

PL/SQL

PL/SQL ? What is it ? Why do we need it ?

PL/SQL is a procedural extension of SQL

PL/SQL = procedural **P**rogramming
Language+ **S**QL

We need PL/SQL to bridge a gap between
high level declarative query language and
procedural programming language

Overview

PL/SQL =

Data Manipulation statements of SQL +

SELECT statement +

variables +

assignment statement +

conditional control statements +

repetition statement +

exception handling +

procedure and function statements +

packages

Program structure

PL/SQL is a block-structured language

It means that its basic units such as anonymous blocks, procedures, and functions are the logical blocks

Logical blocks can be nested to any level

Logical blocks consist of declarative, executable, and exception components

Declarative components

Declarative components contain declarations of variables, constants, cursors, procedures, and functions

DECLARE

stock_num NUMBER(5);

stock_name VARCHAR(30);

stock_date DATE;

limit CONSTANT NUMBER(11,2) := 2.45;

CURSOR Q IS

SELECT s# FROM Student WHERE name = 'Jo';

Executable components

Declarative components assignment statements, conditional control statements, iterative statements, procedure and function calls, SQL statements

student_name VARCHAR(40);

BEGIN

student_num := 910000;

SELECT name INTO student_name

FROM Student

WHERE s# = student_num;

IF a > b THEN a:= a+1 ELSE b:= b+1 END IF;

25. PL/SQL

Exception components

Exception component consists of executable statements that service the exceptional situations during execution

```
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    INSERT INTO Audit VALUES( SYSDATE, s# );
  WHEN OTHERS
    I:= I+1;
    UPDATE Department SET budget = I*budget;
END;
```

© Janusz R. Getta

CSCI235/MCS9235 Databases, SCSSE, Spring 2009

7

25. PL/SQL

Structure of anonymous block

```
DECLARE
  -- declarations
BEGIN
  -- executable statements
EXCEPTION
  -- exception handlers
END;
```

© Janusz R. Getta

CSCI235/MCS9235 Databases, SCSSE, Spring 2009

8

25. PL/SQL

Structure of procedure

```
PROCEDURE procedure_name ( parameters )
  -- declarations
BEGIN
  -- executable statements
EXCEPTION
  -- exception handlers
END procedure_name;
```

© Janusz R. Getta

CSCI235/MCS9235 Databases, SCSSE, Spring 2009

9

25. PL/SQL

Structure of function

```
FUNCTION function_name ( parameters )
  RETURN type-specification IS
  -- declarations
BEGIN
  -- executable statements
EXCEPTION
  -- exception handlers
END function_name;
```

© Janusz R. Getta

CSCI235/MCS9235 Databases, SCSSE, Spring 2009

10

25. PL/SQL

Example of anonymous block

```
DECLARE
  average NUMBER(8,2);
BEGIN
  SELECT avg(budget)
  INTO average
  FROM Department;
  IF average < 3000 THEN
    UPDATE Department
    SET budget = budget+100;
  END IF
END;
```

© Janusz R. Getta

CSCI235/MCS9235 Databases, SCSSE, Spring 2009

11

25. PL/SQL

Data types

```
BINARY,
INTEGER,
NUMBER,
CHAR,
DATE,
VARCHAR,
LONG,
RAW,
LONGRAW,
BOOLEAN,
ROWID,
EXCEPTION
```

© Janusz R. Getta

CSCI235/MCS9235 Databases, SCSSE, Spring 2009

12

25. PL/SQL

Implicit type declarations (%TYPE)

```

DECLARE
    student_no      Student.s#%TYPE;
    student_name    Student.name%TYPE;

BEGIN
    student_no := 1234567;

    SELECT name
    INTO student_name
    FROM Student
    WHERE s# = student_no;

```

© Janusz R. Getta CSC235/MCS9235 Databases, SCSSE, Spring 2009 13

25. PL/SQL

Implicit type declarations (%ROWTYPE)

```

DECLARE
    student_row Student%ROWTYPE;

BEGIN
    student_row.s# := 1234567;
    student_row.name := 'James';

    student_row.dob :=
        TO_DATE('01-DEC-1994', 'DD-MON-YYYY');

    INSERT INTO Student VALUES (
        student_row.s#, student_row.name,
        student_row.dob);

```

© Janusz R. Getta CSC235/MCS9235 Databases, SCSSE, Spring 2009 14

25. PL/SQL

Logical comparisons and Boolean operations

```

<, >, >=, <=, =

AND, OR, NOT

```

© Janusz R. Getta CSC235/MCS9235 Databases, SCSSE, Spring 2009 15

25. PL/SQL

Conditional control statements

```

IF condition THEN

    statement;

    statement;

    ...

ELSE

    statement;

    statement;

    ...

END IF;

```

© Janusz R. Getta CSC235/MCS9235 Databases, SCSSE, Spring 2009 16

25. PL/SQL

Conditional control statements

```

IF condition THEN

    ...

ELSIF condition THEN

    ...

ELSIF condition THEN

    ...

ELSE

    ...

END IF;

```

© Janusz R. Getta CSC235/MCS9235 Databases, SCSSE, Spring 2009 17

25. PL/SQL

Sample database

```

CREATE TABLE Department (
    name      VARCHAR2(50),
    code      CHAR(5),
    total_staff_number NUMBER(2) NOT NULL,
    chair     VARCHAR(50),
    budget    NUMBER(9,1) NULL,
    CONSTRAINT dept_pkey PRIMARY KEY(name),
    CONSTRAINT dept_ckey1 UNIQUE(code),
    CONSTRAINT dept_ckey2 UNIQUE(chairperson),
    CONSTRAINT dept_check1
    CHECK (total_staff_number BETWEEN 1 AND 50) );

CREATE TABLE Course (
    c#        CHAR(7),
    title     VARCHAR2(200) NOT NULL,
    credits   NUMBER(1) NOT NULL,
    offered_by VARCHAR(50) NULL,
    CONSTRAINT course_pkey PRIMARY KEY(c#),
    CONSTRAINT course_check1
    CHECK (credits IN (6, 12) ),
    CONSTRAINT course_fkey1 FOREIGN KEY(offered_by)
    REFERENCES Department(name) );

```

© Janusz R. Getta CSC235/MCS9235 Databases, SCSSE, Spring 2009 18

25. PL/SQL

Anonymous block

```

DECLARE
    current_budget Department.budget%TYPE;
    budget_limit NUMBER(6) := 400000;
BEGIN
    SELECT budget INTO current_budget
    FROM Department WHERE name = 'Math';
    IF current_budget < budget_limit THEN
        UPDATE Department SET budget := budget_limit
        WHERE name = 'Math';
    ELSE
        INSERT INTO Audit VALUES( 'Math budget OK', current_budget);
    END IF;
    COMMIT;
END;

```

© Janusz R. Getta

CSCI235/MCS9235 Databases, SCSSE, Spring 2009

19

25. PL/SQL

Iterative control statements

```

LOOP
    ...
    IF condition THEN EXIT;
    ...
END IF;
...
END LOOP;

```

© Janusz R. Getta

CSCI235/MCS9235 Databases, SCSSE, Spring 2009

20

25. PL/SQL

Iterative control statements

```

FOR variable IN scope
LOOP
    ...
END LOOP;

FOR variable IN REVERSE scope
LOOP
    ...
END LOOP;

```

© Janusz R. Getta

CSCI235/MCS9235 Databases, SCSSE, Spring 2009

21

25. PL/SQL

Iterative control statements

```

WHILE (condition)
LOOP
    ...
END LOOP;

```

© Janusz R. Getta

CSCI235/MCS9235 Databases, SCSSE, Spring 2009

22

25. PL/SQL

Procedures

```

PROCEDURE raise_budget (
    department_name IN VARCHAR,
    budget_limit IN NUMBER ) IS
    current_budget Department.budget%TYPE;
BEGIN
    SELECT budget INTO current_budget
    FROM Department WHERE name = department_name;
    IF current_budget < budget_limit THEN
        UPDATE Department SET budget := budget_limit
        WHERE name = department_name;
    ELSE
        INSERT INTO Audit VALUES( 'Math budget OK', current_budget);
    END IF; COMMIT;
END raise_budget;

```

© Janusz R. Getta

CSCI235/MCS9235 Databases, SCSSE, Spring 2009

23

25. PL/SQL

Functions

```

FUNCTION raise_budget (
    department_name IN VARCHAR,
    budget_limit IN NUMBER ) RETURN NUMBER IS
    current_budget Department.budget%TYPE;
BEGIN
    SELECT budget INTO current_budget
    FROM Department WHERE name = department_name;
    IF current_budget < budget_limit THEN
        UPDATE Department SET budget := budget_limit
        WHERE name = department_name; RETURN budget_limit;
    ELSE
        INSERT INTO Audit VALUES( 'Math budget OK', current_budget);
    RETURN current_budget; END IF; COMMIT;
END raise_budget;

```

© Janusz R. Getta

CSCI235/MCS9235 Databases, SCSSE, Spring 2009

24

25. PL/SQL

Cursors

```

DECLARE
    student_no STUDENT.s#%TYPE;
BEGIN
    SELECT s#
    INTO student_no
    FROM Student
    WHERE name = 'Pam';
    ...

```

ERROR at line 1:
ORA-06503: PL/SQL: error 0 - Unhandled exception ORA-01427: single-row subquery returns more than one row which was raised in a statement ending at line 6

© Janusz R. Getta CSC1235/MCS9235 Databases, SCSSE, Spring 2009 25

25. PL/SQL

Cursors

Cursor is a syntactical construction that allows for processing of relational tables in a "row-by-row" mode

Declaration of a cursor

```

DECLARE
    CURSOR Q IS
        SELECT s#
        FROM Student
        WHERE name = 'Pam';

```

© Janusz R. Getta CSC1235/MCS9235 Databases, SCSSE, Spring 2009 26

25. PL/SQL

Cursors

Cursors name

```

DECLARE
    CURSOR Q IS
        SELECT s#
        FROM Student
        WHERE name = 'Pam';

```

Cursors body

```

DECLARE
    CURSOR Q IS
        SELECT s#
        FROM Student
        WHERE name = 'Pam';

```

© Janusz R. Getta CSC1235/MCS9235 Databases, SCSSE, Spring 2009 27

25. PL/SQL

Explicit cursor processing

```

DECLARE
    student_no STUDENT.s#%TYPE;
    CURSOR Q IS SELECT s# FROM Student WHERE name = 'Pam';
BEGIN
    OPEN Q;
    LOOP
        FETCH Q INTO student_no;
        IF Q%NOTFOUND THEN EXIT; END IF;
        INSERT INTO Pam VALUES(student_no);
    END LOOP;
    CLOSE Q;
    COMMIT;
END;

```

© Janusz R. Getta CSC1235/MCS9235 Databases, SCSSE, Spring 2009 28

25. PL/SQL

Implicit cursor processing

```

BEGIN
    FOR Q_row IN (SELECT s# FROM Student WHERE name = 'Pam')
    LOOP
        INSERT INTO Pam VALUES(Q_row.s#);
    END LOOP;
    COMMIT;
END;

```

© Janusz R. Getta CSC1235/MCS9235 Databases, SCSSE, Spring 2009 29

25. PL/SQL

Cursor attributes

%NOTFOUND

Evaluates to true if the last **FETCH** failed because no more rows were available

%FOUND

Evaluates to true if the last **FETCH** succeeded

%ROWCOUNT

Returns the total number of rows **FETCHED** so far

%ISOPEN

Evaluates to true if the cursor is opened

© Janusz R. Getta CSC1235/MCS9235 Databases, SCSSE, Spring 2009 30

25. PL/SQL

Cursor attributes

```

DECLARE
student_no STUDENT.s#%TYPE;
CURSOR Q IS SELECT s# FROM Student WHERE name = 'Pam';
BEGIN
    OPEN Q;
    LOOP
        FETCH Q INTO student_no;
        IF Q%NOTFOUND THEN EXIT END IF;
        INSERT INTO Pam VALUES(student_no);
    END LOOP;

    IF Q%ROWCOUNT = 0 THEN
        INSERT INTO Messages VALUES ('NO ROWS PROCESSED');
    END IF;

    CLOSE Q;
    COMMIT;
END;

```

25. PL/SQL

Exceptions

Exception is an internally defined or user defined error condition, e.g. divide by zero, no rows selected by **SELECT** statement with **INTO** clause, failure of **FETCH** statement, use of a cursor which has not been opened yet, etc.

25. PL/SQL

Exceptions

```

BEGIN
    error_number NUMBER(5);
    error_message VARCHAR(200);
    ...
EXCEPTION
    WHEN OTHERS THEN
        error_number := SQLCODE;
        error_message := SQLERRM;
        INSERT INTO Errors( error_number, error_message);
        COMMIT;
END;

```

25. PL/SQL

Exception

```

DECLARE
    student_name STUDENT.name%TYPE;
BEGIN
    SELECT name
    INTO student_name
    FROM Student
    WHERE s# = 1234567;
    ...
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        INSERT INTO Messages VALUES( 'Student not found');
        COMMIT;
END;

```

25. PL/SQL

Exceptions

NO_DATA_FOUND
 Raised when **SELECT** statement returns no rows

TOO_MANY_ROWS
 Raised when **SELECT** statement returns more than one row

INVALID_CURSOR
 Raised when PL/SQL call specifies an invalid cursor, e.g. closing an unopened cursor

OTHERS
 Raised when any other exception, not explicitly named happens

25. PL/SQL

References

<https://sai.uow.edu.au/oradocs/>
 PL/SQL User's Guide and Reference