Лабораторная работа № 6.
Хранимые процедуры

Цель работы

Целью выполнения данной лабораторной работы является изучение операторов для создания и вызова хранимых процедур в языке SQL; получение навыков их использования.

Теоретические сведения

Хранимая процедура — объект базы данных, представляющий собой набор SQL-инструкций, который компилируется один раз и хранится на сервере базы данных. Хранимые процедуры очень похожи на обыкновенные процедуры языков высокого уровня, у них могут быть входные и выходные параметры и локальные переменные, в них могут производиться числовые вычисления и операции над символьными данными, результаты которых могут присваиваться переменным и параметрам. В хранимых процедурах могут выполняться стандартные операции с базами данных. Кроме того, в хранимых процедурах возможны циклы и ветвления, то есть в них могут использоваться инструкции управления процессом исполнения.

Создание хранимой процедуры предполагает решение следующих задач:

- определение типа создаваемой хранимой процедуры: временная или пользовательская.
- планирование прав доступа. При создании хранимой процедуры следует учитывать, что она будет иметь те же права доступа к объектам базы данных, что и создавший ее пользователь;
- определение параметров хранимой процедуры. Подобно процедурам, входящим в состав большинства языков программирования, хранимые процедуры могут обладать входными и выходными параметрами;
- разработка кода хранимой процедуры. Код процедуры может содержать последовательность любых команд SQL, включая вызов других хранимых

процедур.

Создание новой и изменение имеющейся хранимой процедуры осуществляется с помощью следующей команды:

```
{CREATE | ALTER } PROC[EDURE] имя_процедуры
[{@имя_параметра тип_данных} [VARYING]
[=default][OUTPUT] ][,...n]
[WITH { RECOMPILE | ENCRYPTION |
RECOMPILE, ENCRYPTION}]
[FOR REPLICATION]
AS
sql_оператор [...n]
```

Рассмотрим параметры данной команды.

Как видно из синтаксиса команды, не допускается указывать имя владельца, которому будет принадлежать создаваемая процедура, а также имя базы данных, где она должна быть размещена. Таким образом, чтобы разместить создаваемую хранимую процедуру в конкретной базе данных, необходимо выполнить команду `CREATE PROCEDURE` в контексте этой базы данных. При обращении из тела хранимой процедуры к объектам той же базы данных можно использовать укороченные имена, т. е. без указания имени базы данных. Когда же требуется обратиться к объектам, расположенным в других базах данных, указание имени базы данных обязательно.

Для передачи входных и выходных данных в создаваемой хранимой процедуре могут использоваться параметры, имена которых, как и имена локальных переменных, должны начинаться с символа `@`. В одной хранимой процедуре можно задать множество параметров, разделенных запятыми. В теле процедуры не должны применяться локальные переменные, чьи имена совпадают с именами параметров этой процедуры.

Для определения типа данных, который будет иметь соответствующий па-

раметр хранимой процедуры, годятся любые типы данных SQL, включая определенные пользователем. Однако тип данных CURSOR может быть использован только как выходной параметр хранимой процедуры, т. е. с указанием ключевого слова OUTPUT.

Наличие ключевого слова OUTPUT означает, что соответствующий параметр предназначен для возвращения данных из хранимой процедуры. Однако это вовсе не означает, что параметр не подходит для передачи значений в хранимую процедуру. Указание ключевого слова OUTPUT предписывает серверу при выходе из хранимой процедуры присвоить текущее значение параметра локальной переменной, которая была указана при вызове процедуры в качестве значения параметра. Отметим, что при указании ключевого слова OUTPUT значение соответствующего параметра при вызове процедуры может быть задано только с помощью локальной переменной. Не разрешается использование любых выражений или констант, допустимое для обычных параметров.

Ключевое слово VARYING применяется совместно с параметром OUTPUT, имеющим тип CURSOR. Оно определяет, что выходным параметром будет результирующее множество.

Ключевое слово DEFAULT представляет собой значение, которое будет принимать соответствующий параметр по умолчанию. Таким образом, при вызове процедуры можно не указывать явно значение соответствующего параметра.

Так как сервер кэширует план исполнения запроса и компилированный код, при последующем вызове процедуры будут использоваться уже готовые значения. Однако в некоторых случаях все же требуется выполнять перекомпиляцию кода процедуры. Указание ключевого слова RECOMPILE предписывает системе создавать план выполнения хранимой процедуры при каждом ее вызове.

Параметр FOR REPLICATION востребован при репликации данных и

включении создаваемой хранимой процедуры в качестве статьи в публикацию.

Ключевое слово ENCRYPTION предписывает серверу выполнить шифрование кода хранимой процедуры, что может обеспечить защиту от использования авторских алгоритмов, реализующих работу хранимой процедуры.

Ключевое слово AS размещается в начале собственно тела хранимой процедуры, т. е. набора команд SQL, с помощью которых и будет реализовываться то или иное действие. В теле процедуры могут применяться практически все команды SQL, объявляться транзакции, устанавливаться блокировки и вызываться другие хранимые процедуры. Выход из хранимой процедуры можно осуществить посредством команды RETURN.

Удаление хранимой процедуры осуществляется командой:

```
DROP PROCEDURE {имя_процедуры}
```

Для выполнения хранимой процедуры используется команда:

```
[[ EXEC [ UTE] имя_процедуры
[[@имя_параметра=]{значение | @имя_переменной}
[OUTPUT ]|[DEFAULT ]]
```

Если вызов хранимой процедуры не является единственной командой в пакете, то присутствие команды EXECUTE обязательно. Более того, эта команда требуется для вызова процедуры из тела другой процедуры или триггера.

Использование ключевого слова OUTPUT при вызове процедуры разрешается только для параметров, которые были объявлены при создании процедуры с ключевым словом OUTPUT.

Когда же при вызове процедуры для параметра указывается ключевое слово DEFAULT, то будет использовано значение по умолчанию. Естественно, указанное слово DEFAULT разрешается только для тех параметров, для которых определено значение по умолчанию.

Из синтаксиса команды EXECUTE видно, что имена параметров могут быть опущены при вызове процедуры. Однако в этом случае пользователь дол-

жен указывать значения для параметров в том же порядке, в каком они перечислялись при создании процедуры. Присвоить параметру значение по умолчанию, просто пропустив его при перечислении нельзя. Если же требуется опустить параметры, для которых определено значение по умолчанию, достаточно явного указания имен параметров при вызове хранимой процедуры. Более того, таким способом можно перечислять параметры и их значения в произвольном порядке.

Отметим, что при вызове процедуры указываются либо имена параметров со значениями, либо только значения без имени параметра. Их комбинирование не допускается.

Пример 1. Процедура без параметров. Разработать процедуру для получения названий и стоимости товаров, приобретенных Ивановым.

```
CREATE PROC my_proc1
AS
SELECT Товар.Название, Товар.Цена*Сделка.Количество
AS Стоимость, Клиент.Фамилия
FROM Клиент INNER JOIN
(Товар INNER JOIN Сделка
ON Товар.КодТовара=Сделка.КодТовара)
ON Клиент.КодКлиента=Сделка.КодКлиента
WHERE Клиент.Фамилия='Иванов'
```

Для обращения к процедуре можно использовать команды:

```
EXEC my_proc1 или my_proc1
```

Процедура возвращает набор данных.

Пример 2. Процедура без параметров. Создать процедуру для уменьшения цены товара первого сорта на 10%.

```
CREATE PROC my_proc2
AS
UPDATE Товар SET Цена=Цена*0.9
```

```
  WHERE Сорт='первый'
```

Для обращения к процедуре можно использовать команды:

```
EXEC my_proc2 или my_proc2
```

Процедура не возвращает никаких данных.

Пример 3. Процедура с входным параметром. Создать процедуру для получения названий и стоимости товаров, которые приобрел заданный клиент.

```
CREATE PROC my_proc3
  @k VARCHAR(20)
AS
SELECT Товар.Название,
       Товар.Цена*Сделка.Количество
  AS Стоимость, Клиент.Фамилия
  FROM Клиент INNER JOIN
  (Товар INNER JOIN Сделка
  ON Товар.КодТовара=Сделка.КодТовара)
  ON Клиент.КодКлиента=Сделка.КодКлиента
  WHERE Клиент.Фамилия=@k
```

Для обращения к процедуре можно использовать команды:

```
EXEC my_proc3 'Иванов' или my_proc3 @k='Иванов'
```

Пример 4. Процедура с входными параметрами. Создать процедуру для уменьшения цены товара заданного типа в соответствии с указанным %.

```
CREATE PROC my_proc4
  @t VARCHAR(20), @p FLOAT
AS
UPDATE Товар SET Цена=Цена*(1-@p)
  WHERE Тип=@t
```

Для обращения к процедуре можно использовать команды:

```
EXEC my_proc4 'Вафли',0.05 или
EXEC my_proc4 @t='Вафли', @p=0.05
```

Пример 5. Процедура с входными параметрами и значениями по умолчанию. Создать процедуру для уменьшения цены товара заданного типа в соответствии с указанным %.

```
CREATE PROC my_proc5
  @t VARCHAR(20)='Конфеты',
  @p FLOAT=0.1
AS
UPDATE Товар SET Цена=Цена*(1-@p)
  WHERE Тип=@t
```

Для обращения к процедуре можно использовать команды:

`EXEC my_proc5 'Вафли',0.05` или

`EXEC my_proc5 @t='Вафли', @p=0.05` или

`EXEC my_proc5 @p=0.05`

В этом случае уменьшается цена конфет (значение типа не указано при вызове процедуры и берется по умолчанию).

```
EXEC my_proc5
```

В последнем случае оба параметра (и тип, и проценты) не указаны при вызове процедуры, их значения берутся по умолчанию.

Пример 6. Процедура с входными и выходными параметрами. Создать процедуру для определения общей стоимости товаров, проданных за конкретный месяц.

```
CREATE PROC my_proc6
  @m INT,
  @s FLOAT OUTPUT
AS
SELECT @s=Sum(Товар.Цена*Сделка.Количество)
  FROM Товар INNER JOIN Сделка
  ON Товар.КодТовара=Сделка.КодТовара
  GROUP BY Month(Сделка.Дата)
```

```
  HAVING Month(Сделка.Дата)=@m
```

Для обращения к процедуре можно использовать команды:

```
DECLARE @st FLOAT
EXEC my_proc6 1,@st OUTPUT
SELECT @st
```

Этот блок команд позволяет определить стоимость товаров, проданных в январе (входной параметр месяц указан равным 1).

Создать процедуру для определения общего количества товаров, приобретенных фирмой, в которой работает заданный сотрудник.

Сначала разработаем процедуру для определения фирмы, где работает сотрудник.

```
CREATE PROC my_proc7
  @n VARCHAR(20),
  @f VARCHAR(20) OUTPUT
AS
SELECT @f=Фирма
  FROM Клиент
  WHERE Фамилия=@n
```

Пример 7. Использование вложенных процедур. Создать процедуру для определения общего количества товаров, приобретенных фирмой, в которой работает заданный сотрудник.

Затем создадим процедуру, подсчитывающую общее количество товара, который закуплен интересующей нас фирмой.

```
CREATE PROC my_proc8
  @fam VARCHAR(20),
  @kol INT OUTPUT
AS
DECLARE @firm VARCHAR(20)
EXEC my_proc7 @fam,@firm OUTPUT
```

```
SELECT @kol=Sum(Сделка.Количество)
  FROM Клиент INNER JOIN Сделка
  ON Клиент.КодКлиента=Сделка.КодКлиента
  GROUP BY Клиент.Фирма
  HAVING Клиент.Фирма=@firm
```

Вызов процедуры осуществляется с помощью команды:

```
DECLARE @k INT
EXEC my_proc8 'Иванов',@k OUTPUT
SELECT @k
```

Ход выполнения работы

1. Изучить теоретические сведения.
2. По указанию преподавателя разработать, реализовать и выполнить несколько хранимых процедур.
3. Составить отчет о выполнении лабораторной работы.
4. Подготовить ответы на контрольные вопросы.

Контрольные вопросы

1. Запишите синтаксис создания хранимой процедуры.
2. Какие типы параметров используются при передаче в хранимую процедуру? Как получить значение в качестве результата выполнения хранимой процедуры?
3. Запишите синтаксис вызова хранимой процедуры.
4. Как определить локальные переменные внутри хранимой процедуры?