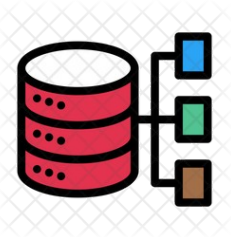
# NoSQL Database Concepts

Tushar B. Kute,
http://tusharkute.com



# NoSQL

- A NoSQL (originally referring to "non-SQL" or "non-relational") database provides a mechanism for storage and retrieval of data that is modeled in means other than the tabular relations used in relational databases.
- Such databases have existed since the late 1960s, but the name "NoSQL" was only coined in the early 21st century, triggered by the needs of Web 2.0 companies.
- NoSQL databases are increasingly used in big data and real-time web applications.
- NoSQL systems are also sometimes called "Not only SQL" to emphasize that they may support SQL-like query languages or sit alongside SQL databases in polyglot-persistent architectures

# NoSQL

- Motivations for this approach include: simplicity of design, simpler "horizontal" scaling to clusters of machines (which is a problem for relational databases), finer control over availability and limiting the object-relational impedance mismatch.
- The data structures used by NoSQL databases (e.g. key–value pair, wide column, graph, or document) are different from those used by default in relational databases, making some operations faster in NoSQL.
- The particular suitability of a given NoSQL database depends on the problem it must solve. Sometimes the data structures used by NoSQL databases are also viewed as "more flexible" than relational database tables.

# NoSQL – Types

- Wide column: Azure Cosmos DB, Accumulo, Cassandra, Scylla, HBase.
- Document: Azure Cosmos DB, Apache CouchDB, ArangoDB, BaseX, Clusterpoint, Couchbase, eXist-db, IBM Domino, MarkLogic, MongoDB, OrientDB, Qizx, RethinkDB
- Key–value: Azure Cosmos DB, Aerospike, Apache Ignite, ArangoDB, Berkeley DB, Couchbase, Dynamo, FoundationDB, InfinityDB, MemcacheDB, MUMPS, Oracle NoSQL Database, OrientDB, Redis, Riak, SciDB, SDBM/Flat File dbm, ZooKeeper
- Graph: Azure Cosmos DB, AllegroGraph, ArangoDB, InfiniteGraph, Apache Giraph, MarkLogic, Neo4J, OrientDB, Virtuoso

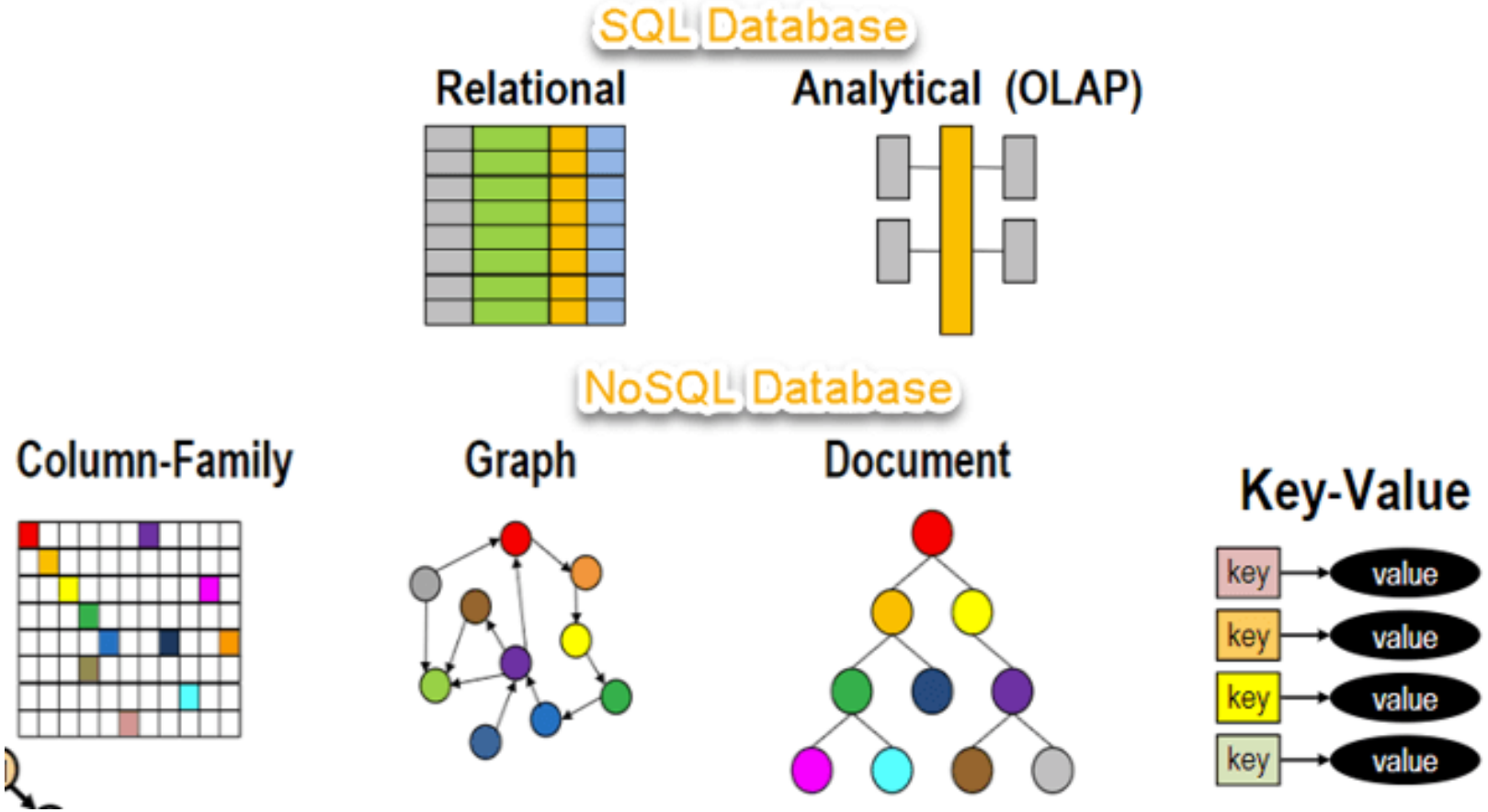
# Database as a Service – SQL

- Amazon Aurora, MySQL based service
- Amazon Relational Database Service
- Clustrix Database as a Service
- Crunchy Bridge, PostgreSQL as a Service.
- EnterpriseDB Postgres Plus Cloud Database
- Google Cloud SQL
- Heroku PostgreSQL as a Service (shared and dedicated database options)
- MariaDB SkySQL
- Microsoft Azure SQL Database (MS SQL)
- Oracle Database Cloud Service
- Snowflake Cloud Data Warehouse
- Xeround Cloud Database* – MySQL front-end (*service no longer available)

# Database as a Service – NoSQL

- Amazon DynamoDB
- Amazon SimpleDB
- Azure Cosmos DB
- Cloudant Data Layer (CouchDB)
- EnterpriseDB Postgres Plus Cloud Database
- Google Cloud Bigtable
- Google Cloud Datastore
- MongoDB Database as a Service (several options)
- RavenDB Cloud Database as a Service
- Oracle NoSQL Database Cloud Service
- Amazon DocumentDB

# Database Types



# Comparison

| SQL | NOSQL |
| --- | --- |
| Relational Database management system | Distributed Database management system |
| Vertically Scalable | Horizontally Scalable |
| Fixed or predifined Schema | Dynamic Schema |
| Not suitable for hierarchical data storage | Best suitable for hierarchical data storage |
| Can be used for complex queries | Not good for complex queries |

# Why NoSQL?

- NoSQL databases are used in nearly every industry.
- Use cases range from the highly critical (e.g., storing financial data and healthcare records) to the more fun and frivolous (e.g., storing IoT readings from a smart kitty litter box).

# Why NoSQL?

- Support large numbers of concurrent users (tens of thousands, perhaps millions)
- Deliver highly responsive experiences to a globally distributed base of users
- Be always available – no downtime
- Handle semi- and unstructured data
- Rapidly adapt to changing requirements with frequent updates and new features

# When NoSQL?

- When deciding which database to use, decision-makers typically find one or more of the following factors lead them to selecting a NoSQL database:
  - Fast-paced Agile development
  - Storage of structured and semi-structured data
  - Huge volumes of data
  - Requirements for scale-out architecture
  - Modern application paradigms like microservices and real-time streaming

# Misconcepts

- Over the years, many misconceptions about NoSQL databases have spread throughout the developer community. Two of the most common misconceptions:
  - Relationship data is best suited for relational databases.
  - NoSQL databases don't support ACID transactions.

# Misconcepts

- A common misconception is that NoSQL databases or non-relational databases don’t store relationship data well. NoSQL databases can store relationship data — they just store it differently than relational databases do.
- In fact, when compared with relational databases, many find modeling relationship data in NoSQL databases to be easier than in relational databases, because related data doesn’t have to be split between tables.
- NoSQL data models allow related data to be nested within a single data structure.

# Misconcepts

- Another common misconception is that NoSQL databases don't support ACID transactions. Some NoSQL databases like MongoDB do, in fact, support ACID transactions.
- Note that the way data is modeled in NoSQL databases can eliminate the need for multi-record transactions in many use cases. Consider the earlier example where we stored information about a user and their hobbies in both a relational database and a document database.
- In order to ensure information about a user and their hobbies was updated together in a relational database, we'd need to use a transaction to update records in two tables.
- In order to do the same in a document database, we could update a single document — no multi-record transaction required.

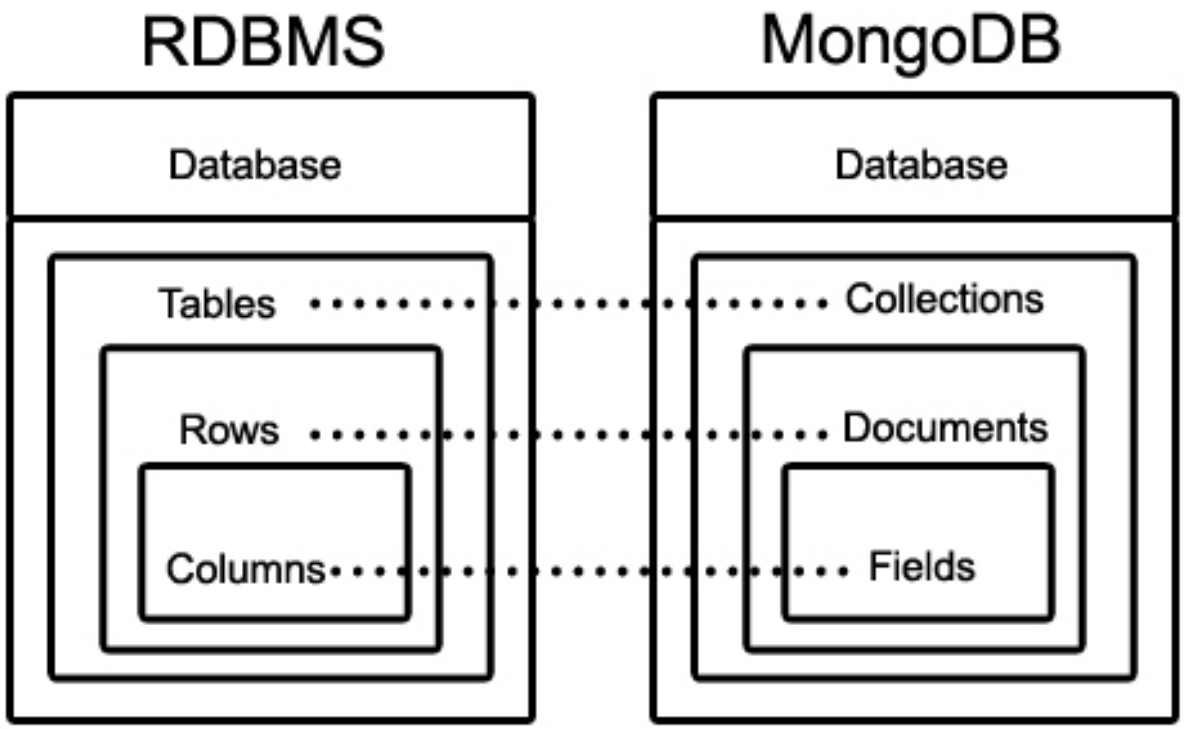
# Terminologies

- Database
  - Database is a physical container for collections. Each database gets its own set of files on the file system. A single MongoDB server typically has multiple databases.
- Collection
  - Collection is a group of MongoDB documents. It is the equivalent of an RDBMS table. A collection exists within a single database.
  - Collections do not enforce a schema. Documents within a collection can have different fields. Typically, all documents in a collection are of similar or related purpose.
- Document
  - A document is a set of key-value pairs. Documents have dynamic schema. Dynamic schema means that documents in the same collection do not need to have the same set of fields or structure, and common fields in a collection's documents may hold different types of data.

# RDBMS vs. MongoDB

| RDBMS | MongoDB |
|---|---|
| Database | Database |
| Table | Collection |
| Tuple/Row | Document |
| column | Field |
| Table Join | Embedded Documents |
| Primary Key | Primary Key (Default key _id provided by MongoDB itself) |
| **Database Server and Client** | |
| mysqld/Oracle | mongod |
| mysql/sqlplus | mongo |

# RDBMS vs. MongoDB



# Example Document

```
{
   _id: ObjectId(7df78ad8902c)
   title: 'MongoDB Overview',
   description: 'MongoDB is no sql database',
   by: 'MiTU Skillologies',
   url: 'https://www.mitu.co.in',
   tags: ['mongodb', 'database', 'NoSQL'],
   comments: [
      {
         user:'user1',
         message: 'My first comment',
         like: 0
      },
      {
         user:'user2',
         message: 'My second comments',
         dateCreated: new Date(2011,1,25,7,45),
         like: 5
      }
   ]
}
```

# Advantages

- Schema less – MongoDB is a document database in which one collection holds different documents. Number of fields, content and size of the document can differ from one document to another.
- Structure of a single object is clear.
- No complex joins.
- Deep query-ability. MongoDB supports dynamic queries on documents using a document-based query language that's nearly as powerful as SQL.
- Ease of scale-out – MongoDB is easy to scale.
- Conversion/mapping of application objects to database objects not needed.
- Uses internal memory for storing the (windowed) working set, enabling faster access of data.

# Why to use mongodb?

- Document Oriented Storage – Data is stored in the form of JSON style documents.
- Index on any attribute
- Replication and high availability
- Auto-Sharding
- Rich queries
- Fast in-place updates
- Professional support by MongoDB

# Where to use ?

- Big Data
- Content Management and Delivery
- Mobile and Social Infrastructure
- User Data Management
- Data Hub

# Redis

- Redis – Remote Dictionary Server – is a key-value store. It supports different kinds of abstract data structures such as strings, lists, maps, sets, sorted sets, and more. It’s also open-source.
- Pros: It supports a large variety of data types and is easy to install.
- Cons: Like MongoDB, it doesn’t support joins. It also requires knowledge of Lua, a high-level programming language.

# Thank you

*This presentation is created using LibreOffice Impress 5.1.6.2, can be used freely as per GNU General Public License*

@mitu_skillologies

/mITuSkillologies

@mitu_group

/company/mitu-skillologies

MITUSkillologies

**Web Resources**
https://mitu.co.in
http://tusharkute.com

**contact@mitu.co.in**

**tushar@tusharkute.com**