अध्याय 15

PL/SQL के आधार बिन्दु

PL/SQL का अर्थ प्रासिजरल लैंग्वेज के एक्सटेशन से है। इसके महत्वपूर्ण बिन्दु SQL पर अधारित है। इसका निर्माण ऑरेकल कम्पनी ने किया है। जो इसके एक्सटेशन और इनहान्समेन्ट के लिये भी जवाब देय है।

साधारण PL/SQL ब्लॉक :–

यह मुख्यतः SQL और PL/SQL स्टेटमेन्टस से मिलकर बनता है। PL/SQL प्रोग्राम में भी इनका उपयोग होता है।

PL/SQL के तीन भाग :–

घोषणा (Declaration) भाग (गैर जरूरी)

लागुकरण (Execution) भाग (जरूरी)

अपवाद संचालन (Exception Handling) (और त्रुटि) भाग (गैर जरूरी)

PL/SQL Declaration भाग :– खंड का यह भाग जो कि गैर जरूरी है DECLAREनाम के keywordजो कि आरक्षित है से शुरू होता है। इस खंड का उद्देश्य constants, records, variables ओर cursors को घोषित करना है। उपरोक्त सभी विचारार्थ शब्द (प्लेसहोल्डर) Execution भाग में ऑकड़ों का हेरफेर करने के लिये उपयोग में लिये जाते है। प्लेसहोल्डर (constants, variable और records) ऑकड़ों को अस्थायी रूप से संग्रहित करते है।कर्सर भी इस खंड में घोषित किए गए हैं।

Execution भाग

PL/SQL (block) का यह भाग आरक्षित keywordBEGIN और END से शुरू और अन्त होता है। Execution भाग किसी कार्य को सफलतापूर्वक पूर्ण करने के लिये वहाँ जरूरी भाग है जहाँ प्रोग्राम का तथ्य लिखा जाता है। प्रोग्रामेटिकconstructs Loops, सशर्त कथन और SQL कथन Execution भाग के भाग है।

Exception भाग

PL/SQL खंड का यह भाग जो कि गैर जरूरी है आरक्षित keyword EXCEPTION से शुरू होता है। और प्रोग्राम में त्रुटियों के संचालन में उपयोग में लाया जाता हैं ।

खंड की बनावट (Structure of PL/SQL Block):–

```
DECLARE
        Variable declaration
BEGIN
      Program Execution
Exception
      Exception handling
END;
```

साधारण PL/SQL प्रोग्राम का खंड(block) जो '' भारत '' शब्द को प्रदर्शित करता है।

```
SQL> set serveroutput on
1 SQL> begin
2  dbms_output.put_line ('BHARAT');
3 end;
4  /
BHARAT
```

PL/SQL procedure सफलतापूर्वक समाप्त हुई ।

PL/SQL प्रोग्राम के कुछ महत्वपूर्ण बिन्दु इस प्रकार है।

- PL/SQL प्रोग्राम खंड (block) का भाग keyword Begin से शुरू होता है एवं keyword END से समाप्त किया जाता है। जो कि PL/SQL खंड का executable भाग कहलाता है।
- PL/SQL प्रोग्राम खंड कथनों से मिलकर बनता है। प्रत्येक कथन सेमीकॉलन (;) से समाप्त होता है।
- PL/SQL प्रोग्राम खंड के अन्त में फॉरवर्ड स्लैस (/) लगाया जाता है। जो कि प्रोग्राम खंड के कथनों के लागुकरण के लिये उत्तरदायी हैं ।

PL/SQL के लाभ :–PL/SQL खंड के कुछ लाभ इस प्रकार है।

PL/SQL खंड की बनावट :–PL/SQL में, प्रोग्राम के खंड एक दूसरे के साथ लिखे जा सकते हें जिसे नेस्टिंग कहते है। प्रत्येक खंड सम्पूर्ण कार्य के छोटे या बड़े अंश के लिये उत्तरदायी होता है। PL/SQL खंड डाटाबेस में संग्रहित किये जाते है। और पुनः उपयोग में भी लाये जाते है।

बेहतर प्रदर्शन :–PL/SQL इंजन एक साथ कई SQL कथनों का एक खंड की तरह संचालित करता है। जो बेहतर प्रर्दशन और कम नेटवर्क traffic के लिये उत्तरदायी है।

पद्धतिकरण (Procedural) भाषा की क्षमता :–PL/SQL को और शक्तिशाली बनाने हेतु कुछ procedural language कन्सट्रक्ट का उपयोग होता हैं जैसे (if else) और (फॉरलूप्स)

PL/SQL प्लेस होल्डरस :–

प्लेस होल्डरस ऑकड़ों की अस्थायी रूप से संग्रहित करने के लिये जगह देतें है। जो कि PL/SQL खंड के execution के दौरान हेरफेर करने के लिए उपयोग किए जाते हैं। वेरियबल्स कॉन्सटेन्ट और रिकार्डस PL/SQL प्लेसहोल्डरस हैं।

PL/SQL के प्लेसहोल्डरस :–

प्लेस होल्डरस को नाम और (datatype) से वर्णित किया जाता हैं यह परिभाषा उन ऑंकड़ों पर निर्भर करती है जो हम संग्रहित करते हैं कुछ ऑंकडों के प्रकार (datatype) जो प्लेस होल्डरस को वर्णित करते है निम्नलिखित है।

Number (n,m) , Char (n) , Varchar2 (n) , Date , Long , Long raw, Raw, Blob, Clob, Nclob and Bfile.

PL/SQL वेरियबल्स :– ये वो प्लेस होल्डरस है जो वेल्यूस को संग्रहित करते है जो PL/SQL खंड के जरिये बदली जा सकती है।

वेरियबल्स को घोषित(declare) करने की साघारण सिन्टेक्स :–

variable_name datatype [NOT NULL := value];

variable_name is the name of the variable.डाटाटाईप यहॉं एक उचित PL/SQL डाटाटाईप है। NOT NULL वेरियबल के लिय एक गैर जरूरी व्याव्ख्या है।Value एवं डिफॉल्ट वेल्यू भी एक गैर जरूरी व्याव्ख्या हैं जहॉं पर आप वेरियबलस को इनिशियलाइज करते है। प्रत्येक वरियबलस की घोषणा एक अलग वाक्य है जो कि सेमीकॉलन से समाप्त होता हैं उदाहरणतः यदि आप एक शिक्षक का वर्तमान वेतन संग्रहित करना चाहते है तो वेरियबल उपयोग में लायेंगे।

DECLARE

salary_variable number (6);

“salary_variable” एक वेरियबल है जिसके datatype नंबर है और जिसकी 6 लम्बाई हैं।

जब कभी वेरियबलस का ब्यौरा NOT NULL से दिया जाता है। उस वेरियबल को initialize करें जब उसे घोषित कर दिया जाता है।

उदाहरणतः निम्नलिखित उदारहण दो वेरियबल की घोषणा करता है जिनमें से एक NOT NULL है।

```
DECLARE
salary_variable number(4);
address varchar2(10) NOT NULL := “ajmer”;
```

कोई भी वेरियबल अपना Value PL/SQL खंड के execution और exception भाग में बदल सकता है जबकि वेरियबलस की Value क्रमशः दो प्रकार से दिया गया हो।

हम मान सीधे तौर पर चर (वेरियबल्स) के लिए असाइन कर सकते हैं। जिसका सिन्टेक्स है।

```
variable_name:=  value;
```

डाटाबेस के स्तंभ से सीधे तौर पर मान वेरियबल्स के लिए SELECT.. INTO वाक्य से असाइन कर सकते हैं। जिसका सिन्टेक्स है।

```
SELECT Column_name INTO variable_name FROM table_name [where condition];
```

उदाहरण :– निम्नलिखित प्रोग्राम एक शिक्षक का salary एवं नाम बताता है। जहाँ शिक्षक का नाम“ radhakrishnan” है।

```
DECLARE
 salary_var number(6);
 tname_var char(15) = “radhakrishnan”;
BEGIN
 SELECT salary
 INTO salary_var
 FROM  teacher
 WHERE tname = tname_var;
 dbms_output.put_line(salary_var);
 dbms_output.put_line('The teacher '
          || tname_var || ' has  salary  ' || salary_var);
END;
/
```

टिप्पणी :– उपरोक्त प्रोग्राम के अन्त में फॉरवर्ड स्लैश (/) PL/SQL खंड के लागुकरण को प्रदर्शित करता हैं।

PL/SQL कॉन्सटेन्ट :–

यदि PL/SQL खंड में प्रयुंक्त Value पूरे प्रोग्राम में परिवर्तित न हो तो इस तरह का Value कॉन्सटेन्ट कहलाता हैं।

उदाहरणत :– यदि एक प्रोग्राम जो एक शिक्षक का वेतन 10 प्रतिशत बढा़ दे आप इस पूरे प्रोग्राम में एक स्थिर मूल्यांकन घोषित कर सकते है। अगली बार यदि फिर से वेतन में वृद्धि करनी पड़े तो उस स्थिर मूल्य की परिवर्तित करने से लक्ष्य प्राप्ति होगी जो कि निसंदेह उस प्रक्रिया से सरल है। जहाँ आप को पूरे प्रोग्राम में Value बदलना पड़ें।

किसी कॉन्सटेन्ट या स्थिर मूल्यांकन की घोषणा

name_constant CONSTANT datatype =VALUE;

name_constant मुख्यतः कॉन्सटेन्ट का नाम है जो कि वेरियबल नाम के ही समान हैं ।

constant (reserved word) नामक शब्द किसी भी स्थिर वेल्यू की घोषणा हेतु उपयोग किया जाता है।

VALUE- यह वह मूल्य या मूल्यांकन है जो घोषित होते समय किसी भी कॉन्सटेन्ट को दिया जाता है। इसके पश्चात् आप मूल्यांकन नहीं कर सकते है।

उदाहरण :– increase_salary की घोषणा हेतु , निम्नलिखित प्रोग्राम आप लिख सकते है।

DECLARE

increase_salary CONSTANT number (4) := 10;

टिप्पणी :– घोषणा के समय कॉन्सटेन्ट का मूल्यांकन आवश्यक है। ऐसा नहीं करने पर प्रोग्राम के execution में error का होना संभव है।

PL/SQL SET Serveroutput ON :–

"SET Serveroutput ON" हमेशा PL/SQL शुरू करने से पूर्व लिखना चाहिये ।PL/SQL प्रोग्राम का execution ऑरेकल इंजन में होता है, इसलिये सर्वर आउटपुट को स्क्रीन पर प्रदर्शित करने हेतु उसकी आवश्यकता पड़ती है अन्यथा परिणाम प्रदर्शित नहीं होगा।

SQL>सर्वर आउटपुट के उपयोग को समझने के लिये हम एक उदाहरण लेते है। प्रोग्राम की पहली लाइन सर्वर आउटपुट को शुरू करती है। तत्पश्चात् वेरियबल्स और कॉन्सटेन्ट का वर्णन होता है। प्लेस होल्डर वर्णित करने के पश्चात् dbms_output.put_line नामक निर्देश का उपयोग वर्णित वेरियबल के नाम को छापने हेतु होता है।

उदाहरण :–

SQL> set serveroutput on

SQL> DECLARE

```
        Sno number(4) NOT NULL := 3
        Sname varchar2(14) := 'Hari';
        Sclass CONSTANT varchar2(10) := '9th';
BEGIN
        dbms_output.put_line('Declared Value:');
        dbms_output.put_line(' Student Number: ' || Sno || ' Student Name: ' || Sname);
        dbms_output.put_line('Constant Declared:');
        dbms_output.put_line(' student Class : ' || Sclass);
END;
/
```

परिणाम प्रदर्शित करने के लिये "set serveroutput on" नामक निर्देश का लागुकरण आवश्यक है।

उपरोक्त कोड़ का आउटपुट

```
Declared Value:
Student  Number: 3 Student Name:  Hari
Constant Declared:
student Class: 9th
```

PL/SQLके नियमबद्ध(Conditional) वाक्य :–

नियमबद्ध वाक्यों के प्रकार एवं रचना निम्न प्रकार है।

IF THEN ELSE वाक्य :– इस प्रकार के वाक्यों में जब अवस्था अनुकूल या TRUE होती है। तब 1 वाक्य का execution होता है, 2 वाक्य को छोड़ दिया जाता है परन्तु जब अवस्था FALSE होती है। तब 1 वाक्य छोड़ कर 2 वाक्य का execution होता है।

वाक्य रचना (Syntaxes)

1)

```
IF condition
THEN
 statement 1;
ELSE
```

```
 statement 2;
END IF;

2)
IF condition 1
THEN
 statement 1;
 statement 2;
ELSIF condtion2 THEN
 statement 3;
ELSE
 statement 4;
END IF
```

PL/SQL में Iterative वाले वाक्य :–

जब हम किसी वाक्य या वाक्यों का execution एक से अधिक बार करना चाहते है तो हम पुनरावृति नियंत्रक वाक्य (iterativecontrol statements) का उपयोग करते हैं।PL/SQL में तीन तरह के लूप होते है

लूप के प्रकार एवं रचना इस प्रकार है

- साधारण लूप
- व्हाइल लूप
- फॉर लूप

साधारण लूप की रचना :–

```
LOOP
  Statements;
  EXIT;
  {or EXIT WHEN condition;}
END LOOP;
```

साधारण लूप का उपयोग करते समय कुछ महत्वपूर्ण बिन्दु पर ध्यान देंवें।

- सदैव लूप के प्रधान भाग से पहले वेरियबल्स का मूल्यांकित(initialize) करें।
- लूप के भीतर वेरियबल के मूल्य में इजाफा जरूर करना चाहिये ।
- यदि लूप से बाहर निकलना चाहे तो EXIT WHEN वाक्य लिखे। यदि EXIT, WHEN के साथ नहीं लिखा जाता हैं तो लूप का execution केवल एक बार होता है।

व्हाइल लूप (While Loop) :–

While Loop के भीतर लिखें सारे वाक्यों का executionतब तक होता है जब तक conditiontrue है। अवस्था का मूल्यांकन प्रत्येक बार तब लूप चलता है के साथ होता है। और जब तक होता है जब तक conditionfalse नहीं होती हैं।

While Loop लूप का साधारण सिन्टेक्स :–

WHILE <condition>

LOOP statements;

END LOOP;

फॉर लूप (FOR LOOP) :–

FOR LOOP के भीतर वाक्यों का तब तक execution होता है जब तक लूप में लिखी संख्या पूरी नहीं हो जाती है। लूप का चलन प्रारम्भिक से अन्तिम तक (integer values given)होता है। काउन्टर के मूल्य में प्रत्येक बार एक संख्या से वृद्धि होना निश्चित है। अन्तिम मूल्य पर पहुँचने पर काउन्टर लूप से बाहर निकल जाता है।

फॉर लूप का साधारण सिन्टेक्स :–

FOR counter IN from….to

LOOP statements;

END LOOP;

- From - Start integer value.
- to - End integer value.

फॉर लूप के execution के लिये कुछ महत्वपूर्ण उपाय :–

- काउन्टर वेरियबल घोशणा खंड में ही घोषित किया जाता है। घोषणा खंड के बाहर घोषित करना अनावश्यक हैं।
- काउन्टर वेरियबल 1 के द्वारा incremented है अनावश्यक वृद्धि करने की जरूरत नहीं है।

- EXIT WHEN वाक्य और EXIT वाक्य फॉर लूप के भीतर उपयोग में लाये जा सकते है परन्तु यह अनुचित है।

PL/SQL कर्सर :–

जब कभी किसी SQL वाक्य का लागुकरण होता है। एक अस्थायी work area मशीन memory में बनता है। जिसे कर्सर कहा जाता है। आँकड़ों की पंक्तियों एवं सलेक्ट वाक्य से सम्बन्धित सूचनायें कर्सर में होती है।

यह अस्थायी कार्य क्षेत्र उन आँकड़ों को संग्रहित करता है जो डाटाबेस से ग्रहण किये जाते है। कर्सर एक से ज्यादा पंक्तियों पर नियंत्रण रख सकता है। परन्तु एक समय में एक पंक्ति का ही क्रियान्वन संभव है। वो सारी पक्तियों का समूह जिस पर कर्सर का नियंत्रण होता है क्रियाशील (active set) समूह कहलाता है।

PL/SQL में कर्सर के प्रकार :– PL/SQL में कर्सर दो प्रकार के होते है।

अंतर्निहित (Implicit)कर्सर :– यह पहले से बने होते है जब किसी DML वाक्य जैसे INSERT, UPDATE, और DELETE का लागुकरण किया जाता है। किसी SELECT वाक्य जो केवल एक पंक्ति परिणाम स्वरूप देता है के लिये भी कर्सर उपयोग में जाते हें।

बाहानिहित (Explicit)कर्सर :– इनका निर्माण जरूरी है जब आप एक SELETE वाक्य का लागुकरण होता हैं एव ये वाक्य परिणाम स्वरूप दो पंक्तियों(more than one) प्रदर्शित करता है तो यद्यपि यह कर्सर एक से ज्यादा तथ्य संग्रहित करता है परन्तु एक समय में एक ही तथ्य का क्रियान्वन संभव है। जिसे वर्तमान कालीन (current) पंक्ति कहा जाता है। जब कभी कोई नयी पंक्ति आती है। तब पुरानी पंक्ति वर्तमान स्थिति से अगली स्थिति में आ जाती है। अंतर्निहित एवं बाहानिहित कर्सर की कार्य प्रणाली (functionality) एक समान होती है। परन्तु क्रियान्वन भिन्न है।

अंतर्निहित कर्सर के उदाहरण :–DML वाक्य जैसे INSERT,DELETE,UPDATE एवं SELECT के लागुकरण के लिये अंतर्निहित कर्सर उपयोग में लाये जाते है।

अंतर्निहित कर्सर के कुछ गुण (attributes)है जो DML की संचालन क्रिया (operations)की स्थिति बताते है। ये गुण %FOUND, %NOTFOUND, %ROWCOUNT, और %ISOPEN है।

उदाहरणतःINSERT, UPDATE, और DELETE वाक्यों के लागुकरण के समय कर्सर के गुण हमें ये बताते है कि इन वाक्यों कापंक्तियो पर प्रभाव हुआ है और कितनी पंक्तियाँ प्रभावित हुई है।जब कभी SELECT... INTO वाक्य का लागुकरण होता है तो अंतर्निहित कर्सर ये पत्ता लगाने के लिये उपयोगी हैं कि एक भी पंक्ति उपरोक्त वाक्य के द्वारा प्रदर्शित हुई है या नहीं।यदि कोई पंक्ति नहीं चुनी गई हैतोPL/SQL error देता है ।

%FOUND गुण :– प्राप्त परिणाम TRUE होगा यदि DML वाक्य जैसे INSERT,UPDATE, DELETE और SELECT... INTO कम से कम एक पंक्ति का संचालन करते है। परिणाम FALSE या अनुचित होगा यदि उपरोक्त वाक्यों के द्वारा एक भी पंक्ति का संचालन नहीं होता है।

%NOTFOUND गुण :– परिणाम असत्य या अनुचित होगा यदि DML वाक्य जैसे INSERT, UPDATE, DELETE और SELECT... INTO कम से कम एक पंक्ति का संचालन करें। यदि उपरोक्त वाक्यों से एक भी पंक्ति का संचालन न हो तो प्राप्त परिणाम उचित व सत्य होगा।

%ROWCOUNT गुण:–परिणाम स्वरूप उन सभी पंक्तियों की संख्या देता है जिनका संचालन INSERT,DELETE , UPDATE एवं SELECT के द्वारा हुआ है।

उदाहरणतः विचार करें कि एक PL/SQL खंड जो अंतर्निहित कर्सर के गुणों का उपयोग करता हैं जो कि निम्नाकिंत है।

```
DECLARE  rows_var number(7);
BEGIN
 UPDATE Teacher
 SET salary = salary + 100;
IF SQL%NOTFOUND THEN
   dbms_output.put_line(' salaries not  updated');
ELSIF SQL%FOUND THEN
   rows_var := SQL%ROWCOUNT;
   dbms_output.put_line('Salaries for ' || rows_var || 'teachers  are updated');
END IF;
END;
```

उपरोक्त PL/SQL खंड में टीचर नामक टेबल में उपस्थित सभी शिक्षकों के वेतन को UPDATE किया गया है। यदि किसी शिक्षक का वेतन UPDATE नहीं होता है। तो हमें त्रटि होने का सन्देश मिलता है। जो कि ' salaries not updated' हैं। अन्यथा हमें सन्देश मिलता हैं कि 'Salaries for 100 teachers are updated' यदि 'Teacher' टेबल में 100 पंक्तियों है।

Explicit कर्सर :–Explicit कर्सर का वर्णन PL/SQL खंड के घोषणा भाग में दिया जाता है इसका निमार्ण उस SELECT वाक्य के संदर्भ में किया जाता है जो परिणाम स्वरूप एक से अधिक पंक्तियों प्रदर्शित करता हैं कर्सर को इच्छानुसार उपयुक्त नाम दे सकते हैं।

कर्सर के निर्माण के लिये साधारण रचना :–

CURSOR cursor_name IS select_statement;

- cursor_name – कर्सर का नाम
- select_statement –एक सलेक्ट query जो एक या एक से अधिक पंक्तियाँ प्रदर्शित करे।

Explicit कर्सर के उपयोग हेतु बिन्दु :–

- घोषणा भाग में कर्सर को घोषित करे।
- Execution भाग में कर्सर को सक्रिय (OPEN)करें।
- Execution भाग में PL/SQL वेरियबल्स या तथ्यों में कर्सर से आँकड़ें लावें।
- PL/SQL खंड कि समाप्ति से पहले कर्सर को निश्क्रिय (CLOSE) करें।

संग्रहित कार्य पद्धति (Stored Procedures):– नामांकित PL/SQL खंड जो एक या एक से अधिक कार्यों का क्रियान्वन करें संग्रहित **Procedures** या साधारण **Procedures** कहलाता हैं। **Procedures** में एक हैडर व बॉड़ी होता है। जहाँ हैडर मे **Procedures** का नाम और पैरामीटर या वेरियबल्स का नाम होता है जो कि **Procedures** को पास किये जातें है। बॉडी में एक declaration भाग execution भाग एवं exception भाग होता हैं। जो कि साधारण PL/SQL खंड के समान ही है। परन्तु एक से अधिक बार उपयोग में लाने हेतु **Procedures** को नामांकित किया जाता है।

Procedures में पैरामीटर को पास करना :–तीन प्रकार से पैरामीटर को **Procedures** में पास कर सकते हैं।

- IN-parameters
- OUT-parameters
- IN OUT-parameters

Procedures के क्रियान्वन हेतु साधारण रचना :–

CREATE [OR REPLACE] PROCEDURE proce_name [parameter list]

```
IS
  Declaration section
BEGIN
  Execution section
EXCEPTION
  Exception section
END;
```

IS :Procedures की बॉडी के प्रारम्भ को चिहिन्त करता है एवं PL/SQL के घोषणा भाग के समान ही हैं। IS एवं BEGIN के मध्य लिखा हुआ भाग ही घोषणा भाग कहलाता हैं।

जो भाग बड़े कोश्ठक [ ] में लिखा जाता है। गैर जरूरी होता है। हम CREATE या REPLACE को एक साथ तभी प्रयुक्त करें जब समान नाम से कोई और **Procedures** न हो या अस्तित्व में जो **Procedures** है वर्तमान कोड़ से बदली जाये।

उदाहरणत :– निम्नलिखित उदाहरण एक **Procedures** जिसका नाम 'student_info' है का निर्माण करता है। एवं इस **Procedures** का कार्य विद्यार्थियों की सूचनाएँ प्रदान करना है।

```
1> CREATE OR REPLACE PROCEDURE student_info
2> IS
3>  CURSOR stu_cur IS
4>  SELECT roll_no, Sname, age FROM Student;
5>  stu_rec stu_cur%rowtype;
6> BEGIN
7>  FOR stu_rec in 1..7
8>  LOOP
9>  dbms_output.put_line(stu_cur.Roll_no || ' ' ||stu_cur.Sname
10>    || ' ' ||stu_cur.age);
11> END LOOP;
12>END;
13> /
```

हम दो प्रकार से **Procedure** का लागुकरण कर सकते है।

(1) SQL prompt से

EXECUTE [or EXEC] procedure_name;

(2) दूसरी **Procedure** के अन्तर्गत –**Procedure** के नाम से

procedure_name;

PL/SQLफंक्शन :– यह एक तरह की नामांकित PL/SQL Block है जो कि **Procedure** के समान ही है। फंक्शन एवं **Procedure** में एक बड़ा अन्तर यह है कि फंक्शन में सदैव एक मान दिया जाता है पर **Procedure** में यह हो भी सकता है और नहीं भी।

फंक्शन की साधारण रचना :–

```
CREATE [OR REPLACE] FUNCTION func_name [parameters list]
RETURN return_datatype;
IS
Declaration_part
BEGIN
Execution_part
Return return_variable;
EXCEPTION
exception_part
Return return_variable;
END;
```

- Return Type हैडर वाला भाग प्रक्रिया के रिर्टन टाईप को वर्णित करता है। रिर्टन डाटा टाईप कोई भी उपयुक्त डाटा टाईप हो सकता है। जैसे varchar, number etc..
- Execution भाग एवं exception भाग हैडर में वर्णित डाटा टाईप का ही मान वापस देते है।

उदाहरण :– हम एक फंक्शन का निर्माण करतें हैं जिसका नाम "student_info_func' है।

1> CREATE OR REPLACE FUNCTION student_info_func

```
2>   RETURN VARCHAR(10);
3> IS
5>   stu_name VARCHAR(20);
6> BEGIN
7>      SELECT Sname INTO stu_name
8>      FROM student WHERE Roll_no = '58';
9>      RETURN stu_name;
10> END;
11> /
```

उपरोक्त उदाहरण में वो 'Sname' प्राप्त किये जिनका अनुक्रमांक 58 है और उन्हें 'stu_name' वेरियबल में संग्रहित किया। प्रक्रिया के रिर्टन टाईप का प्रकार VARCHAR है जिसकी घोषणा 2 नम्बर पंक्ति में की गई है। पूरी फंक्शन 'stu_name' पंक्ति नम्बर 9 मेंपरिणाम स्वरूप देती है। जिसके आँकड़े का प्रकार VARCHAR है।

Executing फंक्शन :–

- क्योंकि फंक्शन परिणाम स्वरूप एक मान देती है। यह वेरियबल को सौंप सकते है।

  student_name := student_info_func;

  यदि 'student_name' के आँकड़ों का प्रकार VARCHAR हो तो हम विद्यार्थी का नाम फंक्शन के रिर्टन टाईप को इसे सौंपकर संग्रहित कर सकते है।

- SELECT वाक्य के एक अंश की तरह

  SELECT student_info_func FROM student;

- PL/SQL के कुछ वाक्यों में

  dbms_output.put_line(student_info_func);

  यह पंक्ति फंक्शन द्वारा रिर्टन मान को प्रदर्शित करती है।

Exception Handling :–

PL/SQL की एक सुविधा यह हैं कि यह PL/SQL खंड में आये Exceptions कि Handling करता है, जिसे Exception Handling कहा जाता है। Exception Handling का उपयोग कर हम वर्तमान प्रोग्राम या कोड़ की जांच कर सकते है।

जब कभी कोई Exception आता है। Exception के कारण की व्याख्या का एक messages प्राप्त किया जाता हैं। PL/SQLExceptionmessages के तीन भाग होते है।

1) Type of Exception
2) An Error Code
3) A message

Exception Handling से यह निश्चित है कि PL/SQL खंड एकाएक समस्या का कारण नहीं बनता है।

अपवाद संचालन की संरचना

Exception भाग की साधारण रचना :–

```
DECLARE
  Declaration part
 BEGIN
  Exception part
 EXCEPTION
 WHEN excep1name THEN
   Error handling statements
 WHEN excep2name THEN
   Error handling statements
 WHEN Others THEN
  Error handling statements
END;
```

Exception खंड में PL/SQL वाक्य :–

जब कभी कोई Exception आता है। उपयुक्त exception handler की खोज की जाती है। उदारहरण उपरोक्त में यदि 'excep1name ' Exception आता है तो Exception Handling नीचे लिखी पंक्ति के अनुरूप होता है। बावजूद इसके यह संभव नहीं है कि कोड़ की जाँच के दौरान सारी त्रुटियों का आंकलन हो सकें। 'WHEN Others' Exceptions का प्रयोग उन Exceptions Handling के लिए किया जाये जिनका explicitly handling संभव न हो। केवल एक अपवाद एक खंड में संभव है और त्रुटि के संचालन के बाद नियंत्रण Execution भाग को वापिस नहीं दिया जाता है।

```
DELCARE
  Declaration part
BEGIN
DECLARE
Declaration part
BEGIN
Execution part
EXCEPTION
    Exception part
END;
EXCEPTION
  Exception part
END;
```

यदि उपरोक्त उदाहरण में नैस्टेड़ PL/SQL खंड हो एवं यदि Exception आँतरिक खंड में हो तो उस Exceptions का निस्तारण PL/SQL के आँतरिक खंड के Exceptions खंड में होना चाहिये अन्यथा नियंत्रण अगले PL/SQL खंड के Exceptions खंड को दे दिया जाता है।

Exception के प्रकार :–तीन प्रकार के Exceptions होते है।

(1) Named System Exceptions

(2) Unnamed System Exceptions

(3) User-defined Exceptions

Named System Exceptions

तंत्र के Exceptionsautomatically ऑरेकल द्वारा चिन्हित किये जाते है यदि कोई प्रोग्राम RDBMS के नियम की पालना नहीं करता है और कुछ ऐसे अपवाद बार–बार विघ्न डालते है। जो कि पूर्व वर्णित एवं ऑरेकल में नामांकित होते है तो ऐसे अपवाद नामांकित तंत्र अपवाद कहलाते है।

उदाहरण :– NO_DATA_FOUND and ZERO_DIVIDENamed System Exceptions है।

Named System Exceptions

(1) बाहयनिहित (explicitly) घोषित नहीं होते है।

(2) अंतर्निहित(implicitly) रूप से जाग्रत होते है जब कभी ऑरेकल की पूर्व वर्णित त्रुटियों होती है।

(3) Exceptions Handling रूटीन में सामान्य नाम के हवालें से चिन्हित कियें जाते है।

उदाहरण :–विचार करिये यदि NO_DATA_FOUND Exceptions किसी कार्यपद्धति में जाग्रत होता है। हम Exceptions Handling हेतु निम्नांकित कोड लिख सकते है।

```
BEGIN
 Execution part
EXCEPTION
WHEN NO_DATA_FOUND THEN
 dbms_output.put_line (' Using SELECT...INTO did not get any row.');
 END;
```

Unnamed System Exceptions

वो सारे तंत्र **Exception** जिनके लिये ऑरेकल की तरफ से नाम नहीं दिया जाता है **Unnamed System Exceptions** कहलाते है। ये **Exceptions** नियमित रूप से जाग्रत नहीं होते हें इन **Exceptions** के साथ एक प्रोग्राम और एक सन्देश होता है।

Unnamed System Exceptions दो प्रकार से होता हैं ।

(1) WHEN OTHERS **Exception handler** का उपयोग कर या

(2) **Exception** कोड़ को नाम देकर **named System Exceptions** की तरह उपयोग करे। EXCEPTION_INIT एक पूर्वनिर्धारित Oracle त्रुटि संख्या उपयोग कर हम प्रोग्राम नामक **Exception** को नाम दे सकते है।

- **Unnamed System Exceptions** को उपयोग करते समय कुछ ध्यान रखने योग्य मुख्य बिन्दु इस प्रकार है।
- ये अंतर्निहित रूप से जाग्रत होते हैं।
- दूसरों के साथ संचालन न होने की स्थिति में ये बाहयनिहित रूप से संचालित होते हैं।
- **Exceptions** को बाहयनिहित रूप से संचालन हेतु उनकी घोषणा PragmaEXCEPTION_INIT

- के उपयोग से होवे एवं संचालन उपयोगकर्ता वर्णित **Exception**के नाम से **Exception** भाग में होना चाहियें।

EXCEPTION_INIT का उपयोग करते हुए Unnamed System Exceptions की घोषणा हेतु साधारण रचना :–

```
DECLARE
excep_name EXCEPTION;
PRAGMA
EXCEPTION_INIT (excep_name, Err_code);
BEGIN
Execution part
EXCEPTION
WHEN excep_name THEN
   handle the exception
END;
```

User-defined Exceptions

System Exceptions के अलावा हम उन **Exceptions** को जो व्यवसायिक नियम पर आधारित है की बाहयनिहित रूप से वर्णित कर सकते है। और इन्हें user-defined exceptions कहते है। user-defined exceptions के उपयोग हेतु मुख्य बिन्दु :–

- इनकी घोषणा बाहयनिहित रूप से जाग्रत किया जावें ।
- user-defined exceptions के नाम का हवाला देते हुए exception खंड में संचालन किया जावें।

ट्रिगरस

परिभाषा :–PL/SQL खंड की संरचना है जिसका उपयोग DML वाक्य जैसे Insert, Delete, Update के डेटाबेस तालिका पर execution के समय किया जाता है। यह automatically प्रयुक्त होता है जब उपरोक्त DML वाक्यों का execution होता है। डाटाबेस ट्रिंगर के तीन अंश है।

(1) **ट्रिगरिंग इवेन्ट** :– ट्रिंगर के execution के लिये जवाबदेय

(2) **शर्त (Condition)** :– ट्रिंगर execution के लिये शर्त का पूरा होना जरूरी है।

(3) गतिविधि (**Action**) :– ट्रिंगर के execution के समय घटित गतिविधि का ब्यौरा

ट्रिंगर की रचना :–

```
CREATE [OR REPLACE ] TRIGGER name_of_trigger
 {BEFORE I AFTER I INSTEAD OF }
 {INSERT [OR] I UPDATE [OR] I DELETE}
 [OF name_of_col]
 ON table_name
[REFERENCING OLD AS O NEW AS N]
 [FOR EACH ROW]
WHEN (condition)
BEGIN
  --- sql statements --
END;
```

ट्रिंगर Syntax:

- CREATE [OR REPLACE] TRIGGER name_of_trigger – In PL/SQL में ट्रिंगर जिसका नाम दिया गया हो के निमार्ण हेतु या फिर ट्रिंगर जो अस्तित्व में है को ओवरराइट करने हेतु इस क्लॉस का उपयोग होता है।
- {BEFORE I AFTER I INSTEAD OF } यह क्लॉस दर्शाता है ट्रिंगर का उपयोग किस समय किया जाना चाहिये उदाहरणतः टेबिल के UPDATE के पहले और पश्चात् INSTEAD OF का उपयोग view के समय ट्रिंगर के निर्माण के लिये किया जाता है।Before और after का उपयोग अपमू के समय ट्रिंगर निर्माण में नहीं किया जा सकता हैं ।
- {INSERT [OR] I UPDATE [OR] I DELETE} यह क्लॉज ट्रिंगरिंग की घटना (event) को बताता है। एक से अधिक ट्रिंगरिंग की घटनाओं के एक साथ होने की स्थिति में घटनाओं को पृथक करनें हेतु keyword का उपयोग होता है। सारी ट्रिंगरिंग की घटनाओं के समय ट्रिगर का उपयोग होता हैं ।
- [OF name_of_col] यह क्लॉज UPDATE ट्रिंगर के साथ उपयोग में लाया जाता है। एक विशेष स्तंभ के जुड़ने पर यदि कोई घटना ट्रिंगर होती है। तो यह क्लॉज उपयोग किया जाता है।

- [ON table_name] यह क्लॉज टेबिल का नाम सत्यापित करता है। या फिर टेबिल का वह view जिसके साथ ट्रिंगर जोड़ा गया है।
- [REFERENCING OLD AS O NEW AS N] इस क्लॉज का उपयोग जो डाटा परिवर्तित हो रहा है पुराने और नए मान को संदर्भित करने के लिए उपयोग किया जाता है।डिफॉल्ट रूप से, आप मानों को old-column_name or :new-column_name के रूप में संदर्भ किया जाता है। संदर्भित OLD or NEW नामों को user-defined नामों से परिवर्तित कर सकतें है। पुराने मान तथ्यों केinserting के समय उल्लेखित नहीं किये जा सकते और नये मान तथ्यों के delete करते समय क्योंकि वो अस्तित्व में नहीं होते हैं।
- [FOR EACH ROW] इसका उपयोग यह बताने में किया जाता है कि ट्रिंगर का उपयोग जब प्रत्येक पंक्ति प्रभावित हो तब किया जावे या नहीं (i.e. a Row Level Trigger)या फिर केवल एक बार उस समय जब पूरे SQL वाक्य का execution हो।
- WHEN (कन्डीशन) यह केवल पंक्ति लेवल टिंगर के लिये उपयुक्त है। ट्रिंगर उन्हीं पंक्तियों के लिये चलते है जो दी हुई शर्तों को पूरी करती हैं।

उदाहरण :– स्टूडेन्ट classes सदैव परिवर्तित होती है। इसलिये students के class के इतिहास को संग्रहित करना आवश्यक है।

PL/SQL Trigger के प्रकार :–

दो प्रकार के ट्रिंगरस है जो मुख्यतः उस स्तर पर आधारित है जहाँ ट्रिंगर का उपयोग होता है।

(1) रो लेवल ट्रिंगर :–जब प्रत्येक रो आधारित प्रविष्टी हटायी या update की जाती है तो तब एक घटना ट्रिंगर होती है।

(2) स्टेटमेन्ट लेवल ट्रिंगर :– प्रत्येक SQL वाक्य के execution हेतु एक घटना टिंगर होती है।

PL/SQL ट्रिंगर execution Hierarchy :– जब ट्रिंगर उपयोग में आता है। तो निम्नलिखित हाइराइकी उपयोग में लायी जाती है।

(1) सबसे पहले जब BEFORE वाक्य ट्रिंगर उपयोग किया जाता है।

(2) उसके बाद जब BEFORE रो लेबल ट्रिंगर उपयोग मे आता है।

(3) उसके बाद जब AFTER रो लेबल ट्रिंगर उपयोग में आता है। यह घटना BEFORE और AFTER रो लेबल ट्रिंगर के मध्य अल्टरनेट देती है।

(4) सबसे अन्त में AFTER स्टेटमेन्ट ट्रिंगर उपयोग में आता है।

हम student table के लिये BEFORE और AFTER स्टेटमेन्ट और रो लेबल ट्रिंगर का निर्माण करते है।

महत्वपूर्ण बिंदु

- PL/SQL के तीन भाग : घोषणा भाग, लागुकरण भाग और अपवाद संचालन भाग होते है।
- "SET Serveroutput ON" हमेशा PL/SQL शुरू करने से पूर्व लिखना चाहिये।
- यदि EXIT, WHEN के साथ नहीं लिखा जाता हैं तो लूप का execution केवल एक बार होता है।
- कर्सर एक से ज्यादा पंक्तियों पर नियंत्रण रख सकता है। परन्तु एक समय में एक पंक्ति का ही क्रियान्वन संभव है। वो सारी पक्तियों का समूह जिस पर कर्सर का नियंत्रण होता है क्रियाशील (active set) समूह कहलाता है।
- फंक्शन एवं **Procedure** में एक बड़ा अन्तर यह है कि फंक्शन में सदैव एक मान दिया जाता है पर **Procedure** में यह हो भी सकता है और नहीं भी।
- ट्रिगर automatically प्रयुक्त होता है जब DML वाक्यों का execution होता है।

अभ्यासार्थ प्रश्न

वस्तुनिष्ठ प्रश्न

प्रश्न 1. जो एक PL/SQLका हिस्सा नहीं है

(अ) Declare (ब)BEGIN (स) Start (द) End

प्रश्न 2. PL/SQL द्वारा विकसित की है

(अ) IBM (ब) ORACLE (स) Microsoft (द) इनमे से कोइनही

प्रश्न 3. शब्द का चयन करें जो select कथन के साथ उपयोग किया जाना चाहिए।

(अ) Goto (ब) Into (स) Do (द) all

प्रश्न 4. कर्सर कितने प्रकार के है।

(अ) 2 (ब) 4 (स) 5 (द) 1

प्रश्न 5. % FOUND का काम विपरीत है।

(अ) %CURSOR (ब) % NOT COUNT

(स) %NOT FOUND (द) % FOUND COUNT

अतिलघुत्रात्मक प्रश्नः

प्रश्न 1. PL/SQL क्या है?

प्रश्न 2. PL/SQL ब्लॉक में कितने भाग हैं?

प्रश्न 3.PL/SQLमें Declare का उपयोग क्योंकरें ?

प्रश्न 4. PL/SQL में & का उपयोग क्या है।

प्रश्न 5.PL/SQL में Variables कहां घोषित किया जाता है?

प्रश्न 6. PL/SQL select कथन का प्रयोग कैसे किया जाता है ?

प्रश्न 7.exception ब्लॉक का क्या उपयोग है ?

प्रश्न 8. PL/SQL.में variable के प्रकार को बताए।

प्रश्न 9. ट्रिगर क्या है?

प्रश्न 10. हम ट्रिगर का उपयोग कैसे करें ?

लघुत्रात्मक प्रश्नः

प्रश्न 1.%TYPE गुणऔर %ROWTYPE गुण प्रकार के बीच क्या अंतर है?

प्रश्न 2. PL/SQL में EXIT कथन का उपयोग है?

प्रश्न 3.before ट्रिगर क्या है ?

प्रश्न 4.implicit और explicitकर्सर के बीच क्या अंतर है ?

प्रश्न 5.for Loop का सिंटैक्स लिखे।

निबंधात्मक प्रश्नः

प्रश्न 1. कर्सर क्या है? कर्सर का उपयोग क्या है? उदाहरण के साथ explicit कर्सर की व्याख्या करें।

प्रश्न 2. विभिन्न प्रकार के डेटाबेस ट्रिगर्स उदाहरण के साथ समझाइए।

प्रश्न 3.exceptions क्याहै? विभिन्न प्रकार की exceptions की व्याख्या करें।

प्रश्न 4. PL/SQL में लूप के विभिन्न प्रकार समझाइए।

प्रश्न 5 function क्या है? यह Procedure से अलग कैसे है? functions और Procedure के लिए सिंटैक्स समझाइए।

उत्तरमाला

उत्तर 1: स **उत्तर** 2: ब **उत्तर** 3: ब

उत्तर 4: अ **उत्तर** 5: स