NoSQL
HOW TO WRITE A CV

DO YOU HAVE ANY EXPERTISE IN SQL?

NO

DOESN'T MATTER. WRITE: "EXPERT IN NO SQL"

Leverage the NoSQL boom
What is NoSQL?

• Not only SQL
• SQL means
  ▪ Relational model
  ▪ Strong typing
  ▪ ACID compliance
  ▪ Normalization
  ▪ ...
• NoSQL means more freedom or flexibility
Relevance to Big Data

• Data gets bigger
• Traditional RDBMS cannot scale well
• RDBMS is tied to its data and query processing models
• NoSQL relaxes some of the restrictions of RDBMS to provide a better performance
Advantages of NoSQL

• Handles Big Data
• Data Models – No predefined schema
• Data Structure – NoSQL handles semi-structured data
• Cheaper to manage
• Scaling – Scale out / horizontal scaling
Advantages of RDBMS

• Better for relational data
• Data normalization
• Well-established query language (SQL)
• Data Integrity
• ACID Compliance
Types of NoSQL Databases

- Document Databases [MongoDB, CouchDB]
- Column Databases [Apache Cassandra]
- Key-Value Stores [Redis, Couchbase Server]
- Cache Systems [Redis, Memcached]
- Graph Databases [Neo4J]
- Streaming Systems [FlinkDB, Storm]
Document Database
Document Data Model

• Relational model (RDBMS)
  ▪ Database
    o Relation (Table) : Schema
      - Record (Tuple) : Data

• Document Model
  ▪ Database
    o Collection : No predefined schema
      - Document : Schema+data

• No need to define/update schema
• No need to create collections

Document 1

```json
{ "id": 1, "name":"Jack", "email": "jack@example.com", "address": {"street": "900 university ave", "city": "Riverside", state: "CA"}, "friend_ids": [3, 55, 123]}
```
Document Format

• MongoDB natively works with JSON documents
• For efficiency, documents are stored in a binary format called BSON (i.e., binary JSON)
• Like JSON, both schema and data are stored in each document
How to Use MongoDB

Install: Check the MongoDB website
https://docs.mongodb.com/manual/installation/

Create collection and insert a document

```
db.users.insert({name: "Jack", email: "jack@example.com"});
```

Retrieve all/some documents

```
db.users.find();
db.users.find({name: "Jack"});
```

Update

```
updateOne, updateMany, replaceOne
```

Delete

```
deleteOne, deleteMany
```

https://docs.mongodb.com/manual/crud/
Schema Validation

• You can still explicitly create collections and enforce schema validation

```javascript
db.createCollection("students", {
  validator: { $jsonSchema: {
    bsonType: "object",
    required: [ "name", "year", "major", "address" ],
    properties: {
      name: {
        bsonType: "string",
        description: "must be a string and is required"
      },
      ...
    }
  }
});
```

https://docs.mongodb.com/manual/core/schema-validation/
Storage Layer

• Prior to MongoDB 3.2, only B-tree was available in the storage layer
• To increase its scalability, MongoDB added LSM Tree in later versions after it acquired WiredTiger

Override default configuration

```bash
mongod --wiredTigerIndexConfigString "type=lsm,block_compressor=zlib"
```
LSM Vs B-tree

https://github.com/wiredtiger/wiredtiger/wiki/Btree-vs-LSM
Indexing

• Like RDBMS, document databases use indexes to speed up some queries

```
Collection

db.users.find({ score: { "$lt": 30 } }).sort({ score: -1 })
```

• MongoDB uses B-tree as an index structure

https://docs.mongodb.com/manual/indexes/
Index Types

- Default unique `_id` index
- Single field index
  - `db.collection.createIndex({name: -1})`;
- Compound index (multiple fields)
  - `db.collection.createIndex( { name: 1, score: -1})`;
- Multikey indexes (for array fields)
  - Creates an index entry for each value

https://docs.mongodb.com/manual/indexes/
Index Types

• Geospatial index (for geospatial points)
  ▪ Uses geohash to convert two dimensions to one dimension
  ▪ 2d indexes: For Euclidean spaces
  ▪ 2d sphere: spherical (earth) geometry
  ▪ Works with multikey indexes for multiple locations (e.g., pickup and dropoff locations for taxis)

• Text Indexes (for string fields)
  ▪ Automatically removes stop words
  ▪ Stems the works to store the root only

• Hashed Indexes (for point lookups)
Additional Index Features

• Unique indexes: Rejects duplicate keys
• Sparse Indexes: Skips documents without the index field
  ▪ In contrast, non-sparse indexes assume a null value if the index field does not exist
• Partial indexes: Indexes only a subset of records based on a filter.

```javascript
db.restaurants.createIndex(
  { cuisine: 1, name: 1 },
  { partialFilterExpression: { rating: { $gt: 5 } } }
)
```
Distributed Processing

• Two methods for distributed processing
  ▪ Replication (Similar to MySQL)
  ▪ Sharding (True horizontal scaling)

Replication
https://docs.mongodb.com/manual/replication/

Sharding
https://docs.mongodb.com/manual/sharding/
Comparison of data types

- Min key (internal type)
- Null
- Numbers (32-bit integer, 64-bit integer, double)
- Symbol, String
- Object
- Array
- Binary data
- Object ID
- Boolean
- Date, timestamp
- Regular expression
- Max key (internal type)

https://docs.mongodb.com/v3.6/reference/bson-type-comparison-order/
Comparison of data types

• Numbers: All converted to a common type
• Strings
  ▪ Alphabetically (default)
  ▪ Collation (i.e., locale and language)
• Arrays
  ▪ <: Smallest value of the array
  ▪ >: Largest value of the array
  ▪ Empty arrays are treated as null
• Object
  ▪ Compare fields in the order of appearance
  ▪ Compare <name,value> for each field