Efficient data query through dense subgraphs

with OpenZeppelin and TheGraph

Hadrien Croubois
hadrien@openzeppelin.com
@amxx

Graph density

Wikipedia

Not logged in Talk Contributions Create account Log in

Article Talk Read View source View history Search Wikipedia

Ethereum

From Wikipedia, the free encyclopedia

Ethereum is a decentralized, open-source blockchain with smart contract functionality. **Ether** (**ETH** or Ξ) is the native cryptocurrency of the platform. After Bitcoin, it is the second-largest cryptocurrency by market capitalization.[1] Ethereum is the most actively used blockchain.[2][3]

Ethereum was proposed in 2013 by programmer Vitalik Buterin. In 2014, development was crowdfunded, and the network went live with an initial supply of 72 million coins on 30 July 2015.[4][5][6][7] The platform allows developers to build and operate decentralized applications that users can interact with.[8][9] Decentralized finance (DeFi) applications provide a broad array of financial services without the need for typical financial intermediaries, such as brokerages, exchanges, or banks, allowing cryptocurrency users to borrow against their holdings or lend them out for interest.[10][11] Ethereum also allows for the creation and exchange of NFTs, which are non-interchangeable tokens connected to digital works of art or other real-world items and sold as unique digital property. Additionally, many other cryptocurrencies operate as ERC-20 tokens on top of the Ethereum blockchain and have utilized the platform for initial coin offerings.

Ethereum has started implementing a series of upgrades called Ethereum 2.0, which includes a transition to proof of stake and aims to increase transaction throughput using sharding.[12][13]

Contents [hide]

Ethereum

Original author(s)	Vitalik Buterin Gavin Wood
Developer(s)	Ethereum Foundation, Hyperledger, Nethermind, OpenEthereum, EthereumJS
Initial release	30 July 2015; 5 years ago
Stable release	Berlin / 15 April 2021; 2 months ago
Development status	Active
Software used	EVM 1 Bytecode
Written in	Go, Rust, C#, C++, Java, Python
Operating system	Cross-platform
Platform	x86-64, ARM
Size	300 GB (2020-03)
Type	Distributed computing
License	Open-source licenses
Active hosts	10,335 (2021-01)
Website	ethereum.org

History

Wikipedia The Free Encyclopedia

Main page
Contents
Current events
Random article
About Wikipedia
Contact us
Donate

Contribute
Help
Learn to edit
Community portal
Recent changes
Upload file

Tools
What links here
Related changes
Special pages
Permanent link
Page information
Cite this page
Wikidata item

Print/export
Download as PDF
Printable version

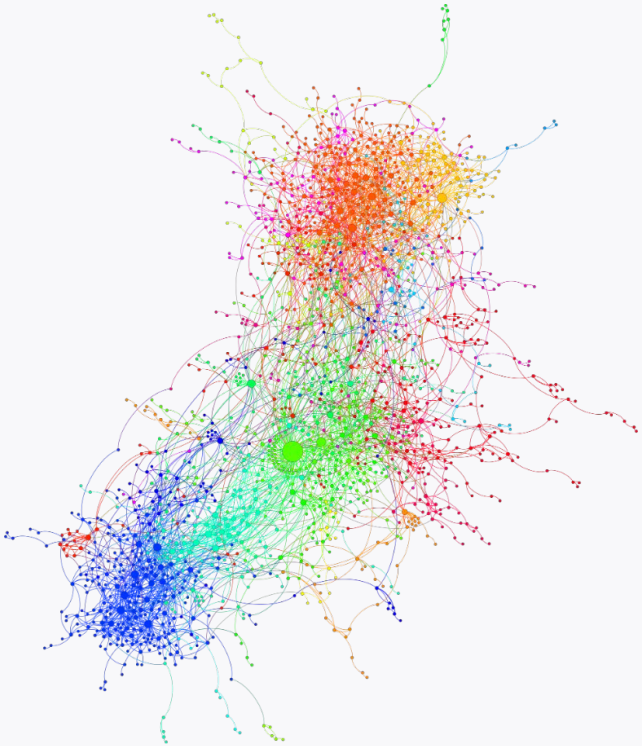
In other projects
Wikimedia Commons

Languages
Català
Deutsch
Español
Euskara
Français
Italiano
Nederlands
Português
43 more

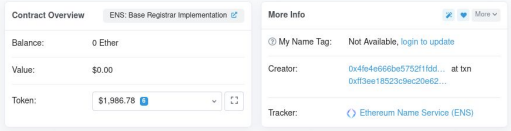
Edit links

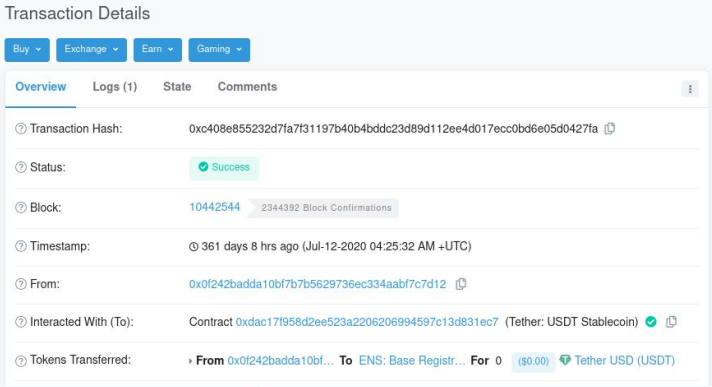
Information is not about the individual pieces of data

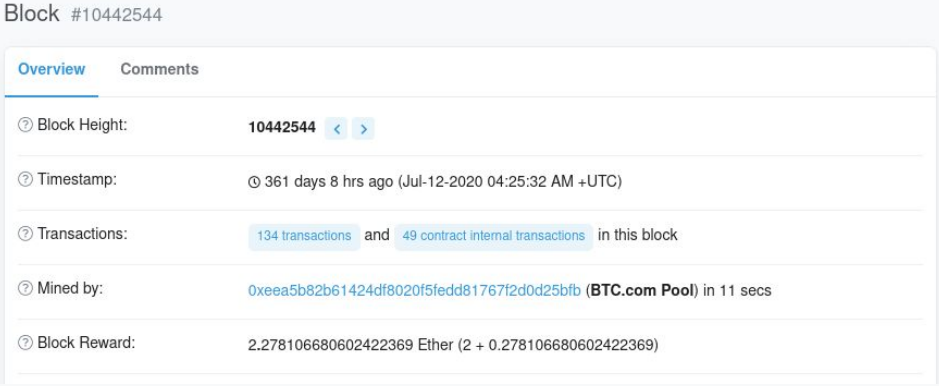
It's about the links between them

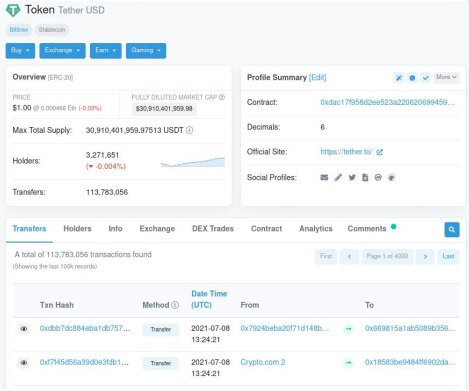


Etherscan


Contract Overview
ENS: Base Registrar Implementation
Balance:
0 Ether
Value:
$0.00
Token:
$1,986.78
More Info
My Name Tag:
Not Available, login to update
Creator:
Tracker:
Ethereum Name Service (ENS)
Transactions
Internal Txns
Erc20 Token Txns
Erc721 Token Txns
Contract
Events
Analytics
Comments
Txn Hash
Method
Block
Date Time (UTC)
From


Transaction Details
Buy
Exchange
Earn
Gaming
Overview
Logs (1)
State
Comments
Transaction Hash:
0xc408e855232d7fa7f31197b40b4bddc23d89d112ee4d017ecc0bd6e05d0427fa
Status:
Success
Block:
10442544
2344392 Block Confirmations
Timestamp:
361 days 8 hrs ago (Jul-12-2020 04:25:32 AM +UTC)
From:
0x0f242badda10bf7b7b5629736ec334aabf7c7d12
Interacted With (To):
Contract 0xdac17f958d2ee523a2206206994597c13d831ec7 (Tether: USDT Stablecoin)
Tokens Transferred:
From 0x0f242badda10bf... To ENS: Base Registr... For 0 ($0.00) Tether USD (USDT)

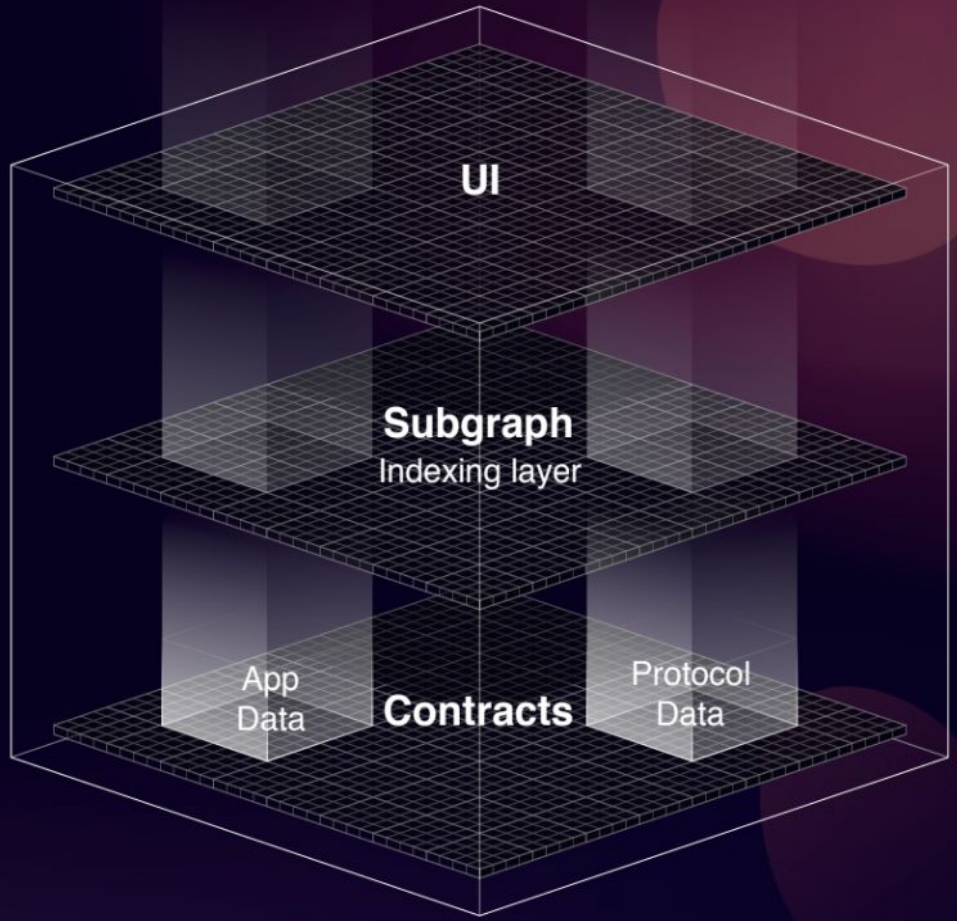

Block #10442544
Overview
Comments
Block Height:
10442544
Timestamp:
361 days 8 hrs ago (Jul-12-2020 04:25:32 AM +UTC)
Transactions:
134 transactions and 49 contract internal transactions in this block
Mined by:
0xeea5b82b61424df8020f5fedd81767f2d0d25bfb (BTC.com Pool) in 11 secs
Block Reward:
2.278106680602422369 Ether (2 + 0.278106680602422369)


Token Tether USD
Overview [ERC-20]
Max Total Supply:
30,910,401,959,97513 USDT
Holders:
3,271,651
Transfers:
113,783,056
Profile Summary [Edit]
Contract:
Decimals:
6
Official Site:
https://tether.to/
Social Profiles:
Transfers
Holders
Info
Exchange
DEX Trades
Contract
Analytics
Comments
A total of 113,783,056 transactions found
Txn Hash
Method
Date Time (UTC)
From
To

Graph density for subgraphs

We've become a default part of the stack

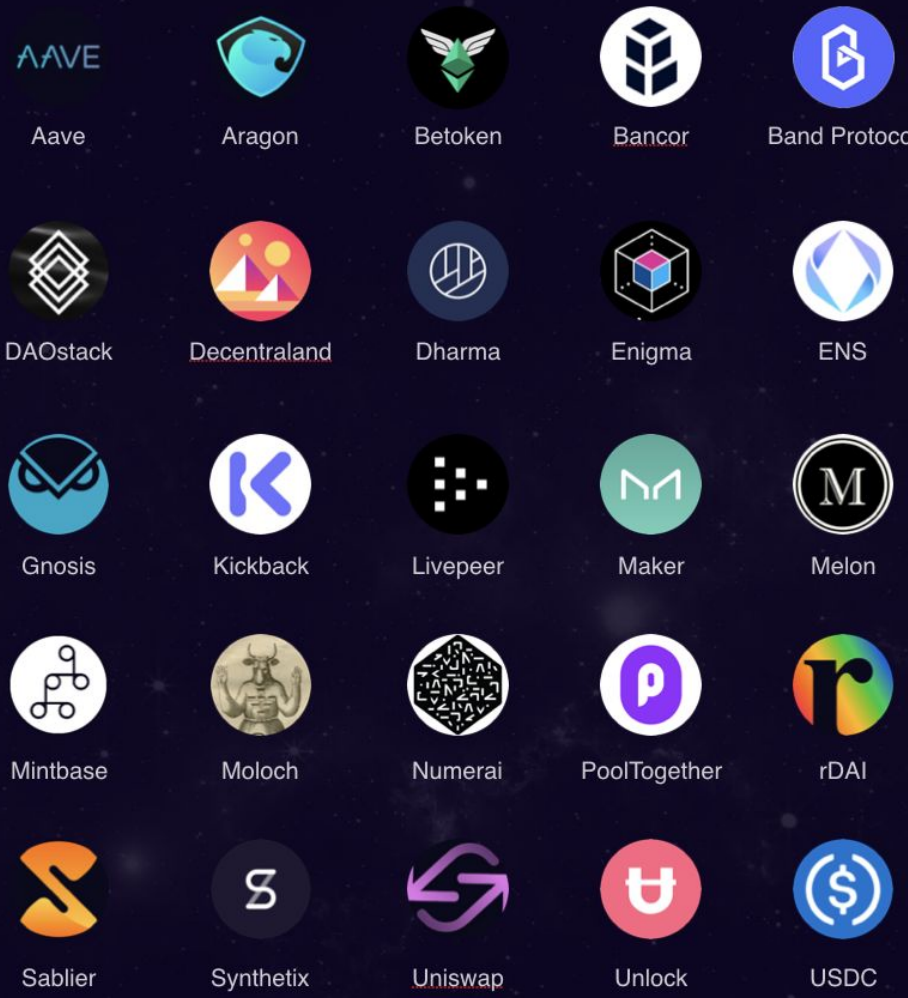
A subgraph defines how to efficiently index data in a deterministic way.



The Graph is Powering DeFi

DeFi dApps are querying on-chain data:

- Trades and exchange volumes
- Total borrowed, supplied, staked
- Asset prices, wallet balances
- Yield farming
- Total Value Locked



Subgraph A: *How would you retrieve the top 100 holders?*

```
TokenContract

id: ID!
address: Bytes!
totalSupply: BigInt!
owner: Bytes!
tokenHolders: [String!]
```

```
TokenHolder

id: ID!
contract: Bytes!
address: Bytes!
balance: BigInt!
```

```
TokenTransfer

id: ID!
txHash: Bytes!
contract: Bytes!
from: Bytes!
to: Bytes!
value: BigInt!
```

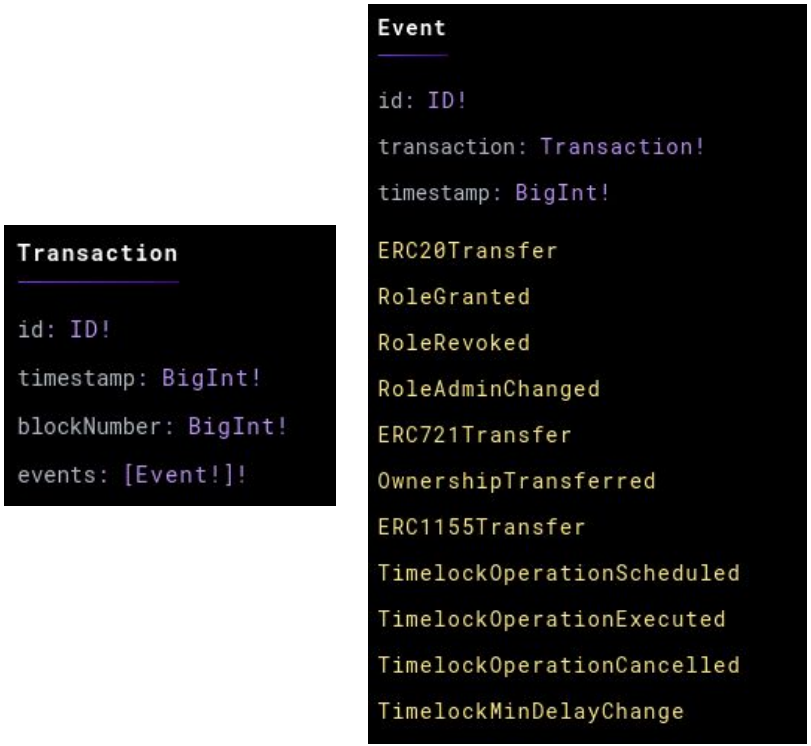
Subgraph B: *How would you retrieve the top 100 holders?*

```
ERC20Contract

id: ID!
asAccount: Account!
name: String
symbol: String
decimals: Int!
totalSupply: ERC20Balance!
balances: [ERC20Balance!]!
approvals: [ERC20Approval!]!
transfers: [ERC20Transfer!]!
```

```
ERC20Balance

id: ID!
contract: ERC20Contract!
account: Account
value: BigDecimal!
valueExact: BigInt!
transferFromEvent: [ERC20Transfer!]!
transferToEvent: [ERC20Transfer!]!
```

```
ERC20Transfer

Event

id: ID!
transaction: Transaction!
timestamp: BigInt!
contract: ERC20Contract!
from: Account!
fromBalance: ERC20Balance
to: Account!
toBalance: ERC20Balance
value: BigDecimal!
valueExact: BigInt!
```

Subgraph B: *What about other ERC20 transfers batched in the same transaction?*



Subgraph B: *What about other tokens held by the same user?*

```
Account

id: ID!
asERC20: ERC20Contract
ERC20balances: [ERC20Balance!]!
ERC20approvalsOwner: [ERC20Approval!]!
ERC20approvalsSpender: [ERC20Approval!]!
ERC20transferFromEvent: [ERC20Transfer!]!
ERC20transferToEvent: [ERC20Transfer!]!
asAccessControl: AccessControl
membership: [AccessControlRoleMember!]!
roleGranted: [RoleGranted!]!
roleGrantedSender: [RoleGranted!]!
roleRevoked: [RoleRevoked!]!
roleRevokedSender: [RoleRevoked!]!
asERC721: ERC721Contract
ERC721tokens: [ERC721Token]!
```

```
ERC721operatorOwner: [ERC721Operator]!
ERC721operatorOperator: [ERC721Operator]!
ERC721transferFromEvent: [ERC721Transfer]!
ERC721transferToEvent: [ERC721Transfer]!
asOwnable: Ownable
ownerOf: [Ownable!]!
ownershipTransferred: [OwnershipTransferred!]!
asERC1155: ERC1155Contract
ERC1155balances: [ERC1155Balance!]!
ERC1155operatorOwner: [ERC1155Operator!]!
ERC1155operatorOperator: [ERC1155Operator!]!
ERC1155transferFromEvent: [ERC1155Transfer!]!
ERC1155transferToEvent: [ERC1155Transfer!]!
ERC1155transferOperatorEvent: [ERC1155Transfer!]!
asTimelock: Timelock
timelockedCalls: [TimelockCall!]!
```

Guidelines for building good subgraphs

- Create entities for high level concepts (tokens, balances, authorizations, loans)
- Create entities for low level objects (addresses, events, transactions)
- Provide as many crosslink as possible between entities

- Design your contracts so that everything can be indexed using only events.

Building complex queries with dense subgraphs

```
{
  erc20Contract(id: "<token-address-in-lowercase>") {
    name
    symbol
    totalSupply { value }
    balances(
      first: 100,
      orderBy: value,
      orderDirection: desc,
      where: { account_not: null }
    ) {
      account { id }
      value
    }
  }
}
```

ERC20 details, including total supply, and balances of the 100 biggest holders

Building complex queries with dense subgraphs

```
{
  account(id: "<user-address-in-lowercase>") {
    ERC20balances {
      contract { name, symbol, decimals }
      value
      transferFromEvent {
        transaction { id, timestamp, blockNumber }
        to { id }
        value
      }
      transferToEvent {
        transaction { id, timestamp, blockNumber }
        from { id }
        value
      }
    }
  }
}
```

All ERC20 balances and corresponding transfers, with details about the tokens, for a account (user)

Building complex queries with dense subgraphs

```
{
  erc20Contract(id: "<erc20-with-accesscontrol-address-in-lowercase>") {
    name
    symbol
    decimals
    totalSupply { value }
    asAccount {
      asAccessControl {
        admins: roles(where: { role: "0x0000000000000000000000000000000000000000000000000000000000000000" }) {
          members {
            account {
              address: id
              balance: ERC20balances(where: { contract: "<erc20-with-accesscontrol-address-in-lowercase>" }) {
                value
              }
            }
          }
        }
      }
    }
  }
}
```

ERC20 balances of all administrators or the access-control powered token

Subgraph Studio | My Subgraphs | API Keys | Billing | 0x7859 — 0ce525

My Subgraphs

Create a Subgraph

Search 9 Subgraphs

ERC1155

MAINNET

ERC721

MAINNET

Compound Governor Bravo

DEPLOYED

Argent

RINKEBY

POAP Mainnet

RINKEBY

POAP xDai

RINKEBY

usdt

RINKEBY

Meta Governor Mainnet

DEPLOYED

Meta Governor Rinkeby

DEPLOYED

OpenZeppelin subgraphs

@openzeppelin/subgraphs

A library for easily building modular dense subgraphs.

Modules available now:

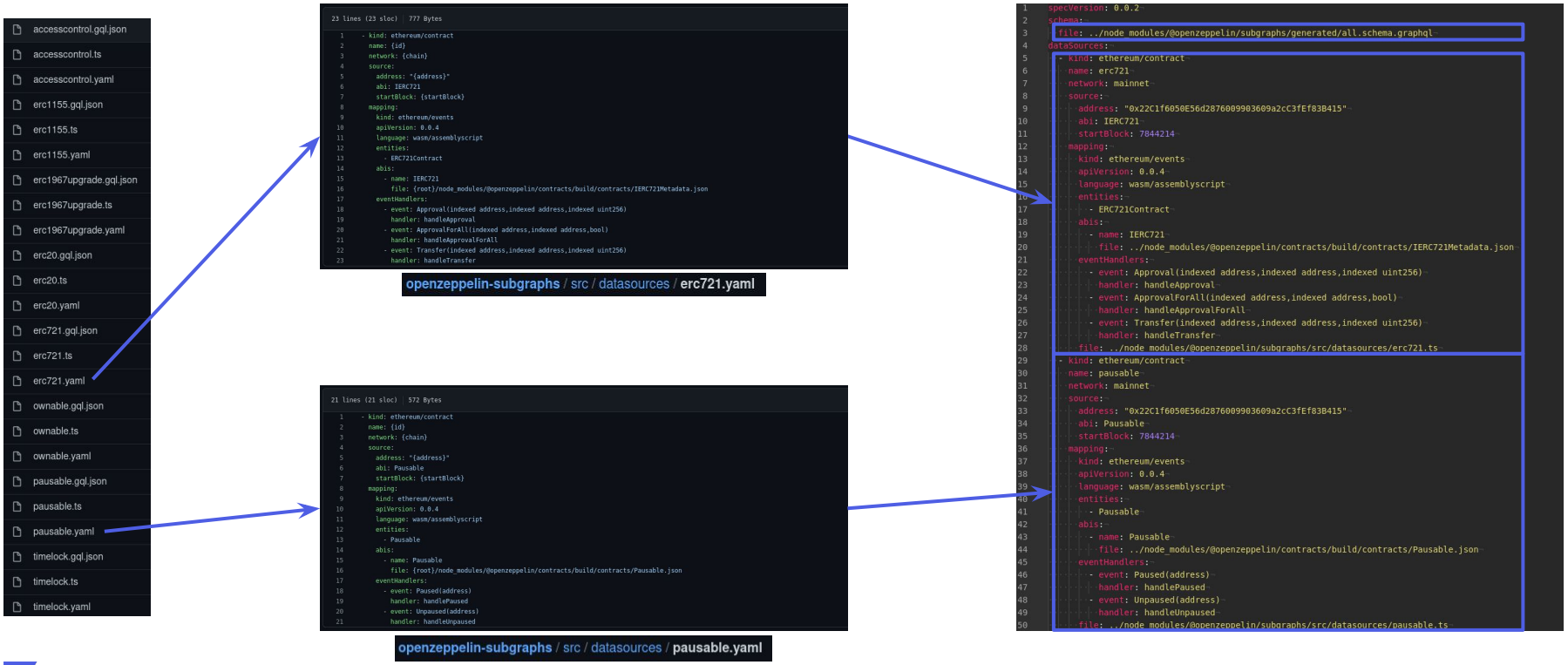
ERC20, ERC721, ERC1155, Ownable, Accesscontrol, Pausable, Timelock

Content of @openzeppelin/subgraphs

- **For each module, the library provides:**
  - A schema of the corresponding entities
  - A datasource template that listen to events
  - Indexing logic in assembly script
- **The library also provides a complete schema with all modules enabled**
- **Subgraphs can be assembled manually or using @amxx/graphprotocol-utils**

Note: Indexing only uses event handlers (no function handlers). Indexing such subgraphs doesn't require access to a node with trace API enabled. Amount of function calls are minimal to improve indexing performance.

Creating a custom subgraph with @openzeppelin/subgraphs



Automated build with @openzeppelin/subgraphs and @amxx/graphprotocol-utils

- **Describe you application** (config.json)

```
{
  "output": "generated/sample.",
  "chain": "mainnet",
  "datasources": [
    { "address": "0xB1C52075b276f87b1834919167312221d50c9D16", "startBlock":  9917641, "module": [ "erc721",    "ownable"       ] },
    { "address": "0x799DAa22654128d0C64d5b79eac9283008158730", "startBlock":  9917642, "module": [ "erc721",    "ownable"       ] },
    { "address": "0xC76A18c78B7e530A165c5683CB1aB134E21938B4", "startBlock":  9917639, "module": [ "erc721",    "ownable"       ] },
    { "address": "0x001d1cd0bcf2e9021056c6fe4428ce15d977cfe0", "startBlock": 11127634, "module": [ "erc1155",   "ownable"       ] },
    { "address": "0xA3B26327482312f70E077aAba62336f7643e41E1", "startBlock": 11633151, "module": [ "erc20",     "accesscontrol" ] },
    { "address": "0x3d85004fa4723de6563909fabbcafee509ee6a52", "startBlock": 12322496, "module": [ "timelock",  "accesscontrol" ] }
  ]
}
```

- **Generate custom schema and manifest**

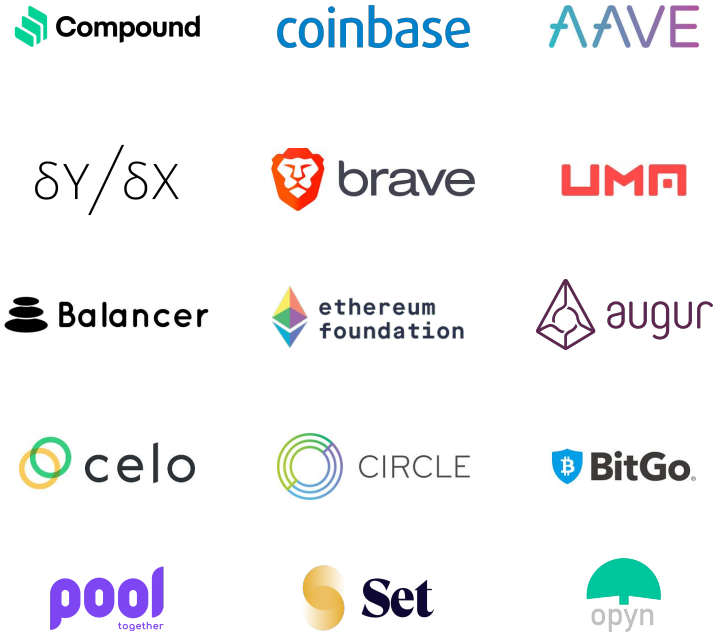
```
npx graph-compiler                                                \
  --config sample.json                                            \
  --include node_modules/@openzeppelin/subgraphs/src/datasources \
  --export-schema                                                 \
  --export-subgraph
```

OpenZeppelin

Our mission is to protect the open economy

OpenZeppelin is a software company that provides **security audits** and **products** for decentralized systems.

Projects from any size — from new startups to established organizations — trust OpenZeppelin to build, inspect and connect to the open economy.



Security, Reliability and Risk Management

OpenZeppelin provides a complete suite of **security and reliability products** to build, manage, and inspect all aspects of software development and operations for Ethereum projects.

@openzeppelin/subgraphs

docs.openzeppelin.com

forum.openzeppelin.com

defender.openzeppelin.com

Thank you!

To learn more about OpenZeppelin Subgraphs join our workshop on July 28th!
zpl.in/subgraphs-workshop

Learn more

openzeppelin.com/**contracts**
forum.openzeppelin.com
docs.openzeppelin.com

Contact

@amxx
hadrien@openzeppelin.com