

# MongoDB Atlas

COSC 516 – Cloud Databases



**MongoDB** is a document data store supporting server and cloud deployments with MongoDB Atlas.

Documents are encoded in JSON and can be queried through the query API.

MongoDB is open source with a community edition.

- Flexible document data model, ad-hoc queries, secondary indexing, real-time aggregations

Subscription enterprise edition includes additional capabilities.

- In-memory storage engine for high throughput and low latency, security (LDAP, Kerberos), encryption for data at rest

# Use Cases

---

## 1) Flexible document store

- Spend time building systems rather than data modeling

## 2) Transactional

- Supports single document and multi-document transactions

## 3) Full-text search

- Store text in a document and index for fast search

## 4) Time series

- Dedicated structure optimized for time series data

# MongoDB Atlas

---

**MongoDB Atlas** is a cloud hosted fully-managed service for MongoDB.

- Pay-as-you-go model and billed on an hourly basis

Functionality includes auto-scaling, serverless instances, full-text search, and data distribution across regions and clouds.

May deploy on three major cloud providers: AWS, Google Cloud, Azure

Free cloud instance with storage up to 512 MB.

# JavaScript Object Notation (JSON)

---

*JavaScript Object Notation (JSON)* is a method for serializing data objects into text form.

## Benefits:

- Human-readable
- Supports semi-structured data
- Supported by many languages (not just JavaScript)

Often used for data interchange especially with AJAX/REST from web server to client.

# JSON Example

JSON constructs:

- **Values:** number, strings (double quoted), true, false, null
- **Objects:** enclosed in { } and consist of set of key-value pairs
- **Arrays:** enclosed in [ ] and are lists of values
- Objects and arrays can be nested.

```
{
  "Employees": [
    {
      "eno": "E1",
      "ename": "J. Doe",
      "title": "EE",
      "salary": 30000,
      "WorksOn": ["P1"]
    },
    {
      "eno": "E2",
      "ename": "M. Smith",
      "title": "SA",
      "salary": 50000,
      "WorksOn": ["P1", "P2"]
    },
    {
      "eno": "E3",
      "ename": "A. Lee",
      "title": "ME",
      "salary": 40000,
      "WorksOn": ["P3"]
    }
  ],
  "Projects": [
    {
      "pno": "P1",
      "pname": "Instruments",
      "budget": 150000
    },
    {
      "pno": "P2",
      "pname": "DB Develop",
      "budget": 135000
    },
    {
      "pno": "P3",
      "pname": "Budget",
      "budget": 250000
    }
  ]
}
```

# JSON versus Relations

	JSON	Relational
Structure	Nested objects + arrays	Tables
Schema	Variable (and not required)	Fixed
Queries	Limited	SQL, RA
Ordering	Arrays are sorted	No
Systems	Used with programming languages and NoSQL systems	Many commercial and open source systems
Case-sensitive?	Yes	No (mostly)



# BSON

---

MongoDB stores JSON documents in a binary representation called BSON (Binary JSON).

BSON encoding extends the popular JSON representation to include additional data types such as int, decimal, long, and floating point.

BSON data has an ordering unlike JSON.



# MongoDB Indexing

---

MongoDB uses B-Tree primary indexes and supports secondary indexes.

By default, creates an index on the document's **\_id** primary key field.  
User-defined indexes are secondary indexes.

Indexes can be created on any part of the JSON document including sub-documents and arrays.

# Query API

MongoDB query API is implemented as methods rather than text-based SQL. MongoDB has drivers for the most popular languages.

SQL	Mongo API
<pre>SELECT n_nationkey, n_name, n_regionkey FROM nation WHERE n_nationkey &lt; 20</pre>	<pre>db.nation.find( {"n_nationkey": {"\$lt": 20}}, {"n_nationkey": 1, "n_name": 1, "n_regionkey": 1, "_id": 0})</pre>

# Aggregation Pipeline

**Aggregation Pipeline** provides similar functionality to GROUP BY, aggregation, and JOIN in SQL. Documents are processed in stages.

- In each stage expressions produce output documents based on calculations performed on input documents.
- Can perform projection, filter, lookups (joins), and convert data types.
- Grouping with **\$group** stage supports aggregation functions (sum, avg, etc.).

SQL	Mongo API
<pre>SELECT n_regionkey, COUNT(n_nationkey) as numCountry, SUM(n_nationkey) as sumIds FROM nation WHERE n_nationkey &lt; 20 GROUP BY n_regionkey</pre>	<pre>db.nation.aggregate([{"\$match": {"n_nationkey": {"\$lt": 20}}}, {"\$group": {"_id": {"n_regionkey": "\$n_regionkey"}, "numCountry": {"\$sum": {"\$cond": [{"\$gt": ["\$n_nationkey", null]}, 1, 0]}}}, {"sumIds": {"\$sum": "\$n_nationkey"}}])</pre>

# Transactions

---

Default is **single document transactions**.

- May make as many changes within a document in a single transaction including arrays and sub-documents.

Added multi-document transaction support in MongoDB 4.0 and distributed transactions across sharded clusters in 4.2.

Uses snapshot isolation for transaction consistency.

# Consistency

---

Replication: Read and write operations go to the primary replica by default for strong consistency.

- May read from secondary replicas

Application may select consistency model at query level:

- Strict serializable consistency
- Read data committed to majority of nodes
- Read from single replica

Ability to adapt consistency model may help balance performance in applications.

# Durability

---

**Write Durability:** MongoDB uses write concerns to control the level of write guarantees for data durability.

- Write without confirmation
- Write with acknowledgement (from one, or some, or all secondaries)

Application specific control of write durability.

# Management

---

Compass GUI visualizes explain output to help identify query issues.

MongoDB Query Profiler displays slow-running queries and shows performance statistics.

MongoDB Atlas has Performance Advisor for query monitoring and suggesting indexes to improve performance.

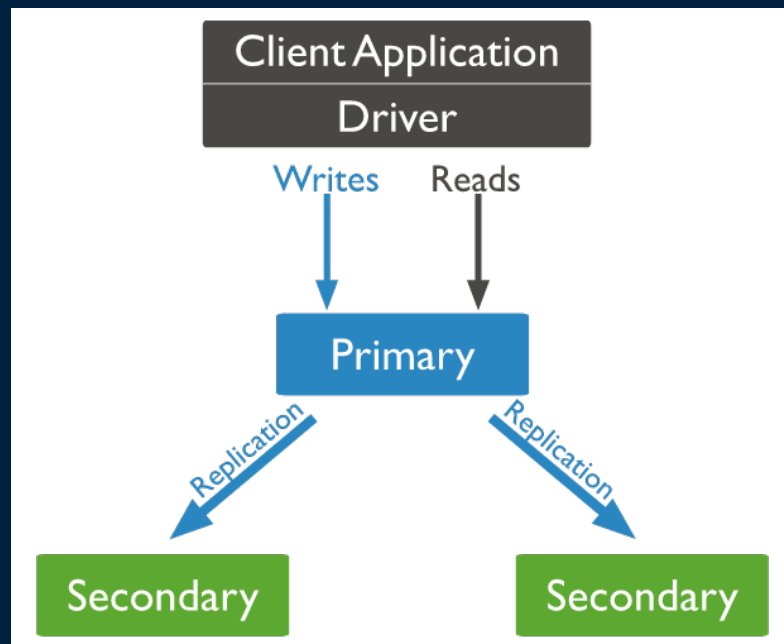


# Replica Sets

**Replica set** is a group of mongod processes providing redundancy and high availability.

**Primary:** receives all write operations.

**Secondaries:** Replicate operations from the primary to maintain an identical data set. May read from secondaries.



# Sharding

**Sharding** distributes data across multiple machines. Allows for larger data sets and higher throughput.

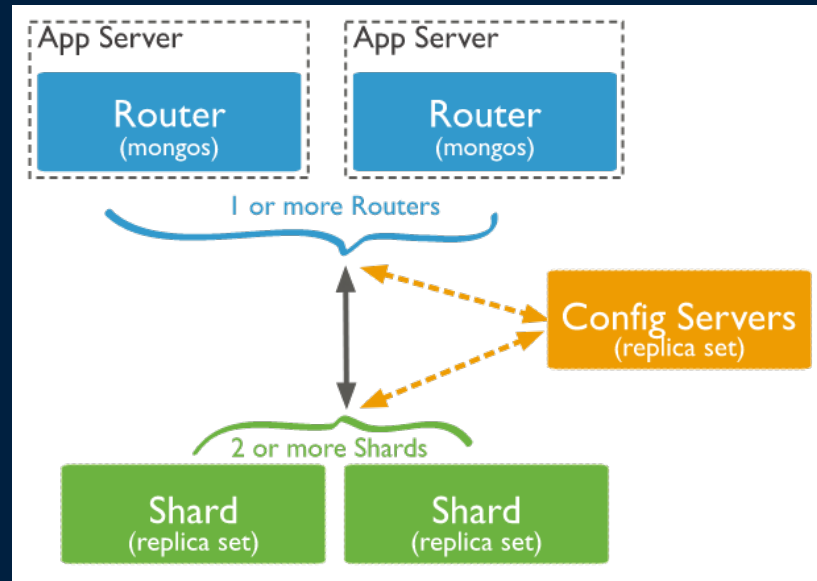
- Capacity can be expanded by adding additional servers.

A **sharded cluster** consists of:

- shard**: subset of data. May be deployed as a replica set.
- mongos**: query routers interfacing between client and cluster
- config servers**: store metadata and configuration settings

A database can have both sharded and unsharded collections.

- Sharded collections are partitioned and distributed across the shards in the cluster.
- Unsharded collections are stored on a primary shard. Each database has its own primary shard.



Source: MongoDB

<https://www.mongodb.com/docs/manual/sharding/>

# Security

---

User authentication is performed at the database level.

Enterprise version supports LDAP and Kerberos.

# Storage

---

WiredTiger is the default storage engine.

- Document-level concurrency, checkpointing, and compression
- Supports encryption at rest in Enterprise version.

In-Memory Storage Engine in Enterprise version retains document in memory for higher performance.

# Data Modeling

---

Modeling data as BSON documents allow for flexibility compared to normalized relational model. Possible to avoid joins by nesting data within sub-documents or arrays:

- Arrays in document model used to represent one-to-many relationships
- Sub-documents used to represent one-to-one relationships

MongoDB markets that dynamic schema is a major advantage over relational databases.

- Collections can be created without defining structure.
- Individual documents can have different structure and may change over time.
- Allows for faster development and prototyping.
- "As MongoDB allows schemas to evolve dynamically, such schema update operations requires upgrading just the application, with typically no action required for MongoDB. Evolving applications is simple, and project teams can improve agility and time to market."
- Downsides to dynamic schemas?

# Relational vs Document Terminology

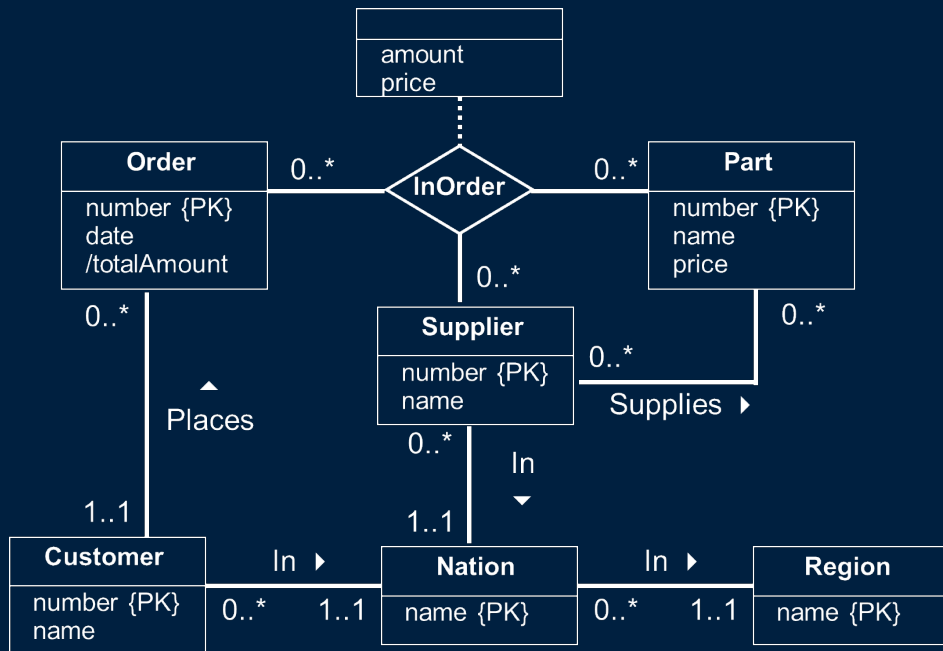
Relational	Document
Database	Database
Table	Collection
Row	Document
Column	Field
Index	Index
Join	Embedded document, document references
ACID Transactions	Multi-document transactions

# Design Problem: TPC-H Standard Schema

## Document Database vs Relational Database



- Each order has a numeric key, a customer, a date, a total order amount, and a list of parts.
- A part in an order has an amount and a price paid and is supplied by a certain supplier.
- Each part has a numeric key, a name, and a price and may be supplied by multiple suppliers.
- A supplier has a key and a name and may supply multiple parts.
- A customer has a key and a name.
- Each supplier and customer is located in a nation.
- Each nation is located in a region (continent).





# Pricing




---

Serverless: \$.10/million reads (up to 1 TB storage)

Shared: Free up to 512 MB storage

Dedicated: Cost depends on host

# Dedicated Cluster Pricing

<div>  Amazon Web Services          Microsoft Azure          Google Cloud Platform       </div>				
Cluster Tier	Storage	RAM	vCPUs	Base Price
M10	10 GB	2 GB	2 vCPUs	\$0.08/hr
M20	20 GB	4 GB	2 vCPUs	\$0.20/hr
M30	40 GB	8 GB	2 vCPUs	\$0.54/hr
M40	80 GB	16 GB	4 vCPUs	\$1.04/hr
M50	160 GB	32 GB	8 vCPUs	\$2.00/hr
M60	320 GB	64 GB	16 vCPUs	\$3.95/hr
M80	750 GB	128 GB	32 vCPUs	\$7.30/hr
M140	1000 GB	192 GB	48 vCPUs	\$10.99/hr
M200	1500 GB	256 GB	64 vCPUs	\$14.59/hr
M300	2000 GB	384 GB	96 vCPUs	\$21.85/hr
M400	3000 GB	488 GB	64 vCPUs	\$22.40/hr
M700	4000 GB	768 GB	96 vCPUs	\$33.26/hr

# Conclusion

---

**MongoDB** is a document database deployable on servers and cloud hosting using MongoDB Atlas.

MongoDB supports high availability using replica sets and high throughput using sharding (sharded clusters).

Representing data as documents is different than relational models as joins/foreign keys are often replaced by nesting using sub-documents or multi-valued arrays.

# Objectives

---

- Explain use cases for a document database like MongoDB
- Understand the basic constructs used to encode JSON data
- Compare JSON representation versus relational model
- Compare and contrast SQL versus MongoDB query API
- Explain the purpose of the aggregation pipeline
- Explain how replica sets support high availability
- Describe sharding and how sharded clusters function.
- Understand how modeling data in a document model is different than a relational model. Describe related to normalization.
- Describe the MongoDB Atlas deployment models and pricing.



THE UNIVERSITY OF BRITISH COLUMBIA

