Chapter 24

NOSQL Databases and Big Data Storage Systems

- Large amounts of data such as social media, Web links, user profiles, marketing and sales, posts and tweets, road maps, spatial data, email
- NOSQL not only SQL
- Most NOSQL systems are distributed database or storage systems with focus on semi-structured data, high performance, availability, data replication and scalability. This is different from emphasis on data consistency, powerful query language and structured data storage in traditional DBMS.

1. Introduction to NOSQL systems

- Millions of users and thousands of emails
- Large amounts of data, images, video and text, Facebook, Youtub etc..
- Traditional DBMSs are not appropriate as the mission is different
- Traditional DBMSs require schema, structured data
- Organizations faced with big data developed their own storage systems
- Bigtable (Google), Dynamo DB (Amazon), Cassandra (Facebook), Mongodb, CouchDB, Neo4 and GaphBase, OrientDB, XML

Characteristics of NOSQL systems:
- Related to distributed database and distributed systems
- Related to data models and query languages
Distributed databases and distributed systems

- **Scalability**: volume grows, horizontal scalability; add more nodes; vertical scalability-add more processors and memory
- **Availability**: replication and eventual consistency: continuous system availability, data replicated over two or more nodes; no need for serializability
- **Replication models**
- **Sharding** of files: sharding known as horizontal partitioning is used to store large amount of data; load balancing and data availability
- **High performance** data access: individual data items need to be found in a millions of records; range partitioning and hashing needed

Data models and query languages

- **Not requiring a schema**: semi-structured, self-describing data; usually schema is not required in most of applications
- **Less powerful query language**: no need for SQL, need to locate objects in a single file; provide set of functions and operations as an API; these operations are called CRUD operations (create, read, update and delete). Also known as SCRUD due to search or find operations. Some languages provide a subset of SQL.
- **Versioning**: storage of multiple versions of the data items with timestamps

Categories of NOSQL systems

1. **Document-based NOSQL systems**

Store data in the form of documents; JSON notation accessed via document id or indexes
2. NOSQL key-value stores

Key and a value access

3. Column-based or wide column NOSQL systems

Each column stored in its own file

4. Graph-based NOSQL systems

Data is represented in graphs; related nodes can be found by traversing the edges using path expressions

Other categories:

1. Hybrid NOSQL: mix of the above 4 categories
2. Object databases
3. XML databases

2. The CAP Theorem

- Distributed DBMS need to enforce ACID properties
- It is hard to implement ACID properties in NOSQL databases
- Serialization is hard to enforce in NOSQL databases
- Replication in NOSQL databases poses challenges in implementing serializability
- The CAP: consistency (among replicated copies), availability (of the system for read and write), partitioning tolerance (in the face of the nodes in the system being partitioned by a network fault). The consistency means, the same data on all copies available for transactions; availability implies, operation is successful or sends a message indicating failure; partitioning implies that the system can continue if there is a fault that results in two or more portioning of the network
- It is not possible to guarantee CAP at the same time while having replication. A weaker consistency in NOSQL is acceptable for some applications. Eventual consistency is used some times to satisfy CAP.

3. **Document-based NOSQL Systems and MongoDB**
   - Store data as collections of similar documents; also known as document stores
   - Resemble complex objects or XML documents
   - No schema required
   - Self-describing data
   - Documents in a collection should be similar, they can have different data elements (attributes)
   - Users can request to create indexes on some of the data elements
   - Documents can be specified in JSON, XML, ...
   - There are many systems; MongoDB, CouchDB,....

**MongoDB Data Model**

- Documents are stored in BSON (binary JSON) format, more efficient than JSON
- Individual documents are stored in a collection
- OID unique similar to a key, can be specified by the user or system generated
- Collection has no schema

```javascript
db.createCollection("project", {capped:true, size:1310720, max:500})
db.createCollection("worker", {capped:true, size:5242880, max:2000})
```

**Fig. 24.1**
(a) project document with an array of embedded workers:
{
    _id: "P1",
Pname: "ProductX",
Plocation: "Bellaire",
Workers: [
    {
        Ename: "John Smith",
        Hours: 32.5
    },
    {
        Ename: "Joyce English",
        Hours: 20.0
    }
]
};
Figure 24.1b  Example of simple documents in MongoDB. Embedded array of document references.

(b) project document with an embedded array of worker ids:

```json
{
    _id: "P1",
    Pname: "ProductX",
    Plocation: "Bellaire",
    WorkerIds: [ "W1", "W2" ]
}
{
    _id: "W1",
    Ename: "John Smith",
    Hours: 32.5
}
{
    _id: "W2",
    Ename: "Joyce English",
    Hours: 20.0
}
```
Figure 24.1c  Example of simple documents in MongoDB. Normalized documents.

(c) normalized project and worker documents (not a fully normalized design for M:N relationships):

```
{  
    _id: "P1",
    Pname: "ProductX",
    Plocation: "Bellaire"
}
{  
    _id: "W1",
    Ename: "John Smith",
    ProjectId: "P1",
    Hours: 32.5
}
{  
    _id: "W2",
    Ename: "Joyce English",
    ProjectId: "P1",
    Hours: 20.0
}
```
(d) inserting the documents in (c) into their collections “project” and “worker”:

db.project.insert( { _id: “P1”, Pname: “ProductX”, Plocation: “Bellaire” } )
db.worker.insert( [ { _id: “W1”, Ename: “John Smith”, ProjectId: “P1”, Hours: 32.5 },

- Workers is embedded in project (complex object), no need for a Worker collection—known as denormalized pattern, [ ] is a list (Fig. 24.1.a)
- Fig. 24.1.b worker references are embedded in the project; WokerIds; separate collection
- Fig. 24.1.c normalized design
- The choice of design depends upon the need for accessing data
**MongoDB CRUD Operations**

db.<collection_name>.insert<document(s)>)

db.<collection_name>.remove<condition>)

db.<collection_name>.find<condition>)

- One or more documents can be inserted
- Any Boolean condition can be used

**MongoDB Distributed System Characteristics**

- Updates are atomic if they refer to a single document
- Transactions on multiple documents
- It is a distributed system and uses 2PC
- Replication in MongoDB: replica set is used to create multiple copies on different nodes; uses master-slave approach; total no of participants must be at least 3
- Sharding: divides the document into partitions known as shards (horizontal partitioning); store shards in different nodes; the partitioning field –known as the shard key must be:
  - Must exist in every document in the collection
  - It must have an index

The documents are partitioned based on the chunks of shard values

**SKIP rest of the chapter**