Лабораторная работа №7:
Оператор JOIN, UNION. Представления.

SQL Server выполняет операции сортировки, пересечения, объединения и поиска различий при помощи технологий хэш-соединений и сортировки в оперативной памяти.

С помощью соединения можно получать данные из двух или нескольких таблиц на основе логических связей между ними. Соединения позволяют указать, как в SQL Server должны использоваться данные из одной таблицы для выбора строк из другой таблицы.

Соединение определяет способ связывания двух таблиц в запросе следующим образом:

- для каждой таблицы указываются столбцы, используемые в соединении. В типичном условии соединения указывается внешний ключ из одной таблицы и связанный с ним ключ из другой таблицы;
- указывается логический оператор (например, = или <>,) для сравнения значений столбцов.

Внутренние соединения можно задавать в предложениях FROM и WHERE. Внешние соединения можно задавать только в предложении FROM. Условия соединения сочетаются с условиями поиска WHERE и HAVING для управления строками, выбранными из базовых таблиц, на которые ссылается предложение FROM.

То, что условия соединения задаются в предложении FROM, помогает отделить их от условий поиска, которые могут быть заданы в предложении WHERE. Объединение рекомендуется задавать именно таким способом.

Ниже приведена простая инструкция SELECT, использующая это соединение:

```
SELECT ProductID, Purchasing.Vendor.BusinessEntityID, Name
FROM Purchasing.ProductVendor JOIN Purchasing.Vendor
    ON (Purchasing.ProductVendor.BusinessEntityID = Purchasing.Vendor.Bus
WHERE StandardPrice > $10
  AND Name LIKE N'F%'
GO
```

Инструкция возвращает наименование продукта и сведения о поставщике для всех сочетаний запчастей, поставляемых компаниями с названиями на букву F и стоимостью продукта более 10 долларов.

ЗАДАНИЕ:

1. **Создайте таблицу, содержащую сведения о должностях сотрудников (Код должности, Наименование должности, Оклад). Заполните таблицу данными.** ***Обратите внимание, что в ранее импортированной таблице Сотрудник уже имеется столбец Код должности, содержащий значения от 1 до 6, соответственно таблица Должность должна будет содержать минимум 6 строк со значениями Код должности от 1 до 6.***
2. **Создайте внешний ключ между Таблицами Сотрудник и Должность по полю Код должности.**

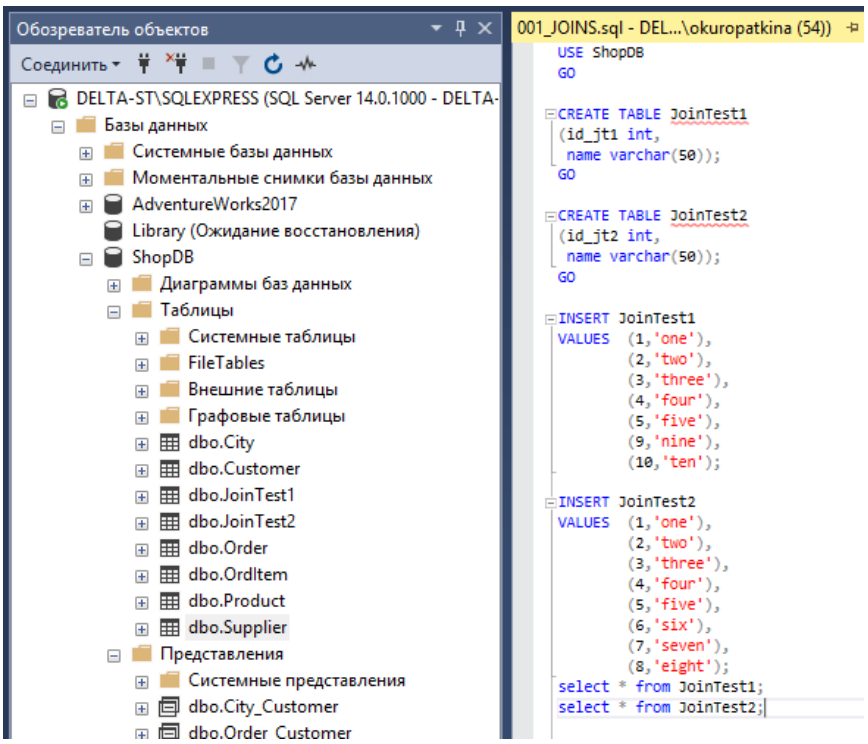
3. **Создайте запрос с использованием конструкции JOIN, который выводит информацию из базы данных ShopDB: вывести ФИО Сотрудника, должность сотрудника, дата рождения сотрудника, с условием, что дата рождения сотрудника больше заданной (на свое усмотрение).**
4. **Создайте запрос, аналогичный п.3 с условием: Фамилия сотрудника начинается на заданную букву.**
5. **При помощи запроса выведите сведения: ФИО Сотрудника, должность сотрудника, количество реализованных заказов.**

ВИДЫ СОЕДИНЕНИЙ.

- INNER JOIN (внутреннее объединение) - объединение, при котором в запросе все записи из таблицы на левой и правой стороне операции INNER JOIN добавляются в результирующий набор записей, при соответствии условию значений в связанных полях.
- · LEFT OUTER JOIN (левое внешнее объединение) - внешнее объединение, при котором в запросе все записи из таблицы на левой стороне операции LEFT JOIN в инструкции SQL добавляются в результирующий набор записей, даже если в таблице на правой стороне отсутствуют совпадающие значения в связанных полях.
- · RIGHT OUTER JOIN (правое внешнее объединение) - внешнее объединение, при котором в запросе все записи из таблицы на правой стороне операции RIGHT JOIN в инструкции SQL добавляются в результирующий набор записей, даже если в таблице на левой стороне отсутствуют совпадающие значения в связанных полях.
- · FULL OUTER JOIN (полное объединение) внешнее объединение, при котором в запросе все записи из таблицы на левой и правой стороне операции FULL JOIN добавляются в результирующий набор записей, при соответствии условию значений в связанных полях, а также:
- 1) значения из правой таблицы, не имеющие соответствий в левой таблице;
- 2) значения из левой таблицы, не имеющие соответствий в правой таблице.

- · CROSS JOIN (перекрестное объединение) - выполняет декартово произведение таблиц, вовлеченных в объединение. В CROSS JOIN не используется конструкция ON.

Для понимания работы данных соединений создайте в своей базе данных две таблицы как указано в примере:



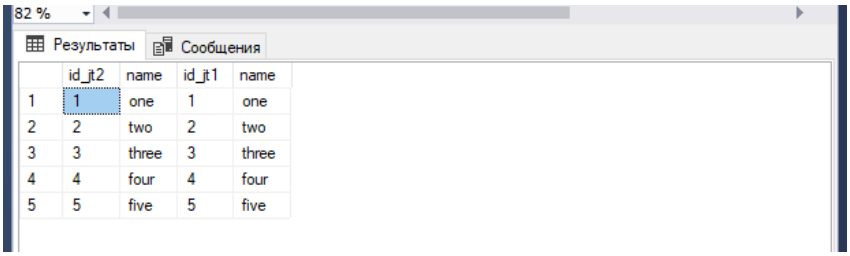
1. **Выполните запрос с использованием внутреннего объединения:** INNER JOIN.

```
    INNER JOIN (внутреннее объединение) - объединение, при котором
    --    в запросе все записи из таблицы на левой и правой стороне операции
    --    INNER JOIN добавляются в результирующий набор записей, при соответствии
    --    условию значений в связанных полях.
Производим выборку всех данных из объединения таблиц JoinTest1 и JoinTest2 по связующим
полям id_jt1 и id_jt2.

SELECT * FROM
       JoinTest2       -- Левая таблица (Таблица JoinTest2)
       INNER JOIN  -- Оператор объединения.
       JoinTest1       -- Правая таблица(Таблица JoinTest1)
ON id_jt1 = id_jt2;          -- Условие объединения при котором значения в сравниваемых
ячейках должны совпадать.

GO
```

Вы получите следующий результат объединения:



	id_jt2	name	id_jt1	name
1	1	one	1	one
2	2	two	2	two
3	3	three	3	three
4	4	four	4	four
5	5	five	5	five

2. **Выполните запрос с использованием левого внешнего объединения:** LEFT OUTER JOIN.

```
    LEFT OUTER JOIN (левое внешнее объединение) - внешнее объединение, при котором
    --    в запросе все записи из таблицы на левой стороне операции LEFT JOIN в
    --    инструкции SQL добавляются в результирующий набор записей, даже если в
    --    таблице на правой стороне отсутствуют совпадающие значения в связанных полях.
```

```
Производим выборку всех данных из результирующего набора данных левого внешнего
-- объединения таблиц JoinTest1 и JoinTest2 по связующим полям id_jt1 и id_jt2.

SELECT * FROM JoinTest2     -- Левая таблица JoinTest2
    LEFT OUTER JOIN JoinTest1      -- LEFT JOIN
ON id_jt1=id_jt2;

      GO
```

Результаты | Сообщения

	id_jt2	name	id_jt1	name
1	1	one	1	one
2	2	two	2	two
3	3	three	3	three
4	4	four	4	four
5	5	five	5	five
6	6	six	NULL	NULL
7	7	seven	NULL	NULL
8	8	eight	NULL	NULL

3. Выполните запрос с использованием правого внешнего объединения: RIGHT OUTER JOIN.

```
RIGHT OUTER JOIN (правое внешнее объединение) - внешнее объединение, при котором
--   в запросе все записи из таблицы на правой стороне операции RIGHT JOIN в
--   инструкции SQL добавляются в результирующий набор записей, даже если в
--   таблице на левой стороне отсутствуют совпадающие значения в связанных полях.
Производим выборку всех данных из результирующего набора данных правого внешнего
-- объединения таблиц JoinTest1 и JoinTest2 по связующим полям id_jt1 и id_jt2.

SELECT * FROM JoinTest2
   RIGHT OUTER JOIN JoinTest1 -- Правая таблица JoinTest2
ON id_jt1 = id_jt2;

GO
```

Вы получите следующий результат объединения:

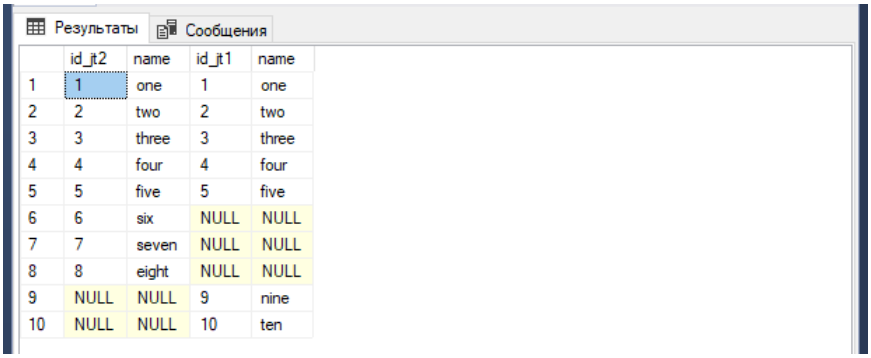
Результаты | Сообщения

	id_jt2	name	id_jt1	name
1	1	one	1	one
2	2	two	2	two
3	3	three	3	three
4	4	four	4	four
5	5	five	5	five
6	NULL	NULL	9	nine
7	NULL	NULL	10	ten

4. Выполните запрос с использованием полного объединения: FULL OUTER JOIN.

```
FULL OUTER JOIN (полное объединение) внешнее объединение, при котором
--   в запросе все записи из таблицы на левой и правой стороне операции
--   FULL JOIN добавляются в результирующий набор записей, при соответствии
--   условию значений в связанных полях, а также:
--         - значения из первой таблицы, не имеющие соответствий в левой таблице;
--         - значения из левой таблицы, не имеющие соответствий в правой таблице.

SELECT *
FROM JoinTest2
FULL OUTER JOIN JoinTest1      --FULL JOIN
ON id_jt1 = id_jt2;                              GO
```


Результаты Сообщения

	id_jt2	name	id_jt1	name
1	1	one	1	one
2	2	two	2	two
3	3	three	3	three
4	4	four	4	four
5	5	five	5	five
6	6	six	NULL	NULL
7	7	seven	NULL	NULL
8	8	eight	NULL	NULL
9	NULL	NULL	9	nine
10	NULL	NULL	10	ten

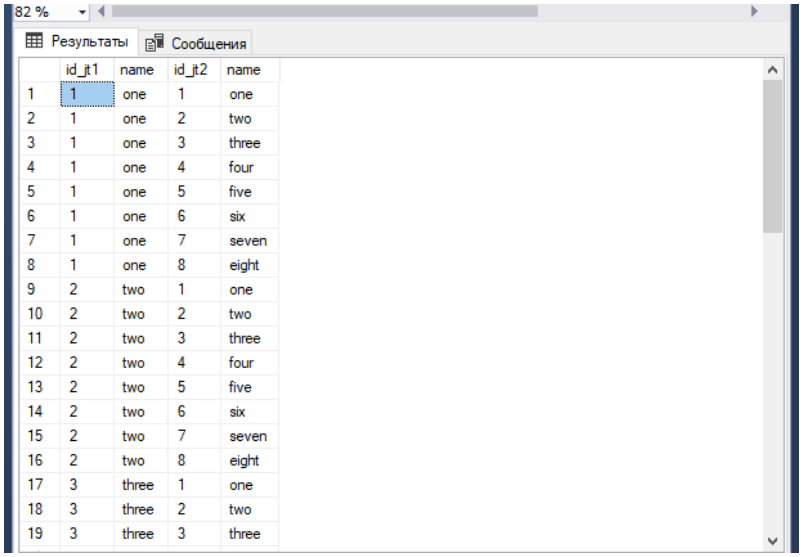
5. **Выполните запрос с использованием полного объединения:** CROSS JOIN.

```
CROSS JOIN (перекрестное объединение) - выполняет декартово произведение таблиц,
-- вовлеченных в объединение. В CROSS JOIN не используется конструкция ON.

-- Производим выборку всех данных из результирующего набора данных перекрестного
-- объединения таблиц JoinTest1 и JoinTest2.

SELECT * FROM JoinTest1
   CROSS JOIN JoinTest2 -- CROSS JOIN
-- ON - не используется

GO
```


82 %

Результаты Сообщения

	id_jt1	name	id_jt2	name
1	1	one	1	one
2	1	one	2	two
3	1	one	3	three
4	1	one	4	four
5	1	one	5	five
6	1	one	6	six
7	1	one	7	seven
8	1	one	8	eight
9	2	two	1	one
10	2	two	2	two
11	2	two	3	three
12	2	two	4	four
13	2	two	5	five
14	2	two	6	six
15	2	two	7	seven
16	2	two	8	eight
17	3	three	1	one
18	3	three	2	two
19	3	three	3	three

Оператор UNION.

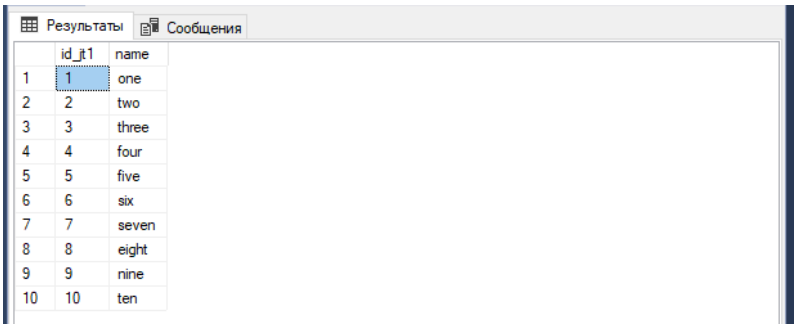
UNION объединяет результаты двух запросов SELECT в единую результирующую таблицу.

Если результаты обоих запросов содержат строки с совпадающими значениями ячеек, то, операция UNION помещает в результирующую таблицу только одну такую строку.

Если в результате одного из запросов имеются строки с уникальными значениями, не совпадающими ни с одной из строк результата другого запроса, то эта строка также помещается в результирующую таблицу.

Операция UNION требует использования таких запросов, каждый из которых возвращает выборку в табличном представлении, при этом, типы и количество столбцов должны совпадать.

```
SELECT * FROM JoinTest1
UNION
SELECT * FROM JoinTest2
```


Результаты Сообщения

	id_jt1	name
1	1	one
2	2	two
3	3	three
4	4	four
5	5	five
6	6	six
7	7	seven
8	8	eight
9	9	nine
10	10	ten

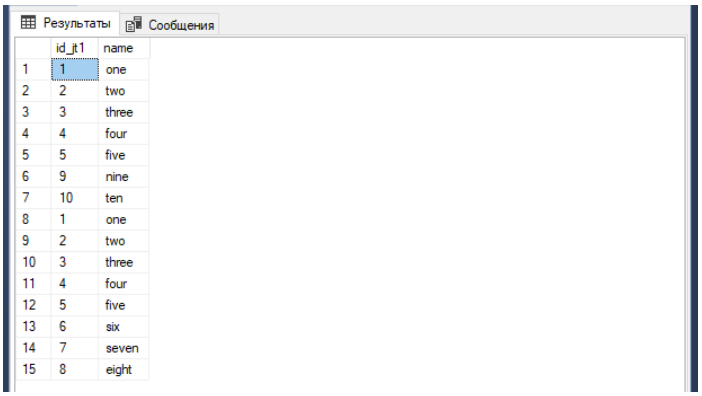
UNION ALL объединяет результаты двух запросов SELECT в единую результирующую таблицу.

Если результаты обоих запросов содержат строки с совпадающими значениями ячеек, то, операция UNION ALL помещает в результирующую таблицу все дублирующийся строки.

Если в результате одного из запросов имеются строки с уникальными значениями, не совпадающими ни с одной из строк результата другого запроса, то эта строка также помещается в результирующую таблицу.

Операция UNION ALL требует использования таких запросов, каждый из которых возвращает выборку в табличном представлении, при этом, типы и количество столбцов должны совпадать.

```
SELECT * FROM JoinTest1
UNION ALL
SELECT * FROM JoinTest2
```


Результаты Сообщения

	id_jt1	name
1	1	one
2	2	two
3	3	three
4	4	four
5	5	five
6	9	nine
7	10	ten
8	1	one
9	2	two
10	3	three
11	4	four
12	5	five
13	6	six
14	7	seven
15	8	eight

Операция EXCEPT исключает результаты правого запроса.

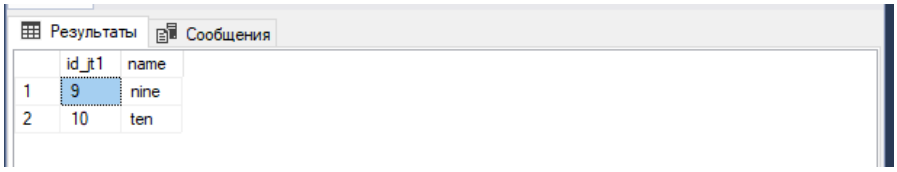
Если результат левого запроса операции EXCEPT содержит уникальные строки, не совпадающие ни с одной из строк правого запроса, то, только такие строки помещается в результирующую таблицу.

Уникальные строки правого запроса операции EXCEPT, никогда не входят в результирующую таблицу.

Если результаты обоих запросов содержат совпадающие строки, то, операция EXCEPT игнорирует их.

Операция EXCEPT требует использования таких запросов, каждый из которых возвращает выборку в табличном представлении, при этом, типы и количество столбцов должны совпадать.

```
SELECT * FROM JoinTest1
EXCEPT
SELECT * FROM JoinTest2
```



	id_jt1	name
1	9	nine
2	10	ten

```
SELECT * FROM JoinTest2
EXCEPT
SELECT * FROM JoinTest1
```

	id_jt2	name
1	6	six
2	7	seven
3	8	eight

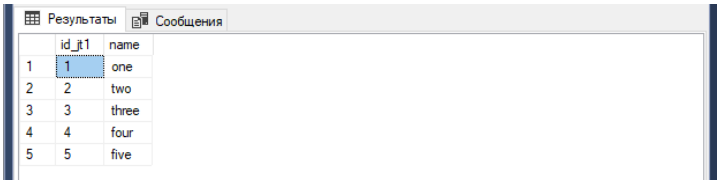
INTERSECT объединяет результаты двух запросов SELECT в единую результирующую таблицу.

Если результаты обоих запросов содержат строки с совпадающими значениями ячеек, то, операция INTERSECT помещает в результирующую таблицу только одну такую строку.

Если в результате одного из запросов имеются уникальные строки, не совпадающие ни с одной из строк результата другого запроса, то такие строки игнорируются операцией INTERSECT.

Операция INTERSECT требует использования таких запросов, каждый из которых возвращает выборку в табличном представлении, при этом, типы и количество столбцов должны совпадать.

```
SELECT * FROM JoinTest1
INTERSECT
SELECT * FROM JoinTest2
```



	id_jt1	name
1	1	one
2	2	two
3	3	three
4	4	four
5	5	five

СОЗДАЙТЕ САМОСТОЯТЕЛЬНО С ИСПОЛЬЗОВАНИЕМ КОНСТРУКЦИИ JOIN, UNION НЕ МЕНЕЕ 7 ЗАПРОСОВ К БАЗЕ ДАННЫХ ShopDB (на свое усмотрение).

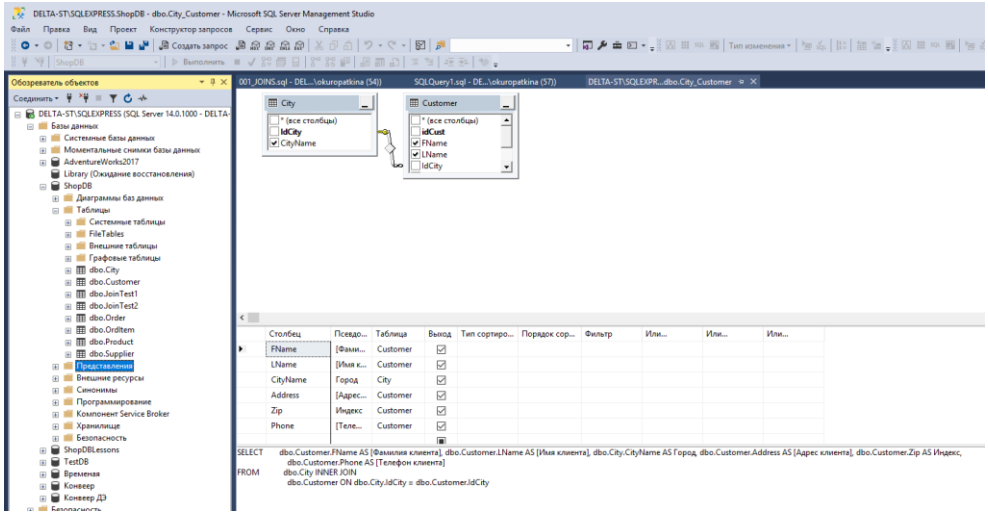
ПРЕДСТАВЛЕНИЕ (Views) — это виртуальная таблица, содержимое которой определяется запросом. Как и таблица, представление состоит из ряда именованных столбцов и строк данных. Пока представление не будет проиндексировано, оно не существует в базе данных как хранимая совокупность значений. Строки и столбцы данных извлекаются из таблиц, указанных в определяющем представление запросе и динамически создаваемых при обращениях к представлению.

Представление выполняет функцию фильтра базовых таблиц, на которые оно ссылается. Определяющий представление запрос может быть инициирован в одной или нескольких таблицах, или в других представлениях текущей или других баз данных.

Представления обычно используются для направления, упрощения и настройки восприятия каждым пользователем информации базы данных. Представления могут использоваться как механизмы безопасности, давая возможность пользователям обращаться к данным через представления, но не предоставляя им разрешений на непосредственный доступ к базовым таблицам, лежащим в основе представлений. Представления могут использоваться для обеспечения интерфейса обратной совместимости, моделирующего таблицу, которая существует, но схема которой изменилась. Представления могут также использоваться при прямом и обратном копировании данных в SQL Server для повышения производительности и секционирования данных.

Для создания представления средствами Microsoft SQL Server Management Studio необходимо в обозревателе объектов в БД «ShopDB» щёлкнуть ПКМ по папке «Представления», затем в появившемся меню выбрать пункт «Создать представление». Появиться окно «Добавить таблицу», предназначенное для выбора таблиц и запросов, участвующих в представлении.

Выберите таблицы City и Customer.



Замечание: Окно конструктора запросов состоит из следующих панелей:

1. Схема данных – отображает поля таблиц и запросов, участвующих в запросе, позволяет выбирать отображаемые поля, позволяет устанавливать связи между участниками запроса по специальным полям связи. Эта панель включается и выключается следующей кнопкой на панели инструментов;

2. Таблица отображаемых полей – показывает отображаемые поля (столбец «Column»), позволяет задавать им псевдонимы (столбец «Alias»), позволяет устанавливать тип

сортировки записей по одному или нескольким полям (столбец «Sort Type»), позволяет задавать порядок сортировки (столбец «Sort Order»), позволяет задавать условия отбора записей в фильтрах (столбцы «Filter» и «Or…»). Также эта таблица позволяет менять порядок отображения полей в запросе.

3. Код SQL – код создаваемого запроса на языке T-SQL.

4. Результат – показывает результат запроса после его выполнения.

Теперь перейдём к связыванию таблиц City и Customer по полям связи «idCity». Чтобы создать связь необходимо в схеме данных перетащить мышью поле «idCity» таблицы «City» на такое же поле таблицы «Customer». Связь отобразиться в виде ломаной линии, соединяющей эти два поля связи.

Замечание: Если необходимо удалить связь, то для этого необходимо щёлкнуть по ней ПКМ и в появившемся меню выбрать пункт «Remove».

Замечание: После связывания таблиц (а также при любых изменениях в запросе) в области кода T-SQL будет отображаться T-SQL код редактируемого запроса.

Теперь определим поля, отображаемые при выполнении запроса. Отображаемые поля обозначаются галочкой (слева от имени поля) на схеме данных, а также отображаются в таблице отображаемых полей. Чтобы сделать поле отображаемым при выполнении запроса необходимо щёлкнуть мышью по пустому квадрату (слева от имени поля) на схеме данных, в квадрате появится галочка.

Замечание: Если необходимо сделать поле невидимым при выполнении запроса, то нужно убрать галочку, расположенную слева от имени поля на схеме данных. Для этого просто щёлкните мышью по галочке.

Замечание: Если необходимо отобразить все поля таблицы, то необходимо установить галочку слева от пункта «* (All Columns)» (Все поля), принадлежащего соответствующей таблице на схеме данных.

Определите отображаемые поля нашего запроса, как это показано на рисунке (Отображаются все поля кроме полей с кодами, то есть полей связи).

Закройте окно создания представления без сохранения.

Теперь создадим данное представление кодом T-SQL. Это будет выглядеть следующим образом:

```
USE ShopDB
GO

CREATE VIEW City_Customer
AS
SELECT Customer.FName, Customer.LName, City.CityName, Customer.Address, Customer.Zip,
Customer.Phone
FROM dbo.City
INNER JOIN dbo.Customer
ON dbo.City.IdCity = dbo.Customer.IdCity
GO
```

Выведите сведения из представления:

```
SELECT * FROM City_Customer
```

В представление вы также можете поместить условия выборки (фильтры) как в обычном запросе.

ЗАДАНИЕ: Самостоятельно создайте представления:

1. **Выведите сведения о заказах за последний месяц с указанием ФИО заказчика, количеством заказов, суммой заказов.**
2. **Выведите сведения о сотрудниках с указанием ФИО, должности, дата рождения которых входит в заданных диапазон дат (на ваше усмотрение).**
3. **Выведите сведения и товарах, и их поставщиках.**
4. **Выведите сведения о поставках, реализованных в заданном диапазоне дат (на ваше усмотрение).**