

2.2.VARIABLES, TIPUS DE DADES I OPERADORS

Contenido

1.	OBJECTIUS	2
2.	DECLARACIÓ DE VARIABLES.....	2
	Secció DECLARE.....	2
	SINTAXI.....	2
3.	TIPUS DE DADES	3
	TIPUS DE DADES ESCALARS	3
	CARÀCTERS (mateixos tipus que SQL)	3
	NUMÈRIC.....	5
	DATE.....	6
	BOOLEAN.....	7
4.	DECLARAR VARIABLES AMB L'ATRIBUT %TYPE.....	8
5.	ASSIGNACIÓ DE VALORS A VARIABLES.....	9
	FUNCIONS DE CARÀCTERS	10
	FUNCIONS NUMÈRIQUES.....	11
	FUNCIONS DE DATA	11
	CONVERSIONS EXPLÍCITES DE TIPUS.....	12
6.	OPERADORS.....	13
	PRECEDÈNCIA D'OPERADORS.....	13
7.	NAMING CONVENCIONS.....	13

1. OBJECTIUS

- ✓ Declaració i inicialització de variables
- ✓ Assignació de valors
- ✓ Assignar valors d'una consulta a una variable

2. DECLARACIÓ DE VARIABLES

Secció DECLARE

- ✓ Secció on es declaren les variables que s'utilitzen en la secció executable del bloc, entre BEGIN i END.
- ✓ Les variables es poden utilitzar en qualsevol sentència del bloc. També a les sentències SQL.

SINTAXI

```
nom_variable [CONSTANT] tipus_de_Dades [NOT NULL][:= expr | DEFAULT expr]
```

- ✓ Només el nom de la variable i el tipus de dades són obligatoris.
- ✓ Per conveni, el nom de les **variables** comença amb **v_**
- ✓ Per conveni, el nom de les variables que emmagatzemen valors **constants** comença amb **c_**
- ✓ Les constants han de tenir valor en la seva declaració.
- ✓ Els **tipus de dades** de les variables són els tipus de dades d'Oracle.
- ✓ **NOT NULL** força a que la variable tingui un valor en la seva declaració. Haurà d'inicialitzar-se amb DEFAULT o amb :=
- ✓ := :operador d'assignació.
- ✓ Es pot donar valor inicial a la variable en la seva declaració amb := o amb DEFAULT.
- ✓ Una variable **no inicialitzada** té el **valor NULL**.
- ✓ Una variable en cada declaració !

3. TIPUS DE DADES

PL/SQL suporta **5 categories** de tipus de dades:

CATEGORIA	DESCRIPCIÓ
SCALARS	Emmagatzemen un valor simple
COMPOSTOS (composite)	Tipus format per múltiples elements interns que poden manipulats individualment.
Large Object (LOB)	Emmagatzema valors anomenats locutors, que indiquen la "localització" d'objectes grans (ie. Imatges gràfiques) emmagatzemades out of line.
REFERÈNCIES (references)	Emmagatzemen valors anomenats pointers, que apunten a una <i>storage location</i> .
OBJECTES	Similar als objectes d'altres llenguatges (mètodes + atributs)

TIPUS DE DADES ESCALARS

Emmagatzemen un valor simple. Es classifiquen en 5 categories:

- Caràcters
- Números
- Dates
- booleans.

NOTA: El tipus booleà no existeix en SQL.

CARÀCTERS (mateixos tipus que SQL)
CHAR [(L)]
Strings de longitud fixa . Fins a 32.767. Si no s'indica longitud, per defecte és 1.
VARCHAR2 (L) – SINÒNIM: VARCHAR(L)
Strings de longitud variable . Fins a 32.767.
LONG(L)
Strings de longitud variable fins a 2 Gg.

EXEMPLE

Declaració de variables de tipus caràcter

```
DECLARE
    v_country_id      CHAR(2);
    v_country_name     VARCHAR(70);
    v_country_climate LONG;
```

EXEMPLE

Declaració i assignació de valor a una variable “string”. Els valors string entre cometes simples.

```
DECLARE
    v_myname VARCHAR2(20);
BEGIN
    DBMS_OUTPUT.PUT_LINE('My name is ' || v_myname);
    v_myname := 'Jhon';
    DBMS_OUTPUT.PUT_LINE('My name is ' || v_myname);
END;
```

```
My name is
My name is Jhon

Sentencia procesada.
```

Una variable no inicialitzada té el valor NULL.

Es “converteix” a l’string buit per escriure el valor.

NUMÈRIC
NUMBER [(P,E)]
El mateix tipus representa tant números enters com reals.
NUMBER
Números reals en coma flotant
NUMBER (n)
Números enters de com a màxim n dígit
NUMBER(n,p)
Números reals en punt fix de n dígit dels quals p indiquen la precisió (part decimal)
BINARY_INTEGER
S'emmagatzema en memòria en format binari per facilitar els càlculs. S'utilitza en comptadors, índexs, etc. Existeixen els subtipus: NATURAL i POSITIVE.
PLS_INTEGER
Similar a BINARY_INTEGER però més ràpid i si es dóna desbordament en el càlcul es genera un error i es llança una excepció

Té subtipus per **compatibilitat** amb SQL i/o per establir restriccions: **DECIMAL**, **NUMERIC**, **INTEGER**, **REAL**, **SMALLINT**, etc.

EXEMPLE

Declaració de variables numèriques.

```
v_salary NUMBER(8,2) := 99.99;
v_count  INTEGER := 0;
```

EXEMPLE

Declaració i inicialització d'una variable entera.

```
DECLARE
    v_counter INTEGER := 0;
BEGIN
    v_counter := v_counter + 1;
    DBMS_OUTPUT.PUT_LINE('Comptador: ' || v_counter);
END;
```

```
Comptador: 1
```

```
Statement processed.
```

DATE
DATE
Emmagatzema dates. El format estàndard és ' mm-dd-aaaa '. També emmagatzema l'hora .
TIMESTAMP
Data i hora
TIMESTAMP WITH TIME ZONE

EXEMPLE

Declaració i inicialització d'una variable data.

```
DECLARE
  v_hire_date DATE := '03/15/2022';
  v_date1      TIMESTAMP := SYSDATE;
BEGIN
  DBMS_OUTPUT.PUT_LINE(v_hire_date);
  DBMS_OUTPUT.PUT_LINE(v_date1);
END;
```

```
03/15/2022
08-MAR-22 01.08.20.000000 PM

Statement processed.
```

Provar: assignar a la data el valor '15/15/2022' Funciona ?

BOOLEAN

Emmagatzema valors **TRUE**, **FALSE** i **NULL**.

- Només és vàlid en PL/SQL, però **no en SQL**.
- No es pot mostrar una variable booleana amb DBMS_OUTPUT.PUT_LINE !! Dona error de conversió de tipus

EXEMPLE

Declaració i inicialització d'una variable booleana.

```
DECLARE  
  v_valid1 BOOLEAN := TRUE;
```

EXEMPLE

Declaració de variables i restriccions de varis tipus

```
DECLARE  
  v_birthdate DATE;  
  v_size      NUMBER(2) NOT NULL:= 10;  
  v_location  VARCHAR2(13):= 'Florida';  
  c_bank      CONSTANT NUMBER:= 50000;  
  v_population INTEGER;  
  v_party_size CONSTANT PLS_INTEGER:= 20;  
  v_book_type VARCHAR2(20) DEFAULT 'fiction';
```

4.DECLARAR VARIABLES AMB L'ATRIBUT %TYPE

Utilitzat per declarar una variable amb el **mateix tipus** que un camp d'una taula, o que una altra variable.

Al declarar una variable del mateix tipus que altre objecte amb %TYPE, s'hereta el **tipus** i la **longitud**, però no l'atribut NOT NULL ni el valor per defecte que tingués l'objecte original.

EXEMPLE

Declara variable **v_last_name** per a que contingui valors del camp **LAST_NAME** de la taula EMPLOYEES:

```
DECLARE
  v_last_name VARCHAR2(25); -- Mateix tipus que LAST_NAME
o
  v_last_name EMPLOYEES.LAST_NAME%TYPE;
```

El tipus de la variable es manté **vinculat** al tipus del camp al que està associat.

Si es canvia la definició de LAST_NAME a la taula EMPLOYEES, per exemple VARCHAR2(40), quan s'executi de nou el bloc de codi es declararà amb el mateix tipus que el camp LAST_NAME de la taula EMPLOYEES.

També es pot utilitzar el %TYPE per definir una variable amb el mateix tipus que una altra variable.

EXEMPLE

```
DECLARE
  v_salary      EMPLOYEES.SALARY%TYPE;
  v_old_salary v_salary%TYPE;
  v_new_salary v_salary%TYPE;
```


5.ASSIGNACIÓ DE VALORS A VARIABLES

A una variable se li poden assignar **valors literals** a valors retornats per una **funció**.

Moltes de les funcions d'SQL són vàlides en PL/SQL.

Les **funcions de grup** (AVG, MIN, MAX,...) **no es poden utilitzar** en PL/SQL, només en SQL.

La resta de funcions SQL sí.

EXEMPLE: Assignar valor a una variable amb una funció SQL.

La funció **LAST_DAY(data)** retorna el darrer dia del més de la data que se li passa com a paràmetre.

SQL

```
SELECT LAST_DAY(SYSDATE)
FROM DUAL
```

LAST_DAY(SYSDATE)
03/31/2022

PL-SQL

```
DECLARE
    V_LAST_DAY DATE;
BEGIN
    V_LAST_DAY := LAST_DAY(SYSDATE);
    DBMS_OUTPUT.PUT_LINE(V_LAST_DAY);
END;
```

03/31/2022
Statement processed.

FUNCIONS DE CARÀCTERS

ASCII	LENGTH	RPAD
CHR	LOWER	RTRIM
CONCAT	LPAD	SUBSTR
INITCAP	LTRIM	TRIM
INSTR	REPLACE	UPPER

EXEMPLE

```
DECLARE
  v_nom VARCHAR2(20) := 'Mortadelo';
  -- Assigna el nombre de lletres del nom
  v_numLletres INTEGER := LENGTH(v_nom);
BEGIN
  DBMS_OUTPUT.PUT_LINE('Nom: ' || v_nom);
  -- Converteix l'string a majúscules
  v_nom := UPPER(v_nom);
  DBMS_OUTPUT.PUT_LINE('Nom: ' || v_nom);
  DBMS_OUTPUT.PUT_LINE('Nom té ' || v_numlletres || ' lletres');
END;
```

```
Nom: Mortadelo
Nom: MORTADELO
Nom té 9 lletres

Sentencia procesada.
```

FUNCIONS NUMÈRIQUES

ABS	EXP	ROUND
ACOS	LN	SIGN
ASIN	LOG	SIN
ATAN	MOD	TAN
COS	POWER	TRUNC

EXEMPLE

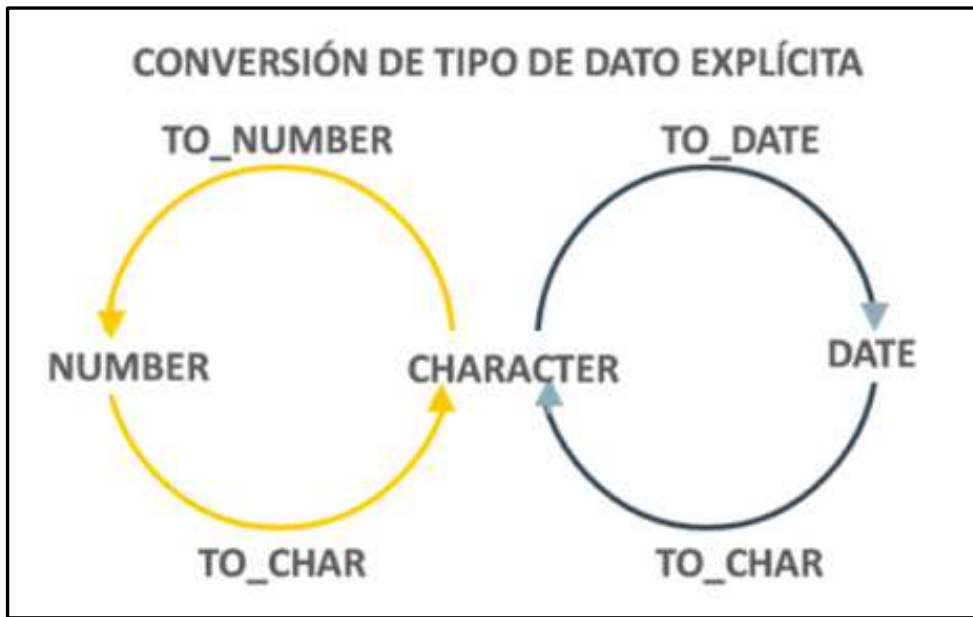
Obté la mitjana de la comissió del employees dels empleats amb dos decimals:

```
DECLARE
  v_comission NUMBER(2,2);
BEGIN
  -- Guarda el resultat d'una consulta en una variable
  SELECT AVG(COMMISSION_PCT) INTO v_comission
  FROM EMPLOYEES;
  DBMS_OUTPUT.PUT_LINE('Promig comissió: ' || ROUND(v_comission, 2));
  -- ROUND redundant pq v_comission ja té 2 dígit de precisió
END;
```

FUNCIONS DE DATA

ADD_MONTHS	MONTHS_BETWEEN
CURRENT_DATE	ROUND
CURRENT_TIMESTAMP	SYSDATE
LAST_DAY	TRUNC

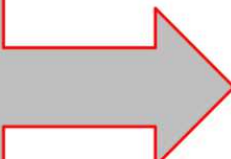
CONVERSIONS EXPLÍCITAS DE TIPUS



TO_NUMBER ()	ROWIDTONCHAR ()
TO_CHAR ()	HEXTORAW ()
TO_CLOB ()	RAWTOHEX ()
CHARTOROWID ()	RAWTONHEX ()
ROWIDTOCHAR ()	TO_DATE ()

6. OPERADORS

- Logical
- Arithmetic
- Concatenation
- Parentheses to control the order of operations
- Exponential operator (**)



Same as
in SQL

PRECEDÈNCIA D'OPERADORS

Operator	Operation
**	Exponentiation
+, -	Identity, negation
*, /	Multiplication, division
+, -,	Addition, subtraction, concatenation
=, <, >, <=, >=, <>, !=, ~=, ^=, IS NULL, LIKE, BETWEEN, IN	Comparison
NOT	Logical negation
AND	Conjunction
OR	Inclusion

7. NAMING CONVENCTIONS

Category	Case Convention	Examples
SQL keywords	Uppercase	SELECT, INSERT
PL/SQL keywords	Uppercase	DECLARE, BEGIN, IF
Data types	Uppercase	VARCHAR2, BOOLEAN
Identifiers (variables, etc.)	Lowercase	v_salary, emp_cursor, c_tax_rate, p_empno
Tables and columns	Lowercase	employees, dept_id, salary, hire_date