Chapter 4: Patterns (Part 2)

Lab: Apply the Bucket Pattern

Problem:

In this lab, we will be applying the Bucket Pattern to address an IoT use case scenario.

Scenario:

You've been called in to help improve the performance profile of an application that provides a dashboard and reporting information of cell tower quality service metrics.

This application collects a set of metric information sent directly from cell towers.

Each cell tower emits a message similar to the following, in 1 minute intervals:

COPY

```
{
  "date": "2019-05-01T17:23:43.042Z",
  "celltower_id": "BBA87930-4A72-4D77-B238-2BA5899C9BEC",
  "coordinates": [
    -8.5837984085083,
    41.161823159286136
  ],
  "established_calls": 50013,
  "dropped_calls": 1231,
  "data_in_gb": 142,
  "data_out_gb": 481
}
```

The information sent in each message corresponds to:

- the current longitude and latitude of the cell tower - `coordinates`
- a unique identifier of the cell tower - `celltower_id`

- the date of the measurement - `date`
- the following accumulated counters, accumulated since last reboot:
    - the number of established cell phone calls - `established_calls`
    - the number of dropped cell phone calls - `dropped_calls`
    - the amount of inbound data traffic - `data_in_gb`
    - the amount of outbound data traffic - `data_out_gb`

This system needs to support the following operational requirements:

- Be capable of storing measurements for at least 500 cell towers
- Be able to produce cell tower cumulative reports on one or all four of the accumulated counters, with the following specifications:
    - Each of these reports consists of a plotted graph of the last 24 hours for each metric, with a granularity of 5 minutes
    - The 95 percentile request latency expected for this report is 100ms

Current Implementation:

The current implementation is to store documents that are very similar to the emitted messages, having all messages stored in a `measurements` collection:

COPY

```
db.measurements.find({"cell_tower.id": "BBA87930-4A72-4D77-
B238-2BA5899C9BEC"})
{
  "_id": ObjectId("5cd3587395f8fbc3fab3092e"),
  "date_received": ISODate("2019-05-01T17:24:00.000Z"),
  "message_date": ISODate("2019-05-01T17:23:43.042Z"),
  "cell_tower": {
    "location": {
      "type": "Point",
      "coordinates": [
          -8.5837984085083,
           41.161823159286136
      ]
    },
    "id": "BBA87930-4A72-4D77-B238-2BA5899C9BEC"
  },
  "established_calls": 50013,
  "dropped_calls": 1231,
  "data_in_gb": 142,
  "data_out_gb": 481
}
...
```

This implementation is expected to fill out the database servers disk space in one month and there is no budget left for hardware improvements in the next 12 months.

The current implementation is unable to produce the reports within the required **100ms**.

Your Solution:

In order to resolve the issues that the current approach is facing, you need to come up with a schema design alternative that allows for:

- an increase in the number of IoT devices and associated workload growth
- all report generation to comply with the expected SLA of less than 100ms
- allow for the application to report status for the last 24 hours, with a granularity of 5 minutes

Using your pattern knowledge, consider the following three choices for the implementation:

A:

One document per hour:

```
COPY
{
  "_id": ObjectId("5cd3587395f8fbc3fab3092e"),
  "date": ISODate("2019-05-01T17:00:00.000Z"),
  "celltower_id": "BBA87930-4A72-4D77-B238-2BA5899C9BEC",
  "established_calls": {
    "minutes": [
      98, 262, 266, 106, 254, 109, 3, 32, 257, 199,
      194, 209, 251, 269, 175, 42, 240, 169, 166, 149,
      238, 43, 128, 119, 120, 134, 267, 87, 228, 56,
      198, 9, 203, 281, 266, 91, 210, 55, 91, 118,
      203, 283, 74, 19, 222, 37, 18, 249, 149, 76,
      165, 29, 44, 94, 277, 253, 79, 100, 182, 127
    ],
    "sum": 9072
  },
  "dropped_calls": {
    "minutes": [
      0, 1, 0, 0, 1, 0, 0, 2, 0, 0,
      0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
      0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
      0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
      0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
      0, 0, 0, 0, 0, 0, 0, 0, 0, 0
    ],
    "sum": 6
  },
  "data_in_gb": {
    "minutes": [
      5, 12, 30, 0, 24, 12, 12, 5, 16, 5,
      4, 18, 6, 2, 13, 7, 11, 2, 8, 30, 25,
      7, 4, 27, 2, 30, 0, 17, 17, 5,
      9, 19, 10, 4, 13, 1, 4, 3, 3, 28,
      12, 8, 1, 21, 6, 4, 29, 23, 3, 16,
      0, 30, 20, 17, 2, 13, 15, 12, 16, 6
    ],
    "sum": 704
  },
  "data_out_gb": {
    "minutes":[
      43, 100, 59, 40, 7, 57, 61, 3, 94, 84, 37,
      67, 25, 80, 40, 34, 8, 20, 69, 66,
      94, 71, 85, 95, 54, 65, 35, 26, 33,
      42, 19, 42, 72, 45, 100, 17, 96, 53, 50, 91,
      34, 79, 45, 34, 51, 96, 90, 5, 12, 30, 50,
      4, 67, 21, 54, 17, 6, 91, 19, 36
    ],
    "sum": 3020
  }
}
```

B:

One document per day per metric:

```
{
  "_id": ObjectId("5cd3587395f8fbc3fab30934"),
  "date": ISODate("2019-05-01"),
  "metric:": "established_calls",
  "celltower_id": "BBA87930-4A72-4D77-B238-2BA5899C9BEC",
  "measurements": {
    "0": [98, 262, 266, 106, 254, 109, 3, 32, 257, 199, ... ],
    "1": [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 ...],
    "2": [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 ...],
    ...
    "23": [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 ...],
  },
  "sum": 9072
}
{
  "_id": ObjectId("5cd3587395f8fbc3fab30933"),
  "date": ISODate("2019-05-01"),
  "metric:": "dropped_calls",
  "celltower_id": "BBA87930-4A72-4D77-B238-2BA5899C9BEC",
  "measurements": {
    "0": [ 0, 1, 0, 0, 1, 0, 0, 2, 0, 0, 0, 1, 0, 0, 0, ... ],
    "1": [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 ...],
    "2": [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 ...],
    ...
    "23": [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 ...]

  },
  "sum": 6
}
{
  "_id": ObjectId("5cd3587395f8fbc3fab30932"),
  "date": ISODate("2019-05-01"),
  "metric:": "data_in_gb",
  "celltower_id": "BBA87930-4A72-4D77-B238-2BA5899C9BEC",
  "measurements": {
    "0": [ 5, 12, 30, 0, 24, 12, 12, 5, 16, 5, 4, 18, 6 ...],
    "1": [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 ...],
    "2": [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 ...],
    ...
    "23": [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 ...],
  },
  "sum": 704
}
{
  "_id": ObjectId("5cd3587395f8fbc3fab30931"),
  "date": ISODate("2019-05-01"),
  "metric:": "data_out_gb",
  "celltower_id": "BBA87930-4A72-4D77-B238-2BA5899C9BEC",
  "measurements": {
    "0":[ 43, 100, 59, 40, 7, 57, 61, 3, 94, 84, 37, 67 ... ],
    "1": [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 ...],
    "2": [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 ...],
    ...
    "23": [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 ...],
  },
  "sum": 3020
}
```

C:

One document per day:

```
{
  "_id": ObjectId("5cd3587395f8fbc3fab3092e"),
  "date": ISODate("2019-05-01"),
  "celltower_id": "BBA87930-4A72-4D77-B238-2BA5899C9BEC",
  "established_calls": {
    "0": 9072,
    "1": 0,
    "2": 0,
    ...
    "23": 0,
  },
  "dropped_calls": {
    "0": 6,
    "1": 0,
    "2": 0,
    ...
    "23": 0
  },
  "data_in_gb": {
    "0": 704,
    "1": 0,
    "2": 0,
    ...
    "23": 0,
  },
  "data_out_gb": {
    "0": 3020,
    "1": 0,
    "2": 0,
    ...
    "23": 0,
  }
}
```

Which of the above solutions provide a valid approach for solving the problems experienced by the application?

Attempts Remaining: **Correct Answer** ✓ ○ ○

Check all answers that apply:

- [✓ A]
- [✓ B]
- C

See detailed answer

Proceed to next section

Assignment is Due

05d:03hr:15m
07 sty, 17:00 UTC

Your Grade

PASS/FAIL
Submitted