Лабораторная работа № 2

SQL запросы

Целью данной лабораторной работы является изучение **наиболее важных** инструкций языка обработки данных DML.
Полный перечень инструкций и предложений DML приведен по адресу
http://msdn.microsoft.com/ru-ru/library/ff848766.aspx
Основные принципы составления запросов представлены по адресу
http://msdn.microsoft.com/ru-ru/library/ms190659(v=sql.105).aspx

Теоретическая часть

Для составления запросов к реляционным базам данных используются инструкции SQL из категории инструкций языка обработки данных (Data Manipulation Language, DML). В SQL Server используются следующие инструкции DML:

- SELECT - возвращает строки из базы данных и позволяет делать выборку одной или нескольких строк или столбцов из одной или нескольких таблиц в SQL Server. Операторы UNION, EXCEPT и INTERSECT можно использовать между запросами на выборку, чтобы сравнить их результаты или объединить в один результирующий набор.
- INSERT, UPDATE, DELETE, MERGE - добавляют, изменяют и удаляют данные в таблице или представлении базы данных в SQL Server.
- BULK INSERT - выполняет импорт файла данных в таблицу или представление базы данных в SQL Server.
- READTEXT, WRITETEXT, UPDATETEXT - считывают и обновляют значения text, ntext или image в столбцах типа text, ntext или image. В будущей версии Microsoft SQL Server эти компоненты будут удалены.

Инструкция BULK INSERT рассматривалась в ЛР № 1. Инструкции READTEXT, WRITETEXT, UPDATETEXT в учебном курсе рассматриваться не будут в виду сделанного замечания. Предметом данной ЛР будут инструкция

Извлечение данных: инструкция SELECT

Инструкция SELECT имеет наибольшее значение для конечного пользователя. Полный синтаксис инструкции SELECT сложен, однако основные предложения можно вкратце описать следующим образом:

[WITH *обобщенное_табличное_выражение*]
SELECT [DISTINCT | ALL] [TOP *выражение* [PERCENT]] { * | *список_выбора* } [INTO *новая_таблица*]
[FROM *список_табличных_источников*]
[WHERE *условие_поиска*]
[GROUP BY *group_by_выражение*]
[HAVING *условие_поиска*]
[ORDER BY *order_выражение* [ASC | DESC]]

Порядок предложений в инструкции SELECT имеет значение. Любое из необязательных предложений может быть опущено; но если необязательные предложения используются, они должны следовать в определенном порядке. При обработке инструкции SELECT составляющие ее предложения выполняются в следующем порядке:

1) FROM
2) ON

3) JOIN
4) WHERE
5) GROUP BY
6) HAVING
7) SELECT
8) DISTINCT
9) ORDER BY
10) TOP

Предложения, составляющие инструкцию SELECT, имеют следующий смысл.

Предложение WITH задает временно именованный результирующий набор, называемый обобщенным табличным выражением (ОТВ). Он получается при выполнении простого запроса и определяется в области выполнения одиночной инструкции SELECT, INSERT, UPDATE, MERGE или DELETE. Это предложение может использоваться также в инструкции CREATE VIEW как часть определяющей ее инструкции SELECT. Обобщенное табличное выражение может включать ссылки на само себя. Такое выражение называется рекурсивным обобщенным табличным выражением.

Предложение SELECT указывает столбцы, возвращаемые запросом. Выражение SELECT может содержать следующие аргументы:

- ALL - указывает, что в результирующем наборе могут появиться повторяющиеся строки. (Является значением по умолчанию.)
- DISTINCT - указывает, что в результирующем наборе могут появиться только уникальные строки.
- TOP (*выражение*) [PERCENT] - указывает на то, что только заданное число или процент строк будет возвращен из результирующего набора запроса. Аргумент *выражение* должен быть целого типа.
- *список_выбора* - столбцы, выбираемые для результирующего набора. Элементом списка выбора может быть имя столбца, скалярная функция, скалярный подзапрос, скалярное выражение с указанием или без указания псевдонима.
- *(звездочка) - указывает на то, что все столбцы из всех таблиц и представлений в предложении FROM должны быть возвращены.

Столбцы возвращаются таблицей или представлением, как указано в предложении FROM, и в порядке, в котором они находятся в таблице или представлении.В целях избежания неоднозначности ссылок, которые могут возникнуть, если в двух таблицах из предложения FROM содержатся столбцы с одинаковыми именами, следует указывать квалификатор для имени столбца.

Предложение INTO в инструкции SELECT создает новую таблицу и вставляет в нее результирующие строки из запроса.

Предложение FROM указывает список табличных источников, где каждый табличный источник может быть именем таблицы, именем представления, табличной переменной, табличной функцией, *joined_таблицей*, *pivoted_таблицей*, табличным подзапросом с указанием или без указания псевдонима. Важно помнить, что производительность выполнения запросов может снизиться из-за большого количества таблиц, указанных предложении FROM.

Важным случаем табличного источника является *joined_таблица* - результирующий набор, полученный из двух или более таблиц. Для множественных соединений следует использовать скобки, чтобы изменить естественный порядок соединений. Упрощенный синтаксис *joined_таблицы* имеет вид:

joined_таблица ::=
левая_таблица CROSS JOIN *правая_таблица* |
левая_таблица [INNER | {{ LEFT | RIGHT | FULL } [OUTER] }] JOIN *правая_таблица*
ON *условие_поиска* |
(*joined_таблица*)

Здесь

CROSS JOIN - указывает произведение двух таблиц.

JOIN - указывает, что данная операция соединения должна произойти между указанными источниками или представлениями таблицы.

INNER - указывает, что возвращаются все совпадающие пары строк. Несовпадающие строки из обеих таблиц отбрасываются. Если тип соединения не указан, этот тип задается по умолчанию.

FULL [OUTER] - указывает, что в результирующий набор включаются строки как из левой, так и из правой таблицы, несоответствующие условиям соединения, а выходные столбцы, соответствующие оставшейся таблице, устанавливаются в значение NULL. Этим дополняются все строки, обычно возвращаемые при помощи INNER JOIN.

LEFT [OUTER] - указывает, что все строки из левой таблицы, не соответствующие условиям соединения, включаются в результирующий набор, а выходные столбцы из оставшейся таблицы устанавливаются в значение NULL в дополнение ко всем строкам, возвращаемым внутренним соединением.

RIGHT [OUTER] - указывает, что все строки из правой таблицы, не соответствующие условиям соединения, включаются в результирующий набор, а выходные столбцы, соответствующие оставшейся таблице, устанавливаются в значение NULL в дополнение ко всем строкам, возвращаемым внутренним соединением.

ON *условие_поиска* - задает условие, на котором основывается соединение. Условие может указывать любой предикат, хотя чаще используются столбцы и операторы сравнения.

Предложение WHERE определяет условие поиска строк, возвращаемых запросом.

Предложение GROUP BY задает группы, в которые должны быть помещены строки вывода. Если в список выбора предложения SELECT включены статистические функции, то предложение GROUP BY вычисляет сводные значения для каждой группы. Если задано предложение GROUP BY, то либо каждый столбец во всех выражениях в списке выбора должен включаться в список *group_by_выражения*, либо *group_by_выражение* должно точно соответствовать списку выбора. *group_by_выражение*, по которому выполняется группирование, представляет собой список столбцов или выражений, которые ссылаются на столбцы, возвращаемые предложением FROM. Псевдоним столбца, который определяется в списке выбора, **не может** использоваться для указания столбца группирования.

Предложение HAVING определяет условие поиска для группы.

Предложение ORDER BY указывает порядок сортировки для столбцов, возвращаемых инструкцией SELECT. Предложение ORDER BY может содержать следующие аргументы:

- *order_by_выражение* - указывает столбец или столбцы, по которым должна выполняться сортировка. Столбец сортировки может быть указан с помощью имени или псевдонима столбца или неотрицательного целого числа, представляющего позицию имени или псевдонима в списке выбора. Имена и псевдонимы столбцов могут быть дополнены именем таблицы или представления.
- ASC - Указывает, что значения в указанном столбце должны сортироваться по возрастанию, от меньших значений к большим значениям.
- DESC - Указывает, что значения в указанном столбце должны сортироваться по убыванию, от больших значений к меньшим.

Из перечисленных предложений инструкции SELECT наиболее часто используются SELECT, FROM и WHERE. Если исключить все опции предложения SELECT, то получается такая простая конструкция:

SELECT { * | *список_выбора* }
FROM *табличный_источник*
WHERE *условие_поиска*

Примеры использования инструкции SELECT можно найти по адресу
https://msdn.microsoft.com/ru-ru/library/ms187731(v=sql.120).aspx

Добавление новых строк: инструкция INSERT
https://msdn.microsoft.com/ru-ru/library/ms174335(v=sql.120).aspx

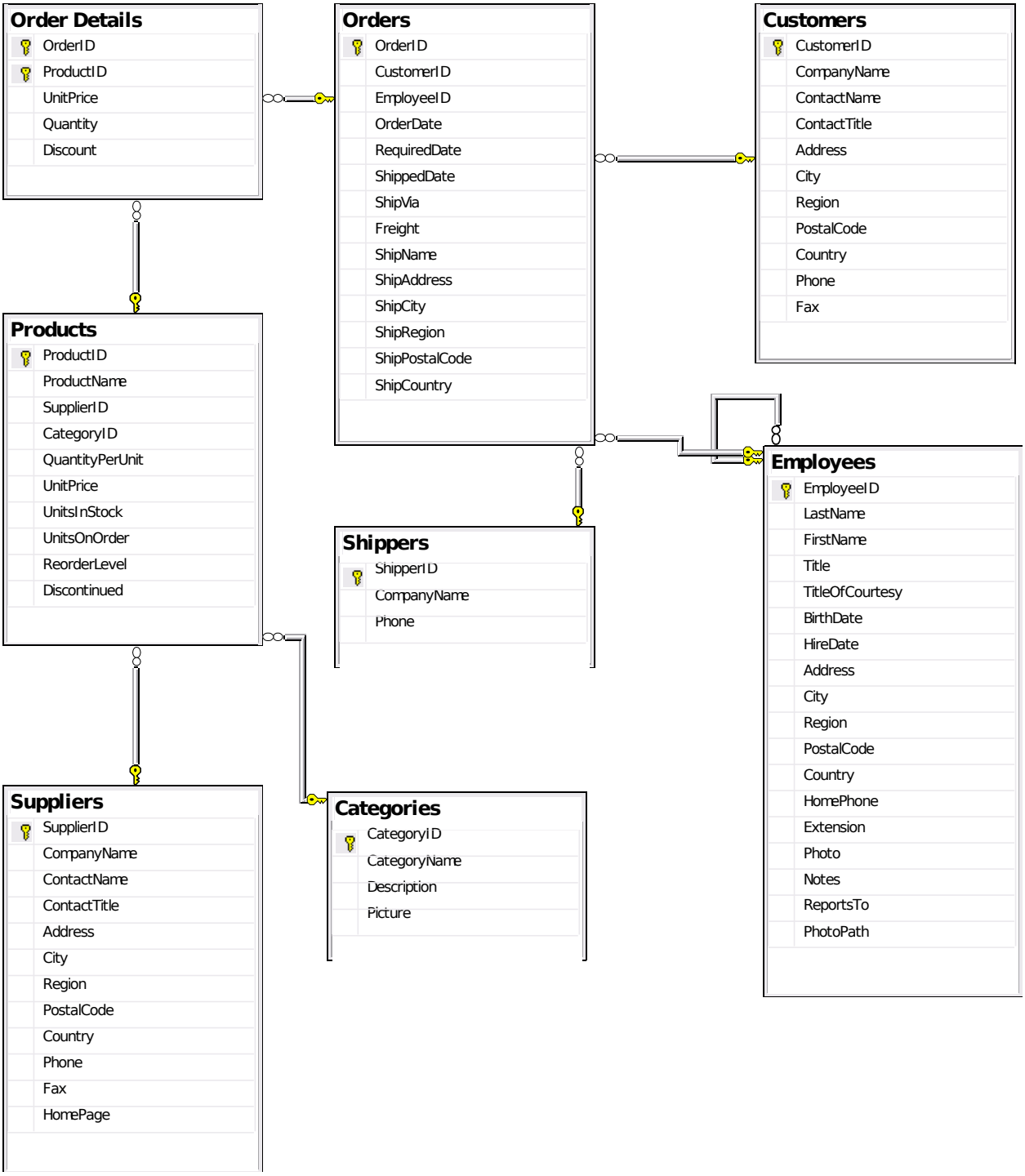
Изменение существующих данных: инструкция UPDATE
https://msdn.microsoft.com/ru-ru/library/ms177523(v=sql.120).aspx

Удаление данных: инструкция DELETE
https://msdn.microsoft.com/ru-ru/library/ms189835(v=sql.120).aspx

Вставка, обновление или удаление: инструкция MERGE
https://msdn.microsoft.com/ru-ru/library/bb510625(v=sql.120).aspx

Практическая часть

В практической части представлены примеры использования инструкций SELECT, INSERT, UPDATE и DELETE для демонстрационной базы данных Northwind. Упрощенная диаграмма базы данных Northwind изображена на рисунке.



1. **Инструкция SELECT, использующая предикат сравнения.**

```
SELECT DISTINCT C1.City, C1.CompanyName
FROM Customers C1 JOIN Customers AS C2 ON C2.City = C1.City
WHERE C2.CustomerID <> C1.CustomerID AND C1.Country = 'Argentina'
ORDER BY C1.City, C1.CompanyName
```

2. **Инструкция SELECT, использующая предикат BETWEEN.**

```
-- Получить список клиентов, сделавших заказы между '1997-01-01' и '1997-03-31'
SELECT DISTINCT CustomerID, OrderDate
FROM Orders
WHERE OrderDate BETWEEN '1997-01-01' AND '1997-03-31'
```

3. **Инструкция SELECT, использующая предикат LIKE.**

```
-- Получить список категорий продуктов в описании которых присутствует слово 'pasta'
SELECT DISTINCT CategoryName
FROM Categories JOIN Products ON Products.categoryID = Categories.CategoryID
WHERE Description LIKE '%pasta%'
```

4. **Инструкция SELECT, использующая предикат IN с вложенным подзапросом.**

```
-- Получить список заказов для клиентов из Лондона, оформленных через сотрудника 1
SELECT OrderID, CustomerID, EmployeeID, OrderDate
FROM Orders
WHERE CustomerID IN
	(
		SELECT CustomerID
		FROM Customers
		WHERE City = 'London'
	) AND EmployeeID = 1
```

5. **Инструкция SELECT, использующая предикат EXISTS с вложенным подзапросом.**

```
-- Получить список продуктов, которые никто никогда не заказывал
SELECT ProductID, ProductName
FROM Products
WHERE EXISTS
	(
		SELECT Products.ProductID
		FROM Products LEFT OUTER JOIN [Order Details]
			ON Products.ProductID = [Order Details].ProductID
		WHERE [Order Details].ProductID IS NULL
	)
```

6. **Инструкция SELECT, использующая предикат сравнения с квантором.**

```
-- Получить список продуктов, цена которых больше цены любого продукта категории 2
SELECT ProductID, ProductName, UnitPrice
FROM Products
WHERE UnitPrice > ALL
	(
		SELECT UnitPrice
		FROM Products
		WHERE CategoryID = 2
	)
```

7. **Инструкция SELECT, использующая агрегатные функции в выражениях столбцов.**

```
SELECT	AVG(TotalPrice) AS 'Actual AVG',
	SUM(TotalPrice) / COUNT(OrderID) AS 'Calc AVG'
FROM (
	SELECT OrderID, SUM(UnitPrice*Quantity*(1-Discount)) AS TotalPrice
	FROM [Order Details]
	GROUP BY OrderID
    )	AS TotOrders
```

8. **Инструкция SELECT, использующая скалярные подзапросы в выражениях столбцов.**

```
SELECT	ProductID,
	UnitPrice,
	(
		SELECT AVG(UnitPrice)
		FROM [Order Details]
		WHERE [Order Details].ProductID = Products.ProductID
	) AS AvgPrice,
	(
		SELECT MIN(UnitPrice)
```

```
                FROM [Order Details]
                WHERE [Order Details].ProductID = Products.ProductID
        ) AS MaxPrice,
        ProductName
FROM Products
WHERE CategoryID = 1
```

9. **Инструкция SELECT, использующая простое выражение CASE.**

```
SELECT CompanyName, OrderID,
        CASE YEAR(OrderDate)
                WHEN YEAR(Getdate()) THEN 'This Year'
                WHEN YEAR(GetDate()) - 1 THEN 'Last year'
                ELSE CAST(DATEDIFF(year, OrderDate, Getdate()) AS varchar(5)) + ' years ago'
        END AS 'When'
FROM Orders JOIN Customers ON Orders.CustomerID = Customers.CustomerID
```

10. **Инструкция SELECT, использующая поисковое выражение CASE.**

```
SELECT ProductName,
        CASE
                WHEN UnitPrice < 10 THEN 'Inexpensive'
                WHEN UnitPrice < 50 THEN 'Fair'
                WHEN UNitPrice < 100 THEN 'Expensive'
                ELSE 'Very Expensive'
        END AS Price
FROM products
```

11. **Создание новой временной локальной таблицы из результирующего набора данных инструкции SELECT.**

```
SELECT ProductID,
     SUM(Quantity) AS SQ,
     CAST(SUM(UnitPrice*Quantity*(1.0-Discount))AS money) AS SR
INTO #BestSelling
FROM [Order Details]
WHERE ProductID IS NOT NULL
GROUP BY productID
```

12. **Инструкция SELECT, использующая вложенные коррелированные подзапросы в качестве производных таблиц в предложении FROM.**

```
SELECT 'By units' AS Criteria, ProductName as 'Best Selling'
FROM Products P JOIN
        (
                SELECT TOP 1 ProductID, SUM(Quantity) AS SQ
                FROM [Order Details]
                GROUP BY productID
                ORDER BY SQ DESC
        ) AS OD ON OD.ProductID = P.ProductID
UNION
SELECT 'By revenue' AS Criteria, ProductName as 'Best Selling'
FROM Products P JOIN
        (
                SELECT TOP 1 ProductID, SUM(UnitPrice*Quantity*(1-Discount)) AS SR
                FROM [Order Details]
                GROUP BY ProductID
                ORDER BY SR DESC
        ) AS OD ON OD.ProductID = P.ProductID
```

13. **Инструкция SELECT, использующая вложенные подзапросы с уровнем вложенности 3.**

```
SELECT 'By units' AS Criteria, ProductName as 'Best Selling'
FROM Products
```

```
WHERE ProductID =
	(
		SELECT ProductID
		FROM [Order Details]
		GROUP BY ProductID
		HAVING SUM(Quantity) =
			(
				SELECT MAX(SQ)
			FROM
					(
						SELECT SUM(Quantity) as SQ
					FROM [Order Details]
					GROUP BY ProductID
					) AS OD
			)
	)
UNION
SELECT 'By revenue' AS Criteria, ProductName as 'Best Selling'
FROM Products
WHERE ProductID =
	(
		SELECT ProductID
		FROM [Order Details]
		GROUP BY ProductID
		HAVING SUM(UnitPrice*Quantity*(1-Discount)) =
			(
				SELECT MAX(SQ)
				FROM
					(
						SELECT SUM(UnitPrice*Quantity*(1-Discount)) AS SQ
						FROM [Order Details]
						GROUP BY ProductID
					) AS OD
			)
	)
```

14. **Инструкция SELECT, консолидирующая данные с помощью предложения GROUP BY, но без предложения HAVING.**

```
-- Для каждого заказанного продукта категории 1 получить его цену, среднюю цену,
-- минимальную цену и название продукта
SELECT P.ProductID,
	P.UnitPrice,
	AVG(OD.UnitPrice) AS AvgPrice,
	MIN(OD.UnitPrice) AS MinPrice,
	P.ProductName
FROM Products P LEFT OUTER JOIN [Order Details] OD ON OD.ProductID = P.ProductID
WHERE CategoryID = 1
GROUP BY P.productID, P.UnitPrice, P.ProductName
```

15. **Инструкция SELECT, консолидирующая данные с помощью предложения GROUP BY и предложения HAVING.**

```
-- Получить список категорий продуктов, средняя цена которых больше общей средней
-- цены продуктов
SELECT CategoryID, AVG(UnitPrice) AS 'Average Price'
FROM Products P
GROUP BY CategoryID
HAVING AVG(UnitPrice) >
	(
		SELECT AVG(UnitPrice) AS MPrice
		FROM Products
	)
```

16. **Однострочная инструкция INSERT, выполняющая вставку в таблицу одной строки значений.**

```
INSERT Products (ProductName, SupplierID, CategoryID, QuantityPerUnit, ReorderLevel, Discontinued)
VALUES ('Donut', NULL, NULL, '6 pieces', DEFAULT, DEFAULT)
```

17. **Многострочная инструкция INSERT, выполняющая вставку в таблицу результирующего набора данных вложенного подзапроса.**

```
INSERT [Order Details] (OrderID, ProductID, Unitprice, Quantity, Discount)
SELECT (
                SELECT MAX(OrderID)
                FROM Orders
                WHERE CustomerID = 'ALFKI'
        ), ProductID, UnitPrice, 10, 0.1
FROM Products
WHERE ProductName = 'Tofu'
```

18. **Простая инструкция UPDATE.**

```
UPDATE Products
SET UnitPrice = UnitPrice * 1.5
WHERE ProductID = 35
```

19. **Инструкция UPDATE со скалярным подзапросом в предложении SET.**

```
UPDATE Products
SET UnitPrice =
        (
                SELECT AVG(UnitPrice)
                FROM [Order Details]
                WHERE ProductID = 37
        )
WHERE ProductID = 37
```

20. **Простая инструкция DELETE.**

```
DELETE Orders
WHERE CustomerID IS NULL
```

21. **Инструкция DELETE с вложенным коррелированным подзапросом в предложении WHERE.**

```
-- Пример для базы данных AdventureWorks
DELETE FROM Production.Product
WHERE ProductID IN
(
                SELECT Product.ProductID
                FROM Production.Product LEFT OUTER JOIN Sales.SalesOrderDetail
                ON Product.ProductID = SalesOrderDetail.ProductID
                WHERE SalesOrderDetail.ProductID IS NULL
                        AND Product.ProductSubCategoryID = 5
        )
```

22. **Инструкция SELECT, использующая простое обобщенное табличное выражение**

```
-- Пример для базы данных SPJ
WITH CTE (SupplierNo, NumberOfShips)
AS
(
   SELECT Sno, COUNT(*) AS Total
   FROM SPJ
   WHERE Sno IS NOT NULL
   GROUP BY Sno
)
SELECT AVG(NumberOfShips) AS 'Среднее количество поставок для поставщиков'
FROM CTE
```

23. **Инструкция SELECT, использующая рекурсивное обобщенное табличное выражение.**

```
-- Создание таблицы.
CREATE TABLE dbo.MyEmployees
(
        EmployeeID smallint NOT NULL,
        FirstName nvarchar(30)  NOT NULL,
        LastName  nvarchar(40) NOT NULL,
        Title nvarchar(50) NOT NULL,
        DeptID smallint NOT NULL,
        ManagerID int NULL,
        CONSTRAINT PK_EmployeeID PRIMARY KEY CLUSTERED (EmployeeID ASC)
) ;
GO
-- Заполнение таблицы значениями.
INSERT INTO dbo.MyEmployees VALUES (1, N'Иван', N'Петров', N'Главный исполнительный
директор',16,NULL) ;
...
GO
-- Определение ОТВ
WITH DirectReports (ManagerID, EmployeeID, Title, DeptID, Level)
AS
(
-- Определение закрепленного элемента
   SELECT e.ManagerID, e.EmployeeID, e.Title, e.DeptID, 0 AS Level
   FROM dbo.MyEmployees AS e
   WHERE ManagerID IS NULL
   UNION ALL
-- Определение рекурсивного элемента
   SELECT e.ManagerID, e.EmployeeID, e.Title, e.DeptID, Level + 1
   FROM dbo.MyEmployees AS e INNER JOIN DirectReports AS d ON e.ManagerID = d.EmployeeID
)
-- Инструкция, использующая ОТВ
SELECT ManagerID, EmployeeID, Title, DeptID, Level
FROM DirectReports ;
```

24. **Оконные функции. Использование конструкций MIN/MAX/AVG OVER()**

```
-- Для каждой заданной группы продукта вывести среднее значение цены
SELECT P.ProductID,
        P.UnitPrice,
        P.ProductName,
        OD.UnitPrice,
        AVG(OD.UnitPrice) OVER(PARTITION BY P.productID, P.ProductName) AS AvgPrice,
        MIN(OD.UnitPrice) OVER(PARTITION BY P.productID, P.ProductName) AS MinPrice,
        MAX(OD.UnitPrice) OVER(PARTITION BY P.productID, P.ProductName) AS MaxPrice,
FROM Products P LEFT OUTER JOIN [Order Details] OD ON OD.ProductID = P.ProductID
```

25. **Оконные фнкции для устранения дублей**

Придумать запрос, в результае которого в данных появляются полные дубли. Устранить дублирующиеся строки с использованием функции ROW_NUMBER()

Дополнительное задание на дополнительные баллы

Создать таблицы:

- Table1{id: integer, var1: string, valid_from_dttm: date, valid_to_dttm: date}
- Table2{id: integer, var2: string, valid_from_dttm: date, valid_to_dttm: date}

Версионность в таблицах непрерывная, разрывов нет (если valid_to_dttm = '2018-09-05', то для следующей строки соответсвующего ID valid_from_dttm = '2018-09-06', т.е. на день больше). Для каждого ID дата начала версионности и дата конца версионности в Table1 и Table2 совпадают.
Выполнить версионное соединение двух талиц по полю id.

Пример:

Table1:

id	var1	valid_from_dttm	valid_to_dttm
1	A	2018-09-01	2018-09-15
1	B	2018-09-16	5999-12-31

Table2:

id	var2	valid_from_dttm	valid_to_dttm
1	A	2018-09-01	2018-09-18
1	B	2018-09-19	5999-12-31

Результат:

id	var1	var2	valid_from_dttm	valid_to_dttm
1	A	A	2018-09-01	2018-09-15
1	B	A	2018-09-16	2018-09-18
1	B	B	2018-09-19	5999-12-31