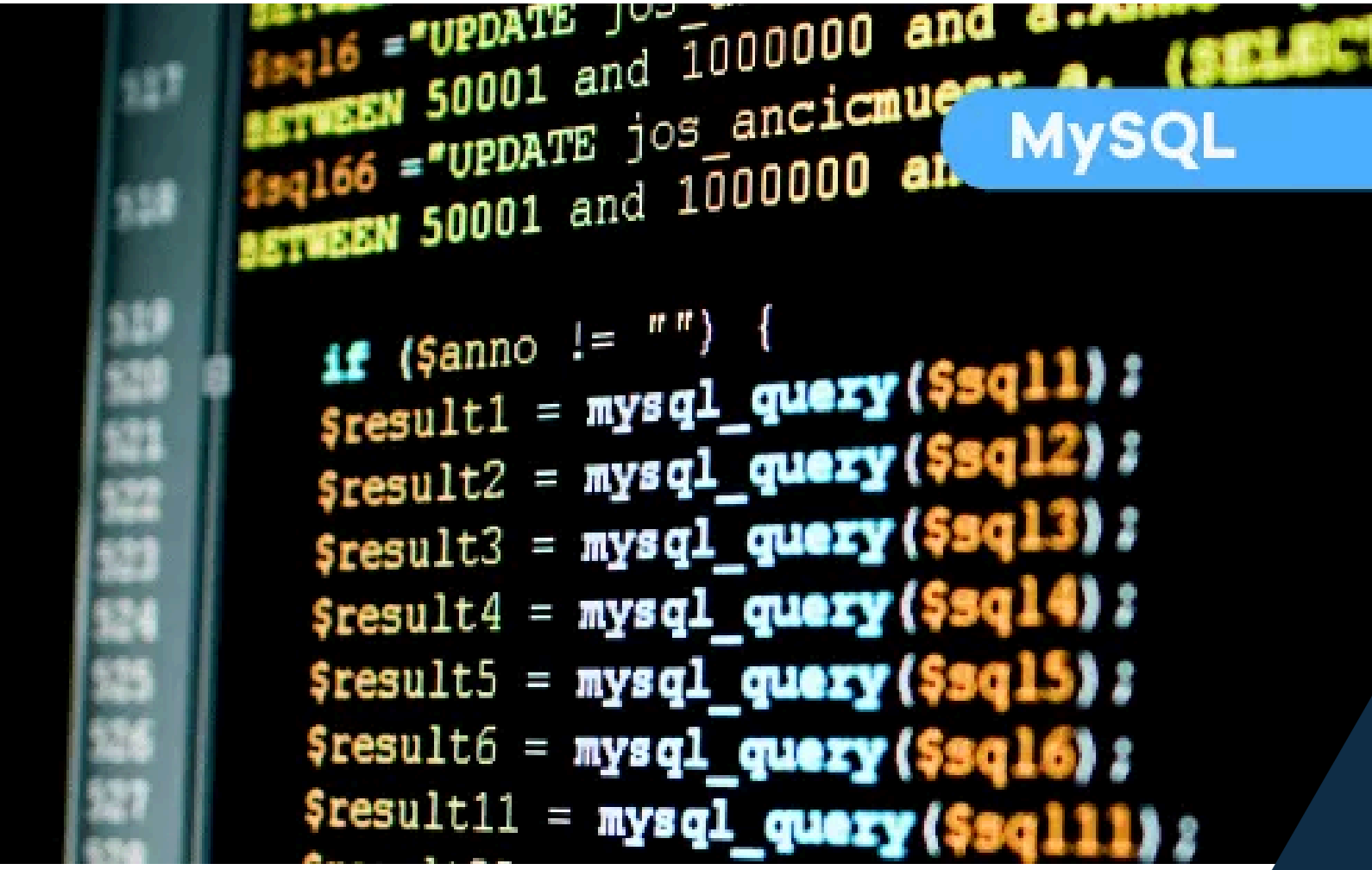

Tarea Tema 19 : Utilización de los objetos

Jurado Alonso, Diego 1 DAW

MEDAC
Instituto Oficial de Formación Profesional

Índice

Utilización de los objetos

1. Inicialización y Uso de Objetos:

Crea un tipo de objeto Curso con atributos nombre (VARCHAR), codigo_curso (INTEGER), y creditos (INTEGER).

Instancia este objeto, y luego muestra cómo cambiar el número de créditos del curso creado.

```
CREATE OR REPLACE TYPE Curso_Type AS OBJECT (
    nombre VARCHAR2(100),
    codigo_curso INTEGER,
    creditos INTEGER
);
/
```

```
Type created.
```

```
DECLARE
    mi_curso Curso_Type;
BEGIN
    mi_curso := Curso_Type('Programación Avanzada', 1001, 3);

    DBMS_OUTPUT.PUT_LINE('Nombre del Curso: ' || mi_curso.nombre);
    DBMS_OUTPUT.PUT_LINE('Código del Curso: ' ||
mi_curso.codigo_curso);
    DBMS_OUTPUT.PUT_LINE('Créditos del Curso: ' ||
mi_curso.creditos);

    mi_curso.creditos := 4;

    DBMS_OUTPUT.PUT_LINE('Créditos del Curso modificado: ' ||
mi_curso.creditos);
END;
/
```

```
Statement processed.
Nombre del Curso: Programación Avanzada
Código del Curso: 1001
Créditos del Curso: 3
Créditos del Curso modificado: 4
```

2. Manipulación de Atributos:

Dado un objeto Empleado con atributos id, nombre y salario, escribe código para: a. Obtener y mostrar el salario. b. Incrementar el salario en un 10%.

```
CREATE OR REPLACE TYPE Empleado_Type AS OBJECT (
    id INTEGER,
    nombre VARCHAR2(100),
    salario NUMBER
);
/
```

```
Type created.
```

```
DECLARE
    mi_empleado Empleado_Type;
BEGIN
    mi_empleado := Empleado_Type(1, 'Juan', 50000);

    -- a. Obtener y mostrar el salario
    DBMS_OUTPUT.PUT_LINE('Salario del Empleado: ' ||
mi_empleado.salario);

    -- b. Incrementar el salario en un 10%
    mi_empleado.salario := mi_empleado.salario * 1.1;
    DBMS_OUTPUT.PUT_LINE('Salario incrementado en un 10%: ' ||
mi_empleado.salario);
END;
/
```

```
Statement processed.
Salario del Empleado: 50000
Salario incrementado en un 10%: 55000
```

3. Uso de Herencia:

Define un tipo de objeto Persona con atributos nombre y apellido.

Crea un tipo Estudiante que herede de Persona y añade un atributo adicional matricula.

Muestra cómo inicializar un Estudiante.

```
CREATE OR REPLACE TYPE Persona_Type AS OBJECT (
    nombre VARCHAR2(100),
    apellido VARCHAR2(100)
) not final;
/

CREATE OR REPLACE TYPE Estudiante_Type UNDER Persona_Type (
    matricula VARCHAR2(20)
);
/
```

```
Type created.

Type created.
```

```
DECLARE
    estudiante Estudiante_Type;
BEGIN
    estudiante := Estudiante_Type('Luis', 'Heredia', '20240001');

    DBMS_OUTPUT.PUT_LINE('Nombre del Estudiante: ' ||
estudiante.nombre);
    DBMS_OUTPUT.PUT_LINE('Apellido del Estudiante: ' ||
estudiante.apellido);
    DBMS_OUTPUT.PUT_LINE('Matrícula del Estudiante: ' ||
estudiante.matricula);
END;
/
```

```
Statement processed.
Nombre del Estudiante: Luis
Apellido del Estudiante: Heredia
Matrícula del Estudiante: 20240001
```

4. Implementación del Método MAP:

Define un tipo de objeto Producto con atributos id_producto, nombre y precio.

Implementa un método MAP para comparar productos basándose en el precio.

```
CREATE OR REPLACE TYPE Producto_Type AS OBJECT (
    id_producto INTEGER,
    nombre VARCHAR2(100),
    precio NUMBER
);
/

CREATE OR REPLACE TYPE Producto_List AS TABLE OF Producto_Type;
/
```

```
Type created.

Type created.
```

```
DECLARE
    productos Producto_List := Producto_List(
        Producto_Type(1, 'Respuestas examen Quique', 20),
        Producto_Type(2, 'Respuestas examen Rafa', 30),
        Producto_Type(3, 'Respuestas examen Javi', 50)
    );
BEGIN
    FOR i IN 1..productos.COUNT LOOP
        DBMS_OUTPUT.PUT_LINE('Producto: ' || productos(i).nombre ||
', Precio: ' || productos(i).precio);
    END LOOP;
END;
/
```

```
Statement processed.
Producto: Respuestas examen Quique, Precio: 20
Producto: Respuestas examen Rafa, Precio: 30
Producto: Respuestas examen Javi, Precio: 50
```

5. Uso del Método ORDER:

Define un método ORDER para el objeto Producto que permita ordenar por nombre de producto.

```
CREATE OR REPLACE TYPE Producto AS OBJECT (
    id_producto INTEGER,
    nombre VARCHAR2(50),
    precio INTEGER,

    MEMBER FUNCTION MAP(p Producto) RETURN INTEGER,
    MEMBER FUNCTION ordenProductos(p1 IN Producto, p2 IN Producto)
RETURN INTEGER
);
```

```
Type created.
```

```
CREATE OR REPLACE TYPE BODY Producto AS
    MEMBER FUNCTION MAP(p Producto) RETURN INTEGER IS
    BEGIN
        IF self.precio < p.precio THEN
            RETURN -1;
        ELSIF self.precio > p.precio THEN
            RETURN 1;
        ELSE
            RETURN 0;
        END IF;
    END MAP;

    MEMBER FUNCTION ordenProductos(p1 IN Producto, p2 IN Producto)
RETURN INTEGER IS
    BEGIN
        RETURN CASE
            WHEN p1.nombre < p2.nombre THEN -1
            WHEN p1.nombre > p2.nombre THEN 1
            ELSE 0
        END;
    END ordenProductos;
END;
```

```
Type created.
```

```
CREATE TABLE productos_tabla (
    producto Producto
);
```

```
Table created.
```

```
INSERT INTO productos_tabla VALUES (Producto(1, 'Respuestas examen Quique', 20));

INSERT INTO productos_tabla VALUES (Producto(2, 'Respuestas examen Rafa', 30));

INSERT INTO productos_tabla VALUES (Producto(3, 'Respuestas examen Javi', 50));
```

```
1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.
```

```
SELECT p.producto.id_producto, p.producto.nombre, p.producto.precio
FROM productos_tabla p
ORDER BY p.producto.ordenProductos(p.producto, Producto(0, '', 0));
```

PRODUCTO.ID_PRODUCTO	PRODUCTO.NOMBRE	PRODUCTO.PRECIO
1	Respuestas examen Quique	20
3	Respuestas examen Javi	50
2	Respuestas examen Rafa	30