Day 7 Database,Sharding,Partitioning, consistent hashing

Attention → what to store?

→ Data store → name, addr, payment info
→ Product data → title, desc, image, price
→ Inventory → stock level, quantity
→ Payment → Card details
→ Order history → order id, invoice, quantity, status
→ User interaction data → search, queries, product viewed

How to choose Data Store?

→ Data Model → Structure, Complexity of data
→ how to represent data
→ Scalability → Traffic, volume of data
→ Throughput, Latency
→ Consistency (ACID) (non-functional Requirement)
→ Availability
→ Cost
→ Performance → Read intensive/write intensive
→ Data security & compliance
→ Future requirements
→ Community support

Attention
1) Write down NFR, FR based on decision

Common Data Store

Relational	Key Value	Document	Wide column	Graph
Postgres	Redis	Mongo DB	Cassandra	Neo4j
Oracle DB	Memcached	Couch DB	Hbase	Dynamo DB
Maria DB		Firebase		

SQL DB

Data stored in tables

→ Key value stores
→ Redis
→ Dynamo DB
→ Riak
→ Column DB
→ Cassandra
→ H base
→ Graph DB
→ Neo4j
→ Neptune
→ Time Series DB
→ Influx DB
→ OpenTSDB

How → Make checklist & following

DB Norms | Usecase | Example Database

Availability/Consistency, Scalability

General Guidelines	SQL (Relational)	NoSQL (Non Relational)
Data model	• Structured schema • Fixed schema	• Semi-structured / unstructured • Complex • Flexible
C/A	• Strong Consistency • ACID compliant	• Eventual consistency
Relational	• SQL Queries	• Query language depends
Scalability	• Vertical (or better)	• Horizontal (or better)

Attention (*)

Horizontal Scaling vs Vertical Scaling

Vertical	Horizontal
→ Upgrade Resources	→ Complex
→ Limit	→ Data Distribution
→ Simple	→ Load Balancing
	→ Highly scalable

Order:

Scaling: → Add Resources
→ Handle Increased Load

Partitioning: → Dividing Data
→ Improving Efficiency
→ Distributing workload

Horizontal Data Partitioning
→ Rows into diff tables
→ Sharding

Vertical Data Partitioning
→ Columns into diff tables

Physical Sharding
→ Different physical servers
If server can't handle more

Logical Sharding
→ Same server
→ DB level
To improve query performance

Based on what, which DB to select?

1) Range/Hashing Sharding
→ Approach → 381
(Pros: easy, Cons: 382 Difficult to update size - unbalanced or be - Easy to update)

2) Dynamic Sharding
→ Lookup service → 381
Pros

Horizontal Partitioning

Consistent Hashing

Hash based horizontal partition done?

→ Hash based Sharding → n-servers (1/n)
→ Range based Sharding → 0-100 → 381
101-200 → 382
200 - 300 → 383

→ Geographical Sharding/ Location based → CDN, AWS, GCP
→ Random Sharding

Scaling

Cache misses

Attention (*) HELD topic

1. Hashing Ring → 270 → 360 DB

→ 54,285

(Hashing, Range)

→ n → 1-50 → 0-17
1-10 → 0-3
1-100 → 0-33

→ 1,285

Range / Hash (Hashing, Range)
1) Hash (Consistent)
2) Range (Consistent)
3) DB 360 → 0-360

Problem → we can't add more DB, which can cause problem

Now we have to redistribute the data

50 → 0-100
51 → 101-200
53 → 201-300
54 → 301-360
54 → 0-100
0-100
101-200
201-300
301-360
(+)

For adding 1 DB means we have to distribute entire data.

This is the problem we are trying to solve.

To solve the problem, we use Consistent Hashing

→ Minimum Redistribution

(Visualization)

360
{ 0-100
101-200
201-300
301-360 }

READ ABOUT CONSISTENT HASHING

→ Cassandra
→ Dynamo DB

Rest will be discussed tomorrow