**Mongo DB Essentials**

MongoDB stores its information in documents rather than rows. MongoDB’s document format is based on JSON, a popular scheme for storing arbitary data structures.

Internally, MongoDB stores documents in a format called Binary JSON, or **BSON**. BSON has a similar structure but is intended for storing many documents. When you query MongoDB and get results back, these will be translated into an easy-to-read data structure. The MongoDB shell uses JavaScript and gets documents in JSON.

Relational databases have tables, MongoDB has collections, and In other words, MySQL keeps its data in tables of rows, while MongoDB keeps its data in collections of documents, which you can think of as a group of documents.

The data in a collection is stored to disk, and most queries require you to specify which collection you’d like to target.

**Indexes**

Indexes in MongoDB are implemented as a B-tree data structure. B-tree indexes, also used in many relational databases, are optimized for a variety of queries, including range scans and queries with sort clauses.

Most databases give each document or row a primary key, a unique identifier for that datum. Like RDBMS allows secondary indexex, MongoDB allows users to optimize for a wide variety of queries.

With MongoDB, you can create up to 64 indexes per collection. These indexes are like ascending, descending, unique, compound-key, hashed, text and even geospatial indexes are supported.

$/>db.**collection**.dropIndex(“index\_name”);

Primary key Id index name is \_id \_ that we can’t drop

$/>db.**collection**.createIndex(‘property’:1); 1 means ascending order -1 descending order

$/>db.**collection**.getIndexes(); return all indexes on that collection.

**NOTE: replace collection with actual collection name.**

http://tech.tulentsev.com/2014/02/limitations-of-mongodb/

**Replication**

MongoDB provides database replication via a topology know as a replica set. Replica sets distribute data across two or more machines for redundancy and automate failover in the event of server and network outages.

Replication used to scale database reads.

The MongoDB command shell is a JavaScript based tool for administering the database and manipulating data

An **ObjectId** is a 12-byte BSON type having the following structure:

* The first 4 bytes representing the **seconds** since the unix epoch
* The next 3 bytes are the **machine identifier**
* The next 2 bytes consists of **process id**
* The last 3 bytes are a **random counter value**

A **capped** collection is a collection with a fixed size, similar to a circular list. When a capped collection is full the data gets overwritten. Data cannot be deleted from a capped collection. For example, create a capped collection catalog with auto indexing and 64 KB size with maximum number of documents as 1000. Before creating the capped catalog collection, drop the collection if created previously using the db.catalog.drop () command.

Creating Collection and Inserting document from Mongo Shell

Creating Collection

**$/> db.createCollection("catalog",{capped:true,autoIndexId:true,size:64\*1024,max:1000});**

Switching collection

**$/> use catalog;**

Creating and inserting document

**$/> doc1={"catalogId":"catalog1","jounrnal":"Oracle Magzine"}**

**$/> db.catalog.insert(doc1);**

Displaying all the documents

**$/>db.catalog.find()** gives all the documents

Bule insert is available

**$/>db.catalog.insert([doc1,doc1])**;