AERGOSQL: A New Smart Contract Engine for Blockchain

Last Updated: 17 July 2018, AERGO

AERGOSQL: A New Smart Contract Engine for Blockchain

Won-Beom Kim, Technical Committee head of the AERGO Limited

LEGAL DISCLAIMER

ABSTRACT

This paper describes AERGOSQL, the proposed smart contract engine that intends to operate on the AERGO platform. It assumes a base level of understanding regarding computer coding, smart contracts, programming and blockchain.

It is proposed that AERGOSQL will support a relational data model and business logic definition through PL/SQL-like scripting language. On AERGOSQL, it is intended that data models can be created using data definition language ("**DDL**") and manipulated or accessed using data manipulation language. Business logic utilizing the data model can be created and invoked PL/SQL-like syntax.

In order to support enterprise-level performance, AERGOSQL proposes to process smart contract definitions and executions through LLVM in order to utilize JIT compilation[1]. Pluggable storage engine support enables leveraging scalable storage engines such as WiredTiger[2].

1. PROBLEMS WITH SMART CONTACTS

Mainstream blockchain implementations such as Ethereum prefer procedural, Turing-complete support for smart contracts.[3] While a procedural smart contract support enables more flexible applications, the flexibility also allows for more errors and vulnerabilities.[4]

Since the majority of procedural smart contract languages are modeled around the principle of object-oriented programming, the data access on procedural smart contracts is often modeled after in-memory data structures as well. Other types of smart contract languages support more robust interfaces for key-value storages or document storages.[5] However, we believe that the data access functionalities offered by procedural smart contract languages are basic compared to more mature NoSQL or SQL databases and often result in complex and lengthy implementations to work around the limitations presented.

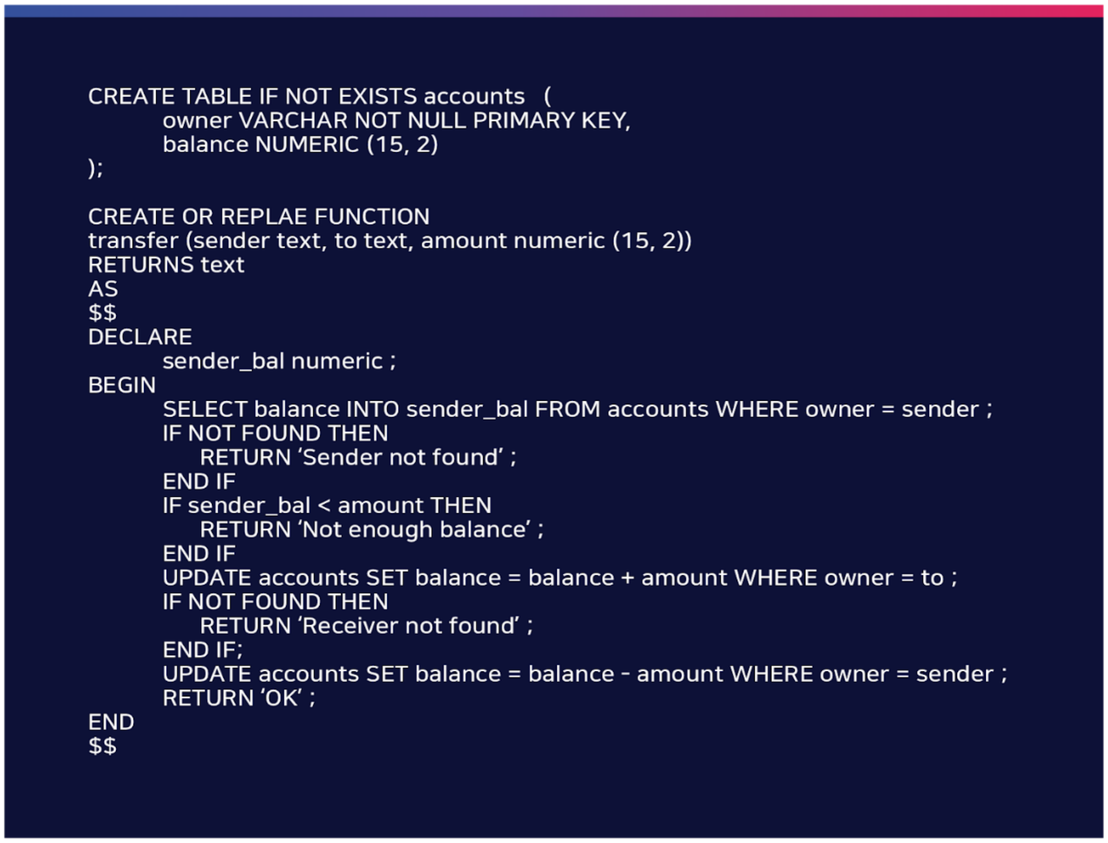
2. SQL AND SMART CONTRACTS

Relational data models and SQL interface provide far more efficient ways to access and manipulate data stored in ledgers than procedural means. In fact, the bulk of business logic

[1] https://llvm.org/devmtg/2016-09/slides/Melnik-PostgreSQLLLVM.pdf
[2] http://www.wiredtiger.com
[3] https://github.com/ethereum/wiki/wiki/White-Paper#scripting
[4] https://eprint.iacr.org/2016/1007.pdf
[5] https://medium.com/wearetheledger/hyperledger-fabric-couchdb-fantastic-queries-and-where-to-find-them-f8a3aecef767

required by most smart contract use-cases can be represented by DDL and constraints only. Scripting languages designed around SQL, such as PL/SQL, provide more complete support for building smart contracts.

The following figure is an illustrative example of a balance accounting system implemented using PL/SQL.



```
CREATE TABLE IF NOT EXISTS accounts   (
        owner VARCHAR NOT NULL PRIMARY KEY,
        balance NUMERIC (15, 2)
);

CREATE OR REPLAE FUNCTION
transfer (sender text, to text, amount numeric (15, 2))
RETURNS text
AS
$$
DECLARE
        sender_bal numeric ;
BEGIN
        SELECT balance INTO sender_bal FROM accounts WHERE owner = sender ;
        IF NOT FOUND THEN
            RETURN 'Sender not found' ;
        END IF
        IF sender_bal < amount THEN
            RETURN 'Not enough balance' ;
        END IF
        UPDATE accounts SET balance = balance + amount WHERE owner = to ;
        IF NOT FOUND THEN
            RETURN 'Receiver not found' ;
        END IF;
        UPDATE accounts SET balance = balance - amount WHERE owner = sender ;
        RETURN 'OK' ;
END
$$
```

Balance accounting system implemented using PL/SQL

3. DESIGN OF ARGOSQL

The proposed design of AERGOSQL consists of three layers of functionality.

The Frontend

The frontend of AERGOSQL parses PL/SQL and creates an AST representation of the script. A dialect of PL/SQL optimized for usage on permissioned ledgers is used for both simplicity and functionality. ANTLR[6] is used to process the EBNF for PL/SQL dialect and generate a parser in Go language.

The Optimizer
Based on the AST emitted by the frontend, the optimizer creates the IR of the script for LLVM. In order to maximize performance, the optimizer allocates the right nodes of execution in IR.

The Backend
The backend of AERGOSQL is intended to provide the abstraction layer for the functionalities required by the execution nodes utilized by IR. In order to enable optimization, the backend provides relevant statistics of the persisted data as well. The preferred choice of backend in the initial design is WiredTiger, but AERGOSQL should be able to utilize any data storage backed by b-tree or LSM tree.

In order to support different types of consensus algorithm implemented by various ledgers, AERGOSQL provides the point-in-block mechanism for rollback and recovery. Such functionality allows blockchain implementations with block reorganization to utilize AERGOSQL.

4. IMPLICATIONS

The familiar SQL interface supported by AERGOSQL is intended to allow blockchains to support more developer-friendly methodologies for building smart contracts. AERGOSQL seeks to improve performance and scalability, which in turn should enable more demanding use-cases to be realized on blockchain as well.

[6] http://www.antlr.org