Block Crawler

Problem Description

Analyzing and querying blockchain data is difficult. For all the novelty and innovation that blockchain technology offers in regards to how people transact, the underlying data structure wasn't designed or optimized to retrieve aggregated insights efficiently. For example, consider the use case of measuring the total volume of a token from the past quarter. One of the ways to perform this would be to export all the transactions within that time frame, transform and load them into a relational database and perform a query based on the transactions involving that token. This is the concept of blockchain indexing, which numerous companies and projects have built infrastructure to do.

Task

The task to complete is comprised of two parts:

Part 1

Write a Python program that retrieves Ethereum Mainnet transactions within a given block range and persists them to a database. You're free to use SQLite or a more sophisticated relational database, like PostgreSQL. It's expected that your program will take in at least three command line inputs:

1. A JSON-RPC endpoint to call an Ethereum client (e.g. `https://rpc.quicknode.pro/key`)
2. The path of the SQLite file to write to or a connection URI (e.g. `db.sqlite3` or `postgresql://user:password@localhost:5432/database`)
3. A block range, formatted as "{start}-{end}", (e.g. `200-300`)

Running your program should look similar to the following:

```
$ python block_crawler.py \
      https://rpc.quicknode.pro/key \
      db.sqlite3 \
      200-300
```

Part 2

Write a simple Python script or raw SQL that queries the populated database for the block that had the largest volume of ether transferred between 2024-01-01 00:00:00 and 2024-01-01 00:30:00. The query should return the block number and the total volume transferred for that block. For populating the database, use a block range from 18908800 to 18909050 (this corresponds with the timeframe above).

Useful Details

It's not expected that you'll be fully knowledgeable on Web3/blockchain technology or the underlying specifics of the Ethereum protocol, so the following is to provide some instruction to help you get started.

JSON-RPC Endpoint

All interaction with the Ethereum blockchain is done via JSON-RPC to Ethereum clients over HTTP or WebSocket. For example, there are remote procedure calls for sending a transaction or retrieving the latest block. An overview of the full JSON-RPC specification can be found in the documentation written by the Ethereum organization but the relevant calls are `eth_getBlockByNumnber` and `eth_blockNumber`:

- `eth_getBlockByNumber`
  Returns information about a block by number
  Parameters:
  - Block: `string` – block number (hexadecimal string) or block tag ("finalized", "safe", "latest", "pending")
  - Hydrated transactions: `boolean` – whether to include full transactions or only their hashes

  Result:
  - Block object or `null`

  Example (using `curl`):

```
$ curl https://docs-demo.quiknode.pro/ \
     -X POST \
     -H "Content-Type: application/json" \
     --data
     '{"method":"eth_getBlockByNumber","params":["0xc5043f
     ",true],"id":1,"jsonrpc":"2.0"}'
```

- `eth_blockNumber`
  Returns the latest block number of the blockchain
  Result:
  - Block number: `string` (hexadecimal string)

  Example (using `curl`):

```
$ curl https://docs-demo.quiknode.pro/ \
     -X POST \
     -H "Content-Type: application/json" \
     --data
     '{"method":"eth_blockNumber","params":[],"id":1,"json
     rpc":"2.0"}'
```

Getting Access to a JSON-RPC Endpoint

There are multiple Ethereum node providers, such as QuickNode, that offer free access to a JSON-RPC endpoint by creating an account.

To access one through QuickNode:

1. Go to QuickNode's Core RPC API.
2. Click "Create a free account" and sign-up for an account.
3. You'll then have access to QuickNode's dashboard. Click "Endpoints" on the left-side tab and click "Create endpoint."
4. Select "Ethereum" as the chain. After clicking "Continue", select "Mainnet." After clicking "Continue" a second time, click "Create Endpoint."
5. This will provide you with endpoints for HTTPS and WSS providers at no cost.

There are also publicly available endpoints, some of which can be found on the ChainList page for Ethereum. You're free to use one of the endpoints listed but they aren't guaranteed to be reliable or have complete Ethereum history like QuickNode's.

Notes on Object Models

Block and Transaction objects defined in the specification contain a lot of properties that are not important for the purposes of this task. Please don't spend time incorporating them into your database schema. Below are the objects as defined by the specification, with a subset of properties you may want to consider focusing on:

Block

- **hash** (hexadecimal string)
- **number** (hexadecimal string)
- **timestamp** (hexadecimal string) – Unix timestamp of when the block was collated
- **transactions** – list of transaction objects (if Hydrated Transaction is **true** in the `eth_getBlockByNumber` RPC Call)

Transaction

- **hash** (hexadecimal string)
- **blockHash** (hexadecimal string)
- **blockNumber** (hexadecimal string)
- **from** (hexadecimal string) -- address that sent the transaction
- **to** (hexadecimal string) -- address that received the transaction
- **value** (hexadecimal string) -- amount of **wei** sent with the transaction
    - Wei is a denomination of Ether, the native token on Ethereum. `1 Ether = 10e-18 Wei`
    - Performing an integer conversion can result in values greater than a signed 8-byte integer

Guidelines

- You're free to use any 3rd-party packages that you find helpful.
- You're free to use whichever version of Python 3 you'd like.
- You're free to use SQLite or a more sophisticated relational database, like PostgreSQL.
- It's expected you'll spend roughly 3-4 hours to complete this task. We want to be respectful of your time – please do not feel obligated to work on it any longer than that.
    - No piece of software is perfect – feel free to document functionality you would add or do differently.
- You'll have 1 week to complete your solution from the time you receive this document. If we don't receive your solution within 1 week and have received no communication, we will assume you are no longer interested in the role. Please email careers@relayer.tech if you need more time.
- If you end up with time left, feel free to include tests or address performance concerns.

Deliverables

Please email careers@relayer.tech with either a `.zip` file of your code or a link to a public repository. In the subject line, please include your full name followed by "Technical Challenge" (e.g. "Vitalik Buterin Technical Challenge").

- Please include all the code that makes up your Python project, including your SQL query.
- Please include a `.txt` file with the resulting block number and total volume from your SQL query in part 2 of the task.
- Please include a `README` file with any information you think is relevant to running and understanding your code. If interested, the quality/detail of your `README` file can be supplemented with a short video (<5min), such as a Loom, where you share your screen and explain your solution.

Expectations

After completing this task, your solution should comprise the following:

- ☐ A runnable program that takes in the required inputs and populates the intended database
- ☐ Consideration for error handling and input validation
- ☐ An appropriate database schema
- ☐ Code that is well structured, readable, and documented