|  |  |
| --- | --- |
| Assessment Title | MongoDB Assignment (Shopping World) |

## Competency Details

|  |  |  |
| --- | --- | --- |
| Unit code/s and title/s | ICTPRG554 – Manage data persistence using NoSQL data stores |  |
| Qualification code/s and title/s | ICT50220 – Diploma of Software Development |  |
| Business unit/Work group | BARTS/IT Studies |  |

## Instructions

|  |  |
| --- | --- |
| Method/s of assessment | Product (Create & Written) |
| Overview of assessment | The assessment has four parts. Each part has multiple tasks. The assessment includes determining the database requirements & the type of noSQL to meet the business needs. It compares the differences of the relational database structures and the noSQL flexible schema are addressed. It involves using the queries to manipulate the data with multiple collections. Indexes are determined to enhance the performance of the retrieval. Database scaling are implemented using the partition and multiple sharding of storages. REST API are implemented for the CRUD database operations which can be requested by the client applications. The last part is to document the database design according to organisation document template. Load balancing |
| Task/s to be assessed | This assessment will require you to complete the following tasks   * Part 1 – Database Requirements * Part 2 – Data manipulation * Part 3 – Query with multiple collections, indexing & partitioning/sharding * Part 4 – REST API & Database Documentation |
| Time allowed | Refer to your schedule for submission dates |
| Location of assessment | Assessment can be completed anywhere with access to the resources required. (see Resources Required section below) |
| Decision making rules | To receive a satisfactory outcome for this assessment you must complete all parts correctly.  Word counts are provided as guidance only. |
| Assessment conditions | This assessment must be undertaken where the conditions replicate noise levels and interruptions that people typically experience working in the ICT industry.  This is unsupervised assessment, and you may access any required resources.  This is not group work and must be completed as an individual. |
| Resources required | To complete this assessment, you will require the following:   * Access to Learn with Internet access * Learn resources (ICTPRG554 – ASDS – MongoDB Assignment (ShoppingWorld) – Part 1, Part 2, Part 3 & part 4 student files) * MongoDB, Mongo Atlas, Mongo Compass * Visual Studio Code, Postman * npm node.js, Express, nodemon, cors, mongoose libraries * VMWare Workstation   Word processing software such as Microsoft Word. |
| Result notification and reassessment information | You will be provided feedback and the result for your assignment on TAFESA Learn. You will be and given the chance to resubmit with required corrections only once.  Refer to the TAFE SA assessment policy for more information <https://www.tafesa.edu.au/apply-enrol/before-starting/student-policies/assessment> |

**MongoDB Assignment (Shopping World)**

**Scenario:**

ShoppingWorld is an online shopping company with head office in South Australia. The company is operating using the MySQL relational database for the last 10 years. The business has extended to capture orders across the world. The company find that there are performance issues when making enquiries for suppliers, customers, and their shopping details. The enquiries are raised by the customers or the sales staff. The ShoppingWorld Database administrator (Joe Black) was working with the relational databases all time and has just heard about some benefits of using a non-relational database. The company is investigating the implementation of noSQL database where the company is gaining many more customers shopping online for their products across the world. The ITWorks got the contract to pilot these changes with the prototype of the database to confirm the requirements before the full implementation as the new system.

The existing ShoppingWorld customer shopping system relational data model is shown in the

ICTPRG544 - ASDS - MongoDB Assignment (ShoppingWorld) - Part 1 Student Files.Zip file.

Joe Black has laid down few database requirements that the system must meet.

As the business expands, there are increasing enquiries about the customers and their shopping details. The system must be flexible to scale up or down to meet the demand of the volume of enquiries and transaction activities. The company is planning to have more local offices distributed in the major cities of different countries to answer the sales and deliveries details. The business unit realized that there are more incoming data are less structured since some transactions may have more data fields which are not defined in advance and some transactions may have less data fields which have no information are provided by users. Joe Black is not sure about which product or type of NoSQL to choose. ShoppingWorld has a sister company in US is using MongoDB and they are happy with the operational performance and technical supports of MongoDB. Joe Black stressed that the database system must be ACID (atomicity, consistency, isolation & durability) compliant and able to achieve the availability & partition tolerance.

You are the ITWorks DBA consultant, the project is divided into four major stages:

* Requirements gathering and product selection
* Database setup, schema validation and data manipulation
* Queries design, indexing, partitioning/sharding and measuring throughput
* REST API coding/prototyping and database documentation

The aim of the tasks/activities in the four stages is to illustrate the features of MongoDB will meet the new database requirements for ShoppingWorld. Please watch the requirements video which covers the stages and tasks included or not included in the development project.

Submission requirements:

* There are four parts in this assignment. Multiple submissions are scheduled throughout the semester. Each part of submission must include:
* A Word document includes your answers for each task. Include commands that you execute, the output of the commands successful or not and/or diagrams if required by the tasks.
* A .js script file includes the mongo commands.
* Screen shots with red circles highlighted the answers.

**Part 1. Requirements gathering, product selection and planning the database development**

In order to give Joe Black more information on noSQL database, complete the following tasks:

Task 1. Watch the ShoppingWorld requirement video which is available in LEARN under the assessment section, list at least 5 business requirements and the two applications being used by the company that will access to the noSQL database.

1. **Business**
   1. Database must be highly available (24/7)
   2. Database must be able to provide a loading balancer, distributing all client requests to available servers (load balanced across different servers) to avoid one server being overloaded
   3. The database must be reliable with web application access, transmission of data must be encrypted, and authenticated
   4. Must maintain existing information (Customer Information, Supplier Information, Product Information, Supplier Product Information, Shopping Cart Information)
   5. Database must be cloud based
2. Applications
   1. Mongo Atlas
   2. MongoDB

**Task 2.** Explain the differences between the relational (SQL database) and non-relational (noSQL) databases in the following table. For each of the comparison, give one difference in term of their technologies being used.

|  |  |  |
| --- | --- | --- |
| Basis of comparison | Relational (SQL) | Non-relational (noSQL) |
| Storage structure | In relational database, the storage structure is in organised in columns and rows | non-relational databases may store data in column-oriented stores, key-value stores, document stores and graph data stores. |
| Query language | The relational database use SQL statements to manipulate the data | The non-relational database has no declarative query language. It means that different language for different data stores. |
| Data schema | The relational databases are very structured. They have defined data schema. (e.g. Use the create table or alter table statements to define table structure) | The non-relational databases are schemeless or they have dynamic schema |
| Suitable for complex queries | The relational databases are suitable for complex queries. They are not good to store the hierarchy or network data structures. | the non-relational databases are not suitable for complex queries. They suit well to store hierarchical data structures |

Section).

**Task 3.** There are four types of noSQL databases (or data stores) in the industry. One of them is the Key-Value data store. Identify the other three types of noSQL data stores. Briefly explain each of the four kinds of the data stores in approximately 50 words each.

1. Column-oriented data store
   1. Column oriented data stores use the columns to store data instead of rows that most SQL databased use, this allows user to directly get data from each column without having non needed data being pulled along with the requested.
2. Key-value data store
   1. Key-value stores uses a key as a unique identifier, each key is paired up with its value, the value can be a range of things such as strings to simple objects to complex data, Key-value have no pre defined structure.
3. Document-based data store
   1. Document data stores are key value pairs stored in a document, these documents are often in a format such as JSON and are much closer to how objects are stored in an applications which means less translation is needed.
4. Graph-based data store.
   1. In a Graph database focuses on the relationship between data. Each piece of data is a node and the connection is a link. Graph databases are grate for social networks to find connections between people or for fraud detection as you can quickly find people who are associated with known fraud activity.

**Task 4**. Identify at least two database products available for each type of noSQL data stores identified in Task 3.

1. Column-oriented data store
   1. BigTable
   2. Redshift
2. Key-value data store
   1. Redis
   2. Riak
3. Document-based data store
   1. MongoDB
   2. Azure Cosmos DB
4. Graph-based data store.
   1. Neo4j
   2. OrientDB

**Task 5.** Watch the ShoppingWorld database requirement video (which is available in LEARN under the Assessment) and complete the following:

1. Determine which database product will best meet the business and application requirements that you have identified in Task 1.

Mongo DB

1. For your selected product, review which product features will meet these business requirements listed. You are expected to write approx. 20 words with the product features how the product meets each of these requirements.

|  |  |
| --- | --- |
|  | Product chosen: xxxxxxx |
| **Business requirements** | **Product features that meet each requirement (Approx. 20 words each)** |
| 1. Cloud based | Mongo Atlas is hosted in the cloud |
| 2. Subscription (scaling) | Supports vertical and horizontal scaling, more suited to horizontal |
| 3. Flexible data structure | Data structures do not need to be enforced in a collection making structures flexible. |
| 4. Support applications with API call | Mongo atlas supports Rest API, able to respond to https request for read and write |
| 5. Distributed & partition | Mongodb provide sharding and partitioning feature for sustaining distributed data across machines, supports deployment with a very large data set, also gives good throughput in operation |
| 6. Optimized using indexing | MongoDB provides indexing to improve query performance, uses B-tree access data. |
| 7. Open source | Mongo DB has community edition which is open source, can move to a subscription model when properly implementing database |
| 8. Enforce integrity (schema validator) | MongoDB does not support direct referential integrity, must use application to enforce integrity, does support schema validation, allows for restrictions in documents. |
| 9. Flexibility (e.g. modelling tools available) | MongoDB has support for modelling tools such as hackolade which can model ERD style, JSON hierarchical structure, documents REST APIs |

**Task 6.** The manager is often receiving complaints from staff about the process of query of the dataset is very slow. Some of the IT support staff mentioned that it could be a database scaling issues. The manager is not sure which kind of scaling is to be adopted. You are asked to:

1. Name two kinds of database scaling methods which are available in the industry.
   1. Vertical Scaling
   2. Horizontal Scaling
2. Describe each of these scaling methods in approx. 30 words.
   1. Vertical
      1. Vertical scaling is adding more resources such as CPU,RAM or storage to a server or database while having the same responsibilities as before, this is best is small to medium sized companies
   2. Horizontal
      1. Horizontal scanning is adding new physical machines to a cluster, this adds more nodes, this means each server/node has reduced responsibilities, this is best fore medium to big companies
3. Recommend which kind of scaling that ShoppingWorld should adopt.
   1. The recommended scaling model for ShoppingWorld is horizontal scaling
4. Give at least two benefits for the kind of scaling that you have chosen in part c.
   1. The benefits for horizontal scaling are
      1. Horizontal scaling can improve read write times by sharing the query to multiple shards in a cluster
      2. Sharding adds the benefit of high availability as if one sharded cluster becomes unavailable, other shards can complete the requests.

**Task 7**. MongoDB ShoppingWorld data structures – i.e. schema diagram.

The existing relational database structure can be found from **ICTPRG544 - ASDS - MongoDB Assignment (ShoppingWorld) - Part 1 Student Files** that included the workbench ERD file and the create table and insert data statements. Even MongoDB has a flexible schema (i.e. schemaless), but initially Joe Black would like to store the existing data first. You are required to design a MongoDB data structures (i.e. presented as schema diagram) that will cover the processing of the existing data according to the business activities needs. See an example of the MongoDB schema diagram in the student file.

The data schema must support the retrieval of data **efficiently** using collections of document structures with the key-value pairs, embedded documents and/or referenced documents in MongoDB.

The design must be able to store all existing data in the relational database, with the trade-off between the consistency and efficiency, the design must also meet the following criteria:

* The data must be able to store both the structured and unstructured data. It can also accommodate additional data fields arise from the business transactions that may not be even expected.
* The data structures must be able to incorporate the data types including int, decimal, …… boo, date, array,………. etc
* The following queries are frequently used which requires the related data are accessed together to achieve the efficiency in data retrieval:
* Display all the customer information such as customer name, email, annual spend together with information of multiple phones, addresses and interests.
* Display the supplier information such as supplier name, email, contact phone together with the information of multiple addresses and their items supplied.
* Display the shopping cart information such as date time, delivery instruction, delivered or not, delivery date, rating from customer etc. together with the shopping cart item information.
* Each collection must have a MongoDB object \_id. However, the documents in the collection also use a surrogated id (e.g. customerID or studentID) that supports the needs of day-to-day business. Note: Do not provide a value of the \_id when inserting document. Let MongoDB to generate the \_id automatically.
* Sub-document in a collection does not need to have an \_id.
* Items supplied by the suppliers must be able to reference to the item details.
* The shopping cart items must also be able to reference to the item details.
* All name of collections and fields must follow with the convention listed in the given naming convention document.

For submission, submit a schema diagram in the word document.

**Task 8.** Submit a plan with the task schedules to complete the database design for Shopping Works. In the plan you must break down the complex tasks into manageable sub-tasks. (Note: You may follow with the major submission dates as the milestone for your major stage of your plan.) Your plan must also include the time on researching, consulting, installing, coding, testing, debugging, and reporting. Use the **ITWorks Database Development Plan Template.docx** which can be downloaded from LEARN. Rename the file without the word “Template”.

**Part 1 Submission:**

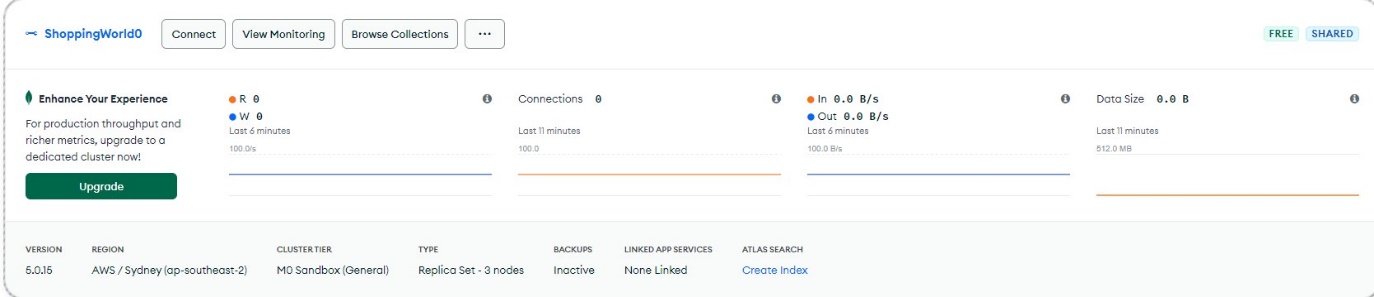
* + Submit a world document includes answers of Task 1 to Task 7. Remember to include the ShoppingWorld MongoDB schema diagram in the Word document.
  + For Task 8, provide the filled ITWorkd Database Development Plan.

**Part 2: Database setup, validation schema and data manipulation**

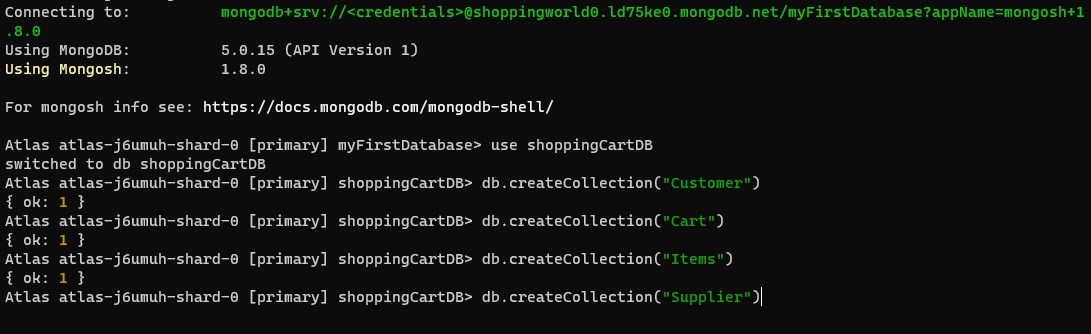
The database administrator, Joe black, would like to see how to store the existing sample data into MongoDB. Completed the following task using Mongo Compass as the client tool.

**Task 1**. Create a MongoDB Atlas account and deploy a free cluster and locate near the ShoppingWorld head office.

Submit a screen shot that the cluster has been created successfully.



**Task 2**. Use the Mongo Compass and the Mongosh shell commands to create the collections under the database name “**shoppingCartDB**”. For submission, provide the commands and screen shot showing all collections have been created. Save the commands used in the file called **part2.js.**



**Task 3.** Use insert statements in Mongo Compass

Joe Black understood that MongoDB has no fixed schema, there is no need to create the collections in advance. You are required to show how data can be inserted into collections without predefining the structures.

a. In the existing relational database, there are three customers in the sample data. Use the same sample data, write 3 separate insert statements to insert one customer document at a time into the customers collection. Each insert statement must insert only one customer with the customer details and the phones, the customer\_addresses and interests as embedded documents. All coding for the queries must comply with the naming conventions described in the **Naming Conventions and Node.js project.docx** document. Provide screen shots to show each insert statement executed successfully.

Your submission must include the following:

* Add the script for the InsertOne statement of the customers collection into the file **part2.js.**
* The screen shots captured for the InsertOne statements executed successfully.

b. There are four items in the existing sample data provided in Part 1 student files, write one insert statement to insert multiple documents (i.e. insert 4 records at once) into the items collections. Provide screen shots to show each insert statement executed successfully.

Your submission must include the following:

* Add the script for the InsertMany statement of the items collection into the file **part2.js**.
* The screen shots captured for the InsertMany statements executed successfully.

c. There are four shopping carts in the existing sample data. Write four separate insert statements to insert into shopping cart collection. Each insert statement must insert one shopping cart at a time each with the shopping cart items as embedded documents. Each shopping cart item is referring to the item ID in the items collection. Provide screen shots to show each insert statement executed successfully.

Your submission must include the following:

* Add the script for the InsertOne statements of each of the carts collection into the file **part2.js**.
* The screen shots captured for each of the Insert statements executed successfully.

d. There are two suppliers in the existing sample data. Write two separate insert statements to insert into suppliers collection. Each insert statement must insert one supplier at a time each with the supplier address as embedded documents and the items as the referenced documents. Each item supplied is referring to the item ID in the items collection. Provide screen shots to show each insert statement executed successfully.

Your submission must include the following:

* Add the script for the InsertOne statements of each of the suppliers collection into the file **part2.js**.
* The screen shots captured for each of the Insert statements executed successfully.

Task 4. Write query statements to show the following outputs:

1. Display all customer with information such as \_id, customer ID, customer name, email, annual spend together with information of multiple phones, addresses and interests. Submit query statement and screen shots with output.
2. Display the supplier “Home Improvement” with information such as supplier name, email, contact phone together with their items supplied. Note: output should not include the \_id, supplier id nor the supplier addresses. Submit query statement and screen shots with output.
3. Display the shopping cart where the shopping cart was created on “2022-02-20 14:30:00” with information such as \_id, cart date time, delivery instruction, delivered or not, delivery date, rating from customer & customer id etc. together with the shopping cart item information. Note: the output should not include the cart ID, the rating from customer information nor the qtyOrdered. Submit query statement and screen shots with output.

Your submission of Task 4 must include the following:

* Add the script of the three query statements into the file **part2.js.**
* The screen shots captured the output of execution of these statements.

**Task 5**. Schema validation.

a. Create another database named “**shoppingCartDB2**”. This time, in the new database, create a schema validation for the **customer** collection which you have design in Part 1 Task 7. The validator must:

1. Define the data fields with correct MongoDB data types that matched with the existing sample data. The type must cover the numeric, string, boolean, complex type like embedded object type. Provide an appropriate description of each field defined.
2. Define all data fields as “required” with the exception of the following fields are not required:
   1. customers.email
   2. customers.firstName & customers.lastName (Note” later we may use company name instead)
   3. carts.ratingFromCustomer
3. The customer\_address.addressType are restricted to these values (i.e. enum data type) e.g. “Postal”, “Delivery”, “Business”.
4. annualSpend has minimum amount $0.00.

Your submission must include the following:

* Add schema validation statements into the file **part2.js**.
* The screen shots captured for successful execution of the validation statements.

1. Write Insert statements that will fail the validation to the following data fields:
   1. customers.email – insert document with no email
   2. customer\_addresses.addressType – insert data value not within the enum() values
   3. annualSpend – insert data with a negative value

Your submission must include the following:

* Add the three Insert statements for testing the validation rules into the file **part2.js**.
* The screen shots capture the output of the statements.

**Task 6**. MongoDB unstructured schema

1. MongoDB can store structured data and unstructured data. To illustrate the benefits of MongoDB’s no fixed schema, write insertOne() statement to test the *additional data* which were *not planned* in the previous Tasks 5.

Insert the following data as document into the customers collection.

customerID: 10

*companyName: “City Shopper Co. Ltd”*

*website: “cityshopper.com.au”*

*registeredDate: “2022-05-01 00:00:00”*

annualSpend: $0.00

phones[ ]:

phoneType: “Business”

phoneNumber: “08-82778888”

customer\_addresses[ ]:

addressType: “Business”

street: “1 First Street”

city: “Adelaide”

postalcode: “5000“

state: "SA"

country: "Australia"

interests[ ]:

interestName: [{"gardening", "tools", null}]

customerID: 11

*companyName: "Country Shopper Co. Ltd"*

*registeredDate: "2022-05-15 00:00:00 "*

*companyType: "Hardware retailer "*

annualSpend: $100.00

phones[ ]:

phoneType: "Business "

phoneNumber: "08-82774444 "

customer\_addresses[ ]:

addressType: "Business"

street: "2 Second Street"

city: "Adelaide"

postalcode: “5000“

state: "SA "

country: "Australia"

interests[ ]:

interestName: [{"power tools", "kitchenware"}]

1. Display all information for the customer “City Shopper Co. Ltd” that you have just inserted in part a.

1. Since the MongoDB do allow to store data without a fixed schema, suggest at least two benefits that using the schemaless database may help the business of ShoppingWorld.

Your submission for Task 6 must include the following:

* Add the insert & query statements into the **part2.js.**
* The screen shots with the output of execution of these statements.

**Task 7**. Update and delete documents

For the following queries, your submission must include the following:

* Add the query statements into the **part2.js.**
* The screen shots with the output of execution of these statements.

1. Update the customer website to “suppershopper.com.au” where the company name is “City Shopper Co. Ltd.”
2. Update the customer registered date from “2022-05-01 00:00:00” to “2022-05-15 17:00:00” for all customers who has postal code = “5000”;
3. Delete the customer where the company name is “City Shopper Co. Ltd.”.
4. Delete all customers where the registered date between “2022-05-01 00:00:00” and “2022-05-16 00:00:00”. Hints: use ISODate() function.

**Part 2 Submission:**

* + Submit a world document includes answers of Task 1 to Task 7. Label each task and sub tasks clearly with the answers, commands and screen shots as required. The screen shots must cover the script and the output of the execution.
  + Submit the script file **part2.js** which include all mongo queries/statements used in each task. Use // or /\* … \*/ as comments to label the task number of your answer.

**Part 3: Queries design, indexing and partitioning/sharding**

Carry out the following tasks. For submission, all queries must be executed and with screen shots of their outputs. Save the mongodb queries/statements into a script file called **part4.js**. Submit a word document including answers to Task 1 to Task 6.

**Task 1.**

1. Joe Black would like to see the retrieval of documents from multiple collections. He would like to see all customer information including the phones, addresses, interests, and the shopping carts with the cart items details for the customer id = 2.
2. Write a query to retrieve the documents from the customers and carts collections.
3. Create indexes to give a better performance for the query.
4. ShoppingWorld has realised the following queries are frequently used many times every day. The query responded very slow. Optimise the following queries to give better performance by creating indexes (i.e. sort key) to the appropriate data fields:

1. Display customer with information such as customer ID, first name, last name or company name and email and the interests where the customer first name equals to “John” and customer last name equals to “Smith”.
2. Write the query.
3. Determine which data field(s) should be indexed so that the query may have better performance.
4. Write the create index statement to optimise the data retrieval.
5. Display all information of the shopping carts together with the shopping cart items where the delivery date is smaller than the system date and the shopping cart has not been delivered.
6. Write the query.
7. Determine which data field(s) should be indexed.
8. Write the create index statement to optimise the data retrieval.

4) Write a command to show the index created in the collection.

1. For the query in Part 2 Task 4 c:

Joe Balck recently found that the following query has a very high demand in retrieving shopping carts by date/time. You are required to determine and to create an additional index to improve the performance of the retrieval.

*“Display the shopping cart where the shopping cart was created on “2022-02-20 14:30:00” with information such as date time, delivery instruction, delivered or not, delivery date, rating from customer etc. together with the shopping cart item information.”*

Note: the output should not include the \_id of the shopping card nor the rating from customer information since that information is not often needed for day-to-day business operation.

1. Write the query first and determine which data fields should be indexed.
2. Write the create index statements to optimise the data retrieval.

Your submission for Task 1 must include the following:

* Add the create index statements into the **part3.js.**
* The screen shots with the output of execution of these statements.

**Task 2**. Time-To-Live for documents

Joe Black has a concern that the logging of the transactions may cause a lot of database storage. He would like to see the logging information will be purged permanently after a certain time. He would like you to illustrate the Time-To-Live feature in MongoDB. Implement a TTL feature with the following steps:

1. Given a sample document of the log file, write a JSON schema with validator for a new collection called “**ttlLogs**” to store the logging of transaction document with information such as object \_id, logDateTime, transactionType & logMessage. Define all data fields are required and with appropriate data types. (i.e. Note: you cannot create the index without the data field defined. Therefore, you must define the JSON schema which first.)

The following is a sample document of a transaction log (given):

{

“\_id” : objectId(“6277d4474d57c31b1ff95f10”),

“loggingDateTime” : “2022-05-01 00:00:00”,

“transactionType” : “insert”,

“logMessage” : “inserted document into customers collection”

}

1. In the collection, create an index on the loggingDateTime with expire after 180 seconds.
2. Write a insertOne() query to insert a transaction log into the ttlLogs collection. Run the insertOne() statement. Hints: the current system datetime is **new Date().** Submit a screen shot with the output showing the document has been inserted. Note: The screen shot must also include the current time of your workstation.
3. Run the find() query after 3 minutes from the completion of Part c. Submit a screen shot to show any document in the ttlLogs collection. Note: The screen shot must include the current time of your workstation.
4. Explain the TTL features with the context of record being deleted. (Approx. 30 words)

Your submission for Task 2 must include the following:

* Add the mongodb statements into the part3.js
* The screen shots with the output of execution of these statements.
* Add the answers of task part e into the word document.

**Task 3**. Multiple users and authentication with permissions

Joe Black would like to know more on the authentication in the MongoDB. He would like you to illustrate the multi-user access in MongoDB with the following features:

a. Configure the user authentication by creating a user called “userAdmin” with the password “Password1” in the “admin” database with the following roles:

userAdminAnyDatabase

readWriteAnyDatabase

dbAdminAnyDatabase

b. Connect as userAdmin. (\*\*Note: you may need to restart the mongod instance o activate the authentication). Create additional user “JimBrown” with password “Password1” to have the ”read” role for the shoppingCartDB.

c. Connect as JimBrown. Test the read role by querying the supplier collection. Display the details of supplier name “Garden Grower”. Show the output message. For submission, the screen shots must show the query executed with the output showing the permission granted or not.

d. Test JimBrown whether he is authenticated to update the supplier collection by changing a supplier contact phone no. to ’08-27788888’. Show the output message.

**Task 4.** Partitioning/sharding and sort key

Joe Black has heard about MongoDB can be scaled vertically and horizontally. He would like to see the illustration on the horizontal scaling with partitioning/sharding features in MongoDB.

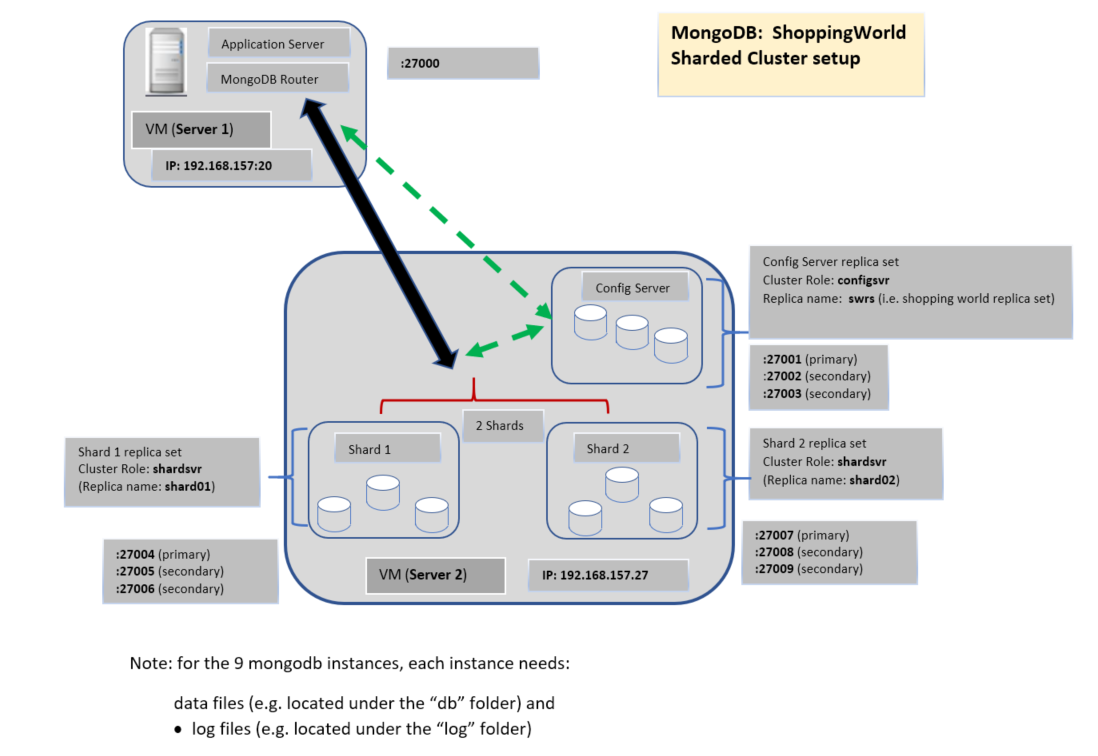
From the past data of the shopping cart, it showed that most customers did not return for shopping (i.e. most customers have one shopping card only). But recently, many customers are return customers and frequently shopping with ShoppingWorld (i.e. most customers have many shopping carts). MongoDB provides the sharding (horizontal scaling) feature which is to distribute data across multiple machines. These features support the deployment of large data sets with higher throughout which may help the performance of the queries.

For showing the features of partitioning to Joe Black, use the MongoDB installed on your local computer to complete this task. The reason is MongoDB Atlas webservice requires to upgrade from the free subscription before you can create replica sets for partitioning data using the partition key.

In general, you are provided with the VMWare Windows Servers with the MongoDB installed to show these features instead. Once when you have started the VMWare server, you need to make sure that the Windows domain controller is able to communicate with the member server. (Note: Instruction are given in the class exercise.)

Run the insert queries which you have saved in the previous tasks **(Part 2 Task 3 a, b, c & d)** to insert the document in the collections for the database. Do not worry about the validator.

You are required to configure the MongoDB instances in the VM Windows servers (Server1 & Server2) as shown in the diagram below so that the MongoDB router would be able to route transactions into the two shards. Without using 9 separate physical computers, use the following port numbers to simulate the mongod instances.



1. Setup a config server, two shards and a router
2. Given the VMWare Windows servers, MongoDB replica setup script and config files in the MongoDB Assignment (ShoppingWorld) - Part 3 Student Files, configure the VMWare servers so that they are communicating using the fixed IP address as shown in the diagram above. Submit screen shots with the two server IP Addresses. Note: you must use the given IP addresses above since servers in practice must have fixed IP addresses..
3. Setup the sharded cluster in the following procedures:

* In the VMWare servers, both MongoDB has been installed locally in Server1 and Server2.
* With the given Server 2 scripts files, copy the MyCluster folder into the C: drive of the VM Sever2.***Note: You must change all folder names used in the scripts, config file names used, IP addresses used to line up with the diagram above.***
* With the given Server 1 scripts files, copy the MyCluster folder into the C: driver of the VM Server1. ***Note: You must change all folder names used in the scripts, config file names used, IP addresses used to line up with the diagram above****.*
* In Server2, run Server2\_Script1 script in the MyCluster folder.
* In Server2, run Server2\_Script1 script in the MyCluster folder.
* It sets up the folder structures for data and log files for the config server and shard01 and shard02 server instances.
* It also configures the replica set for the config server and shard01 and shard02.
* It initiates the config server replica set with one primary and two secondary members
* In Server1, run the Server1\_Script1 script.
* It configures the mongo router (mongos)
* In Sever 2, run Server2\_Script2 script in the MyCluster folder.
* It initiates the shard01 and shard 02 replica set. Each with one primary and two secondary members.
* Run commands to connect to the router port no. (> mongo --port xxxxxxxx) and show the replica status. Capture screen shots of the replica sets status.
* In Server1, run the Server1\_Script2 & Server1\_Start scripts.
* It connects the mongos client and add the shards to the mongos router.
* Run mongos commands to connect to the router port no (> **mongos** --port xxxxxxxx) and show the sharding status.
* Enable the authentication of the mongoDB in the Server1 and Server2 by creating a super user.
* use admin
* create a super user named “myAdmin” with password “password”
* allocate the role “userAdminAnyDatabase” db: “admin” and also with the role of readWriteAnyDatabase
* Write commands to enable sharding for the databases. (\*\*note; need to connect to the port no of the router first > **mongo --port 27000**).
* Show the shard status. Capture screen shot showing the sharding of the shoppingCartDB is enabled.

For submission:

* Screen shot of the Server2 with the 9 mongod instances started.
* Screen shots of the Server 2 with one of the replicas set with 1 primary and at least one of the secondary members
* Screen shot of Server2 showing the sharding of the shoppingCartDB is enabled.

1. ShoppingWorld realised that customers always enquire their “shopping cart” information by providing their customer id when making enquiries. Recommend a shard key for the carts collection that is suitable for sharding. Explain why you choose that field(s) as the shard key. (approx. 20 words)
2. In Server 1, run commands to do the following:

* Create a hashed index for the field to be act as the partition key (i.e. sort key) of the collection.
* Define a shard collection with the partition key.
* Capture a screen shot to show the shard collection distribution.
* Enable the balancer for the collections.

For submission, give the commands being used and provide one screen shot showing no distribution (i.e. 0% ) of records between the two shards yet.

1. In Server1, use the MongoDB Compass in your **windows host machine** to connect to the router.

( e.g. mongodb://router ip: router port).

Use a for loop to insert 500 new documents into the carts collection where the cartID and customerID as the following:

i.e. cartID as cart1, cart2, cart3, …….cart500

customerID as customer1, customer2, customer3 …. , customer500

The document structure can be simplified ads the following:

{

cartID: “cartID” + i,

cartdateTime: new Date(),

customerID: “customerID” + i,

cart\_Items: [ { itemID: 2,

qtyOrdered: 2}]}

1. Since there are 500 records inserted through the router, run a command to show the distribution of the carts documents are partitioned between the two shards. Submit a screen shot of the command used and the output with the % distributed between the two shards.

**Task 5.** Joe Black is also concerned about whether the MongoDB has any utility tools to measure the performance of the database. You are required to analyse the throughput of the transactions with an instance of the connection. Note: this task can be completed in MongoDB Atlas or in local MongoDB.

1. Run a system command to start the MongoTop for connecting to the mongoDB with your administration credential. For simplicity, specify 1 second as the amount of time as the instance for reading and writing by the mongod. Provide the screen shot of mongotop for the statistical reading without any activity in the database yet.
2. Use another connection to your database with the Mongo Atlas. Write a loop to insert one 100,000 documents into a collection called “**perftrans**” in the shoppingCartDB with the following details.

The inserted document should have the following information:

{ “transactionID”: “writing-“ + i,

“transactionName: “bulk write”,

“sysDateTime”: new Date(),

“tranaction amount”: Math.random() \* i}

1. Run the mongotop command, measure the output activity that happened in the “**perftrans**” collection in the database. Submit a screen shot showing the amount of time that the mongod spent in performing the write operations on the collection.
2. Use a mongo shell command to show the amount of time that needs to write the 100,000 document. Note: the unit of the reading is in micro second and you may need to stop the mango service to reset the statistics.

For submission, you must provide the mongo command being used and the screen shot to show the no. of record inserted and the duration of time used.

1. With the above information, calculate how long (in milli second) does it take to insert 4000 documents into the database. For submission, you must show your calculation.
2. Use the same mongo command to measure the read statistic. Use one findMany() command statement to find the following the following records.

transactionID: “writing-80000” or transactionID: “writing-20000”

For submission, show a screen shot with the amount of time that the query will take to locate the two records in the database.

1. Set the profile level for the mongoDB to record all queries that takes longer than two seconds to run. For submission, show the command used.

**Task 6**. Monitor progress of plan and need changes in the customer information requirements.

Joe Black recognised that the customers collection with the phone type and phone number is not practical that they are the required field. He changed his mind that he would like the customers validator to be changed for the phone type and phone number no longer be required.

He also like to have the user password be encrypted when sending request from the client to the server in the user authentication process. All changes must be tested. He would like to see this feature be prototyped by xx/xx/xxxx (e.g. 31/12/2022).

You are required to record these two changes of requirements in the **Development Plan document** under the section of Changes suggested from clients.

|  |  |  |  |
| --- | --- | --- | --- |
| **Date requested the changes** | **Changes to be made** | **Name of the client representative** | **Actions to be taken** |
|  |  |  |  |
|  |  |  |  |

For submission, you must submit the updated ITWorks Database **Development Plan document** to LEARN.

**Part 3 Submission:**

* + Submit a world document includes answers of Task 1 to Task 6. Label each task and sub tasks clearly with the answers, commands and screen shots as required. The screen shots must cover the script and the output of the execution.
  + Submit the script file **part3.js** which include all mongo queries/statements used in each task. Use // or /\* … \*/ as comments to label the task number of your answer.
  + Submit ITWorks Database **Development Plan document**.

**Part 4. REST APIs & Database Documentation**

**Task 1.** Create and call REST APIs

Joe Black has heard about REST API is an architectural style that supports HTTP requests from applications to access data in MongoDB. He would like you to illustrate how the API layer which provides the decoupling of client applications from server through this uniform interface. The illustration must use a client application with HTTP access the URI resources to perform the database actions on JSON data with MongoDB. You are required to build the REST APIs (create, read, update, and delete i.e. CRUD) using Express.js, Node.js, nodemon, and Mongoose. The illustration must also include codes with error trappings to show whether the database operations are executed successfully or not. To complete these tasks, no need to write a full web application, for the purpose of illustrating the REST API calls, use Postman as the development tool without writing the html pages for simplifying the build, test and modify the APIs.

Node.js

Express

MongoDB Driver

APIs

APIs



Postman

MongoDB

The REST API architectural style with MongoDB

1. Login the MongoDB Atlas cluster in the cloud, obtain a connection string for application and the Node.js driver. Show the connection parameters and the code of Node.js driver (in full).

(i.e. no need to show your password.)

1. Create a directory called “shoppingWorld” in the c: drive. Use Visual Studio Code to create the package.json file and run the **npm init** to initialise the project such as name of the package, version, author and description and the dependent packages used etc. Enter/Edit this information with the context of ShoppingWorld database development. You must include the following information:
   * Name of the package
   * Version of the package
   * Description of the package
   * Author name
   * License information
   * List of the dependency packages and their version used.

For submission: In the Visual Studio codes include the **package.json** file which specified the above details.

Note: There are many packages in the market that you may use to develop the REST API, however you may need to install the mongoose library to translate the objects between the code and MongoDB. Note: whichever packages that you have used for the development, you must include those packages in your submission of the codes.

1. Use Node.js to write a REST API route for the purpose of performing operations to create, read, update, delete **(CRUD)** customer document into the customers collection in MongoDB at the end. Test the execution by sending the requests through the Postman as the front-end application. For this task, create a file called **index.js** under the node\_modules to include the following features:
2. At the top of the file, define constants for the Express and Mongoose libraries.
3. Provide a connection function to connect the MongoDB Atlas database using Mongoose. Use the .env file for the database connection string.
4. Define the end point for the routes.

1. Define the listening port for the Express server.

1. Write a REST API route to perform insert a customer document into the customers collection. To simplify the illustration to Joe Black, no need to include the sub-documents in this task. The customer information should have customerID, firstName, LastName, email and annualSpend. Test the execution by sending a post request through Postman with your example data. You are required to insert two customers into the customers collection. For submission, other than the project code, show the screen shots of the Postman request and response from the API call.
2. Write a REST API route to perform read operation to read all customer documents from the customer collections. Test the execution by sending a get request through Postman to retrieve the customers that have been inserted in **Part** **d**. For submission, show the screen shots of the Postman request and response from the API call.
3. Write a REST API route to perform a read operation to find one customer document from the customer collections. Test the execution by sending a get request through Postman using one of the customer object\_id that has been inserted in **Part** **d** as the search criteria. For submission, show the screen shots of the Postman request and response from the API call.
4. Write a REST API route to perform an update operation for a customer document in the customer collections. Test the execution by sending a patch request through Postman for the customer object \_id that has been inserted in **Part** **d** with a new information firstName, lastName and annualSpend. For submission, show the screen shots of the Postman request and response from the API call.
5. Write a REST API route to perform a delete operation to delete a customer document in the customer collections. Test the execution by sending a delete request through Postman to delete the customer document that you have created in **Part d** above. For submission, show the screen shots of the Postman request and response from the API call.
6. This task is to use the debugger in the Visual Studio Code. While you are developing the REST API route for **Task d** (i.e. insert customer document into the customers collection) with the .js code in Visual Studio Code, add or modify your code with the following error trapping if you have not got one:

e.g.

try {

Add a break point at this statement.

const dataToSave = await customers.save();

res.status(200).json(dataToSave);

}

catch(error) {

var myMessage = error.message;

res.status(400).json({message: myMessage});

}

* Add a break point at the first try statement.
* Run npm start with debug mode.

You are required to provide the screen shots for the following tasks:

* Capture the first screen shot that shows the program execution stops at the break point.
* When the program stops at the break point, use Postman to post a customer with your own data so that it failed to insert the customer due to data error.
* Use the debugger to Step over to those lines

**var myMessage = error.message;**

res.status(400).json({message: myMessage});

* Capture the 2nd screen shot that shows the value of the myMessage local variable after the line has been executed.

For submission:

Submit the two screen shots in the word document. Label the question no. clearly.

**Submission for Task 1 sub-tasks (a) to (i):**

* Submit the Visual Studio Codes project including all .js files such as package.js, index.js, .env, model.js and routes.js files.
* Submit screen shots for all REST API operations tested with the Postman requests for sub-tasks (d) ~ (i).

**Task 2.** Database Trigger

The Client Database Administrator (Joe Black) has concern with the change stream handled by the client application. The consistency and integrity of data relies on the client application are always running so that it would not miss any of the change events. MongoDB have the feature of server-side database triggers. You are required to create a database trigger and show the notification of the event has happened.

a. Use MongoDB Atlas web UI, create an update database trigger called “customerChangeTrigger” **change** event to respond when a document customer is changed (e.g. the value of the annualBalance is changed).

For submission:

* + Submit a screen shot with the name of the update event trigger has been created.
  + Submit a screen shot of the notification function to display the update event details through the RealmApp.

b. Write an updateOne statement to test the change event trigger. The update statement can be run either through the REST API call or any client tools like MongoDB Compass or even command line. The update statement must change the customer document for customerID = 2 with the annualSpend of $2000.00. Display the notification of the change event has happened through the Triggers\_RealmApp.

**Task 3.** MongoDB data Encryption

MongoDB supports three types of data encryptions: Encryption in Transit/Transport (TLS), Client-Side Field Level Encryption, and encryption at REST. These features need to be turned on and these will increase the daily cluster pricing. Describe each of the following encryption options: (Approx. 40 words each).

1. Encryption in Transit/Transport layer security (TLS)
2. Client-Side Field Level Encryption
3. Encryption at REST

**Task 4.** Document the database

Document the database by completing the **IT Works MongoDB Database Documentation Template.docx** which can be found in the Assignment ASDS Part 4 student file. All sections must be filled. The answers must be in the context of the Shopping World. (Note: rename the file name **without** the word “**Template**”.

For submission:

Submit the completed **IT Works MongoDB database documentation.docx** document.

**Part 4 Submission:**

* + A world document includes answers of Task 1 to Task 3. Label each task and sub tasks clearly with the answers, commands and screen shots as required. The screen shots must cover the script and the output of the execution.
  + For Task 1, also submit the Visual Studio Codes project including all .js files such as package.js, index.js, .env, model.js and routes.js files. Put these files under the **Part4\_Task1** folder.
  + The completed IT Works MongoDB Database Document.docx.
  + Zip the above files/folders into a single zip file for submission to LEARN.