SubQuery

Making the world's decentralised data more accessible.

Lab Exercise Guide

Table of Contents

Introduction

In this lab, students will have the opportunity to become familiar with SubQuery with some hands-on experience creating a SubQuery project to list all transactions for a given address. This project will use the subql CLI to create an empty project shell, and then code will be provided to query Polkadot mainnet. A Docker environment will be used to run this example for simplicity.

Pre-requisites

You will require the following:

- NPM package manager
- SubQuery CLI (@subql/cli)
- Docker

Package manager

Run the following command in your terminal to install the latest version of node. Node v12 or higher is required.

```
brew update
brew install node
node -v
v18.2.0
```

SubQuery CLI

Install the latest version of the subql cli:

```
npm install -g @subql/cli
subql -v
@subql/cli/1.0.1 darwin-x64 node-v18.2.0
```

Docker

Please visit https://docs.docker.com/get-docker/ for instructions on how to install Docker for your specific operating system.

Exercise 1: Listing all transactions for a given address

High level steps

1. Initialise a project
2. Define the shape of your data
3. Update your manifest file
4. Write your mapping handler
5. Generate, build and deploy your code
6. Run a query

Detailed steps

Step 1: Initialise your project

The first step in creating a SubQuery project is to create a project with the following command:

```
$ subql init
Project name [subql-starter]: subql-list-transactions
? Select a network family Substrate
? Select a network Polkadot
? Select a template project subql-starter     Starter project for
subquery
RPC endpoint: [wss://polkadot.api.onfinality.io/public-ws]:
Git repository [https://github.com/subquery/subql-starter]:
Fetching network genesis hash... done
Author [Ian He & Jay Ji]: Sean
Description [This project can be use as a starting po...]:
Version [1.0.0]:
License [MIT]:
Preparing project... done
subql-list-transactions is ready
```

Note that any text in the square brackets are the default values that will be used if nothing is provided.

This creates a directory scaffold saving you time.

Step 2: Defining the “shape” of our data

Here we want to create a single entity - Transfer.

We design the Transfer entity to contain the amount, and the to and from address.

```
type Transfer @entity {
  id: ID!
  amount: BigInt
  blockNumber: BigInt
  to: String!
  from: String!
}
```

Step 3: Update the manifest file (aka project.yaml)

The initialisation command also pre-creates a sample manifest file and defines 3 handlers. Copy the paste the manifest file from below.

```
specVersion: 1.0.0
name: subql-list-transactions
version: 1.0.0
runner:
  node:
    name: '@subql/node'
    version: '>=1.0.0'
  query:
    name: '@subql/query'
    version: '*'
description: >-
  This project can be use as a starting point for developing your
SubQuery
  project
repository: 'https://github.com/subquery/subql-starter'
schema:
  file: ./schema.graphql
network:
  chainId:
'0x91b171bb158e2d3848fa23a9f1c25182fb8e20313b2c1eb49219da7a70ce90c3'
  endpoint: 'wss://polkadot.api.onfinality.io/public-ws'
  dictionary:
'https://api.subquery.network/sq/subquery/polkadot-dictionary'
dataSources:
  - kind: substrate/Runtime
    startBlock: 1
    mapping:
      file: ./dist/index.js
      handlers:
        - handler: handleTransfer
          kind: substrate/EventHandler
          filter:
            module: balances
            method: Transfer
```

Here we keep the event handler and the filter as well.

Step 4: Write your mappings file

Copy the code from below to your mappingHandler.ts file.

```
import {SubstrateEvent} from "@subql/types";
import {Transfer} from "../types";
import {Balance} from "@polkadot/types/interfaces";

export async function handleTransfer(event: SubstrateEvent):
Promise<void> {
    // Get data from the event
    // The balances.transfer event has the following payload \[from, to,
value\]
    const from = event.event.data[0];
    const to = event.event.data[1];
    const amount = event.event.data[2];

    // Create the new transfer entity
    const transfer = new Transfer(
        `${event.block.block.header.number.toNumber()}-${event.idx}`,
    );
    transfer.blockNumber = event.block.block.header.number.toBigInt();
    transfer.from = from.toString();
    transfer.to = to.toString();
    transfer.amount = (amount as Balance).toBigInt();
    await transfer.save();
}
```

Step 5: Generate, build and deploy

Run the following commands:

```
yarn install
yarn codegen
yarn build
Docker-compose pull && docker-compose up
```

Step 6: Run a query

Run the following query:

```
query {
    transfers (first:3 orderBy: AMOUNT_DESC ) {
        nodes {
          id,
          to,
          from,
          amount,
          blockNumber
        }
    }
}
```

You should get the following results:

```
{
  "data": {
    "transfers": {
      "nodes": [
        {
          "id": "645657-4",
          "to": "13SkL2uACPqBzpKBh3d2n5msYNFB2QapA5vEDeKeLjG2LS3Y",
          "from": "13yk62yQYctYsRPXDFvC5WzBtanAsHDasenooLAxKvf5bNkK",
          "amount": "102000000000000000",
          "blockNumber": "645657"
        },
        {
          "id": "645697-3",
          "to": "13SkL2uACPqBzpKBh3d2n5msYNFB2QapA5vEDeKeLjG2LS3Y",
          "from": "12WLDL2AXoH3MHr1xj8K4m9rCcRKSWKTUz8A4mX3ah5khJBn",
          "amount": "99000000000000000",
          "blockNumber": "645697"
        },
        {
          "id": "303284-59",
          "to": "1vTfju3zruADh7sbBznxWCpircNp9ErzJaPQZKyrUknApRu",
          "from": "15j4dg5GzsL1bw2U2AWgeyAk6QTxq43V7ZPbXdAmbVLjvDCK",
          "amount": "90000000000000000",
          "blockNumber": "303284"
        }
      ]
    }
  }
}
```

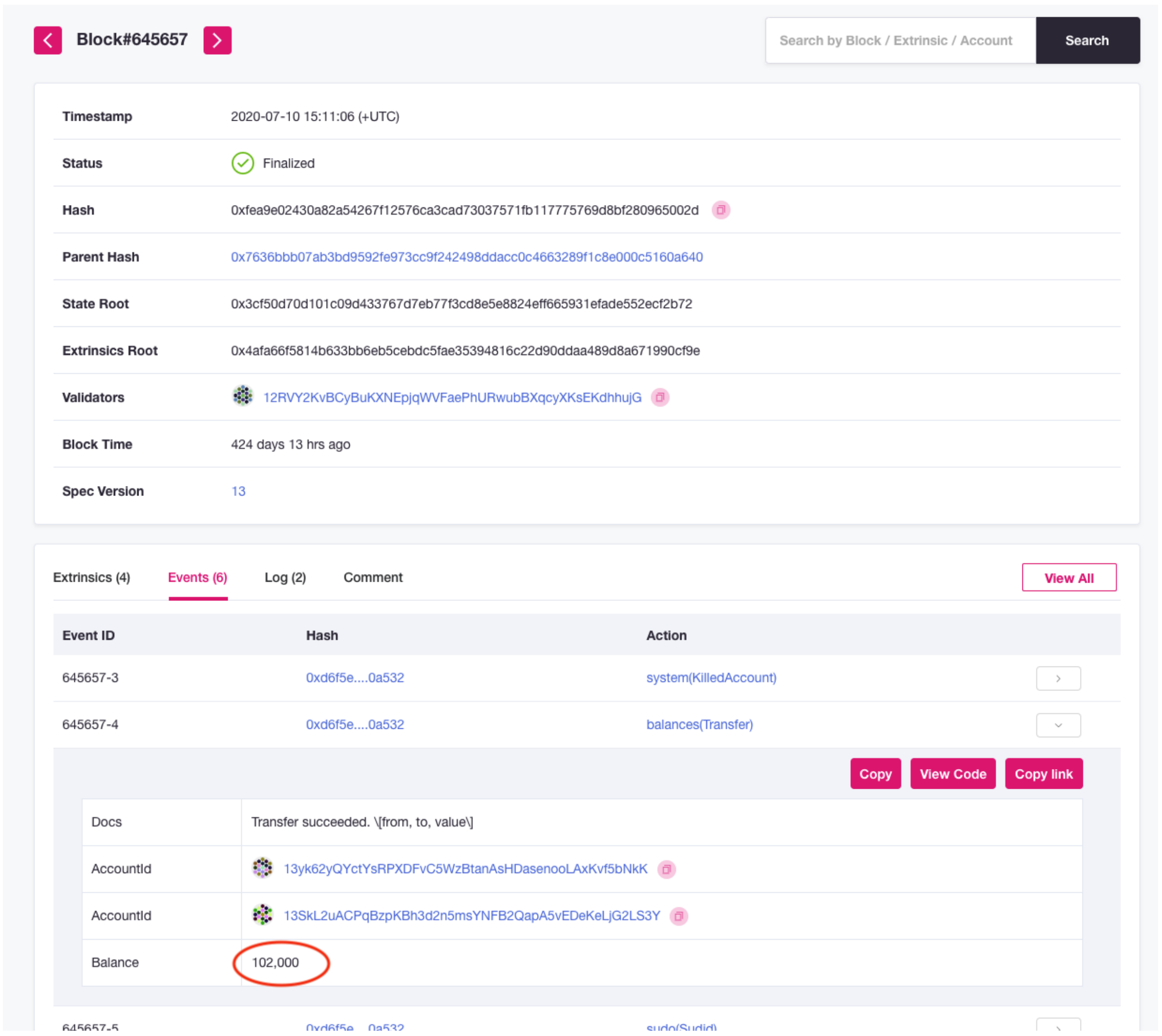
This allows us to see that address "13SkL2uACPqBzpKBh3d2n5msYNFB2QapA5vEDeKeLjG2LS3Y" has received 2 transactions of 102,000 and 99,000. Using this address, let's query for all amounts at this address:

```
query {
    transfers (filter: {to:
{in:"13SkL2uACPqBzpKBh3d2n5msYNFB2QapA5vEDeKeLjG2LS3Y"}}){
        nodes {
          id,
          to,
          from,
          amount,
          blockNumber
        }
    }
}
```

Below we can see 2 transactions at the above address.

```
{
  "data": {
    "transfers": {
      "nodes": [
        {
          "id": "645657-4",
          "to": "13SkL2uACPqBzpKBh3d2n5msYNFB2QapA5vEDeKeLjG2LS3Y",
          "from": "13yk62yQYctYsRPXDFvC5WzBtanAsHDasenooLAxKvf5bNkK",
          "amount": "102000000000000000",
          "blockNumber": "645657"
        },
        {
          "id": "645697-3",
          "to": "13SkL2uACPqBzpKBh3d2n5msYNFB2QapA5vEDeKeLjG2LS3Y",
          "from": "12WLDL2AXoH3MHr1xj8K4m9rCcRKSWKTUz8A4mX3ah5khJBn",
          "amount": "990000000000000",
          "blockNumber": "645697"
        }
      ]
    }
  }
}
```

Cross check with Polkadot subscan:
https://polkadot.subscan.io/block/645657?tab=event



Block#645657

Timestamp	2020-07-10 15:11:06 (+UTC)
Status	Finalized
Hash	0xfea9e02430a82a54267f12576ca3cad73037571fb117775769d8bf280965002d
Parent Hash	0x7636bbb07ab3bd9592fe973cc9f242498ddacc0c4663289f1c8e000c5160a640
State Root	0x3cf50d70d101c09d433767d7eb77f3cd8e5e8824eff665931efade552ecf2b72
Extrinsics Root	0x4afa66f5814b633bb6eb5cebdc5fae35394816c22d90ddaa489d8a671990cf9e
Validators	12RVY2KvBCyBuKXNEpjqWVFaePhURwubBXqcyXKsEKdhhujG
Block Time	424 days 13 hrs ago
Spec Version	13

Event ID	Hash	Action
645657-3	0xd6f5e....0a532	system(KilledAccount)
645657-4	0xd6f5e....0a532	balances(Transfer)

Docs	Transfer succeeded. \[from, to, value\]
AccountId	13yk62yQYctYsRPXDFvC5WzBtanAsHDasenooLAxKvf5bNkK
AccountId	13SkL2uACPqBzpKBh3d2n5msYNFB2QapA5vEDeKeLjG2LS3Y
Balance	102,000

645657-5	0xd6f5e....0a532	sudo(Sudid)