Aggregate and atomicity - QUESTIONS

1. What is an aggregate function in sql? give any example

An aggregate function in SQL returns one value after calculating multiple values of a column.

We often use aggregate functions with the GROUP BY and HAVING clauses of the SELECT statement.

EXAMPLES-:

There are 5 types of SQL aggregate functions:

Count()

Sum()

Avg()

Min()

Max()

2. How can you use the GROUP BY clause in combination with aggregate funtions?

The Group By statement is used to group together any rows of a column with the same value stored in them, based on a function specified in the statement

3.Describe a scenario where automicity is crucial for database operations

a customer transfers money from one account to another. It is is the scenario of automicity

OLAP AND OLTP-Questions

1. Mention any 2 of the difference between OLAP and OLTP

OLAP systems to generate reports, perform complex data analysis, and identify trends. In contrast, you use OLTP systems to process orders, update inventory, and manage customer accounts.

Other major differences include data formatting, data architecture, performance, and requirements.

2. How to do you optimize an OLTP database for better performance? HINT:index

SELECT c.ClaimNumber, a.ItemDate, c.DTN, b.FilePath

FROM items a,

itempages b,

keygroupdata c

WHERE a.ItemType IN (112,115,189,241)

AND a.ItemNum = b.ItemNum

AND b.ItemNum = c.ItemNum

ORDER BY a.DateStored DESC

Data Encryption And Storage- Questions

1.What are the diffence types of data encrption available in MSSQ;?

SSL Transport Encryption. ...

SQL Server Transparent Data Encryption (TDE) ...

Backup Encryption. ...

Column/Cell-Level Encryption. ...

Always Encrypted.

these are the different types of data encrption available in MSSQ

SL, NOSQL, APPLICATION,EMBEDDED - Questions

1. What is the man difference between SQL and NOSQL databases?

SQL databases are relational

NoSQL databases are non-

these are the man difference between sql and nosql

DDL-Questions

1.How do you create a new schema in MSSQL?

Right-click the Security folder, point to New, and select Schema.

New dialog box, on the General page, enter a name for the new schema in the Schema name

this is how we create a new schema in MSSQL

2.Describe the process of altering an existing table?

irstly select the table with the ALTER TABLE command and then define the column name and the data type of that column.

this is h process of altering an existing table

3.What is the difference between a VIEW and a TABLE in MSSQL?

A view is a database object that allows generating a logical subset of data from one or more tables.

A table is a database object or an entity that stores the data of a database.

these are the difference betwwen a VIEW and a TABLE in MSSQL

4.Explain how to create and manage indexes in a table.

Create a table index

Access the Index creator in one of the following ways. Option. ...

Select the fields you want included in the index. The order in which you select the fields affects how the index works. ...

To create a unique index, select the Unique Index check box.

Select Create Index.

this is how to create and manage indexes in a table

DML

1. What are the most commonly used DML commands?

SELECT , INSERT , DELETE , and UPDATE .

these are the most commonly used DML commands

2.how do you retrieve data from multiple table using a JOIN?

utilizing JOIN clauses to combine data from different tables based on specified conditions

this is how we retrive data from multiple table using a JOIN

3.Explain how relational aigebra is used in SQL queries

Relational algebra refers to a procedural query language that takes relation instances as input and returns relation instances as output.

this how relational aigebrA is used in sql queries.

4.What are the implications of using complex queries in terms of performance?

Due to their complexity, they may not be using the most efficient paths to retrieve data, leading to slow performance.

This directly impacts database performance, as complex queries can consume significant resources, slowing down the database for all users.

these are the implications of using coplex queries in terms of performance

AGGREGATE FUNCTIONS -Questions

1. How does the HAVING clause differ form the WHERE caluse when using aggergate function could use having before group by in the select statement?

the HAVING clause allows you to filter data from a group of rows in a query based on conditions involving aggregate values.

Filters- Questions

1.What are the filters in SQL and how are they used in queries?

SQL filters are text strings that you use to specify a subset of the data items in an internal or SQL database data type.

2.How do you use the WHERE caluse to filter data in MSSQL?

SELECT column\_name(s) FROM table\_name WHERE condition;

3. How can you combain multiple filter conditions using logical operators?

AND and OR are two (logical) operators used to combine multiple filter conditions.

When applied, the AND operator returns only those observations that satisfy all filter conditions. On the other hand, the OR operator returns only those cases that satisfy at least one condition.

4.Explain the use of CASE statements for filtering data in a query

CASE statements allow you to categorize data that you're interested in -- and exclude data you're not interested in.

Operators- Questions

1.what are filters in SQL and how are they used in queries?

FILTER is a modifier used on an aggregate function to limit the values used in an aggregation.

2.How do arithmetic operators in SQL?

Arithmetic operators can be used in the SELECT clause as a way of combining the values in different columns and presenting the results as a new column.

3.Explain the use of LIKE operator with wildcards for pattern matching.

The LIKE operator is used in a WHERE clause to search for a specified pattern in a column.

There are two wildcards often used in conjunction with the LIKE operator: The percent sign % represents zero, one, or multiple characters.

The underscore sign \_ represents one, single character.

MultipleTables:Normalization

1.what is normalization and why is it important?

The main use of normalization is to utilize in order to remove anomalies that are caused because of the transitive dependency.

2.describe the basic normal forms Hint:1NF 2NF NF

he first normal form helps to eliminate duplicate data and simplify queries.

Second Normal Form (2NF): 2NF eliminates redundant data by requiring that each non-key attribute be dependent on the primary key.6 days ago

3.Mention anyone impact to normalization on database performance.

Normalization breaks down large tables into smaller, more manageable ones, reducing redundancy and ensuring efficient data organization.

Indexes & Contraints

1. What are indexes and why are they used?

ndexes contain all the necessary information needed to access items quickly and efficiently.

Indexes serve as lookup tables to efficiently store data for quicker retrieval.

Table keys are stored in indexes. Indexes for non-key values can be created with a CREATE INDEX statement.

2.How do yo create a unique constraint on a table column?

Create a unique constraint on a new table

In Object Explorer, connect to an instance of Database Engine.

On the Standard bar, select New Query.

Copy and paste the following example into the query window and select Execute.

The example creates a table and defines a unique constraint on the column TransactionID

3.Explain the diference between clustered and non-clustered indexes.

CLUSTERED

A clustered index is used to define the order or to sort the table or arrange the data by alphabetical order just like a dictionary.

NON-CLUSTERED

A non-clustered index collects the data at one place and records at another place.

4.How would you optimize index usage in a highly transactional database?

Identify key queries and columns. ...

Avoid over-indexing. ...

Regularly monitor and tune indexes. ...

Consider using covering indexes. ...

Monitor and manage index fragmentation. ...

Single-column indexes. ...

Composite indexes. ...

Unique indexes.

More items...

Joins-Questions

1. What are the different types of joins available in MSSQL?

There are four main types of JOINs in SQL:

INNER JOIN, OUTER JOIN, CROSS JOIN, and SELF JOIN.

2.provie an example of a LEFT JOIN query.

To perform left-join on these two tables we will use the following SQL query: SELECT Emp. EmpID, Emp.Name, department.

3. Explain the concept of a self-join and when it might be used.

A Self Join is a type of a JOIN query used to compare rows within the same table.

Unlike other SQL JOIN queries that join two or more tables, a self join joins a table to itself.

To use a self join, a table must have a unique identifier column, a parent column, and a child column.

4.How do you perform a full outer join and what is its significance?

An full outer join is a method of combining tables so that the result includes unmatched rows of both tables.

If you are joining two tables and want the result set to include unmatched rows from both tables, use a FULL OUTER JOIN clause.

The matching is based on the join condition.

Alias

1.What is an alise in SQL and how is it used?

SQL aliases are used to give a table, or a column in a table, a temporary name. Aliases are often used to make column names more readable

2.Give an example of using table aliases in a query.

SELECT id, CONCAT(firstname, ' ', lastname) AS fullname FROM students;

3.How do you use column aliases in conjuction with aggeregate functions?

select col1,count(col2) as col7

from --some join operation

group by col1

having col7 >= 3 -- replace col7 by count(col2) to make the code work

4.Explain the benefits of using aliases in complex queries.

Improves Readability: Simplifies table and column names, making queries easier to read.

Enhances Maintainability: Makes updates and changes more straightforward by reducing repetitive long names.

Reduces Typing: Shortens the amount of code written, speeding up query development.

Avoids Ambiguity: Clearly distinguishes columns from different tables, preventing errors.

Facilitates Complex Operations: Helps manage subqueries and nested joins more efficiently.

Joins vs SubQueries

1. What is he difference between joins and subquries?

Joins: Combine rows from two or more tables based on a related column, allowing you to retrieve and merge data from multiple sources in a single query.

They are typically used to get data from multiple tables in a flat, combined result set.

Subqueries: Nested queries within a larger query that can be used to filter, compute, or provide intermediate results.

They are often used to retrieve data that will then be used in the main query, or to perform operations where joins might be less effective.

2.When whould you prefer a subquery over a join?

Filtering Based on Aggregates:

When you need to filter records based on aggregate results (e.g., finding employees with salaries above the average), subqueries can simplify this.

Complex Conditions:

When dealing with complex conditions that involve multiple levels of filtering or calculations, subqueries can make the query more manageable.

Performance:

In some cases, subqueries may perform better than joins, especially when dealing with large datasets or when the subquery results are used to limit the rows processed in the main query.

3.Explain how correlated subqueries work with an example.

A correlated subquery is a type of subquery that references columns from the outer query. This means the subquery is executed once for each row processed by the outer query, and its result depends on the current row of the outer query.

Example:

Let's say we have two tables, employees and departments:

employees:

id

name

salary

department\_id

departments:

id

department\_name

We want to find employees who have a salary greater than the average salary of their own department. Here's how a correlated subquery could accomplish this:

sql

Copy code

SELECT e.name, e.salary

FROM employees e

WHERE e.salary > (

SELECT AVG(e2.salary)

FROM employees e2

WHERE e2.department\_id = e.department\_id

);

4.Discuss the performance implications of using joins vs subqueries.

Joins:

Efficiency: Joins are typically more efficient because they allow the database to optimize the query execution plan, often using indexing and other techniques to reduce the amount of data processed.

Optimization: Most modern database systems have advanced join optimization algorithms (like hash joins, merge joins) that can handle large datasets effectively.

Scalability: Joins generally scale better for larger datasets, especially when appropriate indexes are in place.

Execution: Subqueries can sometimes be less efficient because they may be executed multiple times, particularly if they are correlated (i.e., they depend on the outer query).

Optimization: While some databases optimize subqueries well, others might not, leading to potential performance issues, especially if the subquery is complex.

Readability: Subqueries can sometimes make SQL code harder to read and maintain, which indirectly impacts performance through longer development cycles or more error-prone code.

General Advice:

Indexes: Ensure that appropriate indexes are used to enhance performance for both joins and subqueries.

Query Plan: Examine the query execution plan to understand how the database handles the query and to identify potential bottlenecks.

Testing: Performance can vary greatly with different data volumes and database systems, so testing both approaches in your specific environment is often the best approach.

Types-Questions

1.What are the different data types available in MSSQL?

Microsoft SQL Server (MSSQL), there are various data types available to handle different kinds of data. Here’s a brief overview:

1. Numeric Types:

INT: Integer, typically 4 bytes.

BIGINT: Large integer, 8 bytes.

SMALLINT: Smaller integer, 2 bytes.

TINYINT: Very small integer, 1 byte.

DECIMAL(p, s): Fixed precision and scale numeric, where p is precision and s is scale.

NUMERIC(p, s): Similar to DECIMAL, used for exact numeric values.

FLOAT: Floating-point number, approximate precision.

REAL: Floating-point number with less precision than FLOAT.

2. Character Types:

CHAR(n): Fixed-length character string, n specifies the length.

VARCHAR(n): Variable-length character string, up to n characters.

TEXT: Variable-length text, deprecated in favor of VARCHAR(MAX).

3. Unicode Character Types:

NCHAR(n): Fixed-length Unicode character string, n specifies the length.

NVARCHAR(n): Variable-length Unicode character string, up to n characters.

NTEXT: Variable-length Unicode text, deprecated in favor of NVARCHAR(MAX).

4. Date and Time Types:

DATE: Date value (year, month, day).

TIME: Time value (hours, minutes, seconds).

DATETIME: Date and time value with accuracy to 1/300th of a second.

SMALLDATETIME: Date and time value with accuracy to the minute.

DATETIME2: Extended precision date and time value.

DATETIMEOFFSET: Date and time with time zone offset.

5. Binary Types:

BINARY(n): Fixed-length binary data, n specifies the length.

VARBINARY(n): Variable-length binary data, up to n bytes.

IMAGE: Variable-length binary data for large objects, deprecated in favor of VARBINARY(MAX).

6. Miscellaneous Types:

BIT: Integer with 1-bit field for binary values (0, 1).

UNIQUEIDENTIFIER: Globally unique identifier (GUID).

XML: XML data type for storing XML documents or fragments.

JSON: While not a native type, SQL Server can handle JSON data through NVARCHAR and specific functions.

7. Large Object Types:

VARCHAR(MAX): Variable-length string with a maximum size of 2^31-1 bytes.

NVARCHAR(MAX): Variable-length Unicode string with a maximum size of 2^31-1 bytes.

VARBINARY(MAX): Variable-length binary data with a maximum size of 2^31-1 bytes.

These types allow you to store and manipulate a wide range of data in SQL Server.

2.how do you choose the appropriate data type for a column?

Data Characteristics:

Type of Data: Use numeric types for numbers, character types for text, date types for dates, and binary types for binary data.

Size: Choose types based on the maximum expected size of the data. For example, use VARCHAR(n) for variable-length strings where n is the maximum length needed.

Storage Efficiency:

Precision: Use exact numeric types (e.g., DECIMAL) for precise calculations, and approximate types (e.g., FLOAT) for scientific calculations where exact precision is less critical.

Space: Use smaller types if possible (e.g., TINYINT vs. INT) to save space and improve performance.

Performance:

Indexes: Choose types that are compatible with indexing needs. For instance, INT is more efficient than VARCHAR for indexing.

Query Patterns: Use types that match how the data will be queried. For example, DATE for date comparisons instead of DATETIME if time information is not needed.

Data Integrity:

Constraints: Ensure the data type supports necessary constraints (e.g., UNIQUEIDENTIFIER for unique IDs, CHECK constraints for valid ranges).

Scalability:

Growth: Consider future growth. For example, use BIGINT if you expect values to exceed the range of INT.

Compatibility:

Applications: Ensure the data type is compatible with application requirements and external systems.

3.how do you handle data type conversions in queries?

CAST and CONVERT Functions:

CAST(expression AS data\_type): Converts an expression to the specified data type.

CONVERT(data\_type, expression [, style]): Converts an expression to a specified data type with optional formatting (style).

Example:

sql

Copy code

SELECT CAST(column\_name AS VARCHAR(50)) AS new\_column

FROM table\_name;

sql

Copy code

SELECT CONVERT(DATE, '2024-07-29', 120) AS formatted\_date

FROM table\_name;

Implicit Conversion:

SQL Server often handles simple conversions automatically (e.g., converting INT to VARCHAR in expressions). However, be cautious with implicit conversions as they might impact performance or lead to unexpected results.

Data Type Compatibility:

Ensure that conversions are meaningful. For instance, converting between numeric types may lead to precision loss, and converting non-numeric strings to numbers may fail.

Handling NULL Values:

Use functions like ISNULL(expression, replacement) or COALESCE(expression, replacement) to manage NULL values during conversions.

Example:

sql

Copy code

SELECT ISNULL(CAST(column\_name AS VARCHAR(50)), 'Default Value') AS new\_column

FROM table\_name;

Error Handling:

Correlation and Non-correlation

1. what is a correlation subquery?

A correlation subquery is a type of subquery that references columns from the outer query. Unlike non-correlated subqueries, which can be executed independently of the outer query, a correlated subquery depends on the values of the outer query's rows for its execution.

2.what is non-correlated subquery.HINT:Outer query checks the data from inner sbquery?

A non-correlated subquery is a type of subquery that operates independently of the outer query.

The outer query uses the result of the non-correlated subquery but does not influence its execution. This means the subquery can be run on its own without needing any values from the outer query.

3.Explain how correlated subqueries can affect query performance

Multiple Executions:

Performance Hit: A correlated subquery is executed once for each row processed by the outer query. This can lead to significant performance degradation if the outer query processes a large number of rows.

Increased Complexity:

Resource Usage: Each execution of the subquery may involve complex computations or data retrieval, increasing CPU and I/O usage, which can slow down the overall query performance.

Index Utilization:

Potential Issues: If the correlated subquery accesses columns that are not indexed or uses complex joins, it may not benefit from indexing, further affecting performance.

Query Optimization Challenges:

Less Optimization: Some database systems may struggle to optimize queries with correlated subqueries effectively, leading to less efficient execution plans compared to other approaches.

INTRODUCTION TO TSQL, PROCEDURES, FUNCTIONS, TRIGGERS, INDICES

1.Write a simple stored procedure to insert a new record into the employees table.

CREATE PROCEDURE InsertEmployee

@EmployeeID INT,

@Name NVARCHAR(100),

@Salary DECIMAL(10, 2),

@DepartmentID INT

AS

BEGIN

INSERT INTO Employees (EmployeeID, Name, Salary, DepartmentID)

VALUES (@EmployeeID, @Name, @Salary, @DepartmentID);

END;

Usage:

To call this stored procedure and insert a new record, use:

sql

Copy code

EXEC InsertEmployee

@EmployeeID = 1,

@Name = 'John Doe',

@Salary = 50000.00,

@DepartmentID = 10;

2.Decribe a scenario where you would use a tigger to enforce business rules.

Trigger Name: check\_balance\_before\_order

Event: BEFORE INSERT on the Orders table

Action: The trigger checks the AccountBalance of the user from the Users table.

Condition: If the balance is less than the required minimum, the trigger raises an error and prevents the order from being placed.

Security and Accessibility

1.Mention any 2 of the common security measures to product a sql server database?

Encryption: Encrypt sensitive data at rest (stored data) and in transit (data being transmitted) to protect it from unauthorized access.

This can be achieved using Transparent Data Encryption (TDE) for data at rest and SSL/TLS for data in transit.

Access Control: Implement strict user authentication and authorization policies.

Use roles and permissions to control who can access and modify the database, ensuring that only authorized users have appropriate levels of access.

2.How do you create a user and assign roles in MSSQL?

Create a Login:

sql

Copy code

CREATE LOGIN [username] WITH PASSWORD = 'your\_password';

Create a User in a Specific Database:

sql

Copy code

USE [YourDatabaseName];

CREATE USER [username] FOR LOGIN [username];

Assign Roles to the User:

sql

Copy code

ALTER ROLE [role\_name] ADD MEMBER [username];Create a Login:

sql

Copy code

CREATE LOGIN [username] WITH PASSWORD = 'your\_password';

Create a User in a Specific Database:

sql

Copy code

USE [YourDatabaseName];

CREATE USER [username] FOR LOGIN [username];

Assign Roles to the User:

sql

Copy code

ALTER ROLE [role\_name] ADD MEMBER [username];

3. explain how encrption can be implemented for data at rest in MSSQL.

Create a Master Key:

sql

Copy code

CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'your\_master\_key\_password';

Create a Certificate:

sql

Copy code

CREATE CERTIFICATE MyEncryptionCert

WITH SUBJECT = 'Encryption Certificate';

Create a Database Encryption Key:

sql

Copy code

USE [YourDatabaseName];

CREATE DATABASE ENCRYPTION KEY

WITH ALGORITHM = AES\_256

ENCRYPTION BY SERVER CERTIFICATE MyEncryptionCert;

Enable Encryption on the Database:

sql

Copy code

ALTER DATABASE [YourDatabaseName]

SET ENCRYPTION ON;

This process encrypts the database files, protecting data at rest from unauthorized access.