Data Warehouse with MongoDB

Migration from RDBMS to MongoDB

Description: Research, Test, Document migrating data from RDBMS to MongoDB using ETL (Extract, Transform, and Load) tool

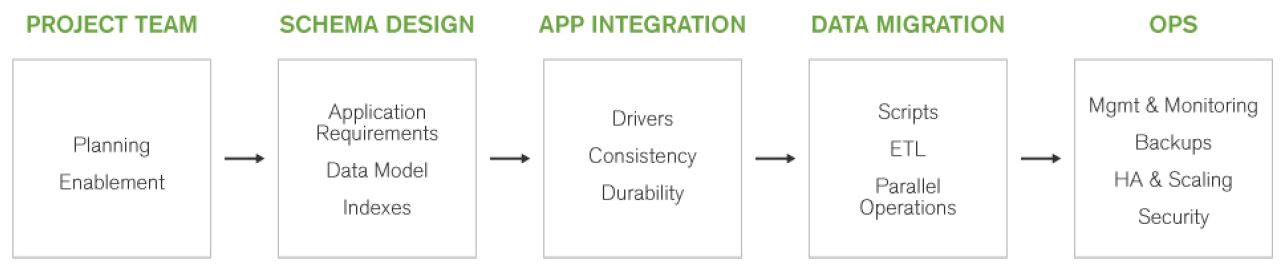
Members: Muthu, Avinash, Olga, Jongkuk​

Date: 2018.09 ~ 2018.10 (10 Weeks)

### Revision History

| **Date** | **Comment** | **Writer** | **Version** |
| --- | --- | --- | --- |
| 2018-08-30 | Trial Version |  | 0.1 |

## RoadMap



See More: [RDBMS to MongoDB Migration Guide](https://github.com/Gs1TestTeam/MongoDB_Task/blob/master/doc/RDBMS_to_MongoDB_Migration_Guide.pdf)

## Planning (Week1)

* Define the requirements (output: business and technical requirements)
* Research MongoDB essential features (output: MongoDB essential features)
* Sign up for Cloud Servers and Install GUIs for RDB, MongoDB (output: account and user guideline)
* Researhing licence, server required resource (output: licence and server required resource)

## Schema Design (Week2)

* Create Schema Desgin Mapping Chart (From RDB to MongoDB) (output: Schema Desgin Mapping Chart.xls)
* Write MongoDB schema using JSON format (output: JSON format for GS1's collections)
* Create a tool to create the sample data to RDB (output: Data generate application)
* Reseaching ETL tools to extract, transform, and load data from RDB to MongoDB (output: ETL tools and test result)

## App Integration (Supply) (Week3)

* Create Application Changes Impact Analysis chart (to identify the scope for changing GS1's application) (output: Application Changes Impact Analysis chart.xls)
* Write guidelines to change application for developers (output: developer guideline)
* Write the scenarios to deploy GS1's application (output: application deploy scenarios)

## Data Migration (Week4)

* Create collections to MongoDB (in Could service or local or team server) (output: collections on MongoDB)
* Migrate data from RDB to MongoDB (measuring time, resource: hard disk, memory, CPU etc) (output: migration result report)
* Deploy application (testing functionalities)

## Operating (Week5, 6)

* Research how to repond to the structure changes on MongoDB (output: how to repond to the structure changes)
* Research and test Replication and Sharding (ouput: Introducign Replication and Sharding)
* Write the strategies to introduce High Availability and Scalability for GS1 Canada's systems (output: Introducing High Availability and Scalability for GS1 Canada's systems)

## Documenting (Week7)

* Organize and write documentations with the outputs of each step (output: GS1 Canada Data Migration Report with MongoDB.doc)

MongoDB Essentials

### Terminology Translation

| **RDBMS** | **MongoDB** |
| --- | --- |
| Database | Database |
| Table | Collection |
| Row | Document |
| Index | Index |
| JOIN | Embedded document, document references or $lookup to combine data from different collections |

Mongo DB supports a hundred levels of depth for your documents​

See More: [Terminology and Concepts](https://www.mongodb.com/compare/mongodb-mysql)

See More: [SQL to MongoDB Mapping Chart](https://docs.mongodb.com/manual/reference/sql-comparison/)

### [Data Types](https://www.tutorialspoint.com/mongodb/mongodb_datatype.htm)

* **String**− This is the most commonly used datatype to store the data. String in MongoDB must be UTF-8 valid.
* **Integer**− This type is used to store a numerical value. Integer can be 32 bit or 64 bit depending upon your server.
* **Boolean**− This type is used to store a boolean (true/ false) value.
* **Double**− This type is used to store floating point values.
* **Min/ Max keys** − This type is used to compare a value against the lowest and highest BSON elements.
* **Arrays**− This type is used to store arrays or list or multiple values into one key.
* **Timestamp**− ctimestamp. This can be handy for recording when a document has been modified or added.
* **Object**− This datatype is used for embedded documents.
* **Null**− This type is used to store a Null value.
* **Symbol**− This datatype is used identically to a string; however, it's generally reserved for languages that use a specific symbol type.
* **Date** − This datatype is used to store the current date or time in UNIX time format. You can specify your own date time by creating object of Date and passing day, month, year into it.
* **Object ID** − This datatype is used to store the document’s ID.
* **Binary data** − This datatype is used to store binary data.
* **Code** − This datatype is used to store JavaScript code into the document.
* **Regular expression** − This datatype is used to store regular expression.

### Query Language



See More: [Query Language](https://www.mongodb.com/compare/mongodb-mysql)

See More: [SQL to MongoDB Mapping Chart](https://docs.mongodb.com/manual/reference/sql-comparison/)

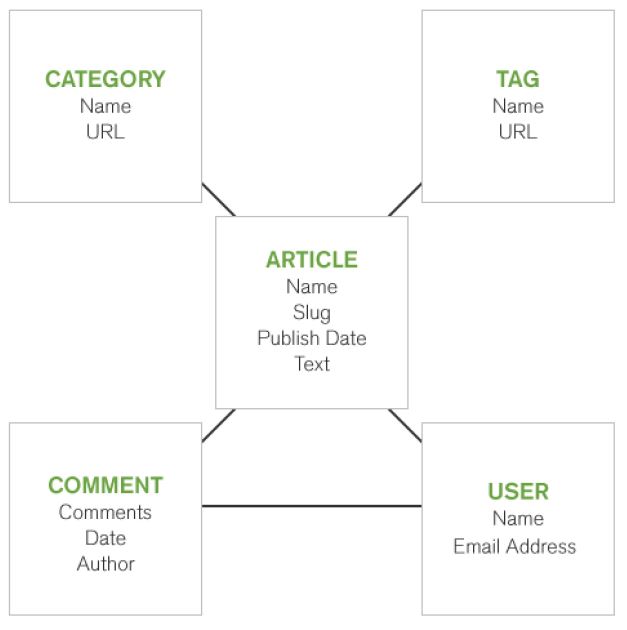
### Why MongoDB is effective

**RDBMS Data**

* Separate tables for different data
* Tied together by primary key
* Complex representation

**SQL Query for data from RDBMS**

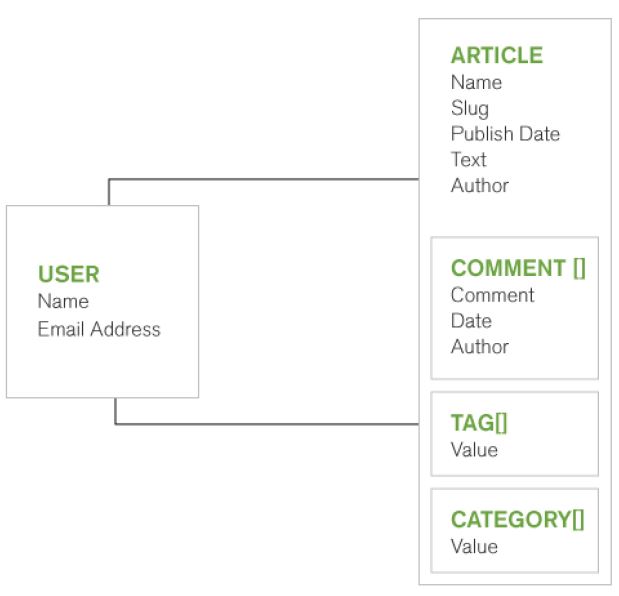
* Data is discrete in several tables​
* Complex to select​





**Select data from MongoDB**

* Data is in only an object​
* Easy to handle​



See More: [Query Language](https://www.lynda.com/Moodle-tutorials/Why-Mongo/573253/611681-4.html?autoplay=true)

### Aggregation

* Similar to GROUP BY function in SQL​



See More: [SQL to Aggregation Mapping Chart](https://docs.mongodb.com/manual/reference/sql-aggregation-comparison/index.html)

### Performance

**Index**

* 64 indexes
* Single field
* Compound
* Unique

Mongo does support up to 64 indexes per collection​

**Sharding**

* Partition data onto different machines
* Provides scalability via software
* Autosharding supported by Mongo
* Challenging to set up

Using sharding means that you can scale your application across several smaller systems rather than investing in a larger, more expensive solution.​

While sharding is aimed at scalability, it's arguably even more important to ensure the uptime of your system, and for this, Mongo provides replication.​

**Replication**

* Reliability
* Maximizes uptime
* Replica set
* Automation failover

With MongoDB, you set up a replica set for your database with a primary server and some number of secondary servers, which keep copies of the primary servers data. If your primary server fails, one of the secondary servers will be promoted to the primary slot, and the database will continue to act as usual. When the original primary comes back online, it will be added back to the system as a new secondary server in the system.

### Debugging​​

db.number.explain("executionStates")​



## Stored Procedures

### Create:

function addNumbers( x , y ) {

return x + y;

}

### Save:

> db.system.js.save({\_id:"addNumbers", value:function(x, y){ return x + y; }});

### Utilize:

> db.system.js.find()

{ "\_id" : "addNumbers", "value" : function cf\_\_3\_\_f\_(x, y) {

return x + y;

} }

### Result:

> db.eval('addNumbers(17, 25)');

42

See More: [Getting Started With Stored Procedures in MongoDB](http://pointbeing.net/weblog/2010/08/getting-started-with-stored-procedures-in-mongodb.html)

There are two ways that you can perform this process.

* First way.
* Second way

Heading 2

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin suscipit rhoncus cursus.

| Existing Role | Description | Responsibilities |
| --- | --- | --- |
| Catalogue Captain | One person.  Table text.   * Table bullet. * Table bullet. | Assigns an administrator to manage the service.   1. Table number. 2. Table number.   Table link. |

Heading 3

Lorem ipsum dolor sit amet, consectetur adipiscing elit:

1. Proin suscipit rhoncus cursus.
2. Nulla quis turpis sit amet nisl sagittis laoreet.

Heading 1(\_)

Heading 4

Integer ipsum lectus, auctor at gravida eget, fermentum a magna.

Heading 5

Nulla blandit blandit interdum.