



EJERCICIO B - ZOO

Presentado a: Juan Esteban Pineda

Presentado por: Jessica Andrea López



**19 DE FEBRERO
SOFKA U**

ZOO (Ejercicio B)

ZOO (Ejercicio B)

https://www.parquedelaconservacion.com/mapa?location=id_anima1z10

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 habitas 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:

- ✓ Indicar que ejercicio fue asignado
 - El ejercicio asignado fue ZOO (Ejercicio B)
- ✓ Realizar el modelo E-R
- ✓ Realizar el modelo relacional
- ✓ Normalizar correctamente
- ✓ Escribir con sentencias SQL toda la definición de la base de datos.
- ✓ Escribir consultas que me permitan ver la información de cada tabla o de varias tablas (10).
- ✓ 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).
- ✓ Generar al menos 4 procedimientos almacenados.
- ✓ Generar al menos 4 triggers
- ✓ Poblar la base de datos (50 registros por tabla) utilizando una conexión desde Java.
- ✓ 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?

ZOO (Ejercicio B)

- ✓ Documente muy bien su proceso (paso a paso) en un archivo PDF escriba todas las aclaraciones o especificaciones necesarias para realizar el ejercicio.

Tabla de Contenido

Modelo E-R4

Modelo Relacional.....5

Normalización.....5

Sentencias SQL.....7

Consultas9

Vistas15

Procedimientos.....19

Triggers22

Base de datos poblada con conexión desde Java26

Respuesta última pregunta40

ZOO (Ejercicio B)

Modelo E-R

Las entidades propuestas para la solución son:

- **Proveedor:** representa a las empresas que proveen alimentos al parque.
- **Alimento:** representa los alimentos que son suministrados a los animales, con su tipo (húmedo, seco, etc.) y dosis correspondiente.
- **Entrenador:** representa los empleados del parque encargados de llevar el peso de cada especie y de informar al veterinario en qué condición están.
- **Logístico:** representa a los empleados del parque encargados de solicitar alimentos a los proveedores.
- **Veterinario:** representa a los profesionales encargados de diseñar la dieta de cada animal.
- **Animal:** representa a los animales en el parque, clasificados en mamíferos, aves, anfibios, peces y reptiles.
- **Orden de compra:** representa la solicitud de un empleado (logístico) del parque para adquirir alimentos a un proveedor determinado.
- **Factura:** representa las facturas generadas por los proveedores en respuesta a las órdenes de compra.
- **Informe animal:** representa el informe generado por el entrenador para reportar el peso y las condiciones del animal.
- **Dieta:** Representa la dieta diseñada por el veterinario para los animales.

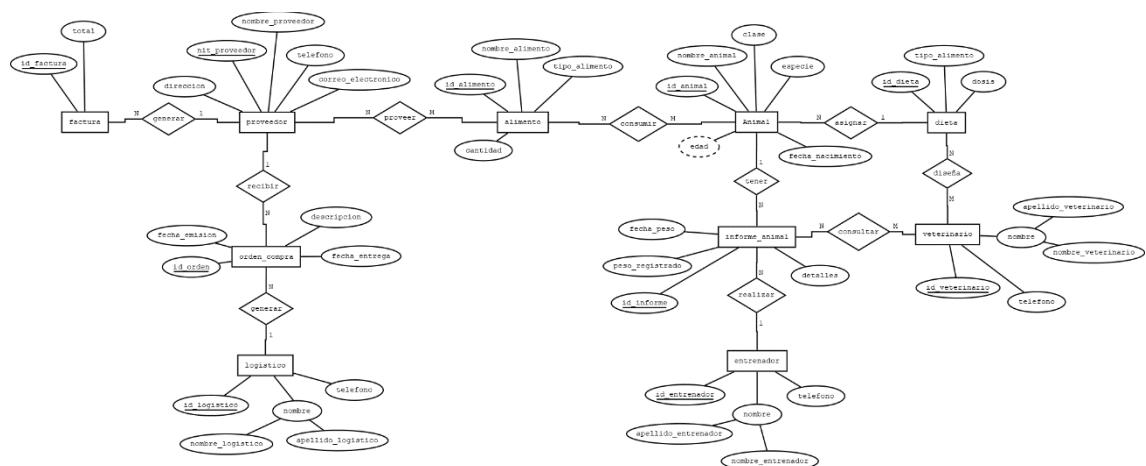


Diagrama E-R

Explicación del diagrama E-R

Proveedor-Factura: Es una relación de uno a muchos, un proveedor puede tener varias facturas, pero una factura solo puede pertenecer a un proveedor.

Proveedor-Orden_Compra: Es una relación de uno a muchos, un proveedor puede tener varias órdenes de compra, pero una orden de compra solo puede pertenecer a un proveedor.

Logístico-Orden_Compra: Es una relación de uno a muchos, un logístico puede estar encargado de varias órdenes de compra, pero una orden de compra solo puede ser entregada por un logístico.

ZOO (Ejercicio B)

Proveedor-Alimento: Es una relación de muchos a muchos, un proveedor puede suministrar varios tipos de alimentos, y un tipo de alimento puede ser suministrado por varios proveedores.

Animal-Dieta: Es una relación de uno a muchos, un animal puede tener solo una dieta, y una dieta puede ser asignada a muchos animales.

Alimento-Animal: Es una relación de muchos a muchos, un alimento puede ser suministrado a varios animales y un animal puede consumir varios alimentos.

Entrenador-Informe Animal: Es una relación de uno a muchos, un entrenador puede hacer varios informes sobre distintos animales, pero un informe solo puede ser realizado por un entrenador.

Veterinario-Informe Animal: Es una relación de muchos a muchos, un veterinario puede consultar muchos informes y un informe puede ser consultado por varios veterinarios.

Veterinario-Dieta: Es una relación de muchos a muchos, una dieta puede ser diseñada por muchos veterinarios y un veterinario puede diseñar varias dietas.

Modelo Relacional

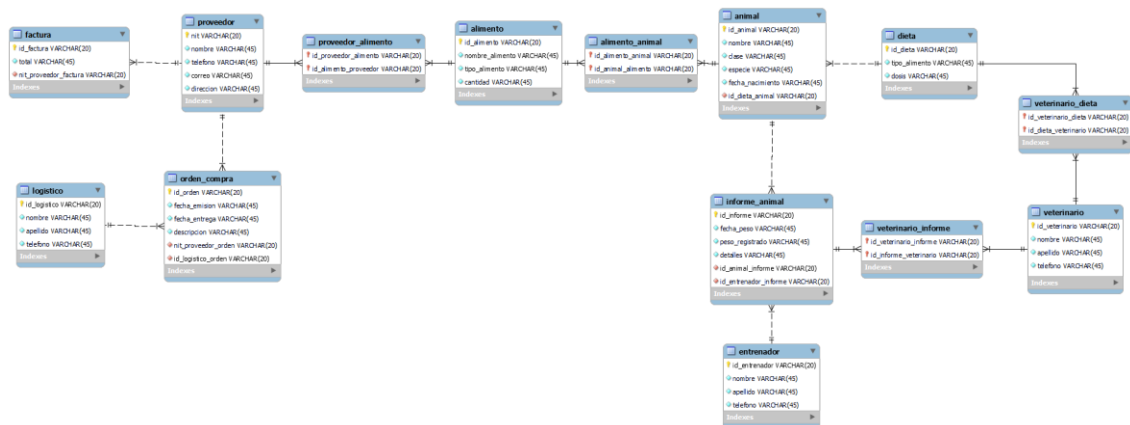


Diagrama Modelo Relacional

Para la realización del Modelo Relacional se tuvieron en cuenta las cardinalidades de uno a muchos, muchos a uno, muchos a muchos. No se tuvieron en cuenta las participaciones ya que en el diagrama entidad relación no hay cardinalidades uno a uno.

Normalización

1N (Primera Normalización)

El Modelo Relacional de la base de datos cumple con la primera normalización porque cada tabla tiene una clave primaria única que identifica de manera única cada fila en la tabla. Cada columna en las tablas representa un solo valor atómico y no hay columnas repetidas que podrían contener datos redundantes.

ZOO (Ejercicio B)

Además, cada tabla contiene solo información relacionada con una sola entidad o relación, lo que significa que no hay tablas que contengan información sobre diferentes entidades o relaciones. Cada tabla también tiene un nombre significativo y descriptivo que refleja claramente su contenido.

2N (Segunda normalización)

El Modelo relacional cumple con la Segunda Forma Normal (2FN) porque todas las tablas tienen una clave primaria única y todas las columnas no clave dependen completamente de la clave primaria.

En el MR, todas las tablas tienen una clave primaria única, y en cada tabla, las columnas no clave dependen completamente de la clave primaria. Por ejemplo, en la tabla "factura", la columna "total" depende de la clave primaria "id_factura" y no hay ninguna columna no clave que dependa solo parcialmente de la clave primaria.

Por ejemplo, en la tabla "orden_compra", se han eliminado las columnas no clave que no dependen de la clave primaria, como "nit_proveedor_orden_nombre", "id_logistico_orden_nombre", ya que esa información se puede obtener a través de la clave primaria "nit_proveedor_orden" y "id_logistico_orden", respectivamente.

En resumen, todas las tablas tienen una única clave primaria y todas las columnas no clave dependen completamente de la clave primaria, lo que hace que todo el esquema cumpla con la Segunda Forma Normal (2FN).

3N (Tercera Normalización)

Para que un modelo cumpla con la tercera forma normal (3FN), debe cumplir con las reglas de la segunda forma normal (2FN) y, además, no debe tener dependencias transitivas.

En este caso, el modelo ya cumple con las reglas de la 2FN, y no hay dependencias transitivas en el esquema. Todas las columnas en cada tabla dependen directamente de la clave primaria de esa tabla, y no hay columnas que dependan de otras columnas que no sean la clave primaria.

Por lo tanto, se puede concluir que el esquema cumple con la tercera forma normal (3FN).

ZOO (Ejercicio B)

Sentencias SQL

```
1  ● CREATE DATABASE zoodb;
2
3  ● USE zoodb;
4
5  ● ⊖ CREATE TABLE proveedor(
6      nit VARCHAR(20) NOT NULL,
7      nombre VARCHAR(45) NOT NULL,
8      telefono VARCHAR(45) NOT NULL,
9      correo VARCHAR(45) NOT NULL,
10     direccion VARCHAR(45) NOT NULL,
11     PRIMARY KEY (nit)
12 );
13
14 ● ⊖ CREATE TABLE factura(
15     id_factura VARCHAR(20) NOT NULL,
16     total VARCHAR(45) NOT NULL,
17     nit_proveedor_factura VARCHAR(20) NOT NULL,
18     PRIMARY KEY (id_factura),
19     FOREIGN KEY (nit_proveedor_factura) REFERENCES proveedor(nit)
20 );
21
22 ● ⊖ CREATE TABLE logistico(
23     id_logistico VARCHAR(20) NOT NULL,
24     nombre VARCHAR(45) NOT NULL,
25     apellido VARCHAR(45) NOT NULL,
26     telefono VARCHAR(45) NOT NULL,
27     PRIMARY KEY (id_logistico)
28 );
```


ZOO (Ejercicio B)

```
29
30 ● ○ CREATE TABLE orden_compra(
31     id_orden VARCHAR(20) NOT NULL,
32     fecha_emision VARCHAR(45) NOT NULL,
33     fecha_entrega VARCHAR(45) NOT NULL,
34     descripcion VARCHAR(45) NOT NULL,
35     nit_proveedor_orden VARCHAR(20) NOT NULL,
36     id_logistico_orden VARCHAR(20) NOT NULL,
37     PRIMARY KEY (id_orden),
38     FOREIGN KEY (nit_proveedor_orden) REFERENCES proveedor(nit),
39     FOREIGN KEY (id_logistico_orden) REFERENCES logistico(id_logistico)
40 );
41
42 ● ○ CREATE TABLE alimento(
43     id_alimento VARCHAR(20) NOT NULL,
44     nombre_alimento VARCHAR(45) NOT NULL,
45     tipo_alimento VARCHAR(45) NOT NULL,
46     cantidad VARCHAR(45) NOT NULL,
47     PRIMARY KEY (id_alimento)
48 );
49
50 ● ○ CREATE TABLE proveedor_alimento(
51     id_proveedor_alimento VARCHAR(20) NOT NULL,
52     id_alimento_proveedor VARCHAR(20) NOT NULL,
53     PRIMARY KEY (id_proveedor_alimento, id_alimento_proveedor),
54     FOREIGN KEY (id_proveedor_alimento) REFERENCES proveedor(nit),
55     FOREIGN KEY (id_alimento_proveedor) REFERENCES alimento(id_alimento)
56 );
57
58 ● ○ CREATE TABLE dieta(
59     id_dieta VARCHAR(20) NOT NULL,
60     tipo_alimento VARCHAR(45) NOT NULL,
61     dosis VARCHAR(45) NOT NULL,
62     PRIMARY KEY (id_dieta)
63 );
64
65 ● ○ CREATE TABLE animal(
66     id_animal VARCHAR(20) NOT NULL,
67     nombre VARCHAR(45) NOT NULL,
68     clase VARCHAR(45) NOT NULL,
69     especie VARCHAR(45) NOT NULL,
70     fecha_nacimiento VARCHAR(45) NOT NULL,
71     id_dieta_animal VARCHAR(20) NOT NULL,
72     PRIMARY KEY (id_animal),
73     FOREIGN KEY (id_dieta_animal) REFERENCES dieta(id_dieta)
74 );
75
76 ● ○ CREATE TABLE alimento_animal(
77     id_alimento_animal VARCHAR(20) NOT NULL,
78     id_animal_alimento VARCHAR(20) NOT NULL,
79     PRIMARY KEY (id_alimento_animal, id_animal_alimento),
80     FOREIGN KEY (id_alimento_animal) REFERENCES alimento(id_alimento),
81     FOREIGN KEY (id_animal_alimento) REFERENCES animal(id_animal)
82 );
83
84 ● ○ CREATE TABLE entrenador(
85     id_entrenador VARCHAR(20) NOT NULL,
86     nombre VARCHAR(45) NOT NULL,
87     apellido VARCHAR(45) NOT NULL,
88     telefono VARCHAR(45) NOT NULL,
89     PRIMARY KEY (id_entrenador)
90 );
--
```

ZOO (Ejercicio B)

```
91
92 ● CREATE TABLE informe_animal(
93     id_informe VARCHAR(20) NOT NULL,
94     fecha_peso VARCHAR(45) NOT NULL,
95     peso_registrado VARCHAR(45) NOT NULL,
96     detalles VARCHAR(45) NOT NULL,
97     id_animal_informe VARCHAR(20) NOT NULL,
98     id_entrenador_informe VARCHAR(20) NOT NULL,
99     PRIMARY KEY (id_informe),
100     FOREIGN KEY (id_animal_informe) REFERENCES animal(id_animal),
101     FOREIGN KEY (id_entrenador_informe) REFERENCES entrenador(id_entrenador)
102 );
103
104 ● CREATE TABLE veterinario(
105     id_veterinario VARCHAR(20) NOT NULL,
106     nombre VARCHAR(45) NOT NULL,
107     apellido VARCHAR(45) NOT NULL,
108     telefono VARCHAR(45) NOT NULL,
109     PRIMARY KEY (id_veterinario)
110 );
111
112 ● CREATE TABLE veterinario_informe(
113     id_veterinario_informe VARCHAR(20) NOT NULL,
114     id_informe_veterinario VARCHAR(20) NOT NULL,
115     PRIMARY KEY (id_veterinario_informe, id_informe_veterinario),
116     FOREIGN KEY (id_veterinario_informe) REFERENCES veterinario(id_veterinario),
117     FOREIGN KEY (id_informe_veterinario) REFERENCES informe_animal(id_informe)
118 );
119
120 ● CREATE TABLE veterinario_dieta(
121     id_veterinario_dieta VARCHAR(20) NOT NULL,
122     id_dieta_veterinario VARCHAR(20) NOT NULL,
123     PRIMARY KEY (id_veterinario_dieta, id_dieta_veterinario),
124     FOREIGN KEY (id_veterinario_dieta) REFERENCES veterinario(id_veterinario),
125     FOREIGN KEY (id_dieta_veterinario) REFERENCES dieta(id_dieta)
126 );
127
```

Consultas

1. Consulta para ver la cantidad y el tipo de alimento existente en la sede

ZOO (Ejercicio B)

```
1 • USE zoodb;
2
3
4 -- 1. Consulta para ver la cantidad y el tipo de alimento existente en la sede
5
6 • SELECT nombre_alimento, tipo_alimento, cantidad FROM alimento;
7
8
9 -- 2. Consulta para ver la dieta de cada animal y el veterinario que la diseñó
10
11 • SELECT animal.nombre AS nombre_animal, dieta.tipo_alimento, dieta.dosis, veterinario.nombre AS diseñado_por
12 FROM animal
13 INNER JOIN dieta ON animal.id_dieta_animal = dieta.id_dieta
14 INNER JOIN informe_animal ON animal.id_animal = informe_animal.id_animal_informe
15 INNER JOIN veterinario_informe ON informe_animal.id_informe = veterinario_informe.id_informe_veterinario
16 INNER JOIN veterinario ON veterinario_informe.id_veterinario_informe = veterinario.id_veterinario;
17
```

Consulta

nombre_alimento	tipo_alimento	cantidad
Sesame Seeds	aliquam	1291 kg
Budowheat Flour	sit	1712 kg
Blue Swimmer Crab	saepe	3574 kg
Sardines	deleniti	2321 kg
Purple Carrot	adipisci	1991 kg
Watermelon	eos	3767 kg
Snowpea sprouts	inventore	3862 kg
Morwong	quos	32 kg

Resultado de la consulta

Retorna todo el alimento existente

Output

#	Time	Action	Message
1	12:27:27	SELECT nombre_alimento, tipo_alimento, cantidad FROM alimento LIMIT 0, 1000	50 row(s) returned

2. Consulta para ver la dieta de cada animal y el veterinario que la diseñó.

```
8
9 -- 2. Consulta para ver la dieta de cada animal y el veterinario que la diseñó
10
11 • SELECT animal.nombre AS nombre_animal, dieta.tipo_alimento, dieta.dosis, veterinario.nombre AS diseñado_por
12 FROM animal
13 INNER JOIN dieta ON animal.id_dieta_animal = dieta.id_dieta
14 INNER JOIN informe_animal ON animal.id_animal = informe_animal.id_animal_informe
15 INNER JOIN veterinario_informe ON informe_animal.id_informe = veterinario_informe.id_informe_veterinario
16 INNER JOIN veterinario ON veterinario_informe.id_veterinario_informe = veterinario.id_veterinario;
17
18
19 -- 3. Consulta para ver las ordenes de compra y el logístico que lo creó
20
```

Consulta

nombre_animal	tipo_alimento	dosis	diseñado_por
fish	aut	1319 gr	Michiko
snail	consequuntur	1962 gr	Margarette
wombat	et	7387 gr	Sydney
guinea pig	velit	9753 gr	Mercedez
mallard	aut	7293 gr	Desmond
goat	nihil	5261 gr	Demetrius
chinchilla	pariatur	2116 gr	Marylynn
porpoise	ipsum	5754 gr	Sammie

Resultado de la consulta

Retorna todos los animales existentes con su dieta y el veterinario que la diseñó

Output

#	Time	Action	Message
1	12:29:30	SELECT animal.nombre AS nombre_animal, dieta.tipo_alimento, dieta.dosis, veterinario.nombre AS diseñado_por FROM animal INNER JOIN...	50 row(s) returned

3. Consulta para ver las órdenes de compra y el logístico que lo creó.

ZOO (Ejercicio B)

```
18 -----
19 -- 3. Consulta para ver las ordenes de compra y el logistico que lo creó
20 -----
21 • SELECT orden_compra.id_orden, orden_compra.fecha_emision, orden_compra.fecha_entrega, orden_compra.descripcion, logistico.nombre as nombre_logistico,
22 logistico.apellido as apellido_logistico
23 FROM orden_compra
24 JOIN logistico ON orden_compra.id_logistico_orden = logistico.id_logistico;
25 -----
26
27 -- 4. Consulta para ver la cantidad de alimento existente segun tipo de alimento,
28 -- y total de alimento diario Requerido segun dietas diseñadas
29 -----
30 • SELECT dieta.tipo_alimento, SUM(dieta.dosis) AS total_alimento_diario_requerido,
31 (SELECT SUM(cantidad) FROM alimento WHERE alimento.tipo_alimento = dieta.tipo_alimento) AS cantidad_existente
32 FROM dieta
```

Consulta

id_orden	fecha_emision	fecha_entrega	descripcion	nombre_logistico	apellido_logistico
ORD-0	Sat Feb 29 20:14:01 COT 2020	Wed Nov 14 12:23:22 COT 2018	v7wgjl1oyy0ndr2okbl4ro9odrd0t436hynxmdvr2c6r	Francene	Lindgren
ORD-1	Wed Oct 08 07:52:40 COT 2014	Tue Oct 06 09:39:09 COT 2020	7e0dxj4ke4vqhl8ptgxcgj61vhudfcrh0hte7mt630a	Marcelino	Quigley
ORD-10	Sun Sep 11 09:37:26 COT 2005	Mon Apr 23 21:12:55 COT 2012	rwkcvylzmrp26ajvmmpf4whle95vm0plvbx1s5kre3	Philip	Leannon
ORD-11	Fri Apr 28 02:09:47 COT 2006	Thu Jan 07 06:00:29 COT 2010	a16qe08m45puceu7jjaq5ek6lsojF86jo1wu32jz23x	Ricardo	Toy
ORD-12	Tue Sep 30 05:12:26 COT 2003	Tue Aug 21 18:16:08 COT 2018	rpmukktip00q14130ja75v60h7iktbpplhs3h32f3d5p	Edith	McDermott
ORD-13	Mon Sep 12 15:13:23 COT 2022	Sun May 31 18:05:29 COT 2020	r4fvtwss50urv6c5j0h2vf1dtfmgfcdoncxqwuzdr	Korey	Price
ORD-14	Wed Oct 13 04:23:47 COT 2021	Wed Jan 16 16:28:42 COT 2008	df8vttj56c4ip1duz68yljh2k890gwx5ts7qbsv1b7wi	Wilburn	Prosacco
ORD-15	Mon Sep 27 11:35:01 COT 2004	Fri Jul 20 07:12:29 COT 2018	09ys8loans8lq2w8m1m4u40aokqzspfm1Sei0tqdx8id	Scottie	Russel

Resultado de la consulta

Retorna todas las ordenes de compra

4. Consulta para ver la cantidad de alimento existente según tipo de alimento, y total de alimento diario Requerido según dietas diseñadas, la comparación solo tiene en cuenta el tipo de alimento diseñado en dietas asignadas a animales

```
26 -----
27 -- 4. Consulta para ver la cantidad de alimento existente segun tipo de alimento,
28 -- y total de alimento diario Requerido segun dietas diseñadas
29 -----
30 • SELECT dieta.tipo_alimento, SUM(dieta.dosis) AS total_alimento_diario_requerido,
31 (SELECT SUM(cantidad) FROM alimento WHERE alimento.tipo_alimento = dieta.tipo_alimento) AS cantidad_existente
32 FROM dieta
33 WHERE dieta.tipo_alimento IN (SELECT tipo_alimento FROM alimento)
34 GROUP BY dieta.tipo_alimento;
35 -----
36
37 -- 5. Consulta para ver el alimento que un animal consume diario segun su dieta
38 -----
39 • SELECT a.id_animal, a.nombre as nombre_animal, d.tipo_alimento as tipo_alimento_que_consume, d.dosis, al.nombre_alimento
40 FROM animal a
41 INNER JOIN dieta d ON a.id_dieta_animal = d.id_dieta
```

Consulta

tipo_alimento	total_alimento_diario_requerido	cantidad_existente
consequuntur	1962	2051
et	10741	6806
nihil	5261	3349
quos	14797	32
consequatur	2832	580
ut	9738	3379
nulla	2770	2642
non	8424	6067

Resultado de la consulta

Retorna 14 filas de cada tipo de alimento diferente y los agrupa sumando la cantidad existente y la requerida. Cuando se pobló la tabla algunos tipos de alimento se repitieron.

Retorna 14 row(s) returned

5. Consulta para ver el alimento que un animal consume diario según su dieta

ZOO (Ejercicio B)

```
36
37 -- 5. Consulta para ver el alimento que un animal consume diario segun su dieta
38
39 • SELECT a.id_animal, a.nombre as nombre_animal, d.tipo_alimento as tipo_alimento_que_consume, d.dosis, al.nombre_alimento
40 FROM animal a
41 INNER JOIN dieta d ON a.id_dieta_animal = d.id_dieta
42 INNER JOIN alimento_animal aa ON a.id_animal = aa.id_animal_alimento
43 INNER JOIN alimento al ON aa.id_alimento_animal = al.id_alimento;
44
45
46 -- 6. Consulta para ver el informe de todos los animales y el entrenador
47
```

Consulta

id_animal	nombre_animal	tipo_alimento_que_consume	dosis	nombre_alimento
ANI-0	fish	aut	1319 gr	Flounder
ANI-1	snail	consequuntur	1962 gr	Anchovies
ANI-10	wombat	et	7387 gr	Soymilk
ANI-11	guinea pig	velit	9753 gr	Nutritional Yeast
ANI-12	mallard	aut	7293 gr	Cannellini Beans
ANI-13	goat	nihil	5261 gr	Oysters
ANI-14	chinchilla	pariatur	2116 gr	Mullet
ANI-15	porpoise	ipsum	5754 gr	Anchovies

Resultado de la consulta

Retorna todos los animales a los que se les asignó una dieta

Output

Action Output

1 12:47:42 SELECT a.id_animal, a.nombre as nombre_animal, d.tipo_alimento as tipo_alimento_que_consume, d.dosis, al.nombre_alimento FROM anim... 50 row(s) returned

6. Consulta para ver el informe de todos los animales y el entrenador

```
45
46 -- 6. Consulta para ver el informe de todos los animales y el entrenador
47
48 • SELECT animal.nombre AS nombre_animal, informe_animal.fecha_peso, informe_animal.peso_registrado,
49 informe_animal.detalles, entrenador.nombre AS nombre_entrenador, entrenador.apellido AS apellido_entrenador
50 FROM animal
51 JOIN informe_animal ON animal.id_animal = informe_animal.id_animal_informe
52 JOIN entrenador ON informe_animal.id_entrenador_informe = entrenador.id_entrenador;
53
54
55 -- 7. Consulta para ver todas las facturas y la informacion del proveedor
56
57 • SELECT factura.id_factura, proveedor.nombre, proveedor.telefono, proveedor.correo,
58 proveedor.direccion, factura.total
59 FROM factura
```

Consulta

nombre_animal	fecha_peso	peso_registrado	detalles	nombre_entrenador	apellido_entrenador
fish	Sat Jan 01 17:12:39 COT 2022	1689 kg	5jfbxy6ozw7vxt7vpwxco0zd7rvvub9tic0vlnojh...	Jeffrey	Rolfson
snail	Mon Feb 13 11:21:00 COT 2023	4043 kg	8p0a5pvhmkaakioxyzz1lgbmkt4fydzg9ty6atl6b...	Toney	Schultz
wombat	Tue Dec 13 07:42:24 COT 2022	2847 kg	rsdzesiu03r040dt4dei6hfu8qg70dbn2en5yywm...	Marybelle	Streich
guinea pig	Fri Dec 23 05:03:11 COT 2022	8578 kg	23elfhv9kg5ni2tmhxxpus5lzt65tc4rmitc4ccb6d4w...	Jaye	Orn
mallard	Thu May 05 05:45:42 COT 2022	8451 kg	vlyd9kirtgxd5szx8yyd67srjba7e5wiwsmgugd04...	Jeane	Schiller
goat	Sat Dec 31 14:21:59 COT 2022	4001 kg	q7h0d3s2inosj3p1c7wgc2kag5kb03mhsagrsvg4...	Aurora	Cruikshank
chinchilla	Wed Nov 30 06:59:51 COT 2022	5211 kg	gv64gm37erj8s94s6rxdzcdqaw9ud4fyt4h4nr...	Mary	Streich
porpoise	Wed Nov 02 21:05:05 COT 2022	6342 kg	d7bh29c54cds9ctu8mdlzdilzq1mca3ddut31kw99...	Shannon	Stroman

Resultado de la consulta

Retorna todos los informes

Output

Action Output

1 12:50:01 SELECT animal.nombre AS nombre_animal, informe_animal.fecha_peso, informe_animal.peso_registrado, informe_animal.detalles, entrenado... 50 row(s) returned

7. Consulta para ver todas las facturas y la información del proveedor

ZOO (Ejercicio B)

```
53
54
55 -- 7. Consulta para ver todas las facturas y la informacion del proveedor
56
57 • SELECT factura.id_factura, proveedor.nombre, proveedor.telefono, proveedor.correo,
58 proveedor.direccion, factura.total
59 FROM factura
60 JOIN proveedor
61 ON factura.nit_proveedor_factura = proveedor.nit;
62
63
64 -- 8. Consulta para ver el nombre, la clase, la especie y la fecha de nacimiento de animales
65
66 • SELECT nombre, clase, especie, fecha_nacimiento FROM animal;
67
68
```

Consulta

Resultado de la consulta

Retorna todas las facturas con informacion adicional

id_factura	nombre	telefono	correo	direccion	total
FAC01	Geneva Mertz DVM	55593	emory.bogisch@hotmail.com	Renea Parkways	\$4945
FAC101	Chadwick Kovacek	55593	oretha.simonis@yahoo.com	Theodore Well	\$2459
FAC11	Blanche Hudson Jr.	55556	vito.schimmel@yahoo.com	Collins Mills	\$5072
FAC111	Bryant Lind	55570	leatha.zieme@gmail.com	Tillman Crossing	\$2115
FAC121	Buford Brown	55514	bernie.tromp@hotmail.com	Kub Plaza	\$1035
FAC131	Hai Breitenberg	55579	derek.heidenreich@yahoo.com	Keshia Drive	\$3128
FAC141	Ms. Maureen Waelchi	55506	earnestine.pollich@hotmail.com	Kihn Port	\$8178
FAC151	Mui Runolfsdottir	55596	kris.quitzon@hotmail.com	Steuber Tunnel	\$5603

Output

Action Output

1 12:51:33 SELECT factura.id_factura, proveedor.nombre, proveedor.telefono, proveedor.correo, proveedor.direccion, factura.total FROM factura JOIN... 50 row(s) returned

8. Consulta para ver el nombre, la clase, la especie y la fecha de nacimiento de animales

```
62
63
64 -- 8. Consulta para ver el nombre, la clase, la especie y la fecha de nacimiento de animales
65
66 • SELECT nombre, clase, especie, fecha_nacimiento FROM animal;
67
68
69 -- 9. Consulta para ver la cantidad de animales segun su clase
70
71 • SELECT clase, COUNT(*) AS cantidad
72 FROM animal
73 GROUP BY clase;
74
```

Consulta

Resultado de la consulta

Retorna todos los animales

nombre	clase	especie	fecha_nacimiento
guinea pig	bat	cheetah	Thu Sep 28 04:00:33 COT 1995
mallard	rhinoceros	cod	Fri May 07 09:17:57 COT 2004
goat	woodchuck	manatee	Thu May 30 21:34:04 COT 2019
chinchilla	butterfly	dog	Wed Oct 30 20:05:52 COT 1996
porpoise	hamster	sardine	Sun Oct 12 23:15:36 COT 2014
chimpanzee	guinea pig	toad	Mon May 02 06:41:04 COT 1994
swan	wolf	elk	Sun May 16 04:34:28 COT 2004
hornet	goose	toad	Sat Sep 06 15:53:41 COT 2008

Output

Action Output

1 12:53:20 SELECT nombre, clase, especie, fecha_nacimiento FROM animal LIMIT 0, 1000 50 row(s) returned

9. Consulta para ver la cantidad de animales según su clase

ZOO (Ejercicio B)

```
67
68
69 -- 9. Consulta para ver la cantidad de animales segun su clase
70
71 • SELECT clase, COUNT(*) AS cantidad
72   FROM animal
73  GROUP BY clase;
74
75
76 -- 10. Consulta para saber la cantidad de alimento existente segun el tipo de alimento
77
78 • SELECT tipo_alimento, SUM(cantidad) as cantidad_total
79   FROM alimento
80  GROUP BY tipo_alimento;
```

Consulta

Resultado de la consulta

Retorna las clases de animales y la cantidad de animales que pertenecen a esa clase

clase	cantidad
worm	1
spider	1
alpaca	1
bat	1
rhinoceros	2
woodchuck	1
butterfly	1
hamster	1

Result 13 x

Output

Action Output

#	Time	Action	Message
1	12:55:31	SELECT clase, COUNT(*) AS cantidad FROM animal GROUP BY clase LIMIT 0, 1000	43 row(s) returned

10. Consulta para saber la cantidad de alimento existente segun el tipo de alimento y lo organiza de mayor a menor

```
75
76 -- 10. Consulta para saber la cantidad de alimento existente segun el tipo de alimento y lo organiza de mayor a menor
77
78 • SELECT tipo_alimento, SUM(cantidad) as cantidad_total
79   FROM alimento
80  GROUP BY tipo_alimento
81  ORDER BY cantidad_total DESC
82
```

Consulta

Resultado de la consulta

Retorna los diferentes tipos de alimento de la tabla alimento y su cantidad

tipo_alimento	cantidad_total
et	6806
non	6067
inventore	5415
ad	4768
laborum	4654
sunt	4128
possimus	4100
in	3981
...	...

Result 16 x

Output

Action Output

#	Time	Action	Message
1	13:07:14	SELECT tipo_alimento, SUM(cantidad) as cantidad_total FROM alimento GROUP BY tipo_alimento ORDER BY cantidad_total DESC LIMIT 0, 40	40 row(s) returned

ZOO (Ejercicio B)

Vistas

1. Vista para ver la dieta de cada animal y el veterinario que la diseñó

-- 1. Vista para ver la dieta de cada animal y el veterinario que la diseñó

```
CREATE VIEW vista_animal_dieta_veterinario AS
SELECT animal.nombre AS nombre_animal, dieta.tipo_alimento, dieta.dosis, veterinario.nombre AS disenado_por
FROM animal
INNER JOIN dieta ON animal.id_dieta_animal = dieta.id_dieta
INNER JOIN informe_animal ON animal.id_animal = informe_animal.id_animal_informe
INNER JOIN veterinario_informe ON informe_animal.id_informe = veterinario_informe.id_informe_veterinario
INNER JOIN veterinario ON veterinario_informe.id_veterinario_informe = veterinario.id_veterinario;
```

Vista

Resultado vista

Retorna todos los animales con su dieta

nombre_animal	tipo_alimento	dosis	disenado_por
fish	aut	1319 gr	Michiko
snail	consequuntur	1962 gr	Margarette
wombat	et	7387 gr	Sydney
guinea pig	velit	9753 gr	Mercedez
mallard	aut	7293 gr	Desmond
goat	nihil	5261 gr	Demetrius
chinchilla	pariatur	2116 gr	Marylynn
porpoise	ipsum	5754 gr	Sammie
chimpanzee	sed	3344 gr	Shalon
swan	dolores	9642 gr	Carlo
hornet	quos	8170 gr	Shila
bat	architecto	9109 gr	Kyle
elk	asperiores	6439 gr	Wilton
porpoise	sapiente	6509 gr	Joe
tortoise	consequatur	2832 gr	Marcel
prairie dog	ut	1399 gr	Arlette
walrus	nulla	2770 gr	Debi
cheetah	non	8424 gr	Clotilde
louse	animi	8590 gr	Jazmine
goose	odit	7977 gr	Megan
jellyfish	ullam	6243 gr	Avelina
hound	sit	6422 gr	Lakendra
mole	animi	8001 gr	Oswaldo

1 13:12:25 SELECT * FROM zoodb.vista_animal_dieta_veterinario LIMIT 0, 1000 50 row(s) returned

Esta vista es importante porque permite una visualización clara de la dieta de cada animal y el veterinario que la diseñó. Esto puede ser de gran utilidad para el personal encargado del cuidado de la salud y el bienestar de los animales. La dieta de un animal es un aspecto clave para llevar el registro del consumo de alimentos de los animales, y conocer quién la diseñó puede ser útil para hacer seguimiento a las recomendaciones y tratamientos prescritos por el veterinario.

También es importante destacar que esta vista puede mejorar la eficiencia en la gestión de los registros y datos relacionados con la alimentación de los animales y la actividad de los veterinarios. Al contar con esta información en un solo lugar y de manera organizada, se puede agilizar el acceso a los datos y mejorar la eficiencia en la gestión de los registros.

2. Vista para ver el alimento y el proveedor de ese alimento

ZOO (Ejercicio B)

-- 2. Vista para ver el alimento y el proveedor de ese alimento

```
CREATE VIEW vista_alimento_proveedor AS
SELECT a.nombre_alimento, p.nombre as proveedor
FROM alimento a
JOIN proveedor_alimento pa ON a.id_alimento = pa.id_alimento_proveedor
JOIN proveedor p ON pa.id_proveedor_alimento = p.nit;
```

Vista

Resultado de la vista

Retorna los alimentos y sus proveedores

nombre_alimento	proveedor
Flounder	Geneva Mertz DVM
Anchovies	Blanche Hudson Jr.
Soymilk	Chadwick Kovacek
Nutritional Yeast	Bryant Lind
Cannellini Beans	Buford Brown
Oysters	Hai Breitenberg
Mullet	Ms. Maureen Wilechski
Anchovies	Mui Runolfsson
Tuna	Faustino Marks
Ricemilk	Jaimie Rath Jr.
Lotus Root	Ms. Charmain Moen
Kiwi Fruit	Alton Bins PhD
Smoked Trout	Vania Gislason
Cranberry	Mr. Deivey Zulauf

Output: 1 13:15:03 SELECT * FROM zoodb.vista_alimento_proveedor LIMIT 0, 1000 50 row(s) returned

Esta vista es importante porque permite conocer de manera rápida y sencilla el proveedor de cada alimento que se utiliza en la alimentación de los animales. Esto es útil para el personal encargado de la gestión de compras en el zoológico, ya que les permite saber con facilidad qué proveedor les suministra cada tipo de alimento y tomar decisiones más informadas a la hora de realizar nuevos pedidos o negociar acuerdos con los proveedores.

Además, esta vista también puede ser útil para el personal encargado de la gestión de la dieta de los animales, ya que les permite conocer de manera rápida y sencilla el origen del alimento utilizado en la dieta de cada animal y hacer seguimiento a la calidad de los alimentos suministrados por los proveedores.

3. Vista para ver el alimento que un animal consume diario según su dieta.

-- 3. Vista para ver el alimento que un animal consume diario según su dieta

```
CREATE VIEW vista_alimento_animal_según_dieta AS
SELECT a.id_animal, a.nombre as nombre_animal, d.tipo_alimento as tipo_alimento_que_consume, d.dosis, al.nombre_alimento
FROM animal a
INNER JOIN dieta d ON a.id_dieta_animal = d.id_dieta
INNER JOIN alimento_animal aa ON a.id_animal = aa.id_animal_alimento
INNER JOIN alimento al ON aa.id_alimento_animal = al.id_alimento;
```

ZOO (Ejercicio B)

The screenshot shows a database management tool interface. On the left, the 'SCHEMAS' pane lists various database objects, including 'vista_alimento_animal_segun_dieta', which is highlighted with a red box and labeled 'Vista'. The main pane displays the SQL query: `SELECT * FROM zoodb.vista_alimento_animal_segun_dieta;`. Below the query, the 'Result Grid' shows the data returned by the view. The data is organized into columns: `id_animal`, `nombre_animal`, `tipo_alimento_que_consume`, `dosis`, and `nombre_alimento`. The results show various animals and their daily food intake. A red box highlights the first few rows of the result grid, labeled 'Resultado vista'. Below the result grid, the 'Output' pane shows the execution details: `1 13:18:55 SELECT * FROM zoodb.vista_alimento_animal_segun_dieta LIMIT 0, 1000`, with a message indicating '50 row(s) returned'.

id_animal	nombre_animal	tipo_alimento_que_consume	dosis	nombre_alimento
ANI-0	fish	aut	1319 gr	Flounder
ANI-1	snail	consequuntur	1962 gr	Anchovies
ANI-10	wombat	et	7387 gr	Soymik
ANI-11	guinea pig	velit	9753 gr	Nutritional Yeast
ANI-12	mallard	aut	7293 gr	Cannellini Beans
ANI-13	goat	nihil	5261 gr	Oysters
ANI-14	chinchilla	pariatur	2116 gr	Mullet
ANI-15	porpoise	ipsum	5754 gr	Anchovies
ANI-16	chimpanzee	sed	3344 gr	Tuna
ANI-17	swan	dolores	9642 gr	Ricemilk
ANI-18	hornet	quos	8170 gr	Lotus Root
ANI-19	bat	architecto	9109 gr	Kiwi Fruit
ANI-2	elk	asperiores	6439 gr	Smoked Trout
ANI-20	porpoise	sapiente	6509 gr	Cranberry

Esta vista es importante porque permite a los usuarios del sistema obtener información clara y concisa sobre el alimento que consume cada animal en función de su dieta. Con esta vista, se puede visualizar rápidamente cuál es el alimento y la cantidad que un animal consume diariamente, lo que puede ser útil para el seguimiento de consumo de alimentos de los animales. Además, también puede ser útil para la toma de decisiones relacionadas con la compra y el suministro de alimentos para el zoológico.

4. Vista para ver el informe de todos los animales y el entrenador.

-- 4. Vista para ver el informe de todos los animales y el entrenador

```
CREATE VIEW vista_informe_de_animales_y_entrenador AS
SELECT animal.nombre AS nombre_animal, informe_animal.fecha_peso, informe_animal.peso_registrado, informe_animal.detalles, entrenador.
FROM animal
JOIN informe_animal ON animal.id_animal = informe_animal.id_animal_informe
JOIN entrenador ON informe_animal.id_entrenador_informe = entrenador.id_entrenador;
```

The screenshot shows a database management tool interface. On the left, the 'SCHEMAS' pane lists various database objects, including 'vista_informe_de_animales_y_entrenador', which is highlighted with a red box and labeled 'Vista'. The main pane displays the SQL query: `SELECT * FROM zoodb.vista_informe_de_animales_y_entrenador;`. Below the query, the 'Result Grid' shows the data returned by the view. The data is organized into columns: `nombre_animal`, `fecha_peso`, `peso_registrado`, `detalles`, `nombre_entrenador`, and `apellido_entrenador`. The results show various animals and their associated trainers. A red box highlights the first few rows of the result grid, labeled 'Resultado vista'. Below the result grid, the 'Output' pane shows the execution details: `1 13:21:36 SELECT * FROM zoodb.vista_informe_de_animales_y_entrenador LIMIT 0, 1000`, with a message indicating '50 row(s) returned'.

nombre_animal	fecha_peso	peso_registrado	detalles	nombre_entrenador	apellido_entrenador
fish	Sat Jan 01 17:12:39 COT 2022	1689 kg	5fby6vz67wvt7pwxvz0z7hvvub99c0vhojh...	Jeffrey	Ralfson
snail	Mon Feb 13 11:21:00 COT 2022	4043 kg	8p0a5pvhmkakoxzyz1gmbkt4fydag9ty6at6b...	Toney	Schultz
wombat	Tue Dec 13 07:42:24 COT 2022	2847 kg	rsdzesw03r040t4deishfudag70don2en5yym...	Marybelle	Streich
guinea pig	Fri Dec 23 05:03:11 COT 2022	8578 kg	23elfhv9k5n2mhpupus5dte65tc4mlic4cb64w...	Jaye	Orm
mallard	Thu May 05 05:45:42 COT 2022	8451 kg	vlyd9krtgud5cx8ydy67arjba7e5wvmsugud04...	Jeanne	Schiller
goat	Sat Dec 31 14:21:59 COT 2022	4001 kg	q7h0d3a2nos3p1c7hvg3ag9b0c3mhag9vq4...	Aurora	Crudshank
chinchilla	Wed Nov 30 06:59:51 COT 2022	5211 kg	gv64gn37erj6946ndvzcdqg9ud4fyu4fw...	Mary	Streich
porpoise	Wed Nov 02 21:05:05 COT 2022	6342 kg	d7bhdc54cd8ktu8mdidzqumca3ddu33kv99...	Shannon	Stroman
chimpanzee	Thu Jan 20 07:31:41 COT 2022	4251 kg	gn53yutakifea8d4pbm3avng5fudbettwmswcv...	Casey	Rath
swan	Tue Mar 15 03:37:54 COT 2022	5885 kg	626mm472ogf4abgvh7zbhmcrrffbtw51786...	Stasia	Emard
hornet	Thu Sep 22 12:25:09 COT 2022	4827 kg	crd0t88a9fn5ypenbshiofy7qzscy604vb30vgrx...	Jonah	Balistreri
bat	Sun Aug 21 17:28:21 COT 2022	3182 kg	yfez8bv62bm9r73ohymd6e4q4k46182w0n2...	China	Brown
elk	Wed Nov 23 11:53:39 COT 2022	5016 kg	x1qeus6vz1pmfb23ume903vh96vfbvz70emqy...	Vicente	Rowe
porpoise	Fri May 19 06:19:58 COT 2022	3746 kg	e20puyvz8t9d8vllvuvn10t0144dofa96v3s8b2c...	Kara	Goodwin

ZOO (Ejercicio B)

Esta vista es importante porque permite ver de manera sencilla el registro del peso de todos los animales y los detalles de su progreso, así como el entrenador responsable de dicho registro. Esta información puede ser muy útil para llevar un control detallado de la evolución de los animales, y para identificar patrones o tendencias que puedan ayudar a optimizar su cuidado. Además, esta vista puede ser de utilidad para los entrenadores, quienes pueden acceder rápidamente a la información de los animales que están a su cargo, los veterinarios, también podría acceder a ella para diseñar las dietas de los animales y si tienen dudas de algún registro sabrán a que entrenador dirigirse.

5. Vista para ver la cantidad de alimento existente según tipo de alimento, y total de alimento diario Requerido según dietas diseñadas.

```
-- 5. Vista para ver la cantidad de alimento existente segun tipo de alimento,
-- y total de alimento diario Requerido segun dietas diseñadas

CREATE VIEW vista_cantidad_alimento_existente_comparando_dieta_necesaria AS
SELECT dieta.tipo_alimento, SUM(dieta.dosis) AS total_alimento_diario_requerido,
      (SELECT SUM(cantidad) FROM alimento WHERE alimento.tipo_alimento = dieta.tipo_alimento) AS cantidad_existente
FROM dieta
WHERE dieta.tipo_alimento IN (SELECT tipo_alimento FROM alimento)
GROUP BY dieta.tipo_alimento;
```

The screenshot shows a database management interface. On the left, a 'SCHEMAS' pane lists various database objects, with 'vista_cantidad_alimento_existente_compara' highlighted. A red arrow points from this view name to the word 'Vistas'. The main pane displays a SQL query: 'SELECT * FROM zoodb.vista_cantidad_alimento_existente_comparando_dieta_necesaria;'. Below the query, the 'Result Grid' shows the output of the view. A red box highlights the result table, with a red arrow pointing to it labeled 'Resultado vista'. The table has three columns: 'tipo_alimento', 'total_alimento_diario_requerido', and 'cantidad_existente'. The data rows are as follows:

tipo_alimento	total_alimento_diario_requerido	cantidad_existente
consequuntur	1962	2051
et	10741	6806
nihil	5261	3349
quos	14797	32
consequatur	2832	580
ut	9738	3379
nulla	2770	2642
non	8424	6067
sit	8494	1712
ipsam	8603	927
autem	1048	299
explicabo	2176	3264
voluptatibus	1262	1786
sequi	5347	1899

At the bottom, the 'Action Output' pane shows a message: '1 13:24:31 SELECT * FROM zoodb.vista_cantidad_alimento_existente_comparando_dieta_necesaria LIMIT 0, 1000 14 row(s) returned'. A red arrow points from the text 'Retorna los diferentes tipos de alimentos asignados a las dietas y la cantidad de alimento diario requerido y la cantidad existente' to the result table.

Esta vista es importante porque permite tener una visión clara de la cantidad de alimento existente según el tipo de alimento y la cantidad necesaria según la dieta de los animales en el zoológico. Esto es fundamental para garantizar que la cantidad de alimento existente de un tipo sea coherente con las dietas diseñadas por el veterinario.

Al tener esta información disponible en una vista, los empleados logísticos encargados de realizar las órdenes de compra pueden tomar decisiones informadas sobre la cantidad de alimento que deben comprar y suministrar a los animales para cumplir con los requisitos de la dieta. También **pueden identificar fácilmente si hay alguna escasez o exceso de algún tipo de alimento, y tomar medidas para corregir la situación antes de que afecte la salud de los animales.**

ZOO (Ejercicio B)

Además, esta vista también puede ser útil para fines de planificación y presupuestación, ya que permite estimar con precisión los costos de alimentación de los animales en función de las dosis requeridas por las diferentes dietas y los precios de los diferentes tipos de alimentos.

Procedimientos

1. Obtener proveedores por tipo de alimento

```
-- 1. Obtener proveedores por tipo de alimento

DELIMITER //
CREATE PROCEDURE obtener_proveedores_por_tipo_alimento(IN tipo_alimento VARCHAR(45))
BEGIN
    SELECT p.nombre, p.telefono, p.correo, p.direccion
    FROM proveedor p
    JOIN proveedor_alimento pa ON p.nit = pa.id_proveedor_alimento
    JOIN alimento a ON pa.id_alimento_proveedor = a.id_alimento
    WHERE a.tipo_alimento = tipo_alimento;
END //
DELIMITER ;

CALL obtener_proveedores_por_tipo_alimento('non');
```

The screenshot shows a database management tool interface. On the left, the 'SCHEMAS' pane lists various database objects, with 'obtener_proveedores_por_tipo_alimento' highlighted under the 'Functions' section. An arrow points from this function to the main editor area. In the main editor, the SQL code for the procedure is displayed, with the call statement 'CALL zoodb.obtener_proveedores_por_tipo_alimento('non');' highlighted. A red arrow points from this call statement to the 'Result Grid' pane. The 'Result Grid' pane shows the output of the procedure, which is a table with 5 columns: 'nombre', 'telefono', 'correo', and 'direccion'. The table contains 3 rows of data. A red arrow points from the 'Result Grid' to the 'Action Output' pane. The 'Action Output' pane shows the execution details of the procedure, including the time taken and the number of rows returned.

Como ejemplo se utiliza el tipo de alimento 'n'

Procedimiento

nombre	telefono	correo	direccion
Geneva Mertz DVM	55593	emory.bogisch@hotmail.com	Renée Parkways
Blanche Hudson Jr.	55556	vito.schimmel@yahoo.com	Collins Mills
Ms. Jeremy Kahlerin	55549	marnie.roberts@gmail.com	Reichel Glens

Resultado del procedimiento

Retorna 3 filas

1 13:32:29 call zoodb.obtener_proveedores_por_tipo_alimento('non') 3 row(s) returned

2. Agregar nueva dieta

ZOO (Ejercicio B)

-- 2. Agregar nueva dieta

DELIMITER //

```
CREATE PROCEDURE agregar_nueva_dieta(  
    dieta_id VARCHAR(20),  
    alimento_tipo VARCHAR(45),  
    alimentoDosis VARCHAR(45)  
)
```

BEGIN

```
    INSERT INTO dieta (id_dieta, tipo_alimento, dosis) VALUES (dieta_id, alimento_tipo, alimentoDosis) ;
```

END//

DELIMITER ;

```
call agregar_nueva_dieta('DIE-006', 'SECA', '500 gr');
```

The screenshot shows a database management interface with a left sidebar containing a 'SCHEMAS' tree. In this tree, the 'Procedimientos' (Procedures) folder is expanded, and the procedure 'agregar_nueva_dieta' is highlighted with a red box. A red arrow points from this box to the word 'Procedimiento' in the text below. The main window displays the execution of the procedure 'call zoodb.agregar_nueva_dieta('DIE-006', 'SECA', '500 gr');'. A red box highlights this call, with a red arrow pointing to it from the text 'Datos de la dieta a agregar' below. Below the main window, the 'Output' tab is selected, showing a table with one row: '1 13:37:59 call zoodb.agregar_nueva_dieta(DIE-006, 'SECA', '500 gr') 1 row(s) affected'. A red arrow points from the text 'Se adiciona la nueva dieta' to this row.

Procedimiento

Datos de la dieta a agregar

Se adiciona la nueva dieta

3. Actualizar dieta de un animal

-- 3. Actualizar dieta de un animal

DELIMITER //

```
CREATE PROCEDURE actualizar_dieta_animal(  
    idAnimal VARCHAR(20),  
    id_dietaAnimal VARCHAR(20)  
)
```

BEGIN

```
    UPDATE animal SET id_dieta_animal= id_dietaAnimal WHERE id_animal = idAnimal ;
```

END//

DELIMITER ;

ZOO (Ejercicio B)

The screenshot displays the SQL Server Enterprise Manager interface. On the left, the 'SCHEMAS' pane shows the 'veterinario_informe' database with various views and stored procedures. The 'actualizar_dieta_animal' stored procedure is highlighted. In the center, the 'VistasSQL ZOO' pane shows the execution of the stored procedure with the following SQL code:

```
call zoodb.actualizar_dieta_animal('ANI-12', 'DIE-006');
```

Below the code, the 'Output' pane shows the results of the execution:

Time	Message
1 13:46:45	call zoodb.actualizar_dieta_animal(ANI-12; 'DIE-006)

Red arrows point from the procedure name in the left pane to the code in the center, and from the code to the output message. A red arrow also points from the output message to the text '1 fila afectada'.

4. Eliminar una factura

```
-- -----  
-- 4. Eliminar una factura  
-- -----  
  
DELIMITER //  
CREATE PROCEDURE eliminar_factura_saldada(  
    IN factura_id VARCHAR(20)  
)  
BEGIN  
    DELETE FROM factura  
    WHERE id_factura = factura_id;  
END//  
DELIMITER ;
```

ZOO (Ejercicio B)

The screenshot shows a database management tool interface. On the left, the 'SCHEMAS' pane lists various database objects, with 'eliminar_factura_saldada' highlighted under the 'Stored Procedures' section. A red arrow points from this procedure name to the main query editor. The query editor contains the following SQL statement:

```
1 call zoodb.eliminar_factura_saldada('FAC131');
2
```

A red arrow points from the value 'FAC131' in the query to the text 'Id de la factura a eliminar'. Below the query editor, the 'Output' pane shows the 'Action Output' table with the following data:

#	Time	Action	Message
1	13:50:07	call zoodb.eliminar_factura_saldada(FAC131)	1 row(s) affected

A red arrow points from the 'Message' column to the text 'Una fila afectada'.

Triggers

```
-- Crear la tabla control de cambios ZOO

CREATE TABLE control_de_cambios_zoo (
    usuario VARCHAR(45),
    accion VARCHAR(20),
    fecha DATETIME
);

-- 1. Triggers Insertar

DELIMITER //
CREATE TRIGGER control_zoo_insert
AFTER INSERT ON dieta
FOR EACH ROW
BEGIN
    INSERT INTO control_de_cambios_zoo (usuario, accion, fecha)
    VALUES (user(), 'insertar', NOW());
END;
//
```

ZOO (Ejercicio B)

The screenshot shows the SQL Studio interface. On the left, the 'SCHEMAS' pane displays a tree view of the database structure, including 'Views' and 'Stored Procedures'. The 'Stored Procedures' list includes 'agregar_nueva_dieta', which is highlighted. The main editor shows the execution of the stored procedure: `call zoodb.agregar_nueva_dieta('DIE-0103', 'HUMEDO', '1000 gr');`. A red box highlights this code, and a red arrow points to it with the text 'Datos para agregar dieta'. Below the editor, the 'Output' pane shows the 'Action Output' table with one row:

#	Time	Action	Message
1	13:54:34	call zoodb.agregar_nueva_dieta('DIE-0103', 'HUMEDO', '1000 gr')	1 row(s) affected

. A red arrow points to this row with the text 'Se agrega la dieta'.

The screenshot shows the SQL Studio interface. On the left, the 'SCHEMAS' pane displays a tree view of the database structure, including 'Tables'. The 'control_de_cambios_zoo' table is highlighted. The main editor shows the execution of the query: `SELECT * FROM zoodb.control_de_cambios_zoo;`. Below the editor, the 'Result Grid' shows the query results:

usuario	accion	fecha
root@localhost	insertar	2023-02-19 13:54:34

. A red box highlights this result, and a red arrow points to it with the text 'El trigger se dispara y guarda el usuario y fecha con la hora en que se inserto un nuevo registro'. Below the result grid, the 'Output' pane shows the 'Action Output' table with two rows:

#	Time	Action	Message
1	13:54:34	call zoodb.agregar_nueva_dieta('DIE-0103', 'HUMEDO', '1000 gr')	1 row(s) affected
2	13:55:59	SELECT * FROM zoodb.control_de_cambios_zoo LIMIT 0, 1000	1 row(s) returned

.

-- 2. Trigger Eliminar

```
DELIMITER //
```

```
CREATE TRIGGER control_zoo_delete_factura
```

```
AFTER DELETE ON factura
```

```
FOR EACH ROW
```

```
BEGIN
```

```
INSERT INTO control_de_cambios_zoo (usuario, accion, fecha)
```

```
VALUES (user(), 'eliminar', NOW());
```

```
END;
```

```
//
```


ZOO (Ejercicio B)

The screenshot shows a database IDE with the 'Schemas' panel on the left. The 'eliminar_factura_saldada' procedure is selected. The main editor shows the SQL command: `call zoodb.eliminar_factura_saldada('FAC141');`. A red arrow points from the text 'Id de la factura a eliminar' to the value 'FAC141'. Below the editor, the 'Output' panel shows the execution result: `1 13:59:11 cal zoodb.eliminar_factura_saldada(FAC141) 1 row(s) affected`. A red arrow points from the text 'Factura eliminada' to this output line. A red box highlights the procedure name in the 'Schemas' panel, with a red arrow pointing to the text 'Se elimina factura'.

The screenshot shows the same database IDE. The 'Schemas' panel on the left has 'control_de_cambios_zoo' selected. The main editor shows the SQL command: `SELECT * FROM zoodb.control_de_cambios_zoo;`. Below the editor, the 'Result Grid' shows the execution result with columns 'usuario', 'accion', and 'fecha'. The data rows are: `root@localhost insertar 2023-02-19 13:54:34` and `root@localhost eliminar 2023-02-19 13:59:11`. A red arrow points from the text 'Tabla de control de cambios' to the 'control_de_cambios_zoo' table in the 'Schemas' panel. Another red arrow points from the text 'El trigger se dispara al eliminarse una factura' to the second row of the 'Result Grid'. Below the 'Result Grid', the 'Output' panel shows the execution result: `1 13:59:11 call zoodb.eliminar_factura_saldada(FAC141) 1 row(s) affected` and `2 14:00:50 SELECT * FROM zoodb.control_de_cambios_zoo LIMIT 0, 1000 2 row(s) returned`.

-- 3. Trigger Actualizar Dieta Animal

```
DELIMITER //
CREATE TRIGGER control_zoo_actualizar_dieta
AFTER UPDATE ON animal
FOR EACH ROW
BEGIN
INSERT INTO control_de_cambios_zoo (usuario, accion, fecha)
VALUES (user(), 'actualizar', NOW());
END;
//
```

ZOO (Ejercicio B)

The screenshot shows a SQL IDE interface. On the left, the 'SCHEMAS' panel displays a tree view of the database structure. The 'zoo' database is expanded, showing tables like 'animal', 'dieta', and 'entrenador'. A red box highlights the 'actualizar_dieta_animal' procedure. A red arrow points from this procedure to the main query editor. In the query editor, the following SQL statement is entered: `call zoodb.actualizar_dieta_animal('ANI-13', 'DIE-22');`. A red box highlights this statement, and a red arrow points to it with the text 'datos a actualizar'. Below the query editor, the 'Output' panel shows the execution results. A red box highlights the second row of the output, which indicates that 1 row was affected. A red arrow points to this row with the text 'Se afecta una fila'.

se utiliza procedimiento para actualizar la dieta animal

datos a actualizar

Se afecta una fila

The screenshot shows a SQL IDE interface. On the left, the 'SCHEMAS' panel displays a tree view of the database structure. The 'zoo' database is expanded, showing tables like 'animal', 'dieta', and 'entrenador'. A red box highlights the 'control_de_cambios_zoo' table. A red arrow points from this table to the main query editor. In the query editor, the following SQL statement is entered: `SELECT * FROM zoodb.control_de_cambios_zoo;`. A red box highlights this statement, and a red arrow points to it with the text 'Tabla de control de cambios'. Below the query editor, the 'Result Grid' panel shows the execution results. A red box highlights the third row of the result grid, which indicates that the trigger was fired. A red arrow points to this row with the text 'Se dispara el trigger cuando se actualiza la dieta de un animal'. Below the result grid, the 'Output' panel shows the execution results. A red box highlights the third row of the output, which indicates that 3 rows were returned. A red arrow points to this row with the text 'Se dispara el trigger cuando se actualiza la dieta de un animal'.

Tabla de control de cambios

Se dispara el trigger cuando se actualiza la dieta de un animal

-- 4. Trigger Agregar nuevo Animal

```
DELIMITER //
```

```
CREATE TRIGGER control_zoo_insertar_animal
```

```
AFTER INSERT ON animal
```

```
FOR EACH ROW
```

```
BEGIN
```

```
INSERT INTO control_de_cambios_zoo (usuario, accion, fecha)
```

```
VALUES (user(), 'insertar', NOW());
```

```
END;
```

```
//
```

ZOO (Ejercicio B)

The screenshot shows a database IDE with the following components:

- SQL Editor:** Contains the query `USE zoodb;` followed by an `INSERT INTO animal` statement. A red box highlights the `INSERT INTO` statement, with a red arrow pointing to it and the text "Se agrega un nuevo animal".
- Action Output:** Shows the execution result of the `INSERT` statement. A red box highlights the output message "1 14:08:58 INSERT INTO animal (id_animal, nombre, clase, especie, fecha_nacimiento, id_dieta_animal) VALUES ('ANI-004', 'Animal 3', 'Clase 1', 'Especie 1', '2021-02-01', 'DIE-006'); 1 row(s) affected", with a red arrow pointing to it and the text "Se añade una fila".
- Navigator:** Shows the database schema. The table `control_de_cambios_zoo` is highlighted with a red box.
- Result Grid:** Shows the output of a `SELECT * FROM zoodb.control_de_cambios_zoo;` query. A red box highlights the first row of the result set, with a red arrow pointing to it and the text "Se dispara el trigger al agregar un nuevo animal".
- Output:** Shows the execution result of the `SELECT` statement. A red box highlights the output message "2 14:10:54 SELECT * FROM zoodb.control_de_cambios_zoo LIMIT 0, 1000 4 row(s) returned".

Base de datos poblada con conexión desde Java

Para poblar la base de datos con 50 registros cada tabla, se utilizó java con las clases y métodos propuestos por el coach Juan Pineda. Se crearon los modelos de cada una de las tablas y se utilizó java faker que es una librería de generación de datos falsos que se utiliza comúnmente en pruebas y prototipos de software. Proporciona una forma sencilla de crear datos de prueba realistas en una variedad de formatos, como nombres, direcciones, fechas, números de teléfono, direcciones de correo electrónico, entre otros.

Para la creación de cada registro se utiliza un for i, dentro del for se realiza el set y el insert de los atributos.

En algunas de las tablas (como se señalará en las próximas imágenes adjuntas) se retorna listas para usar su contenido como FK en otras tablas.

14 usages

```
public static int n = 50;
```

n, es el numero utilizado en el for para indica el numero de registros for(i = 0; i < n; i++)

Método para poblar tabla de proveedores

ZOO (Ejercicio B)

```
private static List<String> generarProveedores() {  
  
    String insertProveedor = "";  
    for (int i = 0; i < n; i++) {  
        Proveedor proveedor = new Proveedor();  
        Faker faker = new Faker();  
  
        proveedor.setNit("NIT-" + i);  
        proveedor.setNombre(faker.name().name());  
        proveedor.setTelefono("555" + faker.numerify(numberStrings: "##"));  
        proveedor.setCorreo(faker.internet().emailAddress());  
        proveedor.setDireccion(faker.address().streetName());  
  
        insertProveedor = "INSERT INTO proveedor (nit, nombre, telefono, correo, direccion) VALUES('%s', '%s', '%s', '%s', '%s')";  
        insertProveedor = String.format(insertProveedor,  
            proveedor.getNit(),  
            proveedor.getNombre(),  
            proveedor.getTelefono(),  
            proveedor.getCorreo(),  
            proveedor.getDireccion());  
  
        mysqlOperation.setSqlStatement(insertProveedor);  
        mysqlOperation.executeSqlStatementVoid();  
  
        nit_proveedores.add(proveedor.getNit());  
    }  
    return nit_proveedores;  
}
```

Retorna los ids de los proveedores como una lista y as
usarlos en otra tablas como FK

Retorno de ids

```
1 • USE zoodb;  
2  
3 • SELECT * FROM Proveedor;  
4 • SELECT * FROM factura;  
5 • SELECT * FROM logistico;  
6 • SELECT * FROM entrenador;  
7 • SELECT * FROM orden_compra;  
8 • SELECT * FROM alimento;  
9 • SELECT * FROM proveedor_alimento;  
10 • SELECT * FROM dieta;  
11 • SELECT * FROM animal;  
12 • SELECT * FROM alimento_animal;
```

Trae la tabla proveedores

Result Grid

nit	nombre	telefono	correo	direccion
NIT-0	Geneva Mertz DVM	55593	emory.bogisich@hotmail.com	Renea Parkways
NIT-1	Blanche Hudson Jr.	55556	vito.schimmel@yahoo.com	Collins Mills
NIT-10	Chadwick Kovacek	55593	oretha.simonis@yahoo.com	Theodore Well
NIT-11	Bryant Lind	55570	leatha.zieme@gmail.com	Tillman Crossing
NIT-12	Buford Brown	55514	bernie.tromp@hotmail.com	Kub Plaza
NIT-13	Hai Breitenberg	55579	derek.heidenreich@yahoo.com	Keshia Drive
NIT-14	Ms. Maureen Waelchi	55506	earnestine.nollich@hotmail.com	Kihn Port

Tabla proveedor

Retorna 50 filas

1 11:23:52 SELECT * FROM Proveedor LIMIT 0, 1000 50 row(s) returned

Método para poblar la tabla facturas

ZOO (Ejercicio B)

```
private static void generarFacturas(List<String> nits_proveedores) {  
  
    String insertFactura = "";  
  
    for (int i = 0; i < n ; i++) {  
        Factura factura= new Factura();  
        Faker faker = new Faker();  
  
        factura.setId_factura("FAC" + i+1);  
        factura.setTotal("$" + faker.random().nextInt(100, 10000));  
        factura.setNit_proveedor_factura(nits_proveedores.get(i));  
  
        insertFactura = "INSERT INTO factura (id_factura, total, nit_proveedor_factura) VALUES('%s', '%s', '%s')";  
        insertFactura = String.format(insertFactura,  
            factura.getId_factura(),  
            factura.getTotal(),  
            factura.getNit_proveedor_factura());  
  
        mySqlOperation.setSqlStatement(insertFactura);  
        mySqlOperation.executeSqlStatementVoid();  
    }  
}
```

Se utilizan los ids de los proveedores como FK

```
1 • USE zoodb;  
2  
3 • SELECT * FROM Proveedor;  
4 • SELECT * FROM factura;  
5 • SELECT * FROM logistico;  
6 • SELECT * FROM entrenador;  
7 • SELECT * FROM orden_compra;  
8 • SELECT * FROM alimento;  
9 • SELECT * FROM proveedor_alimento;  
10 • SELECT * FROM dieta;  
11 • SELECT * FROM animal;  
12 • SELECT * FROM alimento_animal;  
13 • SELECT * FROM informe_animal;  
14 • SELECT * FROM veterinario;  
15 • SELECT * FROM veterinario_informe;
```

Trae la tabla factura

id_factura	total	nit_proveedor_factura
FAC01	\$4945	NIT-0
FAC101	\$2459	NIT-10
FAC11	\$5072	NIT-1
FAC111	\$2115	NIT-11
FAC121	\$1035	NIT-12
FAC131	\$3128	NIT-13
FAC141	\$8178	NIT-14
FAC151	\$5603	NIT-15
FAC161	\$6343	NIT-16

Tabla factura

Retorna filas

#	Time	Action	Message
1	11:29:11	SELECT * FROM factura LIMIT 0, 1000	50 row(s) returned

Método para poblar tabla logístico

ZOO (Ejercicio B)

```
private static List<String> generarLogisticos() {  
    String insertLogistico = "";  
    for (int i = 0; i < n ; i++) {  
        Logistico logistico = new Logistico();  
        Faker faker = new Faker();  
  
        logistico.setId_logistico("LOG-" + i);  
        logistico.setNombre(faker.name().firstName());  
        logistico.setApellido(faker.name().lastName());  
        logistico.setTelefono("111" + faker.numerify( numberString: "##"));  
  
        insertLogistico = "INSERT INTO logistico (id_logistico, nombre, apellido, telefono) VALUES('%s', '%s', '%s', '%s' );";  
        insertLogistico = String.format(insertLogistico,  
            logistico.getId_logistico(),  
            logistico.getNombre(),  
            logistico.getApellido(),  
            logistico.getTelefono());  
  
        mysqlOperation.setSqlStatement(insertLogistico);  
        mysqlOperation.executeSqlStatementVoid();  
  
        id_logisticos.add(logistico.getId_logistico());  
    }  
    return id_logisticos;  
}
```

Metodo que devuelve una lista

Retorna los ids de logistico para usarlos como FK en otras tabla:

Activar Windows

```
1 • USE zoodb;  
2  
3 • SELECT * FROM Proveedor;  
4 • SELECT * FROM factura;  
5 • SELECT * FROM logistico;  
6 • SELECT * FROM entrenador;  
7 • SELECT * FROM orden_compra;  
8 • SELECT * FROM alimento;  
9 • SELECT * FROM proveedor_alimento;  
10 • SELECT * FROM dieta;  
11 • SELECT * FROM animal;  
12 • SELECT * FROM alimento_animal;  
13 • SELECT * FROM informe_animal;  
14 • SELECT * FROM veterinario;  
15 • SELECT * FROM veterinario_informe;
```

Trae la tabla logistico

Result Grid

id_logistico	nombre	apellido	telefono
LOG-0	Francene	Lindgren	11179
LOG-1	Marcelino	Quigley	11143
LOG-10	Philip	Leannon	11141
LOG-11	Ricardo	Toy	11161
LOG-12	Edith	McDermott	11158
LOG-13	Korey	Price	11182
LOG-14	Wilburn	Prosacco	11174
LOG-15	Scottie	Russel	11128
LOG-16	Jayne	Howe	11156

Tabla Logistico

Devuelve 50 filas

Output

Action Output

#	Time	Action	Message
1	11:33:33	SELECT * FROM logistico LIMIT 0, 1000	50 row(s) returned

Método para poblar tabla entrenadores

ZOO (Ejercicio B)

```
private static List<String> generarEntrenadores() {  
    String insertEntrenador = "";  
  
    for (int i = 0; i < n ; i++) {  
  
        Entrenador entrenador = new Entrenador();  
  
        Faker faker = new Faker();  
  
        entrenador.setId_entrenador("ENT-" + i);  
        entrenador.setNombre_entrenador(faker.name().firstName());  
        entrenador.setApellido_entrenador(faker.name().lastName());  
        entrenador.setTelefono("555" + faker.numerify( numberStrings: "##"));  
  
        insertEntrenador = "INSERT INTO entrenador (id_entrenador, nombre, apellido, telefono) VALUES('%s', '%s', '%s', '%s')";  
        insertEntrenador = String.format(insertEntrenador,  
            entrenador.getId_entrenador(),  
            entrenador.getNombre_entrenador(),  
            entrenador.getApellido_entrenador(),  
            entrenador.getTelefono());  
  
        mySqlOperation.setSqlStatement(insertEntrenador);  
        mySqlOperation.executeSqlStatementVoid();  
  
        id_entrenadores.add(entrenador.getId_entrenador());  
    }  
    return id_entrenadores;  
}
```

Metodo que devuelve una lista

Retorna los ids de entrenadores

```
1 • USE zoodb;  
2  
3 • SELECT * FROM Proveedor;  
4 • SELECT * FROM factura;  
5 • SELECT * FROM logistico;  
6 • SELECT * FROM entrenador;  
7 • SELECT * FROM orden_compra;  
8 • SELECT * FROM alimento;  
9 • SELECT * FROM proveedor_alimento;  
10 • SELECT * FROM dieta;  
11 • SELECT * FROM animal;  
12 • SELECT * FROM alimento_animal;  
13 • SELECT * FROM informe_animal;  
14 • SELECT * FROM veterinario;  
15 • SELECT * FROM veterinario_informe;  
16 • SELECT * FROM veterinario_dieta;
```

Trae la tabla entrenador

id_entrenador	nombre	apellido	telefono
ENT-0	Jeffrey	Rolfson	55590
ENT-1	Toney	Schultz	55524
ENT-10	Marybelle	Streich	55596
ENT-11	Jaye	Orn	55567
ENT-12	Jeane	Schiller	55505
ENT-13	Aurora	Cruidshank	55596
ENT-14	Mary	Streich	55513
ENT-15	Shannon	Stroman	55517

Table entrenador

Retorna 50 filas

Output

#	Time	Action	Message
1	12:07:50	SELECT * FROM entrenador LIMIT 0, 1000	50 row(s) returned

Método para poblar tabla orden_compra

ZOO (Ejercicio B)

```
private static void generarOrdenCompra(List<String> nits_proveedores, List<String> ids_logisticos) {  
  
    String insertOrdenCompra = "";  
  
    for (int i = 0; i < n; i++) {  
        OrdenCompra ordenCompra = new OrdenCompra();  
        Faker faker = new Faker();  
  
        ordenCompra.setId_orden("ORD-" + i);  
        ordenCompra.setFecha_emision(String.valueOf(faker.date().between(Date.from(LocalDate.of(year: 2000, month: 1, dayOfMonth: 1)).atTime(0, 0, 0)), Date.from(LocalDate.of(year: 2000, month: 1, dayOfMonth: 1)).atTime(0, 0, 0)));  
        ordenCompra.setFecha_entrega(String.valueOf(faker.date().between(Date.from(LocalDate.of(year: 2000, month: 1, dayOfMonth: 1)).atTime(0, 0, 0)), Date.from(LocalDate.of(year: 2000, month: 1, dayOfMonth: 1)).atTime(0, 0, 0)));  
        ordenCompra.setDescripcion(faker.lorem().characters(fixedNumberOfCharacters: 45));  
        ordenCompra.setNit_proveedor_orden(nits_proveedores.get(i));  
        ordenCompra.setId_logistico_orden(ids_logisticos.get(i));  
  
        insertOrdenCompra = "INSERT INTO orden_compra (id_orden, fecha_emision, fecha_entrega, descripcion, nit_proveedor_orden, id_logistico_orden) VALUES (" +  
            ordenCompra.getId_orden() + ", " +  
            ordenCompra.getFecha_emision() + ", " +  
            ordenCompra.getFecha_entrega() + ", " +  
            ordenCompra.getDescripcion() + ", " +  
            ordenCompra.getNit_proveedor_orden() + ", " +  
            ordenCompra.getId_logistico_orden() + ");  
  
        mysqlOperation.setSqlStatement(insertOrdenCompra);  
        mysqlOperation.executeSqlStatementVoid();  
    }  
}
```

```
1 • USE zoodb;  
2  
3 • SELECT * FROM Proveedor;  
4 • SELECT * FROM factura;  
5 • SELECT * FROM logistico;  
6 • SELECT * FROM entrenador;  
7 • SELECT * FROM orden_compra;  
8 • SELECT * FROM alimento;  
9 • SELECT * FROM proveedor_alimento;  
10 • SELECT * FROM dieta;  
11 • SELECT * FROM animal;  
12 • SELECT * FROM alimento_animal;  
13 • SELECT * FROM informe_animal;  
14 • SELECT * FROM veterinario;  
15 • SELECT * FROM veterinario_informe;  
16 • SELECT * FROM veterinario_dieta;
```

Trae la tabla orden_compra

Tabla orden_compra

id_orden	fecha_emision	fecha_entrega	descripcion	nit_proveedor_orden	id_logistico_orden
ORD-0	Sat Feb 29 20:14:01 COT 2020	Wed Nov 14 12:23:22 COT 2018	v7wgjl1oyy0ndr2okbl4rof9odrd0t436hynxmdvr2c6r	NIT-0	LOG-0
ORD-1	Wed Oct 08 07:52:40 COT 2014	Tue Oct 06 09:39:09 COT 2020	7e0dxj4ke4vghl8ptgxcgj61vhudfcmh0hte7mt630a	NIT-1	LOG-1
ORD-10	Sun Sep 11 09:37:26 COT 2005	Mon Apr 23 21:12:55 COT 2012	rwkdvylznurpzi6ajvmnpf4whle95wm0plwbox1s5ikre3	NIT-10	LOG-10
ORD-11	Fri Apr 28 02:09:47 COT 2006	Thu Jan 07 06:00:29 COT 2010	a16qeo8rm4spuceu7jjaq5sk6l6ojf86jo1wu32fz23x	NIT-11	LOG-11
ORD-12	Tue Sep 30 05:12:26 COT 2003	Tue Aug 21 18:16:08 COT 2018	rpmukktjtp00q14130ja75v60h7ktbpps9h32f3d5p	NIT-12	LOG-12
ORD-13	Mon Sep 12 15:13:23 COT 2022	Sun May 31 18:05:29 COT 2020	r4fvlbtss50urv6c5j0h2vf1d8fgmgfedondxqwuadr	NIT-13	LOG-13
ORD-14	Wed Oct 13 04:23:47 COT 2021	Wed Jan 16 16:28:42 COT 2008	dfj8vtj56c4lp1duz68vjhh2k890gwx5ts7qbsv1b7wii	NIT-14	LOG-14
ORD-15	Mon Sep 27 11:35:01 COT 2004	Fri Jul 20 07:12:29 COT 2018	09vis8loans8ln2w8m1m4i40anknznfm15e0itrdx8id	NIT-15	LOG-15

Retorna 50 filas

#	Time	Action	Message
1	12:09:52	SELECT * FROM orden_compra LIMIT 0, 1000	50 row(s) returned

Método para poblar tabla alimento.

ZOO (Ejercicio B)

```
private static List<String> generarAlimentos() {  
    String insertAlimento = "";  
    for (int i = 0; i < n ; i++) {  
        Alimento alimento = new Alimento();  
        Faker faker = new Faker();  
  
        alimento.setId_alimento("ALI-" + i);  
        alimento.setNombre_alimento(faker.food().ingredient());  
        alimento.setTipo_alimento(faker.lorem().word());  
        alimento.setCantidad(String.valueOf(faker.number().numberBetween(10, 5000)) + " kg");  
  
        insertAlimento = "INSERT INTO alimento (id_alimento, nombre_alimento, tipo_alimento, cantidad) VALUES('%s', '%s', '%s', '%s')";  
        insertAlimento = String.format(insertAlimento, alimento.getId_alimento(), alimento.getNombre_alimento(), alimento.getTipo_alimento(), alimento.getCantidad());  
  
        mySqlOperation.setSqlStatement(insertAlimento);  
        mySqlOperation.executeSqlStatementVoid();  
  
        id_alimentos.add(alimento.getId_alimento());  
    }  
    return id_alimentos;  
}
```

Metodo que devuelve una lista

Retorna una lista de id de alimentos para usarlos como F

```
1 • USE zoodb;  
2  
3 • SELECT * FROM Proveedor;  
4 • SELECT * FROM factura;  
5 • SELECT * FROM logistico;  
6 • SELECT * FROM entrenador;  
7 • SELECT * FROM orden_compra;  
8 • SELECT * FROM alimento;  
9 • SELECT * FROM proveedor_alimento;  
10 • SELECT * FROM dieta;  
11 • SELECT * FROM animal;  
12 • SELECT * FROM alimento_animal;  
13 • SELECT * FROM informe_animal;  
14 • SELECT * FROM veterinario;  
15 • SELECT * FROM veterinario_informe;  
16 • SELECT * FROM veterinario_dieta;
```

Trae la tabla alimento

Result Grid

id_alimento	nombre_alimento	tipo_alimento	cantidad
ALI-0	Flounder	non	3087 kg
ALI-1	Anchovies	non	286 kg
ALI-10	Soymilk	nostrum	454 kg
ALI-11	Nutritional Yeast	inventore	608 kg
ALI-12	Cannellini Beans	in	3981 kg
ALI-13	Oysters	doloribus	2971 kg
ALI-14	Mullet	nihil	1730 kg
ALI-15	Anchovies	ut	3379 kn

Tabla alimento

Retorna 50 filas

1 12:12:24 SELECT * FROM alimento LIMIT 0, 1000 50 row(s) returned

Método para poblar tabla proveedor_alimento

ZOO (Ejercicio B)

```
private static void generarProveedorAlimento(List<String> nit_proveedores, List<String> id_alimentos) {  
  
    String insertProveedorAlimento = "";  
  
    for (int i = 0; i < n ; i++) {  
  
        ProveedorAlimento proveedorAlimento = new ProveedorAlimento();  
  
        proveedorAlimento.setId_proveedor_alimento(nit_proveedores.get(i));  
        proveedorAlimento.setId_alimento_proveedor(id_alimentos.get(i));  
  
        insertProveedorAlimento = "INSERT INTO proveedor_alimento (id_proveedor_alimento, id_alimento_proveedor) VALUES('%s',  
        insertProveedorAlimento = String.format(insertProveedorAlimento,  
            proveedorAlimento.getId_proveedor_alimento(),  
            proveedorAlimento.getId_alimento_proveedor());  
  
        mysqlOperation.setSqlStatement(insertProveedorAlimento);  
        mysqlOperation.executeSqlStatementVoid();  
  
    }  
}
```

Utiliza los ids generados en metodos anteriores para usarlos como FK

Activar Windows

```
1 • USE zoodb;  
2  
3 • SELECT * FROM Proveedor;  
4 • SELECT * FROM factura;  
5 • SELECT * FROM logistico;  
6 • SELECT * FROM entrenador;  
7 • SELECT * FROM orden_compra;  
8 • SELECT * FROM alimento;  
9 • SELECT * FROM proveedor_alimento;  
10 • SELECT * FROM dieta;  
11 • SELECT * FROM animal;  
12 • SELECT * FROM alimento_animal;  
13 • SELECT * FROM informe_animal;  
14 • SELECT * FROM veterinario;  
15 • SELECT * FROM veterinario_informe;  
16 • SELECT * FROM veterinario_dieta;
```

Trae la tabla proveedor_alimento

id_proveedor_alimento	id_alimento_proveedor
NIT-0	ALI-0
NIT-1	ALI-1
NIT-10	ALI-10
NIT-11	ALI-11
NIT-12	ALI-12
NIT-13	ALI-13
NIT-14	ALI-14
NIT-15	ALI-15

Tabla proveedor_alimento

Retorna 50 filas

#	Time	Action	Message
1	12:13:57	SELECT * FROM proveedor_alimento LIMIT 0, 1000	50 row(s) returned

Método para poblar la tabla dieta.

ZOO (Ejercicio B)

```
private static List<String> generarDieta() {  
    String insertDieta = "";  
    for (int i=0; i < n; i++) {  
        Dieta dieta = new Dieta();  
        Faker faker = new Faker();  
  
        dieta.setId_dieta("DIE-" + i);  
        dieta.setTipo_alimento(faker.lorem().word());  
        dieta.setDosis(faker.number().numberBetween(1000, 10000)+ " gr");  
  
        insertDieta = "INSERT INTO dieta (id_dieta, tipo_alimento, dosis) VALUES ('%s', '%s', '%s');";  
        insertDieta = String.format(insertDieta,  
            dieta.getId_dieta(),  
            dieta.getTipo_alimento(),  
            dieta.getDosis());  
  
        mysqlOperation.setSqlStatement(insertDieta);  
        mysqlOperation.executeSqlStatementVoid();  
  
        id_dietas.add(dieta.getId_dieta());  
    }  
    return id_dietas;  
}
```

Devuelta una lista

Retorna una lista con los ids de dietas para usarlos en otras tabla

```
1 • USE zoodb;  
2  
3 • SELECT * FROM Proveedor;  
4 • SELECT * FROM factura;  
5 • SELECT * FROM logistico;  
6 • SELECT * FROM entrenador;  
7 • SELECT * FROM orden_compra;  
8 • SELECT * FROM alimento;  
9 • SELECT * FROM proveedor_alimento;  
10 • SELECT * FROM dieta;  
11 • SELECT * FROM animal;  
12 • SELECT * FROM alimento_animal;  
13 • SELECT * FROM informe_animal;  
14 • SELECT * FROM veterinario;  
15 • SELECT * FROM veterinario_informe;  
16 • SELECT * FROM veterinario_dieta;
```

Result Grid

id_dieta	tipo_alimento	dosis
DIE-0	aut	1319 gr
DIE-1	consequuntur	1962 gr
DIE-10	et	7387 gr
DIE-11	velit	9753 gr
DIE-12	aut	7293 gr
DIE-13	nihil	5261 gr
DIE-14	pariatur	2116 gr
DIE-15	insum	5754 gr

Tabla dieta

Retorna 50 filas

Output

#	Time	Action	Message
1	12:15:45	SELECT * FROM dieta LIMIT 0, 1000	50 row(s) returned

Método para poblar la tabla animal.

ZOO (Ejercicio B)

```
private static List<String> generarAnimal(List<String> id_dietas) {  
    String insertAnimal = "";  
    for (int i = 0; i < n; i++) {  
        Animal animal = new Animal();  
        Faker faker = new Faker();  
  
        animal.setId_animal("ANI-" + i);  
        animal.setNombre(faker.animal().name());  
        animal.setClase(faker.animal().name());  
        animal.setEspecie(faker.animal().name());  
        animal.setFecha_nacimiento(String.valueOf(faker.date().between(Date.from(LocalDate.of(year: 1990, month: 1, dayOfMonth: 1)), Date.from(LocalDate.of(year: 2020, month: 1, dayOfMonth: 1))));  
        animal.setId_dieta_animal(id_dietas.get(i));  
  
        insertAnimal = "INSERT INTO animal (id_animal, nombre, clase, especie, fecha_nacimiento, id_dieta_animal) VALUES ('%s', '%s', '%s', '%s', '%s', '%s')";  
        insertAnimal = String.format(insertAnimal,  
            animal.getId_animal(),  
            animal.getNombre(),  
            animal.getClase(),  
            animal.getEspecie(),  
            animal.getFecha_nacimiento(),  
            animal.getId_dieta_animal());  
  
        mysqlOperation.setSqlStatement(insertAnimal);  
        mysqlOperation.executeSqlStatementVoid();  
        id_animales.add(animal.getId_animal());  
    }  
    return id_animales;  
}
```

Devuelve una lista

Retorna una lista con los ids de los animales para usarlos en otras tablas:

```
1 • USE zoodb;  
2  
3 • SELECT * FROM Proveedor;  
4 • SELECT * FROM factura;  
5 • SELECT * FROM logistico;  
6 • SELECT * FROM entrenador;  
7 • SELECT * FROM orden_compra;  
8 • SELECT * FROM alimento;  
9 • SELECT * FROM proveedor_alimento;  
10 • SELECT * FROM dieta;  
11 • SELECT * FROM animal;  
12 • SELECT * FROM alimento_animal;  
13 • SELECT * FROM informe_animal;  
14 • SELECT * FROM veterinario;  
15 • SELECT * FROM veterinario_informe;  
16 • SELECT * FROM veterinario_dieta;
```

Result Grid

id_animal	nombre	clase	especie	fecha_nacimiento	id_dieta_animal
ANI-0	fish	worm	caribou	Tue Apr 22 09:00:02 COT 2003	DIE-0
ANI-1	snail	spider	mammoth	Tue Nov 05 08:03:31 COT 2019	DIE-1
ANI-10	wombat	alpaca	elk	Fri Jul 13 18:05:52 COT 2007	DIE-10
ANI-11	guinea pig	bat	cheetah	Thu Sep 28 04:00:33 COT 1995	DIE-11
ANI-12	mallard	rhinoceros	cod	Fri May 07 09:17:57 COT 2004	DIE-12
ANI-13	goat	woodchuck	manatee	Thu May 30 21:34:04 COT 2019	DIE-13
ANI-14	chinchilla	butterfly	dog	Wed Oct 30 20:05:52 COT 1996	DIE-14
ANI-15	normoise	hamster	sardine	Sun Oct 12 23:15:36 COT 2014	DIF-15

animal 11 x

Output

Action Output

#	Time	Action	Message
1	12:17:19	SELECT * FROM animal LIMIT 0, 1000	50 row(s) returned

Método para poblar tabla alimento_animal.

ZOO (Ejercicio B)

```
private static void generarAlimentoAnimal(List<String> id_alimentos, List<String> id_animales) {  
    String insertAlimentoAnimal = "";  
    for (int i = 0; i < n ; i++) {  
        AlimentoAnimal alimentoAnimal = new AlimentoAnimal();  
        alimentoAnimal.setId_alimento_animal(id_alimentos.get(i));  
        alimentoAnimal.setId_animal_alimento(id_animales.get(i));  
        insertAlimentoAnimal = "INSERT INTO alimento_animal (id_alimento_animal, id_animal_alimento) VALUES('%s', '%s')";  
        insertAlimentoAnimal = String.format(insertAlimentoAnimal,  
            alimentoAnimal.getId_alimento_animal(),  
            alimentoAnimal.getId_animal_alimento());  
        mysqlOperation.setSqlStatement(insertAlimentoAnimal);  
        mysqlOperation.executeSqlStatementVoid();  
    }  
}
```

Utiliza los id de alimentos y animales para poblar la tabla

```
1 • USE zoodb;  
2  
3 • SELECT * FROM Proveedor;  
4 • SELECT * FROM factura;  
5 • SELECT * FROM logistico;  
6 • SELECT * FROM entrenador;  
7 • SELECT * FROM orden_compra;  
8 • SELECT * FROM alimento;  
9 • SELECT * FROM proveedor_alimento;  
10 • SELECT * FROM dieta;  
11 • SELECT * FROM animal;  
12 • SELECT * FROM alimento_animal;  
13 • SELECT * FROM informe_animal;  
14 • SELECT * FROM veterinario;  
15 • SELECT * FROM veterinario_informe;  
16 • SELECT * FROM veterinario_dieta;
```

Trae tabla alimento_animal

id_alimento_animal	id_animal_alimento
ALI-0	ANI-0
ALI-1	ANI-1
ALI-10	ANI-10
ALI-11	ANI-11
ALI-12	ANI-12
ALI-13	ANI-13
ALI-14	ANI-14
ALI-15	ANI-15

Tabla alimento_animal

Retorna 50 filas

#	Time	Action	Message
1	12:18:43	SELECT * FROM alimento_animal LIMIT 0, 1000	50 row(s) returned

Método para poblar tabla informe_animal

ZOO (Ejercicio B)

```
private static List<String> generarInformeAnimal(List<String> id_animales, List<String> id_entrenadores) {  
    String insertInformeAnimal = "";  
    for (int i = 0; i < n ; i++) {  
        InformeAnimal informeAnimal = new InformeAnimal();  
        Faker faker = new Faker();  
  
        informeAnimal.setId_informe("INF-" + i);  
        informeAnimal.setFecha_peso(String.valueOf(faker.date().between(Date.from(LocalDate.of( year: 2022, month: 1, dayOf  
        informeAnimal.setPeso_registrado(faker.number().numberBetween(1000, 10000)+ " kg");  
        informeAnimal.setDetalles(faker.lorem().characters( fixedNumberOfCharacters: 45));  
        informeAnimal.setId_animal_informe(id_animales.get(i));  
        informeAnimal.setId_entrenador_informe(id_entrenadores.get(i));  
  
        insertInformeAnimal = "INSERT INTO informe_animal (id_informe, fecha_peso, peso_registrado, detalles, id_animal_informe, id_entrenador_informe) VALUES (" +  
        insertInformeAnimal = String.format(insertInformeAnimal,  
            informeAnimal.getId_informe(),  
            informeAnimal.getFecha_peso(),  
            informeAnimal.getPeso_registrado(),  
            informeAnimal.getDetalles(),  
            informeAnimal.getId_animal_informe(),  
            informeAnimal.getId_entrenador_informe());  
  
        mySqlOperation.setSqlStatement(insertInformeAnimal);  
        mySqlOperation.executeSqlStatementVoid();  
        id_informes.add(informeAnimal.getId_informe());  
    }  
    return id_informes;  
}
```

Utiliza los ids de animales y entrenadores para poblar la tabla

```
1 • USE zoodb;  
2  
3 • SELECT * FROM Proveedor;  
4 • SELECT * FROM factura;  
5 • SELECT * FROM logistico;  
6 • SELECT * FROM entrenador;  
7 • SELECT * FROM orden_compra;  
8 • SELECT * FROM alimento;  
9 • SELECT * FROM proveedor_alimento;  
10 • SELECT * FROM dieta;  
11 • SELECT * FROM animal;  
12 • SELECT * FROM alimento_animal;  
13 • SELECT * FROM informe_animal;  
14 • SELECT * FROM veterinario;  
15 • SELECT * FROM veterinario_informe;  
16 • SELECT * FROM veterinario_dieta;
```

Trae tabla informe_animal

Tabla informe_animal

id_informe	fecha_peso	peso_registrado	detalles	id_animal_informe	id_entrenador_informe
INF-0	Sat Jan 01 17:12:39 COT 2022	1689 kg	5jfbxy6ozw7wxt7vpxwco0zd7rvvub9tic0vlnojh...	ANI-0	ENT-0
INF-1	Mon Feb 13 11:21:00 COT 2023	4043 kg	8p0a5pvhmkaakioxyzz1lgmbkt4fydzg9ty6at6b...	ANI-1	ENT-1
INF-10	Tue Dec 13 07:42:24 COT 2022	2847 kg	rsdzesiw03r040dt4dei6hfu8qg70dbn2en5ywwm...	ANI-10	ENT-10
INF-11	Fri Dec 23 05:03:11 COT 2022	8578 kg	23elfnv9kj5ni2tmhxppus5lat65tc4rmitc4ccb6d4w	ANI-11	ENT-11
INF-12	Thu May 05 05:45:42 COT 2022	8451 kg	vlyd9kirtgxd5szx8yyd67srjba7e5wivsmgugd04...	ANI-12	ENT-12
INF-13	Sat Dec 31 14:21:59 COT 2022	4001 kg	q7h0d3e2inosj3p1c7wcp2kag5kb03mhsagrsvg4...	ANI-13	ENT-13
INF-14	Wed Nov 30 06:59:51 COT 2022	5211 kg	gv64gm37erj8s94e6rxdzcdraqw9ud4fytu4h4nr...	ANI-14	ENT-14
INF-15	Wed Nov 02 21:05:05 COT 2022	6342 kg	d7h7z9c54rde9ctu8mdldzln1mra3ddut31kw99...	ANI-15	ENT-15

Retorna 50 filas

```
1 12:20:13 SELECT * FROM informe_animal LIMIT 0, 1000 50 row(s) returned
```

Método para poblar tabla veterinario.

ZOO (Ejercicio B)

```
private static List<String> generarVeterinarios() {  
    String insertVeterinario = "";  
  
    for (int i = 0; i < n ; i++) {  
  
        Veterinario veterinario = new Veterinario();  
        Faker faker = new Faker();  
  
        veterinario.setId_veterinario("VET-" + i);  
        veterinario.setNombre(faker.name().firstName());  
        veterinario.setApellido(faker.name().lastName());  
        veterinario.setTelefono("555" + faker.numerify( numberString: "##"));  
  
        insertVeterinario = "INSERT INTO veterinario (id_veterinario, nombre, apellido, telefono) VALUES('%s', '%s', '%s',  
        insertVeterinario = String.format(insertVeterinario,  
            veterinario.getId_veterinario(),  
            veterinario.getNombre(),  
            veterinario.getApellido(),  
            veterinario.getTelefono());  
  
        mysqlOperation.setSqlStatement(insertVeterinario);  
        mysqlOperation.executeSqlStatementVoid();  
  
        id_veterinarios.add(veterinario.getId_veterinario());  
    }  
    return id_veterinarios;  
}
```

Retorna una lista

Retorna lista de id de veterinarios para usarlos en otras tabla como FK

```
1 • USE zoodb;  
2  
3 • SELECT * FROM Proveedor;  
4 • SELECT * FROM factura;  
5 • SELECT * FROM logistico;  
6 • SELECT * FROM entrenador;  
7 • SELECT * FROM orden_compra;  
8 • SELECT * FROM alimento;  
9 • SELECT * FROM proveedor_alimento;  
10 • SELECT * FROM dieta;  
11 • SELECT * FROM animal;  
12 • SELECT * FROM alimento_animal;  
13 • SELECT * FROM informe_animal;  
14 • SELECT * FROM veterinario;  
15 • SELECT * FROM veterinario_informe;  
16 • SELECT * FROM veterinario_dieta;
```

Trae tabla veterinario

id_veterinario	nombre	apellido	telefono
VET-0	Michiko	Wolff	55545
VET-1	Margarette	Pollich	55567
VET-10	Sydney	McGlynn	55549
VET-11	Mercedez	Schiller	55573
VET-12	Desmond	Koelpin	55514
VET-13	Demetrius	Lowe	55543
VET-14	Marylynn	Nicolas	55565
VET-15	Sammy	Grady	55575

Tabla veterinario

Retorna 50 filas

1 12:22:08 SELECT * FROM veterinario LIMIT 0, 1000 50 row(s) returned

Método para poblar tabla veterinario_informe

ZOO (Ejercicio B)

```
private static void generarVeterinarioInforme(List<String> id_veterinarios, List<String> id_informes) {  
  
    String insertVeterinarioInforme = "";  
  
    for (int i = 0; i < n ; i++) {  
  
        VeterinarioInforme veterinarioInforme = new VeterinarioInforme();  
  
        veterinarioInforme.setId_veterinario_informe(id_veterinarios.get(i));  
        veterinarioInforme.setId_informe_veterinario(id_informes.get(i));  
  
        insertVeterinarioInforme = "INSERT INTO veterinario_informe (id_veterinario_informe, id_informe_veterinario) VALUE  
        insertVeterinarioInforme = String.format(insertVeterinarioInforme,  
            veterinarioInforme.getId_veterinario_informe(),  
            veterinarioInforme.getId_informe_veterinario());  
  
        mySqlOperation.setSqlStatement(insertVeterinarioInforme);  
        mySqlOperation.executeSqlStatementVoid();  
  
    }  
}
```

Utiliza la lista de id de Veterinarios y de Informe para poblar la tabla

Activar Windows

```
1 • USE zoodb;  
2  
3 • SELECT * FROM Proveedor;  
4 • SELECT * FROM factura;  
5 • SELECT * FROM logistico;  
6 • SELECT * FROM entrenador;  
7 • SELECT * FROM orden_compra;  
8 • SELECT * FROM alimento;  
9 • SELECT * FROM proveedor_alimento;  
10 • SELECT * FROM dieta;  
11 • SELECT * FROM animal;  
12 • SELECT * FROM alimento_animal;  
13 • SELECT * FROM informe_animal;  
14 • SELECT * FROM veterinario;  
15 • SELECT * FROM veterinario_informe;  
16 • SELECT * FROM veterinario_dieta;
```

Trae tabla veterinario_informe

id_veterinario_informe	id_informe_veterinario
VET-0	INF-0
VET-1	INF-1
VET-10	INF-10
VET-11	INF-11
VET-12	INF-12
VET-13	INF-13
VET-14	INF-14
VET-15	INF-15

Tabla veterinario_informe

Retorna 50 filas

Output

Action Output

#	Time	Action	Message
1	12:23:44	SELECT * FROM veterinario_informe LIMIT 0, 1000	50 row(s) returned

Método para poblar tabla veterinario_dieta

ZOO (Ejercicio B)

```
private static void generarVeterinarioDieta(List<String> id_veterinarios, List<String> id_dietas) {  
  
    String insertVeterinarioDieta = "";  
  
    for (int i = 0; i < n ; i++) {  
  
        VeterinarioDieta veterinarioDieta = new VeterinarioDieta();  
  
        veterinarioDieta.setId_veterinario_dieta(id_veterinarios.get(i));  
        veterinarioDieta.setId_dieta_veterinario(id_dietas.get(i));  
  
        insertVeterinarioDieta = "INSERT INTO veterinario_dieta (id_veterinario_dieta, id_dieta_veterinario) VALUES('%s',  
        insertVeterinarioDieta = String.format(insertVeterinarioDieta,  
            veterinarioDieta.getId_veterinario_dieta(),  
            veterinarioDieta.getId_dieta_veterinario());  
  
        mySqlOperation.setSqlStatement(insertVeterinarioDieta);  
        mySqlOperation.executeSqlStatementVoid();  
  
    }  
  
}
```

Utiliza la lista de ids de veterinario y dieta para poblar la tabla

```
1 • USE zoodb;  
2  
3 • SELECT * FROM Proveedor;  
4 • SELECT * FROM factura;  
5 • SELECT * FROM logistico;  
6 • SELECT * FROM entrenador;  
7 • SELECT * FROM orden_compra;  
8 • SELECT * FROM alimento;  
9 • SELECT * FROM proveedor_alimento;  
10 • SELECT * FROM dieta;  
11 • SELECT * FROM animal;  
12 • SELECT * FROM alimento_animal;  
13 • SELECT * FROM informe_animal;  
14 • SELECT * FROM veterinario;  
15 • SELECT * FROM veterinario_informe;  
16 • SELECT * FROM veterinario_dieta;
```

Trae tabla veterinario_dieta

id_veterinario_dieta	id_dieta_veterinario
VET-0	DIE-0
VET-1	DIE-1
VET-10	DIE-10
VET-11	DIE-11
VET-12	DIE-12
VET-13	DIE-13
VET-14	DIE-14
VET-15	DIE-15

Tabla veterinario_dieta

Retorna 50 filas

#	Time	Action	Messages
1	12:25:37	SELECT * FROM veterinario_dieta LIMIT 0, 1000	50 row(s) returned

Respuesta última pregunta

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?

La respuesta a esta pregunta la realizo según la complejidad y el tamaño de la aplicación, la naturaleza de los datos y los requisitos de consulta. En general, para aplicaciones que manejan relaciones complejas entre diferentes tipos de datos, como en este caso, es probable que una base de datos relacional sea una buena opción.

ZOO (Ejercicio B)

La base de datos creada incluye varias tablas relacionales que se conectan mediante claves primarias y foráneas, lo que permite la realización de consultas y análisis complejos en los datos. Además, la base de datos relacional es altamente escalable y puede manejar grandes conjuntos de datos, lo que la hace adecuada para aplicaciones de alta demanda.

Por otro lado, una base de datos no relacional podría ser una opción adecuada si la aplicación no requiere relaciones complejas entre los datos y si los datos se presentan en forma de documentos o datos no estructurados.