M2: PL/pgSQL

Sintaxis

- Les paraules reservades com `SELECT, FROM`, etc. en majúscules.
- Les taules i camps en minúscules
- Els tipus de dades en majúscules.

- Les variables de bloc declarades per el programador en minúscules i comencen per "**var_**".
- Les variables fora de bloc per demanar valors a l'usuari en minúscules i comencen per "**v**"
- El nom de les variables que formen part d'una variable tipus `type` creada per el programados en minúscules i comencen per "**vr_**".

- Els paràmetres de les funcions i procediments en mínúsucules i comencen per “**par_**".

- El nom de les funcions en minúscules i comencen per "**func_**".
- El nom del procediments en minúscules i comencen per "**proc_**".
- El nom de les variables tipus registre en minúscules i comencen per "**reg_**".

TIPUS VARIABLES SIMPLES

PSQL

- **Bàsiques**: `NUMERIC, INTEGER, VARCHAR, CHAR, DATE`
- **Tipus específic:** NOMTAULA.NOMCOLUMNA%TYPE

```
//"bàsiques"
var_total NUMERIC;

//"tipus de la columna"
var_salari EMPLOYEES.SALARY%TYPE;
```

NO PL/SQL

`:v_nomVariable`

Són les que demanen les dades per teclat.

```
//Exemple
var_nom NUMERIC = :v_nom;
```

💡 !!!! MOLT IMPORTANT: quan introdueixis les dades per teclat, recorda posar cometes simples `''` si és text.

TIPUS COMPLEX

`ROWTYPE, RECORD i TYPE`

Hi ha variables que ens permeten guardar més d'una dada.

ROWTYPE

És com una variable on guardo tots els valors d'una sola fila. S'adapta a la taula.

- **Declaració** `TAULA%ROWTYPE`

```
DECLARE
   var_emps EMPLOYEES%ROWTYPE;
```

- **Insertar i accedir** `.`

Per accedir als valors del ROWTYPE ho fem com amb els objectes, amb el punt `.` .

```
//...
  BEGIN
     SELECT *
     INTO var_emps
     FROM employees
     WHERE e.employee_id = '101';

           RAISE NOTICE '%',var_emps.first_name;
  END;

$$LANGUAGE plpgsql;
```

!!! Si no fas un select `*`, llavors en el `INTO` has d'indicar els camps (si no se't guardarà en la columna següent, i no la que li pertoca):

```
//...
  BEGIN
     SELECT first_name, edat
     INTO var_emps.first_name, var_emps.edat
     FROM employees
     WHERE e.employee_id = '101';

          RAISE NOTICE '%',var_emps.first_name;
  END;

$$LANGUAGE plpgsql;
```

RECORD

Com el ROWTYPE, pero dintre hi puc guardar els valors que jo vulgui i en l'ordre que defineixi. S'adapta al SELECT.

- **Definició**

`RECORD`

```
//...
DECLARE
   rec_empleado RECORD;

  //...
```

- **Manipular**

```
//...
BEGIN
   SELECT employee_id, first_name, last_name
   INTO rec_empleado
   FROM employees
   WHERE employee_id = 101;

   RAISE NOTICE '% % %', rec_empleado.employee_id, rec_empleado.first_nar

//...
```

TYPE

Tipus de dada complex personalitzat (semblant a una dataclass).

- **Creació**

```
CREATE TYPE dades_empl_type AS (
    vr_nom_empl VARCHAR(20),
    vr_cognom_empl VARCHAR(25)
);
```

!!!! De tipus li poses el type de la columna (l'has de consultar a la base de dades)

- **Declaració**

```
//...
    DECLARE
        var_dades_empl DADES_EMPL_TYPE;
//...
```

- **Insertar i accedir**

```
//...
  BEGIN
     SELECT first_name, last_name
     INTO var_dades_empl
     FROM employees
     WHERE e.employee_id = '101';

          RAISE NOTICE '% %',var_dades_empl.vr_nom_empl, var_dades_emp
  END;

$$LANGUAGE plpgsql;
```

BLOC ANÒNIM

```
DO $$
DECLARE

BEGIN

END;
$$ LANGUAGE plpgsql;
```

▼ Exemple

```
DO $$
   DECLARE
   var_nom NUMERIC = :v_nom;
   BEGIN
      RAISE NOTICE 'Hola %.' , var_nom;
```

```
    END;
$$ LANGUAGE plpgsql;
```

PROCEDIMENTS

No retornen res:

```
CREATE OR REPLACE PROCEDURE proc_nom (par_1 %TYPE, par_2 %TYPE) A
    $$
    DECLARE

    BEGIN

    END;
$$ LANGUAGE plpgsql;
```

▼ Exemple

```
CREATE OR REPLACE PROCEDURE proc_dades_empl (par_empl_id EMPLO
$$
    DECLARE
        var_dades_empl DADES_EMPL_TYPE;
    BEGIN
        SELECT e.first_name, e.last_name, d.department_id, d.location_id
        INTO var_dades_empl
        FROM employees e
        JOIN departments d USING (department_id)
        WHERE e.employee_id = par_empl_id;

        RAISE NOTICE '% % % %', var_dades_empl.vr_nom_empl, var_dades_
    END;

$$LANGUAGE plpgsql;
```

FUNCIONS

```
CREATE OR REPLACE FUNCTION func_nomFuncioProc (par_1 %TYPE, par_2 %
   RETURNS %TYPE AS
   $$
   DECLARE

   BEGIN

      RETURN;
   END;
$$ LANGUAGE plpgsql;
```

▼ Exemple

```
CREATE OR REPLACE FUNCTION func_nom_manager (par_deparment_id
   RETURNS employees.first_name%TYPE AS
   $$
   DECLARE
      var_manager_name employees.first_name%TYPE;
   BEGIN
      SELECT first_name
      INTO var_manager_name
      FROM employees
      WHERE employee_id = (SELECT manager_id
                  FROM departments
                  WHERE departments.department_id = par_deparment_id

      RETURN var_manager_name;
   END;
$$ language plpgsql;
```

ESTRUCTURES DE CONTROL DE FLUX

IF-ELSE

```
IF condició THEN
   --instruccio
ELSE
   --instruccio
END IF;
```

IF-ELSIF-ELSE

```
IF condicio1  THEN
   --instruccio
ELSIF condicio2  THEN
   --instruccio
ELSE
   --instruccio
END IF;
```

!!!! Fixa't que és `ELSIF`

SWITCH-CASE

- Amb **valor**

```
CASE (variable)
   WHEN 'valor1' THEN
      --instruccio
   WHEN 'valor2' THEN
      --instruccio
   WHEN 'valor3' THEN
     --instruccio
   WHEN 'valor4' THEN
```

```
        --instruccio
    ELSE
        --instruccio
END CASE;
```

- Amb **condicions**

```
  CASE
    WHEN var_edat BETWEEN 0 AND 17 THEN
      RAISE NOTICE 'Ets menor d"edat!';
    WHEN var_edat BETWEEN 18 AND 40 THEN
      RAISE NOTICE 'Ja ets major d`edat!';
    WHEN var_edat > 40 THEN
      RAISE NOTICE 'Ja ets força gran!';
    WHEN var_edat < 0 THEN
      RAISE NOTICE 'Error! L´edat no pot ser negativa';
    ELSE
      RAISE NOTICE 'L edat no encaixa amb cap condició';
  END CASE;
```

FOR

```
FOR i IN 5..10 LOOP
  raise notice 'el valor de i és % ', i;
END LOOP;

//reverse
FOR i IN REVERSE 5..10 LOOP
  raise notice 'el valor de i és %',i;
END LOOP;
```

WHILE

```
WHILE condicio LOOP
   --instrucció
END LOOP;
```

▼ Exemple

```
DECLARE
   i NUMBER:=0;
BEGIN
   WHILE <=5 LOOP
      raise notice 'HOLA';
      i=i+1;
   END LOOP;
END;
```

DO-WHILE

```
LOOP
   --instrucció
IF condicio THEN
   EXIT;
END IF;
   --instrucció
END LOOP;
```

▼ Exemple

```
DECLARE
   i NUMBER:=2;
BEGIN
   LOOP
      raise notice 'HOLA';
      IF i>=5 THEN
```

```
            EXIT;
        END IF;
        i=i+1;
    END LOOP;
END;
```