

D597 MKN1 Task 2

Non-relational Database Design and Implementation

Shanay Murdock

smurd32@wgu.edu

011377935

D597 Data Management
MS, Data Analytics

A. Scenario Context

A1. Business Problem

HealthFit Innovations is looking to develop a new platform called "HealthTrack," which will be used to collect and analyze data, and use the data to create personalized health recommendations.

HealthTrack will pull data from multiple sources, including wearable devices, electronic health records (EHRs), medical imaging systems, and patient-reported outcomes, and therefore needs a flexible system designed to adapt to additional business needs in the future.

When HealthFit Innovations' database structure was initially set up, time and budget constraints impacted the design, leaving a database that is beginning to show its limitations.

As the amount of data collected and sources of data grow, and the need for different types of analytics arises, the data is not optimized for either storage or analytics. There is little flexibility for adaptation to changing analytics requirements and to support the rapid growth HealthFit expects as it provides real-time analytics and designs health suggestions based on user metrics.

HealthFit Innovations is in need of a solution to implement a document-based, non-relational NoSQL database so that it can be scalable as new products, features, and analytics needs arise with the growth of the platform, to collect user data, and to provide real-time analytics with data collected from the health trackers.

A2. Database Justification

As HealthFit will be accumulating data from a number of data sources, a database is the best way of collecting and storing this data. A non-relational database, in this case the document-based database MongoDB, is the proposed solution for HealthFit's needs as it will provide an optimized and secure way of storing semi-structured and unstructured data while offering scalability as HealthFit expands the sources of data collection and input. The security, flexibility, optimization, and performance are especially important considerations given that there is personally identifiable information (PII) and real-time data collection and reporting.

MongoDB's horizontal scaling will be particularly helpful as data volumes grow and the number of data sources grow, allowing HealthFit's data collection to scale as new features are added to the HealthTrack platform. All existing data can be preserved and transferred to the new database model.

A3. Database Type

MongoDB is a document-based non-relational database management system that excels at storing semi-structured and unstructured data, including JSON files, maintaining schema flexibility, providing extensive query capabilities, and allowing for horizontal scalability. MongoDB

is an excellent tool for companies and projects that use Agile development practices and allows for the quick testing of changes that may need to occur as HealthFit's business scales.

Developers also find MongoDB quite easy to use as MongoDB stores its data in a JSON-like structure, where JSON is a widely-used data format that's used in web APIs and applications, making it ideal for a HealthFit application. The complex types of data that HealthFit is accumulating will be easily manageable in MongoDB by software engineers and data professionals alike.

A4. Data Usage

The proposed solution will store the data in a non-relational, document-based way. This allows for software engineers and data professionals to perform data collection, data sharing, descriptive analysis, and predictive analysis.

- Data Collection: Software and hardware engineers made it so that HealthTrack is able to get constant, real-time updates of patient data through wearable devices, electronic health records, medical imaging systems, and patient-recorded outcomes.
- Data Sharing: The data accumulated is structured to give patients personalized health insights, such as sleep schedule recommendations, workout recommendations, stress reduction techniques, etc.
- Descriptive and Predictive Analysis: The data accumulated allows data professionals including data analysts, data scientists, and machine learning engineers to program the ability to draw insights from the data, which can be fed into recommendation engines that make the recommendations mentioned above a possibility. The analytics performed can also provide strategic data-informed business insights on how to continue growing as a platform.

B. Scalability

Using MongoDB as the database solution will enable HealthFit to allow for high traffic volume and high data volume through horizontal scaling through sharding, replication, cloud services, and configurations for optimized queries and simultaneous access.

Horizontal scaling through sharding refers to a method of distributing data across multiple servers, with each server holding a portion of the data. This allows for distributed processing and storage. MongoDB allows shard clusters, which provides data distribution (so no single server gets overloaded) and redundancy, so that if a server fails, backups are available. Horizontal scaling allows a database system to be enhanced by adding additional servers so that one single server doesn't need to be continually updated for performance.

Replication is done by maintaining identical copies of data across servers so that data is available in case of a server failure and provides high availability so that data professionals and clients receiving data don't overload any single server.

MongoDB Atlas is a cloud-based MongoDB service that offers the ability to scale computing resources like CPU, memory, and storage based on usage and needs, allowing for rapid expansion without manual intervention.

Optimizations can also be put in place in the form of indexing strategies and query optimizations to ensure that regularly accessed data can be retrieved quickly while still processing large volumes of data efficiently.

C. Privacy & Security

MongoDB features built-in encryption features to ensure data is secure at rest and in transit, meaning while it's stored in the database and when it's being sent to or from the database. This ensures privacy is maintained as personal identifiable information (PII) is collected and needs to remain HIPAA compliant to protect patient health data.

MongoDB also has built-in features around authentication and authorization, giving Role-Based Access Control (RBAC) which follows the principle of least privilege, the principle that gives a user only as much access to the data as they need and only for as long as they need it (Reis & Housley 2022). MongoDB is excellent at giving role permissions at a very granular level.

These strategies will make more efficient use of resources, improve the user experience from both the data professional and the patient experiences, and future proof HealthFit's HealthTrack platform to expand without the need for re-architecting the solution.

D. Implementation

This section walks through the technical implementation of the proposed MongoDB database solution.

D1. Database Instance

The data is stored as a series of document collections in a database called `D597_Task_2`. The primary collections used will be called `medical` and `fitness_trackers`. While we only have data for `medical` and `fitness_trackers` at present, additional information will be provided for recommendations to create the structure for the expansion of the HealthTrack platform to include collections for `users`, `wearable_data`, `ehr` to store electronic health records, `imaging_data`, and `patient_recorded_outcomes`.

```
>_MONGOSH
> use D597_task_2
< switched to db D597_task_2
```

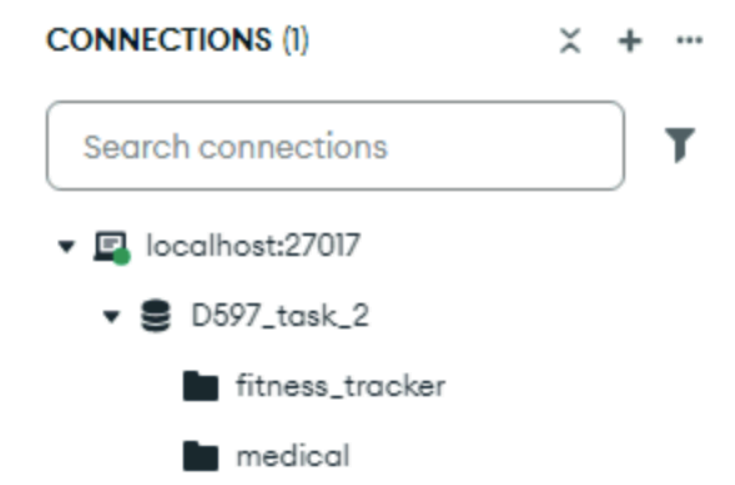
Script to add the `fitness_tracker` collection:

```
> db.createCollection("fitness_tracker")
< { ok: 1 }
D597_task_2>
```

Here is a list of additional collections that can be added to prepare HealthTrack for its future data needs:

```
// Create additional collections
db.createCollection("users");
db.createCollection("wearable_data");
db.createCollection("ehr");
db.createCollection("imaging_data");
db.createCollection("patient_recorded_outcomes");
```

The database instance:



D2. Insert Records

Insert records into `medical` collection:

```
mongoimport --db db --collection medical --file "C:\WGU\D597\Task
2\Scenario 1\medical.json" --jsonArray
```

Insert records into `fitness_tracker` collection:

```
mongoimport --db db --collection fitness_trackers --file "C:\WGU\D597\Task 2\Scenario 1\fitness_trackers.json" -- jsonArray
```

localhost:27017 > D597_task_2

Sort by: Collection Name ⌵

fitness_tracker				
Storage size: 4.10 kB	Documents: 565	Avg. document size: 314.00 B	Indexes: 1	Total index size: 4.10 kB
medical				
Storage size: 4.10 kB	Documents: 100 K	Avg. document size: 235.00 B	Indexes: 1	Total index size: 4.10 kB

D3. Queries

Question 1: Find all patients born before 1990:

localhost:27017 > D597_task_2 > medical

Documents 100.0K Aggregations Schema Indexes 1 Validation

🕒 { "date_of_birth": { \$lt: "1/1/1990" }}






The result of the query without optimizations:

Query Performance Summary


- 📄 190 documents returned
- 🔍 100000 documents examined
- 🕒 108 ms execution time
- 🔄 Is not sorted in memory
- 📄 0 index keys examined
- ⚠️ No index available for this query. ?

The optimizations added will be covered in the next section. Notice that prior to the optimizations, the query took 10ms to run. Following the optimization, the query took 1 ms to run.

Query Performance Summary

-  **158** documents returned
-  **158** documents examined
-  **1 ms** execution time
-  **Is not** sorted in memory
-  **158** index keys examined


Query used the following index:


`date_of_birth` 

Question 2: Find patients with “allergies” information that is not “None”:


localhost:27017 > D597_task_2 > medical


Documents 100.0K Aggregations Schema Indexes 1 Validation

 { "allergies": { \$ne: "None" } }

 ADD DATA

 EXPORT DATA

 UPDATE

 DELETE

The result of the query without optimizations:

Query Performance Summary

 **21000** documents returned

 **100000** documents examined

 **102 ms** execution time

 **Is not** sorted in memory

 **0** index keys examined

 **No index available for this query.** 

The optimizations added will be covered in the next section. Notice that prior to the optimizations, the query took 102ms to run. Following the optimization, the query took 49 ms to run.

Query Performance Summary

 **21000** documents returned

 **21000** documents examined

 **49 ms** execution time

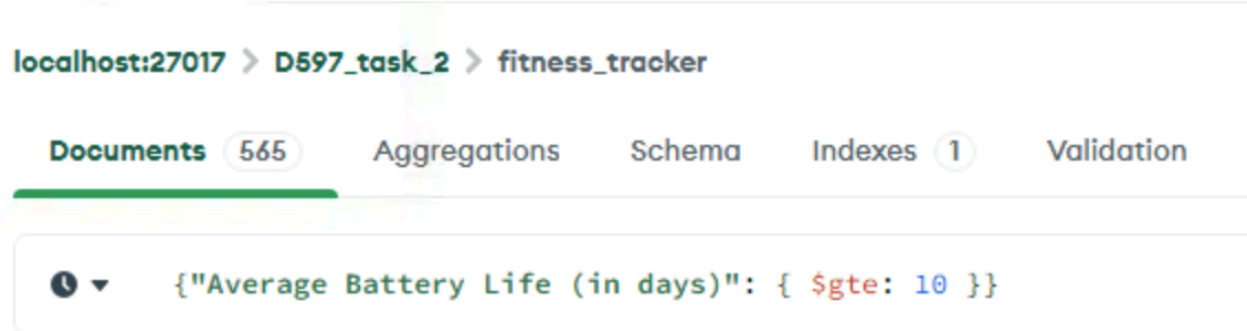
 **Is not** sorted in memory

 **21001** index keys examined

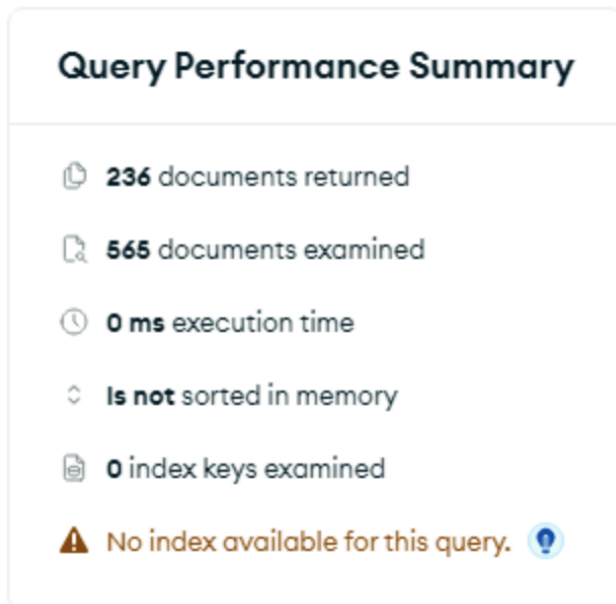
Query used the following index:

allergies 

Question 3: Retrieve all models offering an average battery life of 10 days or more:








The result of the query without optimizations:



The optimizations added will be covered in the next section. Notice that prior to the optimizations, the query took 0ms to run. Following the optimization, the query took 0 ms to run. Note: The “fitness_tracker” collection is much smaller in number of documents than the “medical” collection is.

Query Performance Summary

-  **236** documents returned
-  **236** documents examined
-  **0 ms** execution time
-  **Is not** sorted in memory
-  **236** index keys examined

Query used the following index:

Average Battery Life (in days) ↑

D4. Optimization

It's recommended that indexes are added to fields of data that are regularly accessed, such as gender, fitness tracker model, and last appointment date.

Optimization 1: Index on "date_of_birth":

```
db.medical.createIndex({ "date_of_birth": 1 })
```

```
> db.medical.createIndex({ "date_of_birth": 1 })
< date_of_birth_1
D597_task_2> |
```

Optimization 2: Index on "allergies":

```
db.medical.createIndex({ "allergies": 1 })
```

```
> db.medical.createIndex({ "allergies": 1 })
< allergies_1
D597_task_2>
```

Optimization 3: Index on "Average Battery Life (in days)"

```
db.fitness_tracker.createIndex({ "Average Battery Life (in days)": 1 })
```

```
> db.fitness_trackers.createIndex({ "Average Battery Life (in days)": 1 })
< Average Battery Life (in days)_1
D597_task_2> |
```

E. Panopto Walkthrough

<https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=8e65f48e-fc00-49ce-8278-b26b013c943b>

F. Resources

Bradshaw, S., Brazil, E., & Chodorow, K. (2019). *MongoDB: The Definitive Guide: Powerful and Scalable Data Storage* (3rd ed.). O'Reilly Media.

Reis, J., & Housley, M. (2022). *Fundamentals of Data Engineering: Plan and Build Robust Data Systems* (1st ed.). O'Reilly Media.

Schiller, R. J., & Larochelle, D. (2024). *Data Engineering Best Practices*. Packt Publishing.

WGU Course Resources.