# Database Schema Report

# By: Michael Watts, Gautham Ragav Thirugnanam, Khoushik Rajapandian

## 

## Database Design Schema and Selection

For our backend, we have elected to employ a MongoDB Document Store Schema, presented below in both word and diagram form:

Routes: [

routeNum:{

Name: ‘String’,

Code: int

DepartureData:{

cityName: ‘String’,

cityCode: ‘String’

},

Destination:{

cityName:‘String’,

cityCode: ‘String’

},

Departure:{

Hour: int,

Minutes: int,

minutesFromMidnight: int

},

Travel:{

Hour: int,

Minutes: int,

minutesFromMidnight: int

},

},

]

Drivers:[

Id:{

FName: ‘String’,

LName: ‘String’,

FullName: ‘String’,

Age: int,

Hometown: {

City: ‘String’,

State: ‘String’

}

},

]

Assignments: [

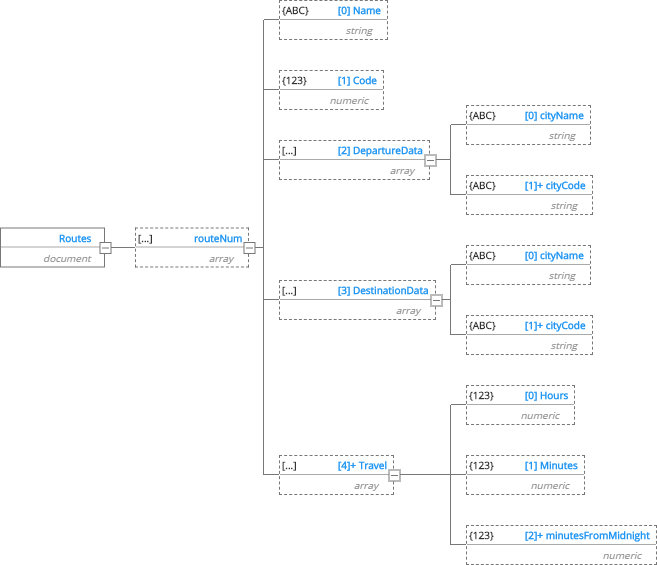
Id:{

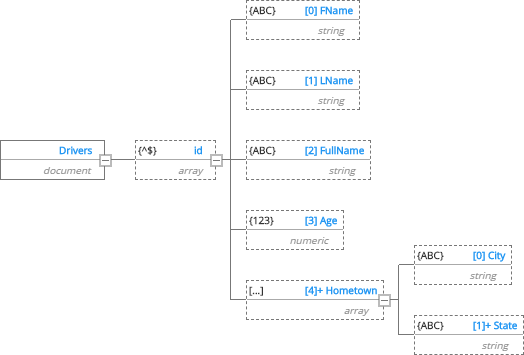
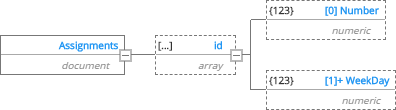
Number: int,

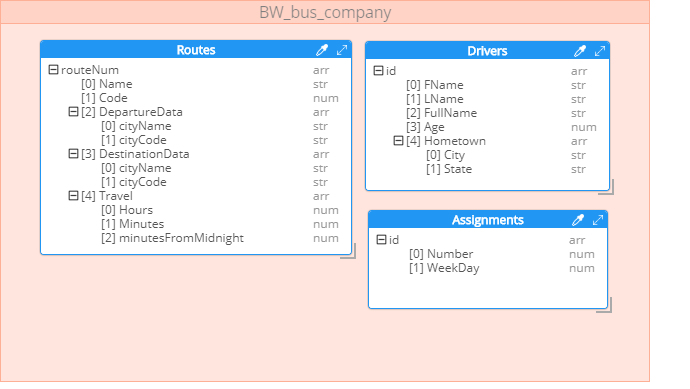
Weekday: int

},

]







We elected to use MongoDB due to its overwhelming popularity in usage for NoSQL backends. According to <https://db-engines.com/en/ranking>, it is the 4th most popular choice for a backend Database overall and more popular than any of the provided options per the assignment. It also provides very well defined support with its documentation and active community across the web, particularly in Stackoverflow. We elected to employ a document based NoSQL database due to this being the supported NoSQL backend for MongoDB. We elected to embed the documents for Route and Driver data below an ID key. This allows us to use referential embedding for the Driver Assignment data, using ID and Number to directly refer to the respective document for each instance for Routes and Drivers. It also allows us to treat the ID column for specific documents as a primary key of sorts for Route and Driver documents, assisting in their management. We also elected to create several combination keys. For Route objects, any time that was divided between hours and minutes, we combined into one attribute representing the total number of minutes from midnight. This aids queries based on time constraints. For example: return all routes that leave before 6:17AM. We can simply convert 6:17 am into its corresponding minutes since midnight, then only check on that one attribute. We can accomplish this conversion with a simple calculation based on each of these attributes. Likewise, for the Driver documents, we combine the first and last name into one attribute to facilitate queries that involve filtering drivers based on entire names. E.G return all drivers with no N in their name.

## Database design choice – advantages.

To design our database, we decided to go with MongoDB as our database with our application being developed in Python.

The reasons for choosing MongoDB are:

* It’s a document-oriented database with data stored in JSON. This implies that storing of information is convenient as the JSON can be nested for elaborate and complex data types.
* The document structure helps make the schema more flexible and dynamic.
* The queries are in JSON, and thus can be easily computed/composed.
* MongoDB has a horizontal, scale-out architecture that can support huge volumes of data
* MongoDB Atlas is cloud database service form MongoDB that helps us to deploy the database on popular cloud service providers such as AWS, Azure with good levels of data security. This also comes with a variety of automation tools making the database accessible, efficient and easy to use.
* MongoDB has a growing community of developers and consultants, so it is easy to get help.
* MongoDB can be used from all major languages. We have chosen to use Python as the development platform.
* MongoDB can be accessed from all major ETL and data management systems.
* Represent data with natural clusters and variability over time or in its structure
* Since the database can be hosted in the cloud with clusters, it enables easy and effective collaboration of multiple teams. As we are all working from home for the most part, this is an important consideration.
* One more benefit of hosting MongoDB in the cloud is that our data repository can be scaled to extremely large sizes.
* As MongoDB can be deployed across different clouds (private, public or hybrid) we could modify our deployment based on the project size and security requirements.
* Python has the pymongo module which is generally used as the driver for the MongoDB database.

<https://pypi.org/project/pymongo/>

<https://www.mongodb.com/>

<https://www.mongodb.com/cloud/atlas>

Limitations:

* The initial flexibility with the schema offered by document stores can eventually result in data being less consistent as the size grows. In NoSQL database systems such as MongoDB data can eventually become inconsistent because of lack of referential integrity as compared to traditional database schemas. It is due to no enforced relationship constraints and each document within a collection can take on its own set of fields. This has resulted in data integrity issues amongst the early adopters of NoSQL technologies. We plan on addressing this problem in our application by referencing documents using document identifiers
* The first limitation of weak referential integrity also leads to weak joins among documents. There is also no equivalent to a direct join operator in SQL and joining documents requires denormalizing which is cumbersome. Although this can be addressed by application code, it affects performance, adds complexity, and executes much slower as compared to executing joins through specialized code at the database level.
* Security is a major concern for NoSQL databases. NoSQL databases like MongoDB lack sophisticated security features offered by mature relational databases like Oracle and MS SQL Server. It lacks encryption for data files and the authentication/ authorization are basic and considered weak compared to industry standards.
* Unstructured queries as compared to traditional SQL queries. Learning a new language can also be tedious and cumbersome for developers who are familiar with SQL which has been the language of choice for database systems for many decades. This can be addressed by keeping an open mind and embracing the newly offered capabilities of document stores.
* NoSQL is still in the early stages compared to mature relational databases, so the constant change and updates means more maintenance, bugs and inconsistent documentation. We plan on addressing this issue by closely monitoring updates posted by developers at MongoDB and keeping up with the changes.
* Another major disadvantage of MongoDB is the limited size of the document which cannot exceed 16MB. Although this may not be a concern for the early stages of the application, this could mean large scale changes as the application matures.