Практическая работа 17
Создание хранимых процедур в базах данных

Цель занятия: Получить практический опыт создания хранимых процедур в базах данных

Перечень оборудования и программного обеспечения

Персональный компьютер
Microsoft Office (Word, Visio, Access)
Microsoft SQL Server Management Studio

Краткие теоретические сведения

1 Хранимые процедуры

В работе с БД часто используются одни и те же запросы, либо набор последовательных запросов. Хранимые процедуры позволяют объединить последовательность запросов и сохранить их на сервере. Они повышают скорость и эффективность использования БД и вызываются командой Exec имя_хранимой_процедуры

Формат хранимой процедуры:

```
CREATEPROCEDURE <Procedure_Name,sysname,ProcedureName>
@Param
AS
BEGIN
операторы
END
GO
```

Параметры - это те данные, которые мы будем передавать процедуре при ее вызове, а операторы - это собственно запросы.

2 Использованиесреды SQL Server Management Studio

1. В **обозревателе объектов** подключитесь к экземпляру компонента Компонент DatabaseEngine и разверните его.
2. Последовательно разверните узел **Базы данных**, выберите вашу базу данных и узел **Программирование**.
3. Щелкните правой кнопкой мыши элемент **Хранимые процедуры** и выберите пункт **Создать хранимую процедуру**.
4. В меню **Запрос** выберите пункт **Задание значений для параметров шаблона**.

5. В диалоговом окне **Задание значений для параметров шаблона** введите для показанных параметров следующие значения.

Параметр	**Значение**
Автор	*Yourname*
Дата создания	*Today'sdate*
Описание	Введите описание своей процедуры
Procedure_name	Имя процедуры
@Param1	Параметр1
@Datatype_For_Param1	**Тип параметра 1**
Default_Value_For_Param1	Значение по умолчанию
@Param2	Параметр2
@Datatype_For_Param2	**Тип параметра 2**
Default_Value_For_Param2	Значение по умолчанию

6. Нажмите кнопку **ОК**.
7. В **редакторе запросов** замените инструкцию SELECT требуемой инструкцией.
8. Для проверки синтаксиса выберите пункт **Синтаксический анализ** в меню **Запрос**. Если возвращается сообщение об ошибке, сравните инструкции с приведенными выше и при необходимости внесите исправления.
9. Чтобы создать процедуру, в меню **Запрос** выберите пункт**Выполнить.** Процедура создается как объект в базе данных.
10. Чтобы увидеть процедуру в обозревателе объектов, щелкните правой кнопкой мыши элемент **Хранимые процедуры** и выберите пункт**Обновить.**
11. Чтобы выполнить процедуру, в обозревателе объектов щелкните правой кнопкой мыши имя хранимой процедуры и выберите пункт **Выполнение хранимой процедуры**.
12. В окне **Выполнение процедуры** введите значения для параметров@Param1, @Param2

Хранимые процедуры это не просто контейнера для групп запросов, как может показаться. Например, хранимые процедуры могут в своей работе использовать операторы ветвления. Вне хранимых процедур такие операторы использовать нельзя.

В среде SQL Server предусмотрен ряд различных управляющих конструкций, без которых невозможно написание эффективных алгоритмов.

Группировка двух и более команд в единый блок осуществляется с использованием ключевых слов BEGIN и END:

BEGIN

{ sql_оператор | блок_операторов }
END

Сгруппированные команды воспринимаются интерпретатором SQL как одна команда. Подобная группировка требуется для конструкций поливариантных ветвлений, условных и циклических конструкций. Блоки BEGIN...END могут быть вложенными.

Некоторые команды SQL не должны выполняться вместе с другими командами (речь идет о командах резервного копирования, изменения структуры таблиц, хранимых процедур и им подобных), поэтому их совместное включение в конструкцию BEGIN...END не допускается.

Нередко определенная часть программы должна выполняться только при реализации некоторого логического условия.

Условные выражения

Для выполнения действий по условию используется выражение IF ... ELSE. SQL Server вычисляет выражение после ключевого слово IF. И если оно истинно, то выполняются инструкции после ключевого слова IF. Если условие ложно, то выполняются инструкции после ключевого слова ELSE.

Если после IF или ELSE располагает блок инструкций, то этот блок заключается между ключевыми словами BEGIN и END:

IF условие
  {инструкция|BEGIN...END}
[ELSE
  {инструкция|BEGIN...END}]
Выражение ELSE является необязательным, и его можно опускать.

Циклы организуются с помощью следующей конструкции:

WHILE лог_выражение
{ sql_оператор | блок_операторов }
[BREAK]
{ sql_оператор | блок_операторов }
[CONTINUE]

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.

Оператор поливариантных ветвлений

Для замены множества одиночных или вложенных условных операторов используется следующая конструкция:

```
CASE входное_значение
WHEN {значение_для_сравнения |
лог_выражение } THEN
вых_выражение [,...n]
[ ELSE иначе_вых_выражение ]
END
```

Если входное значение и значение для сравнения совпадают, то конструкция возвращает выходное значение. Если же значение входного параметра не найдено ни в одной из строк WHEN...THEN, то тогда будет возвращено значение, указанное после ключевого слова ELSE.

Задания

1 Изучить теоретические сведения.

2 В соответствии с вариантом задания создать хранимые процедуры в базе данных:

- хранимые процедуры без параметров;
- хранимые процедуры с входными параметрами;
- хранимые процедуры с входными и выходными параметрами. В процедурах используйте агрегатные функции, сортировку, поиск в интервале.

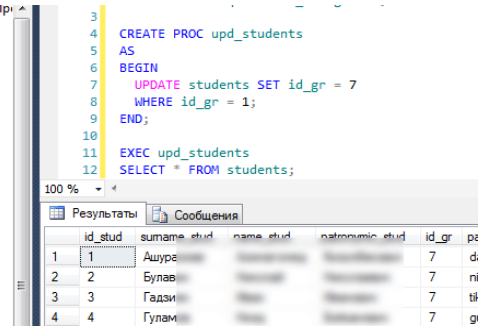
При создании хранимых процедур необходимо учесть:

- целесообразность создания процедуры;
- частоту обращения к процедуре;
- возможность использования хранимых процедур в приложении баз данных.

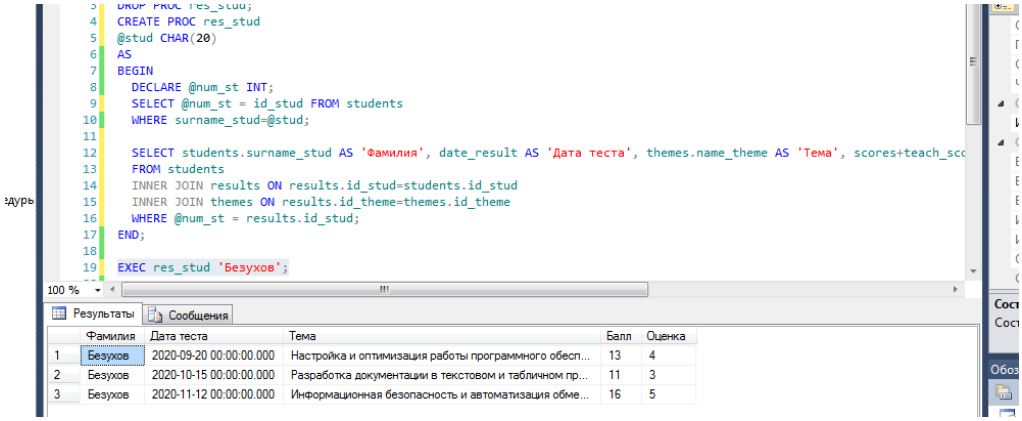
Порядок выполнения работы

1 Процедура без параметров может быть создана для ввода данных в таблицу, для обновления данных нескольких таблиц, например, в информационной системе магазина, когда при оформлении покупки данные добавляются в таблицу продаж и одновременно корректируются в таблице наличия.

В базе данных тестирования были внесены изменения в таблице «Группы», код группы изменился, теперь необходимо внести изменения в таблицу «Студенты»:



2 Процедура с входными параметрами. Пусть в базе данных тестирования необходимо проанализировать результаты прохождения аттестации конкретным студентом. Создадим процедуру, в которой фамилия студента будет параметром. Для достижения наглядности информации в запросе используются таблицы «Студенты», «Темы» и «Результаты», соединенные оператором JOIN. Таким образом, в запросе не присутствуют коды, которые служат для организации связей в базе данных:

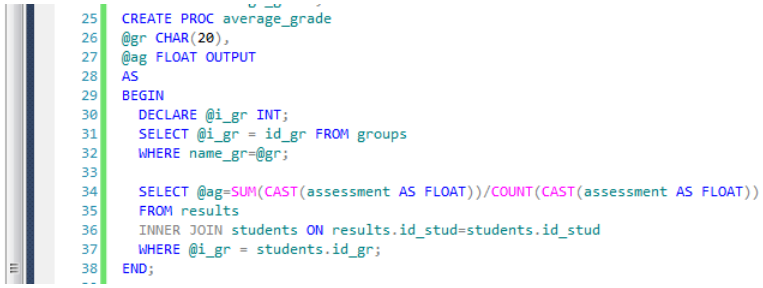


3 Процедура с выходными параметрами. Пусть необходимо просмотреть результаты тестирования всех студентов группы, средний балл которых меньше общего среднего балла группы. Для этого определим процедуру вычисления среднего балла указанной группы. Для этого воспользуемся агрегатными функциями, но значение среднего значения будет неточным, т.к. будет получено целое число, которое не отобразит реальное положение дел в полной мере. Поэтому при использовании агрегатных функций воспользуемся преобразованием целых типов полей в действительные. Для этого можно воспользоваться функцией CAST:

Функция CAST преобразует выражение одного типа к другому. Она имеет следующую форму:

CAST(выражение AS тип_данных)

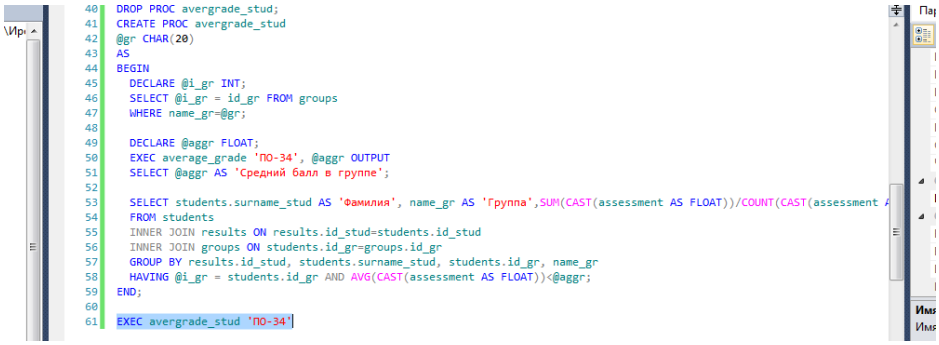
Тогда хранимая процедура вычисления среднего балла имеет один входной параметр – название группы и один выходной – среднее значение.



```
CREATE PROC average_grade
@gr CHAR(20),
@ag FLOAT OUTPUT
AS
BEGIN
  DECLARE @i_gr INT;
  SELECT @i_gr = id_gr FROM groups
  WHERE name_gr=@gr;

  SELECT @ag=SUM(CAST(assessment AS FLOAT))/COUNT(CAST(assessment AS FLOAT))
  FROM results
  INNER JOIN students ON results.id_stud=students.id_stud
  WHERE @i_gr = students.id_gr;
END;
```

Запуск этой процедуры выполним в другой хранимой процедуре, где средний балл каждого студента заданной группы сравнивается со средним значением результатов тестирования группы, в запросе отображаются результаты студентов, средний балл которых меньше общего.



```
DROP PROC avergrade_stud;
CREATE PROC avergrade_stud
@gr CHAR(20)
AS
BEGIN
  DECLARE @i_gr INT;
  SELECT @i_gr = id_gr FROM groups
  WHERE name_gr=@gr;

  DECLARE @aggr FLOAT;
  EXEC average_grade 'ПО-34', @aggr OUTPUT
  SELECT @aggr AS 'Средний балл в группе';

  SELECT students.surname_stud AS 'Фамилия', name_gr AS 'Группа',SUM(CAST(assessment AS FLOAT))/COUNT(CAST(assessment A
  FROM students
  INNER JOIN results ON results.id_stud=students.id_stud
  INNER JOIN groups ON students.id_gr=groups.id_gr
  GROUP BY results.id_stud, students.surname_stud, students.id_gr, name_gr
  HAVING @i_gr = students.id_gr AND AVG(CAST(assessment AS FLOAT))<@aggr;
END;

EXEC avergrade_stud 'ПО-34'
```

Содержание отчета

1 Название работы
2 Цель работы
3 Перечень технических средств обучения
4 Порядок выполнения работы
5 Вывод

Варианты заданий

Варианты заданий представлены в практической работе № 13.

Используемая литература

– Г.Н.Федорова Основы проектирования баз данных. М.: Академия, 2020
– Г.Н.Федорова Разработка, администрирование и защита баз данных. М.: Академия, 2018
– https://metanit.com/sql/sqlserver/8.4.php
– https://metanit.com/sql/sqlserver/11.3.php