DOCUMENT DATABASES

07/01/2024

SANDUNI AALOKA

CONTENT

- RDBMS Characteristics
- ACID
- NoSQL
- Types of NoSQL Databases
- Scalability
- BASE
- CAP
- Aggregate Data Model

RDBMS CHARACTERISTICS

- Data stored in columns and tables
- Relationships represented by data
- Data Manipulation Language & Data Definition Language
- Transactions
- Abstraction from physical layer
- Physical layer can change without modifying applications
  - Create indexes to support queries
  - In Memory databases

ACID

- **Atomic**

All of the work in a transaction completes (commit) or none of it completes.

- **Consistent**

A transaction transforms the database from one consistent state to another consistent state.

- **Isolated**

The results of any changes made during a transaction are not visible until the transaction has committed.

- **Durable**

The results of a committed transaction survive failures.

NOSQL

- NoSQL stands for,
  - No Relational
  - No RDBMS
  - Not Only SQL
- NoSQL is an umbrella term for all databases and data stores that don’t follow the RDBMS principles.
  - A class of products
  - A collection of several (related) concepts about data storage and manipulation
  - Often related to large data sets

NOSQL

- Non-relational DBMSs are not new.
- But NoSQL represents a new incarnation
  - Due to massively scalable Internet applications
  - Based on distributed and parallel computing
- Development
  - Starts with Google
  - The first research paper was published in 2003
  - Progress was further driven by contributions from Lucene's developers, Apache (Hadoop) and Amazon (Dynamo).
  - Then a lot of products and interests came from Facebook, Netflix, Yahoo, eBay, Hulu, IBM, and many more

NOSQL

- NoSQL comes from Internet, thus it is often related to the “big data” concept.
- How much big are “big data”?
  - Over few terabytes enough to start spanning multiple storage units.
- Challenges
  - Efficiently storing and accessing large amounts of data is difficult, even more considering fault tolerance and backups.
  - Manipulating large data sets involves running immensely parallel processes.
  - Managing continuously evolving schema and metadata for semi-structured and un-structured data is difficult.

WHY RDBMS IS NOT SUITABLE FOR BIGDATA?

- The context is Internet.
- RDBMSs assume that data are,
  - Dense
  - Largely uniform (structured data)
- Data coming from Internet are,
  - Massive and sparse
  - Semi-structured or unstructured
- With massive sparse data sets, the typical storage mechanisms and access methods get stretched.

TYPES OF NOSQL DATABASES

- **Stored Ordered Column Store**

Optimized for queries over large datasets, and stores columns of data together, instead of rows.

- **Key-Value Store**

The simplest NoSQL databases. Every single item in the database is stored as an attribute name (or 'key'), together with its value.

- **Graph Databases**

Used to store information about networks of data, such as social connections.

- **Document Databases**

Pair each key with a complex data structure known as a document.

SORTED ORDERED COLUMN ORIENTED STORES

- Data is efficiently stored in a column-oriented way.
- Avoid consuming space for strong nulls.
- Data isn’t stored as a single table but is stored by column families.
- Unit of data is a set of key/value pairs,
    - Identified by “row key”.
    - Ordered and sorted based on row-key.

ex: Google’s Bigtable(used in all Google’s services)
HBase(Facebook, StumbleUpon, Hulu, Yahoo)

KEY-VALUE STORES

- Store data in a schema-less way.
- Store data as maps
  - HashMaps or associative arrays.
  - Provide a very efficient average running time algorithm for accessing data.

ex: Couchbase (Zynga, Vimeo, NAVTEQ, ...)
Redis (Craiglist, Instagram, StackOverfow, flicker, ...)
Amazon Dynamo (Amazon, Elsevier, IMDb, ...)
Apache Cassandra (Facebook, Digg, Reddit, Twitter,...)
Voldemort (LinkedIn, eBay, ...)
Riak (Github, Comcast, Mochi, ...)

GRAPH DATABASES

- Graph-oriented.
- Everything is stored as an edge or a node.
- Both the nodes and edges can be labeled.
- Each node can have any number of attributes.
- Edges store information about the relationship between nodes.

DOCUMENT DATABASES

- Documents
  - Loosely structured sets of key/value pairs in documents, e.g., XML, JSON.
  - Encapsulate and encode data in some standard formats or encodings.
  - Are addressed in the database via a unique key.
  - Documents are treated as a whole, avoiding splitting a document into its constituent name/value pairs.
- Allow documents retrieving by keys or contents.

ex: MongoDB (used in FourSquare, Github, and more)
CouchDB (used in Apple, BBC, Canonical, Cern, and more)

DOCUMENT DATABASES

- The central concept is the notion of a "document“ which corresponds to a row in RDBMS.
- A document comes in some standard formats like JSON (BSON).
- Documents are addressed in the database via a unique key that represents that document.
- The database offers an API or query language that retrieves documents based on their contents.
- Documents are schema free, i.e., different documents can have structures and schema that differ from one another. (An RDBMS requires that each row contain the same columns.)

JSON

- JavaScript Object Notation
- A text format for storing and transporting data
  - Light weight
  - Plain text
- "Self-describing" and easy to understand.
- Language independent

```
{
  "_id":
      ObjectId("5ad88534e3632e1a35a58d00"),
  "name": {
    "first": "John",
    "last": "Doe" },
  "address": [
    { "location": "work",
      "address": {
        "street": "16 Hatfields",
        "city": "London",
        "postal_code": "SE1 8DJ"},
      "geo": { "type": "Point", "coord": [
          51.5065752,-0.109081]}},
+   {...}
  ],
  "phone": [
    { "location": "work",
      "number": "+44-1234567890"},
+   {...}
  ],
  "dob": ISODate("1977-04-01T05:00:00Z"),
  "retirement_fund":
        NumberDecimal("1292815.75")
}
```


NOSQL DISTINGUISHING CHARACTERISTICS

- Large data volumes
  - Google's "big data"
- Scalable replication and distribution
  - Potentially thousands of machines
  - Potentially distributed around the world
- Queries need to return answers quickly
- Mostly query, few updates
- Asynchronous Inserts & Updates
- Schema-less
- ACID transaction properties are not needed – BASE
- CAP Theorem
- Open source development

SCALABILITY

- Issues with scaling up when the dataset is just too big.
- RDBMS were not designed to be distributed.
- Traditional DBMSs are best designed to run well on a “single” machine.
  - Larger volumes of data/operations requires to upgrade the server with faster CPUs or more memory known as ‘scaling up’ or ‘Vertical scaling’
- NoSQL solutions are designed to run on clusters or multi-node database solutions.
  - Larger volumes of data/operations requires to add more machines to the cluster, Known as ‘scaling out’ or ‘horizontal scaling’.
  - Different approaches include,
    - Master-slave
    - Sharding (partitioning)

SCALABILITY

- **Master-Slave**
  - All writes are written to the master. All reads performed against the replicated slave databases.
  - Critical reads may be incorrect as writes may not have been propagated down.
  - Large data sets can pose problems as master needs to duplicate data.
- **Sharding**
  - Any DB distributed across multiple machines needs to know in what machine a piece of data is stored or must be stored.
  - A sharding system makes this decision for each row, using its key.

SCALABILITY

Sharding : Pros & Con

- **Pros**
  - Increased read/write throughput.
  - Increased storage capacity.
  - High availability.
- **Cons**
  - Query overhead
    - Each sharded database must have a separate machine or service which understands how to route a querying operation to the appropriate shard. This introduces additional latency on every operation.
  - Complexity of administration.
  - Increased infrastructure costs.

NOSQL - NO ACID

- RDBMSs are based on ACID (Atomicity, Consistency, Isolation, and Durability) properties.
- NoSQL
  - Does not give importance to ACID properties.
  - In some cases completely ignores them.
- In distributed parallel systems it is difficult/impossible to ensure ACID properties.
  - Long-running transactions don't work because keeping resources blocked for a long time is not practical.

BASE

• **Basically Available**

Rather than enforcing immediate consistency, BASE-modelled NoSQL databases will ensure availability of data by spreading and replicating it across the nodes of the database cluster.

• **Soft state**

Due to the lack of immediate consistency, data values may change over time. The BASE model breaks off with the concept of a database which enforces its own consistency, delegating that responsibility to developers.

• **Eventually consistent**

The fact that BASE does not enforce immediate consistency does not mean that it never achieves it. However, until it does, data reads are still possible (even though they might not reflect the reality).

BASE TRANSACTION CHARACTERISTICS

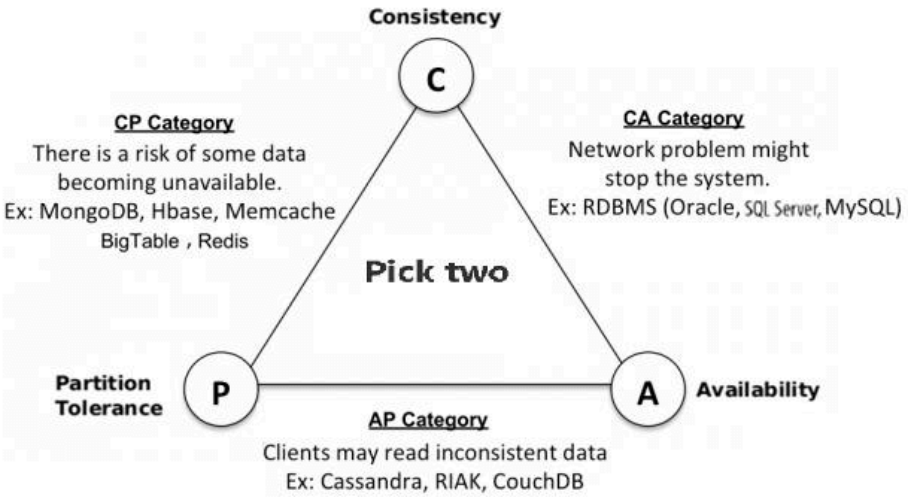
• Weak consistency – stale data OK
• Availability first
• Best effort
• Approximate answers OK
• Aggressive (optimistic)
• Simpler and faster

CAP THEOREM

In theoretical computer science, the CAP theorem, also named “Brewer's theorem” after computer scientist Eric Brewer, states that any distributed data store can provide only two of the following three guarantees.

- **Consistency**
  - Every read receives the most recent write or an error.
- **Availability**
  - Every request receives a (non-error) response, without the guarantee that it contains the most recent write.
- **Partition tolerance**
  - The system continues to operate despite an arbitrary number of messages being dropped (or delayed) by the network between nodes.

CAP THEOREM



CAP THEOREM

Consistency or Availability

- Consistency and Availability is not “binary” decision.
- AP systems relax consistency in favor of availability – but are not inconsistent.
- CP systems sacrifice availability for consistency- but are not unavailable.
- This suggests both AP and CP systems can offer a degree of consistency, and availability, as well as partition tolerance.

PERFORMANCE

- There is no perfect NoSQL database.
- Every database has its advantages and disadvantages.
  - Depending on the type of tasks (and preferences) to accomplish.
- NoSQL is a set of concepts, ideas, technologies, and software dealing with,
  - Big data
  - Sparse un/semi-structured data
  - High horizontal scalability
  - Massive parallel processing
- Different applications, goals, targets, approaches need different NoSQL solutions.

USE OF NOSQL DATA

Do you have somewhere a large set of uncontrolled, unstructured, data that you are trying to fit into a RDBMS?

- Log Analysis.
- Social Networking Feeds (many firms hooked in through Facebook or Twitter).
- External feeds from partners.
- Data that is not easily analyzed in a RDBMS such as time-based data.
- Large data feeds that need to be massaged before entry into an RDBMS.

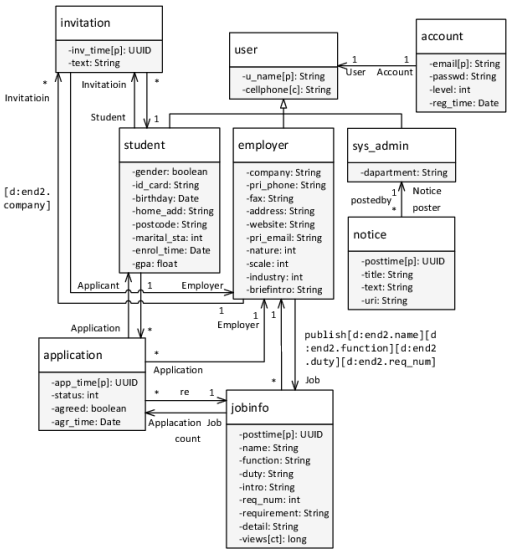
DATABASE ADMINISTRATION

- It does not matter if the data is deployed on a NoSQL platform instead of an RDBMS.
- Still need to address,
  - Backups & recovery
  - Capacity planning
  - Performance monitoring
  - Data integration
  - Tuning & optimization

APPLICATION INTEGRATION

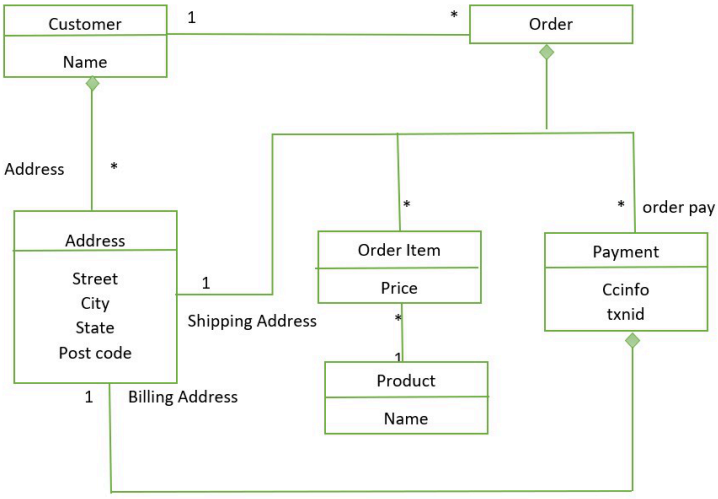
- **Using Databases**
  - This method is used to integrate into RDMS applications.
  - A structure designed to integrate **multiple applications** is complex.
  - Different applications are **developed by separate teams**. They need to coordinate when making changes to the data storage.
  - Application may **store common data**. Optimizing one may affect the other's performance.
- **Using Web services**
  - Encapsulating databases within applications and integrating through services.
- **Using Events**

DATA MODEL



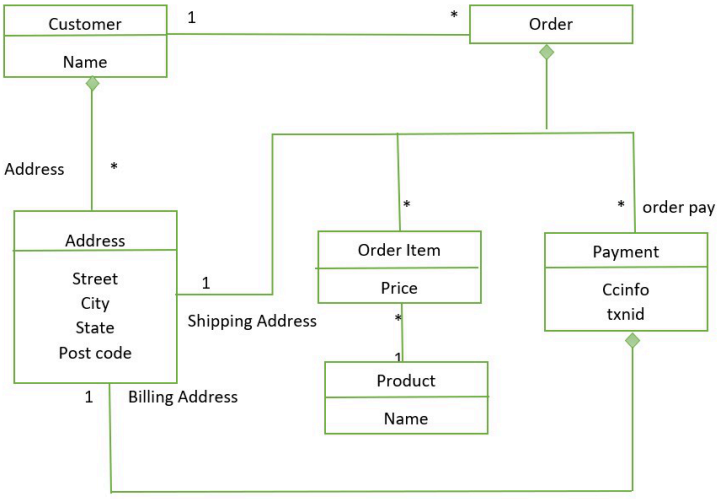
- Data is organized in the database according to a metamodel.
- The dominant data model in the last couple of decades is the relational data model.
- Relational model best visualized as a set of tables, rather like a page of spreadsheet.

AGGREGATE DATA MODEL



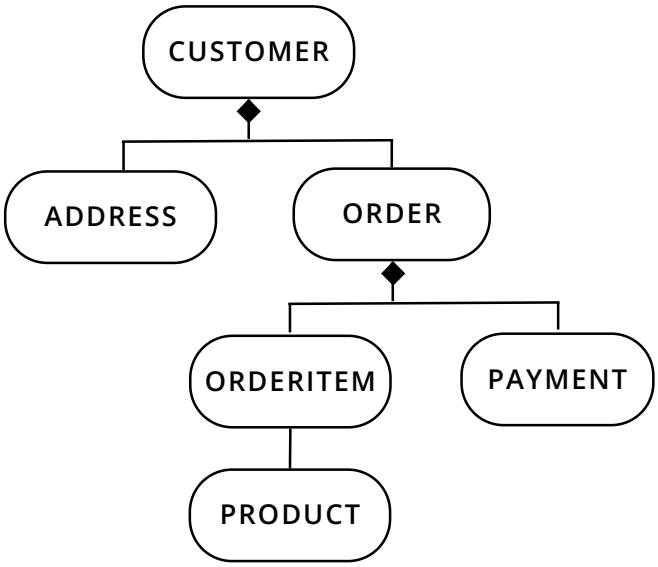
- This shows an online store case study.
- NoSQL are databases store data in another format and uses the Aggregate Data model.
- An aggregate is a collection of data that we interact with as a unit.

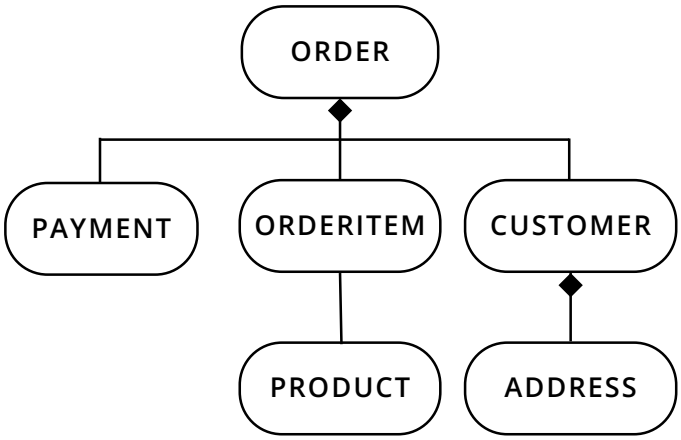
AGGREGATE DATA MODEL



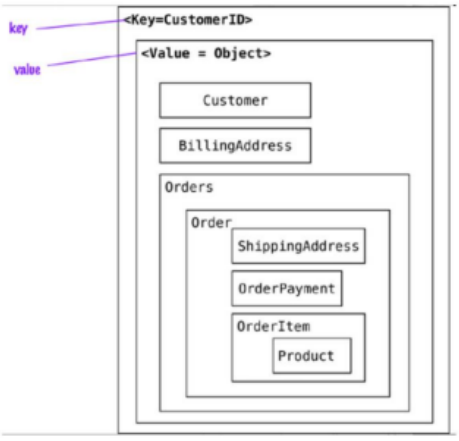
- Has two main aggregates – customer and order.
- The customer contains data related to billing addresses
- The order aggregate consists of ordered items, shipping addresses, and payments.
- The payment also contains the billing address.

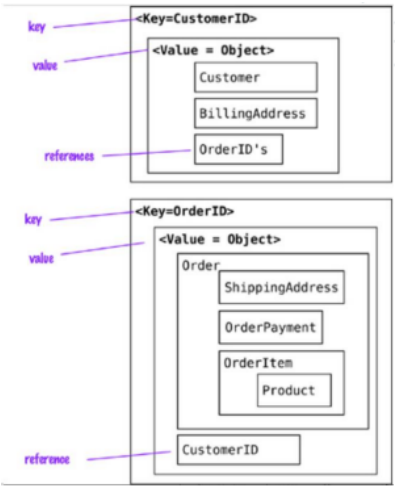
AGGREGATE DATA MODEL





AGGREGATE DATA MODEL





AGGREGATE DATA MODEL

```
{
  "customer": {
    "id": 1,
    "name": "Martin",
    "billingAddress": [{"city": "Chicago"}],
    "orders": [
      {
        "id": 99,
        "customerId": 1,
        "orderItems": [{
            "productId": 27,
            "price": 32.45,
            "productName": "NoSQL Distilled"}
        ],
        "shippingAddress": [
          {"city": "Chicago"}
        ],
        "orderPayment": [{
          "ccinfo": "1000-1000-1000-1000",
          "txnId": "abelif879rft",
          "billingAddress":{
            "city": "Chicago"}}]
      }
    ]
  }
}
```

```
{
  "order": {
    "id": 99,
    "customer": {
      "customerId": 1,
      "name": "Martin",
      "billingAddress": [{"city": "Chicago"}]
    },
    "orderItems": [{
            "productId": 27,
            "price": 32.45,
            "productName": "NoSQL Distilled"
    }],
    "shippingAddress": [{
      "city": "Chicago"
    }],
    "orderPayment": [{
      "ccinfo": "1000-1000-1000-1000",
      "txnId": "abelif879rft",
      "billingAddress":{
          "city": "Chicago"
      }}]
  }
}
```


CONSEQUENCES OF AGGREGATE DATA MODEL

- Good on cluster
- No ACID with multiple aggregates - We are able to preserve atomicity with one aggregate.
- Often difficult to draw aggregate boundaries.