<Course Name>   <Lesson Name>

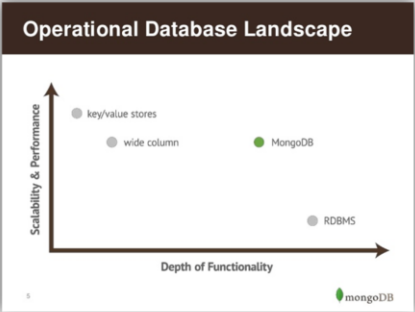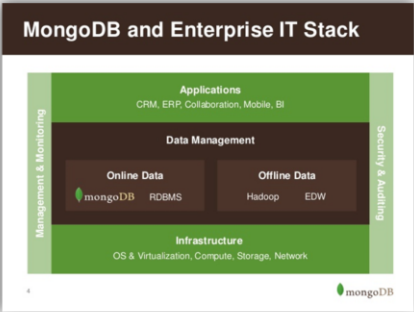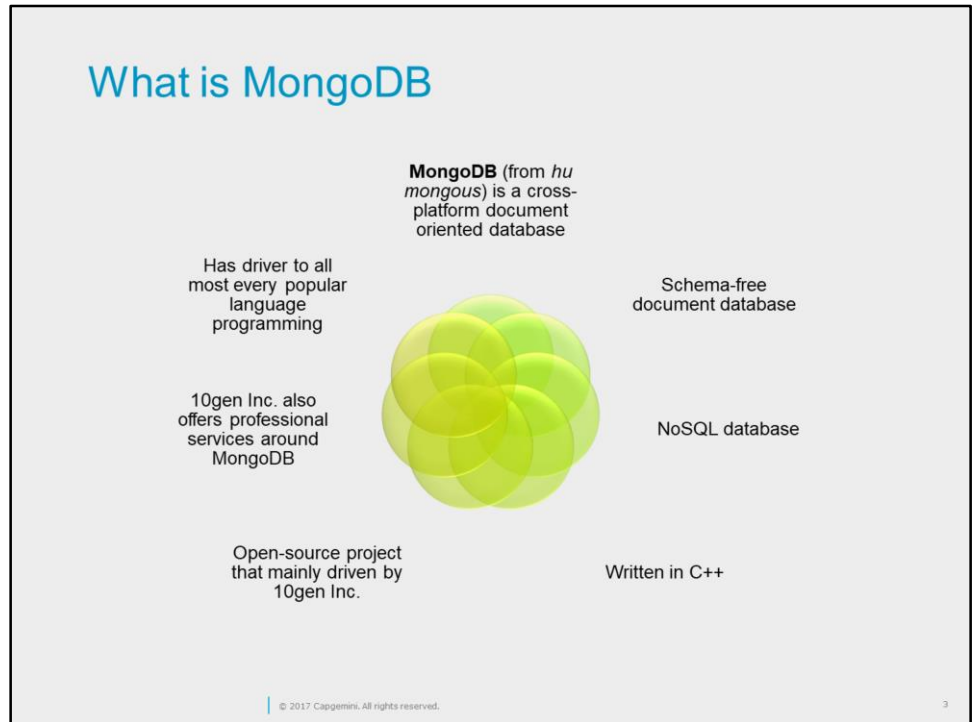MongoDB Introduction

Capgemini

<Course Name>   <Lesson Name>

MongoDB (from *humongous*) is a cross-platform document oriented database . Classified as a NoSQL database, MongoDB eschews the traditional table-based relational database structure in favor of JSON -like documents with
dynamic schemas (MongoDB calls the format BSON), making the integration of data in certain types of applications easier and faster. Released under a combination of the GNU Affero General Public License and the Apache License, MongoDB is free and open-source software

MongoDB stores data in the form of documents, which are JSON-like field and value pairs.
Documents are analogous to structures in programming languages that associate keys with values (e.g. dictionaries, hashes,
maps, and associative arrays).
 Formally, MongoDB documents are BSON documents. BSON is a binary representation of JSON with additional type information. In the documents, the value of a field can be any of the BSON data types, including other documents, arrays, and arrays of documents

<Course Name> <Lesson Name>

Why MongoDB-MongoDB is:

| General Purpose	Rich data model	Full featured indexes	Sophisticated query language
Easy to Use	Easy mapping to object oriented code	Native language drivers in all popular languages	Simple to setup and manage
Fast & Scalable	Operates at in-memory speed wherever possible	Auto-shading built in	Dynamically add / remove capacity with no downtime

4

<Course Name>   <Lesson Name>

Why MongoDB-MongoDB is Easy to Use
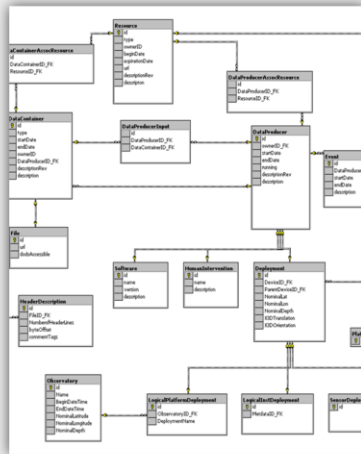
```
{
  title: 'MongoDB',
  contributors: [
    { name: 'Eliot Horowitz',
      email: 'eliot@10gen.com'
},
    { name: 'Dwight Merriman',
      email:
'dwight@10gen.com' }
  ],
  model: {
    relational: false,
    awesome: true
  }
}
```

<Course Name>   <Lesson Name>

Why MongoDB-   Schema Free

MongoDB does not need any pre-defined data schema.

Every document could have different data!

{name: "will",
  eyes: "blue",
  birthplace:
"NY",
  aliases: ["bill",
"la ciacco"],
  loc: [32.7,
63.4],
  boss: "ben"}

{name: "jeff",
  eyes: "blue",
  loc: [40.7, 73.4],
  boss: "ben"}

{name: "brendan",
  aliases: ["el
diablo"]}

{name: "ben",
  hat: "yes"}

{name: "matt",
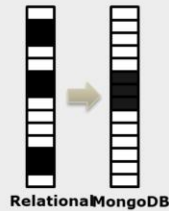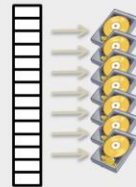 pizza: "DiGiorno",
  height: 72,
  loc: [44.6, 71.3]}

mongoDB

6

Applications enforce the data "schema" and integrity, much like MUMPS does in VistA

Why MongoDB-MongoDB is Fast and Scalable

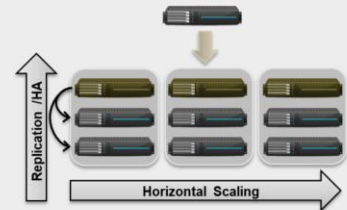**Better data locality**

In-Memory Caching

Distributed Architecture

Relational MongoDB

Replication /HA

Horizontal Scaling

7

Why MongoDB-History

Jan 2015

3.0

2.6.7

2009

C++

10gen

HUMONGOUS

2007

8

<Course Name>   <Lesson Name>

Why MongoDB-Features of MongoDB

Document Oriented Database

Adhoc queries

Indexing

High Performance

High Availability

Sharding

Easy Scalability

File Storage

Rich Query Language

Load Balancing

Replication

9

Document-oriented
Documents (objects) map nicely to programming language data types
Embedded documents and arrays reduce need for joins
Dynamically-typed (schemaless) for easy schema evolution
No joins and **no multi-document transactions for high performance and easy scalability**
High performance
No joins and embedding makes reads and writes fast
Indexes including indexing of keys from embedded documents and arrays
Optional streaming writes (no acknowledgements)
High availability
Replicated servers with automatic master failover
Easy scalability
Automatic sharding (auto-partitioning of data across servers)
Reads and writes are distributed over shards
No joins or multi-document transactions make distributed queries easy and fast
Eventually-consistent reads can be distributed over replicated servers
Rich query language

Document-oriented
Documents (objects) map nicely to programming language data types
Embedded documents and arrays reduce need for joins
Dynamically-typed (schema less) for easy schema evolution
No joins and **no multi-document transactions for high performance and easy scalability**
High performance
No joins and embedding makes reads and writes fast
Indexes including indexing of keys from embedded documents and arrays
Optional streaming writes (no acknowledgements)
High availability
Replicated servers with automatic master failover
Easy scalability
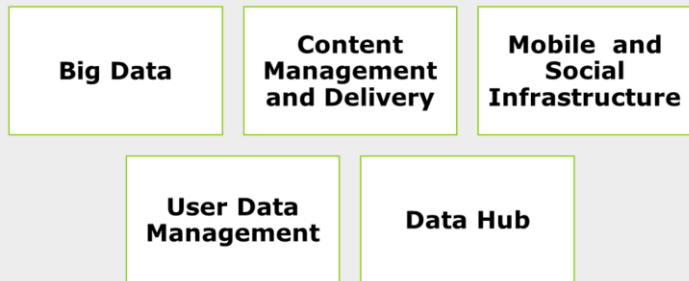Automatic Sharding (auto-partitioning of data across servers)
Reads and writes are distributed over shards
No joins or multi-document transactions make distributed queries easy and fast
Eventually-consistent reads can be distributed over replicated servers
Rich query language

<Course Name>   <Lesson Name>

Why MongoDB-MongoDB to SQL Terminology

MongoDB	SQL
database	database
collection	table
document	record (row)
field	column
linking/embedded documents	join
primary key (_id field)	primary key (user designated)
index	index

11

<Course Name>   <Lesson Name>

Why MongoDB-Important Terminology

Database	• Database is a physical container for collections. Each database gets its own set of files on the file system. A single MongoDB server typically has multiple databases.
Collection	• Collection is a group of MongoDB documents. It is the equivalent of an RDBMS table. A collection exists within a single database. Collections do not enforce a schema. Documents within a collection can have different fields. Typically, all documents in a collection are of similar or related purpose.
Document	• A document is a set of key-value pairs. Documents have dynamic schema. Dynamic schema means that documents in the same collection do not need to have the same set of fields or structure, and common fields in a collection's documents may hold different types of data.

12

<Course Name> <Lesson Name>

Why MongoDB-Data Model

Document based (max 16 MB).

Documents are in BSON formats consisting of field / value pairs.

Each document stored in a collection.

Schema less.

13

<Course Name>   <Lesson Name>

Why MongoDB-MongoDB Data Model

15

<Course Name>   <Lesson Name>

Why MongoDB-The Basics of MongoDB

A MongoDB instance may have one or more Databases.

A database may have one or more Collections.

A collection may have zero or more Documents.

A document may have one or more Fields.

MongoDB indexes function much like their RDBMS counterparts.

16

<Course Name>   <Lesson Name>

Why MongoDB-Database Server and Client

<Course Name>   <Lesson Name>

When not to use

- Not suitable for Highly transactional systems .
- Not suitable for where the data model is designed up front.
- Not suitable for Tightly coupled systems

Capgemini

Documents, Collections and Databases- MongoDB 's Data Model

A Database has "Collections"

Collections have "Documents"

Documents have "Fields"

Fields are key = value pairs

19

<Course Name>   <Lesson Name>

Setting up MongoDB-Getting Started

Install mongodb on windows from the link
given below:

http://www.mongodb.org/downloads

Make sure you get correct version of
MongoDB depending upon your windows
version.

MongoDB for Windows 64-bit: This build
type of MongoDB runs on any 64-bit version
of Windows latest than Windows XP, involve
Windows Server 2008 R2 and Windows 7 64-
bit.

20

<Course Name>   <Lesson Name>

Setting up MongoDB-
What MongoDB does, How it works

MongoDB is a server process that runs on Windows/Linux , Os X.

It can be run both as a 32 or 64-bit application. We recommend running in 64-bit mode, since Mongo is limited to a total data size of about 2GB for all databases in 32-bit mode.

Clients connect to the MongoDB process, optionally authenticate themselves if security is turned on, and perform a sequence of actions, such as inserts, queries and updates.

21

<Course Name>   <Lesson Name>

Starting the MongoDB Server

Create a directory where MongoDB stores all its data.

The MongoDB default data directory path is \data\db.

Create the data folder in D:\

Set the Path.

Run mongod.exe

To start MongoDB server, we need to run mongod.exe

22

<Course Name> <Lesson Name>

Starting the MongoDB Server (contd.)

D:\set up\mongodb>mongod.exe --dbpath "d:\set up\mongodb\data"

This will show **waiting for connections** message on the console output indicates that the mongod.exe process is running successfully.

Now to run the mongodb you need to open another command prompt and issue the following command.

23

Using MongoDB Shell

```
D:\set up\mongodb\bin>mongo.exe

    MongoDB shell version: 2.2.0
    connecting to: test
    Welcome to the MongoDB shell
```

24

<Course Name>   <Lesson Name>

Creating and dropping database-

D>mongo

>help()

>show dbs

>use <dbname>

>show collections

>db.collectionName.findOne()

>db.collectionName.find()

>db.help()

>db.collectionName.help()

```
C:\appservers\mongo-1.6.3\bin\mongo.exe
MongoDB shell version: 1.6.3
connecting to: test
> show dbs
admin
cfmongodb_tests
default_db
local
mongorocks
test
>
> use mongorocks
switched to db mongorocks
> show collections
people
system.indexes
> db.people.findOne()
{
        "_id" : ObjectId("4cb66dae636ac4fa2045ff31"),
        "COUNTER" : NumberLong(1),
        "LOVESMONGO" : true,
        "NAME" : "Marc",
        "BIKE" : "Felt",
        "LOVESSQL" : true,
        "KIDS" : [
                {
                        "NAME" : "Alexis",
                        "AGE" : NumberLong(7),
                        "DESCRIPTION" : "crazy",
                        "HAIR" : "blonde"
                },
                {
                        "NAME" : "Sidney",
                        "AGE" : NumberLong(2),
                        "DESCRIPTION" : "ornery",
                        "HAIR" : "dirty blonde"
                }
        ],
        "WIFE" : "Heather",
        "TS" : "Wed Oct 13 2010 22:40:46 GMT-0400 (Eastern Daylight Time)"
}
>
```

25

>switched to db cgdb >db.dropDatabase()
>{ "dropped" : "cgdb", "ok" : 1 }

<Course Name>   <Lesson Name>

DataTypes

- **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.

Capgemini

<Course Name>   <Lesson Name>

Summary

> What is NOSQL database

> Advantages of NOSQL

> Why MongoDB

> MongoDB Document database

> MongoDB data model

> Mongo Shell

> Establishing Connection

> Understand about Collection
> on, document and fields

28