Sponsorship

Quastor > Posts > How Shopify scaled their Database

How Shopify scaled their Database

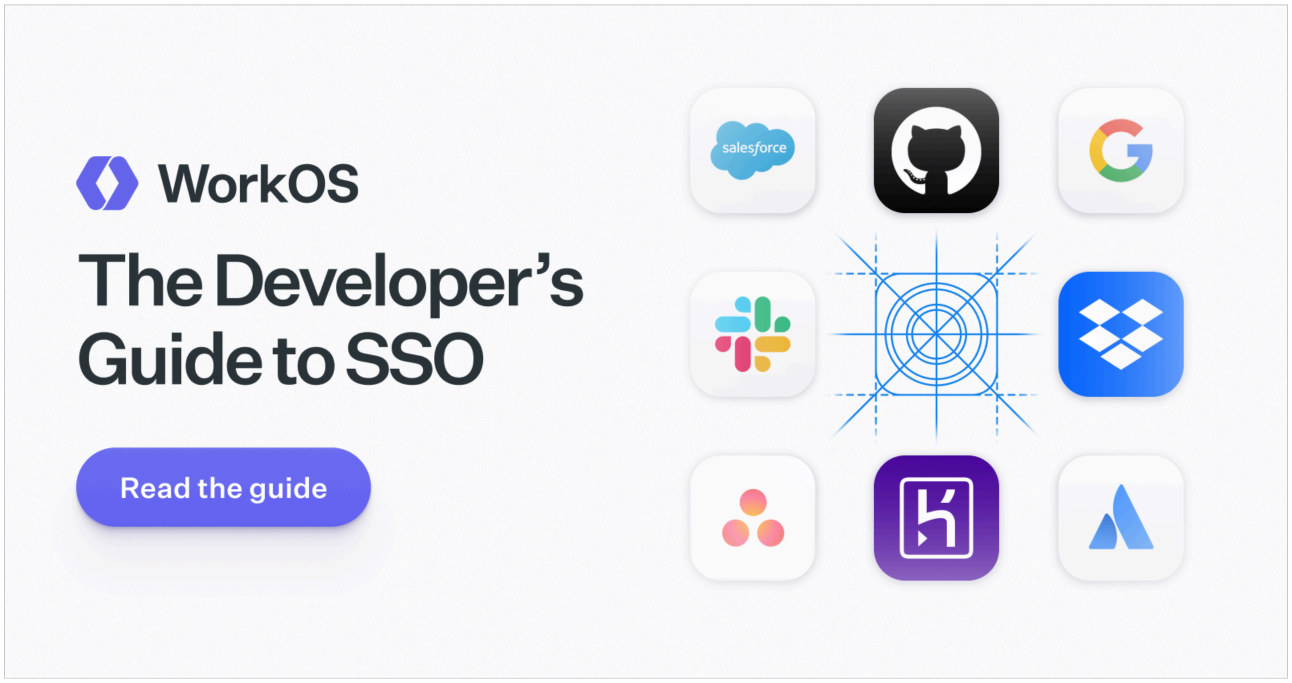
How Shopify scaled their Database with Vitess. Plus, a guide to mentorship for software engineers, a dive into database fundamentals and more.

Arpan KG
April 16, 2024

Hey Everyone!

Today we'll be talking about

- **How Shopify Scaled MySQL**
  - Shopify released the Shop app and quickly began to experience exponential growth. An issue they faced was scaling MySQL.
  - After evaluating several solutions, they went with Vitess, an open source sharding solution developed at YouTube.
  - We'll introduce Vitess and talk about how it works. Then, we'll talk about the process Shopify went through in order to integrate it.
- **Tech Snippets**
  - Guide to Mentorship For Software Engineers
  - 3Blue1Brown's course on Deep Learning and Transformers
  - Debunking Devin: "First AI Software Engineer" Lie Exposed
  - A Dive into Database Fundamentals
  - How leaders can help their staff grow



The Developer's Guide to SSO

One of the most effective ways to increase user engagement is by optimizing your sign up flow. Adding single sign-on (SSO) can help you massively reduce the signup friction visitors face and also help you land larger enterprise deals.

Building SSO in-house means familiarizing yourself with protocols like OAuth, SAML and OpenID Connect.

Some implementation details you'll have to consider include

- A **strategy to correctly authenticate users** in your app based on the attributes that identity providers (IdPs) send back
- **Support for multiple IdPs** (*this will need to be normalized*) with mapping attributes from the different IdPs to the user profile structure in your app
- **Audits and Monitoring** to detect any suspicious activity and immediately respond to potential security breach attempts

If you'd like to learn more about SSO, then this is a super comprehensive guide with details

Read the Guide

sponsored

How Shopify Scaled MySQL

Shopify is an e-commerce platform that helps merchants create online stores to easily sell their products. Close to 5 million stores use Shopify and they account for over $450 billion in total e-commerce sales.

In April of 2020, Shopify unveiled the Shop app. This is a mobile app that allows customers to browse and purchase products from various Shopify merchants.

Since launch, the Shop app has performed *incredibly* well with millions of users purchasing products through the platform.

The company has been experiencing exponential growth in usage so they've had to deal with quite a few scaling pains. Shopify uses Ruby on Rails and MySQL, so one of their main issues was scaling MySQL to deal with the surge in users.

At first, they employed federation as their scaling strategy. Eventually, they started facing problems with that approach, so they pivoted to Vitess. They wrote a fantastic article talking about how they did this.

In this article, we'll talk about federation, the issues Shopify faced, why they chose Vitess, and their process of switching over.

We'll cover a ton of concepts on database sharding in this article, so be sure to check out the Anki Spaced-Repetition flash cards to remember them.

We have an Anki deck with hundreds of flash cards covering all the past concepts we've discussed in Quastor articles (load balancers, caching, vertical vs. horizontal partitioning, Redis and much more).

Federation

Shopify's first strategy to scale MySQL was federation. This is where they took the primary database and broke it up into smaller MySQL databases.

They identified groups of large tables in the primary database that could exist separately - these table groups were independent from each other and didn't have many queries that required joins between them.

Shopify moved these groups of tables onto different MySQL databases.

Ruby on Rails makes it easy to work with multiple databases and Shopify developed Ghostferry to help with the database migrations.

However, Shopify eventually ran into pains with the federated approach. Some of the issues were

- **Couldn't Further Split the Primary Database** - Even with the splitting, the primary database eventually grew to terabytes in size. However, Shopify engineers could no longer identify independent table groups that they could split up. Splitting up the tables further would result in cross-database transactions and create too much complexity to the application layer.
- **Long Schema Migrations** - running schema migrations meant changing table structure on all the smaller, independent MySQL databases. This became a time-consuming and tedious process.
- **Interrupted Background Jobs** - running the background job to split the database tables onto smaller independent databases became time-consuming as well. These jobs were frequently getting throttled because the primary database was too busy with user queries.

Instead, Shopify decided to overhaul their approach to scaling. After exploring a bunch of different options, they found that Vitess was their best bet.

Introduction to Vitess

Vitess is an open source sharding solution for MySQL developed in 2010 at YouTube. It was donated to the Cloud Native Computing Foundation (CNCF) by Google in 2018 and graduated in November 2019. Vitess is used at companies like GitHub, Slack, Bloomberg and many others.

It's a solution for horizontally scaling MySQL and it handles things like

- **Sharding** - Vitess can shard/re-shard your data based on multiple different schemes. It handles the complexity of routing queries to the appropriate shard (*or shards*) and aggregating the results
- **Schema Migrations** - Vitess handles schema migrations and lets you make changes to the schema while minimizing downtime.
- **Transaction Management** - Vitess provides ACID transactions when you're working with a single shard. Distributed ACID transactions are not supported but Vitess offers 2PC transactions that guarantee atomicity.
- **Connection Pooling** - If you have lots of clients using the database then Vitess can efficiently reuse existing connections and scale to serve many clients simultaneously.

And more.

Key Components in Vitess

In Vitess, there are several key components that you'll be working with. These are concepts that tend to repeat when working with any managed sharding service, so they're useful to be aware of.

- **Shard** - each shard in Vitess represents a partition of the overall database and is managed as a separate MySQL database.
- **Keyspace** - a logical grouping of shards. You can have an *unsharded* keyspace (*where it's just a single shard*) or a sharded keyspace (*grouping multiple shards where each shard is a separate MySQL database*)
- **VSchema (Vitess Schema)** - Vitess Schema contains metadata about how tables are organized across keyspaces and shards. It describes the sharding scheme and the relationships between shards within a keyspace.
- **VTTablet** - VTTablet runs on each Vitess shard and serves as a proxy to the underlying MySQL database. It helps with the execution of queries, connection management, replication, health checks and more.
- **VTGate** - VTGate acts as a query router. It's the entry point for client

parsing, planning and routing the queries to the appropriate VTTablets based on the VSchema.

When a user wants to send a read/write query to Vitess, they first send it to VTGate. VTGate will analyze the query to determine which keyspace and shard the query needs data from. Using the info in VSchema, VTGate will route the query to the appropriate VTTablet. The VTTablet then communicates with its respective MySQL instance to execute the query.

For queries that span multiple shards, VTGate will contact multiple VTTablets and aggregate the results.

Migrating to Vitess

The first thing Shopify had to do was choose their shard key. Most of the tables in their database are associated with a user, so `user_id` was chosen as the sharding key. However, Shopify also wanted to migrate data that wasn't associated with any specific users to Vitess.

To implement this, Shopify decided to have two main keyspaces

- **Users** - this keyspace is all the user-owned data. It's sharded by `user_id`
- **Global** - this is the rest of Shopify's data that isn't related to a specific user. It isn't sharded. This could include data like shared resources, metadata, etc.

Phase 1: Vitessifying

The Vitessifying Phase was where Shopify engineers transformed their MySQL databases into keyspaces in their Vitess cluster. This way, they could immediately start using Vitess functionality without having to explicitly move data. They *did not* shard in this step. Instead, all the keyspaces corresponded to a single shard.

In order to implement this, Shopify added a VTablet process alongside each `mysqld` process. Then, they made changes to their application code so it could query VTGate and run the query through Vitess instead of the old, federated system.

Shopify used a dynamic connection switcher to gradually cutover to the new Vitess system. They carefully managed the percentage of requests that were routed to Vitess and slowly

Phase 2: Creating Keyspaces

After Vitessifying, the next step was to split the tables into their appropriate keyspaces. Shopify decided on

- Users
- Global
- Configuration

as their keyspaces.

Vitess provides a MoveTables workflow to move tables between keyspaces. Shopify practiced this thoroughly in a staging environment before making the changes in production.

Phase 3: Sharding the Users Keyspace

After phase 2, Shopify had three unsharded keyspaces: global, users and configuration. The final step was sharding the users keyspace so it could scale as the Shop app grew.

They first had to go through a few preliminary tasks (implementing Sequences for auto-incrementing primary IDs across shards and also Vindexes for maintaining global uniqueness and minimizing cross-shard queries)

After implementing these, they organized a week-long hackathon for all the members of the team to gain a thorough understanding of the sharding process and run through it in a staging area. During this process they were able to identify ~25 bugs spread across Vitess, their application code and their infrastructure layers.

After gaining confidence, they created new shards that matched the specs as the source shard and used Vitess' Reshard workflow to implement sharding in the users keyspace.

Lessons Learned

Some of the lessons the Shopify team learned were

- **Pick a Sharding Strategy Earlier than you Think** - If you pick a sharding on a sharding scheme early and set up linters to enforce the scheme on your database. When you decide to shard, the process will be much easier.

- **Practice Important Steps in a Staging Environment** - Shopify suggests setting up an accurate staging environment and practicing every single step before attempting it in production. They had two staging environments and discovered some bugs along the way. They also recommend using dummy data in the staging environment to help identify potential issues.
- **Invest in Query Verifiers** - Shopify heavily invested in query verifiers, which were critical for their successful move to Vitess. They used these to remove cross keyspace/shard transactions and reduce any cross keyspace/shard queries.

How AI companies like Copy.ai, Jasper, and AI21 Labs are using WorkOS for enterprise-grade auth

Copy.ai is one of the most popular AI tools for sales and marketing. Millions of people use the Copy.ai platform to run their sales and marketing teams.

Over the last few years, they had explosive growth and needed a new auth platform that could support SSO and SCIM provisioning *(features that large customers were explicitly*

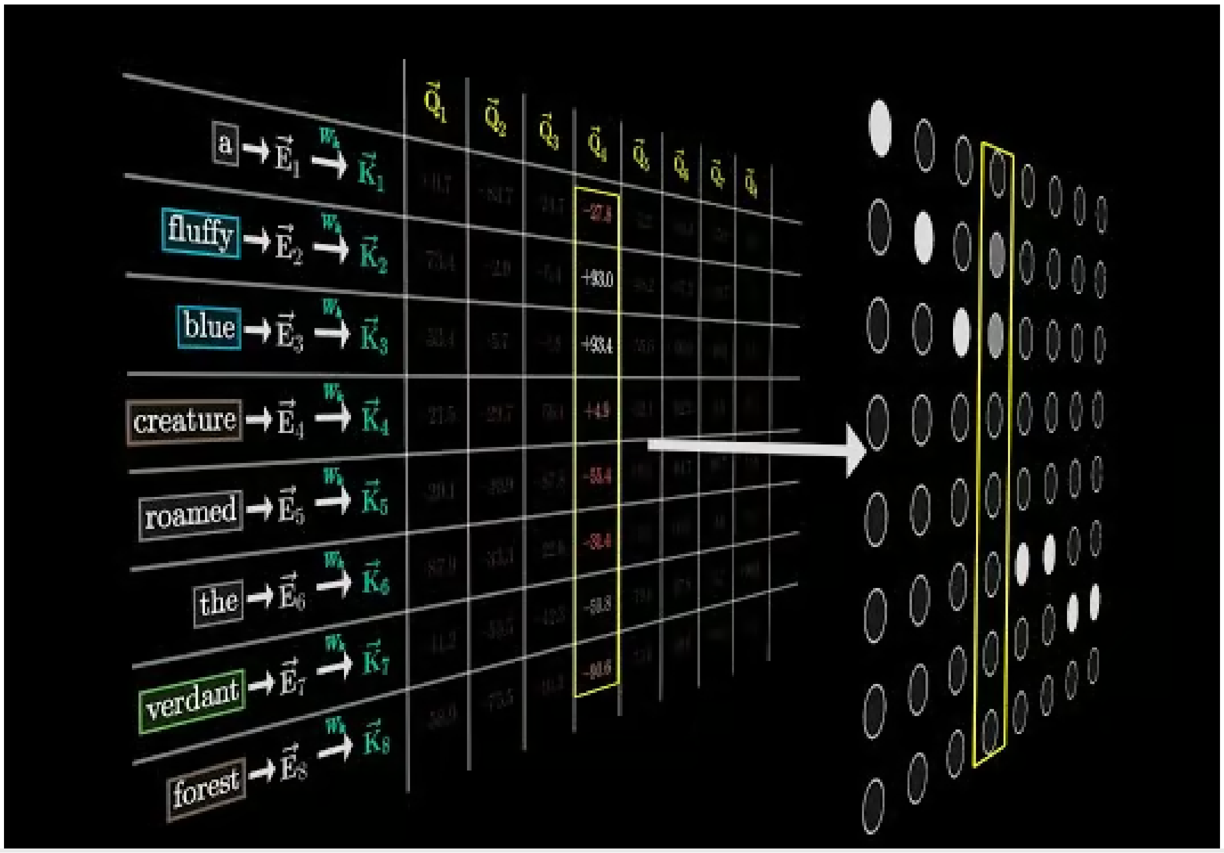
Using WorkOS, they were able to add SSO and Directory Sync in less than 2 weeks, immediately unblocking enterprise deals.They also successfully migrated hundreds of thousands of active users to WorkOS via AuthKit and User Management, which supports up to 1 million monthly active users *for free*.

WorkOS provides easy-to-use APIs and unmatched economies of scale. Hundreds of high-growth startups use WorkOS to quickly move upmarket, enabling their engineers to get back to building core product features.

Learn how to add enterprise-grade auth to your app

sponsored

Tech Snippets

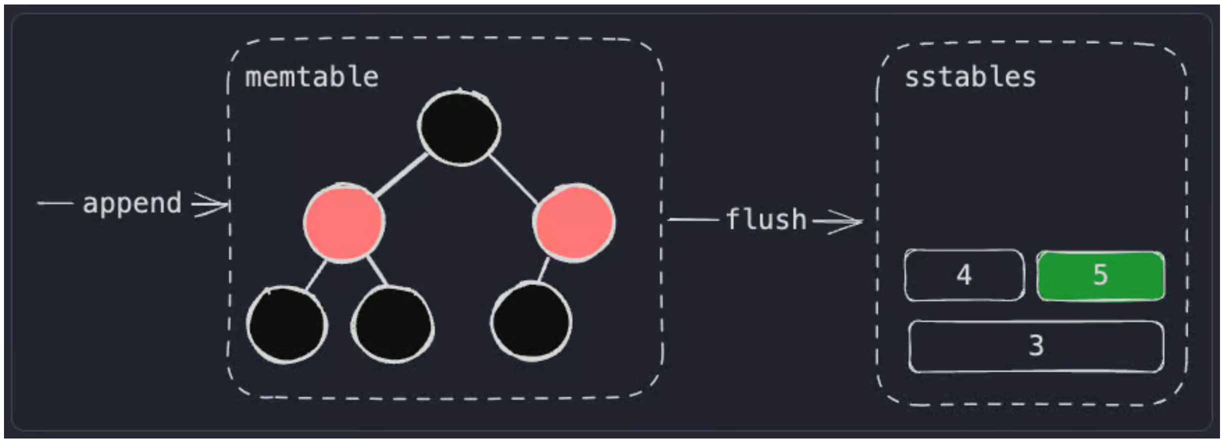


3Blue1Brown Series on Neural Networks

3Blue1Brown is a fantastic YouTube channel with in-depth explanations on topics in mathematics. The videos have amazing visualizations that really help you understand the fundamentals.

They have an awesome series on neural networks and deep learning. Recently, they've updated it with videos on transformers and the attention mechanism. It's a great watch if you'd like to learn more about the latest innovations in ML.

www.3blue1brown.com/topics/neural-networks

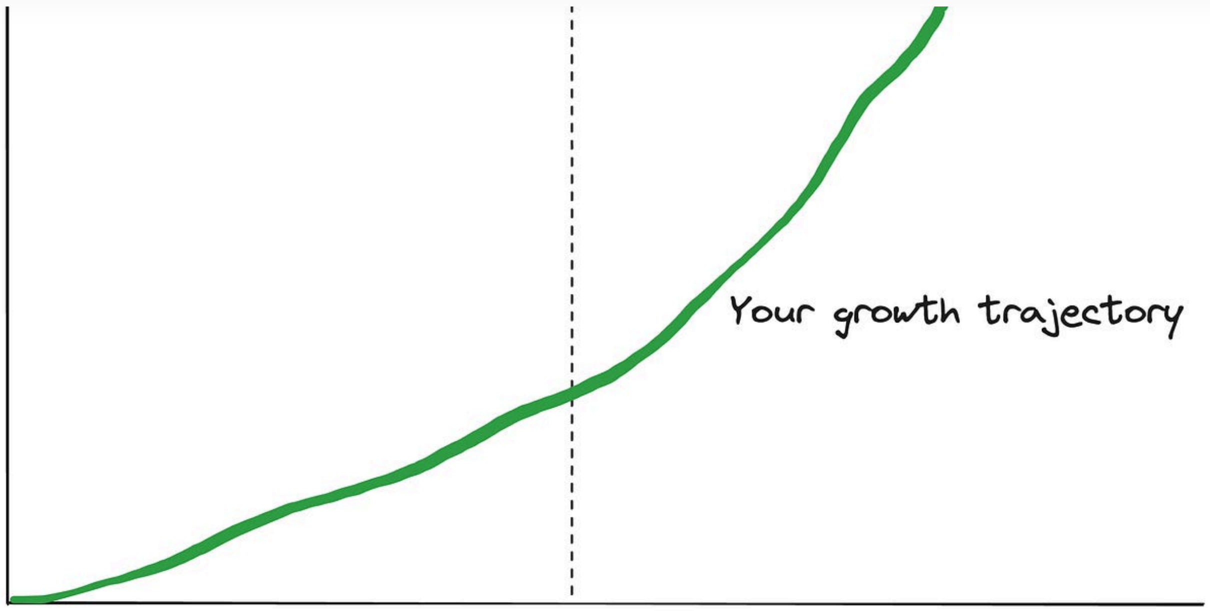


A Dive into Database Fundamentals

This article is a terrific deep dive into database fundamentals. It talks about ACID properties (and how they can be implemented in a toy database), storage engines, indexes, LSM trees, the CAP theorem and much more.

It gives a fantastic overview of many of the most interesting topics in databases and also provides additional resources if you want to go further.

tontinton.com/posts/database-fundementals



Guide to Mentoring for Software Engineers

Mentoring others will obviously help you grow other people but it also helps you develop yourself. It can help you improve your communication and leadership skills while also building your emotional intelligence.

This is an insightful guide on how to get started in mentorship as an engineer. It talks about finding a mentee, understanding their goals, setting up a growth plan and much more.

read.highgrowthengineer.com/p/2024-guide-to-mentoring-for-software

Internet of Bugs
Debunking Devin: "First AI Software Engin...
Watch later
Share
This Is A Lie
Devin's Upwork Side Hustle
Cognition
25.4K subscribers
574
Share
85K views 3 weeks ago
Watch Devin make money taking on messy Upwork tasks!

Debunking Devin: “First AI Software Engineer“ Upwork Lie Exposed

Recently there was a ton of buzz about Cognition Labs. They’re a startup trying to use LLMs and LLM agents to build “Devin“, the world’s first AI software engineer. For their launch, they released an impressive video where Devin was able to fully solve a task on Upwork.

Unfortunately, it seems like the video was more hype than substance. This is a thorough rebuttal video by the Internet of Bugs YouTube channel that dives into the Devin launch video and shows how it was misleading.

https://youtu.be/tNmgmwEtoWE

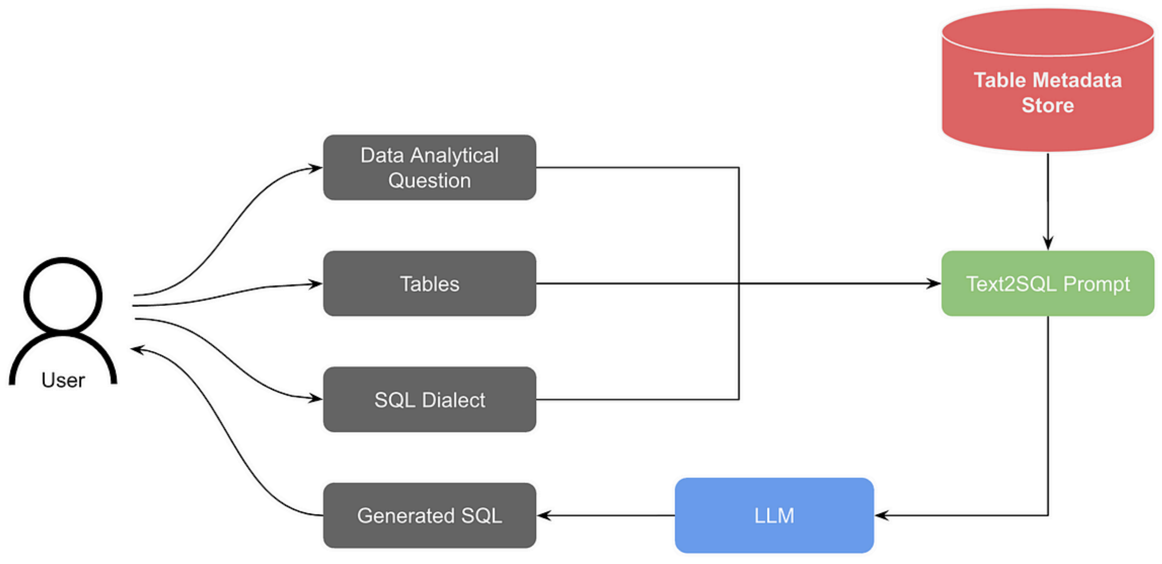
How to help your team grow with mentorship, coaching and sponsorship

One of the main responsibilities of a manager is helping their staff grow and develop. Three tools for doing this are mentorship, coaching and sponsorship.

Mentorship is where you directly pass on relevant knowledge and experience to the mentee. Coaching is where you help them achieve their specific goals through providing feedback and accountability. Sponsorship is where you advocate for their career advancement using your influence and position.

This is an interesting article that delves into all three and how you should apply each tool.

jacobian.org/2024/apr/1/mentorship-coaching-sponsorship



How we built Text-to-SQL at Pinterest

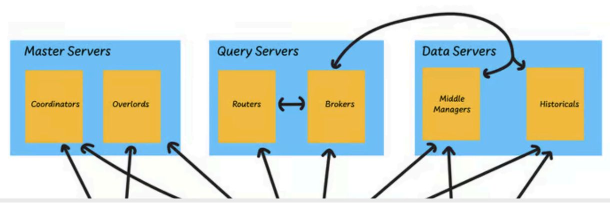
A team at Pinterest used LLMs to assist employees in writing SQL queries. A. user can ask a question like "how many users signed up in the UK over the last 30 days" and the LLM will formulate an SQL query based on Pinterest's data schemas.

This is an in-depth article on the LLM engineering that the Pinterest team had to do in order to build the tool. It talks about how they addressed the LLM context window limit, using Retrieval Augmented Generation for table selection, evaluating results and more.

If you're interested in building your own tooling with LLMs, then this is a terrific read.

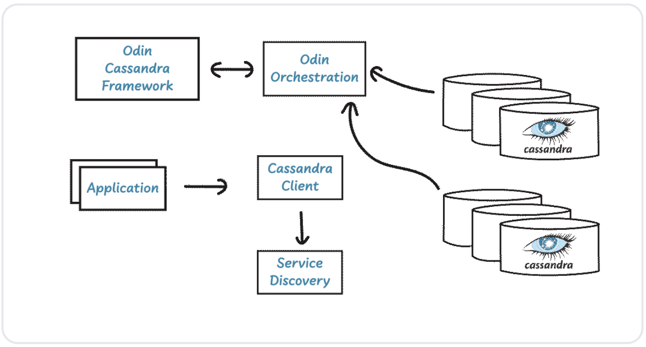
medium.com/pinterest-engineering/how-we-built-text-to-sql-at-pinterest-30bad30dabff

Keep reading



How GitHub Built Their Code Search Feature

We'll talk about the data structures and tech underlying GitHub's code search feature. Plus, Sergey Brin's recent talk on Gemini Pro



Premium

Tech Dive on gRPC (Quastor Pro)

We'll talk about RPCs, gRPC, Pros/Cons, Real World Examples and more. Plus, principles from Deep Work by Cal Newport

The Architecture of Tinder's API Gateway

The design of Tinder's API gateway. Plus, a study on the Ballmer peak, how to review code as a junior dev, Mistral's new LLM and more.

View more ›