Ethereum Query Language

Santiago Bragagnolo, Henrique Rocha, Marcus Denker, Stephane Ducasse

Rahul Panchal - 1911097 - B2

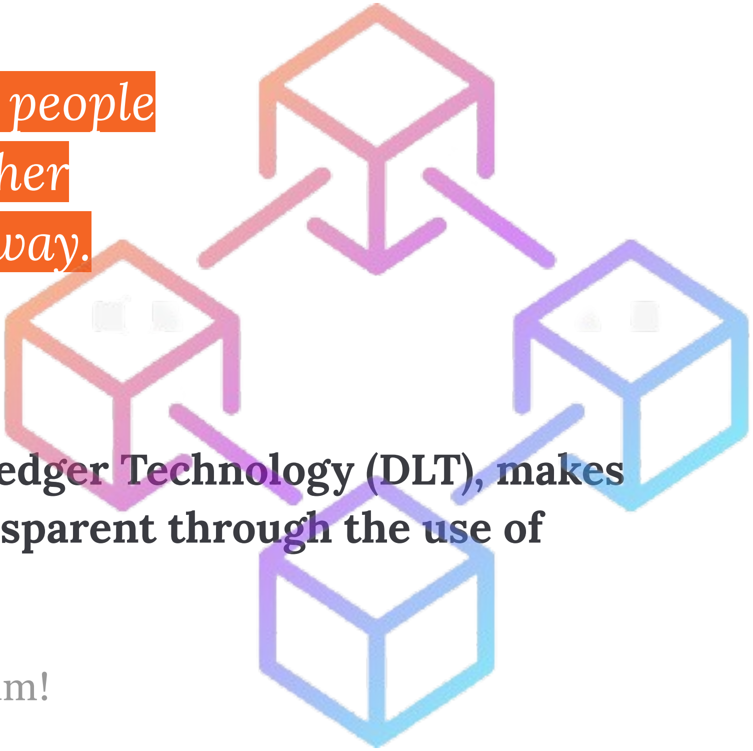
What is Ethereum?

- For that you might first want to know what is Blockchain!

What is BLOCKCHAIN ?

The whole point of using a blockchain is to let people in particular, people who don't trust one another share valuable data in a secure, tamperproof way.

— MIT Technology Review



Blockchain, sometimes referred to as Distributed Ledger Technology (DLT), makes the history of any digital asset unalterable and transparent through the use of decentralization and cryptographic hashing.

- Now we can go ahead and understand what is Ethereum!

What is Ethereum?

It is a Blockchain network.

Ethereum was proposed in 2013 by programmer Vitalik Buterin. Development was crowdfunded in 2014, and the network went live on 30 July 2015.

Like BitCoin?

No! BitCoin is a cryptocurrency.

Ethereum is a network.

Ether (ETH) is the native cryptocurrency of the platform. It is the second-largest cryptocurrency by market value.

How is it special?

Ethereum provides a feature known as *“Smart Contract”*.

A "smart contract" is simply a program that runs on the Ethereum blockchain. It's a collection of code (its functions) and data (its state) that resides at a specific address on the Ethereum blockchain.

Problem that the paper focuses on!

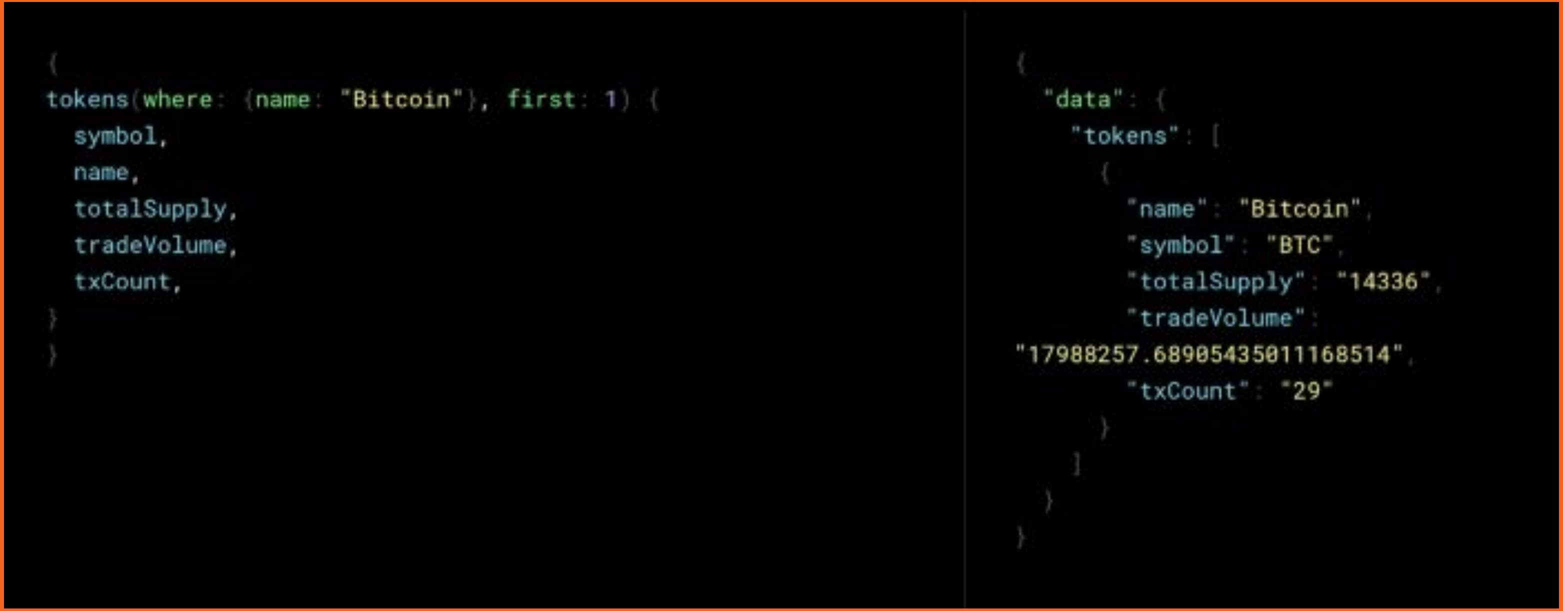
Ethereum and other blockchains store a massive amount of heterogeneous data. Ethereum is estimated to have approximately 300GB of data. Retrieving information from this massive data is not an easy task. Moreover, the Ethereum platform only allows direct access to its first-class data elements, which includes blocks, transactions, accounts, and contracts. Therefore, if we search for a particular information inside a data element, we would need a unique identifier (i.e., either the number or its hash) to access the block containing such information. Another alternative would be to direct access one block and sequentially access its parents to search for a data. Moreover, the information returned by the Ethereum platform when we access its blocks is encoded into a JSON-like structure that we need to interpret to acquire a specific item. Therefore, the Ethereum platform does not provide a semantic way to search for information and neither an easy form to present such information.

Problem that the paper focuses on!

Ethereum and other blockchains store a massive amount of heterogeneous data. Ethereum is estimated to have approximately 300GB of data. Retrieving information from this massive data is not an easy task. Moreover, the Ethereum platform only allows direct access to its first-class data elements, which includes blocks, transactions, accounts, and contracts. Therefore, if we search for a particular information inside a data element, we would need a unique identifier (i.e., either the number or its hash) to access the block containing such information. Another alternative would be to direct access one block and sequentially access its parents to search for a data. Moreover, the information returned by the Ethereum platform when we access its blocks is encoded into a JSON-like structure that we need to interpret to acquire a specific item. Therefore, the Ethereum platform does not provide a semantic way to search for information and neither an easy form to present such information.

Tools used for querying Ethereum Network:

GraphQL: a query language used for requesting data from a database or a public API endpoint. Unlike SQL (which is used for REST APIs), this language uses JSON notation and lets you make multiple queries within the same network request.



```
{
tokens(where: {name: "Bitcoin"}, first: 1) {
  symbol,
  name,
  totalSupply,
  tradeVolume,
  txCount,
}
}
```

```
{
  "data": {
    "tokens": [
      {
        "name": "Bitcoin",
        "symbol": "BTC",
        "totalSupply": "14336",
        "tradeVolume":
"17988257.68905435011168514",
        "txCount": "29"
      }
    ]
  }
}
```

"

In a classical database, when we need to search for a particular information, we usually write a query to fetch, present, fiter, and format such information. Database query languages like SQL provide a rich syntax to describe the data we want to acquire from the database.

Since blockchain can be considered a database, it would be better if we could use a similar way to fetch information inside the blocks. In this paper, we propose the Ethereum Query Language(EQL), a query language that enables its users to acquire information in the blockchain by writing SQL-like queries. Our goal is to provide an easier way to fetch, format, and present information extracted from the blockchain database.

"

CHALLENGES WHILE QUERYING DATA FROM BLOCKCHAIN NETWORK

Massive Data

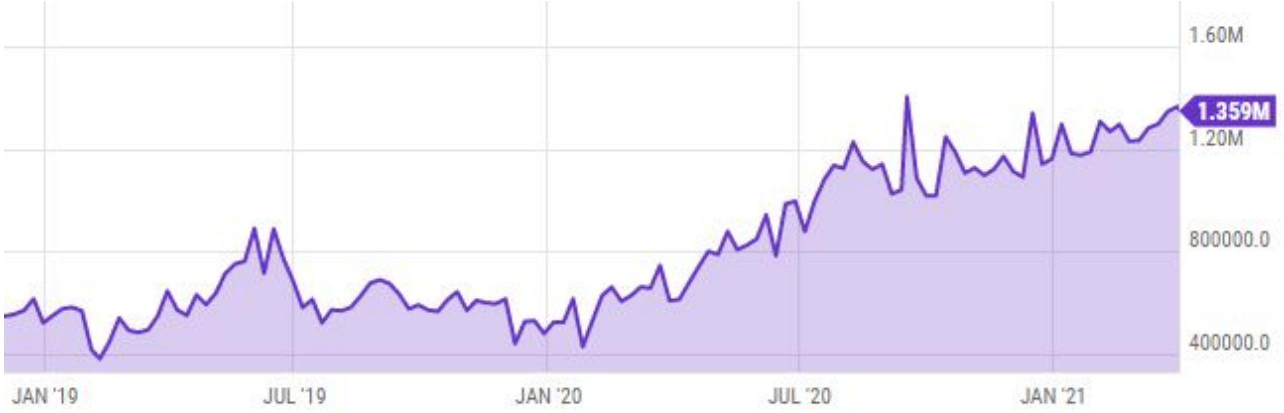
Heterogeneous Data

Direct Access

Data Opaqueness

Ethereum is estimated to have approximately 300GB of data.

Ethereum is currently processing approximately 1.3 million transactions per day.

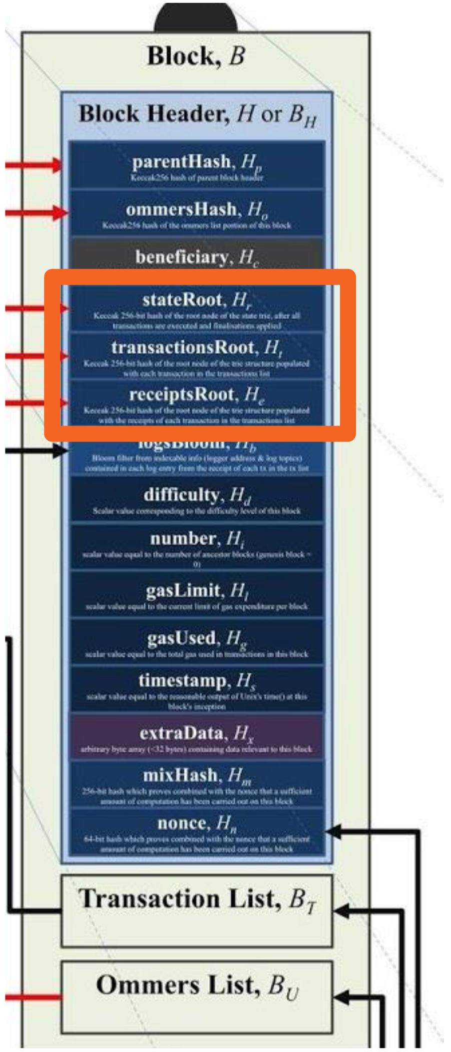


CHALLENGES WHILE QUERYING DATA FROM BLOCKCHAIN NETWORK

- Massive Data
- Heterogeneous Data
- Direct Access
- Data Opaqueness

Blockchain not only stores a great amount of data but also manages a mixture of first-class elements such as transactions, blocks, accounts, and smart contracts.

All of the first-class elements (blocks, transactions, etc.) are different but interrelated by hashes. Even though Ethereum allows access to any of its first-class elements, the heterogeneity of the elements (each one with different meaning and high-level representation)
complicates the acquisition of information


Block, B
Block Header, H or B_H
parentHash, H_p
Keccak256 hash of parent block header
ommersHash, H_o
Keccak256 hash of the ommers list portion of this block
beneficiary, H_c
stateRoot, H_r
Keccak 256-bit hash of the root node of the state trie, after all transactions are executed and finalisations applied
transactionsRoot, H_t
Keccak 256-bit hash of the root node of the trie structure populated with each transaction in the transactions list
receiptsRoot, H_e
Keccak 256-bit hash of the root node of the trie structure populated with the receipts of each transaction in the transactions list
logsBloom, H_b
Bloom filter from indexable info (logger address & log topics) contained in each log entry from the receipt of each tx in the tx list
difficulty, H_d
Scalar value corresponding to the difficulty level of this block
number, H_i
scalar value equal to the number of ancestor blocks (genesis block = 0)
gasLimit, H_l
scalar value equal to the current limit of gas expenditure per block
gasUsed, H_g
scalar value equal to the total gas used in transactions in this block
timestamp, H_s
scalar value equal to the reasonable output of Unix's time() at this block's inception
extraData, H_x
arbitrary byte array (<32 bytes) containing data relevant to this block
mixHash, H_m
256-bit hash which proves combined with the nonce that a sufficient amount of computation has been carried out on this block
nonce, H_n
64-bit hash which proves combined with the nonce that a sufficient amount of computation has been carried out on this block
Transaction List, B_T
Ommers List, B_U

CHALLENGES WHILE QUERYING DATA FROM BLOCKCHAIN NETWORK

- Massive Data
- Heterogeneous Data
- Direct Access
- Data Opaqueness

In general, blockchains only allow direct access to its elements by using a unique identifier. This identifier is a hash number that is generated for every first-class element stored in the blockchain.

However, this direct access by hash can lead to the following issues when we consider the user's point-of-view:

- **Hash Storage:**
  Since the user needs the hash to fetch the information, he/she is also required to store the hash, someway, if he/she wants to acquire the same information later. Otherwise, the user will lose the key to access the desired information in the future.

- **Sequential Access:**
  If the user loses the hash, he/she can still find the relevant data by performing a sequential access in the blockchain. In Ethereum, it is possible to start at the most recent block and sequentially access the block parents.

CHALLENGES WHILE QUERYING DATA FROM BLOCKCHAIN NETWORK

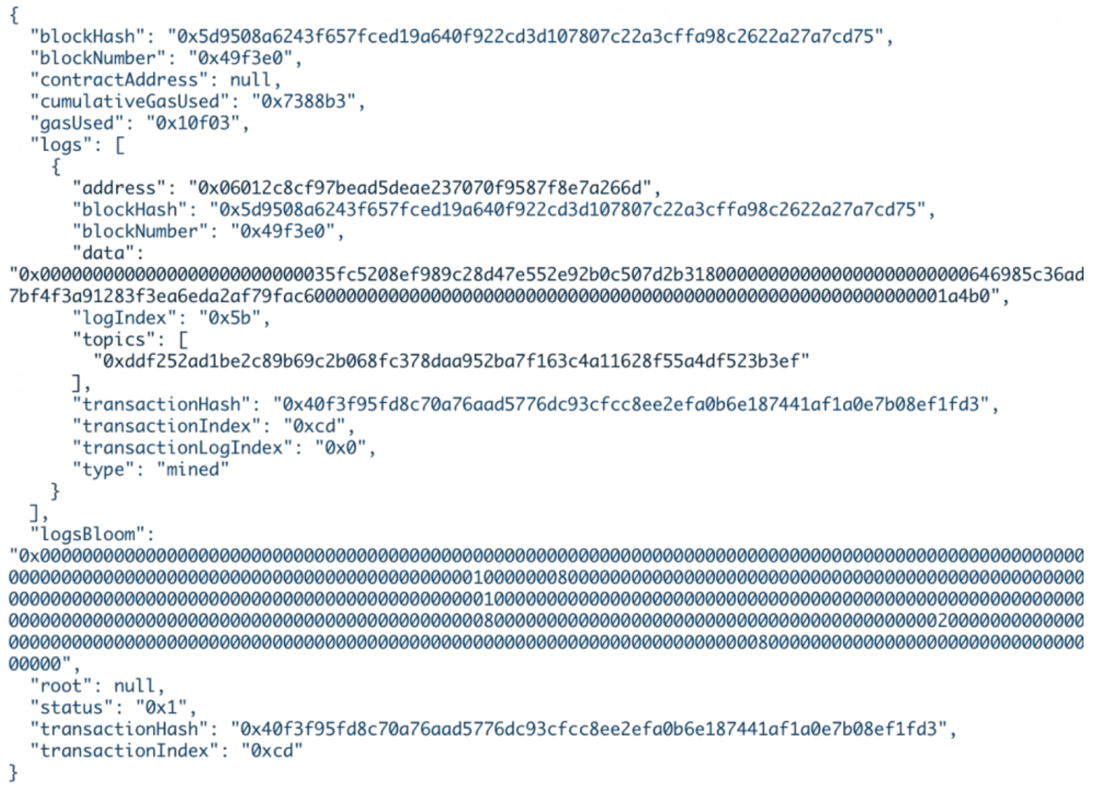
Massive Data

Heterogeneous Data

Direct Access

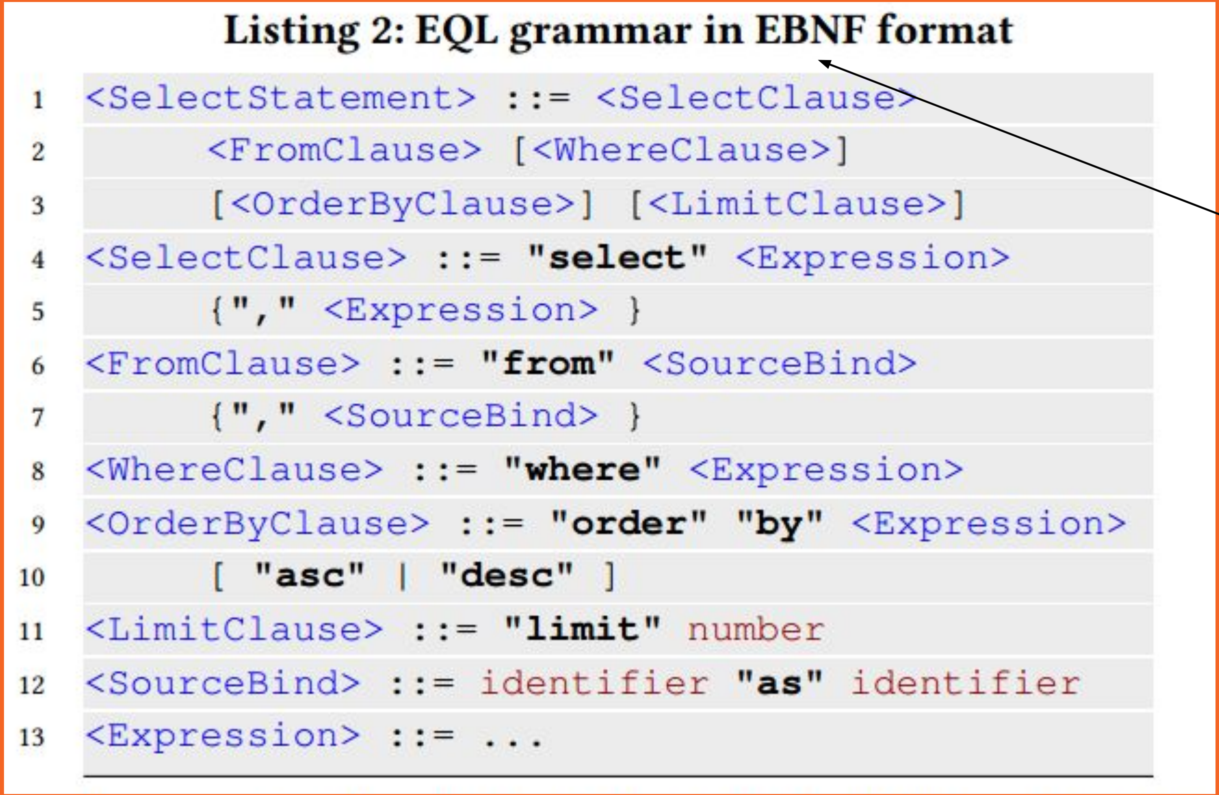
Data Opaqueness

In order to give users flexibility, Ethereum stores arbitrary information (e.g., contracts, transactions) with a generic representation.
Therefore, the stored data is opaque, since there is no meta-data describing the information and neither a simple way to know what was recorded



```
{
  "blockHash": "0x5d9508a6243f657fced19a640f922cd3d107807c22a3cffa98c2622a27a7cd75",
  "blockNumber": "0x49f3e0",
  "contractAddress": null,
  "cumulativeGasUsed": "0x7388b3",
  "gasUsed": "0x10f03",
  "logs": [
    {
      "address": "0x06012c8cf97bead5deae237070f9587f8e7a266d",
      "blockHash": "0x5d9508a6243f657fced19a640f922cd3d107807c22a3cffa98c2622a27a7cd75",
      "blockNumber": "0x49f3e0",
      "data":
"0x0000000000000000000000000035fc5208ef989c28d47e552e92b0c507d2b318000000000000000000000000646985c36ad
7bf4f3a91283f3ea6eda2af79fac600000000000000000000000000000000000000000000000000000000001a4b0",
      "logIndex": "0x5b",
      "topics": [
        "0xddf252ad1be2c89b69c2b068fc378daa952ba7f163c4a11628f55a4df523b3ef"
      ],
      "transactionHash": "0x40f3f95fd8c70a76aad5776dc93cfcc8ee2efa0b6e187441af1a0e7b08ef1fd3",
      "transactionIndex": "0xcd",
      "transactionLogIndex": "0x0",
      "type": "mined"
    }
  ],
  "logsBloom":
"0x000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000001000000080000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000100000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000800000000000000000000000000000000000000000020000000000000
000000000000000000000000000000000000000000000000000000000000000000000008000000000000000000000000000000
00000",
  "root": null,
  "status": "0x1",
  "transactionHash": "0x40f3f95fd8c70a76aad5776dc93cfcc8ee2efa0b6e187441af1a0e7b08ef1fd3",
  "transactionIndex": "0xcd"
}
```

Ethereum Query Language



Listing 2: EQL grammar in EBNF format

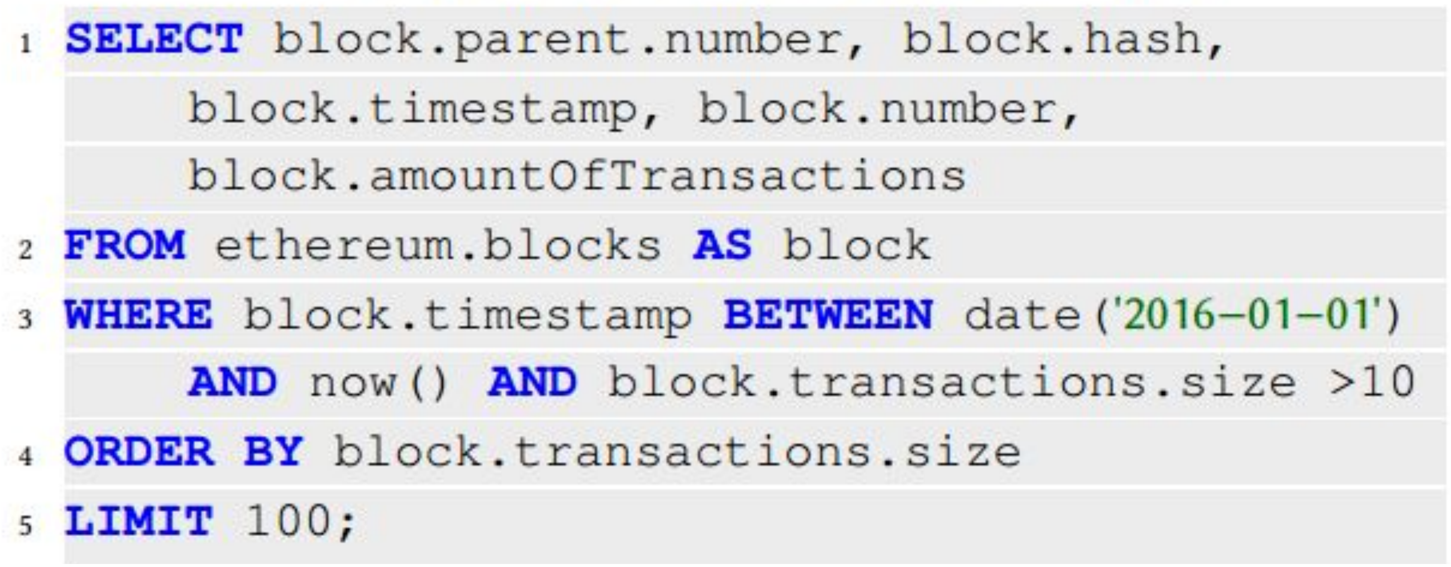
```
<SelectStatement> ::= <SelectClause>
      <FromClause> [<WhereClause>]
      [<OrderByClause>] [<LimitClause>]
<SelectClause> ::= "select" <Expression>
      {"," <Expression> }
<FromClause> ::= "from" <SourceBind>
      {"," <SourceBind> }
<WhereClause> ::= "where" <Expression>
<OrderByClause> ::= "order" "by" <Expression>
      [ "asc" | "desc" ]
<LimitClause> ::= "limit" number
<SourceBind> ::= identifier "as" identifier
<Expression> ::= ...
```

Extended Backus-Naur Form

* Unlike SQL, the EQL “Select” statement does not have a *group by* clause.

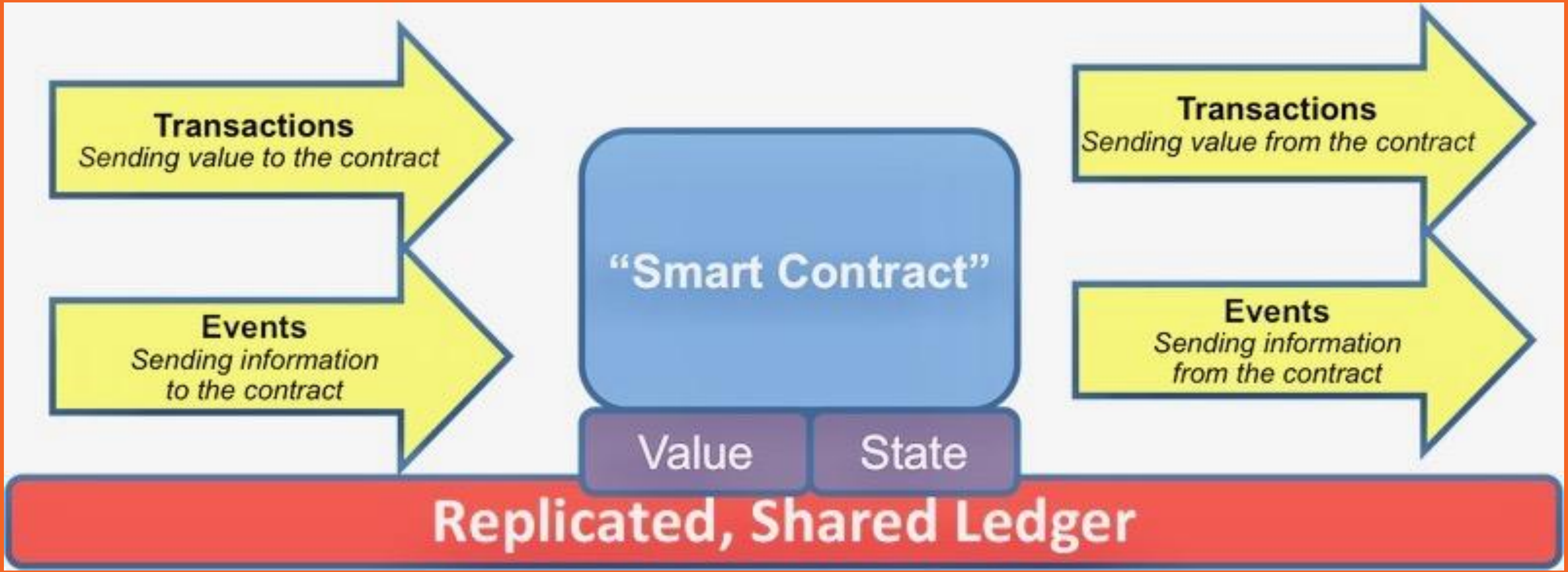
Example Queries

Listing 3: EQL Block Query Example



```
SELECT block.parent.number, block.hash,
    block.timestamp, block.number,
    block.amountOfTransactions
FROM ethereum.blocks AS block
WHERE block.timestamp BETWEEN date('2016-01-01')
    AND now() AND block.transactions.size >10
ORDER BY block.transactions.size
LIMIT 100;
```

Ethereum Smart Contracts


Transactions
Sending value to the contract
Events
Sending information to the contract
"Smart Contract"
Value
State
Transactions
Sending value from the contract
Events
Sending information from the contract
Replicated, Shared Ledger

What are Smart Contracts?

The term “Smart Contract” was first used by Nick Szabo, a computer scientist and cryptographer.

A "smart contract" is simply a program that runs on the Ethereum blockchain. It's a collection of code (its functions) and data (its state) that resides at a specific address on the Ethereum blockchain.

KICKSTARTER

Kickstarter is an American public benefit corporation based in Brooklyn, New York, that maintains a global crowdfunding platform focused on creativity. The company's stated mission is to "help bring creative projects to life".

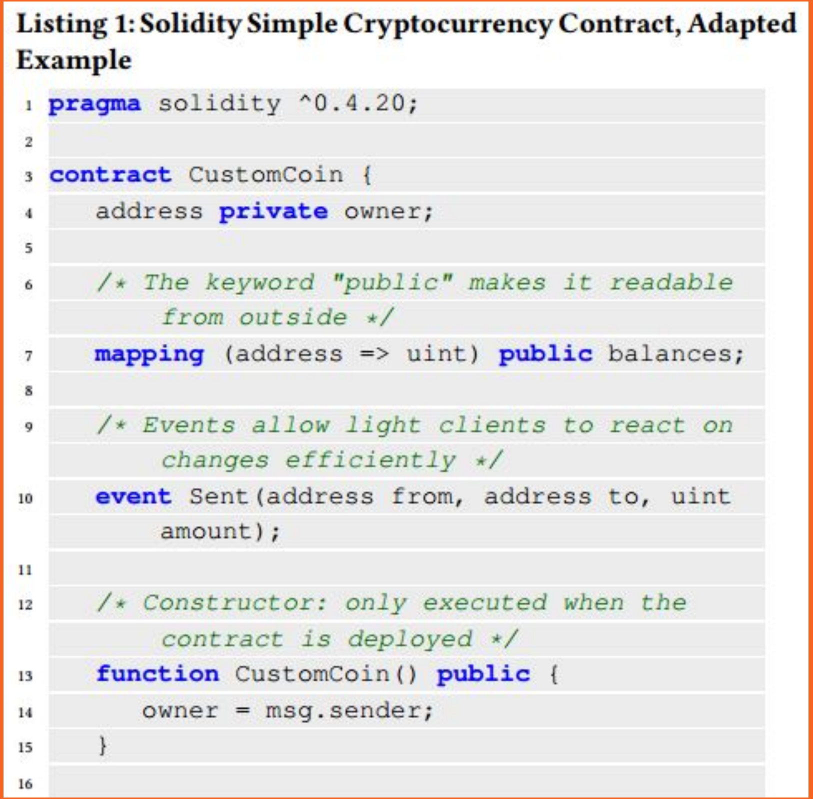
How are Smart Contracts related to Blockchain?

Right now, there are a handful of blockchains that support smart contracts, but the biggest one is Ethereum.

Ethereum was specifically created and designed to support smart contracts. These contracts can be programmed in a special programming language called Solidity.

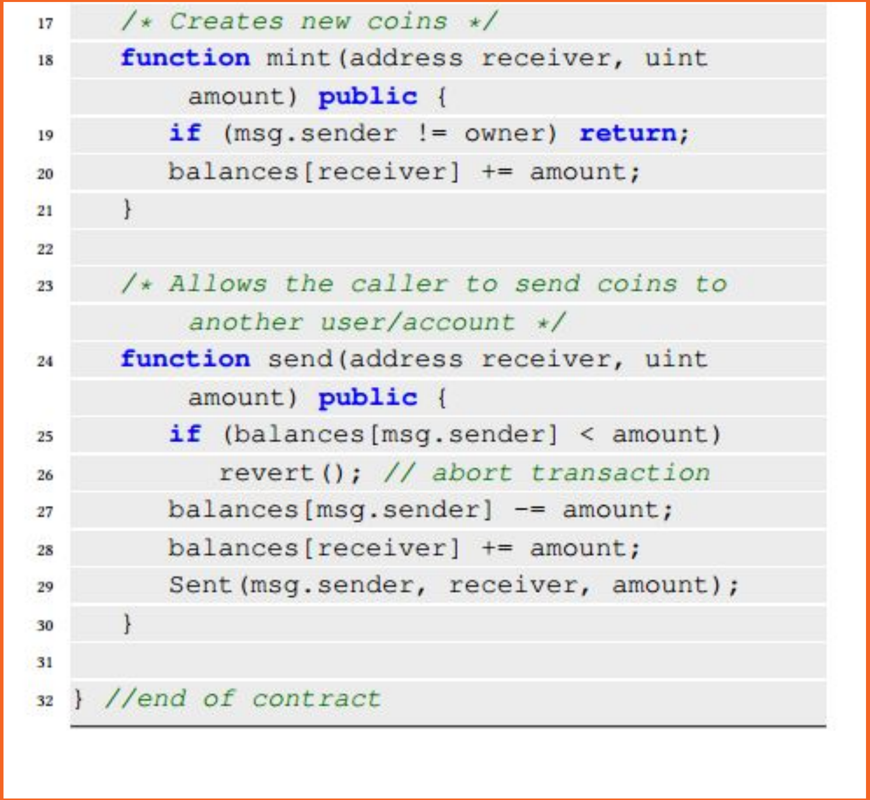
Solidity is a programming language with a syntax that resembles JavaScript.

A Smart Contract written using Solidity Programming Language



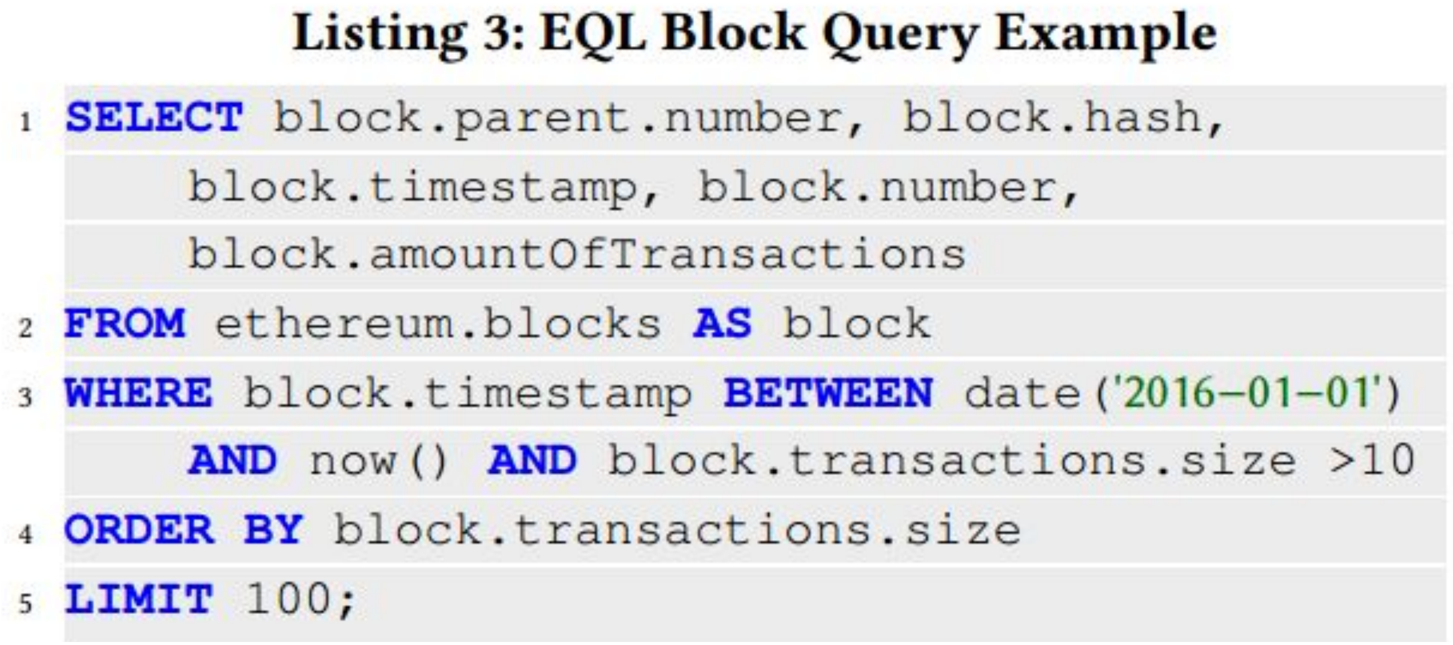
Listing 1: Solidity Simple Cryptocurrency Contract, Adapted Example

```
pragma solidity ^0.4.20;

contract CustomCoin {
    address private owner;

    /* The keyword "public" makes it readable
          from outside */
    mapping (address => uint) public balances;

    /* Events allow light clients to react on
          changes efficiently */
    event Sent(address from, address to, uint
          amount);

    /* Constructor: only executed when the
          contract is deployed */
    function CustomCoin() public {
        owner = msg.sender;
    }

```



```
    /* Creates new coins */
    function mint(address receiver, uint
          amount) public {
        if (msg.sender != owner) return;
        balances[receiver] += amount;
    }

    /* Allows the caller to send coins to
          another user/account */
    function send(address receiver, uint
          amount) public {
        if (balances[msg.sender] < amount)
           revert(); // abort transaction
        balances[msg.sender] -= amount;
        balances[receiver] += amount;
        Sent(msg.sender, receiver, amount);
    }

} //end of contract
```

Example Queries



Listing 3: EQL Block Query Example

```
SELECT block.parent.number, block.hash,
    block.timestamp, block.number,
    block.amountOfTransactions
FROM ethereum.blocks AS block
WHERE block.timestamp BETWEEN date('2016–01–01')
    AND now() AND block.transactions.size >10
ORDER BY block.transactions.size
LIMIT 100;
```

EQL Collections and Objects

Each one of those collections represents all first-class elements from a particular type (blocks, transactions, accounts, or contracts).

- blocks
- transactions
- accounts
- contracts

When we retrieve information from a collection of blocks, the result will be presented as block objects.

Ethereum blocks have attributes related to their storage structure. Blocks queried by EQL have the following attributes:

• number:	integer, the block's number.
• hash:	hash (binary 32 bytes), the block's hash.
• parentHash:	hash, the parent's block hash
• size:	integer, the size of the block in bytes.
• miner:	address (binary 20 bytes), the account who mined.
• transactionsRoot:	hash, the first transaction on this block.

EQL Collections and Objects

Each one of those collections represents all first-class elements from a particular type (blocks, transactions, accounts, or contracts).

- blocks
- transactions
- accounts
- contracts

In EQL, transaction objects have the following attributes:

• hash:	hash, the transaction's hash
• fromHash:	address, the address of sender account.
• from:	account, the sender's account.
• toHash:	address, the address of the receiver of this transaction.
• to:	account, the account of the receiver of this transaction.
• value:	integer, the amount transferred in Wei.
• gasPrice:	integer, the gas price set by the sender in Wei.
• gas:	integer, gas consumed by this transaction.

EQL Collections and Objects

Each one of those collections represents all first-class elements from a particular type (blocks, transactions, accounts, or contracts).

- blocks
- transactions
- accounts
- contracts

Accounts are a simple object that have the following attributes in EQL:

• address:	address, the account's address.
• name:	string, the account's name.
• balance:	integer, the amount of cryptocurrency (in Wei) on account.

EQL Collections and Objects

Each one of those collections represents all first-class elements from a particular type (blocks, transactions, accounts, or contracts).

- blocks
- transactions
- accounts
- contracts

A contract object is a special type of an account.
In the current version, contracts have the following attributes:

• address:	address, the contract's address.
• name:	string, the contract's name.
• balance:	integer, the amount of cryptocurrency (in Wei) on this contract.
• bytecode:	binary variable size, the contract code

Limitations

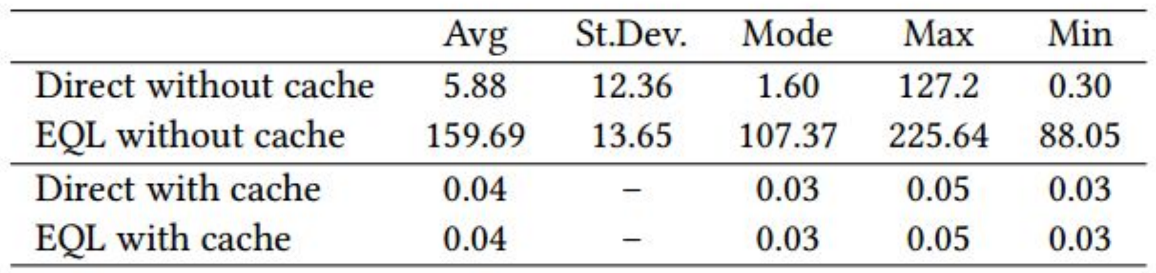
The current version of the EQL implementation (version 0.8) has some limitations.

- ⚠ **We are not able to search inside contract attributes when querying.**
- ⚠ **We are unable to use smart contract functions on EQL expressions.**
- ⚠ **A maximum upper bound on the number of results returned by EQL.**
- ⚠ **The lack of group by clause in EQL is also another limitation.**

Evaluation

For the evaluation, we employed direct access to fetch 100 randomly selected blocks. Then, we used an EQL query to fetch 100 blocks. First, we executed both retrieval operations with an empty cache. Second, we repeated the same operations to verify how both would perform when the searched information is already in the cache.

Table 1: Performance Tests to Fetch Blocks measured in Milliseconds



	Avg	St.Dev.	Mode	Max	Min
Direct without cache	5.88	12.36	1.60	127.2	0.30
EQL without cache	159.69	13.65	107.37	225.64	88.05
Direct with cache	0.04	–	0.03	0.05	0.03
EQL with cache	0.04	–	0.03	0.05	0.03

Conclusions

The Ethereum platform only allows direct or sequential access to its blocks. In this context, searching for information inside the blockchain is a challenging task because we must sequentially access a huge amount of opaque data.

To help in this challenge, we proposed EQL, a query language that allows users to retrieve information by writing SQL like queries.

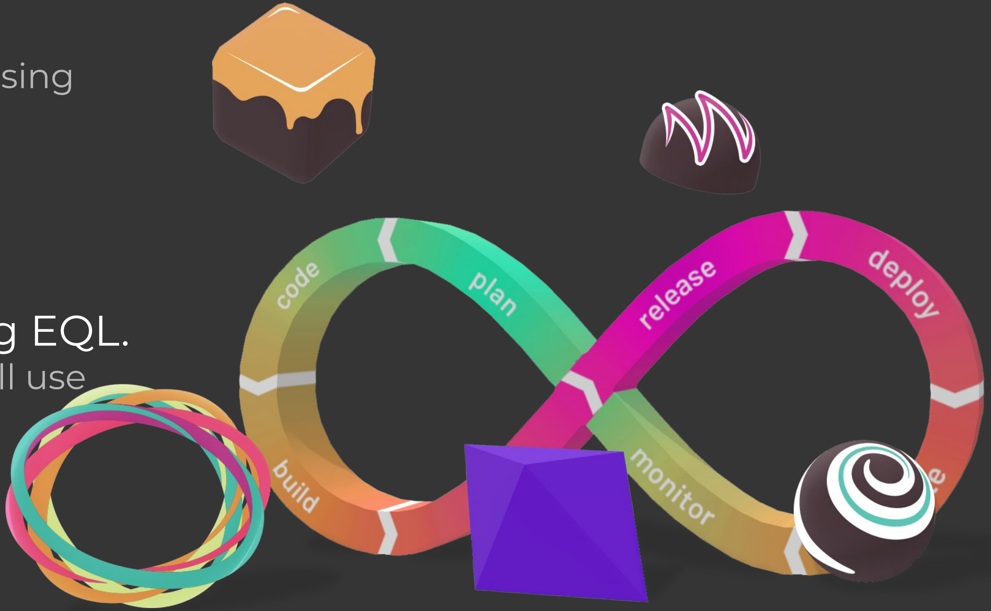
EQL is much slower when the information is not cached, than direct access. However, when the information being searched is already cached, then EQL reach similar results to direct access.

Bibliography

- ➔ Ethereum Transactions Per Day
- ➔ Smart Contracts On The Blockchain: A deep dive in to Smart Contracts
- ➔ Stefano Tempesta – Medium
- ➔ Solidity — Solidity 0.8.3 documentation
- ➔ GraphQL Walkthrough: How to Query Crypto (with Uniswap & DeFi)
- ➔ Truffle Suite: Sweet Tools for Smart Contracts
- ➔ What Is Blockchain Technology? How Does It Work?

Implementation Strategy

- Create an Ethereum BlockChain Application.
  A basic web based to-do list application using Truffle Framework ,Solidity, Node.js.
  Every task will be stored in a transaction.
  A web frontend to view the tasks.
  If successful, try to make a Crowdfunding application.
- Query data from application using EQL.
  Alternative (in case EQL does not run): Will use GraphQL
  or other available querying tools.



Thank You