בסיסי נתונים - מעבדה

SQL

ORACLE

PL/SQL

PL/SQL - הקדמה

- PL/SQL מרחיבה את SQL ומחזקת את כוחו על ידי הענקת כלים משפות פרוצדוראליות.
- בלוק (תוכנית) ב-PL/SQL הוא בעל המבנה הבא:

DECLARE

משתנים, טיפוסים ופרוצדורות. הכרזות

BEGIN

חלק ביצועי: משפטים פרוצדוראליים ומשפטי SQL

* **זהו החלק היחיד שנדרש בבלוק**. (זהו החלק העיקרי שרץ).

EXCEPTION

חלק הטיפול בחריגים. משפטי טיפול בטעויות יבואו כאן.

END;

PL/SQL - הקדמה

- משפטי ה- SQL שיכולים להשתלב בתוכנית PL/SQL הם אלו המתחילים ב-SELECT, UPDATE, DELETE.
- חשוב להזכיר כי **תחביר ה- SELECT שונה מהרגיל** (נראה בהמשך).
- משפטי הגדרה כמו CREATE, DROP, ALTER **אינם** מותרים ב-PL/SQL.
- הכלים הפרוצדוראליים ש-PL/SQL תומך בהם הם:

  - השמות.

  - התניות.

  - לולאות.

  - קריאות לפרוצדורות.

  - והדקים (Triggers).

PL/SQL - הקדמה

- PL/SQL בדומה ל-SQL אינו רגיש לאותיות קטנות/גדולות.
- ישנן שתי דרכים לסימון הערות בקוד :
  - סימון שורה אחת, ע"י התווים '--'
  - סימון כמה שורות בבת אחת (בדומה לשפת C), ע"י התווים /* ..... */.
- כדי להריץ תוכנית PL/SQL עלינו להוסיף 2 שורות בסוף התוכנית:

- הראשונה רק עם נקודה (A line with single dot).

- ולאחריה שורה עם ;run (הסימן נקודה פסיק לאחר המלה run).

או

הוספת סימן /

PL/SQL - טיפוסי נתונים

- אינפורמציה בין תוכנית ה- PL/SQL לבין בסיס הנתונים מועברת דרך משתנים.
- ב- PL/SQL שם משתנה חייב להתחיל באות, ואורכו המקסימאלי יהיה 30 תווים.
- טיפוסי המשתנים המותרים הם:
  - טיפוסים שבהם אנו משתמשים ב-SQL לשדות.,וטיפוסים נוספים כגון BOOLEAN

Datatype	Max Size	Default Size	Description
NUMBER (width,scale)	38	38	Numeric values rounded to a whole number unless a scale is given, i.e. NUMBER(5,2) means a numeric values 5 digits in length allowing for 2 decimal places
VARCHAR2 (width)	32767 *		Variable length character data
CHAR (width)	32767 **		Fixed length character data
DATE			Valid date values
BOOLEAN			Boolean values TRUE and FALSE

PL/SQL - טיפוסי נתונים

דוגמה להגדרה:

```
DECLARE
    i  NUMBER;
    str  VARCHAR2(15);
```

PL/SQL - טיפוסי נתונים

- במקרים רבים, משתנה משמש לצורך ביצוע מניפולציות על מידע שמאוחסן ב- Relation קיים.

  במקרה זה וכדי למנוע טעות בהגדרת המשתנה, מגדירים את המשתנה בדיוק כטיפוס השדה על ידי הסימון %TYPE.

  - לדוגמא:

```
DECLARE
    FirstNum Table1.Num1%TYPE;
```

מגדיר משתנה ששמו FirstNum שהוא כטיפוס השדה Num1 ב-Table1.

PL/SQL - טיפוסי נתונים

- משתנה יכול להיות מטיפוס רשומה עם כמה שדות. הדרך הפשוטה להגדרה היא על ידי %ROWTYPE:

```
DECLARE
    NumbersTuple Table1%ROWTYPE;
```

יוצר משתנה NumbersTuple להיות רשומה עם השדות Num1,Num2,Num3 (מתוך ידיעה שה-Relation ששמו Table1 הוא הסכימה Table1(Num1,Num2,Num3).

PL/SQL - טיפוסי נתונים

- ערכו ההתחלתי של כל משתנה, ללא תלות בטיפוסו, הוא NULL.
- ניתן לבצע השמה למשתנים על ידי האופרטור =: (נקודתיים שווה).
- כדוגמה:

```
 DECLARE
    a NUMBER:=2;
BEGIN
    a := 1;
END;
.
run;
```

Select Into - PL/SQL

- השינוי העיקרי ששונה מהצורה של SQL הוא בצורה של פקודת ה-SELECT ב-PL/SQL.
- אחרי ה- SELECT ושדותיו, **חייבת** לבוא המלה INTO ורשימת משתנים, אחד לכל attribute של ה- SELECT, לשם ייכנסו מרכיבי **<u>השורה</u>** המוחזרת מה-SELECT.
  - חשוב לציין, ששורה מוחזרת ולא כמה שורות מוחזרות, היות וב- PL/SQL ה- SELECT **יעבוד רק אם תוצאת השאילתה היא שורה בודדת**.
  - במקרים שתוצאת השאילתה מחזירה יותר משורה בודדת, יש לעבוד עם Cursor (סמן, מצביע), כפי שיתואר בהמשך.

Select Into - PL/SQL

- דוגמה:

```
CREATE TABLE T1(
   e INTEGER,
   f INTEGER
);

INSERT INTO T1 VALUES(1, 3);
INSERT INTO T1 VALUES(2, 4);
                    /* Above is plain SQL; below is the PL/SQL program. */
DECLARE
   a NUMBER;
   b NUMBER;
BEGIN
   SELECT e,f INTO a,b FROM T1 WHERE e>1;
   INSERT INTO T1 VALUES(b,a);
END;
.
run;
```

PL/SQL – צפייה בתוצאות

- שיטה א' : שימוש בטבלה.

1. ניצור טבלה.

```
CREATE TABLE my_debug
( date_created DATE, text VARCHAR2(500));
```

2. את מה שאנו רוצים לבדוק/פלט נכתוב לטבלה שיצרנו

```
DECLARE
l_x NUMBER := 0;
BEGIN
INSERT INTO my_debug
    VALUES (SYSDATE,'Before='||TO_CHAR(l_x));
l_x := l_x + 10;
INSERT INTO my_debug
    VALUES (SYSDATE,'After='||TO_CHAR(l_x));
END;
.
Run;
```

PL/SQL – צפייה בתוצאות

- המשך.

3. נתשאל את הטבלה.

```
SELECT text
FROM my_debug
ORDER BY date_created;
```

4. התוצאה:

```
TEXT
--------------
Before=0
After=10
```

PL/SQL – צפייה בתוצאות

- שיטה ב' : שימוש במשתנים מקשרים.
  - נוכל להדפיס רק משתנה שהוגדר בצורה המיוחדת
    VARIABLE <name> <type>,
    כאשר הטיפוס הוא מסוג: NUMBER, CHAR(n), CHAR.
    נוכל לבצע השמה למשתנה זה, כאשר לפני המשתנה נשים **נקודותיים**.
    כעת, **מחוץ ל-PL/SQL** נוכל לכתוב: PRINT :<name>;

1. נצהיר על המשתנה.

```
VARIABLE result NUMBER
```

PL/SQL – צפייה בתוצאות

- שיטה ב' : המשך.

2. נכתוב קטע קוד.

```
DECLARE
    l_x NUMBER := 0;
BEGIN
    l_x := l_x + 10;
    :result := l_x;
END;
.
Run;
```

3. נדפיס את הערך.

```
print result;
```

4. תוצאה:

```
RESULT
--------------
10
```

PL/SQL – צפייה בתוצאות

- שיטה ג' : שימוש ב- DBMS_OUTPUT.
- DBMS_OUTPUT - הינה חבילה הכוללת מספר פרוצדורות ופונקציות אשר מכניסות מידע ל- buffer.

1. אפשור הפלט.

SET SERVEROUTPUT ON;

2. להשתמש באחת הפונקציות בכדי להדפיס.

Procedure	Description
PUT	Append text to the current line of the output buffer
NEW_LINE	Put and end-of-line marker into the output buffer, this effectively flushes the buffer.
PUT_LINE	Append text and an end-of-line marker to the current line in the output buffer. This also flushes the buffer.

PL/SQL – צפייה בתוצאות

- שיטה ג' : המשך.

לדוגמה:

```
DBMS_OUTPUT.put_line('Hello World');
```

או

```
DBMS_OUTPUT.put('Hello');
DBMS_OUTPUT.put_line(' World');
```

או

```
DBMS_OUTPUT.put('Hello World');
DBMS_OUTPUT.new_line;
```

PL/SQL – צפייה בתוצאות

- שיטה ג' : המשך.

דוגמה מלאה:

```
SET SERVEROUTPUT ON;
DECLARE
        X NUMBER;
BEGIN
    X:=10;
    DBMS_OUTPUT.put_line('Hello World');
    DBMS_OUTPUT.NEW_LINE;
    DBMS_OUTPUT.PUT_LINE('ERROR ' || x);
END;
.
RUN;
```

UPDATE, INSERT and DELETE - PL/SQL

- ניתן להשתמש בכל אחת מפקודות אלו בתוך PLSQL.
- ניתן לקבל מידע על הפעולה כגון כמה שורות השתנו...

Attribute	Description
SQL%ROWCOUNT	The number of rows processed by the SQL statement
SQL%FOUND	TRUE if at least one row was processed by the SQL statement, otherwise FALSE
SQL%NOTFOUND	TRUE is no rows were processed by the SQL statement, otherwise FALSE.

UPDATE, INSERT and DELETE - PL/SQL

- לדוגמה:

```
DECLARE
   i NUMBER := 1;
BEGIN
      DELETE  FROM t1  WHERE  e>50;
      DBMS_OUTPUT.PUT_LINE(SQL%ROWCOUNT || ' rows deleted');
END;
/
```

input / output - PL/SQL

```
DECLARE
    i integer;
    j integer;
BEGIN
    i:=2;
    DBMS_OUTPUT.PUT_LINE('This is a message and the value of
        i is ' || i);
    DBMS_OUTPUT.PUT_LINE('If we want to input, we put
        ampersand before the variable');
    j:=&k;
    DBMS_OUTPUT.PUT_LINE('The value of j is '|| j);
END;
.
run;
```

נשים לב, **כי k אינו משתנה חוקי בתוכנית.**

- אנו מציבים את ערך הקלט למשתנה j שאיתו נעבוד.

PL/SQL - התניות

- משפט IF נראה כך :

```
IF <condition> THEN <statement_list> ELSE <statement_list> END IF;
```

חלק ה-ELSE אינו הכרחי.

- תנאי מקונן יראה כך :

```
IF <condition_1> THEN
    --Action
ELSIF <condition_2> THEN
    --Action
ELSIF <condition_n> THEN
        --Action
ELSE ...
    --Action
END IF;
```

PL/SQL - התניות

- לדוגמא :

```
DECLARE
   a NUMBER;
   b NUMBER;
BEGIN
   SELECT e,f  INTO a,b FROM T1 WHERE e>1;
   IF b=1 THEN
      INSERT INTO T1 VALUES(b,a);
   ELSE
      INSERT INTO T1 VALUES(b+10,a+10);
   END IF;
END;
.
run;
```

PL/SQL - התניות

- דוגמה נוספת:

```
DECLARE
   i NUMBER := 1;
BEGIN
       DELETE FROM t1 WHERE  e>50;
       IF (SQL%ROWCOUNT>0) THEN
            DBMS_OUTPUT.put_line(SQL%ROWCOUNT || ' rows deleted');
       ELSE
             DBMS_OUTPUT.put_line('no data to change');
      END IF;
END;
.
run;
```

PL/SQL - התניות

- שימוש ב- SQL%found:

```
DECLARE
   i NUMBER := 1;
BEGIN
      DELETE FROM t1 WHERE  e>50;
      IF (SQL%FOUND)  THEN
            DBMS_OUTPUT.put_line(SQL%ROWCOUNT|| ' rows deleted');
      ELSE
            DBMS_OUTPUT.put_line('no data to change' );
      END IF;
END;
.
run;
```

PL/SQL - לולאות

- לולאות נכתבות בצורה הבאה:

```
LOOP
    <loop_body>  /* A list of statements. */
END LOOP;
```

- כאשר לפחות שורה אחת ב- <loop_body> תכלול משפט יציאה מהצורה: EXIT

```
EXIT WHEN <condition>;
```

הלולאה תצא כאשר התנאי מתקיים,
וכמובן במקום שנאמר לה לצאת.

PL/SQL - לולאות

- לדוגמא:

```
DECLARE
   i NUMBER := 1;
BEGIN
   LOOP
      INSERT INTO T1 VALUES(i,i);
      i := i+1;
      EXIT WHEN i>100;
   END LOOP;
END;
.
run;
```

הלולאה תכניס לטבלה T1. את הזוגות (1,1) עד (100,100)

PL/SQL - לולאות

- לולאת WHILE היא מהצורה:

```
WHILE <condition> LOOP
        <loop_body>
END LOOP;
```

- לדוגמה:

```
DECLARE
l_x NUMBER := 10;
BEGIN
        While  l_x <100  LOOP
            DBMS_OUTPUT.put_line(l_x );
            l_x := l_x + 10;
        END LOOP;
END;
.
Run;
```

PL/SQL - לולאות

- לולאת FOR היא מהצורה:

```
FOR <var> IN <start>..<finish> LOOP
    <loop_body>
END LOOP;
```

משתנה ה- <var> הוא לוקאלי ללואלה ולא צריך להיות מוגדר.

<start> ו-<finish> הינם קבועים.

- לדוגמה:

```
BEGIN
   FOR counter IN 1 .. 10 LOOP
         DBMS_OUTPUT.put_line(counter);
   END LOOP;
END;
.
Run;
```

Cursor - PL/SQL

- Cursor הוא משתנה שרץ על שורות (tuples) ב-relation.
- ה- relation יכול להיות טבלה או אפילו פלט של תשאול מבט.
- אנו מעבירים אל ה- Cursor כל שורה ויכולים לעבד את הערכים בשורה זו.
- אם ה-relation מאוחסן פיזית, נוכל גם לעדכן או למחוק שורה שעליה עומד ה-Cursor.
- Cursor משמשים בדרך כלל כדי לאחסן נתונים שאילתות שהחזירו יותר מאשר ערך יחיד.

Cursor - PL/SQL

- **מאפייני הסמן:**
  - **%ISOPEN** - מחזיר true אם הסמן פתוח, אחרת, מחזיר false.
  - **%FOUND** - מחזיר true אם הרשומה יובאה בהצלחה, false אם לא.
  - **%NOTFOUND** - מחזיר trueאם הרשומה לא יובאה בהצלחה, falseאם כן.
  - **%ROWCOUNT** - מחזיר את מספר הרשומות הנכללות בסמן.

Cursor - PL/SQL

- **תפעול הסמן**

תפעול הסמן נעשה ע"י שלוש פקודות פשוטות:

- **Open** - פותחת את הסמן ע"י שאילתה מסויימת.
- **Close** - סוגרת את הסמן.
- **Fetch** - מיבא את ערכי הרשומה הנוכחית לתוך משתנים.

Cursor - PL/SQL

- מבנה ה- Cursor:

```
DECLARE
   CURSOR <cursor_name> IS <SQL_Statment>;
   -- variables declaration
BEGIN
   OPEN <cursor_name>;
    LOOP
          FETCH <cursor_name> INTO <var1>,<var2>...
          EXIT WHEN cursor_name%NOTFOUND;
          //using data

   END LOOP;
   CLOSE <cursor_name>
END;
```

Cursor - PL/SQL

- דוגמא:
  - תוכנית המדפיסה את כל פרטי העובדים (מספר עובד, שם, משכורת).

```
DECLARE
   CURSOR c1 IS SELECT * FROM emp;
   e_rec emp%rowtype;
BEGIN
   OPEN c1;
   LOOP
      FETCH c1 INTO e_rec;
      EXIT WHEN c1%NOTFOUND;
      DBMS_OUTPUT.PUT_LINE('Number: ' || ' ' || e_rec.empno);
      DBMS_OUTPUT.PUT_LINE('Name : ' || ' ' || e_rec.ename);
      DBMS_OUTPUT.PUT_LINE('Salary: ' || ' ' || e_rec.sal);

   END LOOP;
   CLOSE c1;
END;
.
Run;
```

Cursor - PL/SQL

- דוגמא:
  - תוכנית המדפיסה את פרטי 10 העובדים הראשונים (מספר עובד, שם, משכורת).

```
DECLARE
   CURSOR c1 IS SELECT * FROM emp;
   e_rec emp%rowtype;
BEGIN
   OPEN c1;
   LOOP
      FETCH c1 INTO e_rec;
       EXIT WHEN c1%ROWCOUNT >= 10;
      DBMS_OUTPUT.PUT_LINE('Number: ' || ' ' || e_rec.empno);
      DBMS_OUTPUT.PUT_LINE('Name : ' || ' ' || e_rec.ename);
      DBMS_OUTPUT.PUT_LINE('Salary: ' || ' ' || e_rec.sal);

   END LOOP;
   CLOSE c1;
END;
```