NEXTDOOR

Scaling community interactions with FaunaDB

ABOUT THE USER

Nextdoor is a large social network provider, focusing on neighborhood conversations.

PROJECT OVERVIEW

Nextdoor's neighborhood organization means municipal and safety services see them as a useful channel for connecting with residents. One of the most heavily used features in Nextdoor app is the ability for these local government services to send broadcast alerts to users in particular areas. However, the queries to compile lists of users based on group membership are complex and were creating significant performance and operational challenges for their existing PostgreSQL deployment. Similar behavior was also seen in other portions of the Nextdoor application. Therefore, Nextdoor embarked on an effort to create a new "groups subsystem" to offload such queries and minimize the impact on the application's performance.

REQUIREMENTS & CHALLENGES

Nextdoor had multiple business and technical requirements driving the shape of the groups subsystem.

Firstly, Nextdoor is seeing a boom the usage of its mobile app. They anticipate traffic to the group subsystem to increase steadily. The subsystem itself is expected to find use in multiple functions in the Nextdoor app. Therefore, the subsystem must be supported by a database that can scale up with the application traffic, without custom hardware or specialized solutions.

Secondly, Nextdoor mobile app serves users distributed globally. As such, they wanted data to be available worldwide, and highly available in multiple regions.

Thirdly, group subsystem queries are IO intensive, including index lookups, nested joins and set intersections. The group membership model is graph-like, with users as members of neighborhoods, but also of other groups. Neighborhoods are part of larger groups, which can also be nested into regions. Nextdoor required a database with support for complex queries.

Fourthly, Nextdoor is built in the Cloud. They wanted to adopt a database backend that would operate in the cloud as well, thereby minimizing administrative and operational overheads associated with running their application.

Lastly, various teams had been migrating workloads from RDBMS to NoSQL for almost a decade, to take advantage of the scalability of NoSQL systems.. What remained were the workloads that require relational features like joins and ACID transactions, leaving Nextdoor with a hodgepodge of Postgres and NoSQL databases. Nextdoor saw the group subsystems effort as an opportunity for consolidating workloads to lower their total cost of ownership.

WHY FAUNA

FaunaDB gave Nextdoor the general purpose platform they needed for running mission critical workloads in cloud-native environments.

Unlike Postgres, FaunaDB was designed from the ground up as a cloud-native and horizontally scalable database. It delivers the same set of data management capabilities, no matter the data distribution topology. Most importantly, it does so without sacrificing relational features desired by Nextdoor.

Robust multi-region replication with strong consistency means that data committed to FaunaDB is available across all regions, so data is correct and complete even in the face of disasters. This gave Nextdoor the horizontal scalability they require.

FaunaDB features a multi-model interface that includes relational primitives such as ACID transactions, consistent indexes, joins, as well as document- and graph-styled querying, all stored with configurable temporal snapshot retention. Nextdoor's groups subsystem queries include nested joins and set intersections -- a perfect fit for FaunaDB's expressive query language.

In FaunaDB, Nextdoor found a data platform designed to grow with their business, not just a scalable transaction engine with powerful queries. The full suite of platform features include multi-tenancy and object level security mean Nextdoor can expand their FaunaDB installation to support more applications and use cases.

RESULTS

By choosing FaunaDB for the group subsystem, Nextdoor was able to isolate the workload so that group queries do not contend with other application traffic. FaunaDB's query language allows them to express complex queries and compose queries programmatically. This flexibility means Nextdoor can expand the use cases for targeted content, while also exploring more ways to use FaunaDB. Because a single FaunaDB cluster is designed to support multi-tenancy, they can easily add new workloads while continuing to grow the groups subsystem.

> Fauna is unique among databases. It meets our transactional, performance, and scalability requirements without compromising developer productivity.
>
> **- Prakash Janakiraman, Co-Founder & Chief Architect Nextdoor**