##

ЛАБОРАТОРНАЯ РАБОТА N1

ТЕМА: Среда разработки Oracle APEX. Язык PL/SQL.
Обработка ошибок

Используемые объекты:
1. Учебная база данных - таблицы SAL, CUST, ORD
2. Описание учебной базы данных - таблица OPDB

Среда разработки размещена на сайле https://apex.oracle.com Для выполнения лабораторных работ необходимо создать на этом сайте свою рабочую область (WorkSpace). Инструкция по созданию рабочей области содержится в файле «Среда разработки Oracle APEX» на ЭИОС в разделе «Практические задания».

Лабораторные работы выполняются с использованием учебной базы данных (БД). Инструкция по созданию учебной базы данных содержится в файле «Создание учебной БД в Oracle APEX» на ЭИОС в разделе «Практические задания».

ЗАДАНИЕ 1.

Создайте в своей рабочей области таблицы учебной базы данных (Sal, Cust и Ord) с первичными и внешними ключами. Загрузите данные в таблицы. Для создания и заполнения таблиц используйте сценарии, созданные на основе файлов из папки «Учебная БД» раздела «Практические задания» на ЭИОС.

Для просмотра данных в таблицах используйте в APEX пункт SQL Workshop горизонтального меню и в выпадающем меню выберите подпункт Object Browser (или щелкните мышкой по пиктограмме Object Browser). Откроется окно объектов БД. В верхней части в колонке слева выберите в выпадающем списке объектов – Tables. Появится список таблиц БД. Для просмотра сведений о таблице щелкните мышкой по ее названию, и в правой части окна появится набор вкладок для просмотра характеристик и содержимого таблицы. При выборе вкладки Data будет показан набор строк, хранящихся в таблице.

Для изменения данных в строке щелкните по пиктограмме в левой колонке EDIT. Откроется форма с полями строки, содержащими данные. После изменения данных можно сохранить новые значения по кнопке Apply Changes или отказаться от изменений кнопкой Cancel.

PL/SQL - это язык программирования (PL – programmatic language) Oracle, представляющий собой расширение языка структурированных запрозов (SQL). Язык PL/SQL позволяет писать программы с использованием:
- переменных,
- блоков IF ... THEN,
- циклов FOR ... NEXT,
- циклов WHILE ... DO,

- блоков обработки ошибок
и других возможностей.

Одна из областей применения языка PL/SQL - это файлы сценариев SQL. При этом используется следующий синтаксис программы в сценарии:

```
DECLARE
  ...
  (объявления переменных)
  ...
BEGIN
  ...
  (команды SQL и PL/SQL)
EXCEPTION
  ...
  (обработчики исключение)
END;
```

Представленная выше синтаксическая конструкция называется блоком PL/SQL. Каждый блок имеет три раздела. Первый раздел (DECLARE) служит для определения переменных.

Следующий раздел начинается с оператора BEGIN и содержит исполняемый код блока (поэтому он является основным). Этот раздел может содержать команды SQL, PL/SQL, а также комментарии.

Третий раздел блока PL/SQL начинается с оператора EXCEPTION и является обработчиком исключений (ошибочных ситуаций).

Обязательным является только второй раздел.

Блок завершается оператором END, после которого должна стоять точка с запятой. Для запуска блока на выполнение нужно в отдельной строке после оператора END поместить наклонную черту (/) в крайнюю левую позицию.

Для того, чтобы PL/SQL-программа получила доступ к данным, хранящимся в таблицах БД, используются переменные. Переменные определяются в первом разделе блока - разделе DECLARE. Определение переменной состоит в назначении ей имени, за которым следует название типа данных. Наиболее часто применяются следующие типы данных:

CHAR - хранит строку символов ASCII фиксированной длины
VARCHAR2 - хранит строку символов переменной длины
NUMBER - хранит числовые данные, как целые, так и с плавающей точкой
DATE - тип данных даты/времени.

Для переменных типа CHAR, VARCHAR2 нужно, а типа NUMBER можно указать информацию о размере. Размер переменной указывается в круглых скобках после названия ее типа. Для символьных переменных указывается одно число, а для числовых - можно указать два числа: первое показывает общее количество цифр, второе - количество цифр после запятой.

Например: my_var NUMBER(4,2);

В языке PL/SQL для манипуляций с переменными используются:

1) оператор присваивания (:=). Например, для присвоения символьной переменной name конкретного значения можно написать

```
name := 'Jhon';
```

2) арифметические операторы сложения (+), вычитания (-), умножения (*), деления (/) и другие. Основной оператор работы с текстом - конкатенация (||). Например:

```
A := X*6 + 4;
Fio := name || ' Smith ';
```

3) встроенные функции, такие, как:

RTRIM(char) - удаляет в символьной строке все пробелы справа

LTRIM(char) - удаляет в строке все пробелы слева

UPPER(char) - переводит символы строки в верхний регистр.

Например, следующий оператор удаляет пробелы справа в переменной Fio и переводит ее содержимое в верхний регистр:

```
Fio := UPPER(RTRIM(Fio));
```

ЗАДАНИЕ 2.

Используя текстовый редактор набейте (или скопируйте) текст сценария:

```
DROP TABLE n_sal;
CREATE TABLE n_sal (text VARCHAR2 (20),
          cnt  VARCHAR2 (20));
DECLARE
   town VARCHAR2(20);
   count_sal VARCHAR2(20);
BEGIN
   town := 'London';
   SELECT count(*) INTO count_sal
      FROM sal WHERE city = town;
   IF count_sal > 0 THEN
      INSERT INTO n_sal
         VALUES ('In '||town,count_sal);
      COMMIT;
   END IF;
END;
/
   SELECT * FROM n_sal;
```

Измените программу так, чтобы она выводила на экран данные для города San Jose, а затем замените город на Paris.
При этом используйте ключевое слово ELSE, чтобы в таблице n_sal заносилась фраза `No data`, если нет продавцов в данном городе.
Создайте сценарий (скрипт), запустите его на выполнение. Для создания сценария на странице SQL Scripts используйте кнопку Create. После запуска скрипта (кнопкой Run) в окне Results установите переключатель в положение View и нажмите кнопку Go – отобразятся результаты выполнения скрипта.

Для организации циклов в программах на PL/SQL используются конструкции:
LOOP ... END LOOP,
WHILE ... LOOP ... END LOOP,
FOR ... LOOP ... END LOOP.

В конструкции типа LOOP ... для выхода из цикла используется оператор EXIT внутри LOOP, например:

```
BEGIN
  counter := 0;
  LOOP
   INSERT INTO n_sal VALUES (2*counter);
   counter := counter + 1;
   IF counter >= 4 THEN
      EXIT;
   END IF;
  END LOOP;
END;
```

В конструкции типа WHILE ... условие выполнения цикла указывается после ключевого слова WHILE и проверяется перед каждым проходом. Цикл выполняется пока условие истинно, например:

```
BEGIN
  counter := 0;
  WHILE counter < 8 LOOP
   INSERT INTO n_sal VALUES (2*counter);
   counter := counter + 1;
  END LOOP;
END;
```

Конструкция FOR ... используется, когда нужно повторить какой-то набор команд фиксированное число раз. Например:

```
BEGIN
  counter := 0;
  FOR counter IN 1..8 LOOP
   INSERT INTO n_sal VALUES (2*counter);
  END LOOP;
END;
```

Если необходимо вывести на экран результаты работы операторов, размещенных в блоке, то можно использовать служебный пакет

```
DBMS_OUTPUT.PUT_LINE()
```

Так, например, для распечатки количества продавцов из сценария Задания 2 можно поместить в блок такую строку:

```
DBMS_OUTPUT.PUT_LINE('Count of Sals is '||count_sal);
```

ЗАДАНИЕ 3.

Напишите сценарий, который выбирает максимальную дату заказов из таблицы Ord и выводит ее при помощи процедуры DBMS_OUTPUT.PUT_LINE().

EXCEPTION - это раздел блока PL/SQL, который позволяет обрабатывать ошибки, возникающие в программе. В этом разделе можно поместить команды SQL и PL/SQL, которые будут автоматически выполняться при возникновении ошибочных ситуаций, таких, например, как отсутствие запрашиваемых данных. Приведенный ниже фрагмент кода иллюстрирует этот случай:

```
DECLARE
  last_name varchar2(20);
  ...
BEGIN
  SELECT sname INTO last_name FROM sal
   WHERE snum = 5000;
  ...
EXCEPTION
  when NO_DATA_FOUND then
   status := ‘Данные не найдены’;
   return_code := 5;
   DBMS_OUTPUT.PUT_LINE(‘code=’ return_code||’, ‘||status);
  ...
END;
```

Здесь системная переменная NO_DATA_FOUND называется исключением для обработки отсутствия данных. Существуют и другие системные (внутренние, предопределенные) исключения, такие как TOO_MANY_ROWS - слишком много строк, ZERO_DIVIDE - деление на ноль и др., при помощи которых можно обрабатывать конкретные ошибки.

В PL/SQL имеется особое исключение OTHERS, которое можно использовать для распознавания любых возникающих проблем. Для этого в разделе EXCEPTION нужно указать обработку только исключения OTHERS.

ЗАДАНИЕ 4.

Напишите сценарий, в котором отсутствие данных обрабатывалось бы не с помощью исключения NO_DATA_FOUND, а с использованием OTHERS. Запустите сценарий на выполнение.

В PL/SQL можно также ввести свои собственные исключения. Для этого сначала нужно определить исключение в разделе DECLARE. Для того, чтобы исключение сработало, его необходимо активизировать в нужном месте командой RAISE. Например, обработка исключения для слишком больших процентов комиссионных Продавцов может быть выполнена следующим образом:

```
DECLARE
  invalid_com exception;
  m_com NUMBER (4,3);
  ...
BEGIN
  SELECT MAX(comm) INTO m_com FROM sal;
  IF m_com > 0.25 THEN
   RAISE invalid_com;
  END IF;
  ...
EXCEPTION
  when invalid_com then
   status := 'Процент комиссионных слишком велик.';
   return_code := 10;
  ...
  when OTHERS then
   status := 'Ошибка';
   return_code := 99;
  ...
END;
```

ЗАДАНИЕ 5.

Напишите сценарий, в котором определите свое исключение для обработки ситуации, когда минимальный рейтинг Покупателя меньше 200. Предусмотрите обработку всех остальных ошибок. Выполните сценарий.

КОНТРОЛЬНЫЕ ВОПРОСЫ
==========

Контрольные вопросы выдает преподаватель после выполнения всех заданий.