Data Modelling

Sancheeta Kaushal | Ex-EM Blinkit

Airtribe

Disclaimer

- What do you need to think when learning database concepts ?
  - How and when are you going to use them in real world ?
  - What will people ask from you in interview ?
- Do you see databases are more about planning than execution ?
- Build mental models for real world scenarios.
- Nobody is going to expect you to shard a database all by yourself. But they will expect you to be able to explain this concept really well. They will expect you to be able to define what will you use as a shard key.

Disclaimer

- Database learning with some real world flavour but not enough to give you a true picture of how people are making decisions. So, trying something super new, so bear with me.
- The information can be overwhelming for people who haven’t read the blogs from yesterday or even who haven’t revised and practiced the fundamental concepts yet.
- Intent is not for you to understand everything, you haven’t worked on these things so you may not understand it completely.
- Intent is how can you build a mental model for learning these things from the mistakes and learnings of others.
- Build the habit of reading and understanding roughly why they did what they did.

Airtribe

Agenda

- Making compromises in the real world
- Hosted vs Self Managed Databases
- Case Study - Grocery Ordering App
- Polyglot Usage of Databases
- Case Study - Social Media App
- Case Study - Q&A App

Airtribe

Self-Managed vs Hosted Databases

- Nature of the problem
  - Where else do I face this problem in org; what am I using right now ?
  - Can I reuse the same thing ? If not, why ?
  - Eg. Small use-cases solved by Airflow.
- Strategy of the company
  - What fits well with the long term strategy ?
  - Do we want to build expertise in this area ?
- Control vs Limited Customisation

Airtribe

Self-Managed vs Hosted Databases

- Cost
  - Building expertise + Infra + Maintenance vs all managed
  - Look at how cost will grow with data
- Security, Scalability, Performance & Availability vs Hassle Free
- Flexibility vs Vendor Lock In

Case Study – Grocery Delivery App

Scenarios:

- We need to store user profile data, including personal information like address, contact details etc., order history, payment details.
    - RDBMS
- We need to maintain an inventory of groceries from various stores that need to be updated in real-time.
    - Apache Cassandra
- You need to manage customer orders, which includes order placement, processing, and tracking.
    - RDBMS

Airtribe

Case Study – Grocery Delivery App

Scenarios:

- We need to track the status and location of each delivery in real-time.
    - RDBMS/DynamoDB/Cassandra with Postgis
- We need to figure out personalized grocery recommendations based on a customer's past orders and preferences.
    - Neo4j with RDBMS/DynamoDB
- We need to store user session management data.
    - Redis

We don’t always do the right thing.

At Blinkit we used, Entity Value Attribute Paradigm instead of using Mongo

Why ? Maintenance Issues

Polyglot usage

- MySQL, Postgres (Transactional Data)
- DynamoDB (Personalization)
- Elasticsearch (Search)
- Redis (Cache + Message Broker)
- Filestore (S3)
- Postgis (Geospatial Data)

Instacart Case Study – Link

- Instacart migrated from Postgres to DynamoDb for notifications use case
- Reasons
  - Outpacing the largest Amazon EC2 instance size Amazon Web Services.
  - Single instance can’t meet the required throughput for Notifications use case.
  - Capacity can’t be adjusted for low throughput times. Eg. They send very few notifs at night but still they can’t downscale to lower capacity.
- Based on the SLA guarantees and existing literature, they were confident that DynamoDB would fulfill their latency and scaling requirements. But cost was a concern.
- Cost benchmark - Sharded postgres - Estimate the size per sharded node, count the nodes and multiplied it by the cost per node per month.

Instacart Case Study – Link

- DynamoDB cost is based on the amount of data stored in the DynamoDB table, and the number of reads and writes to it.
- Storage is simple to estimate if you know how much data you have, as it costs about 25 cents for every GB stored during the month.
- Usage cost is a bit more complex to estimate. You have to provision capacity based on your usage; cap it to a limit; anything more you use costs more.
- A Read Capacity Unit (RCU) models the rate limit for “fetching data.”
- Write Capacity Units (WCUs) represent the “setting data” request rate limit.
- One WCU is consumed for every write where the row size is smaller than 1KB. About 75% of the rows were over 1 KB.

Airtribe

Instacart Case Study – Link

- Another WCU is consumed for every index created on the table.
- It is important to note that RCUs and WCUs do not have the same price tag associated with them. WCUs cost about 10 times as much as RCUs.
- Users sometimes look up the past messages and hence they had to store them in the database for 7 days.
- They analyzed that they will be using 9 WCU's per record lifecycle.
- They went to optimise from 9 WCU to 4 WCU per record cycle.
- Partition key determines the partition where the item will be stored.
- Sort key allows items with the same partition key to be sorted or "ordered" based on this key's value.

Airtribe

Instacart Case Study – Link

- Primary key is determined based on partition key and sort key and affects the performance.
- They went to optimise from 9 WCU to 4 WCU per record cycle.
- Things they did to achieve this
  - Efficient modelling using partition key and sort key properly. Saved 2 WCU.
  - Further optimised partition key by using UserType#UserID. Saved 1 WCU because additional index not required.
  - GZip is known to have a high compression ratio for JSON. Compressed the data to smaller than 1KB in size more than 99% of the time reduced 2 WCU.

Airtribe

What are your learnings ?

Airtribe

Learnings

- There are physical limitations for infrastructure.
- Cost matters a lot at scale.
- Benchmark on the comparable parameters.
- Efficient modelling is always required.
- Choosing the right partitioning key is very critical for efficiency.
- This is all planning no implementation is done; only prototyping.

Case Study – Social Network – (Facebook)

Scenarios:

- We need to store data related to users such as names, profile pictures, cover photos, friend lists, settings, etc.
  - RDBMS - Sharded MySQL
- Posts, comments, likes, shares, etc. that users create and see on their timelines.
  - Distributed data stores like Apache Cassandra or HBase
  - Facebook build Cassandra but moved away from it later.
- Real-time chat messages between users.
  - Redis + HBase (Insipired by Google BigTable)
- Billions of photos and videos uploaded by users.
  - Haystack for photos/S3

Airtribe

Case Study – Social Network – (Facebook)

Scenarios:

- Representing the connections between users and other entities.
    - Graph Databases/TAO (in-house graph storage)
- Indexing all data for efficient search.
    - Elasticsearch
- Capture real-time user interactions.
    - Redis or Memcached for immediate retrieval and persistence in NoSQL databases for durability.
- Advertisements, user interactions with ads, billing info, etc.
    - Mysql

Airtribe

Facebook Case Study – Link

- Facebook was trying to solve the Inbox Search problem in 2008.
- The challenge is about storing reverse indices of Facebook messages that Facebook users send and receive while communicating with their friends on the Facebook network.
- The amount of data to be stored, the rate of growth of the data and the requirement to serve it within strict SLAs made it very apparent that a new storage solution was absolutely essential.
- Cassandra is a distributed storage system for managing structured data that is designed to scale to a very large size across many commodity servers, with no single point of failure.

Facebook Case Study – Link

- Reliability at massive scale is a very big challenge. Hence Cassandra aims to run on top of an infrastructure of hundreds of nodes (possibly spread across different datacenters).
- At this scale, small and large components fail continuously some are hardware failures and some are software failures.
- The way Cassandra manages the persistent state in the face of these failures drives the reliability and scalability of the software systems relying on this service
- Data is distributed across the nodes in the cluster using Consistent Hashing and on an Order Preserving Hash function. We use an Order Preserving Hash so that we could perform range scans over the data for analysis at some later point.

Facebook Case Study – Link

- Consistent hashing ensures that keys are distributed equally among nodes.
- Order-preserving consistent hashing ensures that the order of keys is maintained.
- Cluster membership is maintained via Gossip style membership algorithm. Failures of nodes within the cluster are monitored using an Accrual Style Failure Detector.
- Gossip style membership involves nodes periodically exchanging information in a decentralized manner, in a way how rumors are spread in social networks.
- Accrual failure detectors give a value that represents the level of suspicion that a node has failed.
- First deployment of Cassandra system within Facebook was for the Inbox search system. The system stored TB’s of indexes across a cluster of 600+ cores and 120+ TB of disk space.

Airtribe

Learnings

- Fault tolerance matters a big deal at scale.
- Always estimate rate of growth of data.
- Management overhead of consistent hashing and ordering is on the database.
- Cluster membership health metrics vs Config server mongodb
- Look at the scale, always ask what’s the maximum scale my data can hit in next 6 months, 1 year and 3 years because if you pick a database you’ll mostly stick with it for good number of years.
- Don’t over-engineer. If your scale is not clear, use RDBMS.

Airtribe

Quora Case Study – Link1

- Quora uses MySQL to store critical data such as questions, answers, upvotes, and comments.
- Data size stored in MySQL is of the order of tens of TB without counting replicas.
- Their queries per second are in the order of hundreds of thousands.
- In order to improve performance, they implemented caching using Memcache as well as Redis. (Redis because of support for list data structure)
- They had to look at sharding to meet the growing needs of reading data faster.
- They implemented the ability to move a table from one MySQL host to another. This allowed them to have multiple masters with different tables on different masters, which we refer to as "vertical sharding"

Airtribe

Quora Case Study – Link1

- Each one of these masters, along with its slaves, is called as a “partition” at Quora.
- Zookeeper (ZK) stores
  - The mapping from a partition to the master/slave physical hosts.
  - The mapping from a partition to the list of tables in that partition.
- Need for Horizontal sharding
  - Online schema change tool encounters issues
    - Needs 2X the space
    - Can take several hours (sometimes more than a day). Occasionally it will abort due to load spikes and need to be restarted from the beginning.
    - Short duration locks at the beginning of schema change.
  - MySQL sometimes chooses the wrong index when reading or writing. Choosing the wrong index on a 1TB table can wreak

Airtribe

Quora Case Study – Link1

- Horizontal Sharding
    - Build vs Buy Decision
        - Build
            - Effort vs Requirement - Only 10 tables needed to go through this
            - Infrastructure reuse from vertical sharding
            - Removal of SQL Dependency from Code - No join
            - Plans to build graph storage, in case of 3rd party integration can potentially cause latency due to multiple middlewares.
    - Storing metadata of shards
        - Reuse zookeeper

Airtribe

Quora Case Study – Link1

- Horizontal Sharding
  - SQL API for access
    - Explicit arguments such as the table name, select clause list, expression for where clause, order by column list, etc
    - Joins, group by, having, and offset are not supported
    - Order by is supported only on the sharding column
    - The caller is responsible for figuring out whether their query is looking for one sharding key, a list of sharding keys, or a range of sharding keys as well as providing that information to the query

Quora Case Study – Link1

- Horizontal Sharding
  - Choosing a sharding column
    - When a table T is sharded by column x one consequence is that the queries which lookup using a different column y will need to examine ALL shards
    - If the QPS of those queries is high OR if they are sensitive to latency, then having a large number of shards could cause performance problems
  - Shard data management
    - Migrating data from unsharded table to sharded table is hard.
    - Once, the shard is created much easier to split it further. You can create a new partition, add the shard metadata, rename the shards and start the traffic.

Quora Case Study – Link1

- Horizontal Sharding
  - No. of shards to create
    - They decided against creating a large number of shards in order to reduce the penalty from examining multiple shards for queries that lookup using a non-sharding column.
    - Shard column auto-increment
      - Yes, most of the growth of the table will occur in the “last” shard. So, few hundred GBs allocated to last shard.
      - No, so all shards of the table are likely to grow over time. A shard size of under 100GB for the shards during the initial sharding.

Airtribe

Quora Case Study – Link2

- Database load increased due to
  - Growth in no. of users and higher engagement.
  - Increase in data size, increases the database load.
  - New features and spammers
- Data load
  - Reads
  - Data Volume
  - Writes

Airtribe

Quora Case Study – Link2

- Optimising reads
  - Complex queries
    - Limit large scans using pagination
    - Instead of running complex aggregation queries, put data in another table via scripts that refresh this data at a given interval.
    - If there is no good index, or if the index does not have enough columns, or if the order of columns in the index is not optimal for the query, optimise it.
  - High Query Per Second
    - Inefficient caching as the main culprit

Airtribe

Quora Case Study – Link2

- Reducing Disk Space
  - The storage engine is responsible for how data is stored and retrieved from disk.
  - Default MySQL Storage Engine - InnoDB
  - Data is written to disk in a block of data called a page. Databases can read/write data one page at a time.
  - When you request a particular piece of data; entire page is fetched into memory.
  - With InnoDB, page sizes are fixed (default 16 KB). If the data doesn't fill the page size, then the remaining space is wasted. This can lead to extra fragmentation.
  - Meet RocksDB where you have variable page sizes. This is one of the biggest reasons RocksDB compresses better.
  - 80% reduction in space for one of their tables. Other tables saw 50-60% reductions in space.

Airtribe

Quora Case Study – Link2

- Optimising writes
  - Very few write-heavy workloads.
  - Most of them moved to HBase.
  - Replication lag between primary and secondary.
    - Parallel replication writes feature

Airtribe

Learnings

- There are physical limitations for infrastructure.
- Just because you are facing some issues with the database, you’ll not switch to another one.
- You’ll first look at how can this problem be solved.
- There is a potential answer for everything in SQL world, the question is how much time and effort are you ready to put ?
    - Compare cost of maintaining the setup vs cost of engineers.
- You question - At what stage will this become a bottleneck that can’t be solved ?

Airtribe

Thank You!

See you again on 3rd June here

Airtribe