DESCRIBE THE WORKLOADS IN MONGODB DATA MODELING

Lecturer: Trần Thế Trung

Content

- Identify the workload
- Case study: Internet of Things
- Quantify and Qualify Operations
- Data Replication

Identify The Workload

- Identify the size of your data, the important reads and writes of the system, and possibly quantify and qualify those operations.

- Example: A workload might consist of a subset of resources in a single AWS account or be a collection of multiple resources spanning multiple AWS accounts.

- A small business might have only a few workloads while a large enterprise might have thousands.

- In addition to designing the **test scenarios** and **deciding what data** to use, you must determine the number of users for running the scenarios and how fast the scenarios are executed. Service level agreements help with planning.

- **The workload** drives the performance test, and the benchmarking goals drive the workload setup.

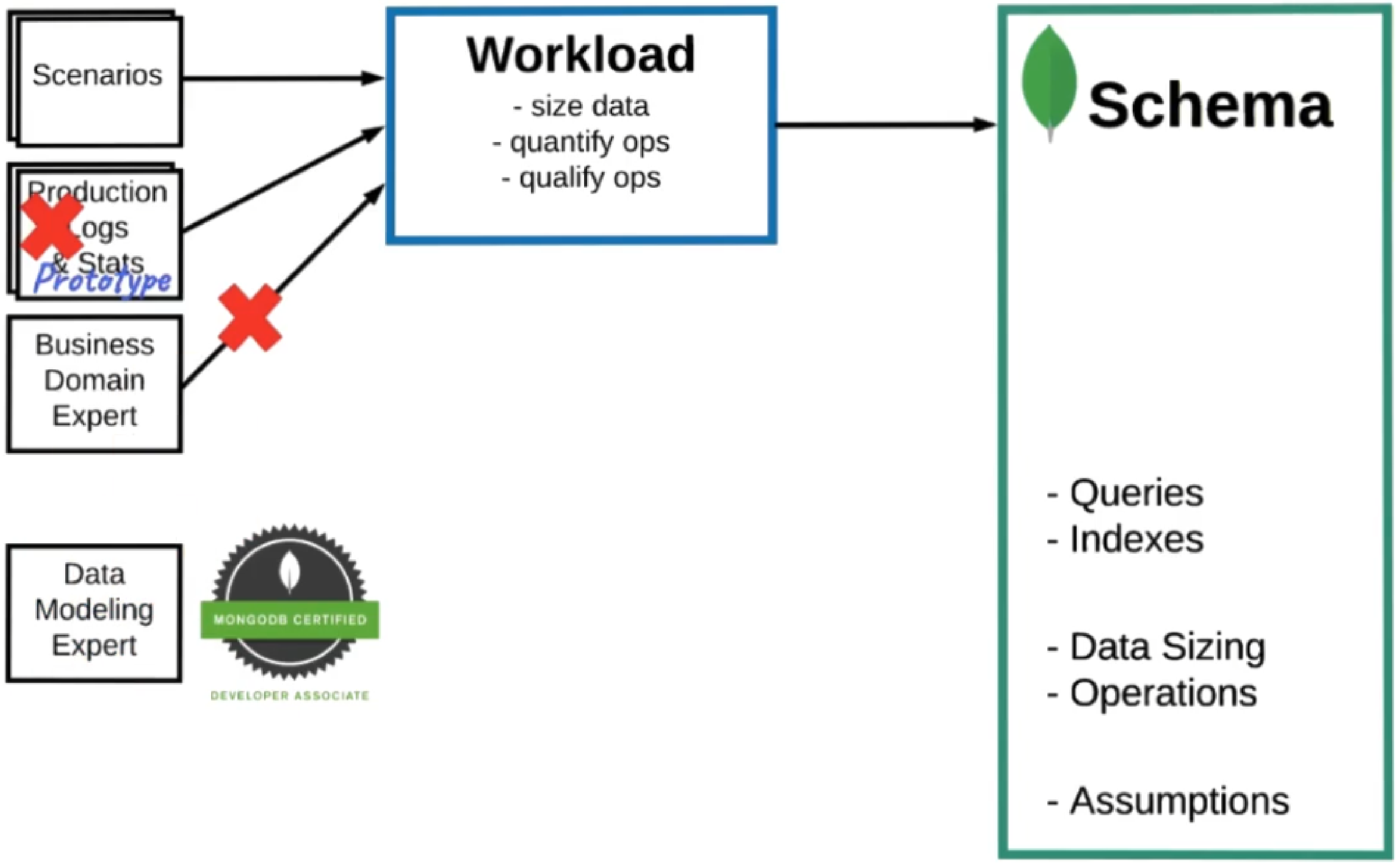
Case study: Internet of Things

- **Organization has 100 million weather sensors**.

- **Need to**:
  - Collect the data form all devices
  - Analyze the data trends with 1 team of 10 data scientists

- **We start with the first phase:** identifying, quantifying, and qualifying the workload.

Case study: Internet of Things



Case study: Internet of Things

Here's the information we have to begin with

- We don't have **production logs and stats**.
- However, we do have a **prototype**.
- We can't rely on the **business domain expert**.
- However, there is only a **handful of things to understand**.
- We have a MongoDB certified developer in our team.
- Our previous team has written a prototype that captures the data we see under the collection data in the database, **sample_weatherdata**.

Sample_weatherdata

```
_id : ObjectId('5553a998e4b02cf7151190b8')
st: "x+47600-047900"
ts: 1984-03-05T13:00:00.000+00:00
position: Object
    type: "Point"
    coordinates: Array (2)
        0: -47.9
        1: 47.6
elevation: 9999
callLetters: "VCSZ"
qualityControlProcess: "V020"
dataSource: "4"
type: "FM-13"
airTemperature: Object
    value: -3.1
    quality: "1"
dewPoint: Object
    value: 999.9
    quality: "9"
pressure: Object
    value: 1015.3
    quality: "1"
```

xác định thiết bị

ngày giờ

vị trí của thiết bị

tọa độ

tên gọi

quy trình kiểm soát chất lượng

nhiệt độ không khí

điểm sương

áp suất

Sample_weatherdata

```
wind: Object
    direction: Object
        angle: 999
        quality: "9"
    type: "9"
    speed: Object
        rate: 999.9
        quality: "9"
visibility: Object
    distance: Object
        value: 999999
        quality: "9"
    variability: Object
        value: "N"
        quality: "9"
skyCondition: Object
    ceilingHeight: Object
        value: 99999
        quality: "9"
        determination: "9"
    cavok: "N"
sections: Array (1)
    0: "AG1"
```

Case study: Internet of Things

Identifying Workloads

- A **unique string identifies** each device using the device position in the field.
- Additional fields store metrics like air temperature, dew point, pressure, wind, for a given measure. A numeric value and the relative quality on the scale of 1 to 9 describe these metrics.
- The prototype was useful to see which queries our meteorologists wanted to run.
- Now it is our job to scale up the prototype to collecting data from a few million devices in the field

Case study: Internet of Things

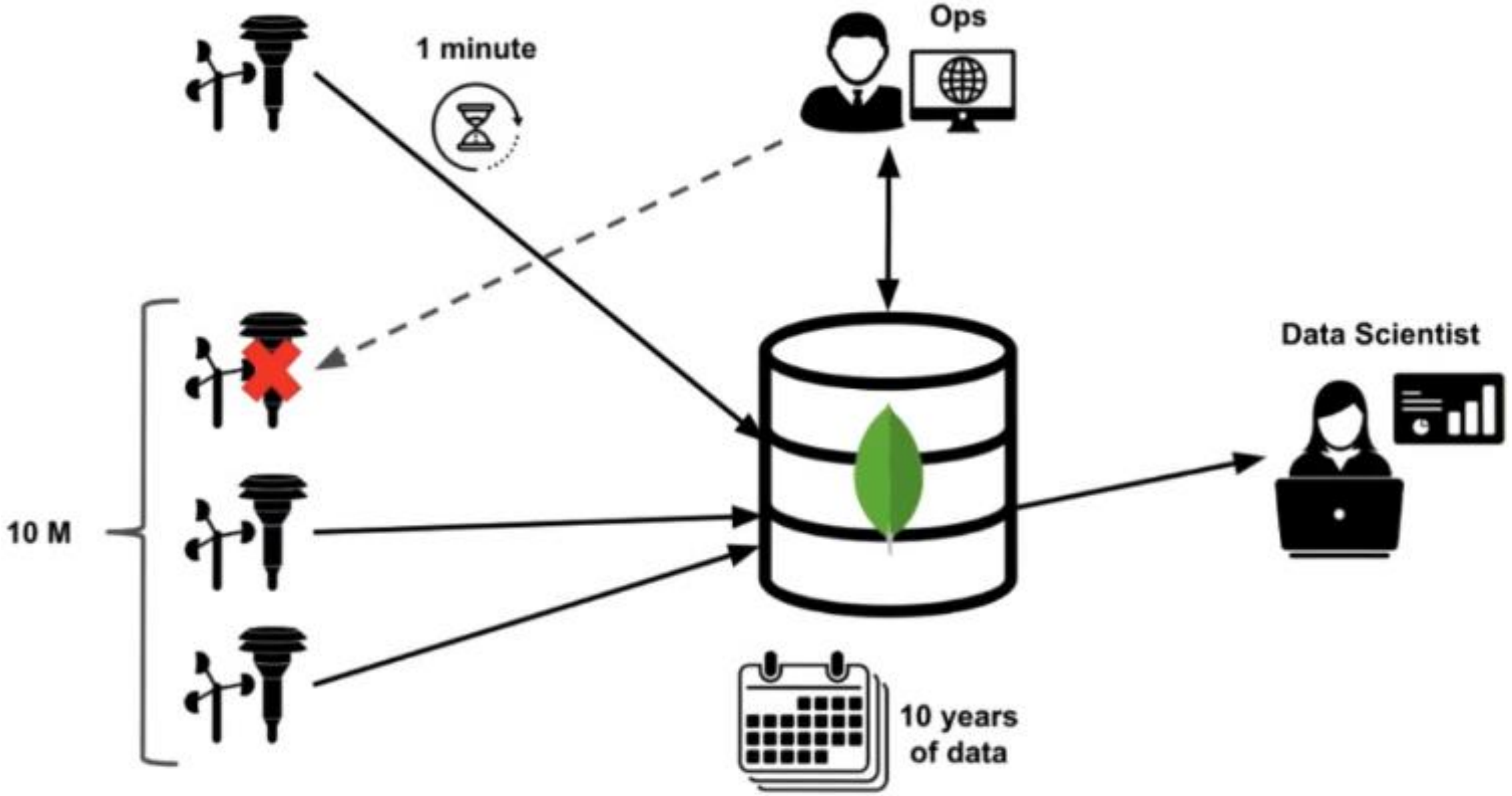
Identifying Workloads

So far, we know the following:

- We also need to run **some script to aggregate data**, to make the data simpler to analyze.
- Our goal for data **scientists** is to **analyze the data to find trends**.
- Maybe some queries are fixed, while the other ones are exploratory ones, meaning they are difficult to predict in advance.

Case study: Internet of Things

Identifying Workloads



Case study: Internet of Things

Scenarios

1. 100 million device send data
2. Devices can collect and send data every minute
3. Most trend (not all) can be identified with hourly data
4. Keep the data for 10 years
5. Ops team need to identify faulty/non-responsive devices
6. Ops team need to assemble/merge data for Data Analyst
7. Data Analysts run analytic queries

Quantify and Qualify Operations

- A list of the CRUD operations, specifically the read and write operations, the system needs to handle.

- Each actor produces one or more operations, which translates to **entities and field**, and we need to i**dentify each operation** as a **read** or a **write** operation.

- In a more complex model, you could have more queries, and you would expand on the type and the fields that are used by the operations.

Quantify and Qualify Operations

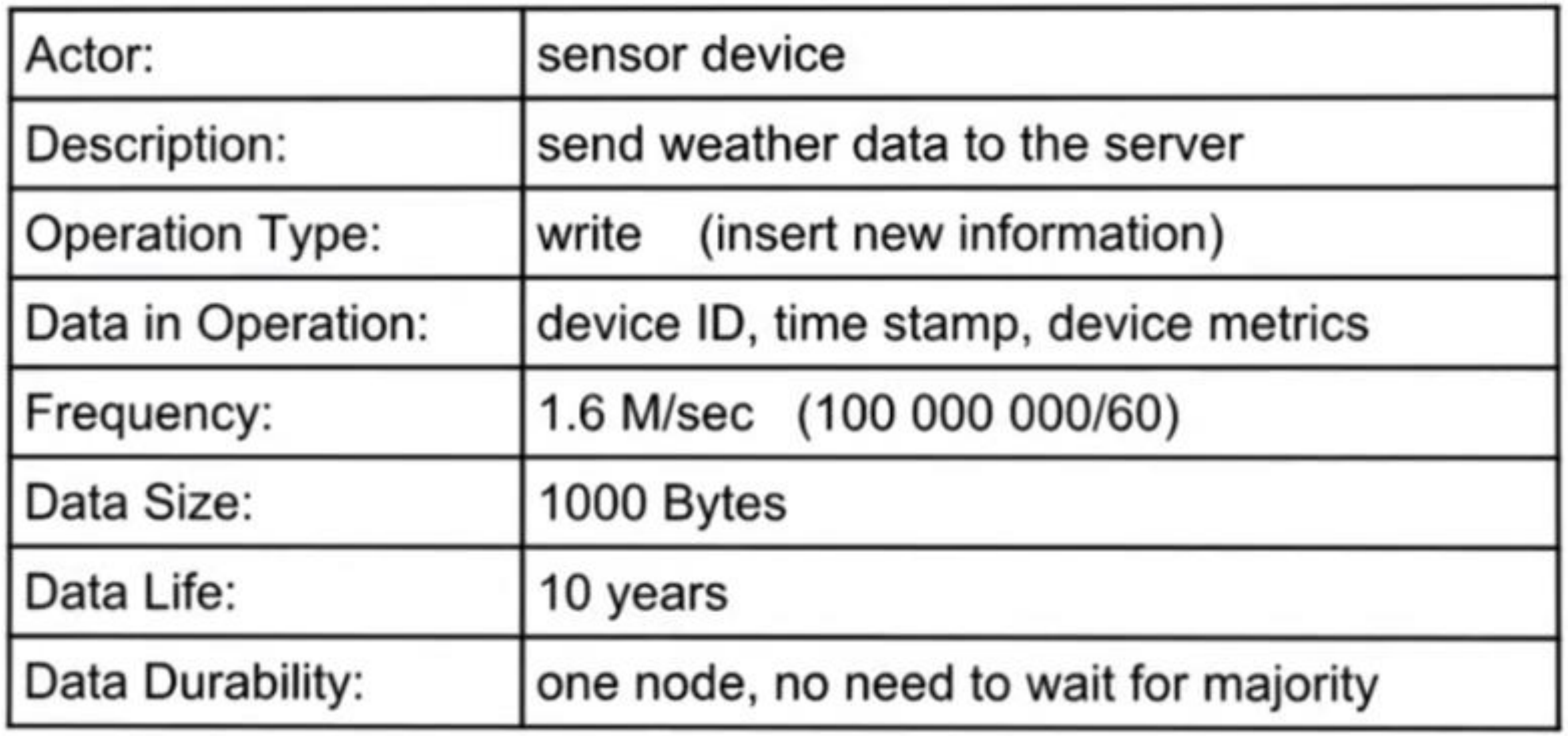
Actor	CRUD Operations	Info Needed	Operation Type
device	send metric data every minute	device_id metrics	write
Ops	identify non-operational devices	device_id Timestamp	read
Ops	aggregate data per hour	device_id metrics	write
Data scientist	run ~10 analytic queries per hour	metrics aggregated metrics	read

The Most Important Operation

- In the **context** of this calculation, we need to identify the **most critical operation**. Most of the time, there is only one query that stands out.

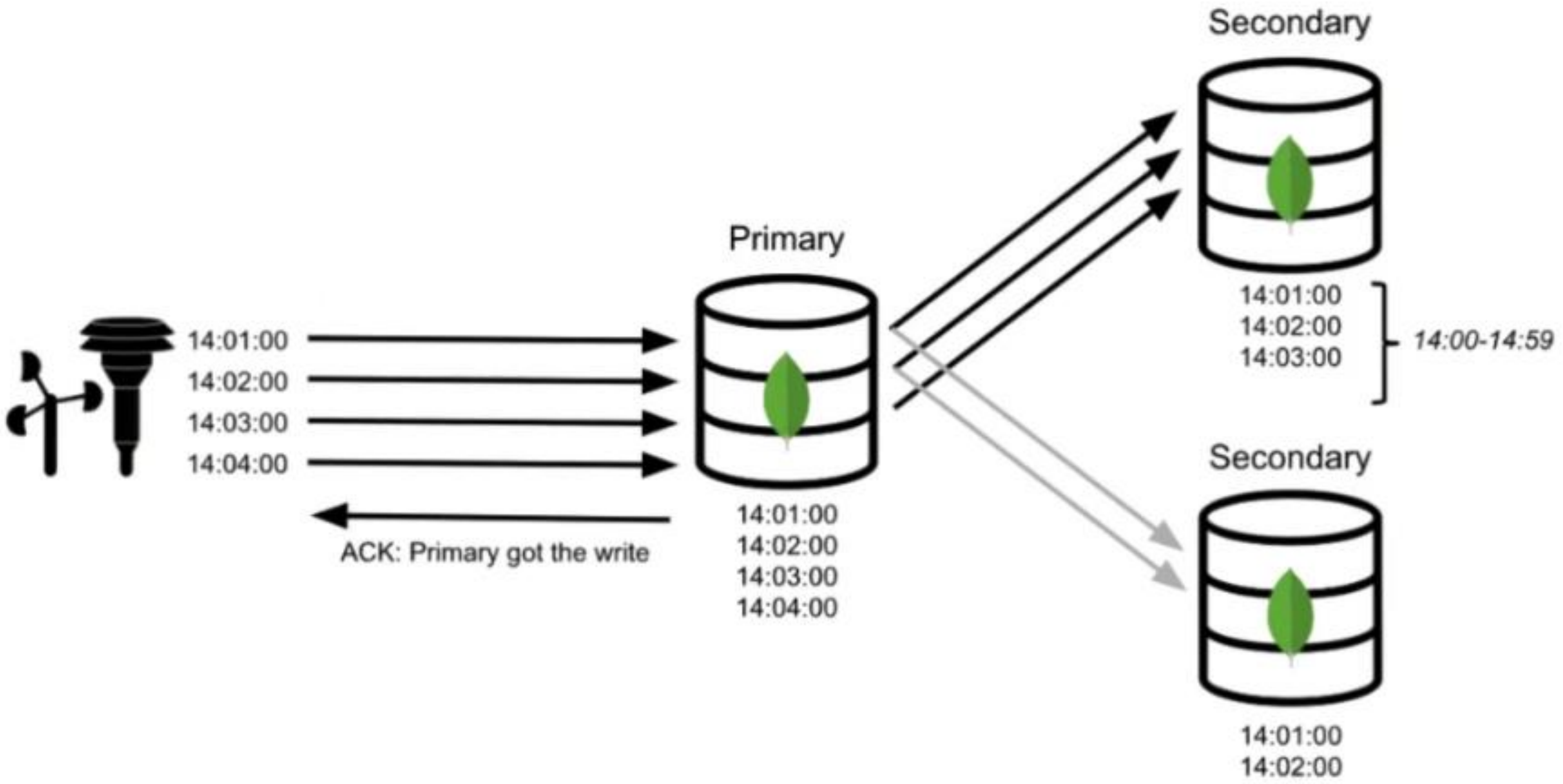
Actor	CRUD Operations	Info Needed	Operation Type
device	send metric data every minute	device_id metrics	write
Ops	identify non-operational devices	device_id Timestamp	read
Ops	aggregate data per hour	device_id metrics	write
Data scientist	run ~10 analytic queries per hour	metrics aggregated metrics	read

Details of a Write Operation



Actor:	sensor device
Description:	send weather data to the server
Operation Type:	write (insert new information)
Data in Operation:	device ID, time stamp, device metrics
Frequency:	1.6 M/sec (100 000 000/60)
Data Size:	1000 Bytes
Data Life:	10 years
Data Durability:	one node, no need to wait for majority

Data Replication



Data Replication

- Each sensor reading sent to the system gets replicated to **other hosts**.

- Each single minute reading is not a crucial piece of information, mainly because we aggregate the data over time.

- In the event of issues with the network or the host, because we did not wait for a majority of nodes writing the data, it could result in a piece of data missing on the newly elected primary.

Data with Their Desired Durability

- **Durability in databases** is the property that ensures transactions are saved permanently and do not accidentally disappear or get erased, even during a database crash.

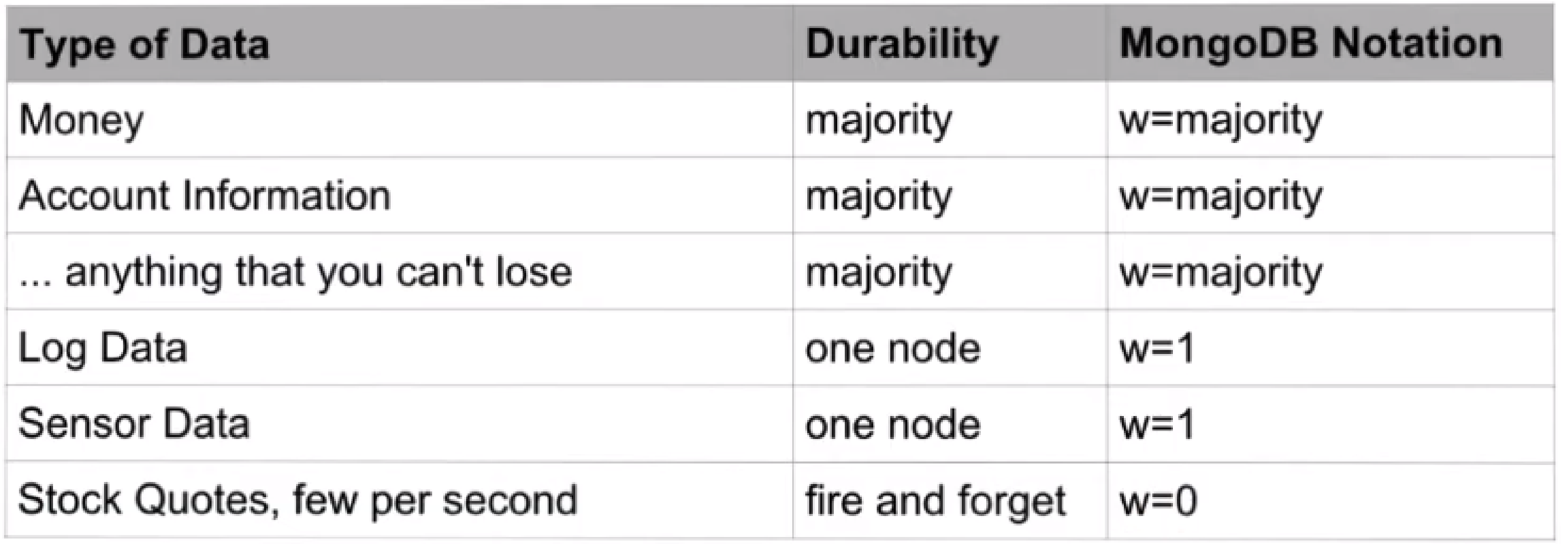
- **Durability is the guarantee** that an operation that is committed will survive permanently.

- MongoDB has highly configurable durability settings, from absolutely no guarantees to completely durable.

Data with Their Desired Durability

- **Typically for IoT projects**, entering a single write as on the primary node is sufficient.
  - Lowering data durability requirements is in contrast to writing a crucial piece of information, like an operation involving money, updates to user account, or anything that you don't want to lose.
  - For this type of data, you want to confirm that it is written to a majority of nodes before the cluster acknowledges your write operation back to the application.

- On the other hand, some data aggregated or refreshed frequently may not need the same care in terms of data durability



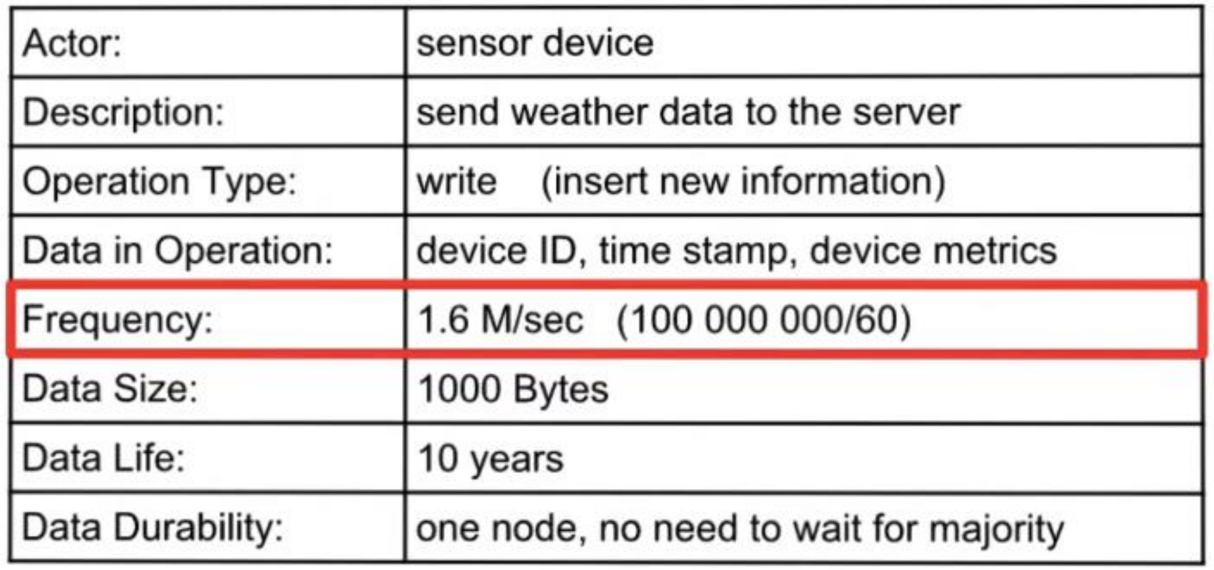
Type of Data	Durability	MongoDB Notation
Money	majority	w=majority
Account Information	majority	w=majority
... anything that you can't lose	majority	w=majority
Log Data	one node	w=1
Sensor Data	one node	w=1
Stock Quotes, few per second	fire and forget	w=0

Data with Their Desired Durability

Details of a Write Operation

Workload analysis

- Did we say 1.6 million operations per second?
- Hosts with many cores in can sustain around 100,000 writes per second.
- To satisfy this order of operation, you could estimate a cluster with a data-partition in 10 to 20 shards to handle this kind of workload.
- In this use case, the system presents a workload where writes correspond to 99% of all database operations.



Actor:	sensor device
Description:	send weather data to the server
Operation Type:	write (insert new information)
Data in Operation:	device ID, time stamp, device metrics
Frequency:	1.6 M/sec (100 000 000/60)
Data Size:	1000 Bytes
Data Life:	10 years
Data Durability:	one node, no need to wait for majority

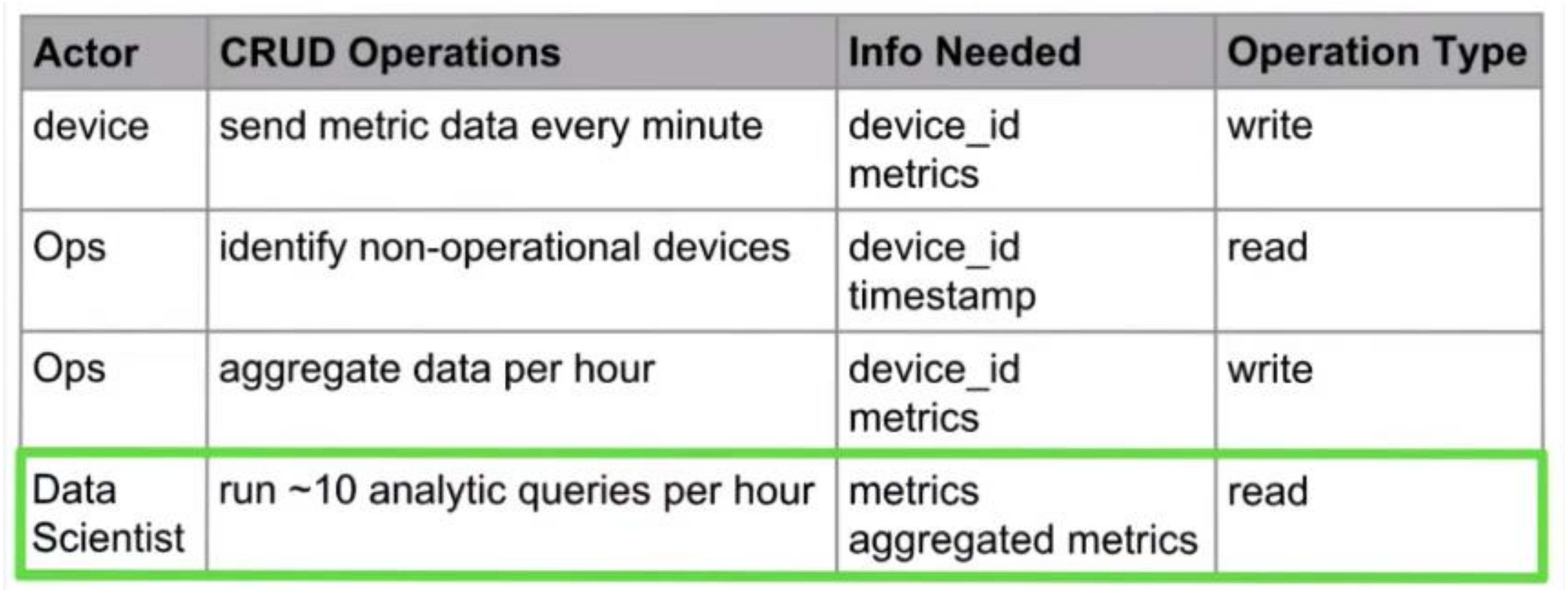
Details of a Write Operation

Workload analysis

- Understanding what data drop rate is acceptable is an important strategy that needs to be used by systems that collect data from many devices
- The system presents a workload where **writes correspond to 99%** of all database operations
  - Optimizing for writes has a lot to do with the hardware.
  - However, on the data modeling side, we want to look into patterns or techniques that allow for **reducing the number of individual writes** or grouping information in individual data structures.

Quantify and Qualify Operations

Workload analysis



Actor	CRUD Operations	Info Needed	Operation Type
device	send metric data every minute	device_id metrics	write
Ops	identify non-operational devices	device_id timestamp	read
Ops	aggregate data per hour	device_id metrics	write
Data Scientist	run ~10 analytic queries per hour	metrics aggregated metrics	read

Details of a Read Operation

Workload analysis

Reads are also essentials:

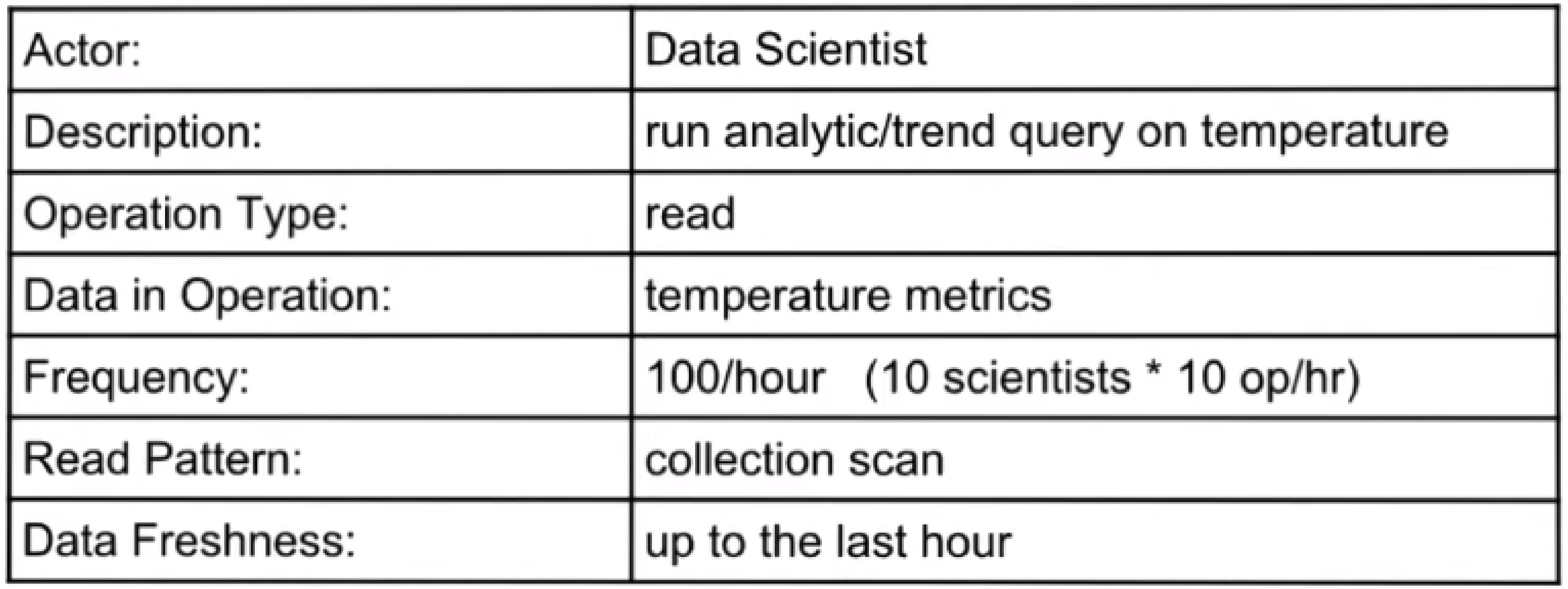
- The data scientists run this read operation on the metrics of the device
- Each data scientist runs about 10 queries per hour.
- So we do get 100 of these analytic queries run every hour.

As for read operations, even if we execute them less often than write operations, they can still require a significant amount of data. So we need to map out the **most common queries to shape the document accordingly.**

Details of a Read Operation

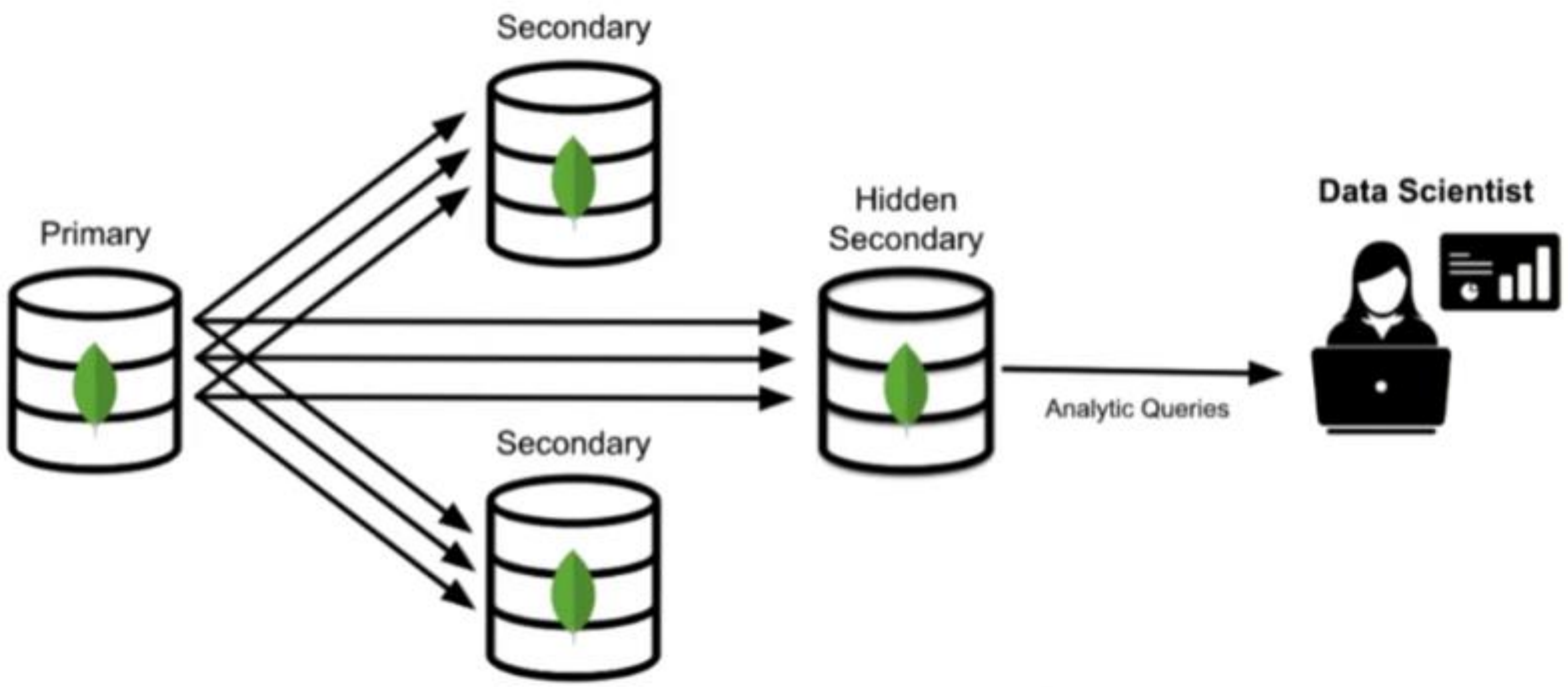
Workload analysis

- Collection scan on data that is one hour old, for these type of queries, using nodes dedicated to data analytics this is a viable solution.
- Our primary node is likely to be very busy processing writes.
- So we may want to **dedicate** a node to process the reads ***and isolate all the collection scans they may require from other nodes***.



Actor:	Data Scientist
Description:	run analytic/trend query on temperature
Operation Type:	read
Data in Operation:	temperature metrics
Frequency:	100/hour (10 scientists * 10 op/hr)
Read Pattern:	collection scan
Data Freshness:	up to the last hour

Dedicated Node for Analytics



Current Observations

- **Workload is mostly composed of writes**, so we need to optimize for writes, first by ensuring proper write distribution, but also by applying techniques and patterns that can reduce the number of write operations.

- **The read operation**, even if we execute them less often than the write operations, they may still require a considerable amount of data.

- So we need to map out the most common queries to shape the documents accordingly.

Current Observations

1. **Workload is ~ 99%/1% reads**
   a. Distribute evenly writes if sharded
   b. Look at reducing or grouping the writes
2. **Most reads are collection scans without required low/latency**
   a. Queries can be run on dedicated “analytics” node
   b. Need more information on which exact queries are run
   c. Pre-computed/aggregate the data to optimized those queries

Recap

- Quantify and Qualify the queries as much as you can
- Few CRUD operations with drive the design