NoSQL

Saturday, June 15, 2024 6:29 PM

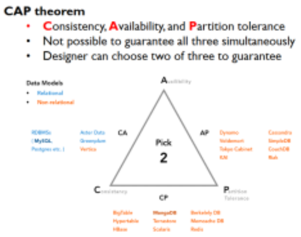
AKA: non-SQL or not only SQL

Four main types:

1- Document
2- Key Value
3- Wide Column
4- Graph

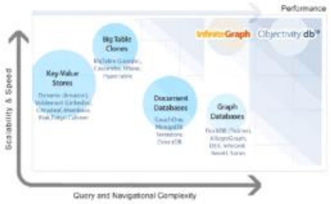
General NoSQL info:

- Database is distributed rather than centralized on a single server
  - Scales horizontally over many servers (DB "replicas/shards")
- Designed to handle "big data" (large volume/variety/velocity)
-



CAP theorem

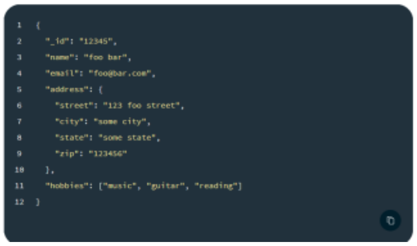
- Consistency, Availability, and Partition tolerance
- Not possible to guarantee all three simultaneously
- Designer can choose two of three to guarantee



1 - Document

- Stores data similar to JSON
- Suited for semi/unstructured data

EX:

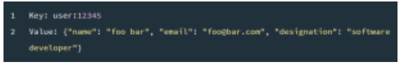


```
{
  "_id": "12345",
  "name": "foo bar",
  "email": "foo@bar.com",
  "address": {
    "street": "123 foo street",
    "city": "some city",
    "state": "some state",
    "zip": "123456"
  },
  "hobbies": ["music", "guitar", "reading"]
}
```

2 - Key Value

- Use key to access value (like python dictionary or java map)
- Provide high performance for reads and writes (due to caching)

EX:



```
Key: user:12345
Value: {"name": "foo bar", "email": "foo@bar.com", "designation": "software developer"}
```

3 - Wide Column

- Stores data in tables (rows and "dynamic" columns)
- Dynamic Columns: different rows can have different sets of columns
  - Column compression techniques reduce storage space and increase performance (especially for sparse and wide data)
  - Allows for a flexable DB schema (adaptable to changing data requirements)

Certainly! Let's use an example of tracking user interactions with products on an e -commerce platform to show how the same information is stored in both a relational database and a wide column NoSQL database, highlighting the advantages of the wide column approach.

EX:

We have the following information:
- Users: Alice and Bob
- Products: Widget A and Gadget B
- User interactions with products (e.g., views, purchases)

Visual Comparison

Relational Database (Tables):

1. Users Table:
   - user_id (Primary Key)
   - name
   - email
   - address

2. Products Table:
   - product_id (Primary Key)
   - name
   - price
   - category

3. Interactions Table:
   - interaction_id (Primary Key)
   - user_id (Foreign Key)
   - product_id (Foreign Key)
   - timestamp
   - action

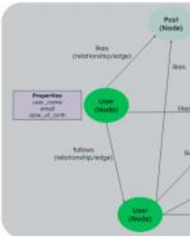
Wide Column NoSQL Database (Column Family):

1. UserInteractions Column Family:
   - Row Key: user_id
   - Columns: Each column is a combination of timestamp, productID, and action

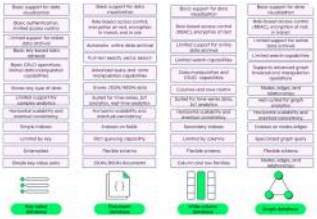
4 - Graph

- Stores data as nodes and edges (connects nodes)
  - Nodes = nouns
  - Edges = relationships between nouns
- Works well for highly connected data

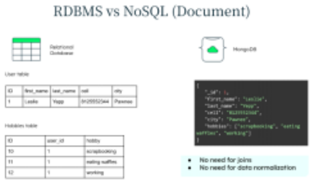
EX:



Comparison:



Basic support for data visualization	Basic support for data visualization	Basic support for data visualization	Basic support for data visualization
Basic authentication, limited access control	Role-based access control, encryption at rest, encryption in transit, and in use	Role-based access control (RBAC), encryption at rest	Role-based access control (RBAC), encryption at rest (in beta)
Limited support for online data archival	Automatic online data archival	Limited support for online data archival	Limited support for online data archival
Basic key-based data retrieval	Full-text search, vector search	Limited search capabilities	Limited search capabilities
Basic CRUD operations, limited data manipulation capabilities	Advanced query and data manipulation capabilities	Data manipulation and CRUD capabilities	Supports advanced graph traversal and manipulation operations
Stores any type of data	Stores JSON/BSON data	Columns and rows matrix	Nodes, edges, and relationships
Limited support for complex analytics	Suited for time-series, IoT analytics, real-time analytics	Suited for time-series data, IoT analytics	Not suited for graph analytics
Horizontal scalability and eventual consistency	Horizontal scalability and eventual consistency	Horizontal scalability and eventual consistency	Horizontal scalability and eventual consistency
Simple indexes	Indexes on fields	Secondary indexes	Indexes on nodes/edges
Limited by key	Rich querying capability	Limited by columns	Specialized graph query
Schemaless	Flexible schema	Flexible schema	Flexible schema
Simple key-value pairs	JSON/BSON documents	Column and row families	Nodes, edges, and relationships
Key-value database	Document database	Wide-column database	Graph database



RDBMS vs NoSQL (Document)

Relational Database

User table

ID	first_name	last_name	cell	city
1	Leslie	Yepp	8125552344	Pawnee

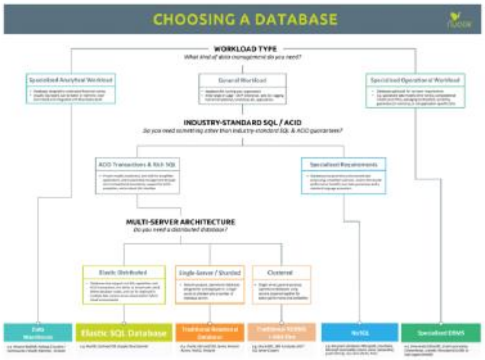
Hobbies table

ID	user_id	hobby
10	1	scrapbooking
11	1	eating waffles
12	1	working

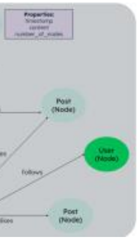
MongoDB

```
{
  "_id": 1,
  "first_name": "Leslie",
  "last_name": "Yepp",
  "cell": "8125552344",
  "city": "Pawnee",
  "hobbies": ["scrapbooking", "eating
waffles", "working"]
}
```

- No need for joins
- No need for data normalization



CHOOSING A DATABASE
WORKLOAD TYPE
What kind of data management do you need?
Specialized Analytical Workload
General Workload
Specialized Operational Workload
INDUSTRY-STANDARD SQL / ACID
Do you need something other than industry-standard SQL & ACID guarantees?
ACID Transactions & Rich SQL
Specialized Requirements
MULTI-SERVER ARCHITECTURE
Do you need a distributed database?
Elastic Distributed
Single-Server / Sharded
Clustered
Data Warehouse
Elastic SQL Database
Traditional Relational Database
Traditional RDBMS Label Plus
NoSQL
Specialized DBMS


Properties:
timestamp
content
number_of_likes
Post
(Node)
es
Follows
User
(Node)
Post
(Node)
likes