MongoDB uses CRUD (Create, Read, Update, Delete) operations to manage documents within database.  
**1. Inserting Documents (Create):**

Think of adding new treasures to data chest. Here's how:

Method: insert\_one() or insert\_many()

# Insert single document

collection.insert\_one({"name": "Priyanka", "age": 30})

# Insert multiple documents

documents = [{"name": "Priya", "age": 25}, {"name": "Parisa", "age": 40}]

collection.insert\_many(documents)

**2. Retrieving Documents (Read):**

Now, time to retrieve precious data pieces!

**Use methods like find() or findOne() to retrieve data**

# Find all documents

documents = collection.find()

# Find documents with specific criteria

for document in collection.find({"age": {"$gt": 25}}):

print(document)

**3.Replace Operation:** In MongoDB, the replaceOne() method facilitates a complete replacement of an existing document with a new one. Unlike relational databases, MongoDB does not support partial updates; instead, it replaces the entire document with an updated version containing the required fields.

Example:

Relational Database: Updating a specific field in a record.

MongoDB: Replacing the entire document with a new version containing updated fields.

# Import MongoClient class from pymongo package

from pymongo import MongoClient

# Connect to MongoDB server

client = MongoClient()

# Access the desired database

db = client.mydatabase

# Access the desired collection

collection = db.mycollection

# Define the filter to identify the document to replace

filter = {"name": "Priya"}

# Define the new document with updated fields

new\_document = {

"name": "Priya",

"age": 35,

"city": "HArtford"

}

# Replace the existing document with the new one

result = collection.replaceOne(filter, new\_document)

# Print the number of documents replaced

print("Number of documents replaced:", result.modified\_count)

**2. Delete Operation:**Similarly, MongoDB's deleteOne() and deleteMany() methods remove entire documents based on specified criteria. This contrasts with relational databases, where operations typically target specific rows or records.

Example:

Relational Database: Deleting a single row from a table.

MongoDB: Deleting all documents in a collection that match a specific age range.

# Import MongoClient class from pymongo package

from pymongo import MongoClient

# Connect to MongoDB server

client = MongoClient()

# Access the desired database

db = client.mydatabase

# Access the desired collection

collection = db.mycollection

# Define the filter to identify documents to delete

filter = {"age": {"$gte": 40}} # Delete documents with age greater than or equal to 40

# Delete a single document matching the filter

result = collection.deleteOne(filter)

# Print the number of documents deleted

print("Number of documents deleted:", result.deleted\_count)

**Key Differences:**

* Granularity: MongoDB's replace and delete operations affect entire documents rather than individual fields.
* Immutability: Replaced documents in MongoDB are entirely new, rather than modifications of existing ones.
* Filtering: Criteria for deletion target entire documents, as opposed to specific rows or fields in relational databases.

**Benefits of MongoDB Approach:**

* + Simplicity: The operations are less complex, potentially making them easier to understand and implement.
  + Performance: Replacing or deleting entire documents can be more efficient, especially for large datasets.
  + Scalability: MongoDB's approach facilitates easier distribution of document operations across servers, enhancing scalability.