Notes By: -
Lokesh Bagora **Reference: - Infytq Resources**

InfyTQ DBMS PART-2
Advanced Topics

Important Topics: -

1. Subquery
2. Transactions
3. No SQL Databases
4. Types of No SQL
5. Mongo DB
6. Mongo DB All Functions

Notes By: -
Lokesh Bagora **Reference: - Infytq Resources**

1. Subquery: -

Subquery in SELECT

```
SELECT Id, EName, Salary,
(SELECT AVG(Salary) FROM Employee) AS AvgSal FROM Employee
```

Subquery in FROM

```
SELECT * FROM (SELECT ID, EName, Salary FROM Employee) A
```

Subquery in WHERE

```
SELECT Id, EName, Salary FROM Employee A WHERE Salary
= (SELECT MAX(SALARY) FROM Employee B)
```

Subquery in HAVING

```
SELECT Dept FROM Employee GROUP BY Dept HAVING SUM(Salary)
= (SELECT MAX(SUM(Salary)) FROM Employee GROUP BY Dept)
```

The subquery is mainly divided into two types based on how it is internally processed to fetch the required details from the tables.

1. **Independent Subquery**

2. **Correlated Subquery**

Notes By: -
Lokesh Bagora

Reference: - Infytq Resources

1. Independent Subquery: -

```
SELECT ID, EName, Salary FROM Employee A WHERE Salary
= (SELECT MAX(Salary) FROM Employee B)
```

Input: Employee table

ID	ENAME	SALARY
1	James Potter	75000
2	Ethan McCarty	90000
3	Emily Rayner	25000

Step 1: Inner query executes
SELECT MAX(Salary) FROM Employee B

ID	ENAME	SALARY
1	James Potter	75000
2	Ethan McCarty	90000
3	Emily Rayner	25000

Step 2: The result of inner query is substituted in outer query
SELECT ID, EName, Salary FROM Employee A WHERE Salary = 90000

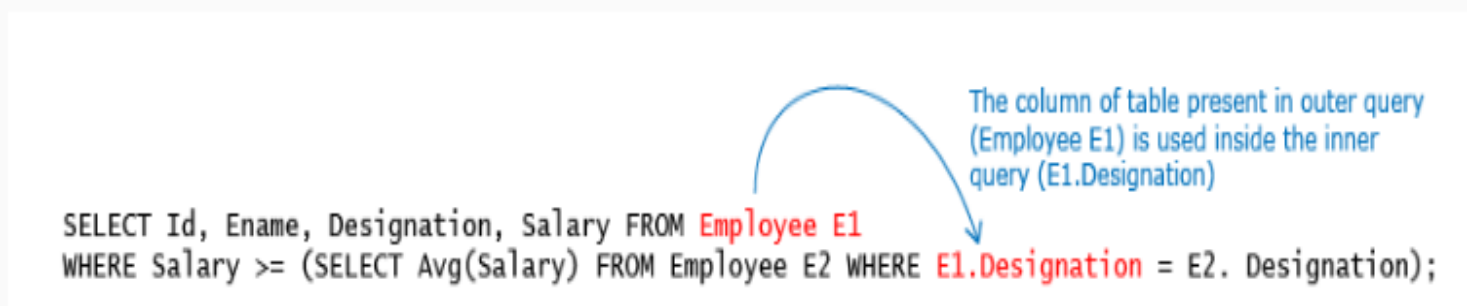
Step 3: Outer query executes

ID	ENAME	SALARY
2	Ethan McCarty	90000

Notes By: -
Lokesh Bagora

Reference: - Infytq Resources

2. Correlated Subquery: -



Notes By: -
Lokesh Bagora **Reference: - Infytq Resources**

#Transactions

What is a transaction?

A transaction is a logical unit of work containing one or more operations on a database. It provides two important functions:

- Ensures that all operations within a transaction happen in an atomic manner
- Provides the capability to undo the partial processing in the event of a failure at any step

Purpose of transaction:

There are two main purposes of transaction

1. Produce a reliable unit of work, to help in recovery during failures in the database
2. Bring in isolation among the programs that access data concurrently from the database

How is a transaction performed in the database?

A transaction is performed by executing single or many SQL queries that alters the state of a database from one state to another consistently over time. The SQL commands we have learnt so far such as CREATE, ALTER, INSERT, DELETE etc., can alter the state of a database.

The diagram given below gives a picture of the execution of various queries that alters the state of the database.

Notes By: -
Lokesh Bagora **Reference: - Infytq Resources**

- Initially, there are two tables in the database (tables with colors orange and blue)
- Create statement is executed to add a new table (table with color green)
- Insert statement is executed to add few data into a table (table with color orange), additional data in the table is added (rows in color red)

All these operations are executed in the database over a time period.

he possibilities for failures of the mentioned operations are:

- Integrity constraint violations
- Inconsistency in the operations
- Connectivity issue with the bank server, etc.

hese two operations should be executed consistently one after the other in the database, then only the fund transfer will be successful.

he database provides the following transaction statements to achieve consistency in transaction management.

Statement	Description
Set transaction	Initiates the transaction
Commit	Successfully completes the transaction. Actions of a transaction cannot be rolled back after commit has been executed.
Rollback	Ends the transaction after undoing all the work performed after begin transaction statement.

hat would happen if the first update succeeds and the second one fails? It leaves the database in an inconsistent state from a business perspective.

Notes By: -
Lokesh Bagora **Reference: - Infytq Resources**

AUTOCOMMIT:

Every operation that changes the state of the database is a transaction. So, till now every operation you have performed was being executed in a transaction. But how is this possible? We never used set transaction, commit or rollback commands for these operations so far.

The database servers start a transaction automatically whenever they encounter the first SQL statement. An easy way to understand this is to imagine that all SQL statements are surrounded by set transaction and commit commands as given below.

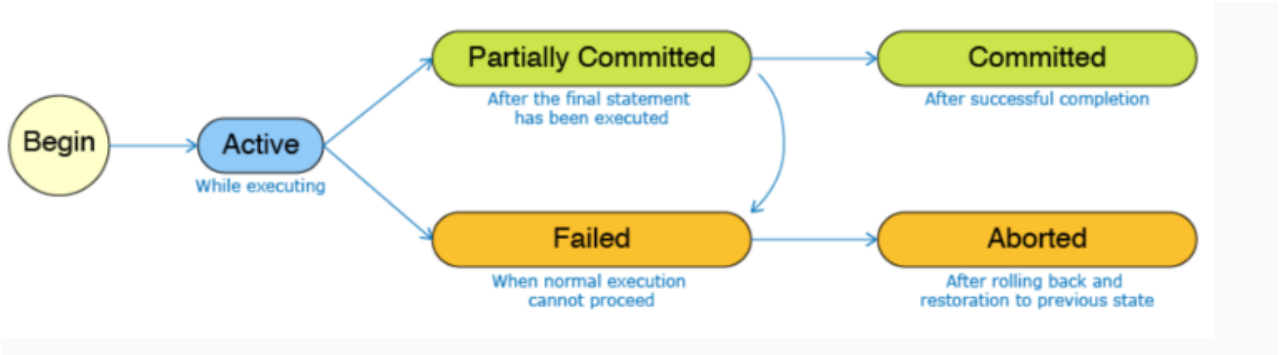
The AUTOCOMMIT property of a connection controls the automatic issue of the commit after the operation. AUTOCOMMIT can have ON or OFF values and it depends on the default setting of the client you are using to connect to the database.

Commands related to AUTOCOMMIT are given below:

Command	Description
SET AUTOCOMMIT ON	Changes the mode of connection to ON. In this mode COMMIT command is automatically issued after every SQL statement that alters the state of a database.
SET AUTOCOMMIT OFF	Changes the mode of connection to OFF. In this mode user is expected to provide an explicit COMMIT or ROLLBACK command to complete the transaction.
SHOW AUTOCOMMIT	Display the current state of AUTOCOMMIT property.

Notes By: -
Lokesh Bagora **Reference: - Infytq Resources**

The states of a transaction are as follows:



ACID properties of transaction:

Every transaction in a database has the following properties:

Property	Description
Atomicity	All operations within the transaction must all succeed or fail.
Consistency	A transaction always moves the database from one consistent state to another. Hence all integrity and data constraints must be satisfied.
Isolation	Transactions execute in isolation of each other. In other words partial execution of one transaction is not visible to other transactions. Only committed data is visible to other transactions.
Durability	Once a transaction is committed, it is permanently saved, the data is preserved even in the case of power failure, hardware failure etc.

Notes By: -
Lokesh Bagora **Reference: - Infytq Resources**

#NoSQL Databases: -

Let's say you need to register on a portal where the form has 6 fields, of which 3 are mandatory. Would you fill in all the fields?

Often users fill only mandatory fields. This leaves the NULL value in the optional fields in the relational data. This is because SQL databases follow a rigid schema wherein every row must store values for all columns. This also results in inefficient storage.

SQL Database

	First Name	Last Name	Email	Gender	Landline	Mobile
Row 1	James	NULL	james@abc.com	Male	123	456
Row 2	Ethan	McCarty	ethan@abc.com	NULL	NULL	789

NoSQL databases allow a flexible schema wherein each row stores only the required data. NULL values are not stored. NoSQL allows multiple values to be stored in a single column. See UserName and Contact Number in the below example.

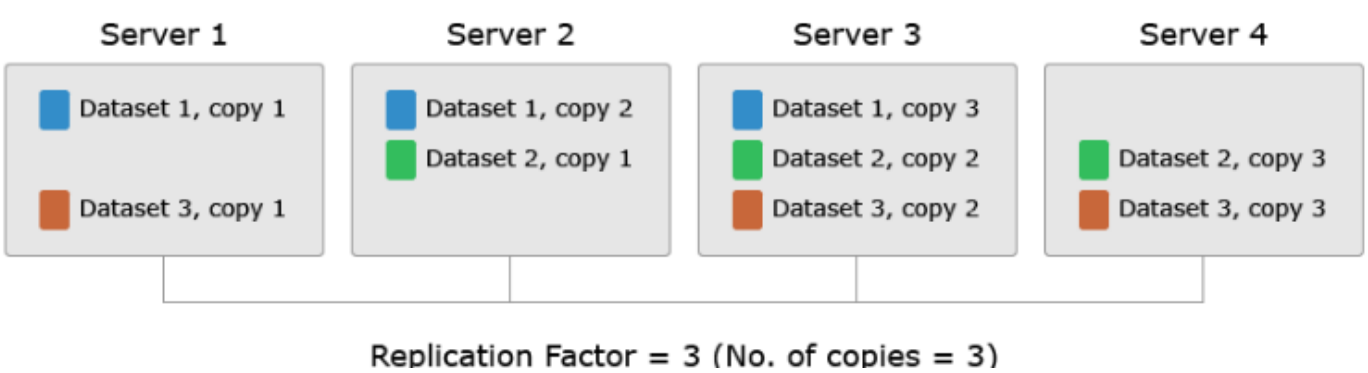
Notes By: -
Lokesh Bagora **Reference: - Infytq Resources**

NoSQL Database

	UserName	Email	Gender	ContactNo
Row 1	{firstName:"James"}	james@abc.com	Male	{landLine:"123", mobile:"456"}

	UserName	Email	ContactNo
Row 2	{firstName:"Ethan", lastName:"McCarty"}	ethan@abc.com	{mobile:"789"}

NoSQL servers usually run on low-cost hardware which can fail anytime. Therefore, the data is replicated and stored on multiple servers. In case any server fails, the data is still available in another server.



Notes By: -
Lokesh Bagora **Reference: - Infytq Resources**

#CAP Theorem:-

All databases should ideally provide the following features:

Consistency – An end-user must be able to see the latest data at all times.

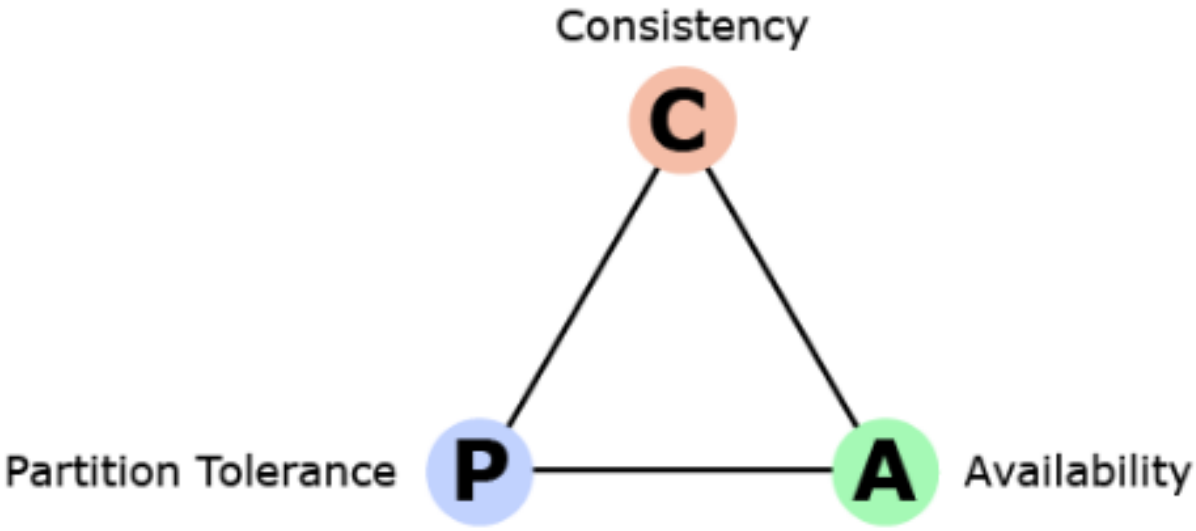
Availability – Every database request must be responded to by the server.

Partition Tolerance – When two systems cannot talk to each other in a network, it is called network partition. Our DB system should continue to function even if there is a network partition.

Consistency, **A**vailability and **P**artition Tolerance is usually abbreviated as **CAP**.

Notes By: -
Lokesh Bagora

Reference: - Infytq Resources



Notes By: -
Lokesh Bagora **Reference: - Infytq Resources**

#Types of NOSQL: -

Key-Value Store characteristics :

- We can search only by using keys. We cannot search by value. In this example below, we cannot search for customers who have chosen to buy a Canon Printer.
- Key-Value stores can be stored entirely in-memory (RAM). Therefore they have very quick response times.
- They are inefficient when you need to query or update only a part of the value.

Column Family Store characteristics :

1. Column Family databases can handle hundreds of terabytes of data easily.
2. Updates are performed without reading the row that contains it. Hence writes are very quick.
3. Column Family databases support the real-time insertion of the huge amounts of data e.g. one million writes per second.

Column Family Store use cases:

1. Real-time weather data (min. temperature, max. temperature, air pressure etc.) collected through sensors at multiple locations.
2. Log files of web servers for data analysis.

Notes By: -
Lokesh Bagora **Reference: - Infytq Resources**

Document Oriented databases have the following characteristics:

1. Supports flexible schema, i.e. each document can have different attributes.
2. Provides a rich query language for storing, fetching, modifying and deleting data. This allows search and filter by any attribute value.
3. Ensures rapid application development e.g. In a fast-changing retail store scenario, where products and their attributes are frequently changing.
4. Are suitable for storing data that does not require frequent updates, but is read many times.

These are a few common operations in MongoDB - a popular document-oriented database. Here _id attribute is the primary key for the product.

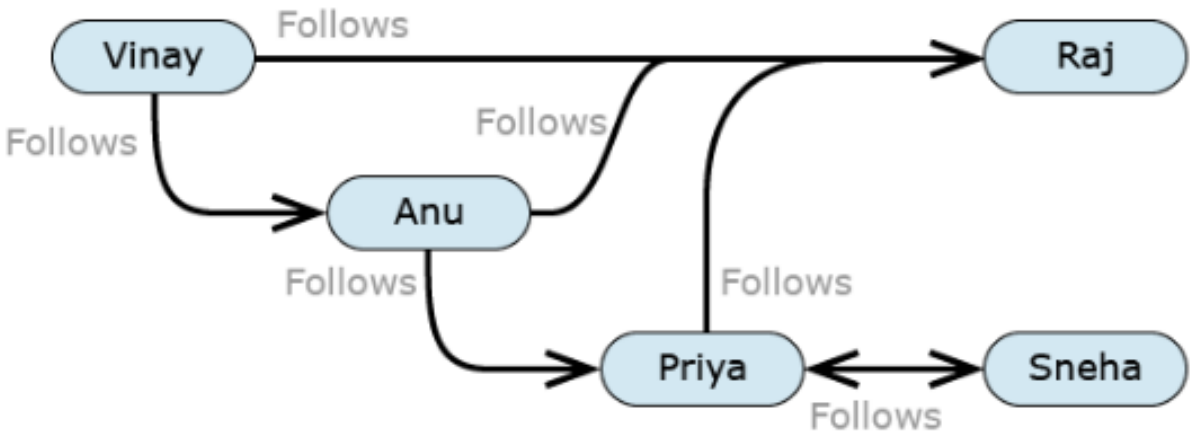
Operation	Query
Insert	db.product.insert ([{ _id: 111, category: "book", price: 250, author: "Herbert Schildt", title: "Java Programming" }, { _id: 222, category: "tv", price: 20000, screensize: 40, display: "LCD" }, { _id: 333, category: "mobile", os: "Android", screensize: "5 inch", camera:"13MP", price: 4000 }]);
Search	db.product.find({ price:{$gt: 100}, category: "book" });
Update	db.product.update({ _id:222}, {$set:{price:1500}});
Delete	db.product.remove({_id:333});

Notes By: -
Lokesh Bagora **Reference: - Infytq Resources**

Graph Database characteristics :

1. Graph databases are suited for data that are heavily interconnected through relationships.
2. Graphs do not need joins for querying.
3. Graph databases use graph theory for traversal. It improves performance by keeping track of and thereby skipping nodes already visited.
4. Graph databases provide Atomicity, Consistency, Isolation and Durability (similar to SQL).

Twitter users and their followers can be represented by this **graph**.



1. Each user is represented as a node that has a two attributes: userid and username.
2. Nodes are connected to each other through relationship called follows.

Notes By: -
Lokesh Bagora

Reference: - Infytq Resources

#MongoDB:-

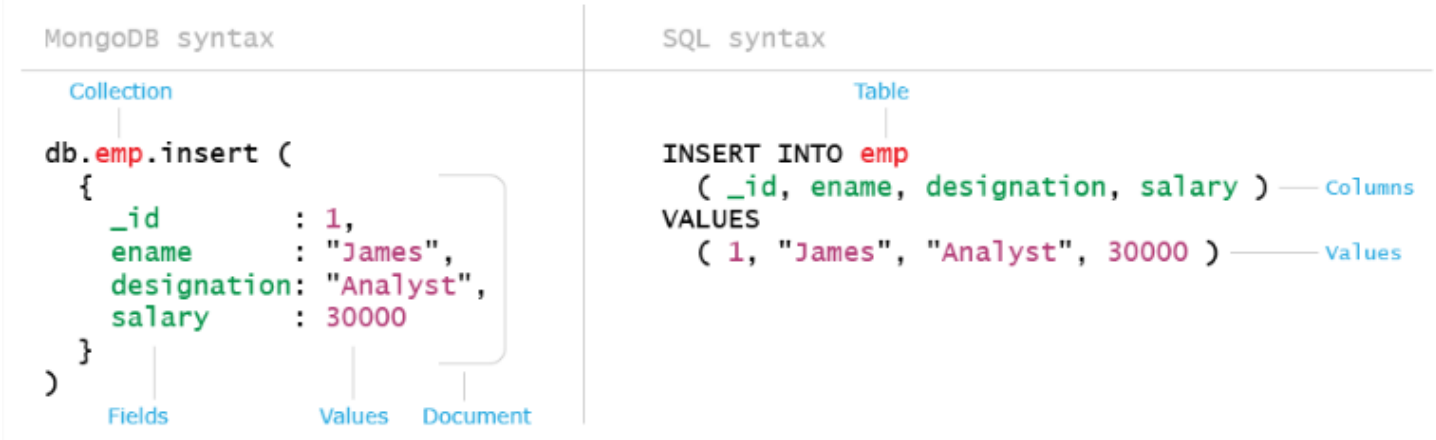
MongoDB is an open source, document-oriented database. It is designed to be highly scalable and offers high developer productivity. MongoDB stores data in JSON-like documents which have dynamic schema.

MongoDB	Oracle
Stores data in Collections.	Stores data in Tables.
Unit of data storage is a Document which is stored in a Collection.	Unit of data storage is a Record (table row).
Collections have dynamic schema i.e. documents in collection can have different attributes.	Tables have fixed schema i.e. attributes are pre defined before inserting data. Explicit NULL value has to be provided if data is missing for an attribute.
CRUD operations are performed through insert, find, update and remove operations on Collection object.	CRUD operations are performed through INSERT, SELECT, UPDATE and DELETE statements.
PRIMARY KEY uniquely identifies a document in a Collection. PRIMARY KEY field has a predefined name _id.	PRIMARY KEY uniquely identifies a record in a Table. You can choose any name for PRIMARY KEY.
NOT NULL, UNIQUE, FOREIGN KEY and CHECK constraints are not supported.	NOT NULL, UNIQUE, FOREIGN KEY and CHECK constraints are supported.
Joins and Subquery are not supported.	Joins and Subquery are supported.

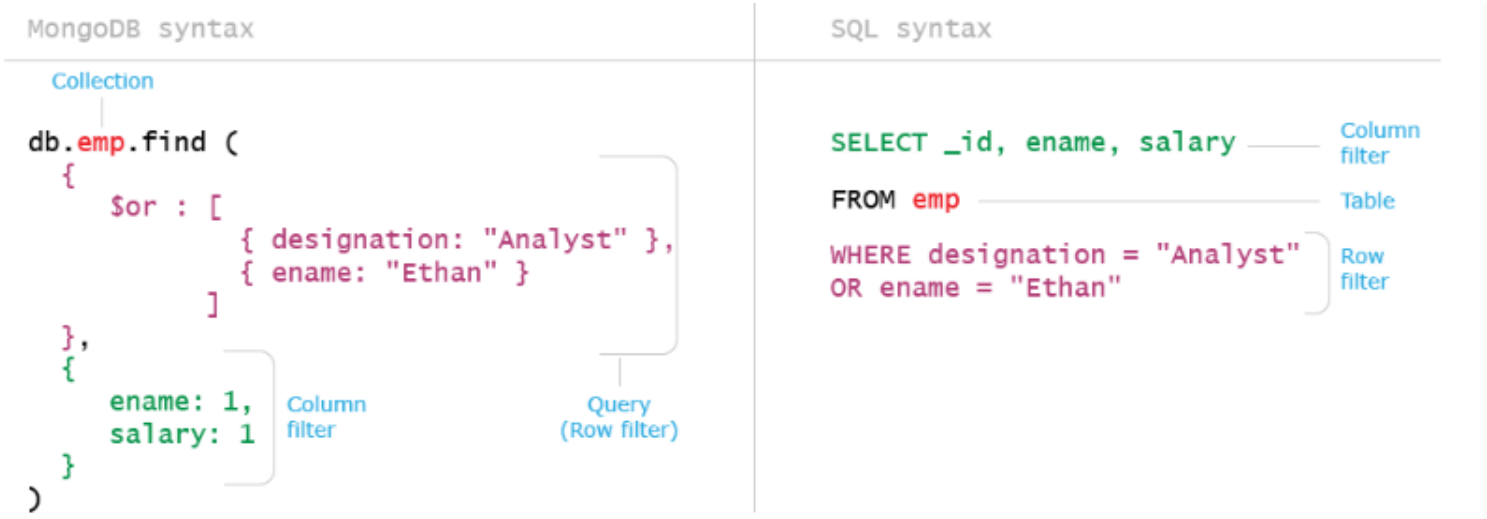
1. Insertion:-

Notes By: -
Lokesh Bagora

Reference: - Infytq Resources

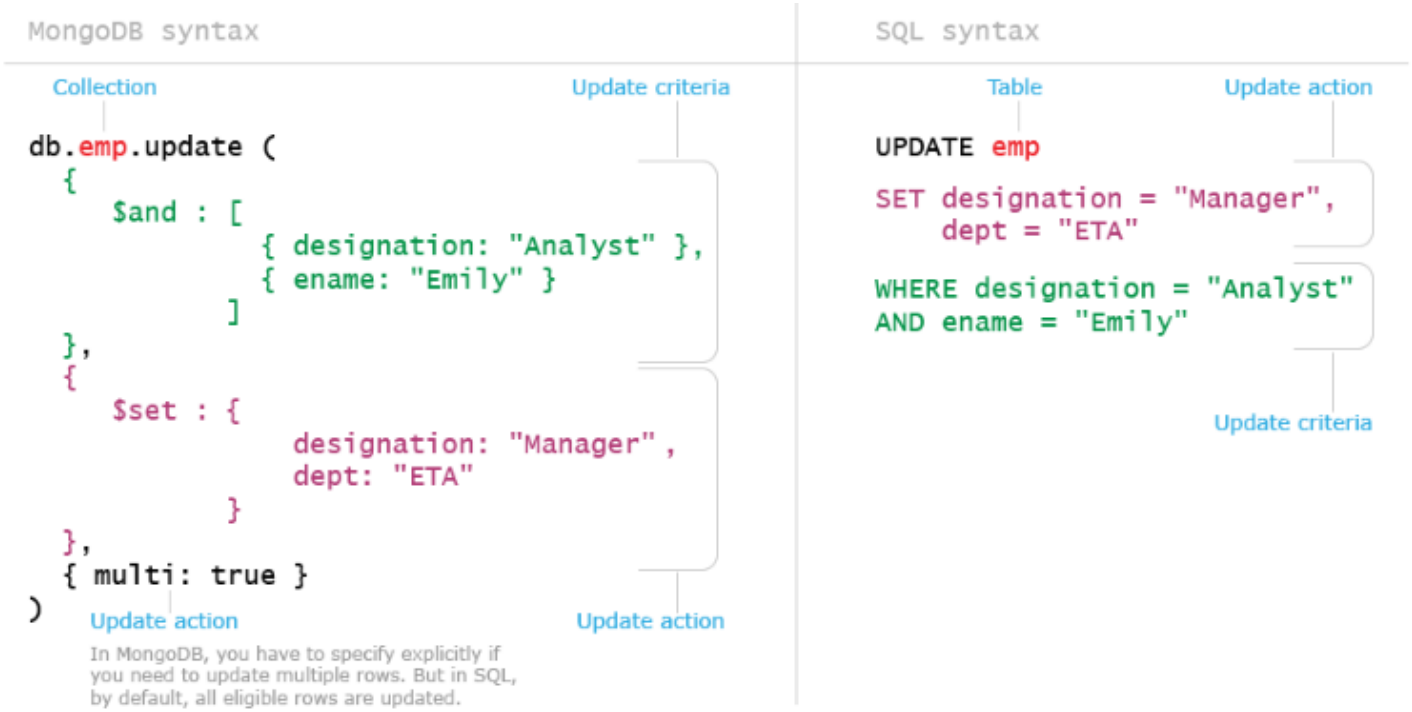


2. Find:-



Notes By: -
Lokesh Bagora **Reference: - Infytq Resources**

3. Update:-



4. Remove:-

Notes By: -
Lokesh Bagora

Reference: - Infytq Resources