BLOCKCHAIN

Using Blockchain as Database Solution
Joseph R. Tursi

Table of Contents

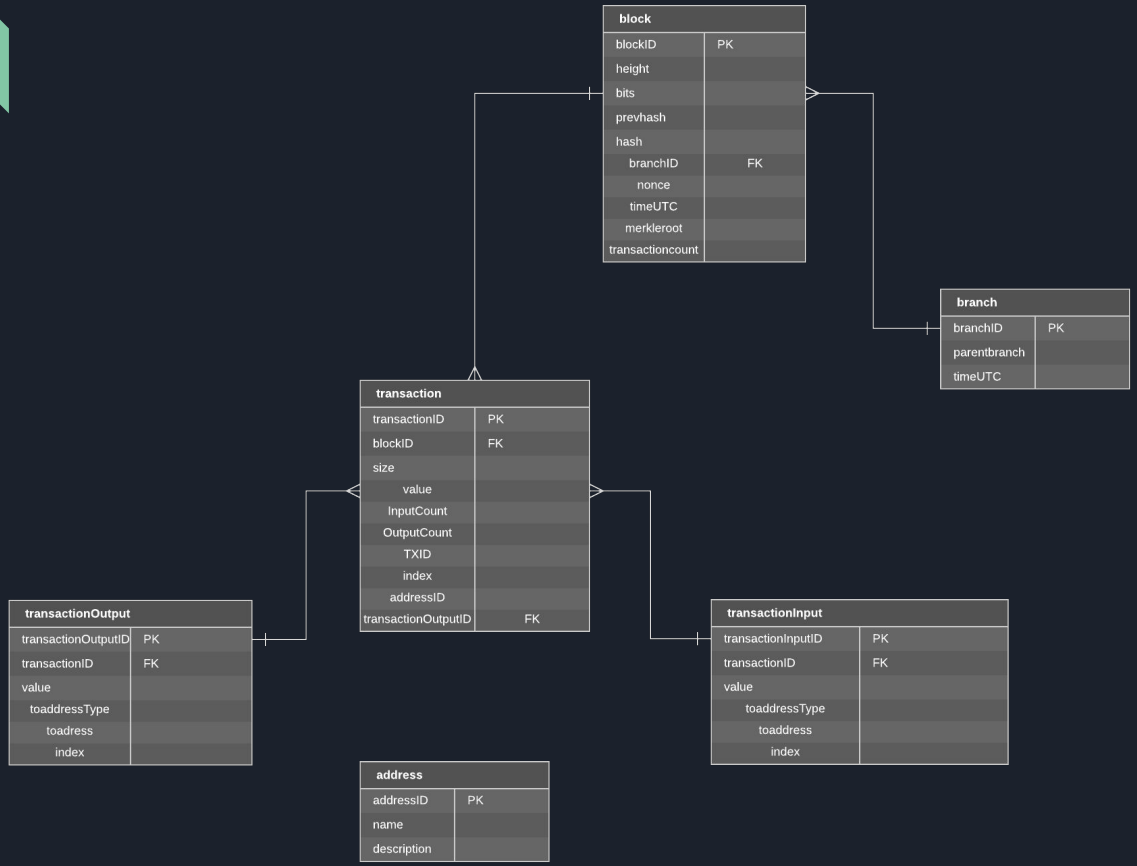
Executive Summary

By design, the blockchain concept is a decentralized technology. Blockchain stores data across its network, so anything that happens on it is a function of the network as a whole. Ever since blockchain burst onto the scene it's been a technology surrounded by hype, with the practical uses of cryptocurrencies.

This document outlines the design of a database which holds data on the cryptocurrency, bitcoin. The design focuses on the transaction of bitcoin, which could be altered to be any product or service. An overview of the database will be presented with explanation of each table and their purposes. The views, reports, triggers, and stored procedures will be implemented, explained, and contain sample data.

This implementation is intended to be used as a test case to demonstrate that blockchain could be easily integrated into your product or service. The blockchain could potentially get rid of the middleman for various transactions . This project is just scratching the surface of the concept of blockchain, which will change the way business is conducted in the future. The design does need improvements and will be built upon.

Entity Relationship Diagram



Branch Table

```
create table branch(
branchID              int     unique,
parentbranch   int     not null,
timestamUTC    date    not null,
primary key(branchID)
);
```

Functional Dependencies
branchID>parentbranch, BlockTimeStamp

branchid integer	parentbranch integer	blocktimestamp date
12	11	2017-01-01

Block Table

Contains information about the Bitcoin blocks.

```
create table block(
blockID                    int      unique,
height          int                 NOT NULL,
bits            bigint              NOT NULL,
prevhash        int                 NOT NULL,
hash            int                 NOT NULL,
branchID        int                 NOT NULL,
nonce           bigint              NOT NULL,
timestamUTC     date                NOT NULL,,
timestamUnix    bigint,
merkleroot      int,
transactioncount bigint,
primary key(blockID)
);
```

blockid bigint	branchid integer	blockchainfileid integer	bitcointransactionid bigint	blockversion integer	blockhash character varying (200)	previousblockhash character varying (200)	blocktimestamp date	transactionhash character varying (32)
100	1	1	25	1	ab	aabc	2017-07-19	cb
101	1	2	26	1	abc	abcd	2017-12-05	ca
102	1	3	27	1	abd	abde	2017-02-22	cd
103	1	4	28	1	abe	aabef	2017-05-12	ce
104	1	5	29	1	abf	abg	2017-07-19	cf

Functional Dependencies

BlockId> branchID, BlockchainFileId, BitcoinTransactionId, BlockVersion, BlockHash, PreviousBlockHash, BlockTimestamp,TransactionHash

Transaction Table

Contains information about the Bitcoin transactions.

```
create table transaction(
transactionID         bigint  unique,
blockID                       int     not null,
size                          bigint  not null,
InputCount                    bigint  not null,
OutputCount                   bigint  not null,
TXID                          bigint  not null,
index                         bigint  not null,
primary key(transactionID)
);
```

Functional Dependencies

BitcoinTransactionId> BlockId, TXID, TransactionHash, TransactionVersion, TransactionLockTime

TransactionInput Table

Contains information about the Bitcoin transaction inputs.

```
create table transactionInput(
transactionInputID                  bigint        unique,
transactionID                       bigint        not null,
transactionOutputID                 bigint        not null,
value                                      bigint        not null,
sequence                                   bigint        not null,
index                                      bigint        not null,
primary key(transactionInputID)
);
```

Functional Dependencies

TransactionInputId> SourceTransactionOutputId, TransactionOutputId, OutputHash, OutputIndex, index

transactioninputid bigint	sourcetransactionoutputid bigint	transactionoutputid bigint	outputhash text	outputindex integer	index integer	bitcointransactionid bigint
335	25	100	zzzf	0	0	25
336	26	101	abcd	0	0	26
337	27	102	dcba	0	0	27
338	28	103	badd	0	0	28
339	29	104	badc	0	0	29

TransactionOutput Table Contains information about the Bitcoin

transaction outputs.

```
create table transactionOutput(
transactionOutputID          bigint  unique,
transactionID                bigint  not null,
value                               bigint  not null,
toaddressType                int     not null,
toaddress                           int     not null,
index                               bigint  not null,
primary key(transactionOutputID)
);
```

Functional Dependencies

TransactionOutputId>BitcoinTransactionId, OutputIndex, OutputValueBtc, OutputScript, index, InputHash, InputIndex

transactionoutputid bigint	bitcointransactionid bigint	outputindex integer	outputvaluebtc numeric (20,8)	outputscript character varying (300)	index integer	inputhash character varying (64)	inputindex integer
7	25	0	5.60000000	Complete	0	a	0
8	26	0	14.20000000	Complete	0	a	0
9	27	0	201.30000000	Complete	0	a	0
10	28	0	32.00000000	Complete	0	a	0
11	29	0	8.20000000	Complete	0	a	0

address Table

Contains information about the Private/Public Keys aka. address

```
create table address(
addressID             bigint        unique,
name                  varchar(255) not null,
description           varchar(255) not null,
primary key(addressID)
);
```

Functional Dependencies

addressID>name, description,

addressid bigint	name character varying (255)	description character varying (255)
12	Depression	So much time invested,...

Stored Procedure

link_transactions() is intended for every transaction input to link and existing output to it.

```
CREATE OR REPLACE FUNCTION link_transactions() RETURNS TRIGGER AS $$
BEGIN
 UPDATE TransactionOutput
 SET InputHash = (SELECT Transaction FROM Bitcoin WHERE BlockId = NEW.TransactionId
 WHERE BitcoinTransactionId = (SELECT BlockId FROM Transaction WHERE Hash = NEW.OutputHash)
  AND index = NEW.OutputIndex;
 RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

Triggers - t_input_linkoutput is intended to update existing Transactions

```
CREATE TRIGGER t_input_linkoutput
BEFORE INSERT ON TransactionInput
  FOR EACH ROW
EXECUTE PROCEDURE link_transactions();
```

TransactionOutput

```
UPDATE TransactionOutput
SET InputHash = data.InputHash,
 InputIndex = data.InputIndex
FROM (SELECT TransactionHash AS InputHash, index AS InputIndex, OutputIndex,
OutputHash
    FROM TransactionInput LEFT JOIN BitcoinTransaction
    ON TransactionInput.BitcoinTransactionId = BitcoinTransaction.BlockId) data
WHERE TransactionOutput.BitcoinTransactionId = (SELECT BlockId FROM BitcoinTransaction
WHERE TransactionHash = data.OutputHash)
AND TransactionOutput.index = data.OutputIndex;
```

Views BlockAggregated

Use this view retrieve aggregated data for a block including the total input, output and transaction fees.

```
DROP VIEW IF EXISTS View_BlockAggregated;
CREATE VIEW View_BlockAggregated AS
SELECT
  Block.BlockId,
  Block.BlockHash,
  Block.PreviousBlockHash,
  Block.BlockTimestamp,
  BlockAggregated.TransactionCount,
  BlockAggregated.TransactionInputCount,
  BlockAggregated.TransactionOutputCount,
FROM Block
INNER JOIN (
  SELECT
    Block.BlockId,
    SUM(1) AS TransactionCount,
    SUM(TransactionInputCount) AS TransactionInputCount,
    SUM(TotalInputBtc) AS TotalInputBtc,
    SUM(TransactionOutputCount) AS Tran
    SUM(TotalOutputBtc) AS TotalOutputB
    --SUM(TransactionFeeBtc) AS Transacti
    SUM(TotalUnspentOutputBtc) AS TotalUnspentOutputBtc
  FROM Block
  INNER JOIN View_TransactionAggregated ON Block.BlockId = View_TransactionAggregated.BlockId
  GROUP BY Block.BlockId
  ) AS BlockAggregated ON BlockAggregated.BlockId = Block.BlockId
```

blockid bigint	blockchainfileid integer	blockversion integer	blockhash character varying (200)	previousblockhash character varying (200)	blocktimestamp date	transactioncount bigint	transactioninputcount numeric	tran num
100	1	1	ab	aabc	2017-07-19	1	1	

transactionoutputcount numeric	totaloutputbtc numeric	totalunspentoutputbtc numeric
1	5.60000000	5.60000000

Sample Query: SELECT * FROM View_BlockAggregated WHERE BlockId = 100

Reports/ Interesting Queries

Query counts the number of blocks in the blockchain..

```
select count(*) from Block
```

	count bigint
1	5

Reports/ Interesting Queries

Selects the transaction with the highest value

SELECT * FROM transactionoutput

ORDER BY value DESC

transactionoutputid bigint	bitcointransactionid bigint	outputindex integer	outputvaluebtc numeric (20,8)	outputscript character varying (300)	index integer	inputhash character varying (64)	inputindex integer
9	27	0	201.30000000	Complete	0	a	0
10	28	0	32.00000000	Complete	0	a	0
8	26	0	14.20000000	Complete	0	a	0
11	29	0	8.20000000	Complete	0	a	0
7	25	0	5.60000000	Complete	0	a	0

Security

```
CREATE ROLE ADMIN;
GRANT ALL ON ALL TABLES IN SCHEMA PUBLIC
TO ADMIN;

CREATE ROLE P_USER;
REVOKE ALL ON ALL TABLES IN SCHEMA PUBLIC
FROM P_USER;
GRANT SELECT ON BlockchainFile, Block, BitcoinTransaction,
TransactionInput, TransactionInputSource, TransactionOutput
```

Implementation Notes – Known Problems – Future Enhancements

Implementation Notes

- With a larger data sample this database would render inefficient not having referential data integrity
- Due to short timeline there is a lack of complex queries(No complex reports) and stored procedures which would've explored this blockchain concept further

Known Problems

- Maintaining referential data integrity
- Not apart of the blockchain network
- Create more stored procedures for a more efficient database
- Moving from a decentralized structured system to relational storage

Future Enhancements

- More Triggers, Stored Procedures
- Implement Queries to determine the time between successive blocks for a given time period
- Implement Queries to determine the distribution of UTXOs, or Unspent Transaction Output
- Creating more schemas concerning unlinked transactions, orphan blocks, and schemas connected the data chain.
- Create a client parallel, updating SQL database to be updated with the new blockdata