**Module 3**

**Writing SELECT Queries**

Оператор SELECT можно использовать для запроса таблиц и представлений. Вполне вероятно, что вы будете использовать оператор SELECT чаще, чем любой другой отдельный оператор в T-SQL. Вы можете манипулировать данными с помощью SELECT, чтобы настроить, как SQL Server возвращает результаты. Этот модуль знакомит вас с основами инструкции SELECT, фокусируясь на запросах к одной таблице.

**Lesson 1**

**Writing Simple SELECT Statements**

Elements of the SELECT Statement

Предложения SELECT и FROM являются основным фокусом этого модуля. Вы узнаете о других положениях в последующих модулях этого курса. Вы уже изучили порядок операций при обработке логических запросов; это поможет вам понять, как правильно формировать операторы SELECT.

Помните, что предложения FROM, WHERE, GROUP BY и HAVING оцениваются механизмом запросов перед содержимым предложения SELECT.

Поэтому элементы, которые вы пишете в предложении SELECT, особенно вычисляемые столбцы и псевдонимы, не будут видны другим предложениям.

**3-3**

**Retrieving Columns from a Table or View**

Предложение SELECT указывает столбцы из исходной таблицы(таблиц) или представления(представлений), которые вы хотите вернуть в качестве результирующего набора запроса. В дополнение к столбцам из исходной таблицы можно добавить другие в виде вычисляемых выражений.

Предложение FROM указывает имя таблицы или представления, которое является источником столбцов в предложении SELECT. Чтобы избежать ошибок в разрешении имен таблиц или представлений, лучше всего включить схему и имя объекта в СХЕМУ SCHEMA.OBJECT—for example Sales.Customer.

Если имя таблицы или представления содержит неправильные символы, такие как пробелы или другие специальные символы, необходимо разделить или заключить имя. T-SQL поддерживает использование стандартных двойных кавычек ANSI “Сведения о заказе на продажу” и квадратных скобок для SQL Server [Сведения о заказе на продажу].

Завершите все операторы символом точки с запятой (;). В SQL Server точки с запятой являются необязательным терминатором для большинства операторов. Однако в будущих версиях потребуется его использование. Для текущего использования, когда требуется точка с запятой, например, некоторые общие табличные выражения (CTE) и некоторые операторы Service Broker, сообщения об ошибках, возвращаемые для отсутствующей точки с запятой, часто являются загадочными. Поэтому вам следует придерживаться практики завершения всех операторов точкой с запятой.

**Displaying Columns**

Чтобы отобразить столбцы в запросе, необходимо создать список столбцов, разделенных запятыми. Порядок столбцов в вашем списке будет определять их отображение в выходных данных, независимо от порядка, в котором вы определили их в исходной таблице. Это дает вашим запросам возможность поглощать изменения, которые другие могут внести в структуру таблицы, такие как добавление или изменение порядка столбцов.

T-SQL поддерживает использование символа звездочки или “звезды” ( \* ) для замены явного списка столбцов. Это приведет к извлечению всех столбцов из исходной таблицы. Хотя звездочка подходит для быстрого тестирования, избегайте ее использования в производственной работе, так как изменения, внесенные в таблицу, приведут к тому, что запрос получит все текущие столбцы в текущем определенном порядке таблицы.

Это может привести к ошибкам или другим сбоям в отчетах или приложениях, ожидающих известное количество столбцов, возвращаемых в определенном порядке. Кроме того, возврат ненужных данных может замедлить выполнение запросов и вызвать проблемы с производительностью, если исходная таблица содержит большое количество строк.

Используя явный список столбцов в предложении SELECT, вы всегда достигнете желаемых результатов при условии, что столбцы существуют в таблице. Если столбец будет удален, вы получите сообщение об ошибке, которое поможет определить проблему и исправить запрос.

**Using Calculations in the SELECT Clause**

In addition to retrieving columns stored in the source table, a SELECT statement can perform calculations and manipulations. Calculations and manipulations can change the source column data, and use built-in T-SQL functions, which you will learn about later in this course.

As the results will appear in a new column, repeated once per row of the result set, calculated expressions in a SELECT clause must be scalar—they must return only a single value.

Calculated expressions can operate on other columns in the same row, on built-in functions, or a combination of the two:

**Calculated Expression**

SELECT unitprice, qty, (unitprice \* qty)

FROM Sales.OrderDetails;

The results appear as follows:

unitprice qty

----------- ---------- -----------------------

14.00 12 168.00

9.80 10 98.00

34.80 5 174.00

18.60 9 167.40

Note that the new calculated column does not have a name returned with the results. To provide a name, you use a column alias, which you will learn about later in this module.

To use a built-in T-SQL function on a column in the SELECT list, pass the name of the column to the function as an input:

**Create a Calculated Column**

SELECT empid, lastname, hiredate, YEAR(hiredate)

FROM HR.Employees;

The results:

empid lastname hiredate

---------- ------------ -------------------------- ---------

1 Davis 2002-05-01 00:00:00.000 2002

2 Funk 2002-08-14 00:00:00.000 2002

3 Lew 2002-04-01 00:00:00.000 2002

Вы узнаете больше о дате и других функциях позже в этом курсе. Использование ГОДА в этом примере приведено только для иллюстрации вычисляемых столбцов.

Примечание: Не все вычисления будут пересчитаны для каждой строки. SQL Server может вычислить результат функции только один раз во время выполнения запроса и повторно использовать значение для каждой строки.

Demo 1

**Lesson 2 3-6**

**Eliminating Duplicates with DISTINCT**

**SQL Sets and Duplicate Rows**

В то время как теория реляционных баз данных требует уникальных строк в таблице, на практике результаты запросов T-SQL не являются истинными наборами. Строки, полученные запросом, не гарантированно будут уникальными, даже если они получены из исходной таблицы, которая использует первичный ключ для различения каждой строки. Строки не возвращаются в каком-либо определенном порядке.

Добавьте к этому тот факт, что поведение оператора SELECT по умолчанию включает ключевое слово ALL, и вы можете начать понимать, почему повторяющиеся значения могут быть возвращены запросом, особенно если вы включаете только некоторые столбцы в таблицу (и опускаете уникальные столбцы).

For example, consider a query that returns country names from the Sales.Customers table:

SELECT country

FROM Sales.Customers;

Частичный результат показывает множество повторяющихся названий стран, которые в лучшем случае слишком длинны, чтобы их можно было легко интерпретировать. В худшем случае это дает неправильный ответ на вопрос: “Сколько стран представлено среди наших клиентов?”country

--------------------

Germany

Mexico

Mexico

UK

Sweden

Germany

Germany

France

UK

…

**All Statement**

SELECT ALL country

FROM Sales.Customers;

Без дальнейших инструкций запрос вернет один результат для каждой строки в таблице продаж. Таблица клиентов однако, поскольку указан только столбец страна, будет показан только этот столбец для всех строк.

**Understanding DISTINCT**

Замена предложения SELECT ALL по умолчанию на SELECT DISTINCT отфильтрует дубликаты в результирующем наборе. SELECT DISTINCT указывает, что результирующий набор должен содержать только уникальные строки.

Однако важно понимать, что опция DISTINCT работает только с набором столбцов, возвращаемых предложением SELECT. Он не учитывает никаких других уникальных столбцов в исходной таблице. DISTINCT также работает со всеми столбцами в списке ВЫБОРА, а не только с первым.

Логический порядок операций также гарантирует, что оператор DISTINCT удалит строки, которые, возможно, уже были обработаны предложениями WHERE, HAVING и GROUP BY.

Продолжая предыдущий пример стран с продаж.Таблица Customers, вы можете заменить тихое значение по умолчанию на DISTINCT, чтобы исключить дубликаты значений:

*SELECT DISTINCT country*

*FROM Sales.Customers;*

This will return the desired results. Note that, while the results appear to be sorted, this is not guaranteed by SQL Server. The result set now contains only one instance of each unique output row:

country

--------

Argentina

Austria

Belgium

Brazil

Canada

Denmar

…

**SELECT DISTINCT Syntax**

Помните, что DISTINCT просматривает строки в выходном наборе, созданном предложением SELECT. Поэтому предложение SELECT DISTINCT будет возвращать только уникальные значения столбцов. Например, если вы запросите таблицу со следующими данными в ней, вы можете заметить, что существует только четыре уникальных имени и четыре уникальные фамилии:

SELECT firstname, lastname

FROM Sales.Customers;

The results:

firstname lastname

------------ -----------------

Sara Davis

Don Funk

Sara Lew

Don Davis

Judy Lew

Judy Funk

Yael Peled

Однако запрос SELECT DISTINCT для обоих столбцов будет извлекать все уникальные комбинации двух столбцов, которые в данном случае являются одними и теми же семью сотрудниками.

For a list of unique first names only, execute a SELECT DISTINCT only against the firstname column:

DISTINCT Syntax

SELECT DISTINCT firstname

FROM Sales.Customers;

The results:

firstname

---------------------

Don

Judy

Sara

Yael

(4 row(s) affected)

Проблема при разработке таких запросов заключается в том, что, хотя вам может потребоваться получить отдельный список значений из одного столбца, вы можете захотеть увидеть дополнительные атрибуты (столбцы) из других. Позже будет описание, как объединить DISTINCT с предложением GROUP BY для дальнейшей обработки и отображения информации о различных списках значений.

Demo 2

**Lesson 3**

**Using Column and Table Aliases**

При извлечении данных из таблицы или представления запрос T-SQL будет называть каждый столбец после его источника. Вы можете переназначить столбцы, используя псевдонимы в предложении SELECT. Однако столбцы, созданные с помощью выражений, не будут называться автоматически. Псевдонимы столбцов можно использовать для предоставления пользовательских заголовков столбцов. На уровне таблицы вы можете использовать псевдонимы в предложении FROM, чтобы обеспечить удобный способ ссылки на таблицу в другом месте запроса, повышая читабельность.

**Use Aliases to Refer to Columns**

Псевдонимы столбцов можно использовать для переназначения столбцов при возврате результатов запроса. Например, загадочные имена столбцов в таблице, такие как "кол-во", могут быть заменены на "количество".

Выражения, не основанные на исходном столбце таблицы, не будут иметь имени, указанного в результирующем наборе. Это включает в себя вычисляемые выражения и вызовы функций. Хотя T-SQL не требует, чтобы столбец в результирующем наборе имел имя, рекомендуется его предоставить.

В T-SQL существует несколько методов создания псевдонима столбца с одинаковыми результатами вывода.

Один из методов заключается в использовании ключевого слова AS для отделения столбца или выражения от псевдонима:

SELECT orderid, unitprice, qty AS quantity

FROM Sales.OrderDetails;

Another method is to assign the alias before the column or expression, using the equals sign as the separator:

SELECT orderid, unitprice, quantity = qty

FROM Sales.OrderDetails;

**Use Aliases to Refer to Tables**

Псевдонимы также могут использоваться в предложении FROM для ссылки на таблицу; это может улучшить читаемость и сэкономить избыточность при ссылке на таблицу в другом месте запроса. Хотя этот модуль был сосредоточен на запросах с одной таблицей, которые не обязательно извлекают выгоду из псевдонимов таблиц, этот метод окажется полезным, когда вы изучите более сложные запросы в последующих модулях.

Чтобы создать псевдоним таблицы в предложении FROM, вы будете использовать синтаксис, аналогичный нескольким методам псевдонимов столбцов.

Вы можете использовать ключевое слово AS, чтобы отделить имя таблицы от псевдонима. Этот стиль предпочтительнее:

SELECT orderid, unitprice, qty

FROM Sales.OrderDetails AS OD;

Вы можете опустить ключевое слово AS и просто следовать за именем таблицы с псевдонимом:

SELECT orderid, unitprice, qty

FROM Sales.OrderDetails OD;

To combine table and column aliases in the same SELECT statement, use the following approach:

SELECT OD.orderid, OD.unitprice, OD.qty AS Quantity

FROM Sales.OrderDetails AS OD;

Нет псевдонима таблицы, эквивалентного использованию знака равенства (=) в псевдониме столбца. Поскольку этот модуль фокусируется на запросах с одной таблицей, вы, возможно, еще не видите преимущества использования псевдонимов таблиц. В следующем модуле вы узнаете, как извлекать данные из нескольких таблиц в одной инструкции SELECT. В этих запросах будет полезно использовать псевдонимы таблиц для представления имен таблиц.

**The Impact of Logical Processing Order on Aliases**

При использовании псевдонимов столбцов может возникнуть проблема. Псевдонимы, созданные в предложении SELECT, не могут упоминаться в других предложениях запроса, таких как предложение WHERE или HAVING. Это связано с обработкой запросов в логическом порядке. Предложения WHERE и HAVING обрабатываются до оценки предложения SELECT и его псевдонимов (предложения HAVING и WHERE будут рассмотрены в отдельном модуле). Исключением из этого правила является предложение ORDER BY.

Пример приведен здесь для иллюстрации и будет выполняться без ошибок:

SELECT orderid, unitprice, qty AS quantity

FROM Sales.OrderDetails

ORDER BY quantity;

Однако в следующем примере будет возвращена ошибка, так как предложение WHERE было обработано до того, как предложение SELECT определит псевдоним:

SELECT orderid, unitprice, qty AS quantity

FROM Sales.OrderDetails

WHERE quantity > 10;

В результате вам часто придется повторять выражение более одного раза в предложении SELECT, где вы можете создать псевдоним для имени столбца, а также в предложении WHERE или HAVING:

SELECT orderid, YEAR(orderdate) AS orderyear

FROM Sales.Orders

WHERE YEAR(orderdate) = '2008'

Кроме того, в предложении SELECT вы не можете ссылаться на псевдоним столбца, определенный в одной и той же инструкции SELECT, независимо от порядка столбцов.

The following statement will return an error:

SELECT productid, unitprice AS price, price \* qty AS total

FROM Sales.OrderDetails;

Demo 3

**Lesson 4**

**Writing Simple CASE Expressions**

Выражение CASE расширяет возможность предложения SELECT манипулировать данными по мере их извлечения. Часто при написании запроса необходимо заменить значение столбца другим значением. Хотя вы узнаете, как выполнить этот вид поиска из другой таблицы позже в этом курсе, вы также можете выполнять базовые замены, используя простые выражения РЕГИСТРА в предложении SELECT. В реальных средах РЕГИСТР часто используется для того, чтобы сделать зашифрованные данные, хранящиеся в столбце, более значимыми.

Выражение CASE возвращает скалярное (однозначное) значение, основанное на условной логике, часто с несколькими условиями. В качестве скалярного значения он может использоваться везде, где можно использовать отдельные значения. Помимо оператора SELECT, выражения CASE могут использоваться в предложениях WHERE, HAVING и ORDER BY.

**Using CASE Expressions in SELECT Clauses**

В T-SQL выражения CASE возвращают одно или скалярное значение. В отличие от некоторых других языков программирования, в T-SQL выражения CASE не являются операторами и не определяют управление программным потоком.

Вместо этого они используются в предложениях SELECT (и других) для возврата результата выражения. Результаты отображаются в виде вычисляемого столбца и для ясности должны быть сглажены.

В запросах T-SQL выражения CASE часто используются для предоставления альтернативного значения для значения, хранящегося в исходной таблице. Например, выражение CASE может использоваться для предоставления удобного текстового имени для чего-либо, хранящегося в виде компактного числового кода.

**Forms of CASE Expressions**

В T-SQL выражения CASE могут принимать одну из двух форм—простой РЕГИСТР или искомый (логический) РЕГИСТР.

Простые выражения СЛУЧАЯ, предмет этого урока, сравнивают входное значение со списком возможных совпадающих значений:

Если совпадение найдено, первое совпадающее значение возвращается в результате выражения CASE. Несколько совпадений не допускаются.

Если совпадение не найдено, выражение CASE возвращает значение, найденное в предложении ELSE, если оно существует.

Если совпадение не найдено и предложение ELSE отсутствует, выражение CASE возвращает значение NULL.

Например, следующее выражение CASE заменяет описательное имя категории значением categoryid, хранящимся в производстве.Таблица категорий. Обратите внимание, что это не операция СОЕДИНЕНИЯ; вместо этого это подстановка с использованием одной таблицы:

SELECT productid, productname, categoryid,

CASE categoryid

WHEN 1 THEN 'Beverages'

WHEN 2 THEN 'Condiments'

WHEN 3 THEN 'Confections'

ELSE 'Unknown Category'

END AS categoryname

FROM Production.Categories

Предыдущий пример приведен только для иллюстрации и не будет работать с образцами баз данных, предоставленными вместе с курсом.

Выражения искомого (логического) СЛУЧАЯ сравнивают входное значение с набором логических предикатов или выражений. Выражение может содержать диапазон значений для сопоставления. Как и в простом выражении CASE, возвращаемое значение находится в предложении THEN соответствующего значения.

Demo 4.

Lab.