Ministry of Higher Education and Scientific Research
University of Abdelhafid Boussouf - Mila
Institute of Mathematics and Computer Science
Department of Computer Science
Master 2 I2A – Big Data
2025/2026

Directed Works TD 9 – Advanced MongoDB Architecture & Integration

Objective: Understand High Availability (Replica Sets), master Horizontal Scaling strategies (Sharding), and design efficient Spark-MongoDB integration workflows.

Exercise 1: High Availability & Replica Sets

We have a MongoDB Replica Set consisting of **3 nodes**:

- Node A (Primary)
- Node B (Secondary)
- Node C (Secondary)

Scenarios:

1. **Failover:** Node A crashes due to a power outage.
   - Describe the exact steps the cluster takes to recover.
   - Can the application continue to **Write** data immediately?
   - Can the application continue to **Read** data immediately?
2. **Split Brain:** A network partition occurs. Node A is isolated (cannot talk to B or C), but B and C can talk to each other.
   - What happens to Node A?
   - What happens to the cluster formed by B and C?
3. **Write Concern:** An application issues a write with `w: "majority"`.
   - How many nodes must acknowledge the write before the application receives a "Success" response?
   - How does this affect **Latency** compared to `w: 1`?

Exercise 2: Sharding Design (The "TechStore" Scaling)

The `orders` collection in our TechStore has grown to **5 Terabytes**. A single server can no longer hold it. We need to **Shard** the collection across 4 shards.

Access Patterns:

- **80% of Queries:** "Find all orders for User X."
- **20% of Queries:** "Find recent orders (last 24 hours)."
- **Writes:** New orders are inserted constantly.

Task: Evaluate the following **Shard Keys**. For each, determine if it is **Good** or **Bad** based on *Cardinality*, *Write Distribution*, and *Query Isolation*.

Candidate Key	Type	Pros	Cons	Verdict
`_id`	Hashed	...	...	?
`order_date`	Ranged	...	...	?
`user_id`	Hashed	...	...	?

Hint: A "Monotonically Increasing" key (like Date) causes the "Hot Shard" problem.

Exercise 3: Consistency vs. Availability (CAP Theorem)

Your client requires specific behaviors for their dashboard. Choose the correct **Read Preference** and **Write Concern** for each requirement.

Options:

- *Read Preference:* `primary`, `primaryPreferred`, `secondary`, `nearest`.
- *Write Concern:* `w: 1`, `w: majority`, `w: 0`.

Requirements:

1. **The Analytics Dashboard:** "I don't care if the data is 1 second old, but the query must be fast and not slow down the checkout process for other users."
2. **The Billing System:** "Data integrity is critical. I must be 100% sure the payment record is saved to disk on multiple servers before I confirm the transaction."
3. **The User Profile:** "When I update my profile picture, I want to see it change immediately. I should never see the old picture after I hit save."

Exercise 4: Spark-MongoDB Integration

We are connecting Apache Spark to MongoDB to train a Machine Learning model.

Query:

```
df = spark.read.format("mongodb").load()
df_filtered = df.filter(df["age"] > 25)
df_filtered.count()
```

1. **Predicate Pushdown:** Does Spark load *all* data into memory and then filter it, or does it ask MongoDB to filter it first? Why does this matter for performance?
2. **Partitioning:** If the MongoDB collection is split into **100 Chunks** (splits) across 4 Shards. How many **Spark Partitions** (RDD partitions) will be created by default?
3. **Locality:** If we run Spark on the *same nodes* as MongoDB, how does the connector optimize data transfer?