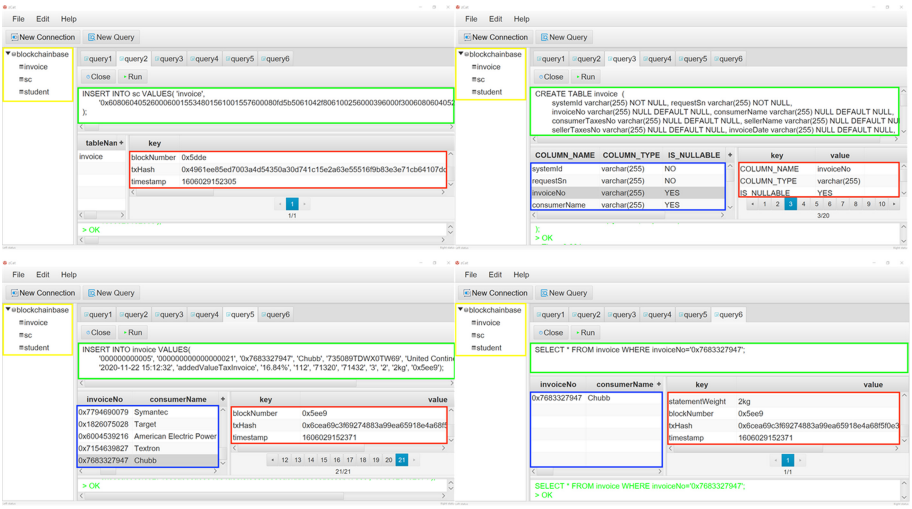

Fig. 2. Terminal panel when performing different operations. (Color figure online)

form assisted work on the blockchain, including designing or viewing the structures or contents of the tables in blockchain. Figure 2 shows the **Terminal** panel when performing different operations. The yellow-marked field represents the locally established schema information. In the green-marked field, we can input SQL statements and send them to our SQL-Middleware, then SQL-Middleware executes the SQL statements and displays the results in the blue-marked field, and red-marked attributes are blockchain attributes of the data. In addition, we develop a blockchain front-end layer based on SQL-Middleware, other systems can access the blockchain system through blockchain front-end layer by **POST** Request, which greatly expands the usage scenarios of blockchain system and deeply excavates the value of blockchain system as a data management system.

Acknowledgments. This work is partially supported by National Science Foundation of China (61972152, U1811264 and U1911203).

References

1. Zhu, Y., Zhang, Z., Jin, C., Zhou, A., Yan, Y.: SEBDB: semantics empowered blockchain database. In: ICDE, pp. 1820–1831. IEEE (2019)
2. El-Hindi, M., Binnig, C., Arasu, A., Kossmann, D., Ramamurthy, R.: BlockchainDB: a shared database on blockchains. In: VLDB Endowment, pp. 1597–1609 (2019)
3. Ji, Y., Chai, Y., Zhou, X., Ren, L., Qin, Y.: Smart intra-query fault tolerance for massive parallel processing databases. Data Sci. Eng. **5**(1), 65–79 (2019). https://doi.org/10.1007/s41019-019-00114-z