# Лабораторная работа № 9

**Исключительные ситуации, курсоры, обработка ошибок**

|  |
| --- |
| Настройка SQL. Необходимо включить режим ECHO и вывести протокол лаб. работы в файл *<Фамилия студента>.txt*. Этот файл является отчетом о проделанной лаб. работе.  SQL> set echo on  SQL> spool c:\spool. txt  ……………………………………..  Завершить протокол лаб. работы (команда spool off) и направить результаты преподавателю.  SQL> spool off |

В PL/SQL предусмотрены механизмы перехвата и обработки ошибок, возникающих при выполнении программы. При обнаружении ошибки генерируется исключительная ситуация, обработка которой производится в разделе EXCEPTION. Существуют два класса исключительных ситуаций: стандартные и определяемые пользователем.

Стандартные исключительные ситуации делятся на два типа:

имеющие и не имеющие предопределенное имя. Имеющие предопределенное имя исключительные ситуации помимо кода имеют еще и стандартное имя, которое используется для идентификации исключения. Ниже приведены примеры некоторых стандартных исключительных ситуаций, имеющих предопределенные имена.

1) ZERO\_DIVIDE – попытка деления на нуль;

2) NO\_DATA\_FOUND – предложение SELECT...INTO не возвращает

ни одной строки;

3) TOO\_MANY\_ROWS – предложение SELECT...INTO возвращает

более одной строки;

4) INVALID\_CURSOR – попытка выполнения запрещенной операции

с курсором (например, закрытие неоткрытого курсора);

5) CURSOR\_ALREADY\_OPEN – попытка открытия уже открытого

курсора;

6) VALUE\_ERROR – арифметическая ошибка, ошибка преобразова-

ния, усечения или ограничения;

7) INVALID\_NUMBER – отказ в преобразовании строки символов в

число.

Пользовательские исключительные ситуации описываются в разделе DECLARE, устанавливаются в выполняемом разделе, а обрабатываются в разделе EXCEPTION. Описание пользовательской исключительной ситуации выполняется заданием имени исключения и фразы

EXCEPTION. Чтобы сгенерировать исключительную ситуацию и передать управление обработчику пользовательской исключительной ситуации в случае обнаружения ошибки, используется оператор RAISE имя\_пользовательского\_исключения

Для перехвата исключительной ситуации любого типа в раздел EXCEPTION должна быть включена фраза WHEN имя\_исключения THEN текст\_обработчика\_исключения;

Тогда при возникновении соответствующей ошибки, вместо прекращения исполнения программы и выдачи типового сообщения об ошибке, будет выполняться созданный пользователем вариант обработки исключения. Если необходимо, чтобы две или более исключительные ситуации обрабатывались одинаково, то они должны быть записаны в одном операторе WHEN, разделенные ключевым словом OR. Для перехвата всех неописанных исключительных ситуаций используется специальный обработчик OTHERS, который записывается последним в разделе EXCEPTION.

Генерация исключительной ситуации с выдачей соответствующего сообщения в рабочую среду в случае обнаружения ошибки может быть выполнена с помощью следующего оператора:

RAISE\_APPLICATION\_ERROR (errnum, errtext);

где errnum – код ошибки, выбираемый в диапазоне –20000 .. –20999;

errtext – поясняющая символьная строка длиной до 512 байт.

При возникновении исключительной ситуации и отсутствии соответ ствующего обработчика в данном блоке система пытается найти такой обработчик в блоках, охватывающих этот блок. При отсутствии обработчика система вернет ошибку «необработанное исключение».

**П р и м е р ы**

1. Необходимо создать таблицу: BOOKS, указав все необходимые ограничения целостности данных.

Перечень, названия и тип данных столбцов таблицы BOOKS:

--------------------------------------------------------------------------

Код книги CODE\_BOOK Number(5)

Название книги TITLE Varchar2(25)

ФИО автора AUTHOR Varchar2(20)

Цена книги PRICE Number(7)

Издательство PUBLISH\_HOUSE Varchar2(15)

Жанр GENRE Varchar2(15)

--------------------------------------------------------------------------

Информация таблицы BOOKS:

Код кн. Назв. книги ФИО автора Цена Изд-во Жанр

----------------------------------------------------------------------------------------------

1 Гибель Богов Перумов Н. 345 Аст Фантастика

2 Казаки Толстой Л. 5568 Нова Роман

3 Ярость Перумов Н. 1385 Аст Детектив

4 Дюна Герберт Ф. 2668 Нова Фантастика

5 Гибель Титана Кристи А. 2345 Аст Роман

6 Дети Дюны Герберт Ф. 2500 Аст Фантастика

----------------------------------------------------------------------------------------------

Создать программу, которая осуществляет в таблице BOOKS повышение цен на книги жанра «Фантастика». При этом при стоимости книги менее 2000 руб., цена увеличивается на 20 %, а при стоимости больше или равной 2000 руб. – на 10 %.

Данная задача реализуется с помощью явно объявленного пользователем курсора. При этом в первых двух вариантах показываются возможности использования обычного курсора и курсора с параметром.

Приведенное решение демонстрирует два варианта обработки явно объявленного курсора: явную и неявную формы. В явной форме обработки по завершении просмотра строк активного набора осуществляется выход из цикла обработки. При этом используется курсорный атрибут

%NOTFOUND. В неявной форме обработки курсора используется конструкция цикл FOR с курсором. Модифицированное значение цены записывается обратно в таблицу BOOKS с использованием конструкции WHERE CURRENT OF, при этом системе с помощью конструкции FOR UPDATE OF PRICE указывается, что будет осуществляться обновление значений столбца PRICE таблицы BOOKS. В программе неявным способом объявлены переменная типа запись ZAP и скалярная переменная

NEW\_ PRICE.

**а) Использование обычного курсора:**

DECLARE

CURSOR KUR IS --явное объявление курсора KUR

SELECT CODE\_BOOK, PRICE FROM BOOKS

WHERE GENRE = 'Фантастика' FOR UPDATE OF PRICE;

ZAP KUR%ROWTYPE; --объявление переменной-записи

NEW\_PRICE BOOKS.PRICE%TYPE;--объявление переменной

BEGIN

OPEN KUR; --явное открытие курсора

LOOP

FETCH KUR INTO ZAP; --выборка текущей записи

EXIT WHEN KUR%NOTFOUND;--выход из цикла

IF ZAP.PRICE < 2000 THEN --изменение цены

NEW\_PRICE := ZAP.PRICE\*1.2;

ELSE

NEW\_PRICE := ZAP.PRICE\*1.1;

END IF;

UPDATE BOOKS

SET PRICE = NEW\_PRICE --обновление цены

WHERE CURRENT OF KUR;

END LOOP;

CLOSE KUR; --явное закрытие курсора

COMMIT; --завершение транзакции

END;

**б) Использование курсора с параметром:**

DECLARE

CURSOR KUR (GANR BOOKS.GENRE%TYPE) IS --курсор

имеет параметр

SELECT CODE\_BOOK, PRICE FROM BOOKS

WHERE GENRE=GANR FOR UPDATE OF PRICE;

ZAP KUR%ROWTYPE;

NEW\_PRICE BOOKS.PRICE%TYPE;

BEGIN

OPEN KUR ('Фантастика'); -- значение параметра

LOOP

FETCH KUR INTO ZAP;

EXIT WHEN KUR%NOTFOUND;

IF ZAP.PRICE < 2000 THEN

NEW\_PRICE := ZAP.PRICE\*1.2;

ELSE

NEW\_PRICE := ZAP.PRICE\*1.1;

END IF;

UPDATE BOOKS SET PRICE = NEW\_PRICE

WHERE CURRENT OF KUR;

END LOOP;

CLOSE KUR;

COMMIT;

END;

**в) Использование цикла FOR с курсором:**

DECLARE

NEW\_PRICE BOOKS.PRICE%TYPE;

CURSOR KUR IS

SELECT CODE\_BOOK, PRICE FROM BOOKS

WHERE GENRE = 'Фантастика' FOR UPDATE OF PRICE;

BEGIN

FOR ZAP IN KUR LOOP --неявная обработка курсора

IF ZAP.PRICE < 2000 THEN --переменная ZAP неявно объяв-

ляется системой

NEW\_PRICE := ZAP.PRICE\*1.2;

ELSE

NEW\_PRICE := ZAP.PRICE\*1.1;

END IF;

UPDATE BOOKS SET PRICE = NEW\_PRICE

WHERE CURRENT OF KUR;

END LOOP;

COMMIT;

END;

**Oбpaбoткa oшибoк в PL/SQL**

Нельзя создать приложение, которое будет безошибочно работать в любых ситуациях: возможны аппаратные сбои, невыявленные ошибки приложения и ошибки из-за некорректных действий пользователей приложения (клиентов). Если при этом программная ошибка произошла в блоке PL/SQL, вложенном в другой блок, а тот, в свою очередь, вложен в третий блок и т.д., то она может дойти до клиентского приложения. Чтобы устранить возможную отмену большого объема ранее выполненных операций и трафик из-за возвращаемых клиенту ошибок, чтобы посылать клиенту точные сообщения о причине ошибки и способе ее устранения (если она все же дошла до клиента), разработчики приложения должны предусматривать возможные программные ошибки и создавать процедуры, адекватно реагирующие на них.

В PL/SQL предусмотрен механизмы перехвата и обработки ошибок, возникающих при выполнении программы. Эти механизмы называются исключительными ситуациями.  
  
Когда программа обнаруживает заданное условие ошибки, то вызывается соответствующая исключительная ситуация. Обработки исключительных ситуаций в программе производится в разделе EXCEPTION.  
При обнаружении исключительной ситуации, обработка основного тела программы останавливается и управление передается соответствующему обработчику исключительной ситуации, который определяет дальнейшие действия.

В PL/SQL используются следующие типы исключительных ситуаций:

* встроенные исключительные ситуации;
* исключительные ситуации, определяемые пользователем;
* обработчик OTHERS.

**Встроенные исключительные ситуации**

Oracle включает четырнадцать встроенных исключительных ситуаций, соответствующих типовым ошибкам, приведенным в следующей таблице:

Исключительная ситуация ORACLE Описание

---------------------- --------- ---------------------------------------------

CURSOR\_ALREADY\_OPEN ORA-06511 Попытка открытия уже открытого курсора

DUP\_VAL\_ON\_INDEX ORA-00001 Попытка вставить дубликат значения для

уникального индекса

INVALID\_CURSOR ORA-01001 Попытка выполнения запрещенной операции с кур-

сором (например, закрытие неоткрытого курсора)

INVALID\_NUMBER ORA-01722 Отказ преобразования строки символов в число

LOGIN\_DENIED ORA-01017 Неправильное имя пользователь/пароль

NO\_DATA\_FOUND ORA-01403 Предложение SELECT...INTO возвращает ноль строк

NOT\_LOGGED\_ON ORA-01012 Нет подключения к Oracle7

PROGRAM\_ERROR ORA-06501 Внутренняя ошибка PL/SQL

STORAGE\_ERROR ORA-06500 Пакет PL/SQL вышел из пределов памяти или если

память разрушена

TIMEOUT\_ON\_RESOURCE ORA-00051 Истекло время ожидания ресурса Oracle7

TOO\_MANY\_ROWS ORA-01422 Предложение SELECT...INTO возвращает более

одной строки

TRANSACTION\_BACKED\_OUT ORA-00061 Удаленный сервер отменил транзакцию

VALUE\_ERROR ORA-06502 Арифметическая ошибка, ошибка преобразования,

усечения или ограничения

ZERO\_DIVIDE ORA-01476 Попытка деления на ноль

Если в раздел EXCEPTION программы (блока) включена фраза

WHEN имя\_исключения THEN  
текст\_обработчика\_исключения;

с именем какого-либо встроенного исключения и возникла соответствующая ошибка, то вместо прекращения исполнения программы и выдачи типового сообщения об ошибке, будет исполняться созданный пользователем текст обработчика исключения.

Такой обработчик может, например, выяснить ситуацию, при которой произошло **деление на ноль**, и выдать правдоподобный результат операции деления или прервать исполнение программы и дать сообщение об изменении каких-либо данных.  
В последнем случае это может быть не типовое сообщение «Вы пытаетесь делить на ноль», а любое подготовленное пользователем сообщение, например, инструкцию длиной до 2048 символов.

Для выдачи сообщения об ошибке, обеспечения возврата в среду, из которой вызывалась текущая программа (блок) и отмены всех действий, выполненных в текущей транзакции, целесообразно использовать процедуру **RAISE\_APPLICATION\_ERROR(errnum,errtext);** где **errnum** – отрицательное целое число в диапазоне -20000 .. -20999 и **errtext** – символьная строка длиной до 2048 символов.

В приведенном ниже триггере «shtins» использованы два типа встроенных исключительных ситуаций: NO\_DATA\_FOUND и TOO\_MANY\_ROWS.

**DROP** **TRIGGER** shtins;

**CREATE** **TRIGGER** shtins

BEFORE **INSERT** **ON** SHTAT

**FOR** EACH ROW

DECLARE

nach DATE;

kon DATE;

str NUMBER;

minraz NUMBER;

maxraz NUMBER;

nach\_kon EXCEPTION;

err\_str EXCEPTION;

nach\_nach EXCEPTION;

err\_razr EXCEPTION;

err\_razr\_pr EXCEPTION;

err\_stavka EXCEPTION;

BEGIN

**SELECT** min\_razr,max\_razr **INTO** minraz,maxraz **FROM** dolgnosti

**WHERE** dolgn = :new.dolgn;

**IF** :new.razr **NOT** **BETWEEN** minraz **AND** maxraz THEN RAISE err\_razr; END **IF**;

**IF** :new.razr\_proc **NOT** **BETWEEN** 50 **AND** 100 THEN RAISE err\_razr\_pr; END **IF**;

**IF** :new.stavka **NOT** **BETWEEN** 0.25 **AND** 100 THEN RAISE err\_stavka; END **IF**;

**IF** :new.nachalo > :new.konec THEN RAISE nach\_kon; END **IF**;

**SELECT** MAX(stroka) **INTO** str **FROM** shtat;

**IF** :new.stroka <> str+1 THEN RAISE err\_str; END **IF**;

<> *-- метка блока, в котором производится поиск строк с*

*-- параметрами, аналогичными вводимым значениям*

BEGIN

**SELECT** nachalo,konec **INTO** nach,kon **FROM** shtat

**WHERE** OTDEL = :new.otdel **AND** DOLGN = :new.dolgn **AND** RAZR = :new.razr

**AND** RAZR\_PROC = :new.razr\_proc **AND** KONEC =

(**SELECT** MAX(konec) **FROM** shtat

**WHERE** OTDEL = :new.otdel **AND** DOLGN = :new.dolgn

**AND** RAZR = :new.razr **AND** RAZR\_PROC = :new.razr\_proc);

**IF** :new.nachalo <= nach THEN RAISE nach\_nach; END **IF**;

**IF** :new.nachalo <= kon THEN

**UPDATE** SHTAT **SET** konec = (:new.nachalo - 1)

**WHERE** OTDEL =:new.otdel **AND** DOLGN = :new.dolgn **AND** RAZR =:new.razr

**AND** RAZR\_PROC = :new.RAZR\_PROC **AND** konec = kon;

END **IF**;

EXCEPTION *-- начало обработчика исключений блока find\_strings*

WHEN NO\_DATA\_FOUND THEN **NULL**; *-- вызывается, если SELECT блока find\_strings*

*-- не возвращает ни одной строки.*

END find\_strings;

EXCEPTION *-- начало обработчика исключений основной программы*

WHEN NO\_DATA\_FOUND THEN

RAISE\_APPLICATION\_ERROR(-20003,'Должности '||:new.dolgn||' не существует !');

WHEN err\_razr THEN

RAISE\_APPLICATION\_ERROR(-20004,'Значение разряда не попадает в "вилку" разрядов');

WHEN err\_razr\_pr THEN

RAISE\_APPLICATION\_ERROR(-20005,'Разрядный процент должен находиться в пределах 50-100');

WHEN err\_stavka THEN

RAISE\_APPLICATION\_ERROR(-20006,'Число ставок должно находиться в пределах 0.25-100');

WHEN nach\_nach THEN

RAISE\_APPLICATION\_ERROR(-20007,'Дата начала должна быть больше '||to\_char(nach));

WHEN TOO\_MANY\_ROWS THEN

RAISE\_APPLICATION\_ERROR(-20008,'Много строк; обратитесь к АБД.');

WHEN nach\_kon THEN

RAISE\_APPLICATION\_ERROR(-20009,'Дата начала не может быть больше даты конца');

WHEN err\_str THEN

RAISE\_APPLICATION\_ERROR(-20010,'Неправильный номер вводимой строки');

WHEN OTHERS THEN

RAISE\_APPLICATION\_ERROR(-20011,'Какая-то другая ошибка');

END shtins;

Так как в большом приложении могут часто повторяться встроенные или пользовательские исключительные ситуации, то целесообразно создать в базе данных таблицу (например, USERERR) с уникальными номерами (error\_number) и текстами (error\_text) исключений. Это позволит избежать определения лишних сообщений об ошибках и сделать их согласованными во всем приложении.

При использовании такой таблицы и процедуры RAISE\_APPLICATION\_ERROR надо описать в в разделе DECLARE блока две переменных (например, errnum типа NUMBER и errtext типа VARCHAR2) и использовать в обработчике исключений конструкцию:

WHEN TOO\_MANY\_ROWS THEN

SELECT error\_number,error\_text INTO errnum,errtext FROM usererr

WHERE error\_number = 20008;

RAISE\_APPLICATION\_ERROR(errnum,errtext);

или

WHEN TOO\_MANY\_ROWS THEN

SELECT error\_number,error\_text INTO errnum,errtext FROM usererr

WHERE errtext LIKE 'Много строк; обр%';

RAISE\_APPLICATION\_ERROR(errnum,errtext);

**Исключительные ситуации, определяемые пользователем**

Кроме встроенных могут быть использованы собственные исключительные ситуации, имена которых необходимо описать в разделе DECLARE блока PL/SQL (например, err\_stavka EXCEPTION). В разделе EXCEPTION блока должен быть описан соответствующий обработчик исключительной ситуации, например

WHEN err\_stavka THEN

SELECT error\_number,error\_text INTO errnum,errtext FROM usererr

WHERE errtext LIKE 'Число ставок должно находиться%';

RAISE\_APPLICATION\_ERROR(errnum,errtext);

В теле основной программы определяемые пользователем ошибки обычно проверяются с помощью операторов условия (IF…THEN). Для передачи управления обработчику пользовательской исключительной ситуации в случае обнаружения ошибки используется оператор **RAISE имя\_пользовательского\_исключения** Например

IF :new.stavka NOT BETWEEN 0.25 AND 100 THEN

RAISE err\_stavka;

END IF;

**Обработчик OTHERS**

Если исключительная ситуация не обрабатывается явным образом в блоке и для ее перехвата не используется обработчик OTHERS, то PL/SQL отменяет выполняемые блоком транзакции и возвращает необработанную исключительную ситуацию обратно в вызывающую среду.  
Обработчик особых ситуаций OTHERS описывается последним в программе (блоке) для перехвата всех исключительных ситуаций, которые не были описаны в этой программе (блоке). Он может иметь вид

WHEN OTHERS THEN

RAISE\_APPLICATION\_ERROR(-20011,'Какая-то другая ошибка');