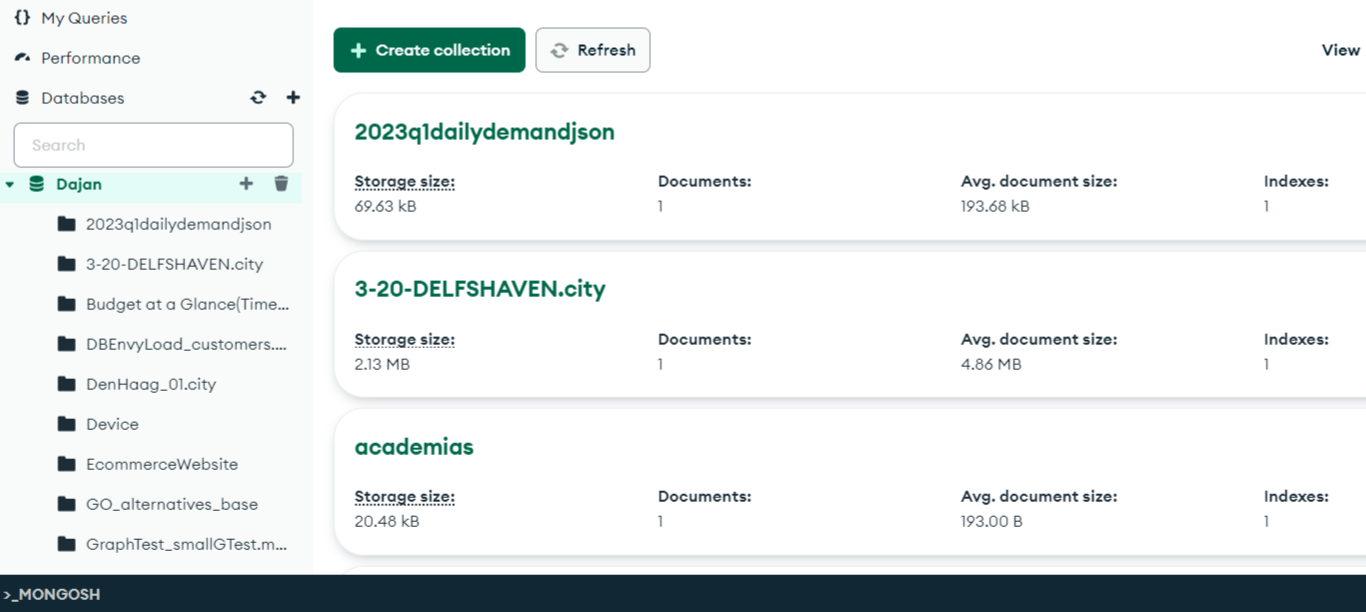
**MongoDB Views and Indexing**

After having imported all .json files in separate collections in the MongoDB database, we continue the subsequent tasks to create views and indexes after having studied data they contain.



*1. Introduction;*

This report explores the use of MongoDB views and indexing to enhance data access and presentation. We will provide an overview of these features, along with practical examples and explanations of how they can be applied to various datasets commonly used with MongoDB.

*2. MongoDB Views;*

Tailoring Data Perspectives MongoDB views are virtual collections that allow you to present data from existing collections in a different format or structure, without modifying the underlying data. Views are valuable tools for simplifying complex queries, aggregating data, or providing alternative perspectives on your information.

*2.1 Views with Filtering* can help you focus on specific subsets of your data.

View 1: Filtered Tweets

db.createView("foda\_tweets", "tweets", [{$match: {"text": /foda/i}}])

The tweets collection contains social media data from Twitter, including tweet content, user information, and metadata.

This view filters the tweets collection to include only tweets containing the word "foda" (case-insensitive). We chose this view to focus on a specific subset of tweets, which could be used for sentiment analysis or tracking the usage of particular terms.

View 2: Popular Stories

db.createView("popularStories", "stories", [ { $match: { diggs: { $gt: 100 } } }])

The stories collection likely contains articles or news items, possibly from a platform similar to Digg or Reddit, where users can vote on content.

This view includes only stories with more than 100 diggs (votes). We chose this view to quickly access popular content, but in real life there might be applications of this for content curation or trend analysis.

View 3: Airline Routes

db.createView("airline\_routes", "routes", [ {$project: {\_id: 0, airline: "$airline.name", src\_airport: "$src\_airport",dst\_airport: "$dst\_airport",stops: "$stops",airplane: "$airplane"}}])

The routes collection contains information about airline routes, including details about airlines, airports, and flight characteristics.

This view projects specific fields from the routes collection, simplifying the data structure. We chose this view to focus on essential route information but flight search applications or route analysis are key points in real-life situations of this dataset.

View 4: Filtered SEMA Data

db.createView("filtered\_sema", "sema", [{ $project: { \_id: 1, user\_id: 1, data: { PARTICIPANT\_ID: 1, STUDY\_ID: 1}}}])

The sema collection likely contains data from scientific or medical studies, with information about participants, studies, and various measurements.

This view projects specific fields from the sema collection, including nested fields within the data object. We chose this view to focus on key study information, which could be valuable for researchers analyzing study results or participant data.

View 5: Filtered Publications

db.createView("filtered\_publications", "publications", [{$project: { \_id: 1, source: 1, request\_date: 1}}])

The publications collection contains information about academic or scientific publications, including metadata like source, publication date, and citation information.

This view projects specific fields from the publications collection. We chose this view to focus on key publication details, which could be beneficial for literature review applications or citation analysis.

View 6: Filtered Metazoa Data

db.createView( "filtered\_metazoa\_simple\_view", "metazoa\_13\_mapping\_GO\_P\_corrected",[{"$match":{"term\_accession": "GO:0000002"}}])

The metazoa\_13\_mapping\_GO\_P\_corrected collection contains data about gene-term associations for metazoa (animals).

This view filters the collection to include only documents where the term\_accession is 'GO:0000002'. This helps in concentrating on a specific biological process or function.

*2.2 Views with Aggregation*

View 7: Aggregated Metazoa View

db.createView( "aggregated\_metazoa\_view", "metazoa\_13\_mapping\_GO\_P\_corrected",[{ "$group": {"\_id": "$term\_accession", "unique\_genes": { "$addToSet": "$gene\_symbol" }, "gene\_count": { "$sum":1}}}, {"$project": {"\_id": 0, "term\_accession": "$\_id","unique\_genes": 1, "gene\_count": { "$size": "$unique\_genes" }}}])

The metazoa\_13\_mapping\_GO\_P\_corrected collection contains data about gene-term associations for metazoa (animals).

This aggregated view groups the dataset by the `term\_accession` (biological process or function) and summarizes the associated `gene\_symbols` (genes mapped to that process). This provides a high-level overview of the data, which could be useful for researchers analyzing the relationships between biological processes and genes.

View 8: Aggregated Data View

db.createView("data\_view","data",[{$project: {\_id: 1,st: 1,ts: 1,position: 1,elevation: 1,callLetters: 1, qualityControlProcess: 1, dataSource: 1, type: 1, airTemperature: 1, dewPoint: 1, pressure: 1, wind: 1, visibility: 1, skyCondition: 1, sections: 1, precipitationEstimatedObservation: 1}},{$group: {\_id: {st: "$st",ts: "$ts",position: "$position",elevation: "$elevation",callLetters: "$callLetters",qualityControlProcess: "$qualityControlProcess",dataSource: "$dataSource",type: "$type"}, airTemperature: { $push: "$airTemperature" }, dewPoint: { $push: "$dewPoint" }, pressure: { $push: "$pressure" }, wind: { $push: "$wind" }, visibility: { $push: "$visibility" }, skyCondition: { $push: "$skyCondition" }, sections: { $push: "$sections" }, precipitationEstimatedObservation: { $push: "$precipitationEstimatedObservation" } } } ] )

The data collection contains meteorological observations, including various weather measurements and attributes.

This aggregated view organizes the data by several key fields (such as station, timestamp, and position) and compiles the remaining fields into arrays. This approach aids in identifying trends and patterns within the weather data.

View 9: Aggregated Publications View

db.createView("publications\_aggregated", "publications", [

{ $group: { \_id: { source: "$source", published: "$published", journal\_ref: "$journal\_ref" }, total\_citations: { $sum: { $toInt: { $first: "$total\_citations" } } }, total\_downloads: { $sum: { $toInt: { $first: "$total\_downloads" } } }, count: { $sum: 1 } } },

{ $project: { \_id: 0, source: "$\_id.source", published: "$\_id.published", journal\_ref: "$\_id.journal\_ref", total\_citations: 1, total\_downloads: 1, count: 1 } }, { $sort: { total\_citations: -1, total\_downloads: -1 } }]);

The publications collection contains information about academic or scientific publications, including metadata like source, publication date, and citation information.

This aggregated view categorizes the publications by source, publication date, and journal reference, while computing the total number of citations, downloads, and publications for each category. This can help in analyzing publication trends, identifying popular journals, or highlighting highly cited works.

View 10: Aggregated Trips View

db.createView("trips\_aggregated", "trips", [

{ $group: { \_id: { start\_station\_id: "$start station id", start\_station\_name: "$start station name", end\_station\_id: "$end station id", end\_station\_name: "$end station name", user\_type: "$usertype", birth\_year: "$birth year", gender: "$gender" }, total\_trips: { $sum: 1 }, total\_duration: { $sum: "$tripduration" }, avg\_duration: { $avg: "$tripduration" } } },

{ $project: { \_id: 0, start\_station\_id: "$\_id.start\_station\_id", start\_station\_name: "$\_id.start\_station\_name", end\_station\_id: "$\_id.end\_station\_id", end\_station\_name: "$\_id.end\_station\_name", user\_type: "$\_id.user\_type", birth\_year: "$\_id.birth\_year", gender: "$\_id.gender", total\_trips: 1, total\_duration: 1, avg\_duration: 1 } },{ $sort: { total\_trips: -1 } }]);

The trips collection contains data about bike sharing trips, including information about the start and end stations, user demographics, and trip duration.

This aggregated view organizes the trip data by fields such as start and end stations, user type, birth year, and gender, and computes the total number of trips, total trip duration, and average trip duration for each group. This helps in examining usage patterns, identifying popular routes, and understanding user behavior in a bike-sharing system.

View 11: Aggregated Fungi View

db.createView("fungi\_0\_mapping\_GO\_P\_corrected\_aggregated", "fungi\_0\_mapping\_GO\_P\_corrected", [{ $group: { \_id: { term\_accession: "$term\_accession", gene\_symbol: "$gene\_symbol" }, count: { $sum: 1 } } },{ $sort: { count: -1 } }

]);

The fungi\_0\_mapping\_GO\_P\_corrected collection contains data about gene-term associations for fungi.

This aggregated view organizes the data by the term\_accession and gene\_symbol fields, and computes the number of documents for each grouping. This can assist researchers in exploring the relationships between biological processes and genes in fungi.

View 12: Aggregated Tag Counts

db.createView("tag\_count\_aggregated","tag\_count",[{$group:{\_id:{item\_id:"$item\_id",tag\_id:"$tag\_id"},total\_count:{$sum:"$num"}}},{$sort:{total\_count:-1}},{$limit:3}])

The tag\_count collection contains information about the number of times a specific tag has been applied to an item.

This aggregated view organizes the data by the item\_id and tag\_id fields and sums the num field for each group. This approach is useful for identifying the most popular tags associated with specific items, which can enhance content recommendation and categorization.

*3. MongoDB Indexing*

Indexing in MongoDB improves query performance by creating data structures that allow for quick location of specific data. Indexes can speed up queries that filter, sort, or perform range operations on specific fields.

3.1 Indexes for the "tweets" Collection

db.tweets.createIndex({ "created\_at": 1 })

This index is created on the `created\_at` field, which will improve the performance of queries that filter or sort by the tweet creation date. This could be useful for time-based analysis or filtering of the tweets.

3.2 Indexes for the "companies" Collection

db.companies.createIndex({ "name": 1 })

This index is created on the `name` field, which will enable efficient lookups of companies by their name. This could be beneficial for a business directory application.

3.3 Indexes for the "grades" Collection

db.grades.createIndex({ "student\_id": 1 })

This index is created on the `student\_id` field, which will allow for fast retrieval of a specific student's grades. This would be crucial for an academic management system.

3.4 Indexes for the "publications" Collection

db.publications.createIndex({ "source": 1, "published": 1, "journal\_ref": 1 })

This index is created on the `source`, `published`, and `journal\_ref` fields. It will help improve the performance of queries that filter or sort by these fields, such as the aggregation pipeline used to create the "publications\_aggregated" view.

3.5 Indexes for the "trips" Collection

db.trips.createIndex({ "start station id": 1, "end station id": 1, "usertype": 1, "birth year": 1, "gender": 1 })

This index is created on the `start station id`, `end station id`, `usertype`, `birth year`, and `gender` fields. It will support the `$group` and `$sort` stages in the aggregation pipeline used to create the "trips\_aggregated" view, allowing MongoDB to efficiently group the documents and sort the results.

3.6 Indexes for the "fungi\_0\_mapping\_GO\_P\_corrected" Collection

db.fungi\_0\_mapping\_GO\_P\_corrected.createIndex({ "term\_accession": 1, "gene\_symbol": 1 })

This index is created on the `term\_accession` and `gene\_symbol` fields. It will improve the performance of queries that filter or group by these fields, such as the aggregation pipeline used to create the "fungi\_0\_mapping\_GO\_P\_corrected\_aggregated" view.

*4. Conclusion*

MongoDB views and indexing are powerful tools for optimizing data access and presentation. Views allow for customized data representations without modifying the underlying data, while indexes significantly improve query performance.

The datasets we've explored (tweets, stories, airline routes, scientific studies, publications, weather data, companies, grades, gene-term associations) represent a diverse range of applications where MongoDB can be effectively used. By creating appropriate views and indexes for each dataset, we can tailor the data access patterns to the specific needs of each application, improving both performance and usability.

When designing MongoDB applications, consider using views for complex data transformations and create appropriate indexes to support your most common query patterns. Always it’s important to analyze your specific dataset and application requirements to determine the most effective view and indexing strategies.