BD Orientada a Objeto (BDOO)

Módulo: Acceso a datos

Ciclo: Desarrollo de Aplicaciones Multiplataforma

Planificación

- Del 21 de octubre al 7 de noviembre Realización de ejercicios resueltos
- Del 11 de noviembre al 28 Realización del proyecto
- Entrega del proyecto Día 1 de diciembre
- Día 10 de diciembre Examen UD 2

BDOR

- Híbrido entre BDR y BDOO Intenta aunar modelos
- Objetivos que persigue:
  - Mejorar la representación de los datos mediante el uso de objetos.
  - Simplificar el acceso a los datos manteniendo el sistema relacional.
- Estructura:
  - Columnas ahora no solo valores escalares y atómicos, también estructurados
  - Filas
  - Tablas pueden estar definidas a partir de otras Herencia

SGBDOR: PostgreSQL

- Schema:
  - Espacio de nombre que contiene los objetos
  - Asegura que los nombres de los elementos sea único
  - Se puede incluir: Tablas, tipos de datos, índices, funciones, etc.
  - Razones para crear un schema:
    - Varios usuarios pueden usar la BD sin que exista interferencia entre ellos.
    - Organiza los objetos de la BD en grupos lógicos haciéndolos más manejables.
    - Separa las aplicaciones entre sí para que no colision en los nombres de sus objetos.
  - Creación de un schema:

```
CREATE SCHEMA nombreEsquema
```

  - Creación de una tabla o objeto dentro de un schema:

```
CREATE TABLE nombreEsquema.nombreTabla(...);

CREATE TYPE nombreEsquema.nombreObjeto(...);
```

SGBDOR: PostgreSQL

- Tipo:
  - Entidad del mundo real
  - Permite encapsular datos y operaciones.
  - Está compuesto por:
    - Nombre: identifica el tipo
    - Atributos: modelan la estructura
    - Métodos: procedimientos o funciones
  - Creación:
    - Tipo básico:

```
CREATE TYPE compfoo AS (f1 int, f2 text);
```

    - Enumerado:

```
CREATE TYPE estado_bug AS ENUM ('nuevo', 'abierto', 'cerrado');
```

    - Objeto:

```
CREATE TYPE informacion AS (maquina varchar(50), ip varchar(50));


CREATE TABLE bug (
        id serial,
        description text,
        status bug_status,
        informacion informacion
);
```

SGBDOR: PostgreSQL

- Tipo:
  - Creación:
    - Array:

```
CREATE TYPE float8_range AS RANGE (subtype = float8, subtype_diff = float8mi);
```

  - Tipos disponibles:

Numeric	Character	Date/Time	Monetary	Binary
Boolean	Geometric	JSON	Enumerated	Text-search
UUID	Network Address Types	Composite	Object Identifiers	Pseudo
BitString	XML	Range	Arrays	pg_lns

SGBDOR: PostgreSQL

- Tablas:
  - Creación:

```
CREATE TABLE [IF NOT EXISTS] nombre (
        nombre_atributo tipo_atributo [restricciones],
);
```

  - Ejemplos

```
CREATE TABLE array_int (
    vector int[][]
);

CREATE TABLE films (
    code char(5),
    title varchar(40),
    did integer,
    CONSTRAINT code_title PRIMARY KEY(code,title) // Creación clave primaria compuesta
);
```

SGBDOR: PostgreSQL

- Operaciones básicas:
  - Consulta:
    - Sin objetos

      SELECT columna1, columna2, ... FROM nombre_tabla WHERE condición;
    - Con objetos

      SELECT (objeto1).atributo1, (objeto1).atributo2, ... FROM nombre_tabla WHERE condición;
  - Inserción:
    - Valores simples

      INSERT INTO nombre_tabla(columna1, columna2, ...) VALUES (valor1, valor1, ...);
    - Objetos:

      INSERT INTO nombre_tabla(columna1, columna2, ...) VALUES (valor1, ROW(valor1, valor2), ...);

SGBDOR: PostgreSQL

- Operaciones básicas:
  - Actualización
    - Valores simples:

      UPDATE nombre_tabla SET columna1 = valor1, columna1 = valor1, … WHERE condición;

    - Objetos:

      UPDATE nombre_tabla SET objeto1.atributo1= valor1, objeto2.atributo2= valor2 WHERE condición;

  - Borrado

    DELETE FROM nombre_tabla WHERE condición;

SGBDOR: PostgreSQL

- Operaciones básicas:
  - Upsert: realizar diferentes acciones cuando se almacena o actualiza una fila (TRIGGERS)

```
INSERT INTO nombre_tabla(lista_columna) VALUES(lista_valores)
ON CONFLICT target action;
```

    - Donde:
      - Target: puede ser:
        - Nombre de una columna
        - ON CONSTRAINT nombre_restricción
        - WHERE predicado
      - Action:
        - DO NOTHING: si la fila existe, no hacer nada
        - DO UPDATE SET columna=valora WHERE *condición:* actualiza ciertos campos

SGBDOR: PostgreSQL

- Relaciones:
  - 1 a 1: Usando un campo único para almacenar el id

```
CREATE TABLE persona (
    id SERIAL PRIMARY KEY,
    nombre VARCHAR(255),
    -- Otras columnas de persona
    direccion_id INT UNIQUE REFERENCES direccion(id)
);

CREATE TABLE direccion (
    id SERIAL PRIMARY KEY,
    calle VARCHAR(255),
    ciudad VARCHAR(255),
    -- Otras columnas de dirección
);
```

SGBDOR: PostgreSQL

- Relaciones:
  - 1 a varios: Usando un campo para almacenar el id

```
CREATE TABLE departamento (
    id SERIAL PRIMARY KEY,
    nombre VARCHAR(255),
    -- Otras columnas del departamento
);

CREATE TABLE empleado (
    id SERIAL PRIMARY KEY,
    nombre VARCHAR(255),
    salario NUMERIC,
    -- Otras columnas del empleado
    departamento_id INT REFERENCES departamento(id)
);
```

SGBDOR: PostgreSQL

- Relaciones:
  - Varios a varios:

```
CREATE TABLE estudiante (
    id SERIAL PRIMARY KEY,
    nombre VARCHAR(255),
    -- Otras columnas del estudiante
);

CREATE TABLE curso (
    id SERIAL PRIMARY KEY,
    nombre VARCHAR(255),
    -- Otras columnas del curso
);

-- Tabla de enlace para representar la relación muchos a muchos
CREATE TABLE inscripcion (
    estudiante_id INT REFERENCES estudiante(id),
    curso_id INT REFERENCES curso(id),
    PRIMARY KEY (estudiante_id, curso_id)
);
```

SGBDOR: PostgreSQL: Java

- Realización de la conexión:

```
Connection conexion = null;

try {
    conexion = DriverManager.getConnection("jdbc:postgresql://localhost:5432/basededatos",
"usuario", "password");
} catch (Exception e) {
    e.prinStackTrace();
}
```

SGBDOR: PostgreSQL: Java

- Creación de una tabla:

```
Statement stmt = conexion.createStatement();
String sql = "CREATE TABLE COMPANY " +
"(ID INT PRIMARY KEY NOT NULL," +
" NAME TEXT NOT NULL, " +
" AGE INT NOT NULL, " +
" ADDRESS CHAR(50), " +
" SALARY REAL)";
stmt.executeUpdate(sql);
stmt.close();
```

SGBDOR: PostgreSQL: Java

- Insertar valores:

```
Statement stmt = conexion.createStatement();
stmt = conexion.createStatement();
String sql = "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) "
+ "VALUES (1, 'Paul', 32, 'California', 20000.00 );";
stmt.executeUpdate(sql);
stmt.close();
```

SGBDOR: PostgreSQL: Java

- Actualizar valores:

```
String updateSQL = "UPDATE tu_tabla SET columna1 = ?, columna2 = ? WHERE condicion";

// Crear una PreparedStatement con la sentencia SQL
PreparedStatement preparedStatement = conexion.prepareStatement(updateSQL);

// Establecer los nuevos valores de las columnas
preparedStatement.setString(1, "nuevo_valor_columna1");
preparedStatement.setString(2, "nuevo_valor_columna2");
stmt.close();
```

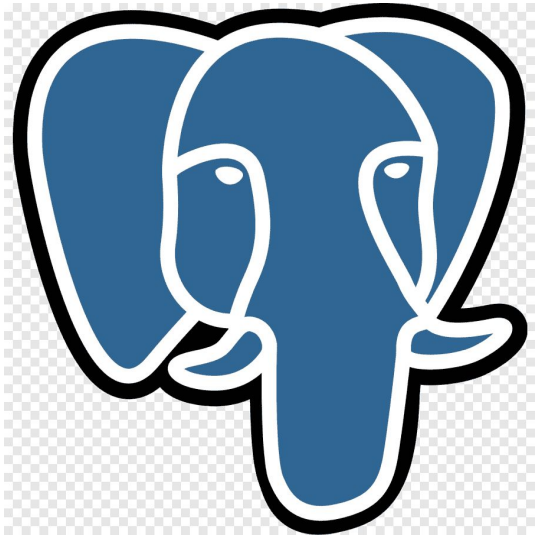
SGBDOR: PostgreSQL: Java

- Eliminar valores:

```
// Definir la sentencia SQL de eliminación
String deleteSQL = "DELETE FROM tu_tabla WHERE condicion";

// Crear una PreparedStatement con la sentencia SQL
PreparedStatement preparedStatement = conexion.prepareStatement(deleteSQL);
stmt.close();
```

Programa: PgAdmin



+

PostgreSQL 12