Data sources

This assignment relies on *Onion Monero Blockchain Explorer* (OMBE)[1] , an open-source block explorer for monero that provides an API interface to query blocks and transaction information and other network-related information like transaction pool and emission rate.

OMBE has been chosen since it comes the following benefits :

1- Free & Open source
2- Can use community hosted version, such as https://community.xmr.to/explorer/mainnet/, which eliminate the need to download the entire network blockchain.
3- Well-documented, as detailed documentation of the browser, can be obtained from their Github repo.
4- Supports a wide range of API calls
5- Can query all transaction & block fields as represented by the raw monero blockchain, which is not popular among other browsers.
6- Can query on testnet and stagenet networks for monero.

DB Desing

The data received from OMBE API calls are in JSON format, and thus some fields needed to be mapped accordingly to SQL format. For example, the extra field is an array of integers with variable length (not fixed length). As a result, it has been mapped to a separate table where each element of the array represent a row entry, and each row entry is linked with the Transaction through the transaction_id field. Similar behavior is done for table vin_txs, vout_txs, where the number of inputs and outputs in a single transaction is variable, and they are represented as an array with variable length in the JSON format. Also, the key offsets were treated similarly to extra, vin_txs, and vout_txs. Figure 1 represents an overview of the SQL schema; further, Table 1 shows a description of the fields, and Table 2 shows a description of the relationship between the tables.

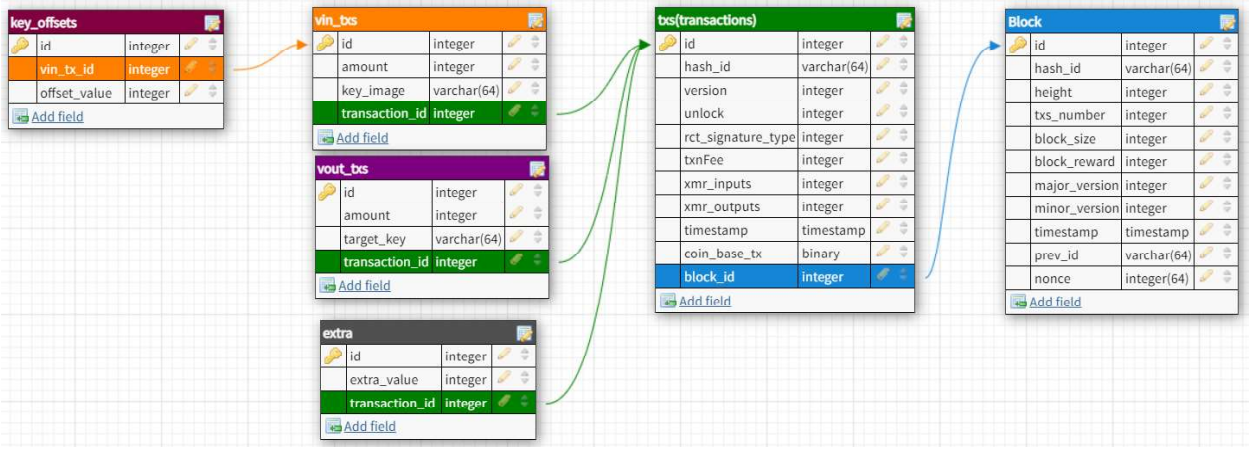

Figure 1 DB_schema

1- https://github.com/moneroexamples/onion-monero-blockchain-explorer

Table 1 Field description

Block	
Field name	**Description**
Id	An autogenerated integer that acts as a primary key of the Table
Hash_id	The hash of the block
Height	The Height of the block represents the blockchain's position, as the first block takes a Height of 0, the second takes 1, and so on.
Txs_number	The number of transactions inside the block
Block_size	The size of the block in the blockchain in bytes.
Block_rewards	The summation of all xmr rewarded for the miner includes transactions fees.
Major_version	protocol version that was used in the block verification.
Minor_version	Theoretically deprecated and not used, but since it occupies low space developer did not remove it.
Timestamp	Block UTC timestamp, as recorded by the miner.
Prev_id	The block id for the previous block in the blockchain.
Nonce	The nonce used by the miner to solve the POW puzzle.
Txs (short for transactions)	
Id	An autogenerated integer that acts as a primary key of the Table
Hash_id	The hash of the block
Version	The algorithm followed by the Transaction, value 2 stands RingCT, and 1 is used on transactions before RingCT implementation
Unlock	A time lock that prevents a transaction from being spent. It is either block height or UNIX time. For the miner reward, the unlock_time is set to 60 blocks after the currently mined block (120 minutes)
Rct_signature_type	Type of the transactions as following : 0- RCTTypeNull, for Miner reward transaction 1- RCTTypeFull, Deprecated for usage RingCT 2- RCTTypeSimple, Deprecated for usage RingCT 4- RCTTypeBulletproof2, current support version of the signature. (it is not a mistake. It has the value 4; there is no option for value 3). Note that if the version is not 2, this field will be zero, but it does not mean it is a miner transaction since only transactions with version 2 use RingCT.
txnFee	The transaction fee
Xmr_inputs	Sum of all input transactions.
Xmr_outputs	Sum of all output transactions.
Timestamp	Transaction UTC stamp, as recorded by the miner.
Coin_base_tx	1 for coin_base trnsaction, and 0 for nomal transactions.
Block_id	The block ID where this Transaction belongs.
Vin_txs (short for input transactions)	
Id	An autogenerated integer that acts as a primary key of the Table
Amount	Amount of the Transaction, a deprecated field used for type 1 transaction (before RingCT)
Key_image	Used by the verifier to find ring member and make sure they are legitimate
Transcation_id	The Transaction where this key_image is spent.

Vout_txs (short for output transactions)	
Id	An autogenerated integer that acts as a primary key of the Table
Amount	Amount of the Transaction, deprecated field used for type 1 transaction (before RingCT)
Target_key	Onetime destination key (stealth address)
Transaction_id	The Transaction that is directred to the target_key.
extra	
Id	An autogenerated integer that acts as a primary key of the Table
Extra_value	Additional information for the Transaction, including the public key(R value) for the Transaction, padding, payment_id, and other information.
Transaction_id	The Transaction that this extra field belongs to.
Key_offsets	
Id	An autogenerated integer that acts as a primary key of the Table
Vin_tx_id	The vin_tx that this offset field belongs to.
Offset_value	Used by the verifier to find ring member and make sure they are legitimate

Table 2 relationship description

Table	Field	description
Txs	Block_id	Many to one relationship with blocks Table, where each Transaction has to belong only one block, and each block contains at minimum 1 transaction (coin_base).
Vin_txs	Transcation_id	Many to one relationship with the transaction table, where each vin_tx belongs to only one tx and tx, contains minimum 1 vin_tx, except for coinbase tx, since the xmrs was generated and not received from previous txs.
Vout_txs	Transaction_id	Many to one relationship with transaction table, where each vout_tx belongs to only one tx and tx contains at minimum 1 vout_tx.
E	Transaction_id	Many to one relationship with transaction table, where each extra value belongs to only one tx and tx contains at minimum 1 extra value.
Key_offsets	Vin_tx_id	Many to one relationship with vin_txs, where each offset value belongs to only one vin_tx and vin_tx contain at minimum 1 offset value.

Some information have not been included by the script,Table 3 summarize them with their reasons.

Table 3 Ignored information

Name	Reason
is orphan (for block)	Orphan blocks are not queryable, as all the blocks that the script can reads are not orphans.
Depth (for block)	Depend on the time of execution, and thus has a concise expiry date, since a new block is added every 2 minutes, which makes the life span of this field 2 minutes only
difficulty (for block)	Not provided by OMBE

EcdhInfo (Transaction)	Information regarding the Elliptic curve Diffie-Hellman exchange process and holds little value for the regular user
All fields inside Rctsig_prunable (transaction)	Since it contains variable used in the range proof and signature algorithms, it holds little value for the normal user.

Program flowchart

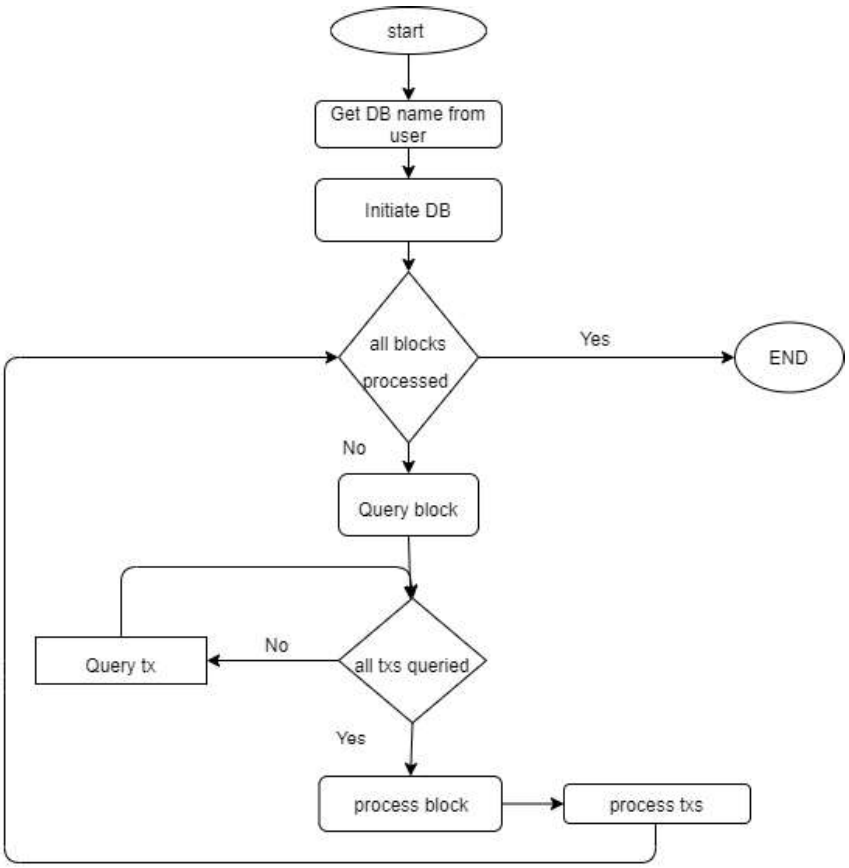

Figure 2 program flowchart

In this section, we explain the operation done by the program. Figure 2 shows a flowchart of the program.

The program starts by requesting some information from the user: the DB file name and the block range (lowest and highest block height). Then the program initiates the DB using the schema as described earlier. Then for each block, the program will query the block information from OMBE, then for each Transaction inside the block, the transaction information is queried from OMBE. After querying all the needed information, the program starts parsing the JSON objects retrieved from OMBE to SQLlite format explained earlier. Then, all the data is inserted in the DB, and the same process is repeated for all the other blocks. Table 4 summarizes the functions in the program with their parameters, return types, and description.

Table 4 functions and their description

Function name	Pararm	Returns	Description
create_connection	DB file name	connection object	Creates DB database
create_table	Connection object, table SQL sting	None	Creates a Table from SQL string
insert_block	Connection object, block object	Block Id	Insert a new block into blocks Table
insert_transaction	Connection object, transaction object	Transaction Id	Insert a new Transaction into txs Table
insert_vin_tx	Connection object, vin_tx object	Vin_tx Id	Insert a new vin_tx into vin_txs Table
insert_vin_key_offset	Connection object, key_offsets array, vin_tx_id	None	Insert all elements of key offset array to key offset Table
insert_vout_tx	Connection object, vout_tx object	Vout_tx Id	Insert a new vout_tx into vout_txs Table
insert_extra	Connection object, extra array, tx_id	None	Insert all elements of an extra array to extra Table
init_db	DB Name	None	Creates DB file by calling **create_connection,** then creates the DB tables by calling **create_table** for each Table.
process_block	Connection object, block_height	None	Query the block and block's Transaction, respectively, uses **insert_*** functions to insert queried data into the SQLite DB referenced by the connection object.
extract_block_data_to_sqlite_DB	None	None	Takes user input from the command line (DB name and block range) and then pass those input to **init_db** and **process_block** respectively.
main	None	None	Handles the main menu and user options. Although the script currently has only one functionality, this design choice supports future expandability.

Attached files

Included in the project files 2 DB files, as following:

- first250.db: a DB containing the first 250 blocks in monero network. Since This was very early in the network, the first 250 blocks contains only 1 transaction.
- Last20.db : a DB containing the last added 20 blocks, the purspoe is to have a DB with a lot of transaction per block.