SQL/PSM и триггеры

Виноградова М.В.

Базы данных

МГТУ им. Н.Э. Баумана (ИУ5)

SQL – Structured Query Lanquage

- 1986 – ANSI – SQL/86 (ISO в 1987)
- 1989 - SQL/86 – запросы и схемы
- **1992 - SQL/92 – схемы, транзакции, соединения, авторизация, триггеры**
- 1995 - SQL/CLI – динамический, ODBC
- **1996 - SQL/PSM – хранимые процедуры**
- 1999 – SQL:1999 (SQL3) – объектное расширение, UDT
- 2003 – SQL:2003 – OLAP, XML
- 2006 – XQuery

Возможности программирования на стороне сервера

- Поддержка сложных типов данных
- Встроенные (системные) функции
- Хранимые процедуры и функции
- Триггеры DDL/DML
- Общие табличные выражения
- Рекурсия
- Ранжирование
- Транспонирование

Хранимые процедуры и функции

- PERSISTENT STORED MODULES
- SQL/PSM, PSM-96

- Объекты схемы БД
- CREATE – ALTER – DROP
- Хранятся (в откомпилированном виде) и исполняются на сервере

PSM – объявления

CREATE PROCEDURE имя(параметры)

Объявление локальных переменных

Тело;

CREATE FUNCTION имя(параметры) **RETURNS** тип

Объявление локальных переменных

Тело;

Параметры: режим – имя – тип

- IN (по ум.) | OUT | INOUT
- OUT | INOUT – запрещено для функции

Возможности операций PSM

- Объявление и использование переменных
- Возврат значений
- Условия
- Циклы
- Работа с курсором
- Перехват, обработка и вызов исключений
- Вызов системных функций
- DML и DDL команды языка SQL
- Динамические запросы

Операторы PSM

- Возврат значения

  RETURN выражение;

- Локальная переменная

  DECLARE имя тип;

- Присваивание

  SET переменная = выражение;

- (= NULL или (SELECT…), если возвращает 1 значение)

- Блоки кода

  BEGIN

  …

  END

- Метки

  имя:

Условия

IF условие **THEN**
    Список выражений
ELSEIF условие **THEN**
    Список выражений
ELSE
    Список выражений
END IF;

CASE перемнная
    WHEN значение1 **THEN** ....
    WHEN значение2 **THEN** ....
END;

Циклы

metka: **LOOP**

код;

условие **LEAVE** metka;

код;

END LOOP;

FOR имя цикла **AS** имя курсора **CURSOR FOR**

Запрос

DO

Список выражений

END FOR;

Циклы - 2

WHILE условие **DO**
  Список выражений
END WHILE;

REPEAT
  Список выражений
  UNTIL условие
END REPEAT;

Пример процедуры

```
CREATE PROCEDURE Move
(
    IN oldAddress VARCHAR(255),
     IN newAddress VARCHAR(255) default ‘Moscow’
)

BEGIN
       UPDATE MovieStar
                SET address = newAddress
                WHERE address = oldAddress;
END
```

Вызов процедур и функций

- Из программы на базовом языке
- Из хранимой процедуры, функции
- Как команда базового интерфейса SQL
- Функции выделяются как часть крупного выражения (из запроса)

CALL имя(аргументы);

EXECUTE имя(аргументы);

Select * from tab where col in fun(col2)

Типы пользовательских функций

- **Скалярные** – возвращают одно значение
- **Табличные** – возвращают набор записей
- **Встроенные** (inline) – как представление с параметром
- **Агрегатные** – применяют к набору значений. Функции инициализации, итерации, и итогового вычисления.

Объявление переменных в MS SQL

- Перед использованием переменной:

  declare @имя тип

- В качестве типа переменной можно использовать любой тип, в том числе и табличный

  declare @str nchar(10);

- Если табличный тип, то необходимо определить структуру таблицы

  declare @person table (name nvarchar(100), age int);

- Для обращения к табличной переменной используются команды SQL для добавления, изменения, удаления и просмотра записей

  Insert into @person values('Иванов И.И.', 20);

Использование переменных в MS SQL

- Присвоение значений оператором «set»:

set @переменная=значение;

- Значением может быть константа, NULL или запрос на извлечение гарантированно одного значения, заключенный в скобки, например

set @dest = 'Москва';

set @src = (select name from Cities where id=1);

Условия в MS SQL

- Условие задается с помощью конструкции **«if .. else ..»**

  if условие

  блок кода

  else

  блок кода

- Условием может быть любое условное выражение, аналогичное используемому в части «WHERE»:

if **not exists(select * from roads where src=@src and dest=@dest)**

Циклы в MS SQL

- Для создания цикла используется оператор «**while**»:

  while условие

  блок кода

- Команда «**break**» используется для прерывания цикла и перехода к оператору, следующему за циклом.
- Команда «**continue**» используется для перехода к началу цикла

Исключения в MS SQL

- Для перехвата и обработки исключения используется конструкция **«try .. catch»:**

  begin try
  блок исходного кода
  end try
  begin catch
  код при возникновении исключения
  end catch;

- Для получения информации о возникшем исключении используются встроенные функции, например, **«error_number()»** или **«error_message()»,** которые возвращают код или текст возникшей ошибки.

Вызов исключения в MS SQL

- Для прерывания программы и вызова исключения используется встроенная функция «**raiserror**()»:

raiserror(сообщение_или_код, уровень серьезности, состояние);

- Указанные в качестве первого параметра функции сообщение или код возвращаются в вызвавшую программу как код или сообщение об ошибке.
- Уровень серьезности задает ошибку или предупреждение и определяет, произойдет ли откат выполняемой транзакции.
- Состояние указывает место возникновения ошибки.

Пример скалярной функции

```
CREATE FUNCTION dbo.countRoad
      ( @frm nchar(10),
        @tto nchar(10)  )
RETURNS int
AS
BEGIN
      -- Declare the return variable here
      DECLARE @ResultVar int;
      -- Add the T-SQL statements to compute the return value here
      set @ResultVar = (select count(*) from roads
                              where src=@frm and dest=@tto);
      -- Return the result of the function
      RETURN @ResultVar;
END
GO
```

Вызов скалярной функции

- может вызываться из запроса как поле данных в частях «SELECT» или «WHERE».

  select src, dest, dbo.countRoad(src,dest) from roads

- вызов вне запроса:

  select dbo.countRoad('Москва','Тверь')

- Ей можно передавать в качестве параметров поля таблиц или константы.

	src	dest	(No column name)
1	москва	СПб	1
2	москва	анапа	1
3	СПб	лондон	1
4	СПБ	париж	1
5	лондон	нью-йорк	1
6	париж	вена	1
7	анапа	сочи	1

Табличные фунции

- Функция типа «inline» содержит только один запрос который подставляется в место обращения к функции.
- Табличная функция типа «Multi-statement» из нескольких команд и запросов, вызывается и выполняется как программа.

CREATE FUNCTION схема.название(параметры)
RETURNS @переменная TABLE (описание таблицы)
AS BEGIN
Код функции
RETURN
END

- Переменная, указанная как возвращаемый тип, содержит набор записей, которые возвращаются в качестве результата функции. Код функции должен обеспечить заполнение этой переменной, например, командой «insert».
- Табличная функция указывается в части «FROM» запроса.

Пример табличной функции

```
CREATE FUNCTION dbo.roadByType(@type nchar(10))
RETURNS TABLE
AS
RETURN
(
      -- Add the SELECT statement with parameter references here
      SELECT src,dest from roads where type=@type
)
GO
```

Вызов:

select distinct * from dbo.roadByType('жд')

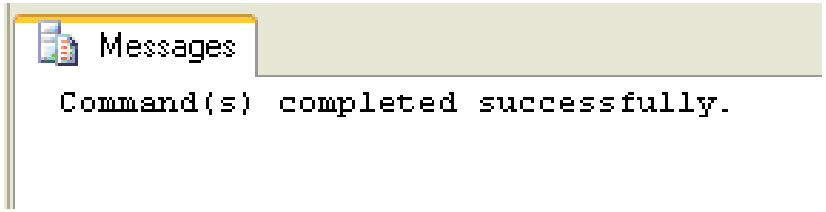
	src	dest
1	москва	анапа
2	москва	СПб
3	СПБ	париж

Пример процедуры

```
CREATE PROCEDURE dbo.addRoad
      @src nchar(10), @dest nchar(10), @type nchar(10) = 'ææ'
AS
BEGIN
      SET NOCOUNT ON;
-- Если NULL на входе, то выдает предупреждение
    if( @src is NULL OR @dest is NULL)
            raiserror('Îøèáêà ââîäà äàííûõ',10,1);
-- Если дорога не существует, то добавить ее с перехватом возможного исключения
      else if not exists(select * from roads
                        where src=@src and dest=@dest and [type]=@type )
            begin try
                  insert into roads(src,dest,[len],[type],cost)
                        values (@src,@dest,1,@type,10)
            end try
            begin catch
-- Вывод своего сообщения при исключении
                  raiserror('Îøèáêà äîáàâëåíèÿ',16,1);
            end catch;
END
GO
```

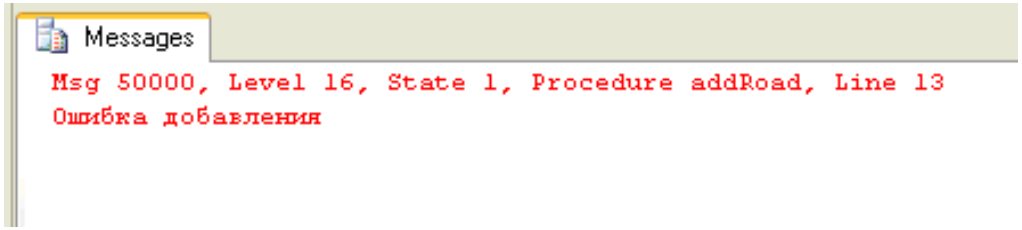
Вызов процедуры

exec dbo.addRoad 'сочи','геленджик'



Проверить перехват исключения
(значение NULL в обязательное поле):

exec dbo.addRoad 'сочи','туапсе',NULL



Курсоры

DECLARE имя **CURSOR FOR** select **...**

OPEN имя

в цикле:

FETCH имя **INTO** переменные;

CLOSE имя

Работа с курсором в MS SQL

- Объявить курсор как запрос

  DECLARE c cursor for select ……
- Открыть курсор

  open c;
- В цикле извлечь записи из курсора

  fetch from c into @str;
- - извлекает данные из текущей записи курсора в перемнную.
- @@fetch_status – состояние курсоа (= 0, если есть записи)

  while @@fetch_status = 0

  код
- Закрыть курсор

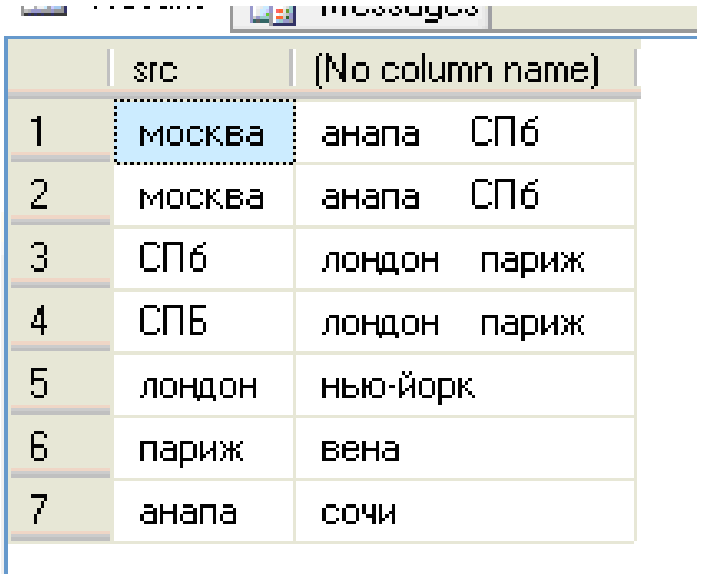
  close c;

Пример курсора

```
CREATE FUNCTION dbo.destRoad(@city nchar(10))
RETURNS nvarchar(100)
AS
BEGIN
	-- Declare the return variable here
	declare @dests nvarchar(100), @str nchar(10);
	DECLARE c cursor for
		select distinct dest from roads where src=@city;
	set @dests='';
	open c;
	fetch from c into @str;
	while @@fetch_status = 0
	begin
		set @dests = @dests + @str;
		fetch from c into @str;
	end;
	close c;
	-- Return the result of the function
	RETURN @dests;
END
GO
```

Обращение к курсору

- Вызов функции:
  select src, dbo.destRoad(src) from roads



	src	(No column name)
1	москва	анапа СПб
2	москва	анапа СПб
3	СПб	лондон париж
4	СПБ	лондон париж
5	лондон	нью-йорк
6	париж	вена
7	анапа	сочи

Общие табличные выражения (пример)

WITH regional_sales AS

(SELECT .....),

top_regions AS

(SELECT ....)

SELECT ...

FROM regional_sales, top_regions

WHERE .....

Пример ранжирования (оконные функции)

```
select ROW_NUMBER() over (order by src),
       rank() over (order by src),
       dense_rank() over (order by src),
       ntile(3) over (order by src),
       src
from roads
```

1	1	1	1	анапа
2	2	2	1	лондон
3	3	3	1	москва
4	3	3	2	москва
5	5	4	2	париж
6	6	5	3	СПб
7	6	5	3	СПБ

Триггеры DML

CREATE TRIGGER Название
AFTER UPDATE OF Таблица **ON** Поле
REFERENCING
    OLD ROW AS oldTuple,
    NEW ROW AS newTuple
FOR EACH ROW
WHEN (условие)
BEGIN
    код
END

Парметры DML триггера

AFTER | BEFORE | INSTEAD OF

UPDATE | INSERT | DELETE

WHEN – не обязательно

FOR EACH ROW -- код работает для кортежа

FOR EACH STATEMENT -- для всего отношения
OLD TABLE AS имя1
NEW TABLE AS имя2

```
CREATE TRIGGER [dbo].[trig1]
   ON  [dbo].[RoadBuilder]
   INSTEAD OF INSERT
AS
BEGIN
    declare @road int;
      if exists( select * from roads, inserted
                        where roads.src = inserted.src and
                        roads.dest = inserted.dest )
            begin
                  set @road = (select id from roads, inserted
                        where roads.src = inserted.src and
                        roads.dest = inserted.dest )
                  insert into builder(title,year_beg,year_end, road)
                        select inserted.title,inserted.year_beg,
                              inserted.year_end, @road
                              from inserted;
            end
      else
            begin
                  insert into roads(src,dest,[type])
                        select inserted.src,inserted.dest,'æä'
                              from inserted;
                  set @road = ident_current('roasd');
                  -- set @road = @@identity;
                  insert into builder(title,year_beg,year_end, road)
                        select inserted.title,inserted.year_beg,
                                    inserted.year_end, @road
                                    from inserted;
            end;
END
```

DDL триггеры

- Уровня БД / уровня сервера
- Изменения структуры объектов БД
- Системные переменные (структуры) для получения информации о событии
- Возможность отката транзакции

Пример DDL триггера

- перехватывает любые изменения структуры таблиц. И выводит на печать их описание.

```
create trigger trig_ddl
on database
for DDL_TABLE_EVENTS
as
begin
      declare @str nvarchar(max);
      set @str = cast(eventdata() as nvarchar(max));
      print @str;
end
```

```
create table tab1
(
          num int
);
```

Messages

```
<EVENT_INSTANCE><EventType>CREATE_TABLE</EventType><PostTime>20
( &#x0D;
    num int&#x0D;
);</CommandText></TSQLCommand></EVENT_INSTANCE>
```

Пример втроенных типов в PostgreSQL

- Числовые, Денежные, Символьные, Двоичные, Типы даты/времени, Логический
- Типы перечислений
- Геометрические типы (Точки, Прямые, Отрезки, Прямоугольники, Пути, Многоугольники, Окружности)
- Типы, описывающие сетевые адреса
- Битовые строки
- Типы, предназначенные для текстового поиска
- Тип UUID
- Тип XML
- Типы JSON
- Массивы
- Составные типы (структуры)
- Диапазонные типы
- Типы доменов
- Идентификаторы объектов (Oid)