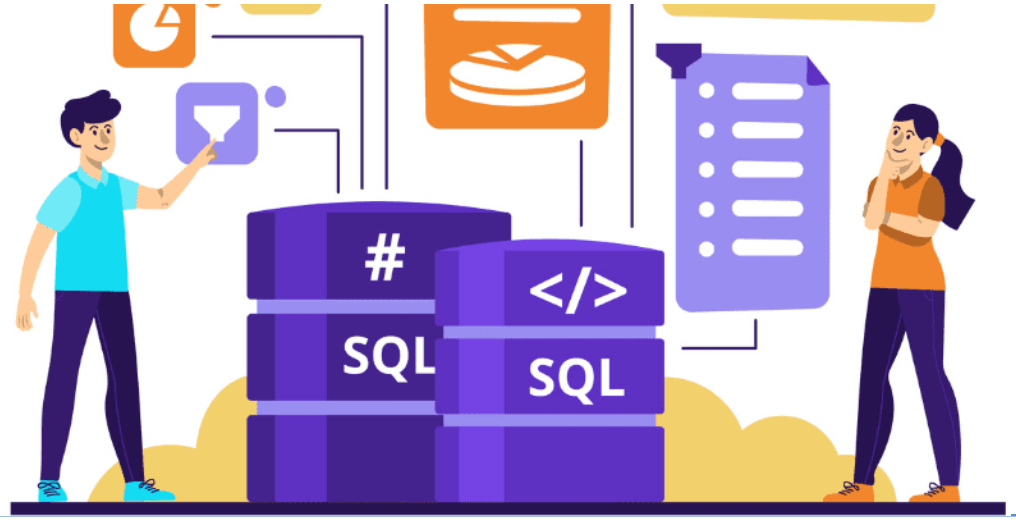
SNS COLLEGE OF TECHNOLOGY

Re-accredited by NAAC with A+ grade, Accredited by NBA(CSE, IT, ECE, EEE & Mechanical)
Approved by AICTE, New Delhi, Recognized by UGC, Affiliated to Anna University, Chenna



Department of Computer Applications

INDEXING in MongoDB

COURSE : 23CAT603- Database Management System
UNIT V : Column Oriented Database
CLASS : I Semester / I MCA

Search / Find Operation

- Searching for some time based on specific category
- No of scanned documents – depends on the location where searching item is placed (single document search)
- Scans all the documents in the collection and match them (multi document search)
- How does normal search works?

  Linear Search

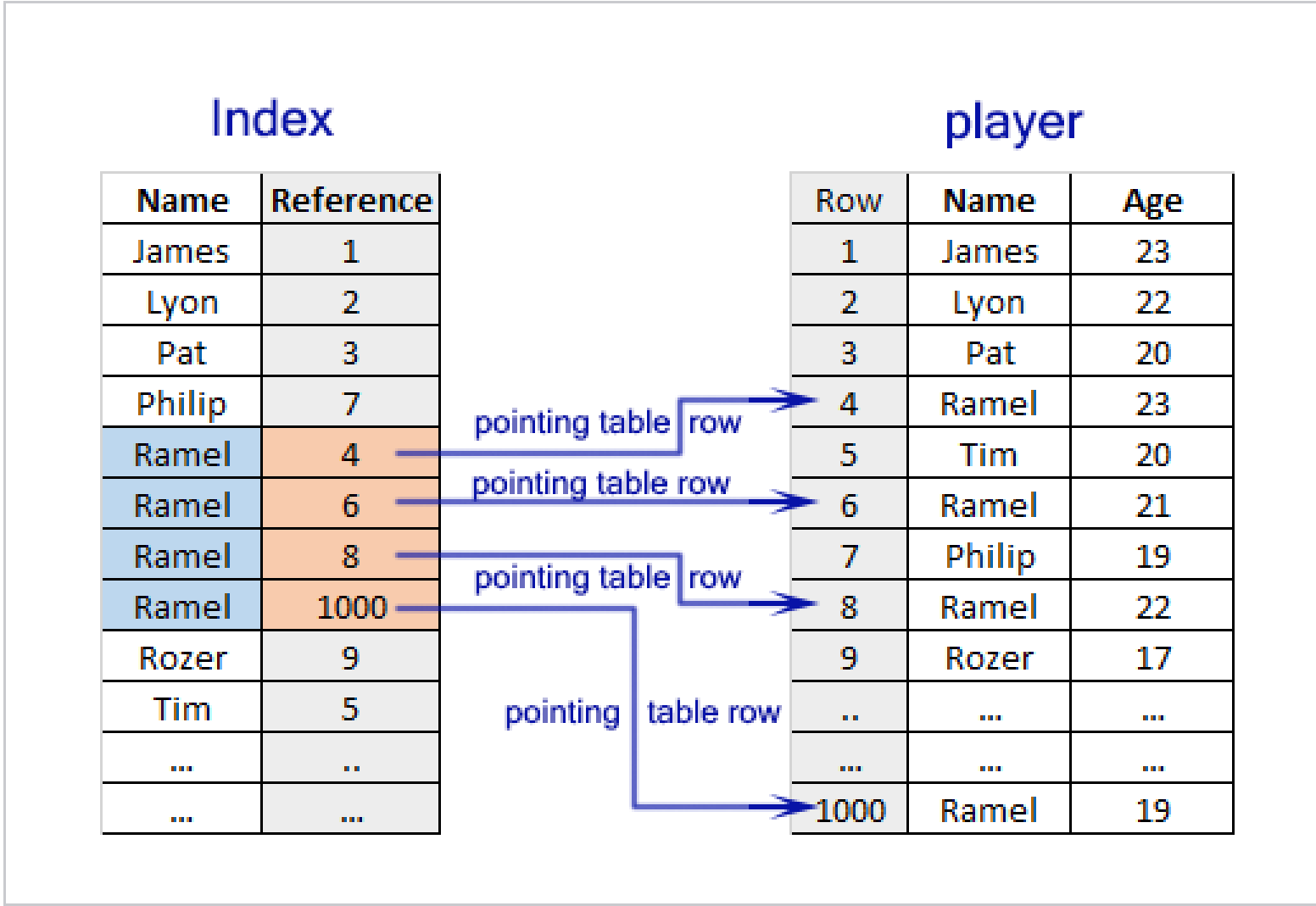
Search / Find Operation

- Indexing is a critical tool to increase database performance
- Without index, **MongoDB** must scan every document in a collection to return query results.
- All the collections have a default index on the _id field, prevents clients from inserting two documents with the same value for the _id field
- MongoDB indexes use a B-tree data structure

Indexing

Indexing is a special data structure contains a portion of data from database used to locate specific document in collection very quickly without traversing every document

Indexing - Example



Name	Reference
James	1
Lyon	2
Pat	3
Philip	7
Ramel	4
Ramel	6
Ramel	8
Ramel	1000
Rozer	9
Tim	5
...	..
...	...

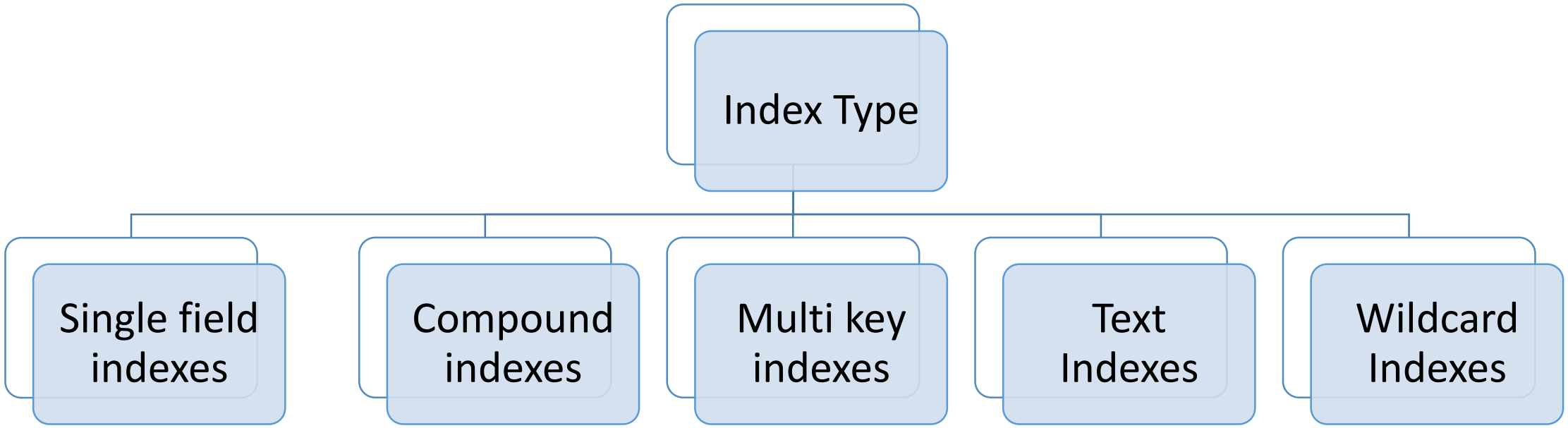
Row	Name	Age
1	James	23
2	Lyon	22
3	Pat	20
4	Ramel	23
5	Tim	20
6	Ramel	21
7	Philip	19
8	Ramel	22
9	Rozer	17
..	...	...
...	...	...
1000	Ramel	19

INDEXING

Index Names

- default name is the concatenation of the indexed keys and each key's direction in the index (1 or -1) using underscores as a separator.
- For Example
  { item : 1, quantity: -1 } has the name item_1_quantity_-1
- cannot rename an index once created

Types of Index



Types of Index

Single Index

- Sort order for single/each field
- Sort order is not important, MongoDB can traverse the data in both directions.

```
db.employee.find().sort({“name”:1})
```

Compound Index

- Specify multiple indexed fields
- order in which you specify the fields is important
- MongoDB recommends
  - first, add the fields against which equality queries are run, that is exact matches on a single value
  - next, add fields that reflect the sort order of the query
  - finally, add fields for range filters

```
db.employee.find().sort({“name”:1,”city”:1})
```

Types of Index

Multi key Index

- we index a field with an array value, MongoDB creates separate index entries for each element of the array
- MongoDB automatically determines whether to create a multikey index if the indexed field contains an array value
- Not need to specify the multikey type explicitly

```
db.employee.find().sort({“name”:1,”city”:1})
```

Text Index

- Searching for string content in a collection
- Text indexes restrict the words in a collection to only store root words (no like “a”, “the”, “or”

```
db.collection_name.createIndex({name:”text”})
```

INDEX Properties

Property Name	Description
Unique	Rejects/Restrict duplicate values for the indexed field For example, **db.employee.createIndex({name:1},{unique:true})**
Partial	Partial Indexes only index the documents that match the filter criteria For example, **db.employee.createIndex({city:1},{partialFilterExpression:{name:"ankit"}})**
Sparse	Ensures that the index only contains entries for documents with the indexed field and skip the documents without the indexed field For Example, **db.employee.createIndex({name:1},{unique:true,sparse:true})**
TTL	"total time to live" indexes are used to auto-delete documents from a collection after the specified time duration. Ideal for machine-generated data, logs and session information For example, **db.employee.createIndex({name:1},{expireAfterSeconds:20})**

INDEXING -Limitations

Range

- Collection cannot have indexed more than 64
- The name of the index can contain only 164 characters.
- A compound index can have 31 fields indexed at max

RAM Usage

- Total size of indexes must not exceed the RAM
- limit exceeds, it will cause deletion of some indexes

Query

- Queries should not use expressions like $nin, $not, etc.
- Queries should not use arithmetic operators like $mod etc.
- A Query should not use $where clause

MongoDB will not create an index if the value of existing index field exceeds the index key limit.

CREATE INDEX

db.collection_name.createIndex({Key name: 1 or -1})

- For Example

db.employee.createIndex({“name”:1})

db.employee.createIndex({“name”:1, “city”:1})

- Output will be like

```
{
"createdCollectionAutomatically" :
false,
"numIndexesBefore" : 1,
"numIndexesAfter" : 2,
 "ok" : 1

}
```

INDEX Options

Parameter	Type	Description
background	Boolean	Builds the index in the background so that building an index does not block other database activities. The default value is false.
unique	Boolean	Creates a unique index so that the collection will not accept insertion of documents where the index key or keys match an existing value in the index.
name	string	The name of the index. If unspecified, MongoDB generates an index name by concatenating the names of the indexed fields and the sort order.
sparse	Boolean	If true, the index only references documents with the specified field. These indexes use less space but behave differently in some situations (particularly sorts). The default value is false.
expireAfterSeconds	integer	Specifies a value, in seconds, as a TTL to control how long MongoDB retains documents in this collection.
weights	document	The weight is a number ranging from 1 to 99,999 and denotes the significance of the field relative to the other indexed fields in terms of the score.
default_language	string	For a text index, the language that determines the list of stop words and the rules for the stemmer and tokenizer. The default value is English.
language_override	string	For a text index, specify the name of the field in the document that contains, the language to override the default language. The default value is language.

DROP INDEX

```
db.COLLECTION_NAME.dropIndex({KEY:1})
```

- For Example

```
db.employee.dropIndex({“name”:1})
```

```
db.COLLECTION_NAME.dropIndexes()
```

```
db.COLLECTION_NAME.getIndexes()
```

CRUD Operations - INSERT

- To insert multi columns

```
db.student.insert({name:”rem”, address {city: “cbe”, place:”mtp”}} )
```

- To insert multivalued attribute

```
db.student.insert(
    {name:”rem”,
     hobbies[“singing”, “dancing”]
    } )
```

- To insert multivalued with multicolumn attribute

```
db.student.insert(
    {name:”rem”,
    award[ {Rank:1, year: 2008, skill: “singing”},
           {Rank:3, year: 2008, skill: “drawing”}
           {Rank:3, year: 2011, skill: “drawing”} ]
          } )
```

CRUD Operations – READ/FIND

- To display data from collection

  db.collectionname.find()

- To display data from collection in the formatted way

  db.collectionname.find().pretty()

- To use query the document

  {<field>:<value>}

- For example

  Db.student.find({name:"rag"})

CRUD Operations – READ/FIND

$eq	equal
$gt	Greater than
$lt	Less than
$gte	Greater than or equal to
$gte	Less than or equal to
$ne	Not equal
$in	Matches any one of value in an array
$nin	Matches none of the values in an array

- For example

  db.student.find({rno: {$gt: 501}})

  db.student.find({rno: {$gt: 501, $lt: 600}})

  To find particular column

  db.student.find({rno: 501})

CRUD Operations – READ/FIND

- To return no.of documents in the collection by

  db.student.find().count()

  To return no.of documents in the collection with condition by

  db.student.find({rno: 501}).count()

  To return first two documents in the collection by

  db.student.find({rno: 501}).limit(2)

  To return documents by skipping first 5 by

  db.student.find({rno: 501}).skip(5)

  To return first document by

  db.student.findOne()

CRUD Operations – READ/FIND

- To return documents whose attribute value is either one

  db.student.find({name:[“riya”, “jaya”]})

  To return documents whose information not in

  db.student.find({rno:{$nin[510,515]}}

  To return documents with distinct value

  db.student.distinct(“address”)

  To return documents by skipping first 5 by

  db.student.find({rno: 501}).skip(5)

  To return first document by

  db.student.findOne()

CRUD Operations – UPDATE

- Syntax for it

```
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, update multiple documents meets criteria

- Example

```
Db.student.update({_id:101}, {$set{age:23}})
Db.student.update({_id:101}, {$unset{age:1}})
```

CRUD Operations – DELETE

- To remove all documents

  db.student.remove({})

- To remove all documents which matches condition

  db.student.remove({type:"food"})

- To remove a single document which matches condition

  db.student.remove({type:"food"},1)

To return documents by skipping first 5 by

db.student.find({rno: 501}).skip(5)

To return first document by

db.student.findOne()

CRUD Operations – DELETE

- https://www.tutorialspoint.com/NoSQL-Databases
- https://www.geeksforgeeks.org/nosql-data-architecture-patterns
- https://www.mongodb.com/nosql-explained
- Shannon Bradshaw, Eoin Brazil, and Kristina Chodorow, “MongoDB: the Definitive Guide”, O’Reilly Media, 3rd Edition


Thank
You!


SQL