МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра «Кибербезопасность информационных систем»

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

ПО ВЫПОЛНЕНИЮ ЛАБОРАТОРНОЙ РАБОТЫ № 2

«ЗНАКОМСТВО С ЯЗЫКОМ PL/SQL. УПРАВЛЯЮЩИЕ КОНСТРУКЦИИ. ИСПОЛЬЗОВАНИЕ КУРСОРОВ»

ПО ДИСЦИПЛИНЕ «ORACLE»

Ростов-на-Дону

ДГТУ

2019

УДК 004.65

Составитель: С.П. Новиков

Методические указания по выполнению лабораторной работы № 2 «Знакомство с языком PL/SQL. Управляющие конструкции. Использование курсоров». – Ростов-на-Дону : Донской гос. техн. ун-т, 2019. – 13 c.

В методической разработке рассматриваются управляющие конструкциями. Рассматривается описания процедур, курсоров, обработки исключительных ситуаций. Даны задания к лабораторной работе, помогающие закрепить на практике полученные знания.

УДК 004.65

Печатается по решению редакционно-издательского совета   
Донского государственного технического университета

Научный редактор и.о. зав. кафедрой «Кибербезопасность информационных систем» канд. техн. наук, доцент Д.А. Короченцев

Ответственный за выпуск старший преподаватель кафедры «Кибербезопасность информационных систем» С.П. Новиков

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

В печать \_\_\_.\_\_\_.2019 г.

Формат 60x84/16. Объем \_\_\_\_ усл. п. л.

Тираж\_\_\_ экз. Заказ № \_\_\_.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Издательский центр ДГТУ

Адрес университета и полиграфического предприятия:

344000, г. Ростов-на-Дону, пл. Гагарина, 1

© Донской государственный   
 технический университет, 2019

**Цель работы:** познакомиться с управляющими конструкциями языка PL/SQL на примере моделирования процессов университета, научиться использованию процедур и курсоров.

1. **Теоретические сведения**
   1. **Процедуры**

Процедура представляет собой модуль, выполняющий одно или несколько действий. Поскольку вызов процедуры в PL/SQL является отдельным исполняемым оператором, блок кода PL/SQL может состоять только из вызова процедуры. Процедуры относятся к числу ключевых компонентов модульного кода, обеспечивающих оптимизацию и повторное использование программной логики [1].

Для создания процедур и триггеров необходимо, чтобы у пользователя были права на создание этих объектов, для этого возможно будет необходимо «дать» эти права.

**Синтаксис оператора создания процедуры**:

CREATE OR REPLACE PROCEDURE <имя\_процедуры> (параметр1 <тип параметра> <тип данных>, параметр2 <тип параметра> <тип данных> ….)

as (is)

-- после as(is) описание переменных без DECLARE

begin

-- тело процедуры

end;

*Тип параметра* in – только входной параметр, out – только выходной параметр, inout – и то и другое.

Для вызова процедуры ее имя должно быть помещено в программе между операторами begin и end.

Чтобы выводить результаты, возможно, будет необходимо подключить пакет DBMS\_OUTPUT.PUT\_LINE(значение) [2, 3].

Процедура вызывается как исполняемая команда PL/SQL. Другими словами, ее вызов должен заканчиваться точкой с запятой (;) и может предшествовать другим командам SQL либо PL/SQL (если таковые имеются) в исполняемом разделе блока PL/SQL или следовать за ними:

BEGIN

procedure\_name(param1, param2);

END;

Если процедура не имеет параметров, она может вызываться с пустыми круглыми скобками или без них:

procedure\_name;

procedure\_name();

Часть определения процедуры, предшествующая ключевому слову IS, называется **заголовком процедуры, или сигнатурой** [1]. Заголовок предоставляет программисту всю информацию, необходимую для вызова процедуры:

* имя процедуры;
* условие AUTHID (если имеется);
* список ACCESSIBLE BY (если имеется — новая возможность Oracle Database 12c).

В идеале программист при виде заголовка процедуры должен понять, что делает эта процедура и как она вызывается.

В **теле процедуры** содержится код, необходимый для реализации этой процедуры; тело состоит из объявления, исполняемого раздела и раздела исключений этой процедуры. Все, что следует за ключевым словом IS, образует тело процедуры. Разделы исключений и объявлений не являются обязательными [1].

Вы можете указать имя процедуры за завершающим ключевым словом END:

PROCEDURE display\_stores (region\_in IN VARCHAR2)

IS

BEGIN

-- тело процедур

END display\_stores;

Имя служит меткой, явно связывающей конец программы с ее началом. Она особенно полезна для процедур, занимающих несколько страниц или входящих в серию процедур и функций в теле пакета.

Ключевое слово RETURN обычно ассоциируется с функциями, поскольку они должны возвращать значения. Однако PL/SQL позволяет использовать команду RETURN в процедурах. Версия этой команды для процедур не принимает выражений и не может возвращать значения в вызывающий программный модуль — она просто прекращает выполнение процедуры и возвращает управление вызывающему коду. Использовать эту разновидность RETURN не рекомендуется, поскольку в этом случае в процедуре появляются две и более точки выхода, а это усложняет логику выполнения. Избегайте использования RETURN и GOTO для обхода нормальной управляющей структуры в программных элементах [1].

* 1. **Курсоры**

При выполнении команды SQL из PL/SQL РСУБД Oracle назначает ей приватную рабочую область, а некоторые данные записывает в системную глобальную область (SGA, System Global Area). В приватной рабочей области содержится информация о команде SQL и набор данных, возвращаемых или обрабатываемых этой командой. PL/SQL предоставляет программистам несколько механизмов доступа к этой рабочей области и содержащейся в ней информации; все они так или иначе связаны с определением курсоров и выполнением операций с ними [1].

* *Неявные курсоры*. Команда SELECT...INTO считывает одну строку данных и присваивает ее в качестве значения локальной переменной программы. Это простейший (и зачастую наиболее эффективный) способ доступа к данным, но он часто ведет к написанию сходных и даже одинаковых SQL-команд SELECT во многих местах программы.
* *Явные курсоры.* Запрос можно явно объявить как курсор в разделе объявлений локального блока или пакета. После этого такой курсор можно будет открывать и выбирать из него данные в одной или нескольких программах, причем возможности управления явным курсором шире, чем у неявного.
* *Курсорные переменные.* Курсорные переменные (в объявлении которых задается тип REF CURSOR) позволяют передавать из программы в программу указатель на результирующий набор строк запроса. Любая программа, для которой доступна такая переменная, может открыть курсор, извлечь из него необходимые данные и закрыть его.
* *Курсорные выражения.* Ключевое слово CURSOR превращает команду SELECT в набор REF CURSOR, который может использоваться совместно с табличными функциями для повышения производительности приложения.
* *Динамические SQL-запросы.* Oracle позволяет динамически конструировать и выполнять запросы с использованием либо встроенного динамического SQL, либо программ пакета DMBS\_SQL. Этот встроенный пакет описывается в документации Oracle.
  1. **Основные принципы работы с курсорами**

Курсор проще всего представить как указатель на таблицу в базе данных. Например, следующее объявление связывает всю таблицу employee с курсором employee\_cur:

CURSOR employee\_cur IS SELECT \* FROM employee;

Объявленный курсор можно открыть:

OPEN employee\_cur;

Далее из него можно выбирать строки:

FETCH employee\_cur INTO employee\_rec;

Завершив работу с курсором, его следует закрыть:

CLOSE employee\_cur;

В этом случае каждая выбранная из курсора запись представляет строку таблицы employee [1-3]. Однако с курсором можно связать любую допустимую команду SELECT. В следующем примере в объявлении курсора объединяются три таблицы:

DECLARE

CURSOR joke\_feedback\_cur

IS

SELECT J.name, R.laugh\_volume, C.name

FROM joke J, response R, comedian C

WHERE J.joke\_id = R.joke\_id AND R.joker\_id = C.joker\_id;

BEGIN

...

END;

В данном случае курсор действует не как указатель на конкретную таблицу базы данных — он указывает на виртуальную таблицу или неявное представление, определяемое командой SELECT. Такая таблица называется виртуальной, потому что команда SELECT генерирует данные с табличной структурой, но эта таблица существует только временно, пока программа работает с возвращенными командой данными. Если тройное объединение возвращает таблицу из 20 строк и 3 столбцов, то курсор действует как указатель на эти 20 строк [1-3].

* 1. **Атрибуты курсоров**

Чтобы обратиться к атрибуту курсора, укажите в виде его префикса имя курсора или курсорной переменной и символ %:

имя\_курсора%имя\_атрибута

В качестве имен неявных курсоров используется префикс SQL, например SQL%NOTFOUND. Ниже приведены краткие описания всех атрибутов курсоров.

Атрибут %FOUND возвращает информацию о состоянии последней операции FETCH с курсором. Если последний вызов FETCH для курсора вернул строку, то возвращается значение TRUE, а если строка не была получена, возвращается FALSE. Если курсор еще не был открыт, база данных инициирует исключение INVALID\_CURSOR.

Атрибут %NOTFOUND по смыслу противоположен %FOUND: он возвращает TRUE, если последняя операция FETCH с курсором не вернула строки (как правило, из-за того, что последняя строка уже была прочитана). Если курсор не может вернуть строку из-за ошибки, инициируется соответствующее исключение. Если курсор еще не был открыт, база данных инициирует исключение INVALID\_CURSOR.

Атрибут %ROWCOUNT возвращает количество записей, прочитанных из курсора к моменту запроса атрибута. При исходном открытии курсора атрибут %ROWCOUNT равен 0. При обращении к атрибуту %ROWCOUNT курсора, который еще не был открыт, инициируется исключение INVALID\_CURSOR. После выборки каждой записи %ROWCOUNT увеличивается на единицу.

Атрибут %ISOPEN возвращает TRUE, если курсор открыт; в противном случае возвращается FALSE.

Атрибут %BULK\_ROWCOUNT, предназначенный для использования с командой FORALL, возвращает количество строк, обработанных при каждом выполнении DML.

Атрибут %BULK\_EXCEPTIONS, также предназначенный для использования с командой FORALL, возвращает информацию об исключении, которое может быть инициировано при каждом выполнении DML.

* 1. **Неявный курсор**

Неявный курсор — это команда SELECT, обладающая следующими характеристиками:

* команда SELECT определяется в исполняемом разделе блока, а не в разделе объявлений, как явные курсоры;
* в команде содержится предложение INTO (или BULK COLLECT INTO), которое относится к языку PL/SQL (а не SQL) и представляет собой механизм передачи данных из базы в локальные структуры данных PL/SQL;
* команду SELECT не нужно открывать, выбирать из нее данные и закрывать. Все эти операции осуществляются автоматически.

Общая структура неявного запроса выглядит так:

SELECT список\_столбцов

[BULK COLLECT] INTO список\_переменных

... продолжение команды SELECT...

Если в программе используется неявный курсор, Oracle автоматически открывает его, выбирает строки и закрывает. Однако он может получить информацию о последней выполненной команде SQL, анализируя значения атрибутов неявного курсора SQL.

**1.6. 2 основных подхода для работы с курсорами**

1. Курсору можно передать параметры. Параметры задаются в момент открытия курсора.

2. Можно определять или не определять тип возвращаемого значения. Возможный тип: указанная запись, строка БД, отдельная переменная.

В любом случае, столбцы, определяемые в SELECT, должны совпадать с используемой для получения возвращаемых значений структурой данных.

После объявления курсор может быть открыт, строки выбраны, состояние проверено, и он может быть закрыт. Курсор может иметь любое имя, рекомендуется использовать префикс get\_ или постфиксы \_cur, \_loop.

В имени курсора рекомендуется использовать имя таблицы или как-то по-другому ссылаться на нее.

Список параметров только принимает значения, но не возвращает их. Входные параметры открытия курсора: вычисляется SQL-код и вычисляется соответствующий набор данных, но строки набора программе не возвращаются. Чтобы получить строки одну за другой используется оператор FETCH: выбранная строка остается текущей, пока не будет выбрана следующая. Выбор строк производится только в прямом направлении.

* 1. **Функции для работы со строками:**
* *LENGTH* (строка) – возвращает длину строки в символах;
* *LOWER* (строка) – преобразует все буквы в нижний регистр;
* *UPPER* (строка) – преобразует все буквы в верхний регистр;
* *ASCII* (символ) – возвращает код символа;
* *CHR* (код) – возвращает символ по ascii-коду;
* *INSTR* (строка, подстрока[, n[, m]]) – возвращает позицию m-го включения подстроки в строку начиная с позиции n;
* *CONCAT* (строка1, строка2) – возвращает соединенние двух строк (эквивалентно строка1 || строка2);
* *SUBSTR* (строка, m, n) – возвращает подстроку длиной n, начиная с m-позиции.
  1. **Исключительные ситуации**

Исключительная ситуация – событие, не приводящее к возникновению непоправимой ошибки, но прерывающее нормальное выполнение программы и вызывающее безусловный переход на обработку исключительной ситуации текущего блока.

Некоторые исключительные ситуации (ROW\_NO\_FOUND) являются событиями нормальной обработки. Исключительные ситуации типа VALUE\_ERROR – программная ошибка или неожиданное событие.

Если не установлен обработчик исключительной ситуации, управление передается блоку верхнего уровня.

Исключительная ситуация поднимается вверх по вложенным блокам, пока не встретит обработчик, иначе передается управление вызывающему контексту (SQL PLUS).

Если ожидаемая исключительная ситуация не включена в список или неизвестна какая исключительная ситуация может возникнуть, либо необходимо обработать все исключительные ситуации, тогда применяют выражение: WHEN OTHERS THEN…

Обработчик исключительных ситуаций будет обрабатывать любую возникнувшую ошибку. Можно применить любую комбинацию имен исключительных ситуаций, указав OTHERS в качестве последней.

* 1. **Предотвращение зацикливания**

Иногда внутри обработчика выполняется действие, которое само по себе может вызвать исключительную ситуацию. Может возникнуть бесконечный цикл.

Чтобы предотвратить зацикливание, сам обработчик заключают в блок и устанавливают обработчик для обработчика. Второй обработчик не должен выполнять никаких действий.

EXCEPTION

WHEN OTHERS THEN

BEGIN

if get\_cursor%isopen then close get\_cursor;

end if;

EXCEPTION

WHEN OTHERS THEN NULL;

END;

END;

Всегда следует устанавливать обработчик исключительных ситуаций для самого внешнего блока PL/SQL. При этом пользователь должен получать короткое сообщение об ошибке.

* 1. **Пользовательские исключительные ситуации**

Если программисту не хватает списка стандартных исключительных ситуаций, то можно определить другие исключительные ситуации, присвоив им имена (это необходимое условие). После этого можно ссылаться на исключительные ситуации, используя их имена.

При определении исключительной ситуации используется специальная конструкция – прагма – это директива компилятора, которая обрабатывается во время компиляции.

**Пример**: использования директивы

SET SERVEROUTPUT ON

DECLARE

invalid\_num\_format exception;

PRAGMA exception\_init (invalid\_num\_format, -1481);

x number(10);

BEGIN

SELECT id INTO x FROM table\_name;

EXCEPTION

WHEN invalid\_num\_format THEN

dbms\_output.put\_line ('Ошибка invalid\_num\_format');

WHEN OTHERS THEN

dbms\_output.put\_line('Другая ошибка');

END;

* 1. **Обработка исключительных ситуаций внутри программы**

Рекомендуется программировать блоки с обработчиками исключительных ситуаций для всех выражений PL/SQL. Это позволит продолжить обработку и выполнение PL/SQL оператора в случае возникновения ошибки. Особо важно это для циклов. Если обработку не предусмотреть, то всякое исключение приведет к немедленному завершению цикла и передаче управления обработчику исключительной ситуации охватывающего блока.

Пример:

LOOP //

FETCH master\_cursor INTO master\_rec;

EXIT WHEN master\_cursor%NOTFOUND;

BEGIN

DELETE FROM child\_table WHERE

master\_f\_key=master\_rec.master\_p\_key;

EXCEPTION WHEN OTHERS THEN

status:=SQLCODE;

--вывод сообщения об ошибке;

END;

END LOOP;

1. **Задание для выполнения**

Рекомендуемое программное обеспечение для выполнения задания – Oracle Database 19c.

Рассматриваемые объекты: студенты, учебные курсы, журнал успеваемости.

Таблица «Студенты» содержит следующую информацию:

* Код число
* ФИО строка
* Группа строка

Таблица «Учебные курсы» содержит следующую информацию:

* Код число
* Наименование строка

Таблица «Журнал успеваемости» содержит следующую информацию:

* Код студента(ки) число
* Код предмета число
* Оценка число

Необходимо:

1. Написать процедуру перевода студента в другую группу:

процедура\_перевода (код\_студента, новая\_группа). Процедура должна проверять, чтобы группы были одинакового курса, иначе выводить ошибку.

2. Написать процедуру перевода всех студентов на следующий курс. При каждом вызове курс группы увеличивается на единицу. Для студентов 5-го курса наименование группы становится пустым.

3. Написать процедуру выводящую на экран фамилии задолжников в данной группе, имеющих количество двоек больше заданного. Процедура (группа, количество\_двоек).

1. **Критерии оценивания**

1. Для получения минимального количества баллов, нужно выполнить задание лабораторной работы. Провести обоснование выбора связей, нормализацию таблиц, если нужно, определение необходимых типов данных. Продемонстрировать работу основных процедур и операторов по заданию. Ответить на один теоретический вопрос.

2. Для получения баллов на оценку «хорошо», так же нужно добавить необходимые поля и данные в таблицы по заданию преподавателя. Написать 2 выборки данных из таблиц с помощью курсоров, написать процедуру по заданию преподавателя. Провести обработку исключительных ситуаций. Ответить на один теоретический вопрос.

3. Для получения баллов на оценку «отлично», дополнительно нужно добавить необходимые поля и данные в таблицы по заданию преподавателя. Написать 2 выборки данных из таблиц с помощью курсоров, написать 2 процедуры по заданию преподавателя. Ответить на два теоретических вопроса.

1. **Контрольные вопросы**

1. Перечислите атрибуты курсоров и опишите их.

2. В чём отличие PL/SQL от SQL?

3. В чём разница между явным и неявным курсорами?

4. Как создать процедуру в Oracle?

5. Перечислите стандартные функции для работы со строками.

6. Что представляет собой курсор в Oracle?

1. **Требования к отчету**

Отчёт выполняется каждым студентом индивидуально. Работа должна быть оформлена в электронном виде в формате .doc или .docx и распечатана на листах формата А4.

На титульном листе указываются: наименование учебного учреждения, наименование дисциплины, название и номер работы, вариант, выполнил: фамилия, имя, отчество, группа, проверил: преподаватель ФИО.

Отчет должен содержать:

* название и цель работы;
* краткие теоретические сведения, ответы на контрольные вопросы;
* протокол выполнения лабораторной работы, содержащий листинг программ, составленных при выполнении работы, и результаты их выполнения (в виде скриншотов);
* выводы по результатам работы.

**ЛИТЕРАТУРА**

1. Фейерштейн С., Прибыл Б. Oracle PL/SQL. Для профессионалов. 6-е изд. — СПб.: Питер, 2015. — 1024 с.
2. Oracle Database PL/SQL: Language Reference 12c Release 1 (12.1) [Электронный ресурс] / L. Morin, S. Moore, D. Alpern и др. – 2017. – Режим доступа: https://docs.oracle.com/database/121/LNPLS/E50727-06.pdf(дата обращения: 02.06.2019).
3. Алапати, Сэм Р. А45 Oracle Database 11g: руководство администратора баз данных.: Пер. с англ. — М.: ООО «И.Д. Вильямс», 2010. — 1440 с.