Программирование в SQL Server

Пакет (batch) — это одна или несколько команд SQL, передаваемых на SQL Server для выполнения. После подключения начинается передача пакетов с различными командами SQL Server.

Команда **Go** используется для определения момента передачи пакета.

Например, следующий пакет состоит из двух команд SELECT:

```
SELECT * FROM Студенты
SELECT * FROM Оценки
GO
SELECT * FROM Предметы
```

В этом примере сначала две команды передаются серверу и выполняются им, а их результаты вместе возвращаются клиенту. Команды этого пакета анализируются, компилируются и выполняются как единая группа. Если сервер обнаруживает синтаксические ошибки, не выполняется весь пакет. А последняя команда, которая возвращает список предметов, выполняется в отдельном пакете.

Комментарии

Существует несколько основных вариантов записи комментариев.

Комментарии бывают однострочными и многострочными. Однострочный комментарий начинается с «--». А многострочный начинается с «/*», а заканчивается «*/».

Например:

```
/* Многострочный комментарий*/
Однострочный комментарий
SELECT * FROM Предметы  -- комментарий
```

Переменные

Идентификаторы – это имена объектов, на которые можно ссылаться в программе, написанной на языке SQL. Первый символ может состоять из букв английского алфавита или «_», «@», «#». Остальные дополнительно из цифр и «$».

Имена идентификаторов не должны совпадать с зарезервированным словом.

Для ограничителей идентификаторов при установленном параметре

SET QUOTED_IDENTIFIER ON

можно использовать как квадратные скобки, так и одинарные кавычки, а строковые значения только в одинарных кавычках (режим по умолчанию).

Если использовать установленный параметр в режиме

SET QUOTED_IDENTIFIER OFF,

то в качестве ограничителей идентификаторов можно использовать только квадратные скобки, а строковые значения указываются в одинарных или двойных кавычках.

Переменные используются для сохранения промежуточных данных в хранимых процедурах и функциях. **Все переменные считаются локальными**. **Имя переменной должно начинаться с @**.

Объявление переменных

Синтаксис в обозначениях MS SQL Server:

DECLARE @имя_переменной1 тип_переменной1, …, @имя_переменнойN тип_переменнойN

Если тип переменной предполагает указанием размера, то используется следующий синтаксис для объявления переменных:

DECLARE @имя_переменной1 тип_переменной1 (размер), …, @имя_переменнойN тип_переменнойN(размер)

Пример:

```
DECLARE @a bigint, @b decimal(2,2)
DECLARE @s varchar(50)
```

Присвоение значений переменным и вывод значений на экран

Присвоение значений локальной переменной осуществляется:

- ✓ используя специальную форму инструкции SELECT;
- ✓ используя инструкцию SET;
- ✓ непосредственно в инструкции DECLARE посредством знака = (например, DECLARE @a bigint=1200).

Пример:

```
DECLARE @a bigint, @b decimal(2,2)
SET @a=5
SET @b=(@a+@a)/3
SELECT @b                    --вывод результата на экран
```

Присвоение с помощью SELECT – помещение результата запроса в переменную. Если в результате выполнения запроса не будет возвращено ни одной строки, то значение переменной не меняется, т.е. остается старым.

Пример:

```
DECLARE @a bigint
SELECT @a=COUNT(*) FROM Студенты
Пример:
DECLARE @s varchar(50)
SELECT @s=Описание FROM Предметы
```

В данном примере в переменную поместиться последнее значение из результата запроса.

Возможно сочетание ключевых слов **SET и SELECT**

```
DECLARE @a bigint
SET @a=(SELECT COUNT(*) FROM Студенты)
```

Работа с датой и временем

Оператор **SET DATEFORMAT dmy|ymd|mdy** задает порядок следования компонентов даты.

Пример:

```
SET DATEFORMAT dmy
DECLARE @d datetime
SET @d='31.01.2012 12:00:15'
SET @d=@d+1
SELECT @d
```

Преобразование типов переменных

Функция **CAST** возвращает значение, преобразованное к указанному типу:

CAST (@переменная или значение AS требуемый_тип_данных)

Пример:

```
DECLARE @d datetime, @s varchar(max)
SET @d='01.01.2000 12:00:15'
SET @s=CAST(@d AS varchar(max))
SELECT @s
```

Функция **CONVERT** возвращает значение, преобразованное к указанному типу по заданному формату

Команда PRINT

Команда **PRINT** передает сообщения длиной до 1024 байтов в качестве служебной информации. В SQL Server Management Studio служебные сообщения выводятся на вкладке **Messages (Сообщения)**

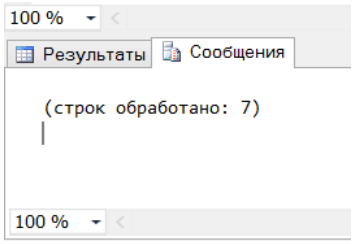


Рисунок 1 Вкладка для вывода служебных сообщений в SQL Server Management Studio

Например, следующая команда **PRINT** просто передает сообщение:

```
PRINT 'Мое первое сообщение'
```

Таким образом можно выводить любую информацию.

Команды условного выполнения

Термин «условное выполнение» означает, что команды выполняются лишь в случае истинности некоторого критерия или условия.

1. Условная конструкция IF…ELSE

Конструкция **IF...ELSE** определяет команды, выполнение которых зависит от некоторого критерия.

Синтаксис:

```
IF логическое условие
        набор операторов 1
ELSE
        набор операторов 2
```

В следующем примере команда **IF...ELSE** проверяет количество книг и возвращает соответствующий текст:

```
IF (SELECT COUNT(*) FROM Предметы)>10
            PRINT 'Количество предметов больше 10'
ELSE
        PRINT 'Количество предметов меньше 10'
```

Операторские скобки

Конструкция **BEGIN...END** используется для создания блока команд. Все, что находится между **BEGIN и END**, является частью блока.

```
BEGIN
```

/* в них нельзя помещать команды, изменяющие структуру объектов базы данных. Операторские скобки должны содержать хотя бы оператор. Требуются для конструкций поливариантных ветвлений, условных и циклических конструкций

***/**

END

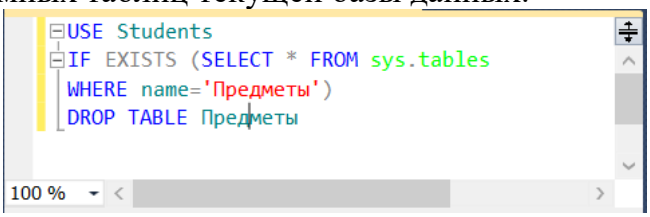
Например, проверка количество изучаемых предметов могла бы выглядеть так:

```
DECLARE @a int
DECLARE @s char(50)
SET @a=(SELECT COUNT(*) FROM Предметы)
IF @a>10
        BEGIN
              SET @s=’Количество предметов больше 10’
              SELECT @s
        END
ELSE
        BEGIN
              SET @s=’Количество предметов=’+str(@a)
              SELECT @s
        END
```

2. Команда IF EXISTS

Команда **IF EXISTS** является частным случаем команды **IF…ELSE** и позволяет узнать, существуют ли какие-либо экземпляры, определяемые условием. Команда **IF EXISTS** заменяет конструкцию **COUNT(*)>0** для проверки существования записей. При обнаружении первой совпадающей записи обработка команды **EXISTS** прекращается.

В следующем примере происходит удаление таблицы Предметы, если таковая есть в списке системных таблиц текущей базы данных.



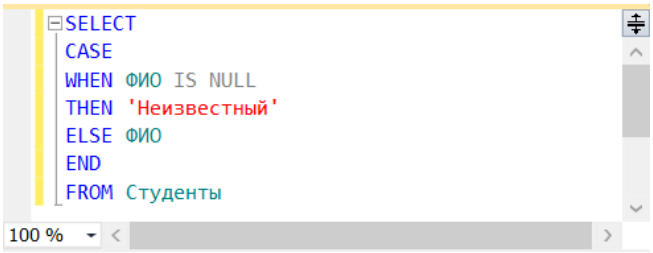
3. Конструкция CASE

Для тех ситуаций, когда проверка нескольких условий требует многих команд **IF**, в SQL Server предусмотрена конструкция **CASE**. Конструкция **CASE**, в отличие от **IF**, может использоваться в команде **SELECT**.

Конструкция **CASE** имеет следующий синтаксис:

```
CASE выражение
        WHEN выражение1 THEN выражение2
          [...]
        [ELSE выражение N]
END
```

В следующем примере рассмотрено применение конструкции CASE. Допустим в таблице Студенты все поля кроме Код студента являются необязательными, тогда для того, чтобы не возвращать пользователю NULL можно применить следующий запрос:



4. Конструкция WHILE

Конструкция **WHILE** используется для многократного выполнения команд. Команда **WHILE** вычисляет условие цикла и, если оно равно **TRUE**, выполняет команду или блок команд.

Синтаксис:

WHILE логическое условие

Оператор 1 или Блок операторов 1

BREAK

Оператор 2 или Блок операторов 2

CONTINUE

Конструкции **BREAK** и **CONTINUE** являются необязательными. Цикл можно принудительно остановить, если в его теле выполнить команду **BREAK**. Если же нужно начать цикл заново, не дожидаясь выполнения всех команд в теле, необходимо выполнить команду **CONTINUE**.

Пример:

```
DECLARE @a int
SET @a=1
WHILE @a<100
        BEGIN
             PRINT @a -- вывод на экран значения переменной
             IF (@a>40) AND (@a<50)
                   BREAK    -- выход и выполнение первой команды за циклом
             ELSE
                   SET @a=@a+rand()*10
            CONTINUE
        END
PRINT @a
```

Команда GOTO

Использование команды **GOTO** позволяет по-другому организовать многократно повторяющиеся вычисления.

Команда WAITFOR

Команда **WAITFOR** переводит запрос в состояние ожидания на некоторое время или до наступления заданного времени.

Команда WAITFOR имеет следующий синтаксис:

WAITFOR [DELAY 'время' | TIME 'время']

Параметр **DELAY** заставляет сделать паузу заданной длины (максимальное значение — 24 часа). Если в команде указан параметр **TIME**, процесс приостанавливается до наступления заданного времени. В обоих случаях время задается в формате hh:mi:ss (дату задать нельзя).

Пример:

1. Пауза до 22:00

```
WAITFOR TIME '22:00:00'
```

2. Выводить текущий список пользователей каждые 30 секунд

```
WHILE 1 < 2
BEGIN
    WAITFOR DELAY '00:00:30'
    EXEC sp_who
END
```

Команда RETURN

Командой **RETURN** осуществляется безусловный выход из обрабатываемого пакета. При желании, при выходе из хранимой процедуры в команде можно задать код возврата.

Команда **RETURN** имеет следующий синтаксис:

RETURN [целое_число]

Пример использования команды **RETURN**:

```
IF EXISTS(SELECT * FROM Предметы WHERE Название = 'Математика')
    BEGIN
       PRINT 'Все в порядке'
       RETURN
    END
PRINT 'Учащиеся изучают не все предметы общеобразовательной программа '
PRINT 'Расширьте перечень изучаемых предметов!'
```

Команда SET

Команда **SET** обычно используется для установки значения переменной. В этом случае она имеет следующий синтаксис:

SET имя_переменной = значение

Пример:

```
SET @MyVar = 12345
```

Также командой **SET** задаются некоторые параметры, определяющие реакцию сервера на некоторые условия. Команда SET имеет следующий синтаксис:

SET условие [ON | OFF | значение]

Примеры некоторых команд **SET**:

SET rowcount 100 – указывает, что запрос вернет только 100 строк

SET nocount on – запрещает вывод количества строк

Параметры настраиваются на уровне сеанса или на уровне хранимой процедуры. Они не сохраняются между сеансами.

Обработка ошибок

Команда RAISERROR

При выполнении SQL-кода может возникнуть ошибка, и потребуется обработать ее. Для этого существует команда **RAISERROR**.

Команда **RAISERROR** передает состояния, коды и сообщения ошибок на программном уровне. **RAISERROR** позволяет использовать стандартное сообщение или определить новое сообщение об ошибке.

Если **RAISERROR** вызывается в блоке **TRY...CATCH**, то управление передается блоку **CATCH**. В противном случае выполнение цепочки команд продолжится (хотя можно использовать команду **RETURN**, чтобы прервать выполнение цепочки команд).

Синтаксис **RAISERROR**, показанный в следующем фрагменте, позволяет ис-

пользовать стандартное сообщение или определить новое сообщение непосредственно в команде:

RAISERROR ({код_ошибки | символьная_строка},
код_серьезности, состояние {список_аргументов})
[WITH параметр]

Коды ошибок должны быть больше 50 и меньше 2 147 483 647. Незапланированные сообщения автоматически инициируют ошибку с кодом 50 000.

Следующий пример вызывает незапланированное сообщение об ошибке:

```
IF NOT EXISTS(SELECT * FROM Предметы)
RAISEERROR ('Таблица с названиями предметов не заполнена. Заполните ее.', 16, 1)
```

Можно использовать функцию **@@ERROR**, которая возвращает код последней ошибки, происшедшей в текущем соединении.

Пример использования функции **@@ERROR**:

```
IF NOT EXISTS(SELECT * FROM Предметы)
RAISEERROR ('Таблица с названиями предметов не заполнена. Заполните ее.', 16, 1)
IF @@ERROR <> 0
   PRINT 'Возникла ошибка при проверке таблицы Предметы. '
ELSE
      PRINT 'Проверка прошла успешно'
GO
```

Конструкция TRY…CATCH

Начиная с SQL Server 2005, существует гораздо более удобный способ обработки ошибок — с помощью конструкции **TRY...CATCH**. Эта конструкция имеет следующий синтаксис:

```
BEGIN TRY
    [sql_выражение | блок_выражений]
END TRY
BEGIN CATCH
    [sql_вьражение | блок_выражений]
END CATCH
```

Каждый блок **TRY...CATCH** должен быть в одном пакете.

Пример обработки ошибки:

```
BEGIN TRY
—      Обновление данных счета клиента
    UPDATE Account SET Balanse = @Sum WHERE ClientID = @ClientID
END TRY
BEGIN CATCH
  SELECT
        ERROR_NUMBER() as ErrorNumber,
        ERROR_MESSAGE() as ErrorMessage;
END CATCH
```

В примере, если при обновлении таблицы Account произойдет ошибка, будет выведено описание этой ошибки.

В блоке **TRY...CATCH** можно использовать некоторые функции, чтобы получить более подробную информацию об ошибке. Их описание представлено в таблице ниже

Имя функции	Описание
ERROR NUMBER()	Возвращает номер ошибки
ERROR MESSAGE()	Возвращает сообщение об ошибке
ERROR LINE()	Возвращает номер строки, где возникла ошибка
ERROR PROCEDURE()	Возвращает имя хранимой процедуры или триггера, где возникла ошибка
ERROR SEVERITY()	Возвращает описание уровня серьезности ошибки
ERROR STATE()	Возвращает номер состояния ошибки

Пример на применение операторов: вычислить значение функции в зависимости от параметров a и b

$$\begin{cases} a + e^{bt}\sin(bt), & \text{если } ab > 0; \\ b + \sin(at), & \text{если } ab < 0; \\ \frac{ab}{4}t^3 + 1, & \text{если } ab = 0; \end{cases}$$

При смене параметров a и b значение функции изменяется.

```
 --Объявление переменных
    declare @a bigint
    declare @b bigint
    declare @t decimal(20,4)
    declare @f decimal(20,4)
 --Присвоение начальных значений
    set @a=1
    set @b=6
    set @t=3.3
 --Посчет значения функции
    if @a*@b>0
       begin
          set @f=@a+EXP(@b*@t)*SIN(@b*@t)
          select @f as 'Результат: выполнено условие a*b>0'-- вывод результата
       end
    else
         if @a*@b<0
            begin
              set @f=@b+SIN(@a*@t)
              select @f 'Результат: выполнено условие a*b<0'-- вывод результата
            end
         else
            begin
               set @f=(@a*@b)/4*POWER(@t,3)+1
               select @f 'Результат: выполнено условие a*b=0'-- вывод результата
            end
```

Результаты | Сообщения

Результат: выполнено условие a*b>0
323207211.0869

Контрольные вопросы

1. Дайте определение понятию *пакет*.
2. Назовите команду, которая используется для определения момента передачи пакета.

3. Назовите варианты записи комментариев: многострочный и однострочный.
4. Дайте определение понятию *идентификатор*.
5. Назовите символ, с которого должны начинаться переменные.
6. Приведите примеры объявления переменных и присвоения им значений.
7. Опишите и поясните синтаксис конструкции IF.
8. Опишите и поясните синтаксис конструкции WHILE.
9. Являются ли конструкции BREAK и CONTINUE обязательными?
10. Как можно остановить цикл принудительно?

Преподаватель С.В. Банцевич