What database

Tuesday, December 31, 2019 12:00 AM

With so many choices, how do you make the decision of which database to choose?

Over the hard obvious considerations

1) Data manipulation

a) Query language
- SQL
- NO SQL flow
- Graph query

b) Google spanner do give SQL query interface

c) Kaem Query Language

d) Write Processing
- Loading
- Data Export & transfer

2) ACID Properties

A : Atomicity
C : Consistency
I : Isolation
D : Durability

Atomicity
- Consistency
- Isolation
- Durability

3) Data Model

SQL | NO-SQL
- Transaction support
- Data schema
- Ordering
- Querying Queries

- Unrelated data
- Frequent updates
- Relational Ordering
- Frequent Order changes
- High Availability

4) Wire Protocol

- custom protocols over TCP/IP
- support for popular database protocols
  - Postgres : Cockroachdb
  - Mongodb : FerretDB
- standard interface
  - JDBC | ODBC | Mongo, Redis, Elastic Search

5) Schema(less Support)

- very flexible schemas where on read
- Only schema for indexing
  - things has hierarchical data schema support

- schema is important to understand structure of data.
- Read schema flexible & easier if there is schema interface
- In between, Systems
  - Indexes, but require schema indexes (e.g. Postgres Binary Jsonb for KV stores) Cassandra

Performance Scalability Workload

1) Key value data (LSM environment)
Pattern why: SSTable
- unstructured
- very
- moderate read throughput
- high latency read.

2) Document
- Flexible schemas
- Not structured
- Not environment variables, templates
- Logs
- high read throughput
- high latency reads

3) Wide column
- Time series
- Sensor data
- high read throughput
- high latency reads

4) Graph
- Social graphs
- Recommendation engines
- Fraud detection
- Knowledge graphs
- high latency random search

5) Time series
- Sensor data
- Financial data
- Monitoring
- Analytics
- Log analytics
- Real time analytics
- Time series analytics
- Low latency indexed reads
- Low latency random search

6) Object
- Unstructured
- Not environment variables, templates
- Logs
- high read throughput
- high latency reads

7) Search
- Full text search
- Log analytics
- Product catalogs
- Recommendations
- Low latency indexed reads
- Low latency random search

8) Multi-model
- Flexible schemas
- Not structured
- Not environment variables, templates
- Logs
- high read throughput
- high latency reads

9) Relational
- Structured
- Not environment variables, templates
- Logs
- high read throughput
- high latency reads

10) Vector
- Semantic search
- Recommendations
- Image search
- Chatbots
- Low latency indexed reads
- Low latency random search

11) Spatial
- Location based services
- Geospatial analytics
- Mapping
- Low latency indexed reads
- Low latency random search

12) Ledger
- Financial transactions
- Audit logs
- Immutable records
- Low latency indexed reads
- Low latency random search

13) Content
- Text search
- Log analytics
- Product catalogs
- Recommendations
- Low latency indexed reads
- Low latency random search

14) Blockchain
- Decentralized applications
- Smart contracts
- Digital assets
- Low latency indexed reads
- Low latency random search

15) Multi-model authentication and packets
- Not structured
- Not environment variables, templates
- Logs
- high read throughput
- high latency reads

16) Relational
- Structured
- Not environment variables, templates
- Logs
- high read throughput
- high latency reads

17) Vector
- Semantic search
- Recommendations
- Image search
- Chatbots
- Low latency indexed reads
- Low latency random search

18) Spatial
- Location based services
- Geospatial analytics
- Mapping
- Low latency indexed reads
- Low latency random search

19) Ledger
- Financial transactions
- Audit logs
- Immutable records
- Low latency indexed reads
- Low latency random search

20) Content
- Text search
- Log analytics
- Product catalogs
- Recommendations
- Low latency indexed reads
- Low latency random search

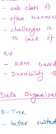
21) Blockchain
- Decentralized applications
- Smart contracts
- Digital assets
- Low latency indexed reads
- Low latency random search

22) CSRS
C - command
Q - Query
R - Responsibility
S - Separation

Reads & writes separate at

1) Application layer
- ask materialized views for reads & writes independently

2) Database layer



UNDER THE HOOD CONSIDERATIONS

3) Performance Benchmarks
- latency throughput concurrency
- speed ups & (dis)advantages
- always looked for independent perf benchmarks
- Twitter read myself instead of Graph10
- Build dashboard TPS on top of MySQL
- support (free, paid, community)
- hosted service: on prem, saas (Dynamo DB)

4) Database Type

a) Relational
- All field values stored together
- B+Tree
- Suited for OLTP (Online Txn Processing)

b) Columnar
- All values of column stored together
- More efficient data compression
- OLAP queries perform better (online analytical processing)

c) Document
- sub data of KV store
- often hierarchical (JSON collection → Document)
- challenges in optimizing storage due to lack of Schemas

d) KV
- RAM based
- Durability through replication & disk sync

5) Data Organization

a) B-Tree
- better suited for in-place updates

b) Log Structured Merged (LSM)
- better for high write throughput
- achieve high data locality of reference

6) Replication (CAP Theorem)

a) Replication:
- sync vs Async
- no of replicas, wan replica, Journaling,
- Single master read-write (CP) vs Replica reads (AP)

b) Consensus:
- leader election
- committing txn, log replication
- Jepsen, Raft etc.

c) Jepsen Tests
- not safety of distributed database.

7) Operations

a) Data Export & restore
- full vs incremental snapshots
- tools for moving data around
- large scale data migration
  - pitfalls: resend data in a moment

b) Mean Time to Recovery
- provisioning not standby to master
- tools to detect failure, validate data, promote read replica.

8) Cost

a) Total Memory Ratio
- NFS makes his memory get up to date stored on disk than needs to read needs to hold memory data.

Design a scalable analytics system for cloud operate
Tech stack choices
Some ideas:

- Spark (Languages: SQL | Affine Flexibility)
- Kaem (support for Batch operations | Kaem Streaming, Kaem Analytics)
- Kaem Streaming (Kaem Analytics)
- Presto (ORC protocol)
- Hive (Hive Metastore Compliant | Hive Column Family)
- Drug (with Drug Batch logging Batch Write)
- Drug (Drug Batch logging Batch Write)
- Presto (ORC structure since I read faster than Parquet)
- Presto (Presto Connector Plugin)
- Drug with (without Drug Batch write)
- Data Export and restore are essential features
- Data Export and restore are essential
- Kaem data older than 3 months on S3 (either Glacier or AWS)

Data

a) partition work by time chunks

b) Keep data in memory and sort on disk.

b) Flush final data to secondary storage at frequent intervals.

- Reader will load this data from S3 and will create indexes.
- I don't know which data resides where?
- Add coordinator to manage active nodes & their tasks
- Add Query broker to split the task query to writer & reader
- Add zookeeper to store data mapping

Data

Reader