Язык PL/SQL

1. Обработка ошибок

```
DECLARE
    (объявления переменных)
BEGIN
    (главная обрабатывающая часть программы)
EXCEPTION
    WHEN no_data_found THEN
        out_status := 'Data not …..';
        return_code := 5;
    WHEN too_many_rows THEN
        out_status := 'Query …..';
        return_code := 6;
END;
```

1. внутренние исключения,
2. исключения, предусмотренные пользователем.

- NO_DATA_FOUND
- STORAGE_ERROR
- TIMEOUT_ON_RESOURCE
- TOO_MANY_ROWS
- ZERO_DIVIDE

1.1. Преимущества исключений

```
  BEGIN
    SELECT ...
    -- проверить на ошибку 'no data found'
    SELECT ...
    -- проверить на ошибку 'no data found'
    …
  END;

BEGIN
    SELECT ...
    SELECT ...
    …
EXCEPTION
  WHEN NO_DATA_FOUND THEN -- этот обработчик перехватывает
                          -- все ошибки 'no data found'
END;
```

1.2. Предопределенные исключения

Имя исключения	Ошибка ORACLE	Код SQLCODE
CURSOR_ALREADY_OPEN	ORA-06511	-6511
DUP_VAL_ON_INDEX	ORA-00001	-1
INVALID_CURSOR	ORA-01001	-1001
INVALID_NUMBER	ORA-01722	-1722
LOGIN_DENIED	ORA-01017	-1017
NO_DATA_FOUND	ORA-01403	+100
NOT_LOGGED_ON	ORA-01012	-1012
PROGRAM_ERROR	ORA-06501	-6501
STORAGE_ERROR	ORA-06500	-6500
TIMEOUT_ON_RESOURCE	ORA-00051	-51
TOO_MANY_ROWS	ORA-01422	-1422
TRANSACTION_BACKED_OUT	ORA-00061	-61
VALUE_ERROR	ORA-06502	-6502
ZERO_DIVIDE	ORA-01476	-1476

OTHERS – обработка всех (остальных) ошибок

SQLCODE – код ошибки
SQLERRM – описание ошибки

EXCEPTION_INIT – присвоение имени исключению

```
DECLARE
   insufficient_privileges  EXCEPTION;
   PRAGMA EXCEPTION_INIT(insufficient_privileges, -1031);
      -------------------------------------------------------
      -- ORACLE возвращает код ошибки -1031, если, например,
      -- вы пытаетесь обновить таблицу, для которой имеете
      -- лишь полномочия SELECT.
      -------------------------------------------------------
BEGIN
…
EXCEPTION
   WHEN insufficient_privileges THEN
      -- обработать ошибку
…
END;
```

1.3. Пользовательские исключения

1.3.1. Объявление исключений

```
DECLARE
        past_due EXCEPTION;
        acct_num NUMBER(5);
BEGIN
        ...
```

1.3.2. Правила сферы

```
DECLARE
      past_due EXCEPTION;
      acct_num NUMBER;
BEGIN
      DECLARE ---------------- начало подблока -------------------------
            past_due EXCEPTION; -- имеет преимущество в подблоке
            acct_num NUMBER;
      BEGIN
            IF ... THEN
                  RAISE past_due; -- это не обрабатывается
            END IF;
      END; ---------------- конец подблока -------------------------
EXCEPTION
      WHEN past_due THEN -- это не относится к исключению,
                              ... -- возбуждаемому в подблоке
END;
```

1.4. Использование raise_application_error

```
PROCEDURE raise_salary (emp_id NUMBER, increase REAL) IS
  current_salary  NUMBER;
BEGIN
  SELECT sal INTO current_salary FROM emp  WHERE empno = emp_id;
     IF current_salary IS NULL THEN
        raise_application_error(-20101, 'Salary is missing');
     ELSE
        UPDATE emp SET sal = sal + increase WHERE empno = emp_id;
     END IF;
END raise_salary;
```

1.5. Как возбуждаются исключения

```
DECLARE
    out_of_stock EXCEPTION;
    number_on_hand NUMBER(4);
BEGIN
    IF number_on_hand < 1 THEN
        RAISE out_of_stock;
    END IF;
EXCEPTION
    WHEN out_of_stock THEN
    -- обработать ошибку
END;

RAISE INVALID_NUMBER;

DECLARE
    acct_type NUMBER;
BEGIN
    IF acct_type NOT IN (1, 2, 3) THEN
        RAISE INVALID_NUMBER;
    END IF;
EXCEPTION
    WHEN INVALID_NUMBER THEN
    ROLLBACK;
END;
```

1.6. Как распространяются исключения

```
DECLARE
    …
BEGIN
    DECLARE ---------- начало подблока ----------
        past_due EXCEPTION;
    BEGIN
        IF ... THEN
            RAISE past_due;
        END IF;
    END; ---------- конец подблока ----------
EXCEPTION
    WHEN OTHERS THEN
        ROLLBACK;
END;
```

1.7. Использование SQLCODE и SQLERRM

User-Defined Exception
ORA-0000: normal, successful completion

SQLERRM(+100) –> ORA-01403: no data found

```
INSERT INTO errors VALUES (SQLCODE, SQLERRM); --  незаконно
```

```
DECLARE
    err_num NUMBER;
    err_msg CHAR(100);
BEGIN
        …
EXCEPTION
    WHEN OTHERS THEN
        err_num := SQLCODE;
        err_msg := SUBSTR(SQLERRM, 1, 100);
        INSERT INTO errors VALUES (err_num, err_msg);
END;
```

1.8. Необработанные исключения

обработчик OTHERS на самом верхнем уровне

1.9. Полезные приемы

1.9.1. Продолжение работы после возбуждения исключения

```
DECLARE
    pe_ratio NUMBER(3,1);
BEGIN
    DELETE FROM stats WHERE symbol = 'XYZ';
    BEGIN -- начало подблока ----
      SELECT price / NVL(earnings,0) INTO pe_ratio FROM stocks
        WHERE symbol = 'XYZ';
    EXCEPTION
      WHEN ZERO_DIVIDE THEN
        pe_ratio := 0;
    END;  ---------------- конец подблока --------------------
    INSERT INTO stats (symbol, ratio) VALUES ('XYZ', pe_ratio);
EXCEPTION
        …
END;
```

1.9.2. Повторение транзакции

```
DECLARE
    name CHAR(20);
    ans1 CHAR(3);
    ans2 CHAR(3);
    ans3 CHAR(3);
    suffix NUMBER := 1;
BEGIN
…
    LOOP -- можно написать "FOR i IN 1..10 LOOP", чтобы
            -- ограничиться максимально десятью попытками
        BEGIN  ---------- начало подблока ----------
            …
            SAVEPOINT start_transaction; -- точка сохранения
             /* Удалить результаты опроса. */
            DELETE FROM results WHERE answer1 = 'NO';
             /* Добавить имя и ответы респондента. */
            INSERT INTO results VALUES (name, ans1, ans2, ans3);
             /* Это может дать исключение DUP_VAL_ON_INDEX, *
             * если два респондента имеют одинаковые имена,*
             * так как индекс по столбцу name уникальный. */
            COMMIT;
            EXIT;
        EXCEPTION
            WHEN DUP_VAL_ON_INDEX THEN
                ROLLBACK TO start_transaction; -- откат
                suffix := suffix + 1; -- попробуем исправить имя
                name := name || TO_CHAR(suffix);
                …
        END;  ---------- конец подблока ----------
    END LOOP;
END;
```