PL/SQL Fundamentals

PL/SQL - Procedural language / Structured Query Language.

- extension of SQL.
- Super set of SQL.
- eliminates restrictions of SQL.

→ adds control structures

Diff.

	SQL	PL/SQL
(1)	No procedural capabilities (Checking conditions, loops, branching) essential for data filtration.	Present
(2)	Each statement, needs a call to oracle engine. ∴ slow execution	Block call. fast.
(3)	No Error & exception handling. Here error messages, with code. Not user friendly msgs.	Yes. Display user friendly appro - priate messages.
(4)	SQL does not support PL/SQL statements.	(4) PL/SQL support SQL block.
(5)	Intermediate Results can not be stored.	(5) Stored in a declared variable.

PL/SQL Fundamentals

PL/SQL → Procedural language / Structured Query Language.

- extension of SQL.
- Superset of SQL.
- eliminates restrictions of SQL.

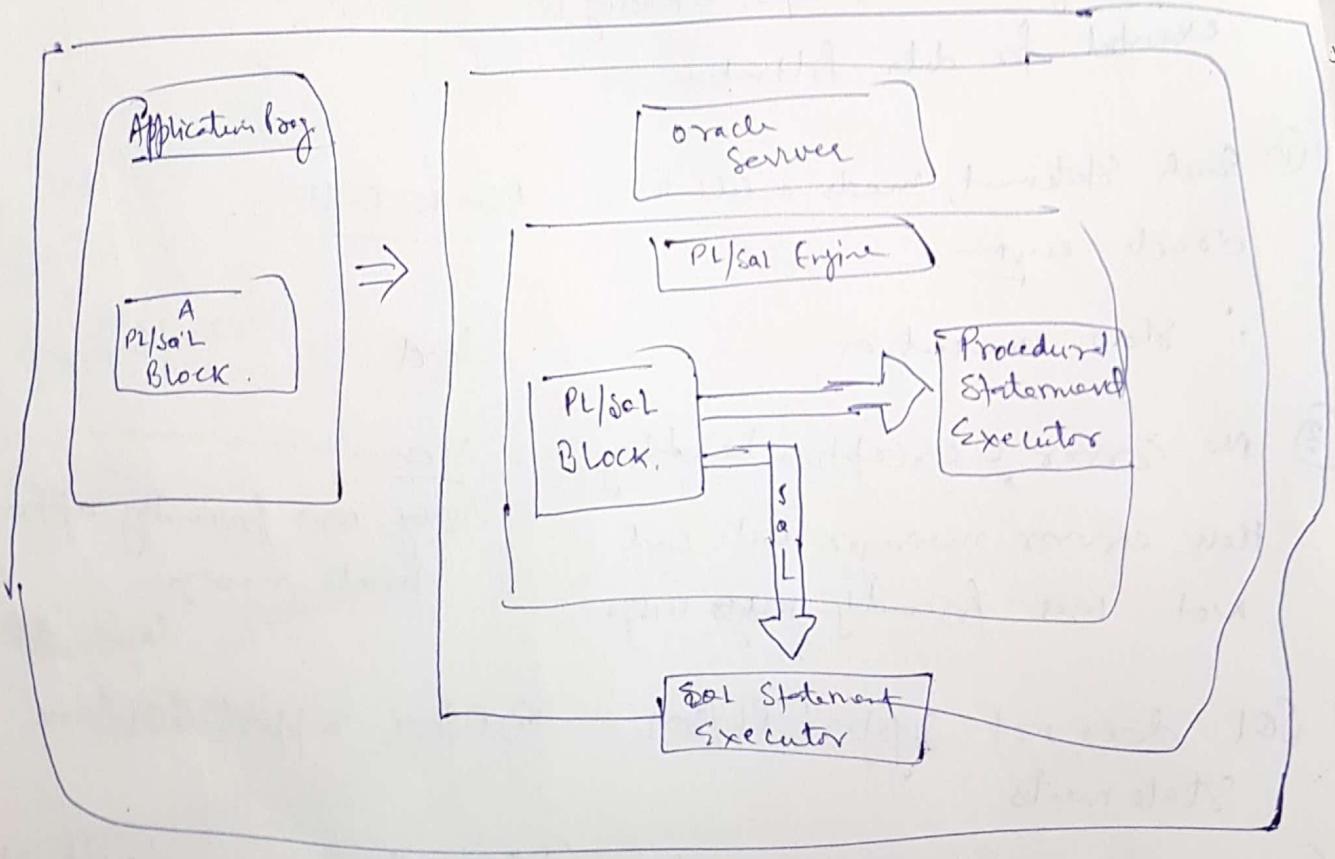
→ adds control structures

Diff:

	SQL	PL/SQL
(1)	No procedural capabilities (Checking conditions, loops, branching) essential for data filtration.	Present
(2)	Each statement, needs a call to oracle engine. ∴ Slow execution	Block call. Fast.
(3)	No Error & exception handling. Here error messages, with code. Not user friendly msgs.	Yes. Display user friendly appro - priate messages.
(4)	SQL does not support PL/SQL statements.	(2) PL/SQL support SQL block.
(5)	Intermediate Results can not be stored.	(1) Stored in a declared variable.

Advantage of PL/SQL

* Supports declaration & manipulation of obj. types
* External functions & procedures calls allowed
* Built in package library. (Package of fn, cursors, SP & ...
* Triggers: Prg stored in DB, executed immediately before/after INSERT, UPDATE & DELETE commands.
* Cursors: named workspace to execute SQL commands.
* Supports SQL: → SQL manipulation commands, TCL, SQL fns., operators



PL/SQL Architecture.

PL/SQL Fundamentals

Structure of PL/SQL Language

PL/SQL Block → Smallest piece of PL/SQL code which groups logically related declarations & statements. Declarations are local to the blocks & cease to exist when block completes

Block has 3 Sections.

Declare → used to declare var & const; optional, also used to declare type declarations, procedures, fns. which are local to module.

Begin → executable section containing code, compulsory

Exception → handles exceptions, predefined error handler & user defined exceptions are placed here, code executed only if error occurs, optional.

End;

Blocks can be nested in executable & exception handling parts of PL/SQL block but not in declarative part

PL/SQL Language Elements.

- operators, Indicators, & punctuations
- Identifiers, - Literals, - Commands, - Expressions &
- Data types & declarations.

Operators +, -, *, /, ** (Raise to the power). → Arithmetic

Expression ops → :=, ... Range (1...4), || (Concatenation).

LIKE, NULL, BETWEEN, IN, AND, OR, !=, etc.

Identifiers, named conventions used for var, const, oracle objects like tables, procedures, fns, packages, triggers, etc.

Literals - Exact value, 10, -2, 'a', 'abc', True
Decimal Number, char, string, Boolean.

Comments -- single line, /* */ multilines.

Expressions & Comparisons

Operator Precedence → from book.

Data Types → Number, Char, varchar, Date, Boolean, long, long Raw, Lob.

Eg: <var_name> Number [(precision, scale)]; for fixed pt.

Number for floating pts.

<variable_name> CHAR [(max_length)].

↳ integer in the range 1 - - 32,767.

→ 32760 bytes
→ store binary data or byte strings.

Ans → 32,760.

→ large objects. data types, BFILE
Text, graphic images, v
upto 4 gigabytes

Variables & Constant

variables declared in dec
body of PL/SQL block.

Assigning value to
- assignment op

by ex

...ng → 32760 bytes

LONG RAW → 32,760.

BLOB → Large objects · datatypes, BFILE, BLOB, CLOB, NCLOB.)
Text, graphic images, video clips · Sound waves,
upto 4 gigabytes.

Variables & Constants

Variables declared in declaration section, & use elsewhere in body of PL/SQL block. eg: age number(4);
Done boolean;

Assigning value to variables

- assignment operator. a := b * c;
ok := false;

by use of Substitute variables (whose values are entered at run time during execution).

a := &enter_number;
↳ substitute variable.

or a := &a.

- With Select INTO clause

eg: Select sal INTO $ From emp where empno = 100;
↳ declared variable.

ASSIGNMENT - 1

WAP to calculate total marks of student whose Roll no 100.
Table Stud. have following columns, Sub1, Sub2, Sub3.

Declare

S1 Number(3);
S2 ——
S3 ——
T —— (3);

Begin

Select Sub1, Sub2, Sub3 INTO S1, S2, S3 from Stud where Roll=

T = S1+S2+S3.

UPDATE Stud Set total = T where Roll = 100.

END

Constant Declaration → declaration section, use elsewhere.

Pi Constant number := 3.14;

Keyword.

Table Attributes → Allows to refer data types & objects from DB.

%Type → used when declaring variables that refer to DB column.

eg: salary emp.sal %.type;

if data type of column change then so the variable data type.

%Rowtype

my_employee employee %Rowtype;

User message on Screen

DBMS_OUTPUT.PUT_LINE ("Enter the no:");

Package that includes no. of procedures & functions that accumulate info. in a buffer so that it can be retrieved later.

DBMS_OUTPUT.PUT_LINE("Result =" || &num);

Before this SET SERVEROUTPUT {ON/OFF};

Scanned by CamScanner

Control Statements

① Conditional ② Iteration ③ Sequential ctrl.

↳ IF-Then , IF-THEN-ELSE ,
IF-THEN-ELSE IF (NESTED IF)

① If Condition Then
Statements.
ENDIF.

② If Condition Then
S1 ;
ELSE
S2 ;
END If .

If Condition1 THEN
S1 ;
ELSE IF Condition 2 Then
S2 .
ELSE
S3 ;
END If ;

ITERAT

① Simple

↳ B

L

L

Pgm for addition, sub, mult & div of 2 nos

```
declare
    a number := &A;   b number := &B;   c number; x number;
Begin
    x := &Enter choice;
If  x=1 THEN
    c := A+B;
ELSE IF  x=2 THEN
    c := A-B
ELSE IF  x=3 THEN
    c := A*B;
ELSEIF  x=4 THEN
    c := A/B
ELSE
    DBMS_OUTPUT.PUT_LINE("Invalid option");
END IF
DBMS_OUTPUT.PUT_LINE("Result is" ||c);
END.
```

ITERATIVE CONTROL.

① Simple Loop Statement ② While Loop ③ For Loop.

↳ Basic (or infinite) loop, which encloses statements b/w keywords LOOP and END LOOP as :-

```
LOOP
  Sequence of statements;
END LOOP;
```

If further processing is undesirable, use an EXIT statement.

Two forms

* EXIT * EXIT-WHEN ..

EXIT → forces loop to complete unconditionally. When an EXIT is encountered, loop completes immediately & control passes to the next statement.

```
LOOP
  If ____ THEN
    :
  EXIT;
END LOOP;
.... Control resume here.
```

EXIT-WHEN Statement → allows loop to complete conditionally

```
LOOP
  EXIT WHEN condition ;
  - - -
END LOOP.
```

i.e. EXIT WHEN replaces IF.

Prg → To print nos from 1 to N

```
Declare
  i Number := 1 ;
  n number := &Enter-N ;
Begin
  Loop
    DBMS_OUTPUT.PUT_LINE (i);
    i := i+1;
    EXIT WHEN i > n ;
  END Loop ;
END;
```

WHILE Loop

```
WHILE condition LOOP
    statements ;
END LOOP
```

May execute zero times

eg:-

```
WHILE i<=10 Loop
  a := n*i ;
  i := i+1 ;
END Loop
```

counter IN [REVERSE] lower_bound . higher_bound LOOP
    Statements;
END LOOP;

eg:- FOR i IN 1.3 LOOP
    Statements;
END LOOP;

* No stop option in a structure, but some languages have.
But you can easily build one.

eg:- FOR j IN 5.15 LOOP
    IF MOD(j,5)=0 THEN
        Statements;
    END IF;
END LOOP;

* Loop Counter is defined only within the loop, ~~not~~ not referenced outside loop.
(defined = scope.)

* Need not declare counter; implicitly INTEGER type.

Using EXIT → allows FOR loop to complete prematurely.

eg:- for i IN 1..10 LOOP
    b:=a*i;
    EXIT when b>9,99;
    - - -
END loop;

EXIT-WHEN Statement

Sequential ctrl statements → GOTO & NULL.

Goto → branches to a label Unconditionally. Label must be unique within its Scope & must precede an executable statement or a PL/SQL block

```
BEGIN
- - -
GOTO insert_row;
- - - - -
- - - - -
<<insert_row>>
INSERT INTO emp VALUES
END;
```

```
LOOP
= = =
IF A>B THEN
  Go to <<abc>>;
END IF
= = =
<<abc>>
END loop; -- illegal ; it does not
             precede the executable
             statement.
END;
```

NULL Statement → explicitly specifies inaction; it does nothing other than pass ctrl to next statement.

Use → It can improve read ability.

```
If A>B then
DD_ _ _ line(A);
ELSE
  NULL;
END if.
```

The Soln of illegal use of Goto is with NULL.

above eg:

```
<<abc>>
NULL;
```

SERVER

STORAGE TIER

Scanned by CamScanner