

DOCUMENTACIÓN

James Arley Muñoz Borja

CC: 1152202244

Reto base de datos (Sofka U)

ZOO (Ejercicio B)

El parque zoo santafe “parque de la conservación” quiere registrar en una base de datos el consumo de alimentos por los animales que tiene en su sede.

Usted acaba de hablar con el administrador y el le comenta que tienen una clasificación para los animales (mamíferos, aves, anfibios, peces y reptiles) de los cuales usted debe seleccionar 3 para el MVP.

- Tierragro empresa de alimentos para diferentes especies es uno de los 5 proveedores del parque, pero se esperan que al menos lleguen 10 nuevos proveedores.
- Dentro del parque hay varios roles para las personas, empleados cuidadores, empleados logísticos, empleados veterinarios, empleados entrenadores, visitantes.
- El veterinario está encargado de realizar consultas a sus especies y de diseñar la dieta de cada especie.
- El alimento de cada especie es diferente y tiene una dosis y un tipo (húmeda, seca, etc).
- Uno de los roles de empleado del Zoo debe contactarse con el proveedor para solicitar alimentos y debe asear cada una de las hábitas de las especies.
- El proveedor recibe una orden de compra revisa que tenga todo el alimento que le piden y con el genera una factura, a final de mes el gerente del Zoo consulta las facturas que debe a sus distintos proveedores y genera su pago correspondiente.
- Los empleados entrenadores son los encargados de llevar el peso de cada especie e informar a un veterinario en que condición están.
- El alimento es una entidad fuerte y debe contener sus características.

Se pide:

- **Hay que indicar que ejercicio fue asignado**

R/: se me asignó el ejercicio B del zoológico.

- Realizar el modelo E-R

R/:

Para el modelo entidad relación, se crearon las entidades: Animal, informe, entrenador, veterinario, logístico, dieta, alimento, proveedores, orden_compra, factura con sus siguientes atributos.

las entidades se relacionan de la siguiente manera:

Entrenador-informe: un entrenador genera uno o muchos informes y un reporte es generado por un entrenador.

Informe-veterinario: un veterinario puede consultar uno o muchos informes y un informe puede ser consultado por uno o muchos veterinarios.

Veterinario-dieta: un veterinario puede diseñar una o muchas dietas y una dieta puede ser diseñada por uno o muchos veterinarios.

Informe-animal: Un informe puede contener un animal y un animal puede contener muchos informes

Animal-dieta: un animal puede tener una dieta y una dieta puede ser consumida por uno o muchos animales.

Dieta-alimento: una dieta puede tener uno o muchos alimentos y un alimento puede tener una o muchas dietas.

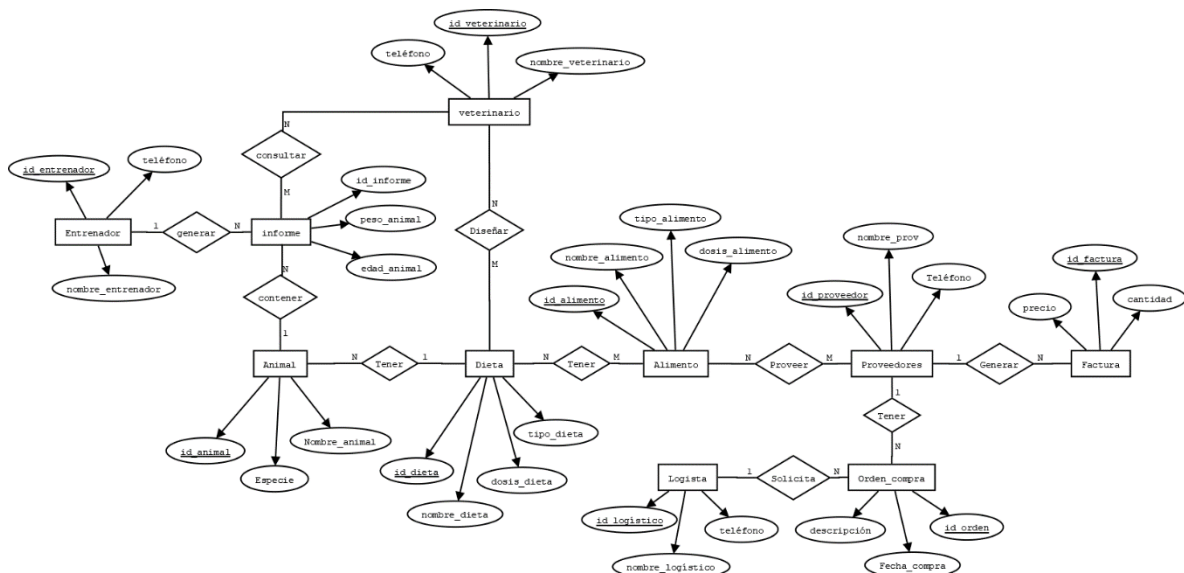
Alimento-proveedor: un alimento se puede proveer por uno o muchos proveedores y un proveedor puede proveer uno o muchos alimentos.

Proveedor-factura: un proveedor puede generar una o muchas facturas, pero una factura puede ser generada por un proveedor.

Orden_compra-proveedor: un proveedor puede tener una o muchas órdenes de compra, pero una orden de compra tiene un solo proveedor.

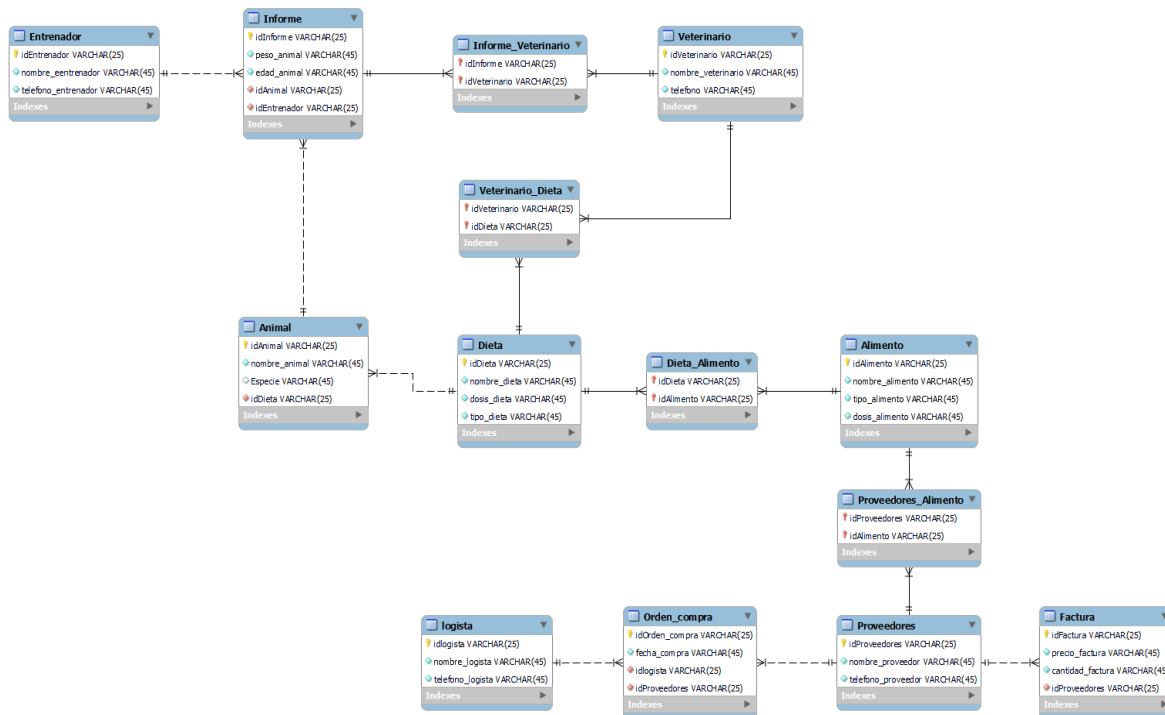
Logístico-orden_compra: un logístico puede generar una o muchas órdenes de compra, pero una orden de compra debe ser generada por un logístico.

se adjunta el modelo entidad relación creado y se adjunta al repositorio:



- **Realizar el modelo relacional**

R/: Creado el MER se procede a crear el modelo relacional quedando de la siguiente forma:



- **Normalizar correctamente**

R/: Normalización

1N (Primera Normalización)

El modelo relacional cumple con la primera normalización porque los atributos de las entidades son atómicos, no hay atributos multivaluados y se garantiza que no hay registros repetidos.

2N (Segunda normalización)

El modelo relacional cumple con la segunda normalización porque está en la primera forma y además todos los valores de las columnas dependen únicamente de una llave primaria de la tabla.

3N (Tercera Normalización)

El modelo relacional cumple con la tercera normalización porque está en la segunda forma y no tiene dependencias transitivas, como se puede observar las columnas en cada tabla dependen directamente de la clave primaria de esa misma tabla.

- Escribir con sentencias SQL toda la definición de la base de datos.

R/: adjunto imágenes de las sentencias SQL, de igual manera estarán adjuntas en el script.

```
1 • CREATE SCHEMA Zoo_Santafe;
2 • USE ZOO_Santafe;
3
4 -- Tabla Dieta
5 • CREATE TABLE Dieta (
6     idDieta VARCHAR(25) NOT NULL,
7     nombre_dieta VARCHAR(45) NOT NULL,
8     dosis_dieta VARCHAR(45) NOT NULL,
9     tipo_dieta VARCHAR(45) NOT NULL,
10    PRIMARY KEY (idDieta));
11
12 -- Tabla Animal
13 • CREATE TABLE Animal (
14     idAnimal VARCHAR(25) NOT NULL,
15     nombre_animal VARCHAR(45) NOT NULL,
16     Especie VARCHAR(45) NULL,
17     idDieta VARCHAR(25) NOT NULL,
18     PRIMARY KEY (idAnimal),
19     FOREIGN KEY (idDieta)
20     REFERENCES Dieta (idDieta));
21
22 -- Tabla logista
23 • CREATE TABLE logista (
24     idlogista VARCHAR(25) NOT NULL,
25     nombre_logista VARCHAR(45) NOT NULL,
26     telefono_logista VARCHAR(45) NOT NULL,
27     PRIMARY KEY (idlogista));
28
29 -- Tabla Entrenador
30 • CREATE TABLE Entrenador (
31     idEntrenador VARCHAR(25) NOT NULL,
32     nombre_eentrenador VARCHAR(45) NOT NULL,
33     telefono_entrenador VARCHAR(45) NOT NULL,
34     PRIMARY KEY (idEntrenador));
35
36 -- Tabla Veterinario
37 • CREATE TABLE Veterinario (
38     idVeterinario VARCHAR(25) NOT NULL,
```

```

39     nombre_veterinario VARCHAR(45) NOT NULL,
40     telefono VARCHAR(45) NOT NULL,
41     PRIMARY KEY (idVeterinario));
42
43     -- Tabla Alimento
44     • CREATE TABLE Alimento (
45         idAlimento VARCHAR(25) NOT NULL,
46         nombre_alimento VARCHAR(45) NOT NULL,
47         tipo_alimento VARCHAR(45) NOT NULL,
48         dosis_alimento VARCHAR(45) NOT NULL,
49         PRIMARY KEY (idAlimento));
50
51     -- Tabla Proveedores
52     • CREATE TABLE Proveedores (
53         idProveedores VARCHAR(25) NOT NULL,
54         nombre_proveedor VARCHAR(45) NOT NULL,
55         telefono_proveedor VARCHAR(45) NOT NULL,
56         PRIMARY KEY (idProveedores));
57
58     -- Tabla Orden_compra
59     • CREATE TABLE Orden_compra (
60         idOrden_compra VARCHAR(25) NOT NULL,
61         fecha_compra VARCHAR(45) NOT NULL,
62         idlogista VARCHAR(25) NOT NULL,
63         idProveedores VARCHAR(25) NOT NULL,
64         PRIMARY KEY (idOrden_compra),
65         FOREIGN KEY (idlogista)
66         REFERENCES logista (idlogista),
67         FOREIGN KEY (idProveedores)
68         REFERENCES Proveedores (idProveedores));
69
70     -- Tabla Animal_Entrenador
71     • CREATE TABLE Animal_Entrenador (
72         idAnimal VARCHAR(25) NOT NULL,
73         idEntrenador VARCHAR(25) NOT NULL,
74         PRIMARY KEY (idAnimal, idEntrenador),
75         FOREIGN KEY (idAnimal)
76         REFERENCES Animal (idAnimal),

```

```

77     FOREIGN KEY (idEntrenador)
78     REFERENCES Entrenador (idEntrenador));
79
80     -- Tabla Entrenador_Veterinario
81 • ● CREATE TABLE Entrenador_Veterinario (
82     idEntrenador VARCHAR(25) NOT NULL,
83     idVeterinario VARCHAR(25) NOT NULL,
84     PRIMARY KEY (idEntrenador, idVeterinario),
85     FOREIGN KEY (idEntrenador)
86     REFERENCES Entrenador (idEntrenador),
87     FOREIGN KEY (idVeterinario)
88     REFERENCES Veterinario (idVeterinario));
89
90     -- Tabla Dieta_Alimento
91 • ● CREATE TABLE Dieta_Alimento (
92     idDieta VARCHAR(25) NOT NULL,
93     idAlimento VARCHAR(25) NOT NULL,
94     PRIMARY KEY (idDieta, idAlimento),
95     FOREIGN KEY (idDieta)
96     REFERENCES Dieta (idDieta),
97     FOREIGN KEY (idAlimento)
98     REFERENCES Alimento (idAlimento));
99
100     -- Tabla Veterinario_Dieta
101 • ● CREATE TABLE Veterinario_Dieta (
102     idVeterinario VARCHAR(25) NOT NULL,
103     idDieta VARCHAR(25) NOT NULL,
104     PRIMARY KEY (idVeterinario, idDieta),
105     FOREIGN KEY (idVeterinario)
106     REFERENCES Veterinario (idVeterinario),
107     FOREIGN KEY (idDieta)
108     REFERENCES Dieta (idDieta));
109
110     -- Tabla Proveedores_Alimento
111 • ● CREATE TABLE Proveedores_Alimento (
112     idProveedores VARCHAR(25) NOT NULL,
113     idAlimento VARCHAR(25) NOT NULL,
114     PRIMARY KEY (idProveedores, idAlimento),

```

```

115     FOREIGN KEY (idProveedores)
116     REFERENCES Proveedores (idProveedores),
117     FOREIGN KEY (idAlimento)
118     REFERENCES Alimento (idAlimento));
119
120 -- Tabla Factura
121 • CREATE TABLE Factura (
122     idFactura VARCHAR(25) NOT NULL,
123     precio_factura VARCHAR(45) NOT NULL,
124     cantidad_factura VARCHAR(45) NOT NULL,
125     idProveedores VARCHAR(25) NOT NULL,
126     PRIMARY KEY (idFactura),
127     FOREIGN KEY (idProveedores)
128     REFERENCES Proveedores (idProveedores));
129
130 • CREATE TABLE Informe (
131     idInforme VARCHAR(25) NOT NULL,
132     peso_animal VARCHAR(45) NULL,
133     edad_animal VARCHAR(45) NULL,
134     idAnimal VARCHAR(25) NOT NULL,
135     idEntrenador VARCHAR(25) NOT NULL,
136     PRIMARY KEY (idInforme),
137     FOREIGN KEY (idAnimal)
138     REFERENCES Animal (idAnimal),
139     FOREIGN KEY (idEntrenador)
140     REFERENCES Entrenador (idEntrenador));

```

- Escribir consultas que me permitan ver la información de cada tabla o de varias tablas (10).

R/: Se realizaron 10 consultas que hacen lo siguiente:

Consulta 1: Ver la lista de animales y su dieta correspondiente:

```

144 -- 1. Ver la lista de animales y su dieta correspondiente
145 • SELECT Animal.nombre_animal, Dieta.nombre_dieta
146 FROM Animal

```

Result Grid

nombre_animal	nombre_dieta
eagle	Sugar
trout	Apples
whale	Sultanas
moose	Hummus
beaver	Green Beans
koala	Jerusalem Artichoke
goose	Pandanus Leaves
walrus	Asian Greens
gnat	Oatmeal
scorpion	Liver
seal	Bonito Flakes

CONSULTA

DATOS DE LA CONSULTA

FILAS DE LA CONSULTA

Output

Action Output

#	Time	Action	Message
1	20:33:11	SELECT Animal.nombre_animal, Dieta.nombre_dieta FROM Animal INNER JOIN Dieta O...	50 row(s) returned

Consulta 2: Ver los detalles de un animal en particular:

```
149 -- 2. Ver los detalles de un animal en particular
150 * SELECT * FROM Animal
151 WHERE idAnimal = 'Anim1';
152
```

→ CONSULTA

	idAnimal	nombre_animal	Especie	idDieta
▶	anim1	eagle	hippopotamus	d1
*	NULL	NULL	NULL	NULL

→ RESULTADO

FILA OBTENIDA

Output

Action Output

#	Time	Action	Message
✓ 1	20:39:46	SELECT * FROM Animal WHERE idAnimal = 'Anim1' LIMIT 0, 2000	1 row(s) returned

Consulta 3: Ver el precio y la cantidad de los productos comprados de la tabla Factura:

```
153 -- 3. Ver el precio y la cantidad de los productos comprados de la tabla Factura:
154 * SELECT precio_factura, cantidad_factura
155 FROM Factura;
156
```

→ CONSULTA

	precio_factura	cantidad_factura
▶	1430	09
	1390	14
	4208	47
	8415	11
	9244	19
	----	--

→ DATOS DE LA CONSULTA

FILAS OBTENIDAS

Output

Action Output

#	Time	Action	Message
✓ 1	20:42:04	SELECT precio_factura, cantidad_factura FROM Factura LIMIT 0, 2000	50 row(s) returned

Consulta 4: Ver todos los detalles de una orden de compra en particular:

The screenshot shows a SQL query in a dark-themed editor and its corresponding result grid. The query is: `-- 4. Ver todos los detalles de una orden de compra en particular`, `SELECT * FROM Orden_compra`, `WHERE idOrden_compra = 'OC1';`. The result grid has columns: `idOrden_compra`, `fecha_compra`, `idlogista`, and `idProveedores`. The first row contains the values: `oc1`, `Sun May 21 04:00:20 COT 1995`, `log1`, and `prov1`. Below the grid, the 'Output' section shows the query execution details: `SELECT * FROM Orden_compra WHERE idOrden_compra = 'OC1' LIMIT 0, 2000` and the message `1 row(s) returned`. Red arrows point from the text labels to the corresponding parts of the interface.

```
157 -- 4. Ver todos los detalles de una orden de compra en particular
158
159 SELECT * FROM Orden_compra
160 WHERE idOrden_compra = 'OC1';
161
```

idOrden_compra	fecha_compra	idlogista	idProveedores
oc1	Sun May 21 04:00:20 COT 1995	log1	prov1

CONSULTA

VALOR OBTENIDO

FILA OBTENIDA

Orden_compra 72 x

Output

Action Output

#	Time	Action	Message
1	20:45:15	SELECT * FROM Orden_compra WHERE idOrden_compra = 'OC1' LIMIT 0, 2000	1 row(s) returned

Consulta 5: Ver todos los proveedores que han suministrado alimentos:

The screenshot shows a SQL query in a dark-themed editor and its corresponding result grid. The query is: `-- 5. Ver todos los proveedores que han suministrado alimentos`, `SELECT DISTINCT nombre_proveedor`, `FROM Proveedores_Alimento`, `JOIN Proveedores ON Proveedores_Alimento.idProveedores = Proveedores.idProveedores;`. The result grid has a single column: `nombre_proveedor`. The first row contains the value: `Halvorson, Huels and Bernier`. Below the grid, the 'Output' section shows the query execution details: `SELECT DISTINCT nombre_proveedor FROM Proveedores_Alimento JOIN Proveedores ...` and the message `50 row(s) returned`. Red arrows point from the text labels to the corresponding parts of the interface.

```
162 -- 5. Ver todos los proveedores que han suministrado alimentos
163 SELECT DISTINCT nombre_proveedor
164 FROM Proveedores_Alimento
165 JOIN Proveedores ON Proveedores_Alimento.idProveedores = Proveedores.idProveedores;
166
```

nombre_proveedor
Halvorson, Huels and Bernier

CONSULTA

DATOS OBTENIDOS

FILAS RETORNADAS

Result 84 x

Output

Action Output

#	Time	Action	Message
1	20:50:27	SELECT DISTINCT nombre_proveedor FROM Proveedores_Alimento JOIN Proveedores ...	50 row(s) returned

Consulta 6: Ver el número de animales que hay en cada especie:

167 -- 6. Ver el número de animales que hay en cada especie
168 • SELECT especie, COUNT(*) FROM Animal
169 GROUP BY especie;
170
171

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

especie	COUNT(*)
hippopotamus	1
seal	1
eel	1
dolphin	2
wolf	2

Result 85 x

Output

Action Output

#	Time	Action	Message
1	20:53:52	SELECT especie, COUNT(*) FROM Animal GROUP BY especie LIMIT 0, 2000	46 row(s) returned

CONSULTA

DATOS OBTENIDOS

FILAS RETORNADAS

Consulta 7: Ver el nombre y teléfono de todos los proveedores de la tabla Proveedores:

171 -- 7. Ver el nombre y teléfono de todos los proveedores de la tabla Proveedores
172 • SELECT nombre_proveedor, telefono_proveedor
173 FROM Proveedores;
174

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

nombre_proveedor	telefono_proveedor
Halvorson, Huels and Bernier	1-852-649-0757
Wolf LLC	(421) 978-4237 x169
Hegmann and Sons	609-446-5388
Corwin-Hermiston	1-286-443-4902
Kovacek-Schowalter	1-167-583-3414

Proveedores 86 x

Output

Action Output

#	Time	Action	Message
1	20:57:19	SELECT nombre_proveedor,telefono_proveedor FROM Proveedores LIMIT 0, 2000	50 row(s) returned

CONSULTA

DATOS OBTENIDOS

FILAS RETORNADAS

Consulta 8: Ver todos los animales y sus entrenadores correspondientes:

```
175 -- 8. Ver todos los animales y sus entrenadores correspondientes
176 • SELECT a.nombre_animal, e.nombre_eentrenador
177 FROM Animal a
178 JOIN Animal_Entrenador ae ON a.idAnimal = ae.idAnimal
179 JOIN Entrenador e ON ae.idEntrenador = e.idEntrenador;
```

Result Grid

nombre_animal	nombre_eentrenador
eagle	Gia Rippin
trout	Bret Boyle
whale	Hugh Waters II
moose	Aron Hill
beaver	Mason Muller

Result 87 x

Output

Action Output

#	Time	Action	Message
1	21:00:42	SELECT a.nombre_animal, e.nombre_eentrenador FROM Animal a JOIN Animal_Entrenad...	50 row(s) returned

CONSULTA

DATOS OBTENIDOS

FILAS RETORNADAS

Consulta 9: Ver el nombre y teléfono de todos los veterinarios:

```
181 -- 9. Ver el nombre y teléfono de todos los veterinarios:
182 • SELECT nombre_veterinario, telefono
183 FROM Veterinario;
184
185
```

Result Grid

nombre_veterinario	telefono
Orville Kiehn	1-786-642-5374
Chuck Moore DDS	1-630-299-7805
King Thiel	366.433.3497
Leandro Robel MD	1-032-258-1110
Ismael Feest	496.403.9903

Veterinario 89 x

Output

Action Output

#	Time	Action	Message
1	21:02:23	SELECT nombre_veterinario, telefono FROM Veterinario LIMIT 0, 2000	50 row(s) returned

CONSULTA

DATOS OBTENIDOS

FILA RETORNADA

Consulta 10: Ver todos los alimentos que se utilizan en cada dieta

The screenshot shows a database query interface. At the top, a SQL query is entered in a text area, highlighted with a red box. The query is:

```
-- 10. Ver todos los alimentos que se utilizan en cada dieta
SELECT Dieta.nombre_dieta, Alimento.nombre_alimento
FROM Dieta
INNER JOIN Dieta_Alimento ON Dieta.idDieta = Dieta_Alimento.idDieta
INNER JOIN Alimento ON Dieta_Alimento.idAlimento = Alimento.idAlimento;
```

Below the query, the results are displayed in a table, also highlighted with a red box. The table has two columns: `nombre_dieta` and `nombre_alimento`. The data is as follows:

nombre_dieta	nombre_alimento
Sugar	Red Lentils
Apples	Kumera
Sultanas	Dates
Hummus	Lavender Flowers
Green Beans	Purple Rice
Jenicalam Artichoke	Pineapple

Below the table, the output of the query is shown in a log, highlighted with a red box. The log entry is:

```
1 21:04:16 SELECT Dieta.nombre_dieta, Alimento.nombre_alimento FROM Dieta INNER JOIN Dieta_... 50 row(s) returned
```

Red arrows point from the text labels "CONSULTA", "DATOS OBTENIDOS", and "FILAS RETORNADAS" to their respective elements in the screenshot.

- Generar de 4 a 6 vistas donde se evidencie lo más importante de cada ejercicio (haga una selección muy responsable de la información realmente importante según el contexto).

R/: Se generan 4 vistas que hacen lo siguiente:

Vista 1: muestra los animales y sus dietas correspondientes lo que puede ser útil para monitorear y asegurarse de que los animales reciban la alimentación adecuada y en las cantidades correctas.

The screenshot shows a database query interface. At the top, a SQL query is entered in a text area, highlighted with a red box. The query is:

```
1 SELECT * FROM zoo_santafe.vista_animal_dieta;
```

Below the query, the results are displayed in a table, also highlighted with a red box. The table has five columns: `idAnimal`, `nombre_animal`, `nombre_dieta`, `dosis_dieta`, and `tipo_dieta`. The data is as follows:

idAnimal	nombre_animal	nombre_dieta	dosis_dieta	tipo_dieta
anim1	eagle	Sugar	03	Kaffir Leaves
anim10	trout	Apples	04	Allspice Ground
anim11	whale	Sultanas	80	Steak Seasoning
anim12	moose	Hummus	58	Curry Chinese

Below the table, the output of the query is shown in a log, highlighted with a red box. The log entry is:

```
1 21:06:54 SELECT * FROM zoo_santafe.vista_animal_dieta LIMIT 0, 2000 50 row(s) returned
```

Red arrows point from the text labels "CONSULTA", "DATOS OBTENIDOS", and "FILAS RETORNADAS" to their respective elements in the screenshot.

Vista 2: muestra los entrenadores y sus animales asignados lo que puede ser útil para asegurarse de que cada animal esté siendo “atendido” adecuadamente.

1 • `SELECT * FROM zoo_santafe.vista_entrenador_animal;`

Result Grid

	idEntrenador	nombre_eentrenador	idAnimal	nombre_animal
▶	ent1	Gia Rippin	anim1	eagle
	ent10	Bret Boyle	anim10	trout
	ent11	Hugh Waters II	anim11	whale
	ent12	Aron Hilll	anim12	moose
	ent13	Mason Muller	anim13	beaver

vista_entrenador_animal 2 x

Output

Action Output

#	Time	Action	Message
1	21:11:53	<code>SELECT * FROM zoo_santafe.vista_entrenador_animal LIMIT 0, 2000</code>	50 row(s) returned

CONSULTA

DATOS OBTENIDOS

FILAS RETORNADAS

vista 3: muestra las dietas y sus alimentos correspondientes lo que puede ser útil para crear planes de alimentación específicos para cada animal y monitorear el uso de los alimentos en el zoológico.

1 • `SELECT * FROM zoo_santafe.vista_dieta_alimento;`

Result Grid

	nombre_dieta	nombre_alimento	tipo_alimento	dosis_alimento
▶	Sugar	Red Lentils	2 teaspoon	84 gramos
	Apples	Kumera	1/4 gallon	05 gramos
	Sultanas	Dates	3 quart	72 gramos
	Hummus	Lavender Flowers	1/2 quart	56 gramos
	Green Beans	Purple Rice	1/2 teaspoon	24 gramos

vista_dieta_alimento2 x

Output

Action Output

#	Time	Action	Message
1	21:16:04	<code>SELECT * FROM zoo_santafe.vista_dieta_alimento LIMIT 0, 2000</code>	50 row(s) returned

CONSULTA

DATOS OBTENIDOS

FILAS RETORNADAS

vista 4: muestra los logistas y las órdenes de compra que han realizado lo que puede ser útil para rastrear el historial de compras y el suministro de alimentos para el zoológico.

1 • `SELECT * FROM zoo_santafe.vista_logista_orden_compra;`

nombre_logista	fecha_compra
Maria Funk	Sun May 21 04:00:20 COT 1995
Eloise Lebsack	Sun Sep 16 09:58:19 COT 1979
Lanie Morar	Thu Jun 20 02:44:37 COT 1974
Josef Barrows	Mon Nov 17 00:28:24 COT 1969
Leoma Fritsch	Sat Jul 24 12:04:29 COT 1993

Output

#	Time	Action	Message
1	21:17:54	<code>SELECT * FROM zoo_santafe.vista_logista_orden_compra LIMIT 0, 2000</code>	50 row(s) returned

Adjunto imagen de las vistas creadas:

```

191  # VISTAS GENERADAS
192
193  -- 1. Vista de animales y sus dietas correspondientes
194  • CREATE VIEW Vista_Animal_Dieta AS
195    SELECT Animal.idAnimal, Animal.nombre_animal, Dieta.nombre_dieta, Dieta.dosis_dieta, Dieta.tipo_dieta
196    FROM Animal
197    INNER JOIN Dieta ON Animal.idDieta = Dieta.idDieta;
198
199
200  -- 2. Vista de entrenadores y sus animales asignados
201  • CREATE VIEW Vista_Entrenador_Animal AS
202    SELECT Entrenador.idEntrenador, Entrenador.nombre_eentrenador, Animal.idAnimal, Animal.nombre_animal
203    FROM Entrenador
204    INNER JOIN Animal_Entrenador ON Entrenador.idEntrenador = Animal_Entrenador.idEntrenador
205    INNER JOIN Animal ON Animal_Entrenador.idAnimal = Animal.idAnimal;
206
207
208  -- 3. Vista de dietas y sus alimentos correspondientes
209  • CREATE VIEW Vista_Dieta_Alimento AS
210    SELECT d.nombre_dieta, a.nombre_alimento, a.tipo_alimento, a.dosis_alimento
211    FROM Dieta d
212    INNER JOIN Dieta_Alimento da ON d.idDieta = da.idDieta
213    INNER JOIN Alimento a ON da.idAlimento = a.idAlimento;
214
215  -- 4. Vista de los logistas y las órdenes de compra que han realizado
216  • CREATE VIEW vista_Logista_Orden_Compra AS
217    SELECT l.nombre_logista, o.fecha_compra
218    FROM logista l
219    INNER JOIN Orden_compra o ON l.idlogista = o.idlogista;

```

- Generar al menos 4 procedimientos almacenados.

R/: Se genera 4 procedimientos que hacen lo siguiente:

procedimiento 1: obtiene el nombre del animal y su dieta.

1 • `call zoo_santafe.obtener_nombre_animal_dieta();`
2

Result Grid

	nombre_animal	nombre_dieta
▶	eagle	Sugar
	trout	Apples
	whale	Sultanas
	moose	Hummus
	beaver	Green Beans

DATOS OBTENIDOS

Result 2 x

Output

Action Output

#	Time	Action	Message
✓ 1	21:22:47	call zoo_santafe.obtener_nombre_animal_dieta()	50 row(s) returned

FILAS OBTENIDAS

CONSULTA PROCESO

procedimiento 2: inserta un nuevo alimento en la base de datos.

```

233 -- 2. Procedimiento que inserta un nuevo alimento en la base de datos
234 drop procedure insertar_alimento;
235 DELIMITER //
236 • CREATE PROCEDURE insertar_alimento(
237     IN p_idAlimento VARCHAR (25),
238     IN p_nombre_alimento VARCHAR(45),
239     IN p_tipo_alimento VARCHAR(45),
240     IN p_dosis_alimento VARCHAR(45)
241 )
242 • BEGIN
243     INSERT INTO Alimento (idAlimento, nombre_alimento, tipo_alimento, dosis_alimento)
244     VALUES (p_idAlimento, p_nombre_alimento, p_tipo_alimento, p_dosis_alimento);
245 END; //
246 call zoo_santafe.insertar_alimento('alim100', 'morcilla', 'chatarra', '500 gramos');
  
```

CONSULTA PROCESO

Output

Action Output

#	Time	Action	Message
✓ 1	21:59:48	call zoo_santafe.insertar_alimento('alim100', 'morcilla', 'chatarra', '500 gramos')	1 row(s) affected

FILA AGREGADA

	idAlimento	nombre_alimento	tipo_alimento	dosis_alimento
▶	alim1	Red Lentils	2 teaspoon	84 gramos
	alim10	Kumera	1/4 gallon	05 gramos
	alim100	morcilla	chatarra	500 gramos
	alim11	Dates	3 quart	72 gramos
	alim12	Lavender Flowers	1/2 quart	56 gramos

}ALIMENTO AGREGADO

procedimiento 3: actualiza la dieta asociada a un animal.

The screenshot shows a database interface with a dark header area containing a SQL query: `1 call zoo_santafe.dieta_de_animal('anim10');`. A red box highlights the query, and a red arrow points to the text "CONSULTA PROCESO". Below the query, a "Result Grid" table is displayed with columns "nombre_dieta", "dosis_dieta", and "tipo_dieta". The first row contains the values "Apples", "04", and "Allspice Ground". A red box highlights this row, and a red arrow points to the text "DIETA ACTUALIZADA". To the right of the table, the text "FILA RETORNADA" is visible. Below the "Result Grid", an "Output" section shows a table with columns "#", "Time", "Action", and "Message". The first row contains the values "1", "22:19:08", "call zoo_santafe.dieta_de_animal('anim10')", and "1 row(s) returned". A red box highlights this row, and a red arrow points to the text "FILA RETORNADA". At the bottom, a larger table lists various diets with columns "idDieta", "nombre_dieta", "dosis_dieta", and "tipo_dieta". The row for "d10" (Apples, 04, Allspice Ground) is highlighted in yellow, and a red arrow points to the text "DIETA ACTUALIZADA".

```
1 call zoo_santafe.dieta_de_animal('anim10');
```

nombre_dieta	dosis_dieta	tipo_dieta
Apples	04	Allspice Ground

DIETA ACTUALIZADA

FILA RETORNADA

#	Time	Action	Message
1	22:19:08	call zoo_santafe.dieta_de_animal('anim10')	1 row(s) returned

idDieta	nombre_dieta	dosis_dieta	tipo_dieta
d1	Sugar	03	Kaffir Leaves
d10	Apples	04	Allspice Ground
d11	Sultanas	80	Steak Seasoning
d12	Hummus	58	Curry Chinese
d13	Green Beans	55	Korma Mix

DIETA ACTUALIZADA

procedimiento 4: obtiene la dieta de un animal dado su ID.

The screenshot shows a database interface with a dark header area containing a SQL query: `1 call zoo_santafe.dieta_de_animal('anim20');`. A red box highlights the query, and a red arrow points to the text "CONSULTA PROCESO". Below the query, a "Result Grid" table is displayed with columns "nombre_dieta", "dosis_dieta", and "tipo_dieta". The first row contains the values "Albacore Tuna", "48", and "Rice Paper". A red box highlights this row, and a red arrow points to the text "DIETA OBTENIDA". To the right of the table, the text "FILA RETORNADA" is visible. Below the "Result Grid", an "Output" section shows a table with columns "#", "Time", "Action", and "Message". The first row contains the values "1", "22:28:41", "call zoo_santafe.dieta_de_animal('anim20')", and "1 row(s) returned". A red box highlights this row, and a red arrow points to the text "FILA RETORNADA".

```
1 call zoo_santafe.dieta_de_animal('anim20');
```

nombre_dieta	dosis_dieta	tipo_dieta
Albacore Tuna	48	Rice Paper

DIETA OBTENIDA

FILA RETORNADA

#	Time	Action	Message
1	22:28:41	call zoo_santafe.dieta_de_animal('anim20')	1 row(s) returned

Adjunto imagen de los procedimientos creados:

```
221      # PROCEDIMIENTOS GENERADOS
222
223      -- 1. Procedimiento para obtener el nombre del animal y su dieta
224      DELIMITER //
225      * CREATE PROCEDURE obtener_nombre_animal_dieta()
226      * BEGIN
227          SELECT Animal.nombre_animal, Dieta.nombre_dieta
228          FROM Animal
229          INNER JOIN Dieta ON Animal.idDieta = Dieta.idDieta;
230      * END; //
231
232      -- 2. Procedimiento que inserta un nuevo alimento en la base de datos
233      DELIMITER //
234      * CREATE PROCEDURE insertar_alimento(
235      *     IN p_nombre_alimento VARCHAR(45),
236      *     IN p_tipo_alimento VARCHAR(45),
237      *     IN p_dosis_alimento VARCHAR(45)
238      * )
239      * BEGIN
240          INSERT INTO Alimento (nombre_alimento, tipo_alimento, dosis_alimento)
241          VALUES (p_nombre_alimento, p_tipo_alimento, p_dosis_alimento);
242      * END; //
243
244      -- 3. Procedimiento que actualiza la dieta asociada a un animal
245      DELIMITER //
246      * CREATE PROCEDURE actualizar_dieta_animal(
247      *     IN p_idAnimal VARCHAR(25),
248      *     IN p_idDieta VARCHAR(25)
249      * )
250      * BEGIN
251          UPDATE Animal
252          SET idDieta = p_idDieta
253          WHERE idAnimal = p_idAnimal;
254      * END; //
255
256      -- 4. Procedimiento para obtener la dieta de un animal dado su ID:
257      DELIMITER //
258      * CREATE PROCEDURE dieta_de_animal(IN animal_id VARCHAR(25))
259      * BEGIN
260          SELECT d.nombre_dieta, d.dosis_dieta, d.tipo_dieta
261          FROM Dieta d
262          INNER JOIN Animal a ON d.idDieta = a.idDieta
263          WHERE a.idAnimal = animal_id;
264      * END; //
```

- Generar al menos 4 triggers

R/: Se genera 4 triggers que hacen lo siguiente:

trigger 1: agrega un animal a la base de datos.

```

272 -- 1. Trigger para agregar un animal
273 CREATE TRIGGER tr_agregar_animal
274 AFTER INSERT ON animal
275 FOR EACH ROW
276 INSERT INTO control_de_cambios_ZOO (usuario, accion, fecha)
277 VALUES (USER(), 'Agregar animal', NOW());
278
279 INSERT INTO animal VALUES('anim300', 'GatoBotas', 'felino', 'd20');
  
```

CONSULTA TRIGGER

DATO AGREGADO

Output: Action Output

#	Time	Action	Message
1	23:30:01	INSERT INTO animal VALUES('anim300', 'GatoBotas', 'felino', 'd20')	1 row(s) affected

FILA AGREGADA

trigger 2: agrega un alimento a la base de datos.

```

282 -- 2. Trigger para agregar un alimento
283 CREATE TRIGGER tr_agregar_alimento
284 AFTER INSERT ON alimento
285 FOR EACH ROW
286 INSERT INTO control_de_cambios_ZOO (usuario, accion, fecha)
287 VALUES (USER(), 'Agregar alimento', NOW());
288
289 INSERT INTO alimento VALUES('alim300', 'hamburguesa', 'callejera', '200g');
  
```

CONSULTA TRIGGER

DATO AGREGADO

Output: Action Output

#	Time	Action	Message
1	23:35:06	INSERT INTO alimento VALUES('alim300', 'hamburguesa', 'callejera', '200g')	1 row(s) affected

FILA AGREGADA

trigger 3: actualiza una dieta de la base de datos.

```

290 -- 3. Trigger para actualizar una dieta
291 CREATE TRIGGER tr_actualizar_dieta
292 AFTER UPDATE ON dieta
293 FOR EACH ROW
294 INSERT INTO control_de_cambios_ZOO (usuario, accion, fecha)
295 VALUES (USER(), 'Actualizar dieta', NOW(), NEW.tipo_dieta, NEW.dosis_dieta, NEW.nombre_dieta, NEW.idDieta);
296
297 UPDATE dieta SET tipo_dieta='altaGrasa', dosis_dieta=50, nombre_dieta='calorica' WHERE idDieta='D30';
  
```

CONSULTA TRIGGER

DIETA ACTUALIZADA

FILA ACTUALIZADA

Output: Action Output

#	Time	Action	Message
1	23:52:47	UPDATE dieta SET tipo_dieta='altaGrasa', dosis_dieta=50, nombre_dieta='calorica' WHERE idDieta='D30'	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0

trigger 4: elimina una dieta de la base de datos.

```
298 -- 4. Trigger para eliminar una dieta
299 CREATE TRIGGER tr_eliminar_dieta
300 AFTER DELETE ON Dieta
301 FOR EACH ROW
302 INSERT INTO control_de_cambios_ZOO (usuario, accion, fecha, idDieta, tipo_dieta, dosis_dieta, nombre_dieta)
303 VALUES (USER(), 'Eliminar dieta', NOW(), OLD.idDieta, OLD.tipo_dieta, OLD.dosis_dieta, OLD.nombre_dieta);
304
305 DELETE FROM Dieta WHERE idDieta = 'D300';
306
```

Output: DIETA ELIMINADA CONSULTA TRIGGER

Action Output

#	time	Action	message
1	00:02:32	DELETE FROM Dieta WHERE idDieta = 'D300'	1 row(s) affected

FILA ELIMINADA

Adjunto registro de los movimientos con las acciones de los 4 triggers:

```
1 * SELECT * FROM zoo_santafe.control_de_cambios_zoo;
```

Result Grid

	usuario	accion	fecha
▶	root@localhost	Agregar animal	2023-02-17 23:30:01
	root@localhost	Agregar alimento	2023-02-17 23:35:06
	root@localhost	Actualizar dieta	2023-02-17 23:52:47
	root@localhost	Eliminar dieta	2023-02-18 00:02:32

- Poblar la base de datos (50 registros por tabla) utilizando una conexión desde Java.

R/: se procede a mostrar la base de datos poblada a través de JAVA, adjuntaré el método que se ejecuta y la evidencia en workbench.

Método para la tabla alimentos:

```
private static void insertarAlimento() { Complexity is 5 Everything is cool!
    for (int i = 1; i <= 50; i++) {
        Alimento alimento = new Alimento();
        alimento.setIdAlimento("alim" + i);
        alimento.setNombre_alimento(faker.food().ingredient());
        alimento.setTipo_alimento(faker.food().measurement());
        alimento.setDosis_alimento(faker.number().digits(count: 2) + " gramos");
        mysqlOperation.setSqlStatement("insert into alimento values('"
            + alimento.getIdAlimento() + "','" + alimento.getNombre_alimento()
            + "','" + alimento.getTipo_alimento() + "','" + alimento.getDosis_alimento()
            + "');");
        mysqlOperation.executeSqlStatementVoid();
    }
}
```

1 • **SELECT * FROM zoo_santafe.alimento;** → CONSULTA

Result Grid

	idAlimento	nombre_alimento	tipo_alimento	dosis_alimento
▶	alim1	Red Lentils	2 teaspoon	84 gramos
	alim10	Kumera	1/4 gallon	05 gramos
	alim11	Dates	3 quart	72 gramos
	alim12	Lavender Flowers	1/2 quart	56 gramos
	alim13	Purple Rice	1/2 teaspoon	24 gramos
	alim14	Pineapple	1/2 teaspoon	78 gramos
	alim15	Vinegar	1/2 quart	31 gramos
	alim16	Melon	3 tablespoon	41 gramos
	alim17	Spelt	1/2 quart	71 gramos
	alim18	Chestnut	1/2 gallon	06 gramos
	alim19	Coconut Oil	1/4 quart	34 gramos

→ DATOS POBLADOS

FILAS AGREGADAS

alimento 1 x

Output

Action Output

#	Time	Action	Message
✓ 1	18:32:47	SELECT * FROM zoo_santafe.dieta LIMIT 0, 2000	50 row(s) returned

Método para poblar la tabla dieta:

```
private static void insertarDieta() { Complexity is 4 Everything is cool!
    for (int i = 1; i <= 50; i++) {
        Dieta dieta = new Dieta();
        dieta.setIdDieta("d" + i);
        dieta.setNombre_dieta(faker.food().ingredient());
        dieta.setDosis_dieta(faker.number().digits(count: 2));
        dieta.setTipo_dieta(faker.food().spice());
        mySqlOperation.setSqlStatement("insert into dieta values('" + dieta.getIdDieta()
            + "', '" + dieta.getNombre_dieta() + "', '" + dieta.getDosis_dieta() + "', '"
            + dieta.getTipo_dieta() + "');");
        mySqlOperation.executeSqlStatementVoid();
    }
}
```

```
1 * SELECT * FROM zoo_santafe.dieta;
```

CONSULTA

Result Grid

	idDieta	nombre_dieta	dosis_dieta	tipo_dieta
▶	d1	Sugar	03	Kaffir Leaves
	d10	Apples	04	Allspice Ground
	d11	Sultanas	80	Steak Seasoning
	d12	Hummus	58	Curry Chinese
	d13	Green Beans	55	Korma Mix
	d14	Jerusalem Artichoke	63	Chilli Ground
	d15	Pandanus Leaves	19	Mustard Seed Black
	d16	Asian Greens	66	Allspice Whole
	d17	Oatmeal	49	Curry Mild
	d18	Liver	38	Vanilla Bean

dieta 2 x

Output

Action Output

#	Time	Action	Message
✓ 1	18:32:47	SELECT * FROM zoo_santafe.dieta LIMIT 0, 2000	50 row(s) returned

DATOS POBLADOS

FILAS AGREGADAS

Método para poblar la tabla animales:

```
private static void insertarAnimales() { Complexity is 5 Everything is cool!
    for (int i = 1; i <= 50; i++) {
        Animal animal = new Animal();
        animal.setIdAnimal("anim" + i);
        animal.setNombre_animal(faker.animal().name());
        animal.setEspecie(faker.animal().name());
        animal.setIdDieta("d" + i);
        mySqlOperation.setSqlStatement("insert into animal values('" + animal.getIdAnimal()
            + "','" + animal.getNombre_animal()
            + "','" + animal.getEspecie() + "','" + animal.getIdDieta() + "');");
        mySqlOperation.executeSqlStatementVoid();
    }
}
```

1 * SELECT * FROM zoo_santafe.animal;

CONSULTA

	idAnimal	nombre_animal	Especie	idDieta
▶	anim1	eagle	hippopotamus	d1
	anim10	trout	seal	d10
	anim11	whale	eel	d11
	anim12	moose	dolphin	d12
	anim13	beaver	wolf	d13
	anim14	koala	goat	d14
	anim15	goose	tortoise	d15
	anim16	walrus	wasp	d16
	anim17	gnat	pig	d17
	anim18	scorpion	wolf	d18
	anim19	seal	hornet	d19
	anim2	termite	lizard	d2

DATOS POBLADOS

FILAS AGREGADAS

Output

Action Output

#	Time	Action	Message
✓ 1	18:47:23	SELECT * FROM zoo_santafe.animal LIMIT 0, 2000	50 row(s) returned

Método para poblar la tabla entrenador:

```
private static void insertarEntrenador() { Complexity is 4 Everything is cool!
    for (int i = 1; i <= 50; i++) {
        Entrenador entrenador = new Entrenador();
        entrenador.setIdEntrenador("ent" + i);
        entrenador.setNombre_eentrenador(faker.name().fullName());
        entrenador.setTelefono_entrenador(faker.phoneNumber().cellPhone());
        mySqlOperation.setSqlStatement("insert into entrenador values('" +
            entrenador.getIdEntrenador() + "','" + entrenador.getNombre_eentrenador()
            + "','" + entrenador.getTelefono_entrenador() + "');");
        mySqlOperation.executeSqlStatementVoid();
    }
}
```

1 • **SELECT * FROM zoo_santafe.entrenador;** → CONSULTA

Result Grid

	idEntrenador	nombre_eentrenador	telefono_entrenador
▶	ent1	Gia Rippin	1-629-587-7629
	ent10	Bret Boyle	087-747-6984
	ent11	Hugh Waters II	792-216-2695
	ent12	Aron Hilll	1-837-503-6136
	ent13	Mason Muller	416.143.9255
	ent14	Carson Halvorson	(519) 874-4615
	ent15	Chuck Funk Sr.	1-472-493-2883

→ DATOS POBLADOS

FILAS AGREGADAS

Output

Action Output

#	Time	Action	Message
✓ 1	18:51:13	SELECT * FROM zoo_santafe.entrenador LIMIT 0, 2000	50 row(s) returned

Método para poblar la tabla animal_entrenador:

```
1 usage  James Muñoz *  
private static void insertarAnimalEntrenador() { Complexity is 5 Everything is cool!  
    for (int i = 1; i <= 50; i++) {  
        Animal_entrenador animal_entrenador = new Animal_entrenador();  
        animal_entrenador.setIdAnimal("anim" + i);  
        animal_entrenador.setIdEntrenador("ent" + i);  
        mySqlOperation.setSqlStatement("insert into animal_entrenador values('"  
            + animal_entrenador.getIdAnimal() + "','" + animal_entrenador.getIdEntrenador() + "')");  
        mySqlOperation.executeSqlStatementVoid();  
    }  
}
```

1 • SELECT * FROM zoo_santafe.animal_entrenador;

CONSULTA

Result Grid

idAnimal	idEntrenador
anim1	ent1
anim10	ent10
anim11	ent11
anim12	ent12
anim13	ent13
anim14	ent14
anim15	ent15

DATOS POBLADOS

FILAS AGREGADAS

Output

#	Time	Action	Message
✓ 1	18:55:34	SELECT * FROM zoo_santafe.animal_entrenador LIMIT 0, 2000	50 row(s) returned

Método para poblar la tabla Dieta_alimento:

```
private static void insertarDietaAlimento() { Complexity is 5 Everything is cool!  
    for (int i = 1; i <= 50; i++) {  
        Dieta_alimento dieta_alimento = new Dieta_alimento();  
        dieta_alimento.setIdDieta("d" + i);  
        dieta_alimento.setIdAlimento("alim" + i);  
        mySqlOperation.setSqlStatement("insert into dieta_alimento values('"  
            + dieta_alimento.getIdDieta() + "','" + dieta_alimento.getIdAlimento() + "')");  
        mySqlOperation.executeSqlStatementVoid();  
    }  
}
```

1 • SELECT * FROM zoo_santafe.dieta_alimento;

CONSULTA

Result Grid

idDieta	idAlimento
d1	alim1
d10	alim10
d11	alim11
d12	alim12
d13	alim13
d14	alim14
d15	alim15
d16	alim16
d17	alim17
d18	alim18
d19	alim19
d2	alim2
d20	alim20

dieta_alimento 2 x

DATOS POBLADOS

Output

Action Output

#	Time	Action	Message
✓ 1	19:07:03	SELECT * FROM zoo_santafe.dieta_alimento LIMIT 0, 2000	50 row(s) returned

Método para poblar la tabla veterinario:

```
private static void insertarVeterinario() { Complexity is 4 Everything is cool!
    for (int i = 1; i <= 50; i++) {
        Veterinario veterinario = new Veterinario();
        veterinario.setIdVeterinario("vet" + i);
        veterinario.setNombre_veterinario(faker.name().fullName());
        veterinario.setTelefono(faker.phoneNumber().cellPhone());
        mySqlOperation.setSqlStatement("insert into veterinario values('" +
            veterinario.getIdVeterinario() + "','" + veterinario.getNombre_veterinario()
            + "','" + veterinario.getTelefono() + "');");
        mySqlOperation.executeSqlStatementVoid();
    }
}
```

```
1 SELECT * FROM zoo_santafe.veterinario;
```

CONSULTA

Result Grid

	idVeterinario	nombre_veterinario	telefono
▶	vet1	Orville Kiehn	1-786-642-5374
	vet10	Chuck Moore DDS	1-630-299-7805
	vet11	King Thiel	366.433.3497
	vet12	Leandro Robel MD	1-032-258-1110
	vet13	Ismael Feest	496.403.9903
	vet14	Dr. Mitzi Weissnat	(336) 402-4767
	vet15	Catheryn Barrows	275.716.5207
	vet16	Miss Marcellus Heller	992-760-4595
	vet17	Gerald Fisher V	1-912-465-3758
	vet18	Joycelyn Jerde II	1-120-166-6649
	vet19	Cherilyn Cummerata	1-889-241-5269
	vet2	Jc Breitenberg	085.671.0209
	vet20	Liz Mayer	(991) 461-5129

Output

Action Output

#	Time	Action	Message
1	19:17:16	SELECT * FROM zoo_santafe.veterinario LIMIT 0, 2000	50 row(s) returned

DATOS POBLADOS

FILAS AGREGADAS

Método para poblar la tabla Entrenador_veterinario:

```
private static void insertarEntrenadorVeterinario() { Complexity is 5 Everything is cool!  
    for (int i = 1; i <= 50; i++) {  
        Entrenador_veterinario entrenador_veterinario = new Entrenador_veterinario();  
        entrenador_veterinario.setIdEntrenador("ent" + i);  
        entrenador_veterinario.setIdVeterinario("vet" + i);  
        mySqlOperation.setSqlStatement("insert into entrenador_veterinario values('"  
            + entrenador_veterinario.getIdEntrenador() + "','" +  
            entrenador_veterinario.getIdVeterinario() + "')");  
        mySqlOperation.executeSqlStatementVoid();  
    }  
}
```

```
1 • SELECT * FROM zoo_santafe.entrenador_veterinario;
```

CONSULTA

Result Grid

idEntrenador	idVeterinario
ent1	vet1
ent10	vet10
ent11	vet11
ent12	vet12
ent13	vet13
ent14	vet14
ent15	vet15
ent16	vet16
ent17	vet17
ent18	vet18
ent19	vet19
ent2	vet2
ent20	vet20

DATOS POBLADOS

FILAS AGREGADAS

Output

Action Output

#	Time	Action	Message
✓ 1	19:24:23	SELECT * FROM zoo_santafe.entrenador_veterinario LIMIT 0, 2000	50 row(s) returned

Método para poblar la tabla proveedores:

```
private static void insertarProveedor() { Complexity is 4 Everything is cool!
    for (int i = 1; i <= 50; i++) {
        Proveedores proveedor = new Proveedores();
        proveedor.setIdProveedores("prov" + i);
        proveedor.setNombre_proveedor(faker.company().name());
        proveedor.setTelefono_proveedor(faker.phoneNumber().phoneNumber());
        mySqlOperation.setSqlStatement("INSERT INTO proveedores VALUES('" +
            proveedor.getIdProveedores() + "','" + proveedor.getNombre_proveedor()
            + "','" + proveedor.getTelefono_proveedor() + "');");
        mySqlOperation.executeSqlStatementVoid();
    }
}
```

1 • **SELECT * FROM zoo_santafe.proveedores;** → CONSULTA

Result Grid

idProveedores	nombre_proveedor	telefono_proveedor
prov 1	Halvorson, Huels and Bernier	1-852-649-0757
prov 10	Wolf LLC	(421) 978-4237 x169
prov 11	Hegmann and Sons	609-446-5388
prov 12	Corwin-Hermiston	1-286-443-4902
prov 13	Kovacek-Schowalter	1-167-583-3414
prov 14	Metz-Bruen	(105) 111-1010
prov 15	Bruen-Ebert	116.188.4070
prov 16	Ferry and Sons	396.288.0136
prov 17	Feil-Prosacco	(146) 462-9091
prov 18	Grimes-Beahan	859.383.3089
prov 19	Waelchi-Bins	065.134.3372
prov 2	Walsh-Goyette	584-034-7230
prov 20	Ebert-Waters	080.589.1690

→ DATOS POBLADOS

FILAS AGREGADAS

Output

Action Output

#	Time	Action	Message
✓ 1	19:33:11	SELECT * FROM zoo_santafe.proveedores LIMIT 0, 2000	50 row(s) returned

Método para poblar la tabla Factura:

```
private static void insertarFactura() { Complexity is 5 Everything is cool!
    for (int i = 1; i <= 50; i++) {
        Factura factura = new Factura();
        factura.setIdFactura("fact" + i);
        factura.setPrecio_factura(faker.number().digits(count: 4));
        factura.setCantidad_factura(faker.number().digits(count: 2));
        factura.setIdProveedores("prov" + i);
        mySqlOperation.setSqlStatement("insert into factura values(' + factura.getIdFactura()
            + ',' + factura.getPrecio_factura() + ',' + factura.getCantidad_factura()
            + ',' + factura.getIdProveedores() + ')");
        mySqlOperation.executeSqlStatementVoid();
    }
}
```

1 * **SELECT * FROM zoo_santafe.factura;** → CONSULTA

Result Grid

	idFactura	precio_factura	cantidad_factura	idProveedores
▶	fact1	1430	09	prov1
	fact10	1390	14	prov10
	fact11	4208	47	prov11
	fact12	8415	11	prov12
	fact13	9244	19	prov13
	fact14	9830	20	prov14
	fact15	0212	03	prov15
	fact16	3507	62	prov16
	fact17	1166	90	prov17
	fact18	9812	46	prov18
	fact19	8847	05	prov19
	fact2	8054	74	prov2
	fact20	3130	16	prov20

factura 2 x

→ DATOS POBLADOS

FILAS AGREGADAS

Output

Action Output

#	Time	Action	Message
1	19:40:11	SELECT * FROM zoo_santafe.factura LIMIT 0, 2000	50 row(s) returned

Método para poblar la tabla Informe:

```
private static void insertarInforme() { Complexity is 6 It's time to do something...
    for (int i = 1; i <= 50; i++) {
        Informe informe = new Informe();
        informe.setIdInforme("inf" + i);
        informe.setPeso_animal(String.valueOf(faker.number().randomDouble( maxNumberOfDecimals: 1, min: 1, max: 500)));
        informe.setEdad_animal(String.valueOf(faker.number().numberBetween(1, 20)));
        informe.setIdAnimal("anim" + i);
        informe.setIdEntrenador("ent" + i);
        mySqlOperation.setSqlStatement("insert into informe values('" +
            informe.getIdInforme() + "','" + informe.getPeso_animal()
            + "','" + informe.getEdad_animal() + "','" + informe.getIdAnimal() + "','"
            + informe.getIdEntrenador() + "');" );
        mySqlOperation.executeSqlStatementVoid();
    }
}
```

1 • **SELECT * FROM zoo_santafe.informe;** → CONSULTA

Result Grid

	idInforme	peso_animal	edad_animal	idAnimal	idEntrenador
▶	inf1	267.6	11	anim1	ent1
	inf10	436.5	12	anim10	ent10
	inf11	9.1	18	anim11	ent11
	inf12	196.4	15	anim12	ent12
	inf13	368.0	1	anim13	ent13
	inf14	450.9	9	anim14	ent14
	inf15	260.4	12	anim15	ent15
	inf16	62.3	16	anim16	ent16
	inf17	62.1	9	anim17	ent17
	inf18	145.8	13	anim18	ent18
	inf19	173.4	2	anim19	ent19
	inf2	54.4	11	anim2	ent2
	inf20	470.4	13	anim20	ent20

informe 2 x

→ DATOS POBLADOS

FILAS AGREGADAS

Output

Action Output

#	Time	Action	Message
✓ 1	19:48:26	SELECT * FROM zoo_santafe.informe LIMIT 0, 2000	50 row(s) returned

Método para poblar la tabla Logista:

```
private static void insertarLogista() { Complexity is 4 Everything is cool!
    for(int i = 1; i <= 50; i++) {
        Logista logista = new Logista();
        logista.setIdlogista("log" + i);
        logista.setNombre_logista(faker.name().fullName());
        logista.setTelefono_logista(faker.phoneNumber().cellPhone());
        mySqlOperation.setSqlStatement("insert into logista values('" +
            logista.getIdlogista() + "','" + logista.getNombre_logista()
            + "','" + logista.getTelefono_logista() + "');");
        mySqlOperation.executeSqlStatementVoid();
    }
}
```

1 • `SELECT * FROM zoo_santafe.logista;` → CONSULTA

Result Grid

	idlogista	nombre_logista	telefono_logista
▶	log1	Maria Funk	961-481-0555
	log10	Eloise Lebsack	645-906-1935
	log11	Lanie Morar	1-254-668-7032
	log12	Josef Barrows	498.949.3245
	log13	Leoma Fritsch	1-766-368-2630
	log14	Simon Jacobson	336.909.1581
	log15	Norine Torphy	(588) 112-2823
	log16	Efren Macejkovic	(894) 622-3839
	log17	Carrie Rohan	1-798-502-1345
	log18	Levi Emard	639.720.8585
	log19	Courtney Beier MD	1-044-031-6538
	log2	Leroy Sawayn	1-550-647-3403
	log20	Xian Knenn	170.901.6184

→ DATOS POBLADOS

FILAS AGREGADAS

Output

Action Output

#	Time	Action	Message
✓ 1	19:52:49	SELECT * FROM zoo_santafe.logista LIMIT 0, 2000	50 row(s) returned

Método para poblar la tabla Orden_compra:

```
private static void insertarOrdenCompra() { Complexity is 6 It's time to do something...
    for (int i = 1; i <= 50; i++) {
        Orden_compra orden_compra = new Orden_compra();
        orden_compra.setIdOrden_compra("oc" + i);
        orden_compra.setFecha_compra(String.valueOf(faker.date().birthday()));
        orden_compra.setIdlogista("log" + i);
        orden_compra.setIdProveedores("prov" + i);
        mySqlOperation.setSqlStatement("insert into orden_compra values('"
            + orden_compra.getIdOrden_compra()
            + "','" + orden_compra.getFecha_compra() + "','" +
            orden_compra.getIdlogista() + "','"
            + orden_compra.getIdProveedores() + "');");
        mySqlOperation.executeSqlStatementVoid();
    }
}
```

1 • **SELECT * FROM zoo_santafe.orden_compra;** → CONSULTA

Result Grid

	idOrden_compra	fecha_compra	idlogista	idProveedores
▶	oc1	Sun May 21 04:00:20 COT 1995	log1	prov1
	oc10	Sun Sep 16 09:58:19 COT 1979	log10	prov10
	oc11	Thu Jun 20 02:44:37 COT 1974	log11	prov11
	oc12	Mon Nov 17 00:28:24 COT 1969	log12	prov12
	oc13	Sat Jul 24 12:04:29 COT 1993	log13	prov13
	oc14	Tue Oct 19 15:18:26 COT 2004	log14	prov14
	oc15	Sat Jan 15 03:37:04 COT 1966	log15	prov15
	oc16	Thu Dec 03 09:06:05 COT 1959	log16	prov16
	oc17	Wed Dec 24 14:10:55 COT 1969	log17	prov17
	oc18	Tue Oct 30 16:56:53 COT 1973	log18	prov18
	oc19	Sun Oct 05 04:27:06 COT 1975	log19	prov19
	oc2	Sun Sep 01 02:35:01 COT 1968	log2	prov2
	oc20	Tue Feb 06 15:03:37 COT 2001	log20	prov20

orden_compra 2 x

Output

Action Output

#	Time	Action	Message
✓ 1	20:01:59	SELECT * FROM zoo_santafe.orden_compra LIMIT 0, 2000	50 row(s) returned

→ DATOS POBLADOS

→ FILAS AGREGADAS

Método para poblar la tabla Proveedores_alimento:

```
private static void insertarProveedoresAlimento() { Complexity is 5 Everything is cool!
    for (int i = 1; i <= 50; i++) {
        Proveedores_alimento proveedoresAlimento = new Proveedores_alimento();
        proveedoresAlimento.setIdProveedores("prov" + i);
        proveedoresAlimento.setIdAlimento("alim" + i);
        mySqlOperation.setSqlStatement("insert into proveedores_alimento values('"
            + proveedoresAlimento.getIdProveedores() + "','" +
            proveedoresAlimento.getIdAlimento() + "')");
        mySqlOperation.executeSqlStatementVoid();
    }
}
```

1 • `SELECT * FROM zoo_santafe.proveedores_alimento;` → CONSULTA

Result Grid

	idProveedores	idAlimento
▶	prov1	alim1
	prov10	alim10
	prov11	alim11
	prov12	alim12
	prov13	alim13
	prov14	alim14
	prov15	alim15
	prov16	alim16
	prov17	alim17
	prov18	alim18
	prov19	alim19
	prov2	alim2
	prov20	alim20

→ DATOS POBLADOS

FILAS AGREGADAS

Output

#	Time	Action	Message
✓ 1	20:07:15	SELECT * FROM zoo_santafe.proveedores_alimento LIMIT 0, 2000	50 row(s) returned

Método para poblar la tabla Veterinario_dieta:

```
private static void insertarVeterinarioDieta() {  
    for(int i = 1; i <= 50; i++) {  
        Veterinario_dieta veterinarioDieta = new Veterinario_dieta();  
        veterinarioDieta.setIdVeterinario("vet" + i);  
        veterinarioDieta.setIdDieta("d" + i);  
        mySqlOperation.setSqlStatement("insert into veterinario_dieta values('"  
            + veterinarioDieta.getIdVeterinario()  
            + "', '" + veterinarioDieta.getIdDieta() + "'),");  
        mySqlOperation.executeSqlStatementVoid();  
    }  
}
```

1 • `SELECT * FROM zoo_santafe.veterinario_dieta;`

CONSULTA

idVeterinario	idDieta
vet1	d1
vet10	d10
vet11	d11
vet12	d12
vet13	d13
vet14	d14
vet15	d15
vet16	d16
vet17	d17
vet18	d18
vet19	d19
vet2	d2
vet20	d20

DATOS POBLADOS

FILAS AGREGADAS

#	Time	Action	Message
✓ 1	20:16:23	SELECT * FROM zoo_santafe.veterinario_dieta LIMIT 0, 2000	50 row(s) returned

- Al terminar el ejercicio responda ¿Está conforme con el resultado obtenido según el contexto o cree que hubiera obtenido un mejor resultado con una base de datos no relacional?

R/: Si estoy conforme, ya que con la base relacional que se usó fue posible alcanzar el objetivo que se requería en el ejercicio sin embargo algunas cosas como el manejo de los datos de las dietas de los animales hubieran podido ser más óptimos en una base de datos no relacional pero aun así se pudo trabajar de manera adecuada con esta base que se utilizó.