Лабораторная работа 1

Цель работы: Изучение базового синтаксиса инструкции SELECT, выражения WHERE и ORDER BY.

Базовый синтаксис инструкции SELECT

Общий синтаксис инструкции SELECT весьма сложен, однако достаточное представление дает следующее описание:

```
SELECT лист выборки [INTO новая таблица ]
[ FROM источник данных ] [ WHERE условия поиска ]
[ GROUP BY выражение для группировки ]
[ HAVING условия поиска по группам ]
[ ORDER BY выражение для сортировки [ ASC | DESC ] ]
```

В данном описании надо учитывать, что курсивом указаны пользовательские параметры синтаксиса языка T-SQL, а в угловых скобках, [], необязательные элементы синтаксиса. Не надо путать использование угловых скобок [] в описании синтаксиса, и при использование угловых скобок в указании идентификаторов. Фигурные скобки, {}, говорят о том, что пользователь обязан выбрать один из вариантов выражений, перечисленных в скобках через символ вертикальная черта, |. Традиционно все ключевые слова T-SQL пишутся в верхнем регистре, т.е. прописными буквами, но это правило не строгое, язык T-SQL регистронезависимый.

Инструкция следующего вида вполне допустима:

```
SELECT 'мама мыла раму'
```

В результате выполнения данной инструкции пользователь получит таблицу (результат выполнения инструкции SELECT всегда является таблицей) состоящую из одной строки, и одного столбца, при этом столбец не будет иметь имени.

Следующее изменение инструкции позволит получить таблицу с именованным столбцом:

```
SELECT 'мама мыла раму' AS [просто текст]
```

В данном случае ключевое слово AS говорит о том, что далее будет следовать псевдоним для столбца. Псевдонимы могут быть у столбцов, таблиц и результатов выполнения запросов. Псевдоним, в данном примере [*просто текст*], всегда должен подчиняться правилу идентификаторов (начинаться с буквы, возможно, использовать цифры, все символы без пробелов), если же, по каким то причинам, правило идентификаторов нарушается, то необходимо использовать квадратные скобки. Существует общая рекомендация повсеместного использования квадратных скобок, но на практике их опускают ради повышения скорости набора кода.

В качестве источника данных в общем случае выступают таблицы базы данных. Так же в качестве источника данных могут выступать отдельные таблицы, результат объединения таблиц, представления, табличные переменные, обобщенные табличные выражения, производные таблицы.

Пример выборки всего содержимого из таблицы:

```
SELECT *
FROM [Production].[Product]
```

Необходимо дать несколько пояснений. Символ звездочка, *, говорит о том, что из таблицы должны быть выбраны все столбцы. Использовать подобную конструкцию надо с крайней осторожностью, так как она создает значительную нагрузку на сервер и сетевую инфраструктуру. В случае если используется выборка из нескольких таблиц использование «звездочки» может привести к ошибке, так как в результате объединения таблиц может возникнуть ситуация наличия двух и более столбцов с одним именем в результирующем множестве (данный вопрос будет рассмотрен в последующих разделах). Конструкция [Production].[Product] представляет собой полное имя таблицы, где имя таблицы [Product], а [Production] это название схемы. Схема, это поименованное логическое объединение таблиц, используется для упрощения понимания архитектуры базы данных.

Для получения данных из столбца, необходимо указать соответствующее имя столбца, например запрос возвращающий все названия продуктов из таблиц [Product]:

```
SELECT [Name]

FROM [Production].[Product]
```

Рассмотрим еще один вариант написания запроса:

```
SELECT p.[Name]

FROM [Production].[Product] AS p
```

В данном случае для таблицы Product из схемы Production введен псевдоним p. Разумное использование псевдонимов позволяет увеличить скорость написания кода в MS SQL Management Studio.

Следующая инструкция позволяет получить таблицу с двумя столбцами:

```
SELECT [ProductID], [Name]

FROM [Production].[Product]
```

Рассмотрим запрос:

```
SELECT [ProductID], [Name], [ListPrice]-[StandardCost] AS
[discount size]

FROM [Production].[Product]
```

В этом примере пользователь получает таблицу из трех столбцов, данные первых двух столбцов взяты непосредственно из таблицы [Product], а третий столбец является результатом выполнения операции вычитания данных одного столбца из данных другого столбца, с использованием псевдонима, для упрощения понимания результата.

Еще один вариант:

```
SELECT [ProductID], [Name], 'empty column', GETDATE()

FROM [Production].[Product]
```

Пользователь может так же получать не только столбцы данных из таблицы, но и столбец с константным значением, третий столбец, а так же столбец содержащий результат выполнения функции, в данном случае в четвертом столбце выводиться текущее время сервера, которое возвращает функция GETDATE():

Обратите внимание на формат даты и времени, которые вернул сервер. Формат зависит от локальных настроек сервера, и соответствует одному из нескольких десятков поддерживаемых стандартов.

По устаревшим стандартам требовалось использование ключевого слова ALL:

```
SELECT ALL [Size]

FROM [Production].[Product]
```

Ключевое слово ALL говорит о том, что в результат выборки войдут все возможные значения столбца [Size], эта конструкция в текущей версии T-SQL является конструкцией по умолчанию, и на практике всегда опускается.

Получение данных без повторений:

```
SELECT DISTINCT [Size]

FROM [Production].[Product]
```

Ключевое слово DISTINCT определяет, что в выборке будут данные из столбца [Size] без повторений.

```
SELECT [Color], [Size]

FROM [Production].[Product]
```

Этот запрос вернет все существующие в таблице пары значений [Color] и [Size], без повторений.

Выражение ORDER BY

Учебные примеры могут дать ложную иллюзию того, что данные, получаемые в результате выполнения инструкции SELECT находятся в каком-то упорядоченном виде. Рассмотренный ранее запрос, SELECT [Name] FROM [Production].[Product], действительно возвращает название всех продуктов в упорядоченном по алфавиту виде, но это всего лишь сечение обстоятельств. В общем случае запрос возвращает данные в том порядке, который определен их физическим расположением в файлах БД, и операциями которые выполнила СУБД для их получения. Однако есть инструкция, позволяющая получить упорядоченный набор данных.

Упорядоченный вывод:

```
SELECT [Name], [ListPrice]

FROM [Production].[Product]

ORDER BY [ListPrice] DESC
```

Инструкция ORDER BY определяет порядок вывода строк, в данном случае будут выведены: названия продукта, его цена в виде упорядоченной по цене, по убыванию, таблицы. Ключевое слово DESC определяет порядок упорядочивания по убыванию, ASC – по возрастанию. Если направление упорядочивания не указано явно, то упорядочивание будет произведено по возрастанию. Инструкция ORDER BY всегда является завершающей частью запроса.

Упорядочение по нескольким столбцам, с указанием направления для каждого столбца:

```
SELECT [FirstName], [MiddleName], [LastName]

FROM [Person].[Person]

ORDER BY [FirstName] ASC, [MiddleName] DESC, [LastName] ASC
```

Упорядочение по позиции в выводе, в данном примере по второму столбцу:

```
SELECT [Name], [StandardCost] - [ListPrice]
FROM [Production].[Product]
ORDER BY 2
```

Так же возможно использование функции, для получения параметра для упорядочивания:

```
SELECT BusinessEntityID, JobTitle, HireDate
FROM HumanResources.Employee
ORDER BY DATEPART(year, HireDate)
```

Рассмотрим следующий пример:

```
SELECT TOP 3 [Name], [ListPrice]
FROM [Production].[Product]
ORDER BY [ListPrice] DESC
```

Инструкция TOP ограничивает количество строк в результирующем наборе, таблице, в примере соответственно будут выведены три продукта, и их цены, в отсортированном наборе. Необходимо сделать замечание, что возможно существует некоторый продукт, условно говоря, четвертый, цена которого равна цене третьего продукта, но он не попадет в выборку для пользователя. К сожалению, мы не можем заранее определить какой из продуктов, третий или четвертый, с равной ценой, в отсортированном списке, попадет в пользовательскую выборку при использовании конструкции TOP 3.

Инструкция WITH TIES, позволяет выводить все «соперничающие, за последнее место, строки»:

```
SELECT TOP 3 WITH TIES [Name], [ListPrice]
FROM [Production].[Product]
ORDER BY [ListPrice] DESC
```

Приведенный пример использования инструкции WITH TIES позволяет вывести на экран не только первые три продукта из набора, отсортированного по убыванию цены, но и все продукты, цена которых равна цене третьего продукта. Данная инструкция работает исключительно с SELECT, и только при использовании оператора упорядоченного вывода ORDER BY.

Вывод доли строк из результирующего набора:

```
SELECT TOP 3 PERCENT [Name], [ListPrice]
FROM [Production].[Product]
ORDER BY [ListPrice] DESC
```

В случае использования конструкции TOP *число* PERCENT, пользователь получает некоторый процент строк, равный *числу*, из результирующего набора. Процент строк округляется до следующего целого.

Использование инструкции INTO позволяет создать новую таблицу, и поместить в нее результат выборки:

```
SELECT TOP 3 PERCENT [Name], [ListPrice] INTO Tmp

FROM [Production].[Product]

ORDER BY [ListPrice] DESC
```

Выполнения данного запроса будет разбито на два этапа. На первом этапе будет создана таблица Tmp. На втором этапе в нее будет помещен результат выполнения запроса. Необходимо добавить несколько замечаний. Если по каким-то причинам при выполнении запроса произойдет сбой, то будет создана пустая таблица. Использование инструкции упорядоченного выводы выборки, ORDER BY, не гарантирует сохранения порядка строк в созданной таблице. В случае, если результирующий набор дает набор строк, не обладающий свойством уникальности, то все равно таблица будет создана, данные будут добавлены, но никакие первичные ключи пользователю не будут доступны, хотя на физическом уровне они будут реализованы СУБД. Для использования инструкции INTO необходимо иметь права, аналогичные правам создания обычной таблицы.

Выражение WHERE

Инструкция SELECT не позволяет выбрать отдельные строки из источника, но можно выбрать строки, удовлетворяющие определенным условиям, наличию тех или иных значений в столбцах.

Простой пример использование предложения WHERE:

```
SELECT [Name], [ListPrice]

FROM [Production].[Product]

WHERE [ListPrice]=3578.27
```

В данном случае будут выведены названия продуктов, а также их цена, у которых цена равна 3578.27. Так как столбец [ListPrice] имеет тип money, то использование оператора сравнения на равенство допустимо.

В следующем примере ограничения затрагивают два столбца:

```
SELECT [Name], [ListPrice]

FROM [Production].[Product]

WHERE [ListPrice]=3578.27 AND [Size]=62
```

Пользователь получит название таких товаров и их цену, у которых цена равна 3578.27 и размер равен 62, для этого был использован оператор AND – логическое И. Существует еще два логических оператора, OR -логическое ИЛИ и NOT – логическое отрицание.

Можно написать достаточно сложный запрос:

```
SELECT [Name] FROM [Production].[Product]

WHERE

([ListPrice]>20 AND [Color]='Red')

OR

([ListPrice]>25 and [Color]='Black')
```

В данном случае пользователь получит название таких товаров, у которых цена выше 20 и цвет Red, или цена выше 25 и цвет Black.

При работе с числовыми данными бывают ситуации, когда необходимо использовать некоторый диапазон данных. Например, стоит вопрос – найти названия товаров, цена которых лежит в диапазоне от 20 до 40, включая границы диапазона. Реализовать этот запрос модно с помощью логического оператора AND.

```
SELECT [Name] FROM [Production].[Product]

WHERE [ListPrice]>=20 AND [ListPrice]<=40
```

Однако существует специальный оператор BETWEEN, который определяет диапазон для проверки. Общий синтаксис этого оператора выглядит так:

выражение [NOT] BETWEEN *начало_диапазона* AND *конец_диапазона*

Оператор возвращает значение TRUE если *выражение* входит, или не входит, если используется оператор NOT, в диапазон от *начало_диапазона* до *конец_диапазона* включительно.

С помощью этого оператора предыдущий запрос можно реализовать следующим образом:

```
SELECT [Name] FROM [Production].[Product]

WHERE [ListPrice]>=20 AND [ListPrice]<=40
```

Если стоит задача сравнения выражения с некоторым набором значений, то могут применяться следующие операторы – IN, ANY (или его синоним - SOME), ALL.

Оператор IN определяет принадлежность значения одному и значений в списке.

```
SELECT [Name] FROM [Production].[Product]

WHERE [Color] in ('Red','Black','Silver')
```

Приведенный выше запрос возвращает название товаров, у которых цвет либо Red, либо Black, либо Silver. Стоит обратить внимание что в скобках может быть не просто набор значений, но и выражение, которое формирует такой набор, например оператор SELECT.

Операторы ALL и ANY будут рассмотрены позже, в работе с подзапросами.

При работе со строковыми значениями возникает необходимость проверки на соответствие значение не строке, а шаблону, например требуется найти все продукты, начинающиеся на букву ‘D’. Для этого используется оператор LIKE:

```
SELECT [Name] FROM [Production].[Product]

WHERE [Name] LIKE 'D%'
```

При использовании оператора LIKE можно использовать оператор NOT. Необходимо помнить, что что существуют параметры сравнения, определяемые настройкой СУБД и отдельных таблиц, но в общем случае СУБД не делает различия между строчными и прописными буквами.

В строковых шаблонах допускаются следующие символы:

% - символ-шаблон, заменяющий любую последовательность символов;

_ (подчеркивание) – символ-шаблон, заменяющий любой одиночный символ;

[] – заменяет одиночный символ, указанный в угловых скобках, можно перечислить символы, или диапазон (через дефис) символов;

[^] – заменяет одиночный символ, не указанный в угловых скобках, можно перечислить символы, или диапазон (через дефис) символов.

Допускается использование ESCAPE последовательностей:

WHERE СТРОКА LIKE ‘%[a-f][^xyz]_30!%%’ ESCAPE ‘!’

Данный пример является частью запроса, который в частности проверяет совпадают ли значения в столбце СТРОКА со следующим шаблоном – любое количество символов, далее один из символов диапазоне от ‘a’ до ‘f’ включительно, далее любой символ кроме символов ‘x’, ‘y’ или ‘z’, далее еще один любой символ, далее 30%, и опять последовательность любых символов. Символ ‘!’ является эскейп символом, и говорит о том, что следующий за ним символ, в данном случае ‘%’, не надо рассматривать как управляющий, таким образом в пример включен символ %, который в общем случае является служебным.

В реляционных базах данных существует особое значение – NULL. Это значение не является нулевым значением, в математическом понимании нуля, это неопределённость, которое сообщает пользователям что в данным момент времени значение атрибута не определено. Для работы с этим значением нельзя использовать оператор сравнения на равенство или любой другой оператор сравнения. Проверка на определенность осуществляется с помощью оператора IS.

```
SELECT [Name] FROM [Production].[Product]

WHERE [Color] IS NULL
```

Данный запрос возвращает название товаров, для которых цвет не определен.

```
SELECT [Name] FROM [Production].[Product]

WHERE [Size] IS NOT NULL
```

Этот запрос возвращает название товаров название товаров у которых определен размер, т.е. он не является NULL.

Примеры запросов с ответами

Описание функций и архитектура приведены в приложении.

1. Получить все названия товаров из таблицы Product

```
select p.Name
from [Production].[Product] as p
```

В данном случае для таблицы Product из схемы Production введен псевдоним p. Разумное использование псевдонимов позволяет увеличить скорость написания кода в MS SQL Management Studio

2. Получить все названия товаров в системе цена которых (listprice) выше 200

```
select p.Name
from [Production].[Product] as p
where p.ListPrice>200
```

3. Получить все названия товаров в системе цена которых (listprice) выше 200 и у которых первая буква в названии “S”

```
select p.Name
from [Production].[Product] as p
where p.ListPrice>200 and p.Name like 's%'
```

4. Получить все названия товаров в системе цена которых (listprice) выше 200 и у которых в названии есть сочетание символов “are”

```
select p.Name
from [Production].[Product] as p
where p.ListPrice>200 and p.Name like '%are%'
```

5. Получить все названия товаров в системе, в названии которых третий символ либо буква “s”, либо буква “r”. Решить задачу как минимум двумя способами.

```
select p.Name
from [Production].[Product] as p
where p.Name like '__s%' or p.Name like '__r%'

select p.Name
from [Production].[Product] as p
where p.Name like '__[sr]%'
```

7. Получить все названия товаров в системе, в названии которых ровно 5 символов

```
select p.Name
from [Production].[Product] as p
where p.Name like '_____'

select p.Name
from [Production].[Product] as p
where len(p.Name)=5
```

Во втором примере используется строковая функция len. Строковые функции даны в приложении.

8. Найти, без повторений, номера товаров, которые были куплены хотя бы один раз

```
select distinct sod.ProductID
from [Sales].[SalesOrderDetail] as sod
```

9. Найти список всех возможных стилей (style) продукта, без повторений

```
select distinct p.Style
from [Production].[Product] as p
where p.Style is not null
```

10. Найти список товаров, названия, которые были произведены между мартом 2011 года и мартом 2012 года включительно. (необходимо учитывать формат даты)

```
select  p.Name
from [Production].[Product] as p
where p.SellStartDate>='2011-01-03'
and p.SellStartDate<='2012-31-03'
```

11. Найти максимальную стоимость товара, (отпускная цена ListPrice) были произведены начиная с марта 2011 года

```
select  max(p.ListPrice) as [max price]
from [Production].[Product] as p
where p.SellStartDate>='2011-01-03'
```

12. Вывести название продукта, и цвет продукта, отсортированные по названию продукта по возрастанию

```
select  p.Name, p.Color
from [Production].[Product] as p
order by p.Name asc
```

13. Вывести названия продукта, цвет и отпускную цену для таких товаров, у которых цвет определен, цена отлична от нуля, и отсортировать полученный список по возрастанию цвета товара, и убыванию отпускной цены

```
select  p.Name, p.Color, p.ListPrice
from [Production].[Product] as p
where p.Color is not null and p.ListPrice!=0
order by p.Color,  p.ListPrice desc
```

14. Получить название продукта и разницу между отпускной стандартной ценой продукта и стандартной ценой продукта, для тех товаров у которых эти показатели не равны нулю

```
select p.Name, p.ListPrice-p.StandardCost
from [Production].[Product] as p
where p.ListPrice!=0 and p.StandardCost!=0
```

15. Найти название самого дорогого товара, исходя из предположения что нет двух товаров с одинаковой ценой

```
select top 1 with ties p.Name
from [Production].[Product] as p
order by p.ListPrice desc
```

16. Найти список товаров произведенный в 2005 году

```
elect p.Name
from [Production].[Product] as p
where datepart(YEAR,p.SellStartDate)=2005
```

Задания для самостоятельного работы:

1. Найти и вывести на экран названия продуктов, их цвет и размер

2. Найти и вывести на экран названия, цвет и размер таких продуктов, у которых цена более 100

3. Найти и вывести на экран название, цвет и размер таких продуктов, у которых цена менее 100 и цвет Black

4. Найти и вывести на экран название, цвет и размер таких продуктов, у которых цена менее 100 и цвет Black, упорядочив вывод по возрастанию стоимости продуктов

5. Найти и вывести на экран название и размер первых трех самых дорогих товаров с цветом Black

6. Найти и вывести на экран название и цвет таких продуктов, для которых определен и цвет, и размер

7. Найти и вывести на экран не повторяющиеся цвета продуктов, у которых цена находится в диапазоне от 10 до 50 включительно

8. Найти и вывести на экран все цвета таких продуктов, у которых в имени первая буква ‘L’ и третья ‘N’

9. Найти и вывести на экран названия таких продуктов, которых начинаются либо на букву ‘D’ либо на букву ‘M’, и при этом длинна имени более трех символов

10. Вывести на экран названия продуктов, у которых дата начала продаж не позднее 2012 года

11. Найти и вывести на экран название всех подкатегорий товаров

12. Найти и вывести на экран названия всех категорий товаров

13. Найти и вывести на экран имена всех клиентов, из таблицы Person, у которых обращение (Title) указано как «Mr.»

14. Найти и вывести на экран имена всех клиентов, из таблицы Person, для которых не определено обращение (Title)

Приложения

Ниже приведена таблица со всеми логическими операторами, их использование будет иллюстрировано на практических примерах.

Оператор	Синтаксис	Комментарий
ALL	*скаляр* { = | <> | != | > | >= | !> | < | <= | !< } ALL (*подзапрос*)	Сравнивает *скалярное* значение с набором значений, находящиеся в столбце, который вернул *подзапрос*. *Скаляр* должен удовлетворять условию для всех значений.
AND	*выражение* AND *выражение*	Стандартная конъюнкция
SOME, ANY	*скаляр* { = | <> | != | > | >= | !> | < | <= | !< } { SOME | ANY } (подзапрос)	Сравнивает *скалярное* значение с набором значений, находящиеся в столбце который вернул *подзапрос*. *Скаляр* должен удовлетворять условию хотя бы одного из значений. SOME и ANY – эквиваленты.
BETWEEN	*выражение* [NOT] BETWEEN *выражение1* AND *выражение2*	Определяет диапазон для проверки, *выражение* должно находиться в диапазоне от *выражения1* до *выражения2* включительно. Выражения должны иметь один формат.
EXISTS	EXISTS(подзапрос)	Возвращает истину, если подзапрос возвращает хотя бы одно значение, в противном случае возвращает ложь. Особенность данного оператора в том, что подзапрос прекращается после нахождения первого значения, что существенно экономит ресурсы.
IN	*выражение* [NOT] IN (*подзапрос* | *список[,…n]*)	Определяет совпадает ли указанное значение *выражения* с одним из значений возвращаемым *подзапросом* или содержащимся в *списке*.
LIKE	*выражение* [NOT] LIKE *шаблон*	Определяет, совпадает ли символьное *выражение* с указанным *шаблоном*. Для уточнения параметров шаблона смотри пояснения после таблицы.
NOT	NOT *логическое выражение*	Стандартное логическое отрицание.
OR	*выражение* OR *выражение*	Стандартная дизъюнкция

В строковых шаблонах допускаются следующие символы:

% - символ-шаблон заменяющий любую последовательность символов;

_ (подчеркивание) – символ-шаблон заменяющий любой одиночный символ;

[] – заменяет одиночный символ указанный в угловых скобках, можно перечислить символы, или диапазон (через дефис) символов;

[^] – заменяет одиночный символ не указанный в угловых скобках, можно перечислить символы, или диапазон (через дефис) символов.

Допускается использование ESCAPE последовательностей:

WHERE СТРОКА LIKE ‘%[a-f][^xyz]_30!%%’ ESCAPE ‘!’

Данный пример является частью запроса, который в частности проверяет совпадают ли значения в столбце СТРОКА со следующим шаблоном – любое количество символов, далее один из символов диапазоне от ‘a’ до ‘f’ включительно, далее любой символ кроме символов ‘x’, ‘y’ или ‘z’, далее еще один любой символ, далее 30%, и опять последовательность любых символов. Символ ‘!’ является эскейп символом, и говорит о том, что следующий за ним символ, в данном случае ‘%’, не надо рассматривать как управляющий, таким образом в пример включен символ %, который в общем случае является служебным.

Строковые функции

Функция	Синтаксис	Комментарий
ASCII	ASCII(*строковое выражение*)	Функция возвращает код ASCII первого символа *строкового выражения*.
CHAR	CHAR(*числовое выражение*)	Преобразует код ASCII символа, *числовое выражение*, в символ.
CHARINDEX	CHARINDEX(строка для поиска, строка поиска [, *номер начала поиска*])	Функция выполняет поиск строки, *строка для поиска*, в строке, *строка поиска*. Можно указать номер символа, *номер начала поиска*, с которого начать поиск в строке, *строке поиска*. Функция возвращает номер позиции, *строки для поиска*, если таковая найдена.
CONCAT	CONCAT(*строка1, строка2*[,*строкаN*])	Возвращает строку, результат объединения строки1, строки2 … строкиN.
CONCAT_WS	CONCAT_WS(*разделитель , строка1, строка2*[,*строкаN*])	Возвращает строку, результат объединения *строки1, строки2 … строкиN*, разделенные *разделителем*.
DIFFERENCE	DIFFERENCE(*строка1, строка2*)	Возвращает число, разницу между значениями SOUNDEX *строки1* и *строки2*.
FORMAT	FORMAT(*значение, формат* [, *язык*])	Возвращает *значение* в указанном *формате*, возможно указание языкового/регионального параметра, *языка*.
LEFT	LEFT(*строка, число*)	Возвращает указанное *число* символов *строки* слева.

LEN	LEN(*строка*)	Возвращает длину *строки*.
LOWER	LOWER(*строка*)	Возвращает *строку*, все символы которой преобразованы в те же символы нижнего регистра
LTRIM	LTRIM(*строка*)	Возвращает *строку*, у которой удалены начальные пробелы.
NCHAR	NCHAR(*число*)	Возвращает символ Юникода с указанным номером.
PATINDEX	PATINDEX(*шаблон, выражение*)	Возвращает начальную позицию первого вхождения *шаблона* в *выражение*, или ноль, если такого нет. Выражение это строка или столбец.
QUOTENAME	QUOTENAME(*строка* [, *разделитель*])	Возвращает *строку* с *разделителями* в виде правильного идентификатора SQL Server.
REPLACE	REPLACE(*строка, шаблон, замена*)	Заменяет в *строке* все последовательности символов по *шаблону* на строку *замены*.
REVERSE	REVERSE(*строка*)	Возвращает строку, где символы переставлены в обратном порядке относительно *строки* параметра.
RIGHT	RIGHT(*строка, число*)	Возвращает указанное *число* символов *строки* справа.
RTRIM	RTRIM(*строка*)	Возвращает *строку*, в которой удалены все завершающие пробелы.
SOUNDEX	SOUNDEX(*строка*)	Возвращает четырехсимвольный код *строки*
SPACE	SPACE(*число*)	Возвращает строку из пробелов. Количество пробелов определяется *числом*.
STR	STR(*число*[, *длинна*[, *количество*]])	Возвращает символьные денные преобразованные из *числа*, заданной *длинны*, с заданным *количеством* знаков после запятой.
STRING_AGG	STRING_AGG(*выражение, разделитель*)	Сцепляет строковые *выражения*, используя *разделитель*. Допускается использование с GROUP BY.
STRING_ESCAPE	STRING_ESCAPE (*строка, тип*)	Возвращает *строку* с экранированными по *типу* символами.
STRING_SPLIT	STRING_SPLIT(*строка, разделитель*)	Возвращает таблицу, созданную из подстрок *строки* по *разделителю.*
STUFF	STUFF(*строка*, начало, *длинна*, *выражение*)	Вставляет одну *выражение* в *строку. Начало* определяет, с какого символа начнется вставка, и какова будет *длинна* удаленной подстроки.
SUBSTRING	SUBSTRING(*строка, начало, длинна*)	Возвращает подстроку, из *строки,* указанной *длинны* с позиции *начала*.
TRANSLATE	TRANSLATE(*аргумент1, аргумент2, аргумент3*)	Возвращает строку, представленную в качестве *аргумента1*, после преобразования

		символов, *аргумент2*, в конечный набор символов, *аргумент3*.
TRIM	TRIM([*символы* from] *строка*)	Удаляет пробелы, ли удаляемые *символы* из начала и конца *строки*.
UNICODE	UNICODE(*строка*)	Возвращает целочисленное значение соответствующее стандарту Юникод для первого символа *строки*.
UPPER	UPPER(*строка*)	Возвращает *строку*, преобразованную в строку, где все символы переведены в верхний регистр.

Функции работы с датой и временем

Функция	Синтаксис	Комментарий
SYSDATETIME	SYSDATETIME ()	Возвращает системное время компьютера, на котором запущен экземпляр СУБД. Возвращаемый тип данных - datetime2(7).
SYSDATETIMEOFFSET	SYSDATETIMEOFFSET ( )	Возвращает системное время компьютера, на котором запущен экземпляр СУБД. Возвращаемый тип данных - datetimeoffset(7).
SYSUTCDATETIME	SYSUTCDATETIME ( )	Возвращает системное время компьютера, на котором запущен экземпляр СУБД. Возвращаемый тип данных - datetime2(7) в формате UTC.
CURRENT_TIMESTAMP	CURRENT_TIMESTAMP	Возвращает системное время компьютера, на котором запущен экземпляр СУБД. Возвращаемый тип данных – datetime. Данная конструкция не является функцией, это эквивалент функции GETDATE ( ).
GETDATE	GETDATE ( )	Возвращает системное время компьютера, на котором запущен экземпляр СУБД. Возвращаемый тип данных – datetime.
GETUTCDATE	GETUTCDATE ( )	Возвращает системное время компьютера, на котором запущен экземпляр СУБД. Возвращаемый тип данных - datetime2 в формате UTC.
DATENAME	DATENAME (*часть* , *дата*)	Возвращает строку символов, которая является *частью даты*. Формат *части* даты: • year или yy, yyyy • quarter или qq, q • month или mm, m • dayofyear или dy, y • day или dd, d • week или wk, ww • weekday или dw, w • hour или hh

		• minute или mi, n • second или ss, s • millisecond или ms • microsecond или mcs • nanosecond или ns • TZoffset или tz • ISO_WEEK или ISOWK, ISOWW
DATEPART	DATEPART (*часть* , дата)	Возвращает целое число, являющееся *частью даты*. Формат *части* такой же, как и у функции DATENAME
DAY	DAY (*дата*)	Возвращает целое число, являющееся *частью даты*.
MONTH	MONTH (*дата*)	Возвращает целое число, являющееся *частью даты*.
YEAR	YEAR (*дата*)	Возвращает целое число, являющееся *частью даты*.
DATEFROMPARTS	DATEFROMPARTS (*год* , *месяц* , *день*)	Возвращает дату в формате date соответствующие *году*, *месяцу* и *дню*
DATETIME2FROMPARTS	DATETIME2FROMPARTS (*год*, *месяц*, *день*, *час*, *минуты*, *секунды*, *доли_секунд*, *точность*)	Возвращает дату в формате datetime2 соответствующие *году*, *месяцу*, *дню, часу, минуте, секунде, доли секунды* с заданной *точностью.*
DATETIMEFROMPARTS	DATETIMEFROMPARTS (*год*, *месяц*, *день*, *час*, *минуты*, *секунды*, *миллисекунды*)	Возвращает дату в формате datetime соответствующие *году*, *месяцу*, *дню, часу, минуте, секунде, миллисекунде.*
DATETIMEOFFSETFROMPARTS	DATETIMEOFFSETFROMPARTS (*год*, *месяц*, *день*, *час*, *минуты*, *секунды*, *доли_секунд*, смещение_в_часах, смещение_в_минутах, *точность*)	Возвращает дату в формате datetimeoffset соответствующие *году*, *месяцу*, *дню, часу, минуте, секунде, доли секунды* с заданной *точностью*, с учетом *смещения в часах и минутах.*
SMALLDATETIMEFROMPARTS	SMALLDATETIMEFROMPARTS (*год*, *месяц*, *день*, *час*, *минуты*)	Возвращает дату в формате smalldatetime соответствующие *году*, *месяцу*, *дню, часу, минуте.*
TIMEFROMPARTS	TIMEFROMPARTS (*час*, *минуты*, *секунды*, *доли_секунд*, *точность*)	Возвращает дату в формате time соответствующие *часу, минуте, секунде, доли секунды* с заданной *точностью.*
DATEDIFF	DATEDIFF (*часть* , начальная_дата , конечная_дата)	Возвращает целое число, int, разницу между *начальной датой* и *конечной датой* в *частях*. Формат *части* аналогичен формату части для функции DATENAME.
DATEDIFF_BIG	DATEDIFF_BIG (*часть* , начальная_дата , конечная_дата)	Возвращает целое число, bigint, разницу между *начальной датой* и *конечной датой* в *частях*. Формат *части* аналогичен формату части для функции DATENAME.

DATEADD	DATEADD (*часть* , *число* , *дата*)	Возвращаетс новую дату, типа datetime, соответствующею *дате* увеличенной на то количество *частей*, которое определено *числом*.
EOMONTH	EOMONTH (*дата* [, *число*])	Возвращает дату последнего дня того месяца, который содержит указанную *дату*. К указанной дате можно добавить *число*, определяющее на сколько месяцев сместить *дату*.
SWITCHOFFSET	SWITCHOFFSET (*дата* , *зона*)	Возвращает дату, типа datetimeoffset, соответствующую введенной *дате*, типа datetimeoffset, со смещением на часовую *зону*.
TODATETIMEOFFSET	TODATETIMEOFFSET (дата , зона)	Возвращает дату, типа datetimeoffset, соответствующую введенной *дате*, типа datetime2, со смещением на часовую *зону*.
ISDATE	ISDATE (*выражение*)	Возвращает 1 если *выражение* соответствует формату datetime или smalldatetime, и 0 во всех других случаях.

Модель данных