SNS COLLEGE OF TECHNOLOGY

Coimbatore-35
An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A++' Grade Approved
by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF COMPUTER APPLICATIONS

19CAE730 – Fundamentals of NOSQL database System

II YEAR III SEM

UNIT III - **CURD in MongoDB**

MongoDB Terminologies for RDBMS concepts

RDBMS		MongoDB
Database	⇒	Database
Table, View	⇒	Collection
Row	⇒	Document (JSON, BSON)
Column	⇒	Field
Index	⇒	Index
Join	⇒	Embedded Document
Foreign Key	⇒	Reference
Partition	⇒	Shard

JSON

"JavaScript Object Notation"

Easy for humans to write/read, easy for computers to parse/generate

Objects can be nested

Built on

- name/value pairs
- Ordered list of values

http://json.org/

BSON

"Binary JSON"

Binary-encoded serialization of JSON-like docs

Embedded structure reduces need for joins

Goals

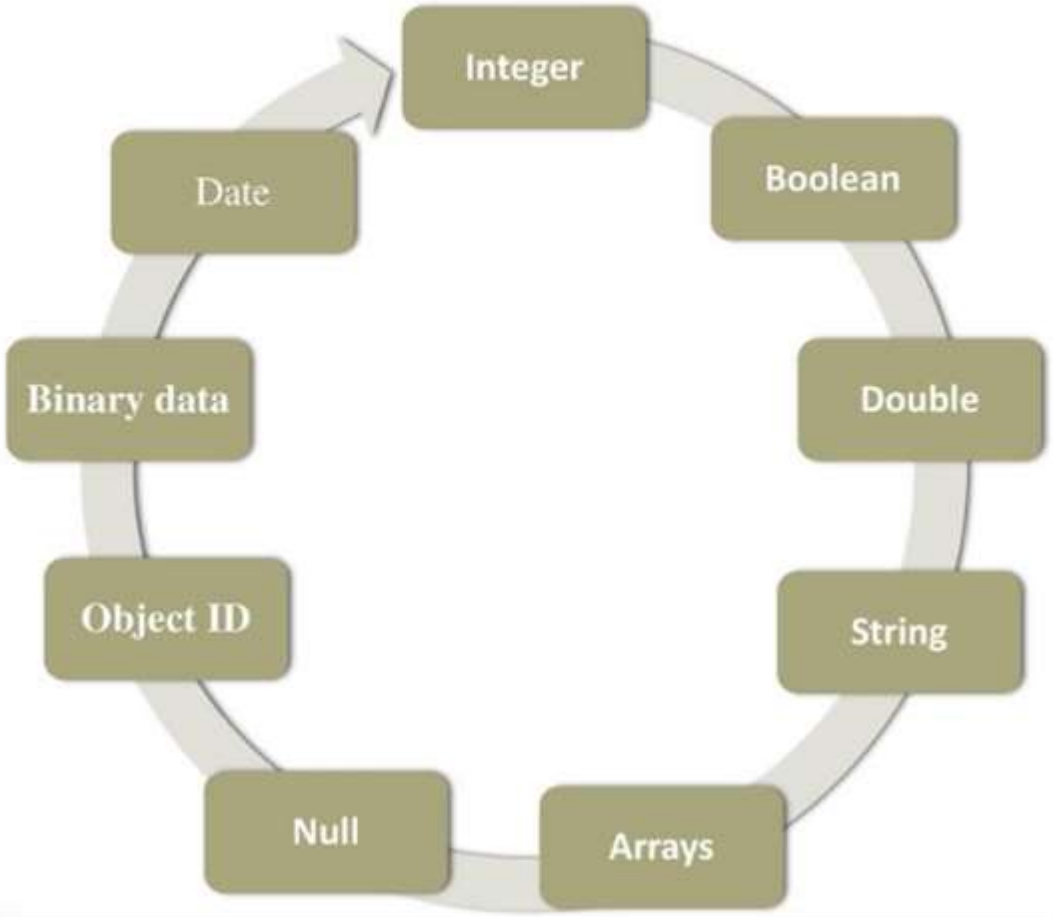
- Lightweight
- Traversable
- Efficient (decoding and encoding)

http://bsonspec.org/

BSON Example

```
{
"_id" :         "37010"
“City" :        “Nashik",
“Pin" :          423201,
"state" :       “MH",
“Postman” : {
                    name: “Ramesh Jadhav”
                    address: “Panchavati”
                  }
}
```

Data Types of MongoDB



Basic Database Operations- **Database**

use *<database name>*	• switched to database provided with ciommand
db	• To check currently selected database use the command **db**
show dbs	• Displays the list of databases
db.dropDatabase ()	• To Drop the database

Basic Database Operations- **Collection**

db.createCollection (name) Ex:- db.createCollection(Stud)	• To create collection
>show collections	• List out all names of collection in current database
db.*databasename*.insert **({Key : Value})** **Ex:- db.Stud.insert({{Name:"Jiya"})**	• In mongodb you don't need to create collection. MongoDB creates collection automatically, when you insert some document.
db.collection.drop() Example:- **db.Stud.drop()**	• MongoDB's **db.collection.drop()** is used to drop a collection from the database.

CRUD Operations

- Insert
- Find
- Update
- Delete

CRUD Operations - **Insert**

- **The insert() Method:-** To insert data into MongoDB collection, you need to use MongoDB's **insert()** or **save()**method.
- **Syntax**

```
>db.COLLECTION_NAME.insert(document)
```

- **Example**

```
>db.stud.insert({name: “Jiya”, age:15})
```

CRUD Operations - **Insert**

- **_id Field**
- If the document does not specify an *_id* field, then MongoDB will add the _id field and assign a unique *ObjectId* for the document before inserting.
- The _id value must be unique within the collection to avoid duplicate key error.

_Id field

_id is 12 Byte field

- 4 Bytes – Current time stamp
- 3 Bytes- Machine Id
- 2 Bytes- Process id of MongoDB Server
- 3 Bytes- Incremental Value.

CRUD Operations - **Insert**

- **Insert a Document without Specifying an _id Field**
  - db.stud.insert({ Name : “Reena", Rno: 15 })
  - db.stud.find()

    { "_id" : "5063114bd386d8fadbd6b004”, “Name” : “Reena", “Rno”: 15 }
- **Insert a Document Specifying an _id Field**
  - db.stud.insert({ _id: 10, Name : “Reena", Rno: 15 })
  - db.stud.find()

    { "_id" : 10, “Name” : “Reena", “Rno”: 15 }

- **Insert Single Documents**

```
db.stud.insert
( {Name: “Ankit”, Rno:1, Address: “Pune”} )
```

Insert Multiple Documents

```
db.stud.insert
( [
 { Name: “Ankit”, Rno:1, Address: “Pune”} ,
 { Name: “Sagar”, Rno:2},
 { Name: “Neha”, Rno:3}
 ] )
```

- **Insert Multivalued attribute**

```
db.stud.insert(
  {
    Name : “Sneha",
    Hobbies: [“Singing”, “Dancing” , “Cricket”] ,
    Rno:8
  }
)
```

- **Insert Multicolumn attri**

```
db.stud.insert(
  {
   Name: “Ritu",
   Address: { City: “Pune",
              State: “MH” },
   Rno: 6
  }
)
```

- **Insert Multivalued with Multicolumn attribute**

```
db.stud.insert(
  {
    Name : “Sneha",
    Awards: [ { Award : “Dancing”, Rank: “1st”, Year: 2008 },
              {Award : “Drawing”, Rank: “3rd”, Year: 2010 } ,
              {Award : “Singing”, Rank: “1st”, Year: 2015 } ],
    Rno: 9
  }
)
```

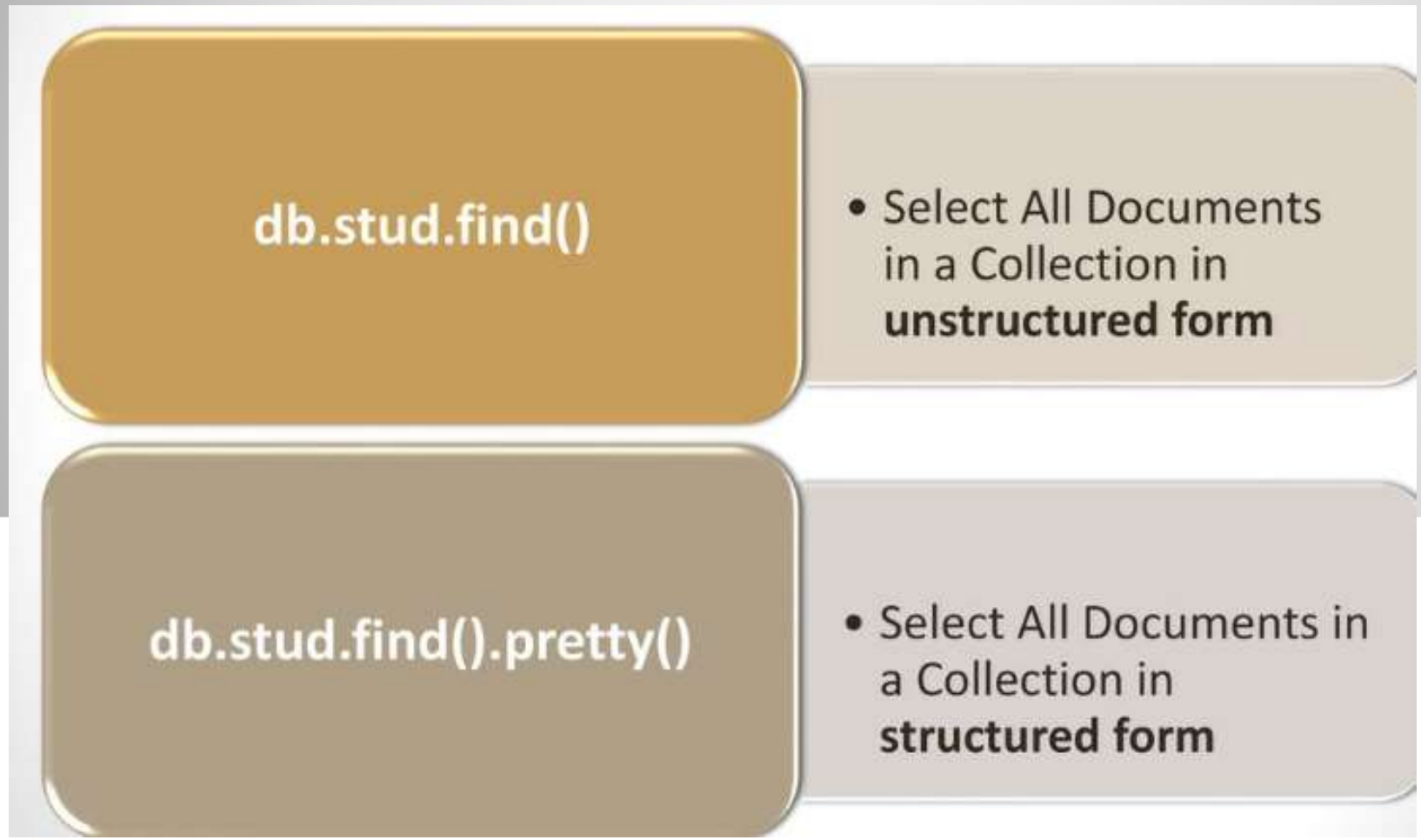
CRUD Operations - **Find**

- **The find() Method-** To display data from MongoDB collection. Displays all the documents in a non structured way.
- **Syntax**

```
>db.COLLECTION_NAME.find()
```

- **The pretty() Method-** To display the results in a formatted way, you can use **pretty()** method.
- **Syntax**

```
>db. COLLECTION_NAME.find().pretty()
```


db.stud.find()
• Select All Documents in a Collection in **unstructured form**
db.stud.find().pretty()
• Select All Documents in a Collection in **structured form**

Specify Equality Condition

- use the query document
  { <field>: <value> }
- Examples:
- db.stud.find(name: “Jiya" })
- db.stud.find({ _id: 5 })

Comparison Operators

Operator	Description
$eq	Matches values that are equal to a specified value.
$gt	Matches values that are greater than a specified value.
$gte	values that are greater than or equal to a specified value.
$lt	Matches values that are less than a specified value.
$lte	Matches values that are less than or equal to a specified value.
$ne	Matches all values that are not equal to a specified value.
$in	Matches any of the values specified in an array.
$nin	Matches none of the values specified in an array.

CRUD Operations – Find Examples with comparison operators

```
db.stud.find( { rno: { $gt:5} } )
```

Shows all documents whose rno>5

```
db.stud.find( { rno: { $gt: 0, $lt: 5} } )
```

Shows all documents whose rno greater than 0 and less than 5

db.stud.find({name: “Jiya”},{Rno:1})

To show the rollno of student whose name is equal to Jiya (by default _id is also shown)

db.stud.find({name: “jiya”},{_id:0,Rno:1})

show the rollno of student whose name is equal to Jiya (_id is not shown)

db.stud.find().sort({ Rno: 1 })

Sort on age field in Ascending order (1)

db.stud.find().sort({ Rno: -1 })

Sort on age field in Ascending order(-1)

db.stud.find().count()

Returns no of documents in the collection

db.stud.find({Rno:2}).count()

Returns no of documents in the collection which satisfies the given condition Rno=2

db.stud.find().limit(2)

Returns only first 2 documents

db.stud.find().skip(5)

Returns all documents except first 5 documents

db.stud.findOne() - *Find first document only*

db.stud.find({"Address.city": "Pune"})-
Finding in Multicolumned attribute

db.stud.find({name: "Riya",age:20})
Find documents whose name is Riya and Rno is 20

db.stud.find({name:{$in:["riya","jiya"]}})

Find information whose name is riya or jiya

db.stud.find({Rno:{$nin:[20,25]}})

Find information whose rollno is not 20 or 25

db.stud.distinct("Address")

Find from which different cities students are coming

db.stud.find({name:/^n/})

Find students whose name starts with n

db.stud.find({name:/n/})

Find students whose name contains n letter

db.stud.find({name:/n$/})

Find students whose name ends with n

db.collection.stats()

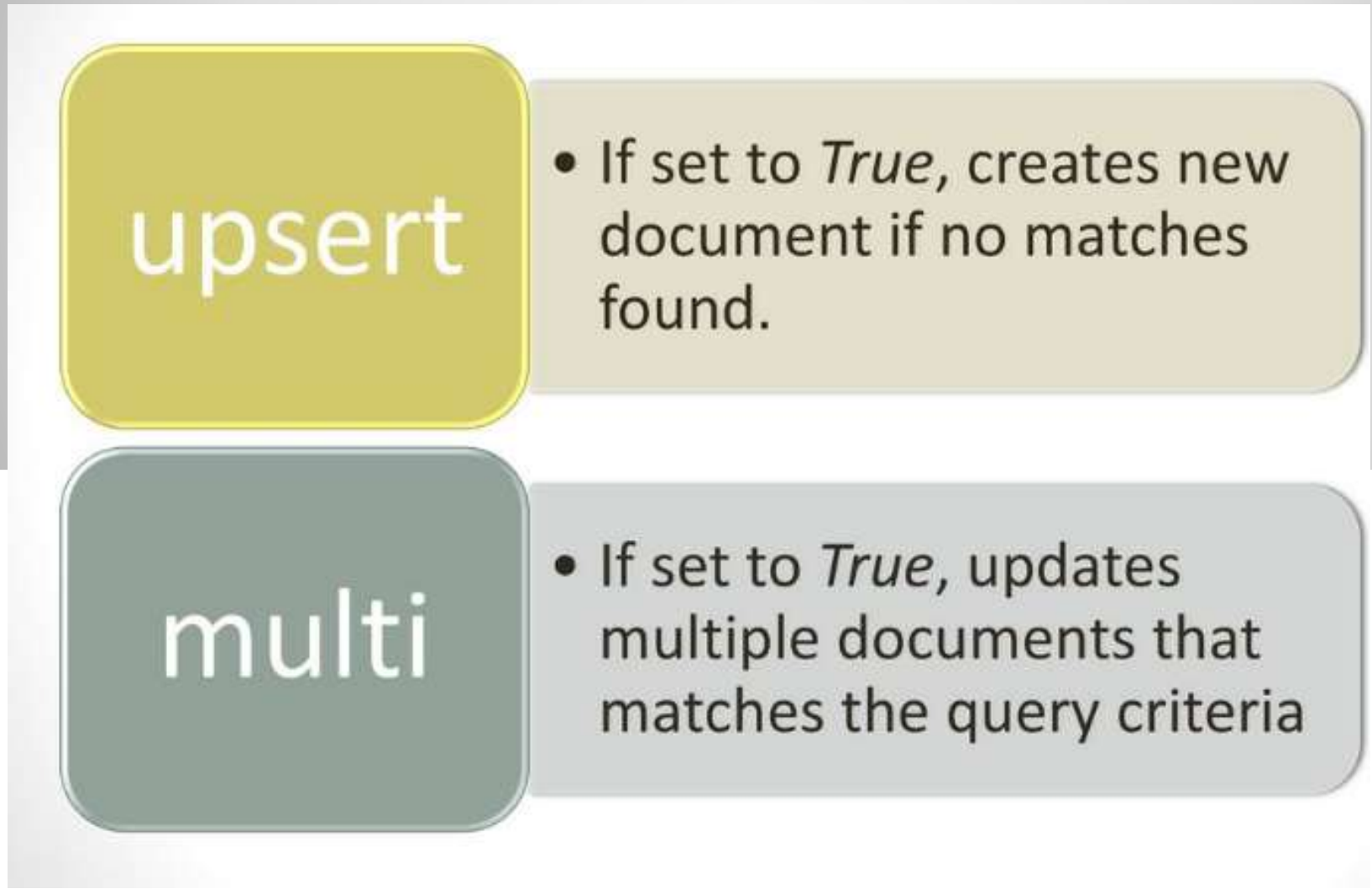
db.collection.explain().find()

db.collection.explain().find().help()

CRUD Operations – **Update**

Syntax

```
db.CollectionName.update(
  <query/Condition>,
  <update with $set or $unset>,
  {
    upsert: <boolean>,
    multi: <boolean>,
  }
)
```


upsert
• If set to *True*, creates new document if no matches found.
multi
• If set to *True*, updates multiple documents that matches the query criteria

Examples

```
db.stud.update(
{ _id: 100 },
{ age: 25})
```

- Set age = 25 where id is 100
- First Whole document is replaced where condition is matched and only one field is remained as age:25

```
db.stud.update(
{ _id: 100 },
{ $set:{age: 25}})
```

- Set age = 25 where id is 100
- Only the age field of one document is updated where condition is matched

```
db.stud.update(
{ _id: 100 },
{ $unset:{age: 1}})
```

- To remove a age column from single document where id=100

Examples

```
db.stud.update(
{ _id: 100 },
{ $set: { "marks.dmsa": 50} })
```

- Set marks for dbms subject as 50 where id = 100 (only one row is updated)

```
db.stud.update(
{ class: "TE" },
{ $set: { "marks.dmsa": 50} } ,
{ multi: true } )
```

- Set marks for dbms subject as 50 where class is TE (all rows which matches the condition were updated)

```
db.stud.update(
{ class: "TE" },
{ $set: { "marks.dmsa": 50} } ,
{ upsert: true } )
```

- Set marks for dbms subject as 50 where class is TE (all rows which matches the condition were updated)
- If now row found which matches the condition it will insert new row.

CRUD Operations – **Remove**

Remove All Documents	• db.inventory.remove({})
Remove All Documents that Match a Condition	• db.inventory.remove ({ type : "food" })
Remove a Single Document that Matches a Condition	• db.inventory.remove ({ type : "food" }, 1)

References

- https://docs.mongodb.com/manual/introduction/
- http://metadata-standards.org/Document-library/Documents-by-number/WG2-N1501-N1550/WG2_N1537_SQL_Standard_and_NoSQL_Databases%202011-05.ppt
- https://www.slideshare.net/raviteja2007/introduction-to-mongodb-12246792
- https://docs.mongodb.com/manual/core/databases-and-collections/
- https://docs.mongodb.com/manual/crud/