[Team]
Team LinkedIn Account
[Ahmed M.Osman] [Link]
[Khaled Abdelrahman] [Link]
[Mohamed Adel] [Link]
[Ibrahim Hafez] [Link]

A) THEORETICAL

Come up with a scenario where using Nosql databases makes more sense.

There are few reasons why e-commerce leaders choose NoSQL databases so why?

- More customer are going online

Nowadays people spend a lot of their time in the internet. What it means for you as store owner? Of course, a lot. It increases chances that people would find your Grand Node store and then they will consider your store as a trustworthy and will buy your products. That trend is also closely linked to the mobile popularity.

Better flexibility and agility

NoSQL products support a whole range of new data types, and this is a major area of innovation in NoSQL. We have: column-oriented, graph, advanced data structures, document-oriented, and key-value. Complex objects can be easily stored without a lot of mapping. Developers love avoiding complex schemas and ORM frameworks. Lack of structure allows for much more flexibility. We also have program and programmer friendly compatible datatypes likes JSON.

Less complexity

NoSQL databases are used to complex applications. Why is this happening? It's simple. High performance e-commerce applications on a relational database like MsSQL require large amount of add ones, components and resources. In the case of MsSQL it generates large amount of money invested in application development. In MongoDB it's

much easier, you don't have to buy any specified additional components or extensions and for sure as earlier mentioned it's cheaper. Treating Grand Node as a machine to fully exploit the potential of Real Time Marketing, usage of Big Data the choice was easy - MongoDB.

Content Management

Decades ago, websites could be built with just static text. However, with the constant evolution of the technology industry, creating a website in such a manner will be considered outdated. Today, a website must have videos, audio, an array of text, and social media to catch its user's attention. Building quality websites on a Relational Database is challenging. Thankfully, MongoDB provides compatibility and access to such content on a single database. This is because it supports several unstructured and structured data.

Internet of Things (IoT)

IoT is one of the most appreciated technological innovations in the world today, connecting billions of devices globally. With IoT, companies improve productivity, redefine their revenue models, and leverage operational efficiency. MongoDB helps to maximize the full potential of IoT devices. Its intelligent Data Platform speeds up the operation and delivery of IoT devices.

Mobile Apps With Huge Numbers Of Users

Mobile phone and tablet use recently surpassed desktops as the top online platform for searching, shopping and otherwise viewing web content. Interestingly, as much of 90% of mobile data is served via apps and only 10% through browsers, an overwhelming shift in recent years.

Rapidly scaling mobile apps globally to serve tens of millions of users with acceptable performance (think mobile gaming or popular social media apps) often calls for distributed databases, which in turn calls for NoSQL. Flexible NoSQL data models also support rapid app update cycles better than relational data models in many cases.

For these reasons, more and more businesses looking to monetize web content are using NoSQL data stores for their apps. A popular case in point is The Weather Channel, whose MongoDB database instance handles millions of requests per minute while also processing user data and juggling weather updates from tens of thousands of worldwide locations.

PRACTICAL

Create a migration file with the required SQL statements for creating the tables.

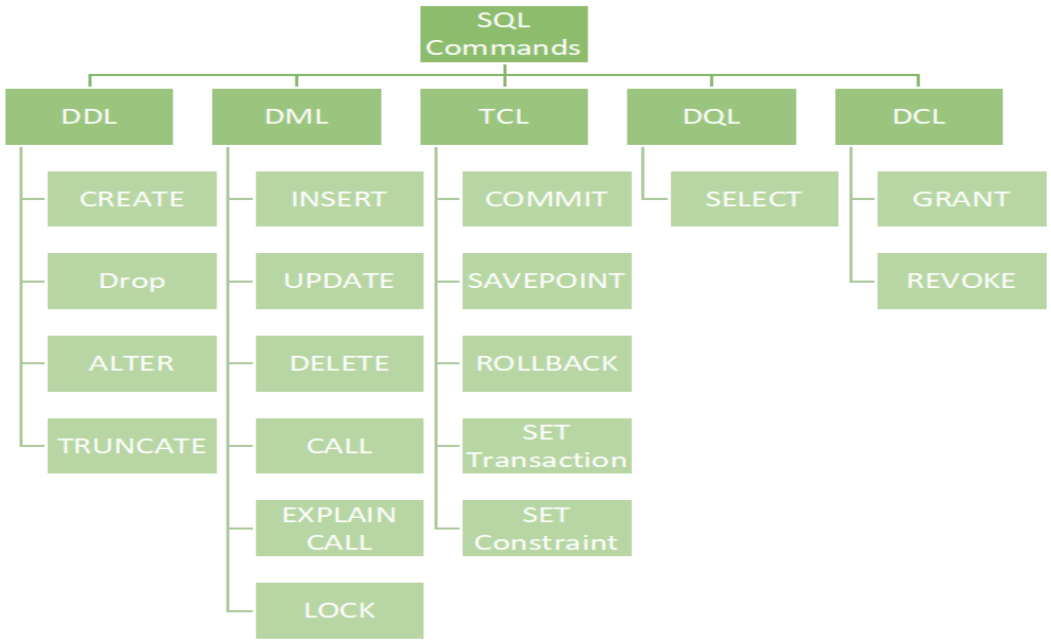
What's is SQL?

SQL stands for Structured Query Language and it is an ANSI standard computer language for accessing and manipulating database systems. It is used for managing data in relational database management system which stores data in the form of tables and relationship between data is also stored in the form of tables. SQL statements are used to retrieve and update data in a database.

Structured Query Language (SQL) as we all know is the database language by the use of which we can perform certain operations on the existing database and also we can use this language to create a database. SQL uses certain commands like Create, Drop, Insert, etc. to carry out the required tasks.

These SQL commands are mainly categorized into four categories as:

1. ***DDL – Data Definition Language***
2. ***DQl – Data Query Language***
3. ***DML – Data Manipulation Language***
4. ***DCL – Data Control Language***



DDL (Data Definition Language):

DDL or Data Definition Language actually consists of the SQL commands that can be used to define the database schema. It simply deals with descriptions of the database schema and is used to create and modify the structure of database objects in the database.DDL is a set of SQL commands used to create, modify, and delete database structures but not data. These commands are normally not used by a general user, who should be accessing the database via an application.
List of DDL commands:

- **CREATE**: This command is used to create the database or its objects (like table, index, function, views, store procedure, and triggers).
- **DROP**: This command is used to delete objects from the database.
- **ALTER:** This is used to alter the structure of the database.
- **TRUNCATE:** This is used to remove all records from a table, including all spaces allocated for the records are removed.
- **COMMENT**: This is used to add comments to the data dictionary.
- **RENAME:** This is used to rename an object existing in the database.

DQL (Data Query Language):

DQL statements are used for performing queries on the data within schema objects. The purpose of the DQL Command is to get some schema relation based on the query passed to it. We can define DQL as follows it is a component of SQL statement that allows getting data from the database and imposing order upon it. It includes the SELECT statement. This command allows getting the data out of the database to perform operations with it. When a SELECT is fired against a table or tables the result is compiled into a further temporary table, which is displayed or perhaps received by the program i.e. a front-end.
List of DQL:

- **SELECT:** It is used to retrieve data from the database.

DML(Data Manipulation Language):

The SQL commands that deals with the manipulation of data present in the database belong to DML or Data Manipulation Language and this includes most of the SQL statements. It is the component of the SQL statement that controls access to data and to the database. Basically, DCL statements are grouped with DML statements.

List of DML commands:

- **INSERT** : It is used to insert data into a table.
- **UPDATE:** It is used to update existing data within a table.
- **DELETE** : It is used to delete records from a database table.
- **LOCK:** Table control concurrency.
- **CALL:** Call a PL/SQL or JAVA subprogram.
- **EXPLAIN PLAN:** It describes the access path to data.

DCL (Data Control Language):

DCL includes commands such as GRANT and REVOKE which mainly deal with the rights, permissions, and other controls of the database system.

List of DCL commands:

- **GRANT:** This command gives users access privileges to the database.
- **REVOKE:** This command withdraws the user's access privileges given by using the GRANT command.

TCL (Transaction Control Language):

Though many resources claim there to be another category of SQL clauses TCL – Transaction Control Language. So we will see in detail about TCL as well. TCL commands deal with the transaction within the database.
List of TCL commands:

- **COMMIT:** Commits a Transaction.
- **ROLLBACK:** Rollbacks a transaction in case of any error occurs.
- **SAVEPOINT:**Sets a savepoint within a transaction.
- **SET TRANSACTION:** Specify characteristics for the transaction.

MERGE Statement in SQL Explained:

Prerequisite – MERGE Statement
As MERGE statement in SQL, as discussed before in the previous post, is the combination of three INSERT, DELETE and UPDATE statements. So if there is a **Source table** and a **Target table** that are to be merged, then with the help of MERGE statement, all the three operations (INSERT, UPDATE, DELETE) can be performed at once.

A simple example will clarify the use of MERGE Statement.

Example:
Suppose there are two tables:

- **PRODUCT_LIST** which is the table that contains the current details about the products available with fields P_ID, P_NAME, and P_PRICE corresponding to the ID, name and price of each product.
- **UPDATED_LIST** which is the table that contains the new details about the products available with fields P_ID, P_NAME, and P_PRICE corresponding to the ID, name and price of each product.

PRODUCT_LIST

P_ID	P_NAME	P_PRICE
101	TEA	10.00
102	COFFEE	15.00
103	BISCUIT	20.00

UPDATED_LIST

P_ID	P_NAME	P_PRICE
101	TEA	10.00
102	COFFEE	25.00
104	CHIPS	22.00

SQL | MERGE Statement:

Prerequisite – INSERT, UPDATE, DELETE
The **MERGE** command in SQL is actually a combination of three SQL statements: **INSERT, UPDATE and DELETE**. In simple words, the MERGE statement in SQL provides a convenient way to perform all these three operations together which can be very helpful when it comes to handle the large running databases. But unlike INSERT, UPDATE and DELETE statements MERGE statement requires a source table to perform these operations on the required table which is called as target table.
Now we know that the MERGE in SQL requires two tables : one the target table on which we want to perform INSERT, UPDATE and DELETE operations, and the other one is source table which contains the new modified and correct data for target table and is actually compared with the actual target table in order to modify it.

In other words, the MERGE statement in SQL basically merges data from a source result set to a target table based on a condition that is specified. The syntax of MERGE statement can be complex to understand at first but its very easy once you know what it means.So,not to get confused first let’s discuss some basics. Suppose you have two tables: source and target, now think if you

want to make changes in the required target table with the help of provided source table which consists of latest details.

SQL | DROP, TRUNCATE:

DROP is used to delete a whole database or just a table.The DROP statement destroys the objects like an existing database, table, index, or view.
A DROP statement in SQL removes a component from a relational database management system (RDBMS).

TRUNCATE

TRUNCATE statement is a Data Definition Language (DDL) operation that is used to mark the extents of a table for deallocation (empty for reuse). The result of this operation quickly removes all data from a table, typically bypassing a number of integrity enforcing mechanisms. It was officially introduced in the SQL:2008 standard.

The TRUNCATE TABLE mytable statement is logically (though not physically) equivalent to the DELETE FROM mytable statement (without a WHERE clause).

```
TRUNCATE TABLE  table_name;
table_name: Name of the table to be truncated.
DATABASE name - student_data
```

DROP vs TRUNCATE

- **Truncate is normally ultra-fast and its ideal for deleting data from a temporary table.**
- **Truncate preserves the structure of the table for future use, unlike drop table where the table is deleted with its full structure.**
- **Table or Database deletion using DROP statement cannot be rolled back, so it must be used wisely.**

Student

ROLL_NO	NAME	ADDRESS	PHONE	Age
1	Ram	Delhi	xxxxxxxxxx	18
2	RAMESH	GURGAON	xxxxxxxxxx	18
3	SUJIT	ROHTAK	xxxxxxxxxx	20
4	SURESH	Delhi	xxxxxxxxxx	18
3	SUJIT	ROHTAK	xxxxxxxxxx	20
2	RAMESH	GURGAON	xxxxxxxxxx	18

Student_Details

ROLL_NO	Branch	Grade
1	Information Technology	O
2	Computer Science	E
3	Computer Science	O
4	Mechanical Engineering	A

- **To delete the whole database**
- `DROP DATABASE student_data;`

  After running the above query whole database will be deleted.

- **To truncate Student_details table from student_data database.**
- `TRUNCATE TABLE Student_details;`

  After running the above query Student_details table will be truncated, i.e, the data will be deleted but the structure will remain in the memory for further operations.

Commened used :

```
psql -U postgres

CREATE DATABASE Inventory;

psql -U postgres -b Inventory

CREATE TABLE Categories(ID INT PRIMARY KEY,Name CHAR(50));

CREATE TABLE subCategories(ID INT PRIMARY KEY,Name CHAR(50),CategoryId INT, FOREIGN KEY (CategoryId) REFERENCES Categories(ID));

CREATE TABLE Products(ID INT PRIMARY KEY,Name CHAR(50),subCategoryId INT,Price INT , FOREIGN KEY (subCategoryId) REFERENCES subCategories(ID));

CREATE TABLE Customers(ID INT PRIMARY KEY,Name CHAR(50));

CREATE TABLE Orders(ID INT PRIMARY KEY,CustomerId INT,ProductId INT , FOREIGN KEY (ProductId) REFERENCES Product(ID), FOREIGN KEY (CustomerId) REFERENCES Customer(ID));


INSERT INTO Customers(ID,Name) VALUES(1,'Mohamed');

INSERT INTO Customers(ID,Name) VALUES(2,'Ibrahim');

INSERT INTO Customers(ID,Name) VALUES(3,'Mostafa');


INSERT INTO Categories(ID,Name) VALUES(1,'Stationery');

INSERT INTO Categories(ID,Name) VALUES(2,'Cleaning tools');

INSERT INTO Categories(ID,Name) VALUES(3,'Document photocopying');


INSERT INTO subCategories(ID,Name,CategoryId) VALUES(1,'NoteBook',1);

INSERT INTO subCategories(ID,Name,CategoryId) VALUES(2,'Dettol',2);

INSERT INTO subCategories(ID,Name,CategoryId) VALUES(3,'color photocopying',3);


INSERT INTO Products(ID,Name,Price,subCategoryId) VALUES(1,'NoteBook 50 paper',20,1);

INSERT INTO Products(ID,Name,Price,subCategoryId) VALUES(2,'Dettol 50gm',50,2);

INSERT INTO Products(ID,Name,Price,subCategoryId) VALUES(3,'color photocopying 100 paper',100,3);
```

```
INSERT INTO Orders(ID,CustomerId,ProductId) VALUES(1,1,1);
INSERT INTO Orders(ID,CustomerId,ProductId) VALUES(2,2,2);
INSERT INTO Orders(ID,CustomerId,ProductId) VALUES(3,3,3);
```