Choosing the Right Database: A Guide for System Design Interviews

Agustin Ignacio Rossi Follow

3 min read · Nov 17, 2024

24

When designing systems, one of the critical decisions you'll make is selecting the right database. Different workloads require different types of databases, and understanding the nuances of *read-heavy* vs. *write-heavy* systems can make or break your design. This guide will help you navigate these choices and shine in your next system design interview.

ORACLE
mongoDB
Azure
Microsoft SQL Server
IBM DB2
redis
SAP HANA

• • •

Understanding Workloads: Read-Heavy vs. Write-Heavy

Before diving into database types, you must assess whether your system is **read-heavy** or **write-heavy**. Here’s what that means:

Read-Heavy Workloads

These systems prioritize fast and frequent data retrieval. Examples include:

1. Content delivery platforms (e.g., blogs, video streaming sites).
2. Search engines or dashboards with analytics.

Write-Heavy Workloads

These systems prioritize storing large volumes of data quickly. Examples include:

1. Event logging systems.
2. IoT platforms or real-time monitoring systems.

. . .

Choosing the Right Database

Depending on your workload, you'll want to evaluate different database options:

1. For Read-Heavy Systems

- **Relational Databases (SQL):** MySQL, PostgreSQL — Use efficient indexing to optimize query performance.
- **Key-Value Stores**: Redis, Memcached — Excellent for ultra-fast, in-memory data retrieval.
- **Search Databases:** Elasticsearch — Ideal for full-text search and query-heavy systems.

- **Replication Strategies:** Employ *read replicas* to distribute load and improve availability.

2. For Write-Heavy Systems

- **NoSQL Databases:** MongoDB, Cassandra — Designed for horizontal scaling and high write throughput.
- **Time-Series Databases:** InfluxDB, TimescaleDB — Optimized for time-stamped data, perfect for continuous writes.
- **Columnar Databases:** HBase, Bigtable — Handle analytical workloads with frequent writes.
- **Queue-Based Systems:** Kafka, RabbitMQ — Buffer writes using queues to manage throughput efficiently.

. . .

Scalability: Adapting to Growth

Horizontal Scaling

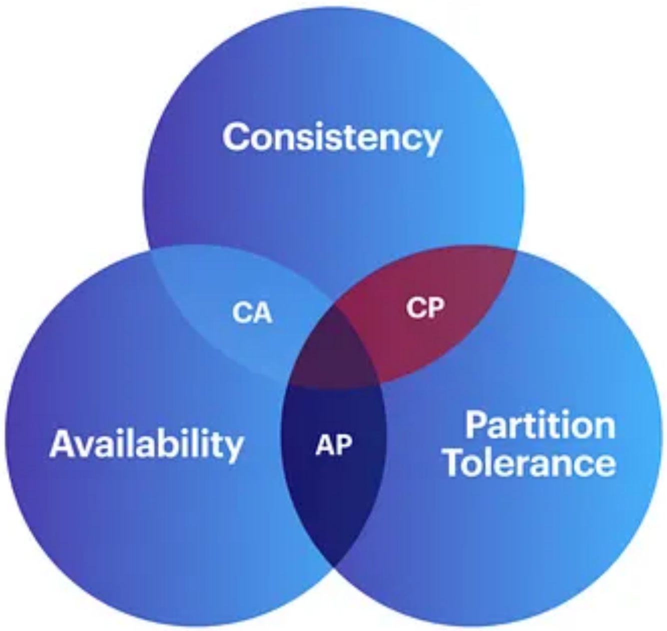
- **For Reads:** Add read replicas or shard data across multiple nodes.
- **For Writes:** Use distributed databases like Cassandra or DynamoDB that handle partitioning seamlessly.

Vertical Scaling

- Upgrade hardware for performance boosts, but note that this has physical limits.

. . .

Consistency vs. Availability



When designing for different workloads, trade-offs are inevitable.

- **Read-Heavy Systems:** Consistency might be critical (e.g., for analytics or financial data). Use relational databases or strongly consistent NoSQL options.

- **Write-Heavy Systems:** Availability often takes precedence, especially for event logging or monitoring. Use eventually consistent databases like Cassandra or DynamoDB.

. . .

Latency: A Key Consideration

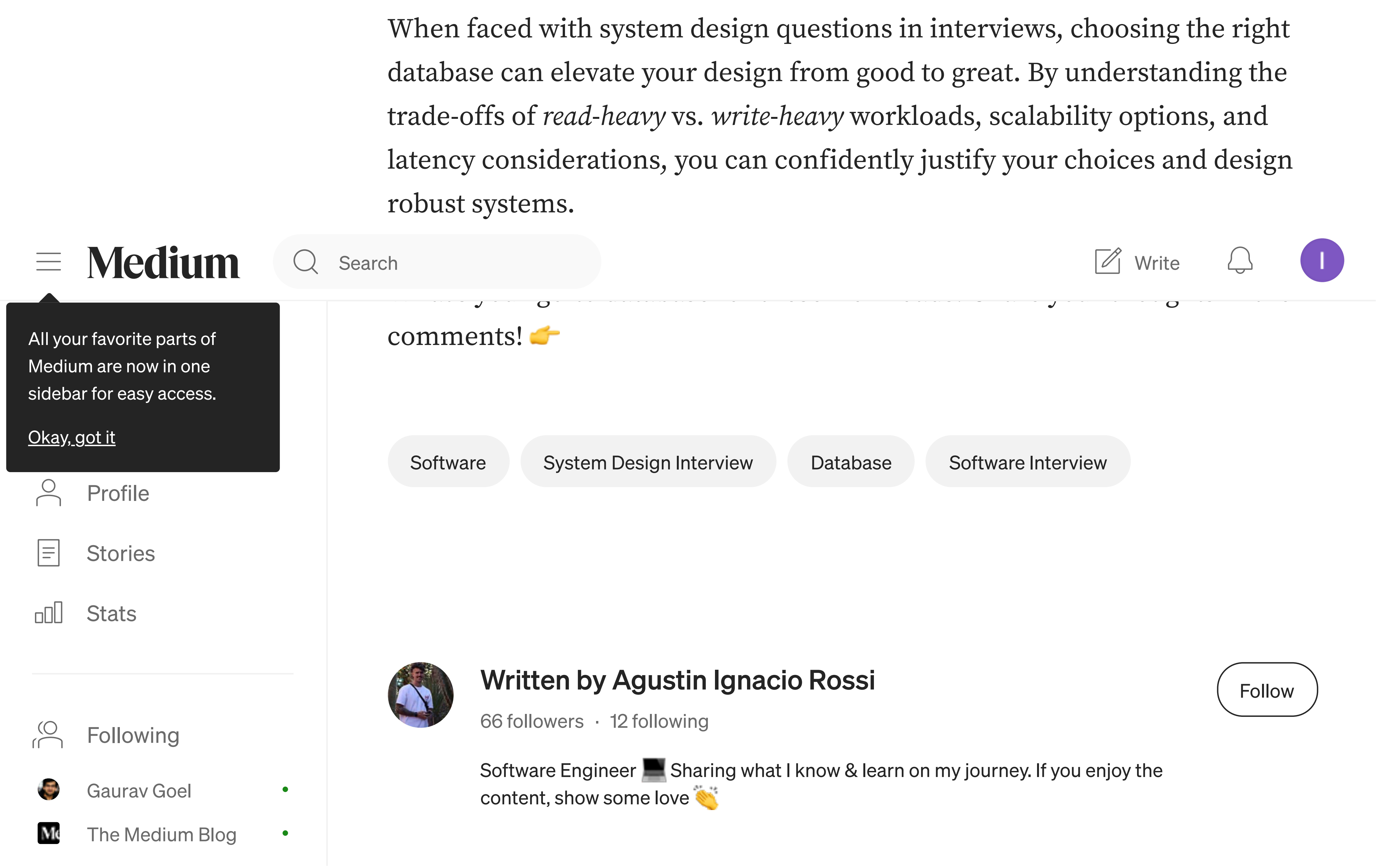
For Reads:

- Use caching layers (e.g., Redis, Memcached) to minimize latency.
- Optimize query patterns and database indexes.

For Writes:

- Use batch writes or asynchronous writes to handle high loads efficiently.
- Avoid heavy constraints or triggers that can slow down write operations.

. . .



When faced with system design questions in interviews, choosing the right database can elevate your design from good to great. By understanding the trade-offs of *read-heavy* vs. *write-heavy* workloads, scalability options, and latency considerations, you can confidently justify your choices and design robust systems.

comments! 👉

All your favorite parts of Medium are now in one sidebar for easy access.

Written by Agustin Ignacio Rossi

66 followers · 12 following

Software Engineer 💻 Sharing what I know & learn on my journey. If you enjoy the content, show some love 👏

Reshma Bidikar

Find writers and publications to follow.

See suggestions

No responses yet

Ishu

What are your thoughts?

More from Agustin Ignacio Rossi

Agustin Ignacio Rossi

Agustin Ignacio Rossi

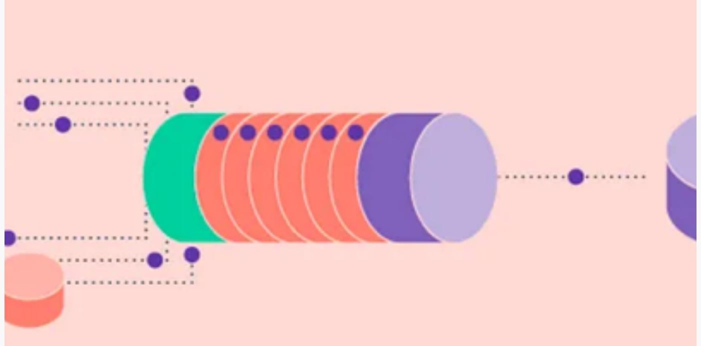
Setting Up an Android Emulator in IntelliJ IDEA Without Android...

Streamline Your Development Workflow by Running Android Emulators in IntelliJ IDEA

Aug 11, 2024 8 1

Geohash or Quadtree? Ready, set and go! for System Design...

Oct 13, 2024 106



Agustin Ignacio Rossi

Understanding the Difference Between Commands and Events i...

Nov 30, 2024

Agustin Ignacio Rossi

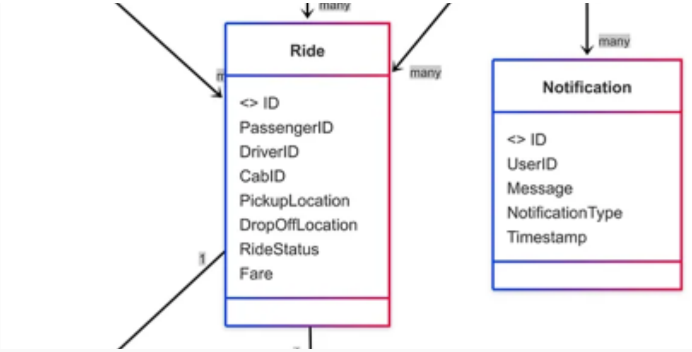
Mastering Video Streaming Protocols: HLS, DASH, and more...

A deep dive into the technologies powering seamless video delivery. Prepare to ace your...

Mar 8, 2024 10

See all from Agustin Ignacio Rossi

Recommended from Medium



Ankit Kumar Srivastava

Design Uber Backend

System requirements

✦ May 25 👏 2

Nikhil Gupta

Google Interview | Senior Software Engineer | Level 5 | Data Structur...

Implement a Birthday Reminder

✦ 6d ago 👏 22 💬 1

Prem Chandak

How Airbnb, Uber, and Calendly

In Level Up Coding by Fareed Khan

Building an Agentic Deep-Thinking

Solved the Double-Booking

Behind the scenes of how tech giants tamed one of the hardest concurrency problems on…

✦ Oct 13 2

RAG Pipeline to Solve Complex

Planning, Retrieval, Reflection, Critique, Synthesis and more

✦ 5d ago 941 8

In Nerd For Tech by Nikhil Bhatnagar

System Design of a Ride Booking System (Uber/Ola/Rapido)

This article deals with the Hld and Lld of a ride booking system where we cover the ap…

Jun 15 6 1

In ITNEXT by Animesh Gaitonde

Solving Double Booking at Scale: System Design Patterns from Top…

Learn how Airbnb, Ticketmaster, and booking platforms handle millions of concurrent…

✦ Oct 8 1.8K 22

See more recommendations