**Лабораторная работа 1. Расширенные возможности SELECT**

Предметная область БД, на которой необходимо производить упражнения, может быть любой. Рекомендуется использовать БД, которая была разработана в ходе лабораторного практикума по дисциплине «Управление данными». Структура таблиц должна включать в себя, как минимум, одну главную таблицу и две связанные с ней подчинённые таблицы. Например, «Продажи»-«Продавцы»-«Товары», или «Сотрудники»-«Отделы»-«Должности». Таблицы должны быть заполненными тестовыми данными, достаточными для демонстрации работы запросов (не менее 10 записей в каждой таблице).

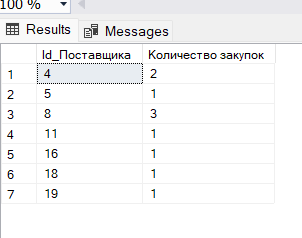
**Задание:**

⦁ Продемонстрируйте работу простого запроса на группировку с отбором записей в исходном наборе до группировки и отбором получившихся групп в результирующем наборе:

SELECT Id\_Поставщика, COUNT(\*) AS [Количество закупок]

FROM Закупки

GROUP BY Id\_Поставщика



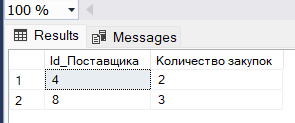
Запрос с отбором получившихся групп в результирующем наборе:

SELECT Id\_Поставщика, COUNT(\*) AS [Количество закупок]

FROM Закупки

GROUP BY Id\_Поставщика

HAVING COUNT(\*) > 1

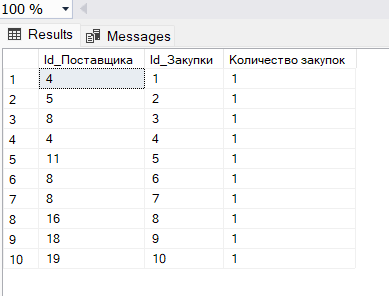


⦁ Повторите п.1, используя при этом группировку по двум-трём столбцам.

SELECT Id\_Поставщика, Id\_Закупки, COUNT(\*) AS [Количество закупок]

FROM Закупки

GROUP BY Id\_Поставщика, Id\_Закупки

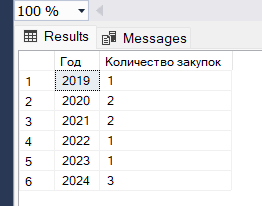


⦁ Повторите п.1, используя группировку по вычисляемому выражению.

SELECT YEAR([Дата поставки]) AS Год, COUNT(\*) AS [Количество закупок]

FROM Закупки

GROUP BY YEAR([Дата поставки])



⦁ Продемонстрируйте работу простого запроса с использованием группировки по результату соединения (join) имеющихся таблиц. Покажите проблему группировки кортежей подчинённой таблицы по неуникальному полю одной из связанных таблиц. Продемонстрируйте более правильный вариант группировки.

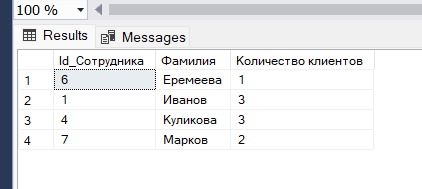
--До

SELECT Сотрудники.Id\_Сотрудника, Сотрудники.Фамилия, COUNT(\*) AS [Количество клиентов]

FROM Сотрудники

JOIN Заказ ON Сотрудники.Id\_Сотрудника = Заказ.Id\_Сотрудника

GROUP BY Сотрудники.Id\_Сотрудника, Сотрудники.Фамилия



Проблема в том, что у одного сотрудника есть несколько записей(клиентов) в таблице Заказ. Чтобы решить эту проблему, необходимо использовать вложенный запрос для предварительной группировки данных в таблице Заказ и затем объединить результаты с таблицей Сотрудники. Ниже представлен правильный вариант запроса:

--После

SELECT Сотрудники.Id\_Сотрудника, COUNT(\*) AS [Количество клиентов]

FROM Сотрудники

JOIN (

SELECT Id\_Сотрудника, COUNT(\*) AS [Количество клиентов]

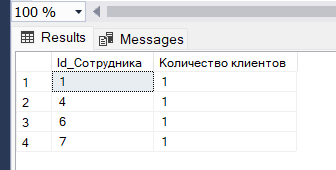
FROM Заказ

GROUP BY Id\_Сотрудника) AS ЗаказСотр

ON Сотрудники.Id\_Сотрудника = ЗаказСотр.Id\_Сотрудника

GROUP BY Сотрудники.Id\_Сотрудника

Здесь сначала идет группировка по столбцу Id\_Сотрудника таблицы Заказ, затем этот результат соединяется с таблицей Сотрудники по столбцу Id\_Сотрудника (подсчет кол-ва записей для каждого сотрудника) и выполняется окончательная правильная группировка двух таблиц



⦁ Модифицируйте запрос из п.4 так, чтобы в нём появились подитоги по иерархии значений в столбцах группировки. Используйте для этого оператор ROLLUP.

SELECT Сотрудники.Id\_Сотрудника, Сотрудники.Фамилия, COUNT(\*) AS [Количество клиентов]

FROM Сотрудники

JOIN (

SELECT Id\_Сотрудника, COUNT(\*) AS [Количество клиентов]

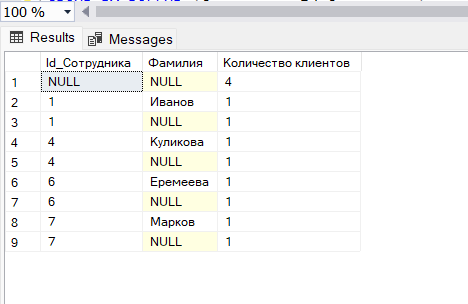
FROM Заказ

GROUP BY Id\_Сотрудника) AS ЗаказСотр

ON Сотрудники.Id\_Сотрудника = ЗаказСотр.Id\_Сотрудника

GROUP BY ROLLUP (Сотрудники.Id\_Сотрудника, Сотрудники.Фамилия)

ORDER BY Сотрудники.Id\_Сотрудника



При этом покажите использование различного количества столбцов в операторе ROLLUP.

SELECT Сотрудники.Id\_Сотрудника, COUNT(\*) AS [Количество клиентов]

FROM Сотрудники

JOIN (

SELECT Id\_Сотрудника, COUNT(\*) AS [Количество клиентов]

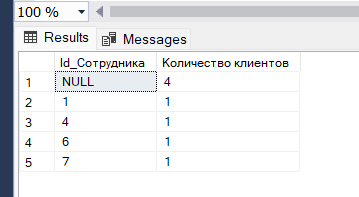
FROM Заказ

GROUP BY Id\_Сотрудника) AS ЗаказСотр

ON Сотрудники.Id\_Сотрудника = ЗаказСотр.Id\_Сотрудника

GROUP BY ROLLUP (Сотрудники.Id\_Сотрудника)

ORDER BY Сотрудники.Id\_Сотрудника



⦁ Модифицируйте запрос из п.4 так, чтобы в нём появились подитоги по комбинациям значений в столбцах группировки. Используйте для этого оператор CUBE. При этом покажите использование различного количества столбцов в операторе CUBE.

SELECT Сотрудники.Id\_Сотрудника, COUNT(\*) AS [Количество клиентов]

FROM Сотрудники

JOIN (

SELECT Id\_Сотрудника, COUNT(\*) AS [Количество клиентов]

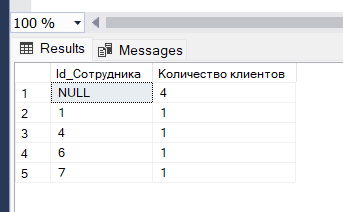
FROM Заказ

GROUP BY Id\_Сотрудника) AS ЗаказСотр

ON Сотрудники.Id\_Сотрудника = ЗаказСотр.Id\_Сотрудника

GROUP BY CUBE (Сотрудники.Id\_Сотрудника)

ORDER BY Сотрудники.Id\_Сотрудника



SELECT Сотрудники.Id\_Сотрудника, Сотрудники.Фамилия, COUNT(\*) AS [Количество клиентов]

FROM Сотрудники

JOIN (

SELECT Id\_Сотрудника, COUNT(\*) AS [Количество клиентов]

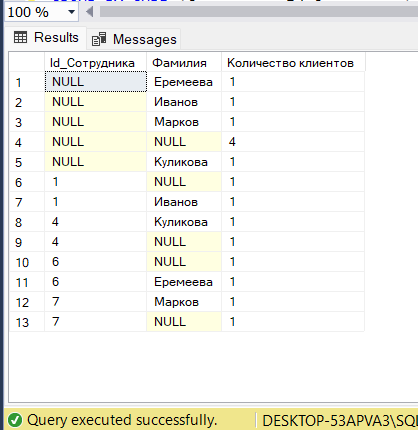
FROM Заказ

GROUP BY Id\_Сотрудника) AS ЗаказСотр

ON Сотрудники.Id\_Сотрудника = ЗаказСотр.Id\_Сотрудника

GROUP BY CUBE (Сотрудники.Id\_Сотрудника, Сотрудники.Фамилия)

ORDER BY Сотрудники.Id\_Сотрудника



⦁ Модифицируйте запрос из п.4. установив при помощи оператора GROUPING SETS произвольный набор конфигураций уровней блокирования.

SELECT Сотрудники.Id\_Сотрудника, Сотрудники.Фамилия, COUNT(\*) AS [Количество клиентов]

FROM Сотрудники

JOIN (

SELECT Id\_Сотрудника, COUNT(\*) AS [Количество клиентов]

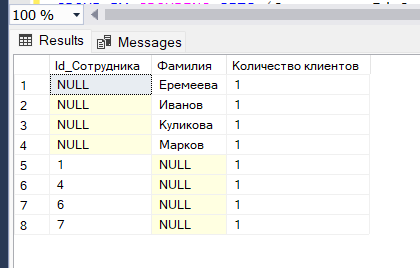
FROM Заказ

GROUP BY Id\_Сотрудника) AS ЗаказСотр

ON Сотрудники.Id\_Сотрудника = ЗаказСотр.Id\_Сотрудника

GROUP BY GROUPING SETS (Сотрудники.Id\_Сотрудника, Сотрудники.Фамилия)

ORDER BY Сотрудники.Id\_Сотрудника



⦁ Продемонстрируйте примеры эквивалентных соотношений между операторами ROLLUP, CUBE и оператора GROUPING SETS.

ROLLUP:

SELECT Сотрудники.Id\_Сотрудника, Сотрудники.Фамилия, COUNT(\*) AS [Количество клиентов]

FROM Сотрудники

JOIN (

SELECT Id\_Сотрудника, COUNT(\*) AS [Количество клиентов]

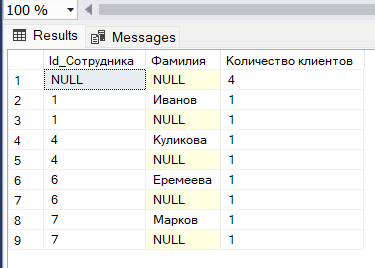
FROM Заказ

GROUP BY Id\_Сотрудника) AS ЗаказСотр

ON Сотрудники.Id\_Сотрудника = ЗаказСотр.Id\_Сотрудника

GROUP BY ROLLUP (Сотрудники.Id\_Сотрудника, Сотрудники.Фамилия)

ORDER BY Сотрудники.Id\_Сотрудника



CUBE:

SELECT Сотрудники.Id\_Сотрудника, Сотрудники.Фамилия, COUNT(\*) AS [Количество клиентов]

FROM Сотрудники

JOIN (

SELECT Id\_Сотрудника, COUNT(\*) AS [Количество клиентов]

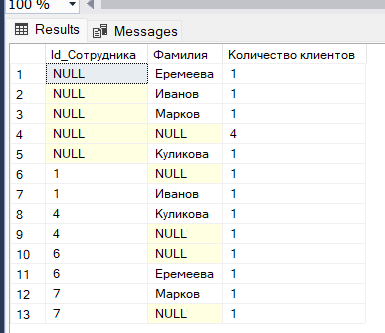
FROM Заказ

GROUP BY Id\_Сотрудника) AS ЗаказСотр

ON Сотрудники.Id\_Сотрудника = ЗаказСотр.Id\_Сотрудника

GROUP BY CUBE (Сотрудники.Id\_Сотрудника, Сотрудники.Фамилия)

ORDER BY Сотрудники.Id\_Сотрудника



GROUPING SETS:

SELECT Сотрудники.Id\_Сотрудника, Сотрудники.Фамилия, COUNT(\*) AS [Количество клиентов]

FROM Сотрудники

JOIN (

SELECT Id\_Сотрудника, COUNT(\*) AS [Количество клиентов]

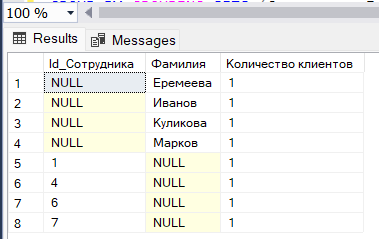
FROM Заказ

GROUP BY Id\_Сотрудника) AS ЗаказСотр

ON Сотрудники.Id\_Сотрудника = ЗаказСотр.Id\_Сотрудника

GROUP BY GROUPING SETS (Сотрудники.Id\_Сотрудника, Сотрудники.Фамилия)

ORDER BY Сотрудники.Id\_Сотрудника



Все три запроса создают результат, который содержит общее количество клиентов на каждого сотрудника. Различия в операторах ROLLUP, CUBE и GROUPING SETS состоят в способе группировки и иерархии данных в итоговом результате

⦁ При помощи оператора GROUPING отделите в итоговом наборе запроса из п.4 значения NULL, показывающие исключение соответствующего атрибута из группирования от значений NULL, показывающие отсутствующие значения. Для этого замените первые - на строку «ВСЕ», а вторые – на строку «НЕИЗВЕСТНО».

SELECT CASE WHEN Сотрудники.Id\_Сотрудника IS NULL THEN 'ВСЕ' ELSE 'НЕИЗВЕСТНО' END,

Сотрудники.Фамилия, COUNT(\*) AS [Количество клиентов]

FROM Сотрудники

JOIN (

SELECT Id\_Сотрудника, COUNT(\*) AS [Количество клиентов]

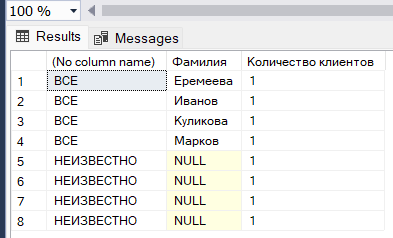
FROM Заказ

GROUP BY Id\_Сотрудника) AS ЗаказСотр

ON Сотрудники.Id\_Сотрудника = ЗаказСотр.Id\_Сотрудника

GROUP BY GROUPING SETS (Сотрудники.Id\_Сотрудника, Сотрудники.Фамилия)

ORDER BY Сотрудники.Id\_Сотрудника, Сотрудники.Фамилия



⦁ Продемонстрируйте работу ранжирующих функций RANK, DENSE\_RANK, ROW\_NUMBER и NTILE. Наглядно покажите разницу между ними.

1. RANK: присваивает ранг каждой строке на основе заданного условия.

2. DENSE\_RANK: присваивает ранг каждой строке без пропусков.

3. ROW\_NUMBER: присваивает уникальный номер каждой строке.

SELECT Сотрудники.Id\_Сотрудника, Сотрудники.Фамилия, COUNT(\*) AS [Количество клиентов],

RANK() OVER (ORDER BY COUNT(\*) DESC) as RANK\_,

DENSE\_RANK() OVER (ORDER BY COUNT(\*) DESC) as DENSE\_RANK\_,

ROW\_NUMBER() OVER (ORDER BY COUNT(\*) DESC) as ROW\_NUMBER\_,

NTILE(2) OVER (ORDER BY COUNT(\*) DESC) as NTILE\_

FROM Сотрудники

JOIN (

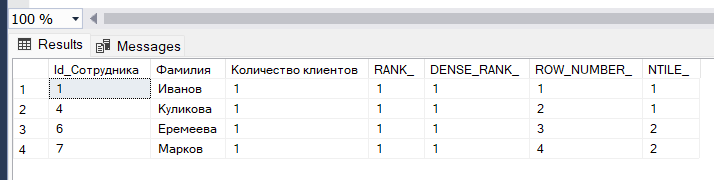
SELECT Id\_Сотрудника, COUNT(\*) AS [Количество клиентов]

FROM Заказ

GROUP BY Id\_Сотрудника) AS ЗаказСотр

ON Сотрудники.Id\_Сотрудника = ЗаказСотр.Id\_Сотрудника

GROUP BY Сотрудники.Id\_Сотрудника, Сотрудники.Фамилия



⦁ Повторите пункт 10, но с применением оконных функций. В качестве критерия выделения окон можно выбрать отдел или должность для таблицы «Сотрудники» или категорию для таблицы «Товары».

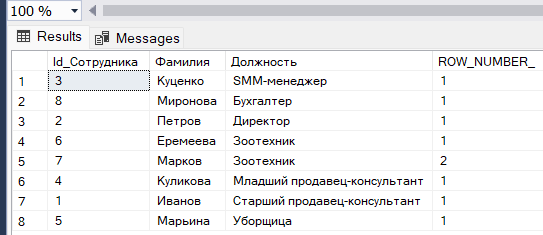
SELECT Сотрудники.Id\_Сотрудника, Сотрудники.Фамилия, Сотрудники.Должность, ROW\_NUMBER() OVER(

PARTITION BY Сотрудники.Должность

ORDER BY Сотрудники.Фамилия

) AS ROW\_NUMBER\_

FROM Сотрудники



⦁ Добавьте в одну из таблиц Вашей схемы атрибут, который будет являться внешним ключом, указывающим на записи этой же таблицы. Например, в таблицу «Сотрудники» добавьте информацию о руководителе для каждого сотрудника или для таблицы «Товары» - информацию о сопутствующем товаре, который прилагается к данному товару в подарок по рекламной акции. Составьте рекурсивное табличное выражение, в котором наглядно выведите записи вашей таблицы в порядке их иерархии. Предусмотрите визуальное отображение иерархии, например, при помощи отступов различной величины.

WITH Рекурсия AS (

SELECT Id\_Сотрудника, Фамилия, Имя, Отчество, Id\_Руководителя, 0 AS Уровень

FROM Сотрудники

WHERE Id\_Руководителя IS NULL

UNION ALL

SELECT c.Id\_Сотрудника, c.Фамилия, c.Имя, c.Отчество, c.Id\_Руководителя, r.Уровень + 1

FROM Сотрудники c

INNER JOIN Рекурсия r ON c.Id\_Руководителя = r.Id\_Сотрудника

)

SELECT

REPLICATE(' ', Уровень) + Фамилия + ' ' + Имя + ' ' + Отчество AS Иерархия

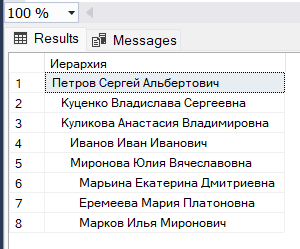
FROM Рекурсия;

В данном запросе - в первой строке - сам директор

2-3 - группа первого руководителя

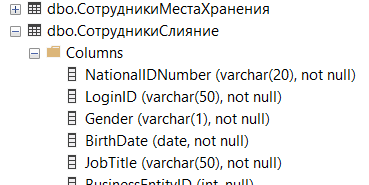
4-5 - группа второго руководителя

6-8 - группа третьего руководителя



⦁ Продемонстрируете владение оператором слияния наборов MERGE. Для этого организуйте слияние с существующей в Вашей схеме таблицы набора данных, полученного при помощи запроса к любой другой таблице. Например, в качестве источника строк можно использовать базу данных Adventure Work, входящую в комплект поставки MS SQL Server. В качестве критерия слияния выберите соответствие значений в столбцах, которые могут служить естественным идентифицирующим признаком для сущностей (например, серия и номер паспорта для сотрудника или артикул для товара).

Создание таблицы СотрудникиСлияние:



MERGE INTO СотрудникиСлияние AS target

USING AdventureWorks2022.HumanResources.Employee AS source

ON target.BusinessEntityID = source.BusinessEntityID

WHEN MATCHED THEN

UPDATE SET

target.NationalIDNumber = source.NationalIDNumber,

target.LoginID = source.LoginID,

target.Gender = source.Gender,

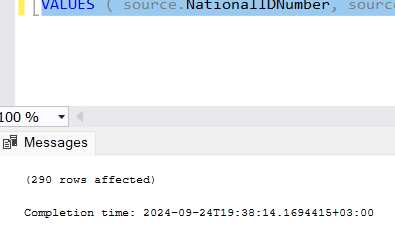
target.BirthDate = source.BirthDate,

target.JobTitle = source.JobTitle

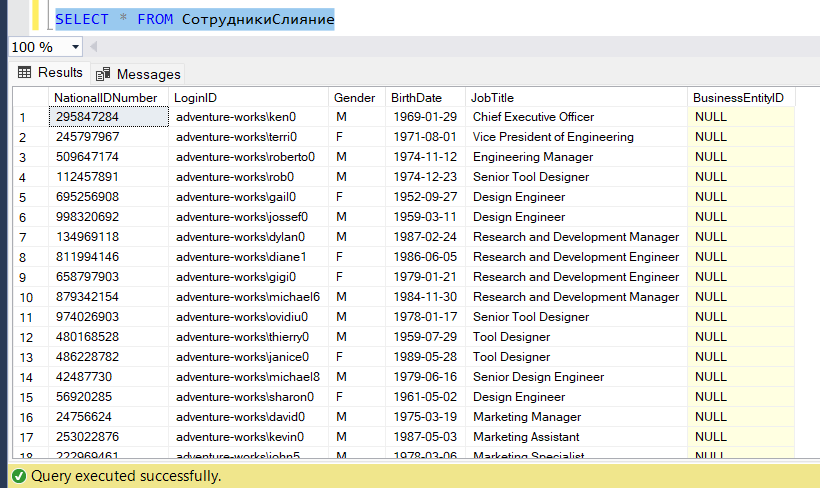
WHEN NOT MATCHED THEN

INSERT ([NationalIDNumber], [LoginID], [Gender] , [BirthDate],[JobTitle] )

VALUES ( source.NationalIDNumber, source.LoginID, source.Gender, source.BirthDate, source.JobTitle );



Результат слияния данных двух таблиц из двух баз данных:



Вопросы к отчету:

1. Простая группировка данных. Операторы GROUP BY и HAVING:

Простая группировка данных используется для объединения строк с одинаковым значением в определенном столбце или столбцах. Оператор GROUP BY используется для определения критериев группировки, а оператор HAVING применяется после группировки для фильтрации результатов по условиям, применяемым к агрегированным данным.

2. Ограничения оператора GROUP BY:

Оператор GROUP BY не позволяет использовать в списке выбора столбцы, которые не были включены в оператор GROUP BY (если они не являются агрегатными функциями). Это ограничение обеспечивает ясность и однозначность результата запроса.

3. Группировка по вычисляемым выражениям:

Группировка по вычисляемым выражениям позволяет использовать результаты вычислений в качестве критериев группировки. Например, вы можете группировать данные по году, извлеченному из столбца с датой.

4. Составная группировка по двум и более столбцам. В чём её смысл?

Составная группировка позволяет группировать данные по комбинациям значений из двух или более столбцов. Это полезно, когда требуется более детальное разбиение данных на группы для анализа.

5. Промежуточные подитоги. ROLLUP, CUBE и GROUPING SETS:

- Операторы ROLLUP, CUBE и GROUPING SETS используются для создания промежуточных итогов в результирующих наборах данных при выполнении операций группировки.

- ROLLUP генерирует итоги для всех возможных комбинаций указанных столбцов, предоставляя промежуточные итоги на каждом уровне группировки.

- CUBE создает итоги для всех возможных комбинаций указанных столбцов, обеспечивая более полное покрытие всех вариантов группировки.

- GROUPING SETS позволяет явно указать, какие комбинации столбцов следует использовать для создания промежуточных итогов.

6. Ранжирующие функции. Их возможности и сферы применения:

Ранжирующие функции присваивают ранг каждой строке внутри группы результатов в соответствии с заданными критериями сортировки. Они часто используются для выявления лучших или худших значений внутри группы, ранжирования результатов и анализа топ-N значений.

7. Оконные функции. Применение совместно с агрегатными или ранжирующими функциями:

Оконные функции позволяют выполнять вычисления на определенном диапазоне строк (окне) в пределах результирующего набора данных. Они часто используются с агрегатными и ранжирующими функциями для более гибкого анализа данных, чем традиционные агрегатные функции.

8. Обобщённые табличные выражения. Сферы применения обобщённых табличных выражений:

Обобщенные табличные выражения позволяют выполнять операции с данными, подобные операциям с таблицами, включая их создание, изменение и удаление. Они широко используются для решения сложных задач обработки данных, включая рекурсивные запросы.

9. Организация рекурсивных запросов при помощи обобщённых табличных выражений:

Обобщенные табличные выражения используются для создания рекурсивных запросов, которые требуют многократного обращения к тем же данным. Это часто используется для работы с иерархическими данными, такими как деревья или организационные структуры.

10. Оператор слияния наборов MERGE. Возможности и сферы его применения:

Оператор слияния наборов MERGE используется для выполнения операций вставки, обновления и удаления данных в целевой таблице на основе результатов запроса к другой таблице. Он широко применяется для синхронизации данных между двумя таблицами, обновления данных по заданным критериям или выполнения сложных операций по обновлению данных.