Министерство образования и науки Российской Федерации

Калужский филиал
федерального государственного бюджетного образовательного
учреждения высшего образования
**«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»**
(КФ МГТУ им. Н.Э. Баумана)

С.А. Глебов

ИСПОЛЬЗОВАНИЕ ХРАНИМЫХ ПРОЦЕДУР

Методические указания по выполнению лабораторной работы
по курсу «Базы данных»

Калуга – 2018

УДК 004.65
ББК 32.972.134
Г53

Методические указания составлены в соответствии с учебным планом КФ МГТУ им. Н.Э. Баумана по направлению подготовки 09.03.04 «Программная инженерия» кафедры «Программного обеспечения ЭВМ, информационных технологий и прикладной математики».

Методические указания рассмотрены и одобрены:

- Кафедрой «Программного обеспечения ЭВМ, информационных технологий и прикладной математики» (ФН1-КФ) протокол № 7 от « 21 » февраля 2018 г.

И.о. зав. кафедрой ФН1-КФ _____ к.т.н., доцент Ю.Е. Гагарин

- Методической комиссией факультета ФНК протокол № 2 от «28» 02 2018 г.

Председатель методической комиссии факультета ФНК _____ к.х.н., доцент К.Л. Анфилов

- Методической комиссией КФ МГТУ им.Н.Э. Баумана протокол № 2 от «06» 03 2018 г.

Председатель методической комиссии КФ МГТУ им.Н.Э. Баумана _____ д.э.н., профессор О.Л. Перерва

Рецензент:
к.т.н., доцент кафедры ЭИУ6-КФ _____ А.Б. Лачихина

Авторы
к.ф.-м.н., доцент кафедры ФН1-КФ _____ С.А. Глебов

Аннотация

Методические указания по выполнению лабораторной работы по курсу «Базы данных» содержат руководство по созданию хранимых процедур выборки данных и действия над данными, а также задание на выполнение лабораторной работы.

Предназначены для студентов 3-го курса бакалавриата КФ МГТУ им. Н.Э. Баумана, обучающихся по направлению подготовки 09.03.04 «Программная инженерия».

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Настоящие методические указания составлены в соответствии с программой проведения лабораторных работ по курсу «Базы данных» на кафедре «Программное обеспечение ЭВМ, информационные технологии и прикладная математика» факультета фундаментальных наук Калужского филиала МГТУ им. Н.Э. Баумана.

Методические указания, ориентированные на студентов 3-го курса направления подготовки 09.03.04 «Программная инженерия», содержат руководство по созданию хранимых процедур выборки данных и действий над данными и задание на выполнение лабораторной работы.

ЦЕЛЬ И ЗАДАЧИ РАБОТЫ, ТРЕБОВАНИЯ К РЕЗУЛЬТАТАМ ЕЕ ВЫПОЛНЕНИЯ

Целью выполнения лабораторной работы является сформировать практические навыки разработки хранимых процедур.

Основными задачами выполнения лабораторной работы являются:

- Научиться создавать хранимые процедуры выборки
- Изучить оператор suspend
- Научиться создавать хранимые процедуры действия.

Результатами работы являются:

- Разработанные хранимые процедуры выборки данных и действий над данными.
- Подготовленный отчет.

ХРАНИМЫЕ ПРОЦЕДУРЫ

Хранимая процедура — это модуль, написанный на процедурном языке, компилятор которого встроен в ядро сервера и хранящийся в базе данных как метаданные (то есть как данные о данных). Хранимую процедуру можно вызывать из программы.

Существует две разновидности хранимых процедур:

Процедуры выбора могут возвращать более одного значения. В приложении имя хранимой процедуры выбора подставляется в оператор `SELECT` вместо имени таблицы или просмотра.

Процедуры действия вообще могут не возвращать данных и используются для реализации каких-либо действий.

Хранимым процедурам можно передавать параметры и получать обратно значения параметров, измененные в соответствии с алгоритмами работы хранимых процедур.

Преимущества использования хранимых процедур:

- одну процедуру можно использоваться многими приложениями;
- разгрузка приложений клиента путем переноса части кода на сервер и вследствие этого — упрощение клиентских приложений;
- при изменении хранимой процедуры все изменения немедленно становятся доступны для всех клиентских приложений; при внесении же изменений в приложение клиента требуется повторное распространение новой версии клиентского приложения между пользователями;
- улучшенные характеристики выполнения, связанные с тем, что хранимые процедуры выполняются сервером, в частности — уменьшенный сетевой трафик.

Создание хранимой процедуры

Хранимая процедура создается оператором

CREATE PROCEDURE <stored procedure name>
  [(<in_parametr> <datatype>
  [,<in_parametr > <datatype>...])]

```
[RETURNS
   (<out_parametr> <datatype>
  [,<out_parametr > <datatype>...])]
AS
[<declare local variable>]
BEGIN
  < statements>
END
```

Входные параметры служат для передачи в процедуру значений из вызывающего приложения. Входные параметры по области видимости приравниваются к локальным переменным, т.е. существуют только в данной процедуре. Изменять значения входных параметров в теле процедуры можно, но следует помнить об их области видимости.

Выходные параметры служат для возврата результирующих значений. Значения выходных параметров устанавливаются в теле процедуры и после окончания ее работы передаются в вызывающее приложение.

И входные и выходные (вместе со словом RETURNS) параметры могут быть опущены, если в них нет необходимости.

Тип должен быть указан для каждого параметра в отдельности.

Следующая хранимая процедура FIND_MAX_LEN возвращает в выходном параметре max_len максимальную длину фильмов режиссера, имя которого передается во входном параметре in_name_prod*:*

```
CREATE PROCEDURE find_max_len (in_name_prod
VARCHAR(30))
RETURNS(max_len INTEGER)
AS
BEGIN
  SELECT MAX(len)
  FROM movie
  WHERE id_pr = (SELECT id_pr FROM prod
                 WHERE fio = :in_name_prod)
  INTO :max_len;
  SUSPEND;
END
```

В приведенной процедуре за ненадобностью отсутствует определение локальных переменных.

Алгоритмический язык хранимых процедур

Для написания тела хранимой процедуры применяют особый алгоритмический язык. Этот язык применяется также для написания триггеров.

Рассмотрим конструкции алгоритмического языка хранимых процедур и триггеров.

Локальные переменные, если они определены в процедуре, имеют срок жизни от начала выполнения процедуры и до ее окончания. Вне процедуры такие локальные переменные неизвестны, и попытка обращения к ним вызовет ошибку. Локальные переменные используют для хранения промежуточных значений.

Формат объявления локальных переменных:

DECLARE VARIABLE <имя переменной > <тип данных>;

Также как и для параметров, тип должен быть указан для каждой переменной. Н-р,

DECLARE VARIABLE A INTEGER, B INTEGER, C INTEGER;

Операторные скобки BEGIN... END, во-первых, ограничивают тело процедуры, а во-вторых, могут использоваться для указания границ составного оператора.

Под простым оператором понимается единичное разрешенное действие, например: Len = 92;

Под составным оператором понимается группа простых или составных операторов, заключенная в операторные скобки BEGIN... END.

Оператор присваивания служит для занесения значений в переменные. Его формат:

Имя переменной = выражение;

где в качестве выражения могут выступать переменные, арифметические и строковые выражения, в которых можно использовать встроенные функции, функции, определенные пользователем, а также генераторы. Пример:

OUT_TITLE = UPPER(TITLE);

Условный оператор IF ... THEN ... ELSE имеет такой же формат, как и в Object Pascal:

IF (<условие>) THEN < оператор 1> [ELSE < оператор 2>]

В случае, если условие истинно, выполняется оператор 1, если ложно — оператор 2. В отличие от Object Pascal условие всегда должно заключаться в круглые скобки.

Оператор SELECT используется в хранимой процедуре для выдачи единичной строки. По сравнению с синтаксисом обычного оператора SELECT в процедурный оператор добавлено предложение

INTO :переменная [, :переменная...]

Оно служит для указания переменных или выходных параметров, в которые должны быть записаны значения, возвращаемые оператором SELECT (те результирующие значения, которые перечисляются после ключевого слова SELECT*).*

Приводимый ниже оператор SELECT возвращает среднее и сумму по столбцу LENGTH и записывает их соответственно в AVG_LEN и SUM_LEN, которые могут быть как локальными переменными, так и выходными параметрами процедуры. Расчет среднего и суммы по столбцу LEN производится только для записей, у которых значение столбца TITLE совпадает с содержимым IN_TITLE (входной параметр или локальная переменная).

```
SELECT AVG(LEN), SUM(LEN)
FROM MOVIE
WHERE TITLE = :IN_TITLE
INTO :AVG_LEN, :SUM_LEN;
```

Использование символа <:> перед именем переменной (или параметра) принято только в операторах внутренних запросов, с тем чтобы отличать имя столбца от имени переменной. В теле ХП вне запросов идентификаторы переменных используются без <:>.

Оператор *FOR SELECT... DO* имеет следующий формат:

FOR <оператор SELECT> DO <оператор>

Оператор SELECT представляется в расширенном синтаксисе для алгоритмического языка хранимых процедур и триггеров, то есть в нем может присутствовать предложение *INTO.*

Алгоритм работы оператора FOR SELECT ... DO заключается в следующем. Выполняется оператор SELECT, и для каждой строки полученного результирующего набора данных выполняется оператор, следующий за словом DO. Этим оператором часто бывает SUSPEND (см. ниже), который приводит к возврату выходных параметров в вызывающее приложение.

Следующая процедура выдает все фильмы режиссера, определяемого входным параметром IN_PROD.

```
CREATE PROCEDURE HIS_MOVIES (IN_PROD VARCHAR(30) CHARACTER SET WIN1251)
RETURNS (OUT_TITLE VARCHAR(40) CHARACTER SET WIN1251,
         OUT_YEAR INTEGER,
         OUT_LEN INTEGER)
AS
BEGIN
  FOR SELECT TITLE, "YEAR", LEN
    FROM MOVIE M, PROD P
    WHERE (FIO = :IN_PROD) AND (M.ID_PR = P.ID_PR)
    INTO :OUT_TITLE, :OUT_YEAR, :OUT_LEN
  DO SUSPEND;
END
```

Рассмотрим логику работы оператора FOR SELECT... DO этой процедуры. Сначала выполняется оператор SELECT, который возвращает название фильма, его год и длину для каждой записи, у которой столбец FIO содержит значение, идентичное значению во входном параметре IN_PROD. Указанные значения записываются в выходные параметры (соответственно OUT_TITLE, OUT_YEAR, OUT_LEN). Имени параметра в этом случае предшествует двоеточие После выдачи каждой записи результирующего НД выполняется оператор, следующий за словом DO. В данном случае это оператор SUSPEND. Он возвращает значения выходных параметров вызвавшему приложению и приостанавливает выполнение процедуры. До запроса следующей порции выходных параметров от вызывающего приложения.

Такая процедура является процедурой выбора, поскольку она может возвращать множественные значения выходных параметров в вызывающее приложение. Обычно запрос к такой хранимой процедуре из вызывающего приложения осуществляется при помощи оператора SELECT, например:

SELECT MAX_LEN FROM FIND_MAX_LEN("Э.Рязанов")

Оператор SUSPEND передает в вызывающее приложение значения результирующих параметров (перечисленных после слова RETURNS в описании функции), имеющие место на момент выполнения SUSPEND. После этого выполнение хранимой процедуры приостанавливается. Когда от оператора SELECT вызывающего приложения приходит запрос на следующее значение выходных параметров, выполнение хранимой процедуры возобновляется.

Процедура MOVIESTARLIST возвращает актерский состав каждого фильма (у которых он есть) в виде иерархии:

```
CREATE PROCEDURE MOVIESTARLIST
RETURNS (OUT_PARAM VARCHAR(50))
AS
DECLARE VARIABLE "tmp" VARCHAR(50);
DECLARE VARIABLE ID_M INTEGER;
begin
  for select M."TITLE", M."ID_M"
      from "MOVIE" M
      into :"OUT_PARAM", :"ID_M"
  do
    begin
     if (EXISTS (SELECT *
                 FROM "StarIN"
                 WHERE "ID_M"=:"ID_M")) then
      begin
       suspend;
       "tmp" = "OUT_PARAM";
       for select S."FIO"
           from "StarIN" SI INNER JOIN "STAR" S
```

OUT_PARAM
Джентльмены удачи
- Г. Вицин
- Ю. Никулин
- Е. Леонов
Операция "Ы" и др.
- Г. Вицин
- Ю. Никулин
- Е. Моргунов
Гараж
- А. Мягков
- Л. Ахеджакова
....

```
            ON S."ID_ST"=SI."ID_S"
          where SI."ID_M"=:"ID_M"
          into :"OUT_PARAM"
       do
         begin
          "OUT_PARAM" = '  - ' || "OUT_PARAM";
           suspend;
         end
      end
    end
END
```

Оператор WHILE ... DO имеет такой формат:

WHILE (<условие>) DO < оператор>

Алгоритм выполнения оператора: в цикле проверяется выполнение условия; если оно истинно, выполняется оператор. Цикл продолжается до тех пор, пока условие не перестанет выполняться.

Рассмотрим процедуру SUM_0_N, которая подсчитывает сумму всех чисел от 0 до числа, определяемого входным параметром N. Вычисление суммы реализовано в цикле с использованием оператора WHILE...DO.

```
CREATE PROCEDURE BBB (N INTEGER)
RETURNS (S INTEGER)
AS
BEGIN
  WHILE (N>0) DO
    BEGIN
      S=S+N;
      N=N-1;
    END
END;
```

Оператор EXIT инициирует прекращение выполнения процедуры и выход в вызывающее приложение.

Процедура MAX_VALUE возвращает максимум из двух чисел, передаваемых как входные параметры; в случае, если одно из чисел

имеет значение NULL, процедур завершается (в этом случае выходной параметр содержит значение NULL):

```
CREATE PROCEDURE MAX_VALUE (A INTEGER, B INTEGER)
RETURNS (M_V INTEGER)
AS
BEGIN
  IF (A IS NULL OR B IS NULL) THEN EXIT;
  IF (A > B) THEN M_V = A;
        ELSE M_V = B;
END
```

EXIT пользуются достаточно редко и в основном для того, чтобы прервать цикл при достижении некоторого условия. При этом, последняя извлеченная строка не будет возвращена. Если же эта строка нужна, то стоит воспользоваться последовательностью операторов:

```
SUSPEND;
EXIT;
```

Оператор

```
EXECUTE PROCEDURE имя [параметр [, параметр ...]];
[RETURNING_VALUES параметр [, параметр ...]];
```

вызывает другую хранимую процедуру. При этом после имени вызываемой процедуры перечисляются входные параметры, если они есть, а после RETURNING_VALUES выходные параметры.

Перепишем приведенную выше процедуру STAR_LIST таким образом, чтобы из ее тела вызывалась другая процедура, AVG_LEN, возвращающая среднюю длину фильмов указанного жанра:

```
CREATE PROCEDURE AVG_LEN (
   KND VARCHAR(20))
RETURNS (
   OUT_AVG_LEN INTEGER)
AS
BEGIN
  SELECT AVG(LEN)
  FROM MOVIE
  WHERE KIND = :KND
  INTO :OUT_AVG_LEN;
  SUSPEND;
END
```

```
CREATE PROCEDURE STAR_LIST1 (IN_KIND VARCHAR(20))
RETURNS (FIO_STAR VARCHAR(20), DOHOD VARCHAR(20))
AS
DECLARE VARIABLE AVG_LENGTH INTEGER;
BEGIN
  EXECUTE PROCEDURE AVG_LEN (IN_KIND)
  RETURNING_VALUES AVG_LENGTH;
  FOR SELECT FIO
    FROM (MOVIE M INNER JOIN "StarIN" SI ON
M.ID_M=SI.ID_M)
        INNER JOIN STAR S ON SI.ID_S=S.ID_ST
    WHERE LEN > :AVG_LENGTH AND Kind=:IN_KIND
    INTO :FIO_STAR, :DOHOD
  DO
    BEGIN
      IF (DOHOD IS NULL) THEN DOHOD = 'Доход не указан';
      SUSPEND;
    END
END
```

Нетрудно заметить, что EXECUTE PROCEDURE возвращает только единственное значение каждого выходного параметра. Если сама ХП является процедурой выбора, которая возвращает набор данных, то в этом случае выходные параметры получат значения из первой строки этого набора.

ЗАДАНИЕ НА ЛАБОРАТОРНУЮ РАБОТУ

В базе данных, созданной в предыдущей лабораторной работе, разработать хранимые процедуры выборки и действия.

КОНТРОЛЬНЫЕ ВОПРОСЫ И ЗАДАНИЯ

1. Дайте определение термину «хранимая процедура».
2. Раскройте сущность процедур выбора.
3. Раскройте сущность процедур действия.
4. Перечислите преимущества использования хранимых процедур.
5. Приведите синтаксис команды создания хранимой процедуры.
6. Приведите синтаксис команды объявления локальных переменных в хранимой процедуре.
7. Приведите синтаксис условного оператора.
8. Приведите синтаксис операторов цикла.
9. Опишите назначение оператора SUSPEND.

ФОРМА ОТЧЕТА ПО ЛАБОРАТОРНОЙ РАБОТЕ

На выполнение лабораторной работы отводится 2 занятия (4 академических часа: 3 часа на выполнение и сдачу лабораторной работы и 1 час на подготовку отчета).

Номер варианта студенту выдается преподавателем.

Отчет на защиту предоставляется в печатном виде.

Структура отчета (на отдельном листе(-ах)): титульный лист, формулировка задания (вариант), ход выполнения работы, результаты выполнения работы, выводы.

ОСНОВНАЯ ЛИТЕРАТУРА

1. Карпова, Т.С. Базы данных: модели, разработка, реализация : учебное пособие / Т.С. Карпова. - 2-е изд., исправ. - Москва : Национальный Открытый Университет «ИНТУИТ», 2016. - 241 с. : ил. ; То же [Электронный ресурс]. - URL: http://biblioclub.ru/index.php?page=book&id=429003

2. Давыдова, Е.М. Базы данных [Электронный ресурс] : учеб. пособие / Е.М. Давыдова, Н.А. Новгородова. — Электрон. дан. — Москва : ТУСУР, 2007. — 166 с. — Режим доступа: https://e.lanbook.com/book/11636. — Загл. с экрана.

3. Харрингтон, Д. Проектирование объектно ориентированных баз данных [Электронный ресурс] — Электрон. дан. — Москва : ДМК Пресс, 2007. — 272 с. — Режим доступа: https://e.lanbook.com/book/1231. — Загл. с экрана.

ДОПОЛНИТЕЛЬНАЯ ЛИТЕРАТУРА

4. Голицина О.Л., Максимов Н.В., Попов И.И. Базы данных: учеб. пособие. – М.: Форум:Инфра-М, 2007.

5. Гагарин Ю.Е. Применение языка SQL в MS Access: учебно-методическое пособие. – М.: МГТУ им. Н.Э. Баумана, 2012.

Электронные ресурсы:

1. Научная электронная библиотека http://eLIBRARY.RU
2. Электронно-библиотечная система http://e.lanbook.com
3. Электронно-библиотечная система «Университетская библиотека онлайн» http://biblioclub.ru
4. Электронно-библиотечная система IPRBook http://www.iprbookshop.ru