QueryAPI Overview

Near QueryAPI is 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.

Indexing complexity

Blockchain Indexers are known to be challenging to create, maintain, and operate. You have to focus on the business logic of your indexer, while you also have to take care of everything else around it. A dedicated team member could be needed to deal with all these challenges.

Common indexing challenges include:

Creation

- Design Database Schema and provision it with correct configurations for security, data retention, and performance.
- Write and test indexer code that interacts with the database
- Deploy Indexer to a Cloud provider. Ensure network permissions firewalls, PCs, or other network-related settings are setup correctly.
- Create an API endpoint to retrieve data from your database for your fronted applications

Maintenance

- Monitor the performance of your database and scale it as needed
- Manage permissions and database access with changing requirements

Operation

- Re-index data due to issues and updates. Ensuring that production environments don't get disrupted.
- Perform database schema migrations
- Scale the API as your application grows
- Keep up with all the underlying blockchain nodes and upgrades

QueryAPI

As you can see, running indexers is a complex and comprehensive set of processes and Near QueryAPI tries to cover most (or all) of these needs, offering an open-source solution for creating, managing, and exploring indexers.

NEAR Component

QueryAPI has aQueryApi.App NEAR widget , hosted under thedataplatform.near account. With this component, you can see all the public indexers currently available on the Near blockchain.

If you would like to create a new indexer, simply clickCreate New Indexer .

Indexers stored on-chain

QueryAPI stores all the indexer logic and schemas used to provision the databases on-chain. Whenever you interact with the QueryAPI NEAR component, in the background it's making an RPC query toqueryapi.dataplatform.near , where a smart contract stores all of your indexer logic as well as your schemas.

For example, if you select thefeed-indexer and click onView indexer you'll see all the details about an indexer that powers thenear.org 's main posts feed. You're free to review the JavaScript code of the indexer function, or check the SQL that defines the database schema.

Known limitations

- Currently under closed beta testing.
- Only supports JavaScript indexers. (we plan to support Rust in the future)
- It always takes the latest@near-lake/primitives
- library.
- It doesn't support schema migrations.* If you have an indexer whose schema needs to change, you may need to create a new indexer and do a historical backfill on that new indexer again.
- There's no way to stop or restart your indexer, truncating all tables.
- Historical backfill works in parallel to the real-time indexing.* Historical processing won't happen in order. (it will

happen at the same time as top of network)

- Keep that in mind to ensure you don't have unintended side effects.

Join the Beta If you want to join the closed beta, please fill out this form for access. Pricing Pagoda currently doesn't charge for storing your indexer code and data or running the indexer, but we will introduce this soon. Edit this page Last updatedonJan 9, 2024 bygagdiez Was this page helpful? Yes No

Previous BigQuery Next How it works