Обработка ошибок. Исключительные ситуации и их обработка. Триггеры базы данных Отслеживание изменений в базе данных. Установка флагов на события. Объекты база данных Объектно-ориентированных подход для работы с базой данных. Обработка больших объектов. Планирование задач. Планировщик задач. Взаимодействие с операционной системой. Распределенные базы данных.

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 курсором (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');
```

Задание:

Вам представлен следующий код PL/SQL:

```
SET SERVEROUTPUT ON;

DECLARE

nNum1 number(10,0);

nNum2 number(10,0);

nRezult Number(10,5);

Begin

nNum1 := &Делимое;

nNum2 := &Делитель;

nRezult := nNum1/nNum2;

DBMS_OUTPUT.PUT_LINE(nRezult);

END;
```

- если пользователь выбирает для второго числа 0, выводилось сообщение "Деление на ноль!";
- если пользователь вводит символьные значения типа 'abc', выводилось сообщение "Неверный тип данных";
- в других ситуациях выводилось сообщение "Неизвестная ошибка".

Результат выполнения кода должен быть таким, как представлено на рис. Лаб. 18.1-1.

Примечание. Некоторые исключения все равно перехватываться не будут. Убедитесь, что ваш код перехватывает исключения для 0 и 'abc'.

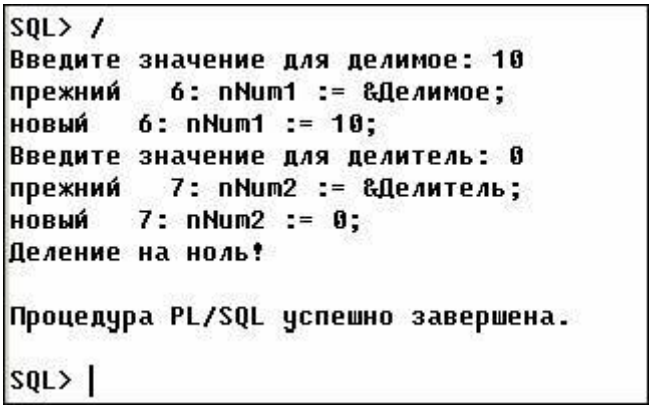

Рис. Лаб. 18.1-1.

Решение:

Итоговый код PL/SQL может быть таким:

```
DECLARE

nNum1 number(10,0);

nNum2 number(10,0);

nRezult Number(10,5);

Begin

nNum1 := &Делимое;

nNum2 := &Делитель;

nRezult := nNum1/nNum2;

DBMS_OUTPUT.PUT_LINE(nRezult);

EXCEPTION

WHEN ZERO_DIVIDE THEN

DBMS_OUTPUT.PUT_LINE('Деление на ноль!');

WHEN VALUE_ERROR THEN

DBMS_OUTPUT.PUT_LINE('Неверный тип данных!');
```

```
WHEN OTHERS THEN

DBMS_OUTPUT.PUT_LINE('Неизвестная ошибка!');

END;
```