DMV - MONGODB2

19 January 2021 17:43

PATTERNS

- Attribute: subset of fields that share characteristics

+: fewer indexes, simpler queries

ex:

```
releases: [
  {"location": <String>, "date": <Date>}, ...
]
```

- Bucket: instead of one document per measurement per ID bucket the data under the ID's

+: reduces number of documents, improve index performance and simplify data access

ex:

```
{
  sensor_ID: 1234
  measure: ...
  DATE: ...
}
...
```

→ SAME PATTERN & ID

```
{
  sensor_ID: <...>
  start: <date>
  end: <date>
  measurements: [{ measur:
                   date:
                 }, ...]
}
```

- computed: leave to the program the operation (like avg/sum)

+: reduced CPU workload for frequent computations
-: difficult to identify the need

- document versioning: added a field for "revision" (incrementing)

+: easy to implement with no performance impact on queries
-: double the writes and need to target the correct collection

- extended reference: copy frequently accessed fields from other referenced documents

+: improve performance on join operations with faster reads and less data fetching
-: data is duplicated so could be bad for rapidly changing attributes

- outlier: if few documents in a collection require a certain field → add a boolean field to check for extras (Y/N)
  ↳ add new collection with space for overflow info and <object ID> linking

+: prevents a few documents or queries from determining application solution, queries are the same only outliers change
-: ad hoc queries may not perform well, much of this pattern is application side

- pre-allocation

+: simplify design when the document structure is known in advance -: simplicity VS performance

- polymorphic: keep fields in common across all documents and [illegible] (add attributes) depending

- polymorphic: keep fields in common across all documents and change only specific attributes (add attributes) depending on the category they belong to
  +: easy to implement, with queries that run across multiple collections
  -: different code path required in the application based on the document
- schema versioning
  +: no downtime in schema updating , control the migration with less future debt
  -: might need two indexes
- SUBSET → Stored the collections in a more subcollections (product + reviews → product & reviews)
  +: reduction in the overall size of the working set and shorter disk access time for the most used data
  -: the subset must be managed and pulling extra data requires extra time
- TREE: Store the full path from a node to the top of the hierarchy as a list of the parents
  +: increased performance by avoiding multiple JOINs
  -: updates to the graph must be managed in the application

QUERIES

db.xyz.find({},{})
.aggregate([])
.findOne({},{})
.updateOne({},{})
.updateMany({},{})

- OR → $or [{"x": ...}, ...]
- NOT → {$ne: "x"}

same for $nor, $and

- AGGREGATE
  - $limit
  - $group → $sum, $min, $avg, $max, $min ($sum:1 = count)
  - $match → filter
  - $unwind → one document per element in specified array
  - $project → filter certain fields specifically
  - $sort → -1: DESC, 1: ASC

- UPDATE →
  - $inc: amount (increment)
  - $currentdate
  - $set
    - {"x": "y"} set field x value to y
  - $mul (multiply)
  - $unset → remove specific value

$push: value → Array (adds value to array)

$mul → $unset ~ remove specific value

multiply

- the $ before a field is the value of said field ✓

$in: [ ] — match any value of the array

$nin: [ ]

$gt / $gte $eq

$lt / $lte