

PRÁCTICA DE LABORATORIO - BASES DE DATOS ORIENTADAS A OBJETOS PREREQUISITOS

- PostgreSQL instalado (versión 12 o superior)
- pgAdmin 4 o acceso a psql

PARTE 1: MODELADO RELACIONAL TRADICIONAL

Paso 1: Crear la base de datos

Abrir pgAdmin o psql y ejecutar:

The screenshot shows a pgAdmin interface. In the top-left corner, there is a number '1'. To its right, a purple box highlights the SQL command 'CREATE DATABASE lab_bdoo;'. Below this, there are three tabs: 'Data Output' (disabled), 'Messages' (selected and underlined in blue), and 'Notifications'. Under the 'Messages' tab, the command 'CREATE DATABASE' is shown in gray. At the bottom of the message area, the text 'Query returned successfully in 237 msec.' is displayed in gray.

Conectarse a la base de datos:

```
sql  
\c lab_bdoo
```

Paso 2: Crear tablas relacionales tradicionales

Vamos a modelar un sistema de biblioteca con el enfoque relacional tradicional:

```
5  -- Tabla de direcciones
6  CREATE TABLE direcciones (
7      id SERIAL PRIMARY KEY,
8      calle VARCHAR(100),
9      numero VARCHAR(10),
10     ciudad VARCHAR(50),
11     codigo_postal VARCHAR(10)
12 );
13
14  -- Tabla de autores
15  CREATE TABLE autores (
16      id SERIAL PRIMARY KEY,
17      nombre VARCHAR(100),
18      apellido VARCHAR(100),
19      fecha_nacimiento DATE,
20      direccion_id INTEGER REFERENCES direcciones(id)
21 );
22
23  -- Tabla de libros
24  CREATE TABLE libros (
25      id SERIAL PRIMARY KEY,
26      titulo VARCHAR(200),
27      isbn VARCHAR(20),
28      año_publicacion INTEGER,
29      precio NUMERIC(10, 2),
30      autor_id INTEGER REFERENCES autores(id)
31 );
```

Data Output [Messages](#) Notifications

CREATE TABLE

Query returned successfully in 42 msec.

Paso 3: Insertar datos en el modelo relacional

```

34 INSERT INTO direcciones (calle, numero, ciudad, codigo_postal)
35 VALUES ('Av. Arequipa', '1234', 'Lima', '15001');
36
37 -- Insertar autor
38 INSERT INTO autores (nombre, apellido, fecha_nacimiento, direccion_id)
39 VALUES ('Mario', 'Vargas Llosa', '1936-03-28', 1);
40
41 -- Insertar libros
42 INSERT INTO libros (titulo, isbn, año_publicacion, precio, autor_id)
43 VALUES
44 ('La ciudad y los perros', '978-8420471839', 1963, 45.50, 1),
45 ('Conversación en La Catedral', '978-8466326070', 1969, 52.00, 1);

```

Data Output Messages Notifications

INSERT 0 2

Query returned successfully in 42 msec.

Paso 4: Consultar datos (requiere JOINs)

```

49 SELECT
50     l.titulo,
51     l.isbn,
52     a.nombre || ' ' || a.apellido AS autor,
53     d.ciudad
54 FROM libros l
55 JOIN autores a ON l.autor_id = a.id
56 JOIN direcciones d ON a.direccion_id = d.id;

```

Data Output Messages Notifications

	titulo character varying (200)	isbn character varying (20)	autor text	ciudad character varying (50)
1	La ciudad y los perros	978-8420471839	Mario Vargas Llosa	Lima
2	Conversación en La Catedral	978-8466326070	Mario Vargas Llosa	Lima

OBSERVACIÓN: Note que necesitamos 2 JOINs para obtener información completa.

PARTE 2: MODELADO OBJETO-RELACIONAL

Paso 5: Crear tipos de datos personalizados

PostgreSQL permite crear tipos compuestos (característica objeto-relacional):

```

60  -- Tipo personalizado para dirección
61  CREATE TYPE tipo_direccion AS (
62      calle VARCHAR(100),
63      numero VARCHAR(10),
64      ciudad VARCHAR(50),
65      codigo_postal VARCHAR(10)
66  );
67
68  -- Tipo personalizado para autor
69  CREATE TYPE tipo_autor AS (
70      nombre VARCHAR(100),
71      apellido VARCHAR(100),
72      fecha_nacimiento DATE,
73      direccion tipo_direccion
74  );

```

Data Output Messages Notifications

CREATE TYPE

Query returned successfully in 52 msec.

Paso 6: Crear tabla con tipos compuestos

```

78
79  CREATE TABLE libros_oo (
80      id SERIAL PRIMARY KEY,
81      titulo VARCHAR(200),
82      isbn VARCHAR(20),
83      año_publicacion INTEGER,
84      precio NUMERIC(10, 2),
85      autor tipo_autor
86  );

```

Data Output Messages Notifications

CREATE TABLE

Query returned successfully in 43 msec.

OBSERVACIÓN: Note que el campo 'autor' es un tipo complejo que contiene toda la información del autor, incluyendo su dirección anidada.

Paso 7: Insertar datos en el modelo objeto-relacional

```

88      PASO 7. Insertar los valores en el modelo objeto relacional
89      -- Insertar libro con autor y dirección anidados
90      INSERT INTO libros_oo (titulo, isbn, año_publicacion, precio, autor)
91          VALUES
92          (
93              'La ciudad y los perros',
94              '978-8420471839',
95              1963,
96              45.50,
97              ROW('Mario', 'Vargas Llosa', '1936-03-28',
98                  ROW('Av. Arequipa', '1234', 'Lima', '15001'))::tipo_autor
99          );
100
101     INSERT INTO libros_oo (titulo, isbn, año_publicacion, precio, autor)
102         VALUES
103         (
104             'Conversación en La Catedral',
105             '978-8466326070',
106             1969,
107             52.00,
108             ROW('Mario', 'Vargas Llosa', '1936-03-28',
109                 ROW('Av. Arequipa', '1234', 'Lima', '15001'))::tipo_autor
110         );

```

Data Output Messages Notifications

INSERT 0 1

Query returned successfully in 42 msec.

Paso 8: Consultar datos sin JOINs (navegación por puntos)

```

113      -- Acceso directo a campos anidados usando notación de punto
114      SELECT
115          titulo,
116          isbn,
117          (autor).nombre || ' ' || (autor).apellido AS nombre_autor,
118          (autor).direccion.ciudad AS ciudad_autor
119      FROM libros_oo;

```

Data Output Messages Notifications

	titulo character varying (200)	isbn character varying (20)	nombre_autor text	ciudad_autor character varying (50)
1	La ciudad y los perros	978-8420471839	Mario Vargas Llosa	Lima
2	Conversación en La Catedral	978-8466326070	Mario Vargas Llosa	Lima

OBSERVACIÓN: No necesitamos JOINs. Accedemos directamente a los datos anidados usando notación de punto, similar a la programación orientada a objetos.

PARTE 3: COMPARACIÓN Y ANÁLISIS

Paso 9: Comparar consultas

Ejecute ambas consultas y compare los resultados:

Consulta Relacional:

```
122 -- Para obtener libro con datos del autor y su dirección
123 SELECT
124     l.titulo,
125     l.isbn,
126     a.nombre || ' ' || a.apellido AS autor,
127     d.ciudad
128 FROM libros l
129 JOIN autores a ON l.autor_id = a.id
```

Data Output Messages Notifications

	titulo	isbn	autor	ciudad
1	La ciudad y los perros	978-8420471839	Mario Vargas Llosa	Lima
2	Conversación en La Catedral	978-8466326070	Mario Vargas Llosa	Lima

Showing rows: 1 to 2 Page No: 1 of 1

Total rows: 2 Query complete 00:00:00.065 CRLF Ln 123, Col 1

✓ Successfully run. Total query runtime: 65 msec. 2 rows affected.

Consulta Objeto-Relacional:

```
132 SELECT
133     titulo,
134     (autor).nombre || ' ' || (autor).apellido AS autor,
135     ((autor).direccion).ciudad AS ciudad
136 FROM libros_oo;
```

Data Output Messages Notifications

	titulo	autor	ciudad
1	La ciudad y los perros	Mario Vargas Llosa	Lima
2	Conversación en La Catedral	Mario Vargas Llosa	Lima

Showing rows: 1 to 2 Page No: 1 of 1

Total rows: 2 Query complete 00:00:00.068 CRLF Ln 132, Col 1

✓ Successfully run. Total query runtime: 68 msec. 2 rows affected.

Paso 10: Ventajas y desventajas observadas

Complete la siguiente tabla basándose en su experiencia:

Característica		
Modelo Relacional		
Número de tablas necesarias	3	1
Complejidad de consultas	Alta (Requiere join)	Baja (Acceso a la información anidada)
Normalización	Alta (Sigue reglas estrictas de normalización, eliminando redundancia de datos).	Baja (La información del autor y dirección está duplicada dentro de cada registro de libro en libros_oo).

Facilidad de actualización	Alta (Si cambia la dirección del autor, solo se actualiza un registro en la tabla direcciones).	Baja (Si cambia la dirección del autor, se deben actualizar todos los registros en libros_oo que contenga a ese autor).
Consistencia de datos	Alta (Asegurada por Claves Foráneas y la Normalización.	Dependiente del programador (No hay claves foráneas en los tipos compuestos).