Accessing and Querying Historical data

In this article, you'll find a high-level overview about the two most common use-cases for blockchain indexing, and how they can be solved using NEARQueryAPI andBigQuery .

Overview

Building a blockchain indexer depends on the kind of historical blockchain data that you want to access and query. Let's consider these two most common blockchain indexing use cases:

- an indexer doingblockchain analytics
- , reporting, business intelligence, and big-data queries.
- an indexer built as a backend for aweb3 dApp
- building interactive and responsive UIs, that tracks interactions over a specific smart contract.

tip Want to learn more about indexing? Check theIntroduction to Indexers article.

Analytics

When building a Blockchain analytics solution, you'll be creating queries to track NEAR assets, monitor transactions, or analyze on-chain events at a massive scale. Handling that huge amount of data in an optimal way so you can provide accurate results quickly requires a well designed solution.

NearBigQuery solution provides instant insights, and let you access Historical on-chain data and queries at scale. It also eliminates your need to store and process bulk NEAR protocol data; you can just query as little or as much data as you want.

BigQuery does not require prior experience with blockchain technology. As long as you have a general knowledge of SQL, you'll be able create queries and unlock insights about transactions, smart contract utilization, user engagement, trends, and much more.

tip Learn more about BigQuery inthis article .

Analytics use cases

Common Blockchain analytics use cases that can be managed withBigQuery :

- create queries to track NEAR assets
- monitor transactions
- analyze on-chain events at a massive scale.
- use indexed data for data science tasks, including on-chain activities
- identifying trends
- feeding AI/ML pipelines for predictive analysis
- use NEAR's indexed data for deep insights on user engagement
- smart contract utilization
- insights across tokens and NFT adoption.

WebApps

Building Web3 apps that handle historical blockchain data require dedicated solutions that manage the data and reduce the latency of requests, as it's not possible to scan the whole blockchain when a user makes a request. For example, if your dApp needs to keep track of minted NFTs from a specific smart contract, you'll need to keep historical data related to that contract, the NFTs, and all the transactions in an optimized way so the dApp can provide fast responses to the user.

A simple solution for developers building dApps onNEAR is usingQueryAPI , a fully managed solution to build indexer functions, extract on-chain data, store it in a database, and be able to query it using GraphQL endpoints.

In the NFT example, with QueryApi you can create an indexer that follow the activity of yourmy-nft-contract.near smart contract, records all activities related to it (such as minting and transfers), and provides simple endpoints to communicate with your dApp, when your application needs to display all the minted NFTs, or the related transactions to a specific NFT.

QueryApi can also reduce app development time, by letting you auto-generateNEAR component code straight from a GraphQL query. By creating the boilerplate code, you can use it to render a UI and publish your new NEAR component.

tip Learn more about QueryAPI in thisOverview article .

dApp use cases

For a technical implementation deep-dive, check these QueryAPI tutorials:

- NFTs Indexer
- : an indexer that keeps track of newly minted NFT on Near blockchain.
- Posts Indexer
- : this indexer keeps track of every new NEAR Social post found on the blockchain.
- Social Feed Indexer
- : this indexer keeps track of new posts, comments, and likes on NEAR Social, so a social feed can be rendered quickly. Edit this page Last updatedonJan 9, 2024 bygagdiez Was this page helpful? Yes No

Previous Context object Next Migrate from Lake framework