GALGOTIAS UNIVERSITY

School of Computing Science and Engineering

Program: BCA - IOP

Course Code: BCAS3031

Course Name: PL/SQL & Cursors and Triggers

Dr. T. Poongodi

Associate Professor

Initialize Variables

It is a good programming practice to initialize variables properly otherwise, sometimes programs would produce unexpected results.

```
DECLARE
   a integer := 10;
   b integer := 20;
   c integer;
   f real;
BEGIN
   c := a + b;
   dbms_output.put_line('Value of c: ' || c);
   f := 70.0/3.0;
   dbms_output.put_line('Value of f: ' || f);
END;
/
```

```
Value of c: 30
Value of f: 23.3333333333333333

PL/SQL procedure successfully completed.
```

Variable Scope in PL/SQL

- PL/SQL allows the **nesting of blocks**, i.e., each program block may contain another inner block.
- If a variable is declared within an inner block, it is not accessible to the outer block.
- However, if a variable is declared and accessible to an outer block, it is also accessible to all nested inner blocks. There are two types of variable scope –
- **Local variables** – Variables declared in an inner block and not accessible to outer blocks.
- **Global variables** – Variables declared in the outermost block or a package.

```
DECLARE
  -- Global variables
  num1 number := 95;
  num2 number := 85;
BEGIN
  dbms_output.put_line('Outer Variable num1: ' || num1);
  dbms_output.put_line('Outer Variable num2: ' || num2);
  DECLARE
    -- Local variables
    num1 number := 195;
    num2 number := 185;
  BEGIN
    dbms_output.put_line('Inner Variable num1: ' || num1);
    dbms_output.put_line('Inner Variable num2: ' || num2);
  END;
END;
/
```

Outer Variable num1: 95

Outer Variable num2: 85

Inner Variable num1: 195

Inner Variable num2: 185

PL/SQL procedure successfully completed.

Assigning SQL Query Results to PL/SQL Variables

- Use the **SELECT INTO** statement of SQL to assign values to PL/SQL variables.
- For each item in the **SELECT list**, there must be a corresponding, type-compatible variable in the **INTO list**.

Create a table named CUSTOMERS,

```
CREATE TABLE CUSTOMERS(
   ID   INT NOT NULL,
   NAME VARCHAR (20) NOT NULL,
   AGE INT NOT NULL,
   ADDRESS CHAR (25),
   SALARY   DECIMAL (18, 2),
   PRIMARY KEY (ID)
);
```

Table Created

Insert some values in the table,

```
INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (1, 'Ramesh', 32, 'Ahmedabad', 2000.00 );

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (2, 'Khilan', 25, 'Delhi', 1500.00 );

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (3, 'kaushik', 23, 'Kota', 2000.00 );

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (4, 'Chaitali', 25, 'Mumbai', 6500.00 );

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (5, 'Hardik', 27, 'Bhopal', 8500.00 );

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (6, 'Komal', 22, 'MP', 4500.00 );
```

Assigns values from the above table to PL/SQL variables using the **SELECT INTO clause** of SQL,

```
DECLARE
  c_id customers.id%type := 1;
  c_name  customers.name%type;
  c_addr customers.address%type;
  c_sal  customers.salary%type;
BEGIN
  SELECT name, address, salary INTO c_name, c_addr, c_sal
  FROM customers
  WHERE id = c_id;
  dbms_output.put_line
  ('Customer ' ||c_name || ' from ' || c_addr || ' earns ' || c_sal);
END;
/
```

Customer Ramesh from Ahmedabad earns 2000

PL/SQL procedure completed successfully

Declaring a Constant

A constant is declared using the **CONSTANT** keyword. It requires an initial value and does not allow that value to be changed.

```
PI CONSTANT NUMBER := 3.141592654;
DECLARE
                -- constant declaration
               pi constant number := 3.141592654;
               -- other declarations
                radius number(5,2);
                dia number(5,2);
                circumference number(7, 2);
                area number (10, 2);
BEGIN
                -- processing
               radius := 9.5;
                dia := radius * 2;
                circumference := 2.0 * pi * radius;
               area := pi * radius * radius;
                -- output
               dbms_output.put_line('Radius: ' || radius);
                dbms_output.put_line('Diameter: ' || dia);
                dbms_output.put_line('Circumference: ' || circumference);
                dbms_output.put_line('Area: ' || area);
END;
/
```

Radius: 9.5

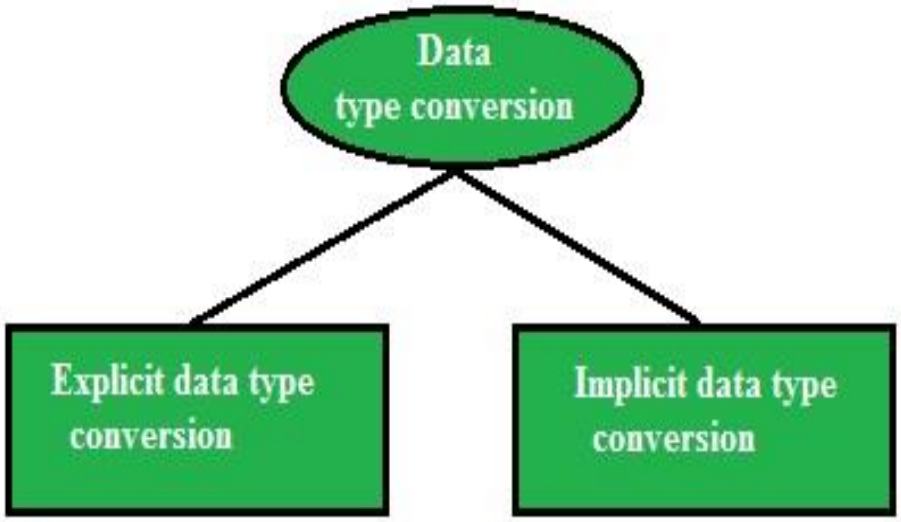
Diameter: 19

Circumference: 59.69

Area: 283.53

Pl/SQL procedure successfully completed.

Data type conversion



- Server uses data of one type where it expects data of a different data type.
- Server can automatically convert the data to the expected data type.
- This data type conversion can be done implicitly by the Server, or explicitly by the user.

Implicit Data-Type Conversion :

Data is converted from one type to another implicitly (by itself/automatically)

FROM	TO
VARCHAR2 or CHAR	NUMBER
VARCHAR2 or CHAR	DATE
DATE	VARCHAR2
NUMBER	VARCHAR2

```
SELECT employee_id,first_name,salary FROM employees
WHERE salary > 15000;
```

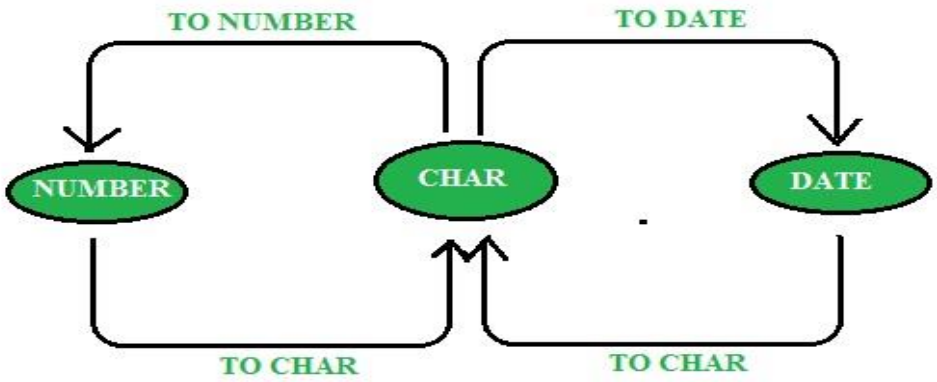
EMPLOYEE_ID	FIRST_NAME	SALARY
100	Steven	24000
101	Neena	17000
102	lex	17000

```
SELECT employee_id,first_name,salary FROM employees
WHERE salary > '15000';
```

EMPLOYEE_ID	FIRST_NAME	SALARY
100	Steven	24000
101	Neena	17000
102	lex	17000

Output of both queries came out to be same, inspite of 2nd query using **‘15000’** as text, it is automatically converted into **int** data type.

Explicit Data-Type Conversion :



Explicit Data Type Converson

TO_CHAR Function :

TO_CHAR function is used to typecast a numeric or date input to character type with a format model (optional).

- Must be enclosed in single quotation marks and is case sensitive
- Can include any valid date format element

```
TO_CHAR(date, 'format_model')
```

```
SELECT employee_id, TO_CHAR(hire_date, 'MM/YY')
Month_Hired FROM employees
WHERE last_name = 'Higgins';
```

EMPLOYEE_ID	MONTH_HIRED
205	06/94

TO_CHAR Function with Numbers :

Use with the TO_CHAR function to display a number value as a character.

```
TO_CHAR(number, 'format_model')
```

```
SELECT TO_CHAR(salary, '$99,999.00') SALARY FROM employees
WHERE last_name = 'Ernst';
```

SALARY
$5000

Using the TO_NUMBER and TO_DATE Functions :

Convert a character string to a number format using the **TO_NUMBER** function :

```
TO_NUMBER(char[, 'format_model'])
```

Convert a character string to a date format using the **TO_DATE** function:

```
TO_DATE(char[, 'format_model'])
```

These functions have an **fx** modifier. This modifier specifies the exact matching for the character argument and date format model of a **TO_DATE** function.

```
SELECT last_name, hire_date FROM employees WHERE hire_date =
TO_DATE('May 24, 1999', 'fxMonth DD, YYYY');
```

LASTNAME	HIREDATE
Kumar	24-MAY-99


Thank You