This week we delved into the NoSQL databases, and gained useful knowledge on the efficacy of NoSQL databases. We gained an understanding of what types of information is better housed in NoSQL, including structured an unstructured data, and the scenarios where a traditional RBMS database would be more useful. Below we will delve into the topics of NoSQL and gain a deeper understanding of the usefulness of NoSQL, types of NoSQL databases out in the marketplace. Following it all up with a summary of the pros and cons of utilizing this type of database.

Why is NoSQL useful?

Since we’re working with MongoDB this week, I thought I would take advantage of their article entitled “NoSQL Explained.” NoSQL is useful because they’re scalable databases that are designed with the ability to handle a large quantity of rapidly changing structured and unstructured data, the rapidly changing and unstructured part is what is not handled well by your standard relational database. NoSQL offers the ability to work through quick agile sprints, with the flexibility to push to production frequently even multiple times per day, which again is not a functionality that the standard relational database can handle. You also have the ability to scale out architecture as you expand operations, again not handled well by RBMS. Lastly flexible object-oriented programming integration is offered via these platforms, allowing for a well-organized database and avoiding the downfalls of what many refer to as “spaghetti code.”

NoSQL Databases Explained Retrieved from https://www.mongodb.com/nosql-explained

What are use cases for NoSQL?

Tim Stephan of Network World provided a great article outlining several use cases where NoSQL would outperform SQL. I will outline five of the use cases covered and provide a brief summary of his findings:

1. Personalization: in order for an organization to provide a personalized experience it requires vast amounts of data including demographic, behavioral, contextual etcetera. Capturing this amount/type of data requires a database to scale to the requirements needed, which would overload your traditional relational database.
2. Profile Management: user profile management for web and application base services requires a vast amount of data storage, a NoSQL database can easily distribute out databases by adding commodity servers which makes it easier and less costly to scale.
3. Real-time big data: for an organization or service to access data in real time using a traditional relational database, it would require different environments for the databases to be maintained in. NoSQL allows for both a front end solution that can manage operational data and back end to store analytic results from Hadoop.
4. Content Management: NoSQL offers the ability to select a variety of content for aggregation and presentation easily through databases that do not need to be pre-defined.
5. Catalog: since catalogs are not just a listing and serve as point of service terminals and self service kiosks; the relational database would lag in performance of such tasks due to the large amount of data being collected, and the need to utilize multiple databases. NoSQL can with its flexible data model aggregate data in a single catalog with a single database – bypassing the complexity that a relational database would be confronted with.

Stephan, Tim. 10 use cases where NoSQL will outperform SQL October 30, 2015 Retrieved from: https://www.networkworld.com/article/2999856/10-use-cases-where-nosql-will-outperform-sql.html

What are some of the different NoSQL databases, and in what cases are they useful?

Trust Radius published an article outlining the best NoSQL databases available in 2020, below is a summary of four of the higher rated ones:

|  |  |
| --- | --- |
| NoSQL Database Name | Features |
| Redis | In-memory, multi model database that supports multiple data structures: strings, hashes, lists sets, range queries, bitmaps, hyperloglogs, and geospacial indices. It is open source, and is supported on your standard operating systems: Windows, Mac, Linux, as well as your standard mobile os including Apple, Android, Windows, Blackberry and Mobilweb. Its is supported across the globe. It performs well with: atomic lookups, quick lookups, and is widely supported . |
| Couchbase | I personally just like the name of this one. This database features a multi-model NoSQL database. Its data platform was built on top of an open source foundation, and boasts a “massively interactive enterprise.” It features: multi-dimensional scaling, cross data center replication, flexible schema, full text search, analytics, mobile an IOT integration, memory first architecture, query, cloud native, and key value. It works on your standard OS, Mac, Windows, and Linux, but does not offer free trials. |
| Amazon DynamoDB | Offered from our friends from Amazon, and is a top rated choice by reviewers on Trust Radius. It offers serverless webapps, mobile apps, and backend storage. Reviews state that it has low latency, and high performance. Real-time encryption offers a high level of security. |
| MongoDB | An open-source document oriented database, it stores data in JSON documents with dynamic schemas; however users can change their views to the traditional table format. This use of JSON storage increases speed of and ease of implementing data integration in certain applications. It offers full monitoring, automated database monitoring, and fully managed backups. Free trials are available a plus, but entry level set up will come at a cost. It get high ratings from peer-reviewers due to its high performance, good documentation, and scalability. |

NoSQL Databases <https://www.trustradius.com/nosql-databases>

What are pros and cons of NoSQL vs SQL?

Now that we are well versed in the differences between SQL and NoSQL we can delve into the use cases for each type of database, for this I leveraged an article written on the Software Testing help website as cited below.

NoSQL database usage/implementation is good for:

* Large volumes of structured and unstructured data
* When you need to push updates frequently to production, for an example with Scrum workstreams
* When you prioritize easy scalability

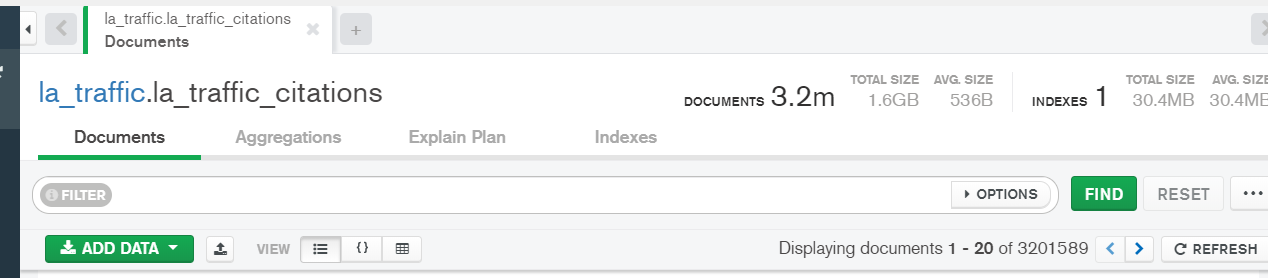
SQL or traditional RBMS function as a better alternative when:

* When you use complex dynamic querying and reporting as NoSQL does not handle this process well
* When consistency is a priority and there are no large-scale changes in data volume
* Time-flexibility for applications is not handled well by NoSQL, again where the standard approach would function better
* NoSQL does not perform dynamic operations well, and cannot guarantee ACID properties

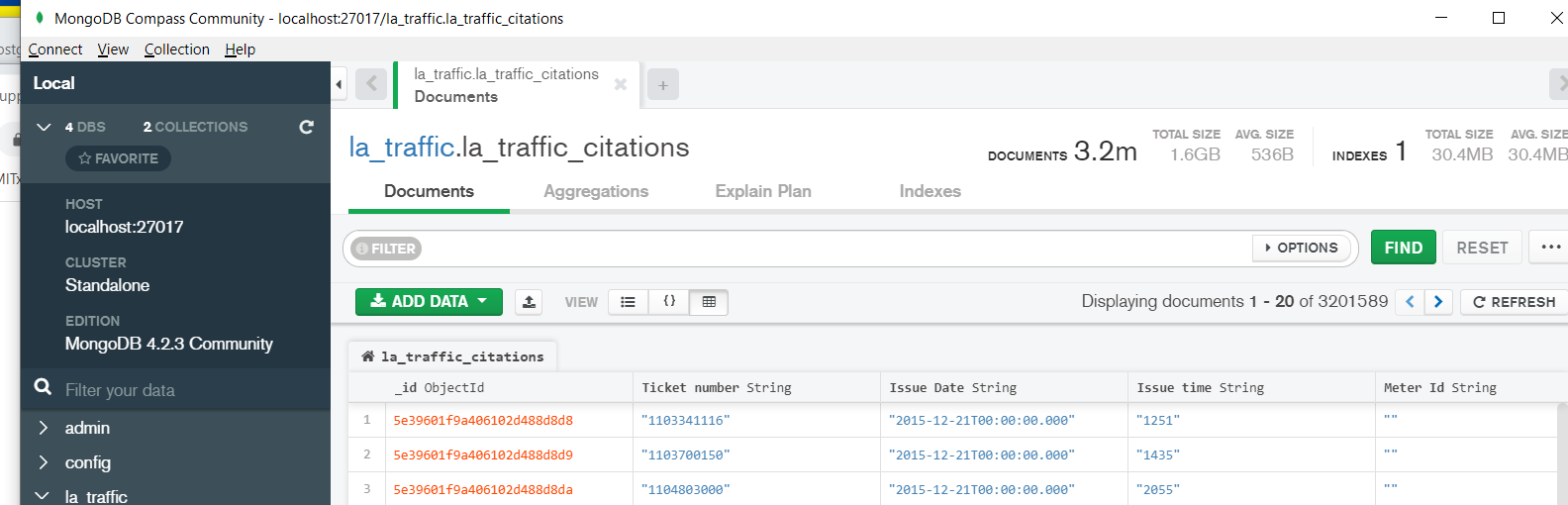
SQL Vs NoSQL Exact Differences And Know When To Use NoSQL And SQL November 10, 2019. <https://www.softwaretestinghelp.com/sql-vs-nosql/>

**Technical Portion**

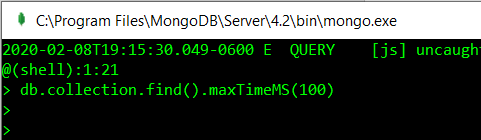
For the technical portion of this week 4 assignment, I installed MongoDB locally, and then loaded in the LA Traffic Citations csv. That I obtained from Kaggle.



I was unable to get the entire dataset loaded, but at 3.2m documents, I felt that there was a good sample size with that load of data. Below is a change of view into a tabular dataset as opposed to the JSON format that was defaulted.

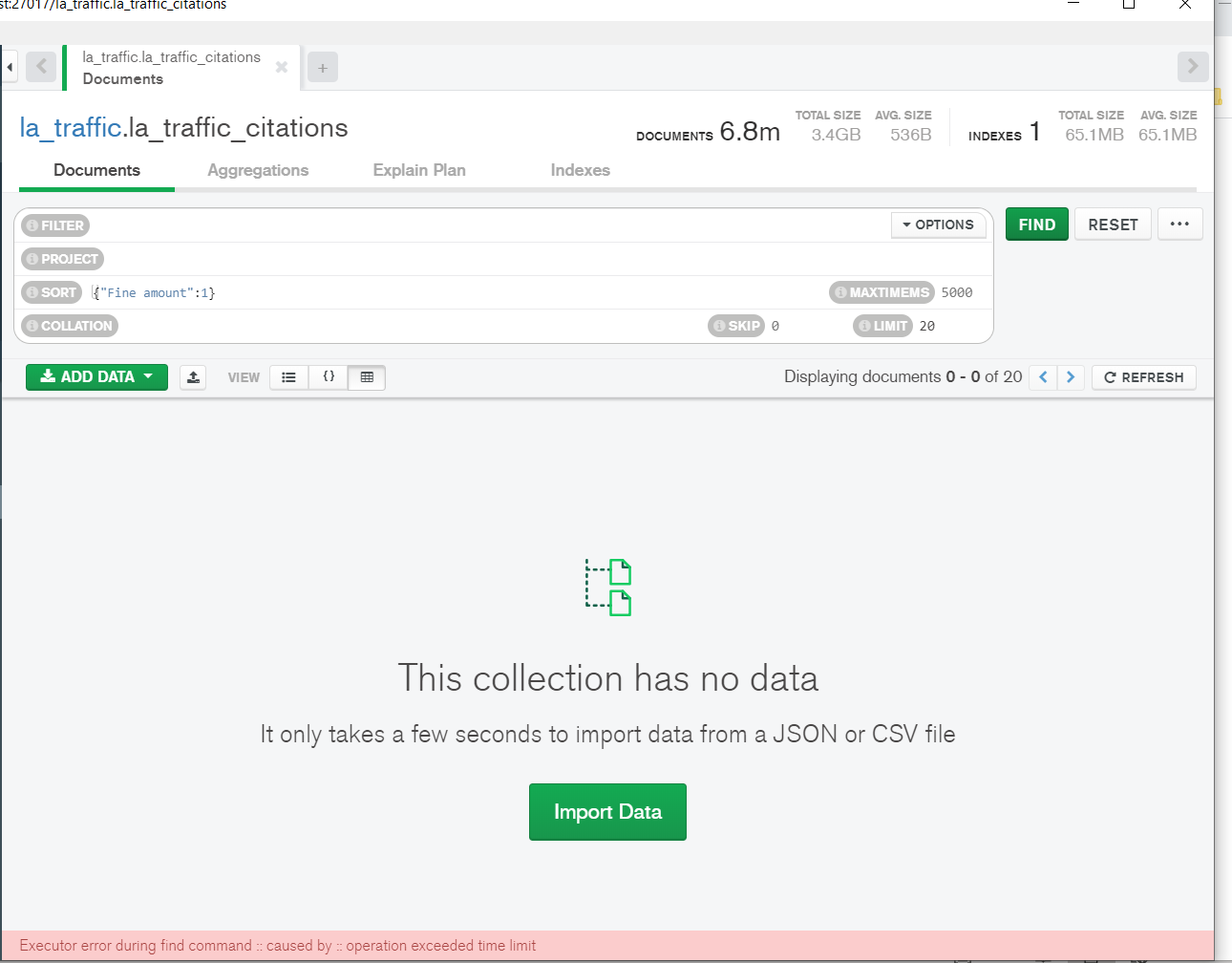


I opened up the options on the query bar, as noted on the supplemental – I conducted a query filtering on the CB1801 meter and sorted by ticket: a couple things to note, the meter data did not contain data from the 6 million or so records that I loaded. In addition, I was running into issues where my query was timing out, when I opted to use a query that didn’t contain the meter id, to later find out that my time out settings had issues. I then navigated to the MongoDB shell; and worked to change my timeout settings:



Which still unfortunately did not resolve the time out errors.

I found that I was running into an issue where the query was timing out, and this would occur after only a couple second 3-4. In researching this issue

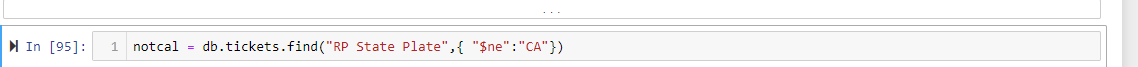


I did end up reloading the database and starting over again.

What are the amounts and violation descriptions for the top 10 most expensive tickets?

|  |  |
| --- | --- |
| Violation Description | Fine amount |
| 8755\*\* | $505 |
| HANDICAP/NO DP ID | $363 |
| 22522 | $363 |
| HANDICAP/CROSS HATCH | $363 |
| DP- RO NOT PRESENT | $363 |
| DISABLED PARKING/NO | $363 |
| DISABLED PARKING/CRO | $363 |
| DP-BLKNG ACCESS RAMP | $363 |
| 2251157A | $363 |
| 3 FT. SIDEWALK RAMP | $363 |

Within Python, extract the entries from the DB where the license plate is not CA.



Create a sorted bar chart of the states from #3: I followed the instructions, and registration tickets does include a make. I’m unsure why the Make errored out on this call.



