Anchored declarations y Composite types

Anchored declarations

Anchored declarations

Las Anchored Declarations son declaraciones de variables especiales en las que se "asocia" la declaración de la variable **a un objeto de la base de datos**.

Cuando asociamos de esta manera un tipo de dato estamos indicando que queremos establecer como tipo de dato de nuestra variable **al tipo de dato de una estructura de datos previamente definida**, que puede ser una **tabla** de la base de datos, una **columna** de una tabla, o un TYPE predefinido.

T-SQL no soporta anchored declarations.

Scalar anchoring

Scalar anchoring

Un scalar anchoring define una variable en base a una columna de una tabla.
Se especifican a través del atributo `%TYPE`.

Sentencias SELECT Single-row y scalar anchoring

Recuperar valores escalares usando scalar anchoring

```
DECLARE
   price1 titles.price%TYPE;
   type1 titles.type%TYPE;

BEGIN
   SELECT price, type INTO price1, type1
      FROM titles
      WHERE title_id = 'BU1032';

   DBMS_OUTPUT.PUT_LINE('El precio es ' || price1);
END;
```

Record anchoring
(tipo de dato para la tupla completa)

Record anchoring
Table-based record

Un record anchoring define una variable –que poseerá una estructura de record- en base al schema de una tupla de una tabla.
Cada campo corresponde a –y tiene el mismo nombre que- una columna en la tabla.

Se especifican anexando al nombre de la tabla el atributo `%ROWTYPE`. Por ejemplo:

```
DECLARE
   titleRec titles%ROWTYPE;
```

Luego de recuperados los datos, podemos acceder a cada uno de los componentes de la estructura de record usando una sintaxis de punto. Por ejemplo:

```
titleRec.price
```

Sentencias SELECT Single-row y record anchoring

Recuperar la tupla completa usando record anchoring

ORACLE

```
DECLARE
   titleRec titles%ROWTYPE;

BEGIN
   SELECT * INTO titleRec
      FROM titles
      WHERE title_id = 'BU1032';

   DBMS_OUTPUT.PUT_LINE('El precio es ' || titleRec.price);
END;
```

Cuando la variable es un record asociado a una tupla completa, en la sentencia **SELECT** solo podemos recuperar *la totalidad de las columnas de la tupla*. No podemos recuperar una projection de tuplas.

Records no anclados

Record no anclado

Llamamos record no anclado a una estructura de record que, en contraposición a un record anchoring –por ejemplo, `titleRec titles%ROWTYPE`- no está basado en el schema de una tupla de una tabla.

En PL/pgSQL podemos definir una estructura de record para un set de algunas columnas o para expresiones calculadas. PostgreSQL llama a esto un **composite-type.**
Se crean con la sentencia `CREATE TYPE`. Luego de la cláusula `AS`, definimos la estructura del type de manera similar a como hacemos con la definición de una tabla:

Por ejemplo:

```
CREATE TYPE publisherCT
   AS (
       pub_id CHAR(4),
       totalPrice numeric
      );
```

SPs que retornan sets de proyecciones de tuplas

Si se necesita obtener un set de algunas columnas de un set de tuplas podemos definir un **composite-type** con la estructura deseada. Por ejemplo:

```
CREATE TYPE publisherCT
   AS (
       pub_id CHAR(4),
       totalPrice numeric
      );
```

Luego nuestra function es definida para retornar este composite-type. En el cuerpo del procedimiento también utilizamos `RETURN QUERY`:

```
CREATE OR REPLACE FUNCTION test502()
   RETURNS setof publisherCT
   LANGUAGE plpgsql
   AS
   $$
   DECLARE
   BEGIN

     RETURN QUERY
        SELECT pub_id, SUM(price) AS totalPrice
           FROM titles
           WHERE price IS NOT NULL
           GROUP BY pub_id;

   END
   $$;
```

```
SELECT  *
   FROM test502();
```

	pub_id character(4)	totalprice numeric
1	1389	80.58
2	0736	36.16
3	0877	51.73

SPs que retornan sets de records no anclados o composite types

También podemos retornar un composite type sin que el mismo sea necesariamente resultado de un query.

Definimos un **composite-type** con la estructura deseada. Por ejemplo:

```
CREATE TYPE publisherCT
   AS (
       pub_id CHAR(4),
       totalPrice numeric
      );
```

Y luego:

```
CREATE OR REPLACE FUNCTION test520()
   RETURNS setof publisherCT          (A)
   LANGUAGE plpgsql
   AS
   $$
   DECLARE
      fila publisherCT%rowtype;       (B)
   BEGIN
      fila.pub_id:='0736';
      fila.totalPrice := 240.00;
      RETURN NEXT fila;               (C)
   END
   $$;
```

Nuestra function es definida para retornar este composite-type (**A**).

Definimos una variable especial de tipo `<composite-type>%rowtype` (**B**).

En el cuerpo del procedimiento utilizamos una cláusula `RETURN NEXT <tupla-del-composite-type>` (**C**).