

Module 4 – Introduction to DBMS (Theory Questions)

Shivam Gangwani

June 27, 2025

1 Introduction to SQL

1.1 What is SQL, and why is it essential in database management?

SQL (Structured Query Language) is used to store, retrieve, and modify data in a Relational Database Management System (RDBMS) such as MySQL, SQL server, PostgreSQL, etc, using specific commands.

Importance of SQL in database management:

1. It helps read, write and manipulate all or specific data in RDBMS.
2. SQL's code is standardized by ISO and ANSI making it portable to different database platforms.
3. It also allows link different tables and secure specific data by allowing applying access restriction to specific users.
4. SQL supports indexing making searching data quicker.
5. It also enables automatic logging by using triggers and potentially help lessen length by using procedures which can be used multiple times throughout the code.
6. SQL can also be integrated with various programming languages enabling to make data-driven applications.
7. SQL's syntax is quite intuitive and have a big community support making it easier to learn.

1.2 Explain the difference between DBMS and RDBMS.

DBMS	RDBMS
Data is stored in files	Data is stored in tables
No explicit relationships between data	Explicit relationship between data is possible

Does not support normalization, potentially increasing data redundancy	Supports normalization, potentially reduces data redundancy
Does not support distributed database	Supports distributed database
Less security and access control features	More security and access control features
Does not support multiple users	Supports multiple users
Lower hardware and software requirements	Higher hardware and software requirements
Eg. XML, window registry, forxpro, etc.	Eg. MySQL, SQL server, Oracle, etc.

1.3 Describe the role of SQL in managing relational databases.

1. It is used to define structure, organization and relation of the stored data.
2. It is also used to retrieve stored data from a database.
3. It is also used to update data stored in database by enabling adding, removing, or modifying stored data.
4. SQL supports access control enabling restricted access to specified data for specified users.
5. Due to support for integrity constraints in SQL, it is used to prevent data corruption due to system failure.

1.4 What are the key features of SQL?

1. It supports adding, modifying and deleting data, the structure it is stored in, relation between them, and databases.
2. It also supports retrieving specific data which meet specified conditions and ability to sort it.
3. SQL supports transaction control allowing to revert back to specified point.
4. SQL also supports restricted data input allowing only particular data to be entered in a given column.
5. SQL lets make different users and giving each individual access to particular databases and perform certain actions on it.
6. It is a standardized language making it so it can be written for one database management system and used in a different one without making a lot of changes.

2 SQL Syntax

2.1 What are the basic components of SQL syntax?

1. Databases
2. Tables

3. Queries
4. Constraints
5. Stored procedures
6. Transactions

2.2 Write the general structure of an SQL **SELECT** statement.

```
1 select column1, column2, ...  
2 from table_name;
```

2.3 Explain the role of clauses in SQL statements.

Clauses in SQL statements enables various features of SQL such as ordering, filtering, or grouping of data, etc. They help user to communicate to the computer what and how it wants the data from the database.

3 SQL Constraints

3.1 What are constraints in SQL? List and explain the different types of constraints.

Constraints in SQL are used to enforce rules on data stored in a database.

Different types of constraints:

1. **not null**: used to make sure some data is added in the specified column.
2. **unique**: used to restrict repeated entry of data.
3. **primary key**: used to assign a unique id to each row.
4. **foreign key**: used to link tables to each other with a shared column between them.
5. **check**: used to check if the values in the specified column satisfies a specified condition.
6. **default**: used to set a specified value to all entries in the specified column.

3.2 How do **PRIMARY KEY** and **FOREIGN KEY** constraints differ?

PRIMARY KEY	FOREIGN KEY
used to uniquely identify each row of a table	used to link tables with same columns
a table can have only one primary key	a table can have multiple foreign keys
restricts the column to have duplicate values in different rows of the column	allows the columns to have duplicate values in different rows of the columns

Does not support distributed database	Supports distributed database
Less security and access control features	More security and access control features
Does not support multiple users	Supports multiple users
Lower hardware and software requirements	Higher hardware and software requirements
Eg. XML, window registry, forxpro, etc.	Eg. MySQL, SQL server, Oracle, etc.

3.3 What is the role of **NOT NULL** and **UNIQUE** constraints?

1. **NOT NULL:** it is used to make sure values in the specified column are not empty.
2. **UNIQUE:** it is used to make sure values in the specified column are different from each other and there are no duplicates in the column.

4 Main SQL Commands and Sub-commands (DDL)

4.1 Define the SQL Data Definition Language (DDL).

Data Definition Language (DDL) is used to create and modify data in SQL such as table, database, rows, etc.

4.2 Explain the **CREATE** command and its syntax.

It is used to create and define new database objects in SQL such as database, table, view, etc.

General syntax:

```
1 create [database_object] [database_object_name];
```

4.3 What is the purpose of specifying data types and constraints during table creation?

The purpose of specifying data types during table creation is to define what kind of data will be stored in that column and to allocate appropriate amount of memory. This ensures that if a different type of data is tried to be entered, compiler gives error. It also helps compiler to perform appropriate operations based on the data type.

The purpose of specifying constraints during table creation is to enforce rules on a specific column or table to ensure data consistency and integrity or restrict data that meet certain condition.

5 ALTER Command

5.1 What is the use of the **ALTER** command in SQL?

ALTER command in SQL is used to make changes to the structure of a table that already exists.

5.2 How can you add, modify, and drop columns from a table using **ALTER**?

1. Add columns:

```
1 alter table [table_name]
2 add [column_1_name] datatype_1,
3 add [column_2_name] datatype_2,
4 ...;
```

2. Modify columns:

```
1 alter table [table_name]
2 modify column [column_1_name] datatype_1,
3 modify column [column_2_name] datatype_2,
4 ...;
```

3. Drop columns:

```
1 alter table [table_name]
2 drop column [column_name_1],
3 drop column [column_name_2],
4 ...;
```

6 DROP Command

6.1 What is the function of the **DROP** command in SQL?

The DROP command in SQL is used to permanently delete entire database objects such as database, table, constraint, index, etc.

6.2 What are the implications of dropping a table from a database?

When a table is dropped from a database, all the data stored in it, the structure of and constraints on the columns, triggers associated to it, access privileges granted to the table, and its indexes are permanently deleted.

7 Data Manipulation Language (DML)

7.1 Define the **INSERT**, **UPDATE**, and **DELETE** commands in SQL.

1. **INSERT**: it is used to add new rows to a table.
2. **UPDATE**: it is used to change existing data in a table.
3. **DELETE**: it is used rows from a table.

7.2 What is the importance of the **WHERE** clause in **UPDATE** and **DELETE** operations?

The importance of the **WHERE** clause in **UPDATE** and **DELETE** operations is that it helps ensure only those rows get updated or deleted which satisfy a particular condition. Without the **WHERE** clause, all the rows of the table will be updated or deleted.

8 Data Query Language (DQL)

8.1 What is the **SELECT** statement, and how is it used to query data?

The **SELECT** statement is used to get all or selected data from one or multiple table with all or specified columns with options to group, sort, or filter the data.

```
1 select [column_1], [column_2], ...  
2 from table_name;
```

8.2 Explain the use of the **ORDER BY** and **WHERE** clauses in SQL queries.

The **ORDER BY** clause in SQL queries is used to view selected data with a particular column sorted by either ascending or descending order.

```
1 select [column_1], [column_2], ...  
2 from table_name  
3 order by [column_name] ASC/DESC;
```

The **WHERE** clause in SQL queries is used to show only that data which satisfies a given condition.

```
1 select [column_1], [column_2], ...  
2 from table_name  
3 where [condition];
```

9 Data Control Language (DCL)

9.1 What is the purpose of **GRANT** and **REVOKE** in SQL?

GRANT in SQL is used to grant a user access to a database object, whereas REVOKE is used to revoke access.

```
1 grant/revoke [privilege_name]
2 on [object_name]
3 to [user_name/public/role_name];
```

9.2 How do you manage privileges using these commands?

10 Transaction Control Language (TCL)

10.1 What is the purpose of the **COMMIT** and **ROLLBACK** commands in SQL?

COMMIT and ROLLBACK commands in SQL are used in transactions for data integrity and consistency.

COMMIT is used to save all the changes in a transaction performed after last COMMIT or ROLLBACK.

```
1 start transaction;
2 [1st operation];
3 [2nd operation];
4 ...
5 commit;
```

Whereas ROLLBACK is used in the same way but to revert the changes.

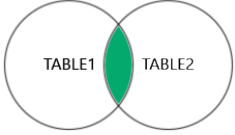
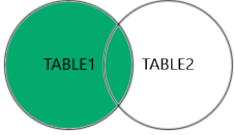
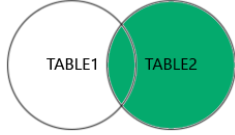
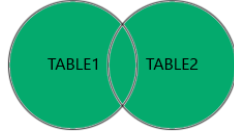
```
1 start transaction;
2 [1st operation];
3 [2nd operation];
4 ...
5 rollback;
```

10.2 Explain how transactions are managed in SQL databases.

11 SQL Joins

11.1 Explain the concept of **JOIN** in SQL. What is the difference between **INNER JOIN**, **LEFT JOIN**, **RIGHT JOIN**, and **FULL OUTER JOIN**?

JOIN in SQL is used to combine multiple tables based on a common column between two tables.

INNER JOIN	LEFT JOIN	RIGHT JOIN	FULL OUTER JOIN
Returns only rows which matching values in both tables.	Returns all rows in the left table, and the rows which match in the right.	Returns all rows in the left table, and the rows which match in the right.	Returns all rows from both tables which have a match.
			

11.2 How are joins used to combine data from multiple tables?

Joins are used in SQL to combine data from multiple tables by using JOIN clause in the SELECT statement and mentioning the two tables whose data is to be combined and the common columns between them based on which they are to be combined.

```

1 select *
2 from [table_1]
3 [inner/left/right/full] join [table_2]
4 on [table_1].[common_column_name] = [table_2].[common_column_name];

```

12 SQL Group By

12.1 What is the GROUP BY clause in SQL? How is it used with aggregate functions?

The GROUP BY clause in SQL is used to group rows in a table by the values that are common in a specified column.

```

1 select [column_1], [column_2], ...
2 from [table_name]
3 group by [column_a], [column_b], ...;

```

The GROUP BY clause in SQL is used with aggregate functions, they operate on each individual group created by the GROUP BY and give the value of the respective operation for each of them.

```

1 select [column_1], [aggregate_funciton]([column_2]), ...
2 from [table_name]
3 group by [column_a], [column_b], ...;

```

12.2 Explain the difference between GROUP BY and ORDER BY.

GROUP BY	ORDER BY
----------	----------

It is used to group rows with same values in a specified column.	It is used to sort rows in ascending or descending order based on the values in a specified column.
Allowed in CREATE VIEW statement.	Not allowed in CREATE VIEW statement.
Used before ORDER BY in SELECT statement.	Used after GROUP BY in SELECT statement.
Controls the presentation of the rows.	Controls the presentation of the columns.
Aggregate functions are mandatory.	Aggregate functions are not mandatory.

13 SQL Stored Procedure

13.1 What is a stored procedure in SQL, and how does it differ from a standard SQL query?

A stored procedure in SQL is a pre-compiled block of SQL statements that are stored in the database and can be executed anytime by calling it without having to rewriting all the statements in the procedure.

- A stored procedure can be written once and used multiple times, eliminating the need to rewrite all the SQL queries multiple times to perform the same task.
- It is faster than a standard SQL query as it is precompiled and stored in the database, resulting in reduced parsing time.
- It is also more secure, as a user can be granted access to just the procedure instead of the table to perform operations defined in the procedure.
- A stored procedure occupy disk space while a standard SQL statement only exists on memory when it is executed.
- As a stored procedure is encapsulated, it is more difficult to test it as compared to a standard SQL query.
- It is not possible to debug or version control it unlike a standard SQL query.

13.2 Explain the advantages of using stored procedures.

- They are faster and more efficient as they only need to be compiled once and can be executed multiple times.
- They also increase productivity as the same code does not have to be written multiple times, resulting in decreased development and maintenance time.
- As the commands in a stored procedure are executed as a single block of code, it helps reduce server/client traffic.
- A user can only be given permission to a stored procedure but not the table/s it performs operations on, allowing the user to only perform operations defined in the stored procedure make the data more secured.

14 SQL View

14.1 What is a view in SQL, and how is it different from a table?

A view is a virtual table whose columns and data are chosen from preexisting table/s using SELECT query.

View	Table
It is Virtual table.	It is physical object.
Dependent on other table/s.	Independent of other table/s.
Do not occupy space on storage.	Do occupy space on storage.
Views are read-only.	CRUD operations can be performed on tables.
Can aggregate data from multiple tables.	Cannot aggregate data from multiple tables.

14.2 Explain the advantages of using views in SQL databases.

- View is more as a user can be given access to a part of the data instead of the entire database or a table.
- View can be used to combine data from multiple tables with multiple complex queries combined in a single query.
- Changing the structure of the table does not affect the views associated with it, which ensures the structure is not affected by the changes to the table.

15 SQL Triggers

15.1 What is a trigger in SQL? Describe its types and when they are used.

A trigger in SQL is a stored procedure in a database which can be set to be executed automatically whenever data is added, updated, or deleted in the database.

Types of trigger in SQL:

1. **DDL trigger:** This type of trigger gets activated when a database object is created, deleted, or changed. It is used to keep track of changes made to the database.
2. **DML trigger:** This type of trigger gets activated when data is added, deleted, or changed. It is used to keep track of changes made to a table.
3. **Logon trigger:** This trigger gets activated when a users logs into the database. It is used to keep track of users logging in to the database.

15.2 Explain the difference between **INSERT**, **UPDATE**, and **DELETE** triggers.

The key difference between **INSERT**, **UPDATE**, and **DELETE** triggers is that **INSERT** is executed when a new data row is added to the specified table, **UPDATE** is executed when existing data in the specified table is changed, and **DELETE** is executed when a data row/s is/are removed from the specified table.

16 Introduction to PL/SQL

16.1 What is PL/SQL, and how does it extend SQL's capabilities?

PL/SQL (Procedural Language/Sequence Query Language) is a combination of SQL and procedural programming.

PL/SQL extends SQL's capabilities by adding the ability to have loops (allowing execution of block of SQL statements multiple times), conditions (allowing execution of block of SQL statements only when given condition/s is/are satisfied) and exception handling (allowing the rest of the statements to be executed after error is detected).

16.2 List and explain the benefits of using PL/SQL.

- 1. Tight integration with SQL:** All SQL operations can be performed using PL/SQL as it supports all of its queries and data types removing the need to switch between both languages.
- 2. High performance:** PL/SQL sends statements in the form of blocks reducing traffic. Also, SQL statements in PL/SQL can be reused, in turn improving performance.

17 PL/SQL Control Structures

- 17.1 What are control structures in PL/SQL? Explain the **IF-THEN** and **LOOP** control structures.
- 17.2 How do control structures in PL/SQL help in writing complex queries?

18 SQL Cursors

- 18.1 What is a cursor in PL/SQL? Explain the difference between implicit and explicit cursors.
- 18.2 When would you use an explicit cursor over an implicit one?

19 Rollback and Commit Savepoint

- 19.1 Explain the concept of **SAVEPOINT** in transaction management. How do **ROLLBACK** and **COMMIT** interact with savepoints?

SAVEPOINT in transaction management is used to enable undoing of changes made by execution of SQL statements after a certain point in a transaction.

ROLLBACK to a **SAVEPOINT** undo all the changes made by running SQL statements after the specified **SAVEPOINT**.

COMMIT is used to save all the changes made by execution of SQL statements in a transaction, inevitably disabling the ability to undo any changes made by executing SQL statements after a **SAVEPOINT** in the transaction.

- 19.2 When is it useful to use savepoints in a database transaction?

Savepoints in a database transaction are used to separate it into smaller blocks for error recovery or for debugging/testing.