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 (mamiferos, aves, anfibios, peces y reptíles) 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 logisticos, empleados veterinarios, empleados entrenadores, visitantes.
- El veterinario esta 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 aclaraciónes o especificaciones necesarias para realizar el ejercicio.

ZOO (Ejercicio B)

Tabla de Contenido

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.

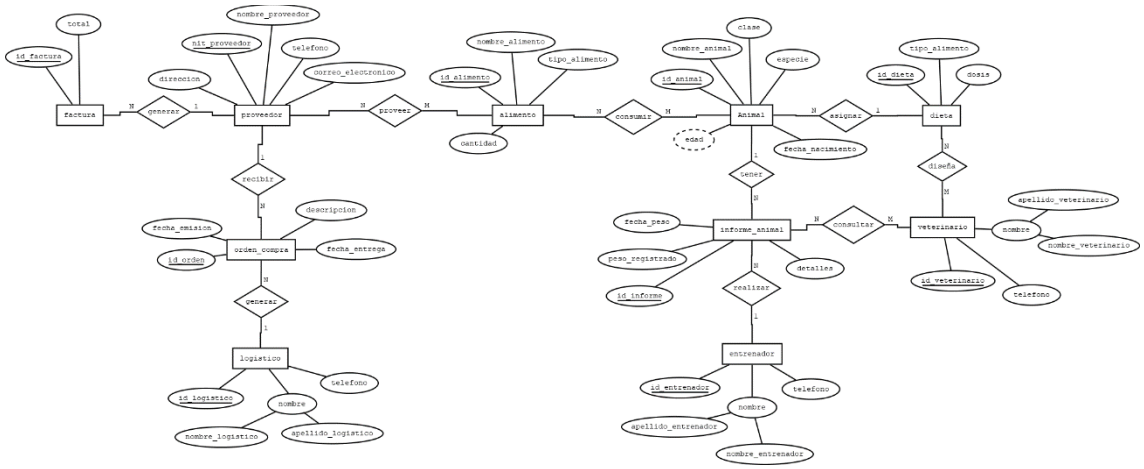


Diagrama E-R

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.

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

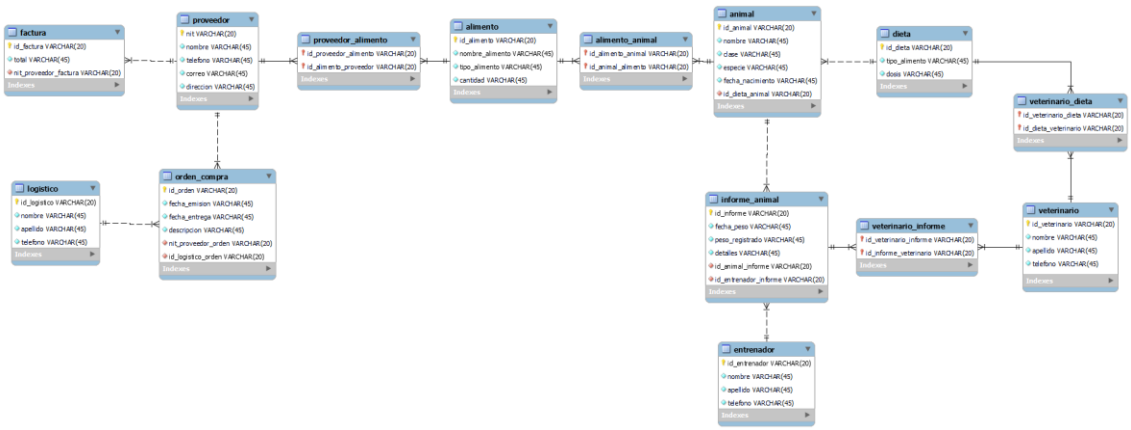


Diagrama Modelo Relacional

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.

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

```sql
CREATE DATABASE  zoodb;

USE zoodb;

CREATE TABLE proveedor(
    nit VARCHAR(20) NOT NULL,
    nombre VARCHAR(45) NOT NULL,
    telefono VARCHAR(45) NOT NULL,
    correo VARCHAR(45) NOT NULL,
    direccion VARCHAR(45) NOT NULL,
    PRIMARY KEY (nit)
);

CREATE TABLE factura(
    id_factura VARCHAR(20) NOT NULL,
    total VARCHAR(45) NOT NULL,
    nit_proveedor_factura VARCHAR(20) NOT NULL,
    PRIMARY KEY (id_factura),
    FOREIGN KEY (nit_proveedor_factura) REFERENCES proveedor(nit)
);

CREATE TABLE logistico(
    id_logistico VARCHAR(20) NOT NULL,
    nombre VARCHAR(45) NOT NULL,
    apellido VARCHAR(45) NOT NULL,
    telefono VARCHAR(45) NOT NULL,
    PRIMARY KEY (id_logistico)
);
```

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  );
```

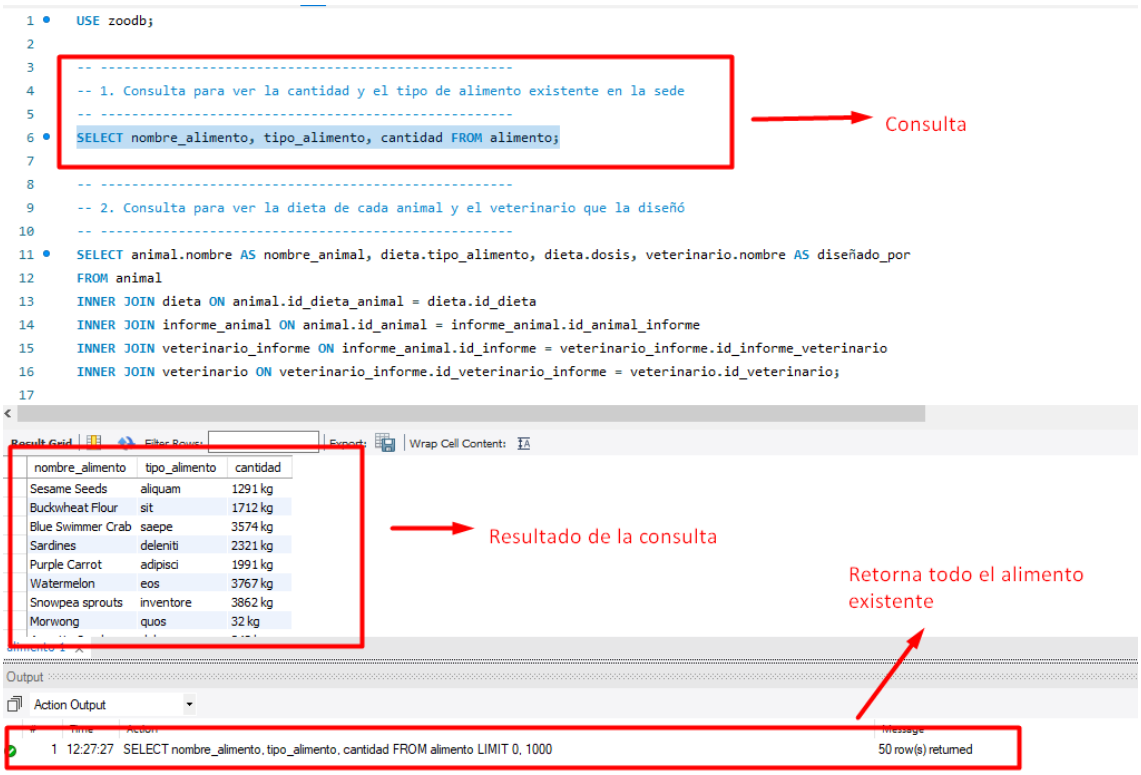
ZOO (Ejercicio B)

```
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
```

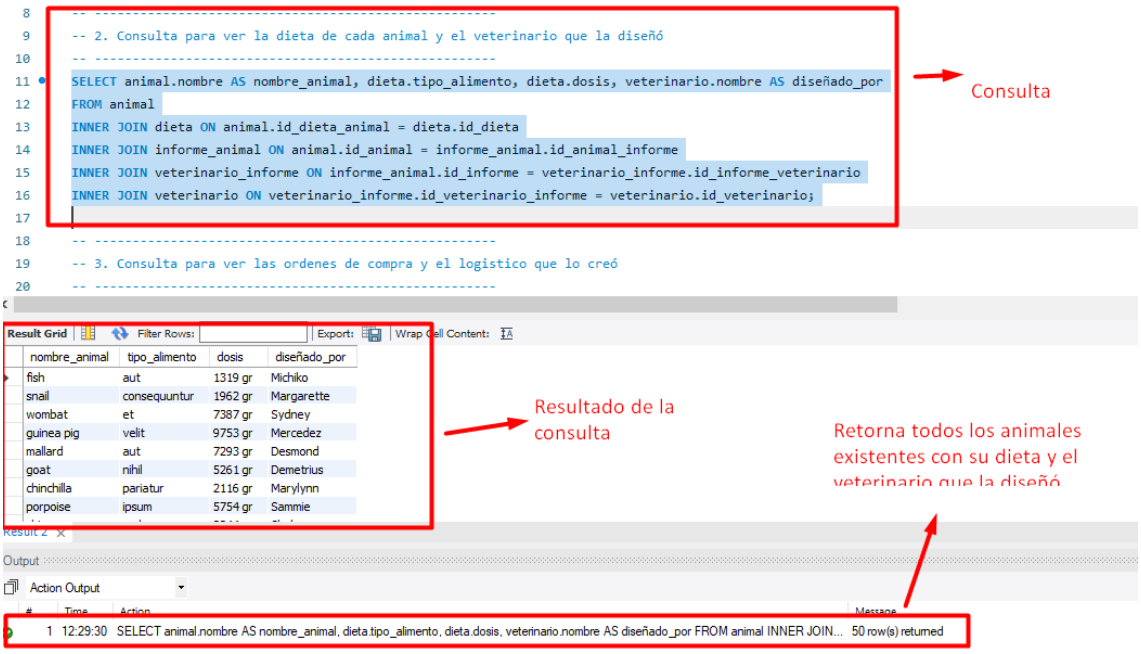
Consultas

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

ZOO (Ejercicio B)

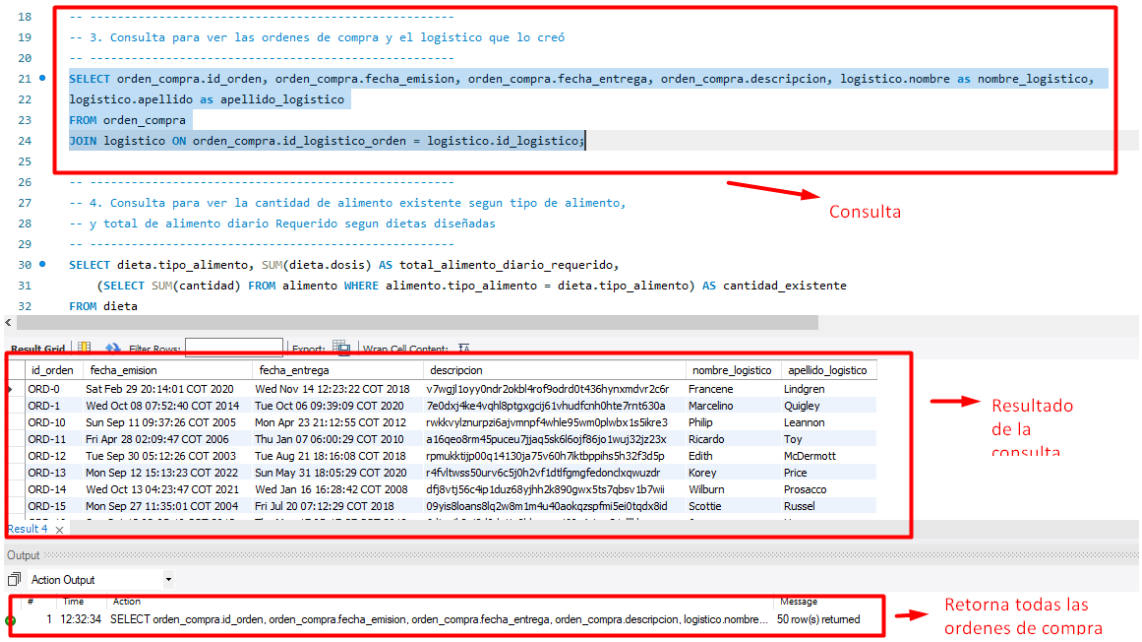


2. Consulta para ver la dieta de cada animal y el veterinario que la diseñó.



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

ZOO (Ejercicio B)



```
-- ------------------------------------------------------
-- 3. Consulta para ver las ordenes de compra y el logistico que lo creó
-- ------------------------------------------------------
SELECT orden_compra.id_orden, orden_compra.fecha_emision, orden_compra.fecha_entrega, orden_compra.descripcion, logistico.nombre as nombre_logistico,
logistico.apellido as apellido_logistico
FROM orden_compra
JOIN logistico ON orden_compra.id_logistico_orden = logistico.id_logistico;

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

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	v7wgjl1oyy0ndr2okbl4rof9odrd0t436hynxmdvr2c6r	Francene	Lindgren
ORD-1	Wed Oct 08 07:52:40 COT 2014	Tue Oct 06 09:39:09 COT 2020	7e0dxj4ke4vqhl8ptgxgcjj61vhudfcnh0hte7rnt630a	Marcelino	Quigley
ORD-10	Sun Sep 11 09:37:26 COT 2005	Mon Apr 23 21:12:55 COT 2012	rwkkvylznurpzi6ajvmnpf4whle95wm0plwbx1s5ikre3	Philip	Leannon
ORD-11	Fri Apr 28 02:09:47 COT 2006	Thu Jan 07 06:00:29 COT 2010	a16qeo8rm45puceu7jjaq5sk6l6ojf86jo1wuj32jz23x	Ricardo	Toy
ORD-12	Tue Sep 30 05:12:26 COT 2003	Tue Aug 21 18:16:08 COT 2018	rpmukktjp00q14130ja75v60h7iktbppihs5h32f3d5p	Edith	McDermott
ORD-13	Mon Sep 12 15:13:23 COT 2022	Sun May 31 18:05:29 COT 2020	r4fvltwss50urv6c5j0h2vf1dtlfgmgfedonclxqwuzdr	Korey	Price
ORD-14	Wed Oct 13 04:23:47 COT 2021	Wed Jan 16 16:28:42 COT 2008	dfj8vtj56c4ip1duz68yjhh2k890gwx5ts7qbsv1b7wii	Wilburn	Prosacco
ORD-15	Mon Sep 27 11:35:01 COT 2004	Fri Jul 20 07:12:29 COT 2018	09yis8loans8lq2w8m1m4u40aokqzspfmi5ei0tqdx8id	Scottie	Russel

Resultado de la consulta

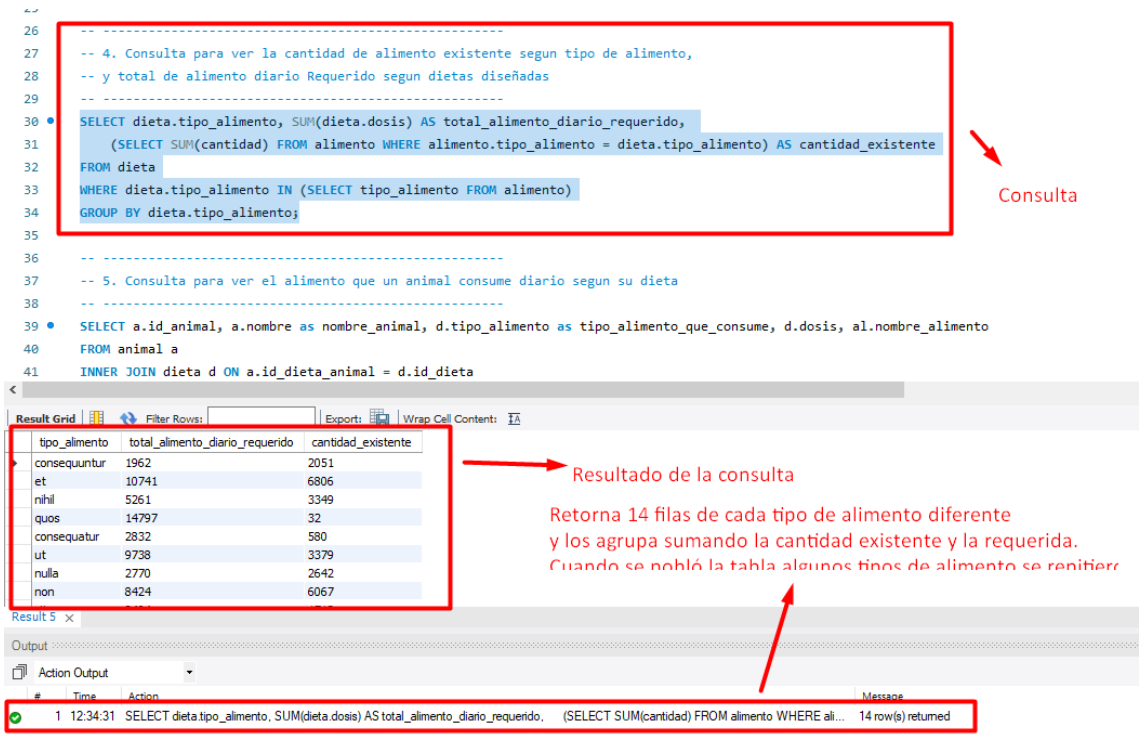
Output

Action Output

#	Time	Action	Message
1	12:32:34	SELECT orden_compra.id_orden, orden_compra.fecha_emision, orden_compra.fecha_entrega, orden_compra.descripcion, logistico.nombre...	50 row(s) returned

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



```
-- ------------------------------------------------------
-- 4. Consulta para ver la cantidad de alimento existente segun tipo de alimento,
-- y total de alimento diario Requerido segun dietas diseñadas
-- ------------------------------------------------------
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;

-- ------------------------------------------------------
-- 5. Consulta para ver el alimento que un animal consume diario segun su dieta
-- ------------------------------------------------------
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
```

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 repitierc

Output

Action Output

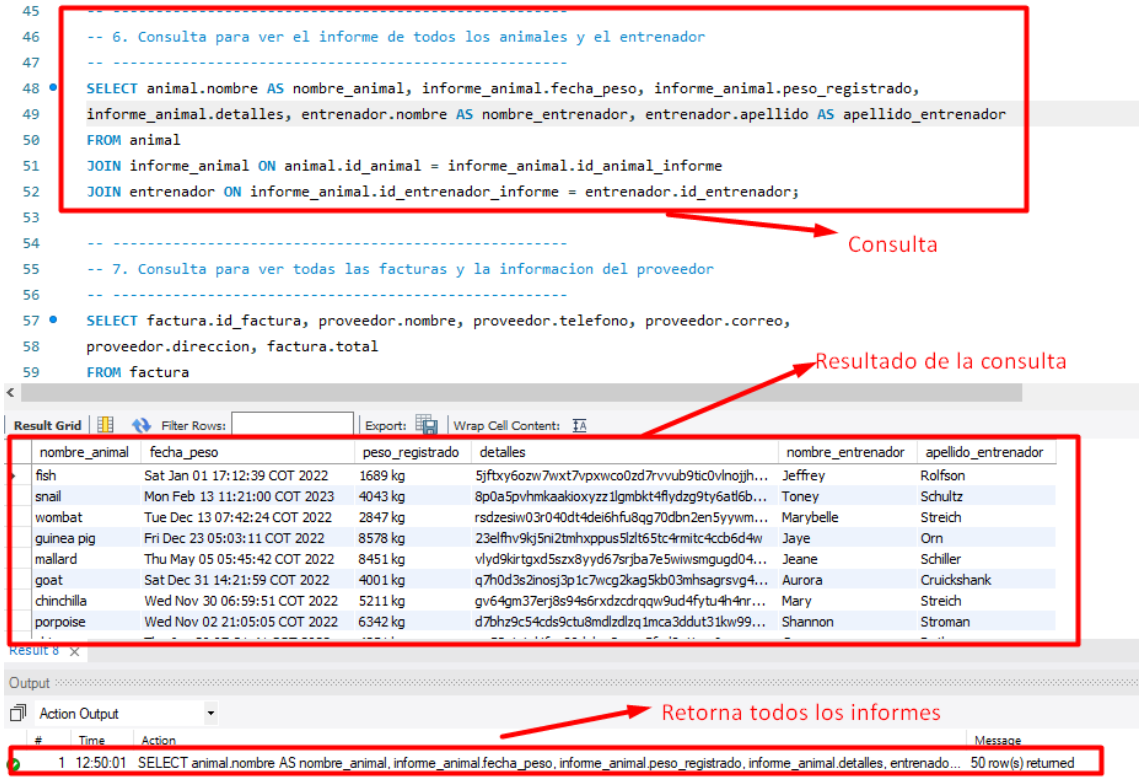
#	Time	Action	Message
1	12:34:31	SELECT dieta.tipo_alimento, SUM(dieta.dosis) AS total_alimento_diario_requerido, (SELECT SUM(cantidad) FROM alimento WHERE ali...	14 row(s) returned

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

ZOO (Ejercicio B)

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

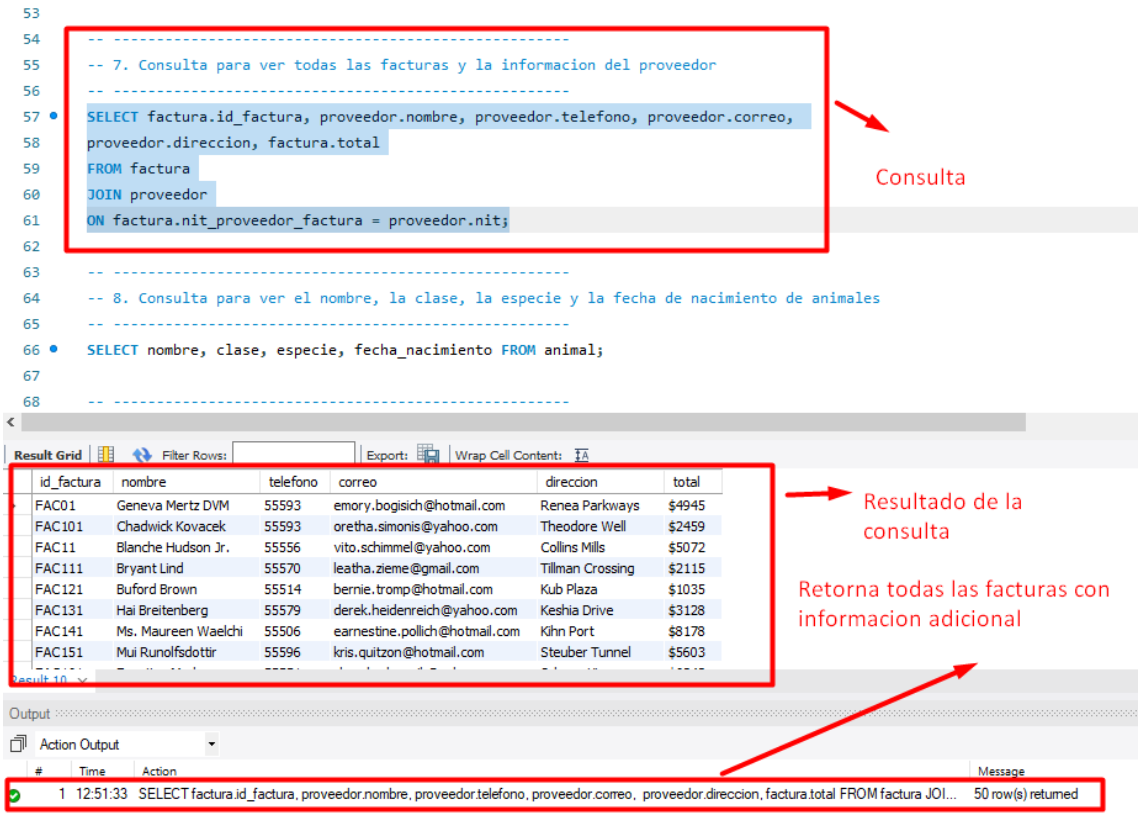
6. Consulta para ver el informe de todos los animales y el entrenador



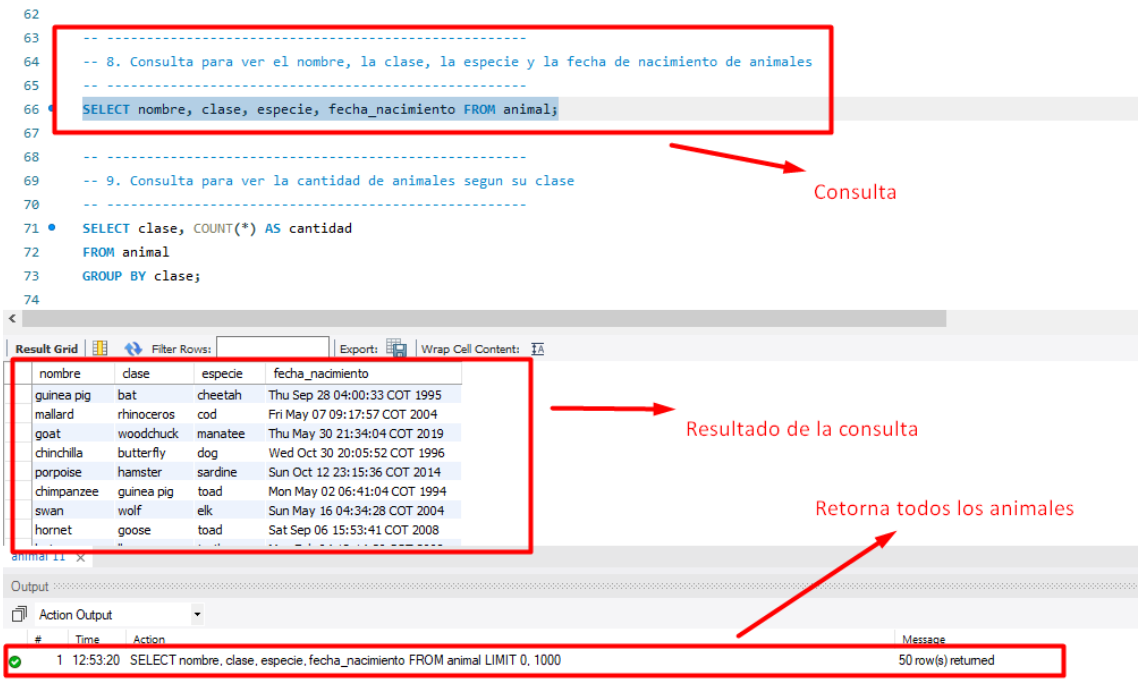
nombre_animal	fecha_peso	peso_registrado	detalles	nombre_entrenador	apellido_entrenador
fish	Sat Jan 01 17:12:39 COT 2022	1689 kg	5jftxy6ozw7wxt7vpxwco0zd7rvvub9tic0vlnojjh...	Jeffrey	Rolfson
snail	Mon Feb 13 11:21:00 COT 2023	4043 kg	8p0a5pvhmkaakioxyzz1lgmbkt4flydzg9ty6atl6b...	Toney	Schultz
wombat	Tue Dec 13 07:42:24 COT 2022	2847 kg	rsdzesiw03r040dt4dei6hfu8qg70dbn2en5yywm...	Marybelle	Streich
guinea pig	Fri Dec 23 05:03:11 COT 2022	8578 kg	23elfhv9kj5ni2tmhxppus5lzlt65tc4rmitc4ccb6d4w	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	q7h0d3s2inosj3p1c7wcg2kag5kb03mhsagrsvg4...	Aurora	Cruickshank
chinchilla	Wed Nov 30 06:59:51 COT 2022	5211 kg	gv64gm37erj8s94s6rxdzcdrqqw9ud4fytu4h4nr...	Mary	Streich
porpoise	Wed Nov 02 21:05:05 COT 2022	6342 kg	d7bhz9c54cds9ctu8mdlzdlzq1mca3ddut31kw99...	Shannon	Stroman

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

ZOO (Ejercicio B)

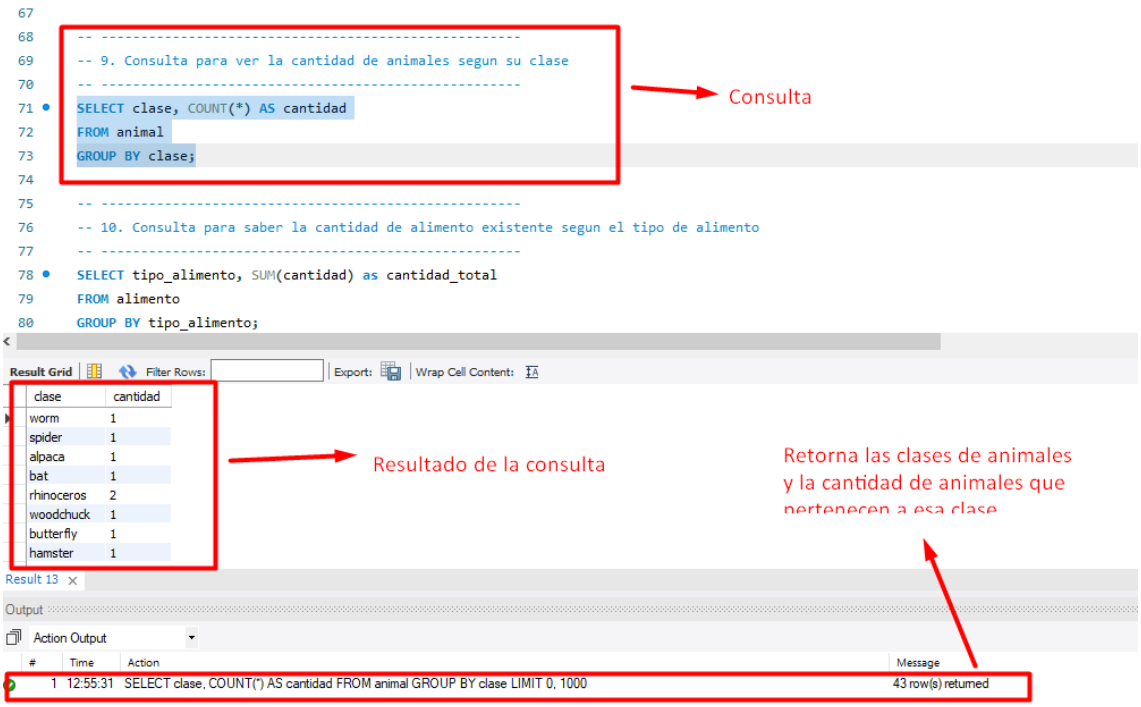


8. Consulta para ver el nombre, la clase, la especie y la fecha de nacimiento de animales

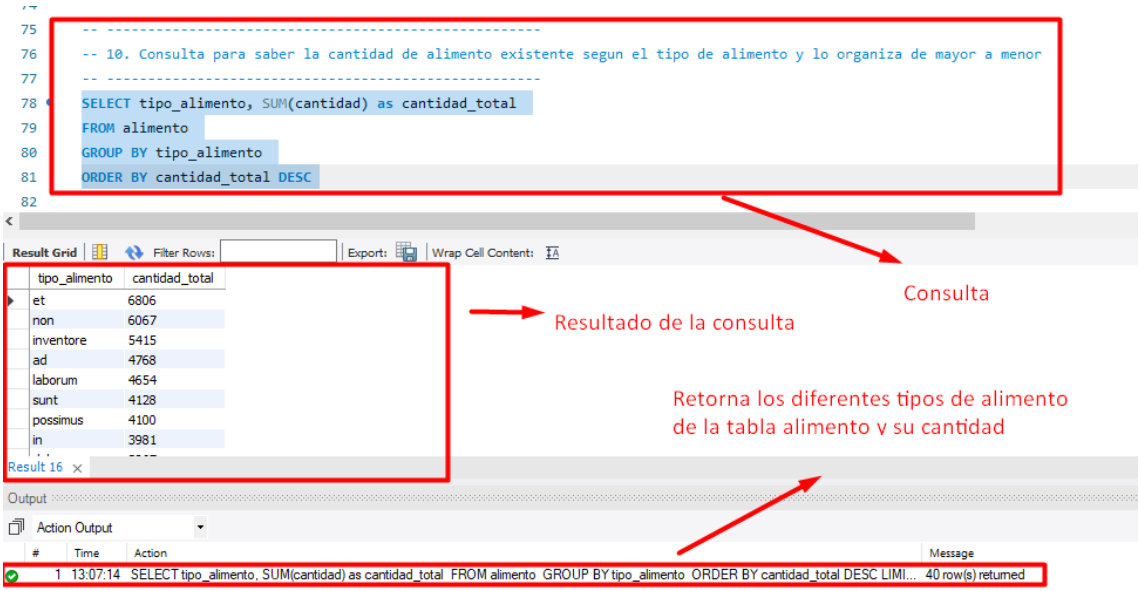


9. Consulta para ver la cantidad de animales según su clase

ZOO (Ejercicio B)



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

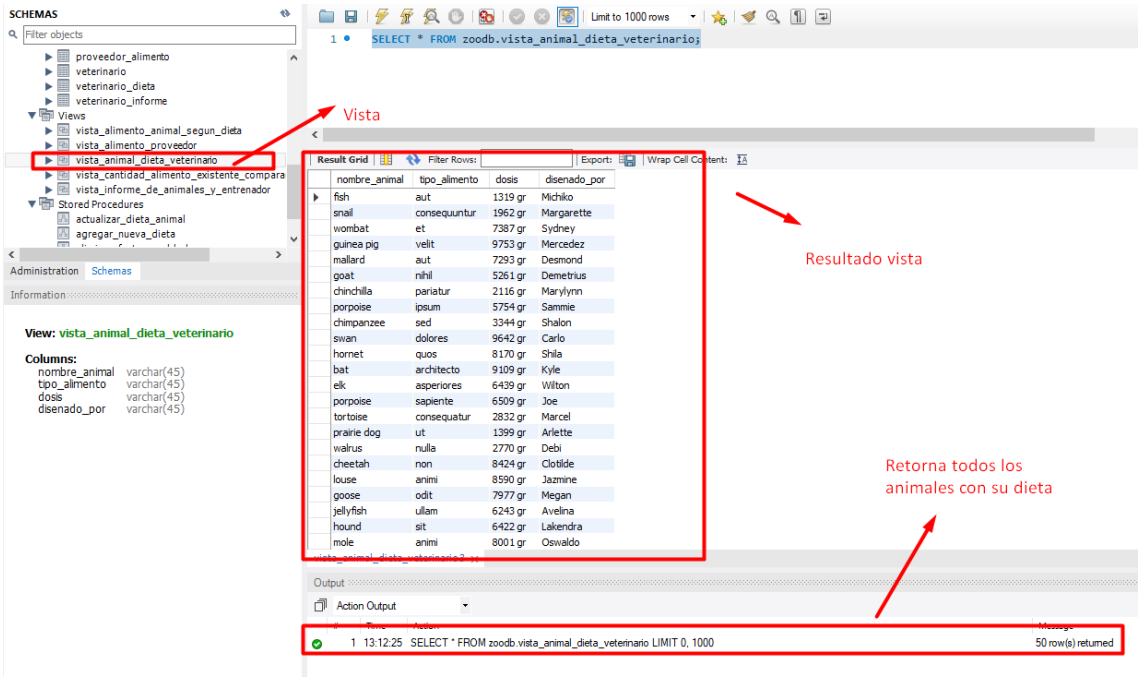


ZOO (Ejercicio B)

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;
```



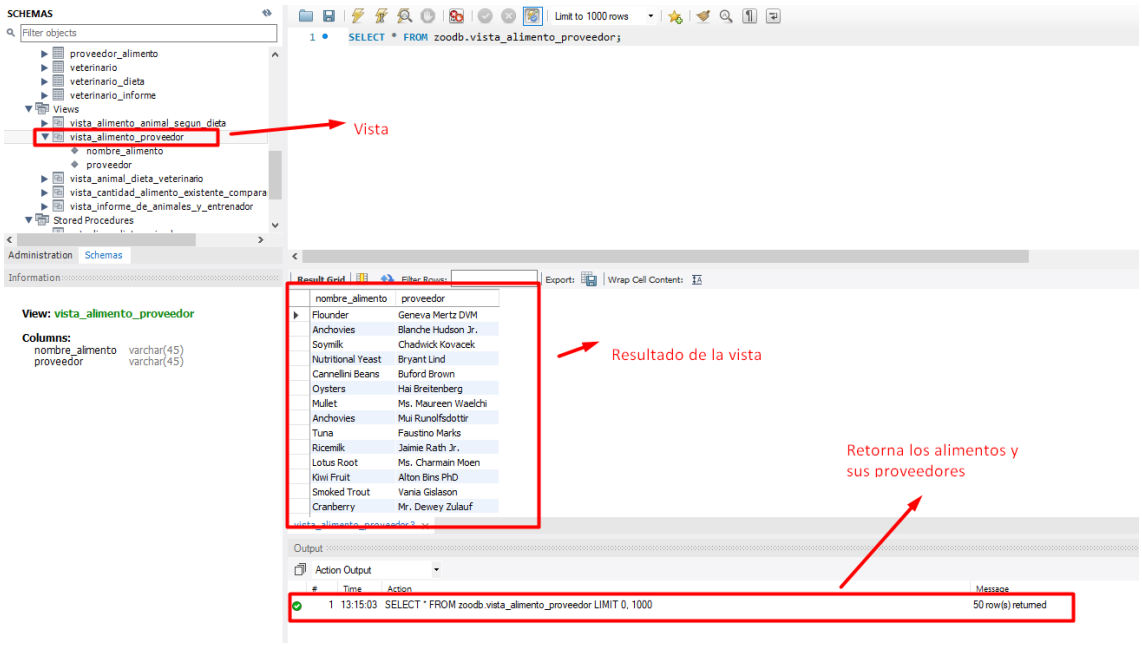
Esta vista es importante porque permite una visualización clara de la dieta de cada animal y el veterinario que la diseñó. Esto puede ser de gran utilidad para el personal encargado del cuidado de la salud y el bienestar de los animales. La dieta de un animal es un aspecto clave para llevar el registro 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;
```



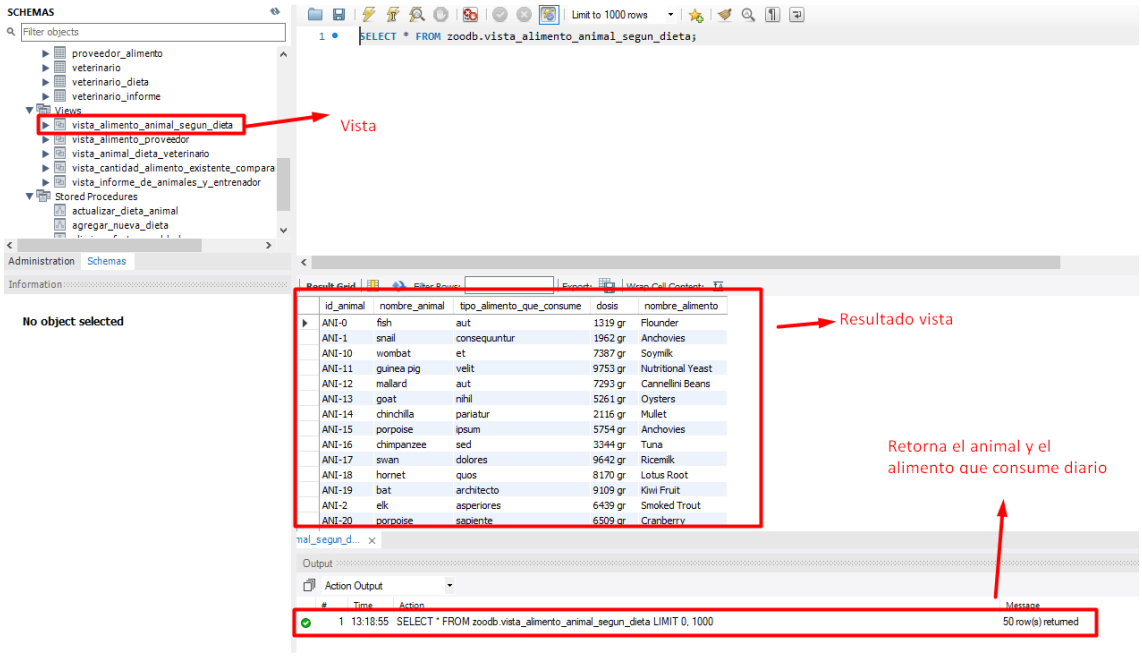
Esta vista es importante porque permite conocer de manera rápida y sencilla el proveedor de cada alimento que se utiliza en la alimentación de los animales. Esto es útil para el personal encargado de la gestión de compras en el zoológico, ya que les permite saber con facilidad qué proveedor les suministra 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 segun su dieta
-- -----------------------------------------------------
CREATE VIEW vista_alimento_animal_segun_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;
```

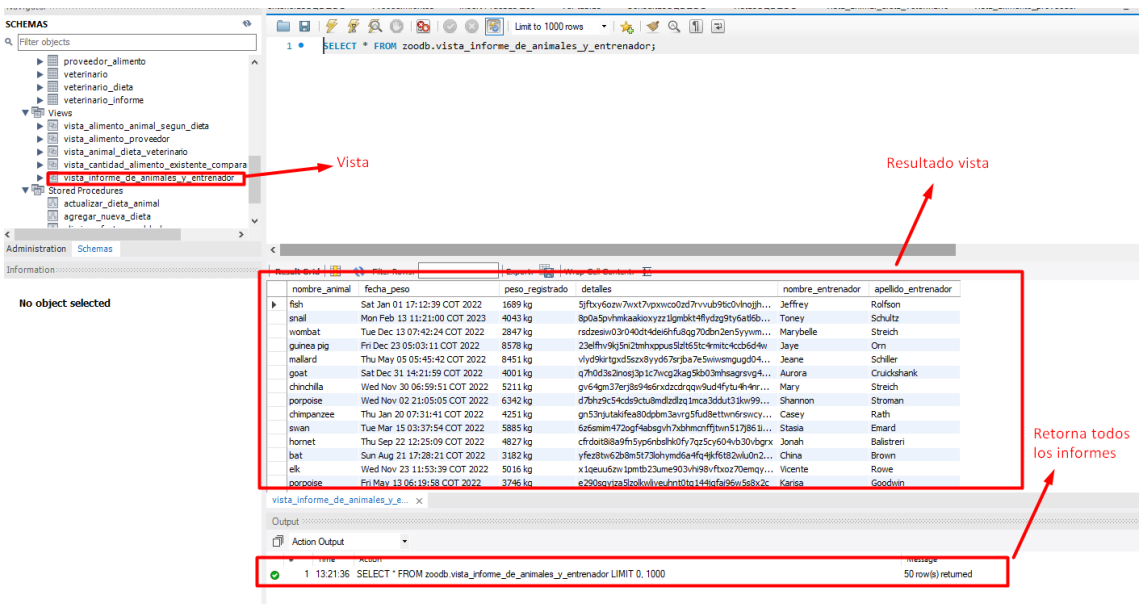
ZOO (Ejercicio B)



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.

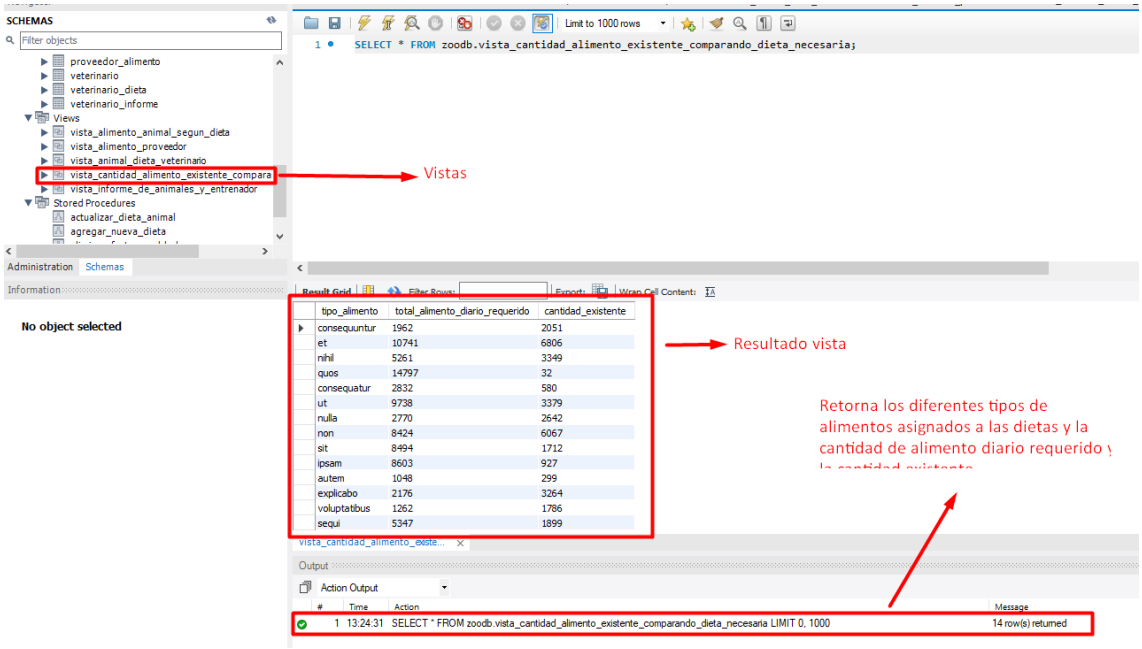
```
-- ------------------------------------------------------
-- 4. Vista para ver el informe de todos los animales y el entrenador
-- ------------------------------------------------------
CREATE VIEW vista_informe_de_animales_y_entrenador AS
SELECT animal.nombre AS nombre_animal, informe_animal.fecha_peso, 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;
```



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;
```



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.***

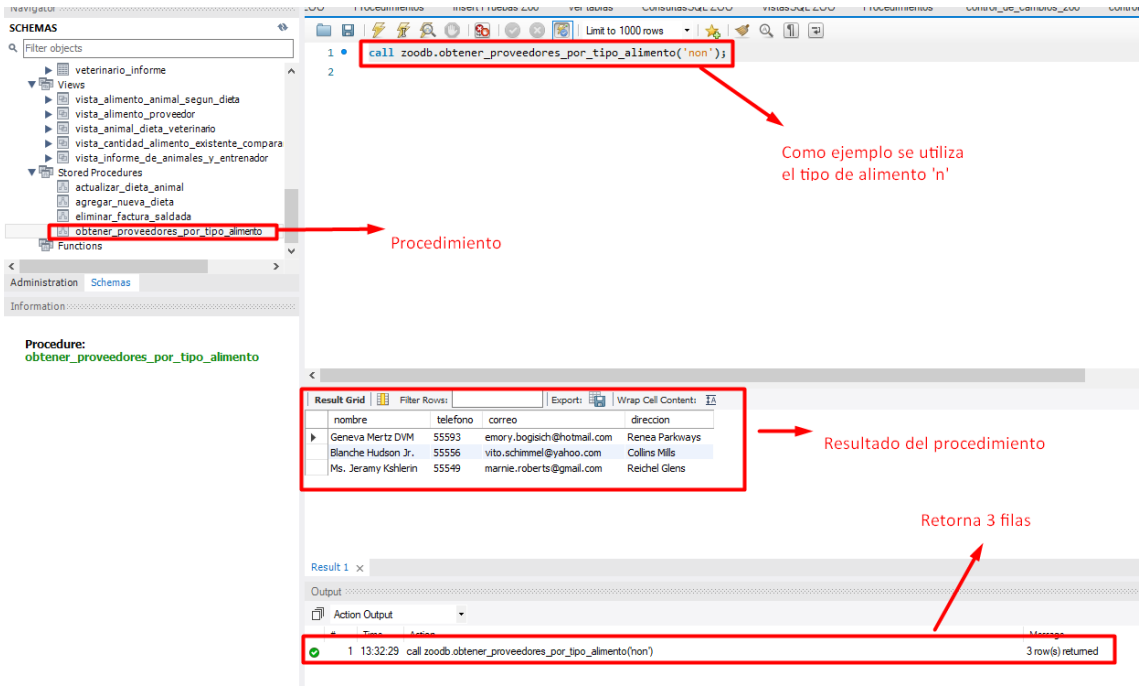
Además, esta vista también puede ser útil para fines de planificación y presupuestación, ya que permite estimar con precisión los costos de alimentación de los animales en función de las dosis requeridas por las diferentes dietas y los precios de los diferentes tipos de alimentos.

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 ;
```

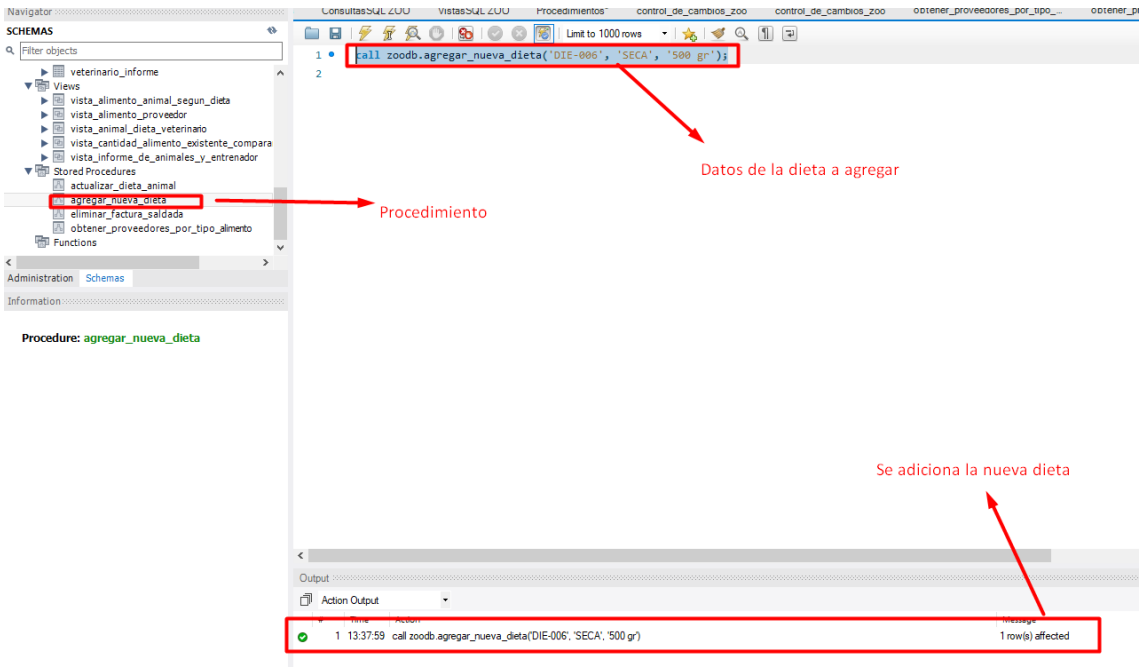
```
CALL obtener_proveedores_por_tipo_alimento('non');
```



2. Agregar nueva dieta

ZOO (Ejercicio B)

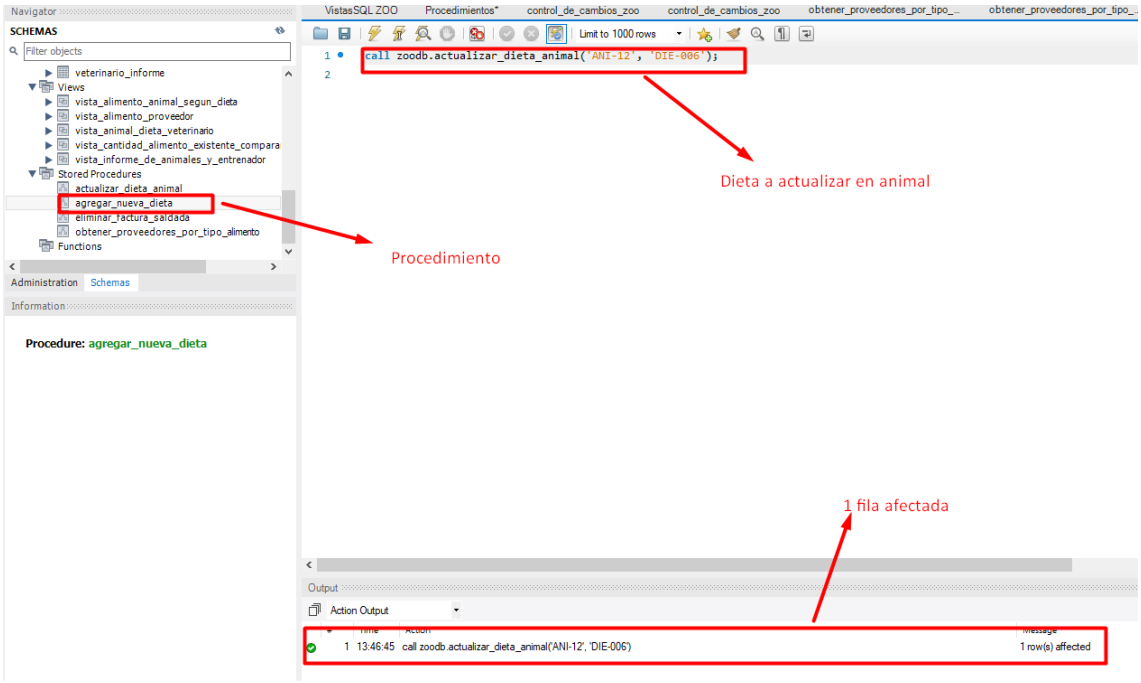
```
-- -----------------------------------------------------
-- 2. Agregar nueva dieta
-- -----------------------------------------------------
DELIMITER //
CREATE PROCEDURE agregar_nueva_dieta(
    dieta_id VARCHAR(20),
    alimento_tipo VARCHAR(45),
    alimentoDosis VARCHAR(45)
)
BEGIN
    INSERT INTO dieta (id_dieta, tipo_alimento, dosis) VALUES (dieta_id, alimento_tipo, alimentoDosis) ;
END//
DELIMITER ;

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



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 ;
```

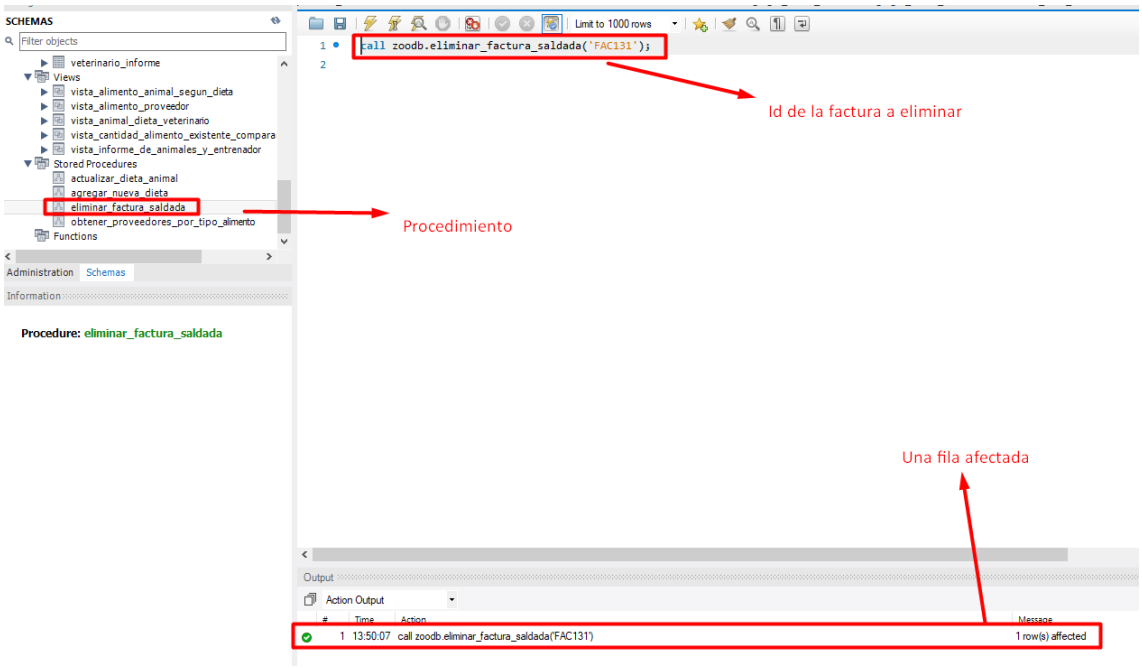
ZOO (Ejercicio B)



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 ;
```

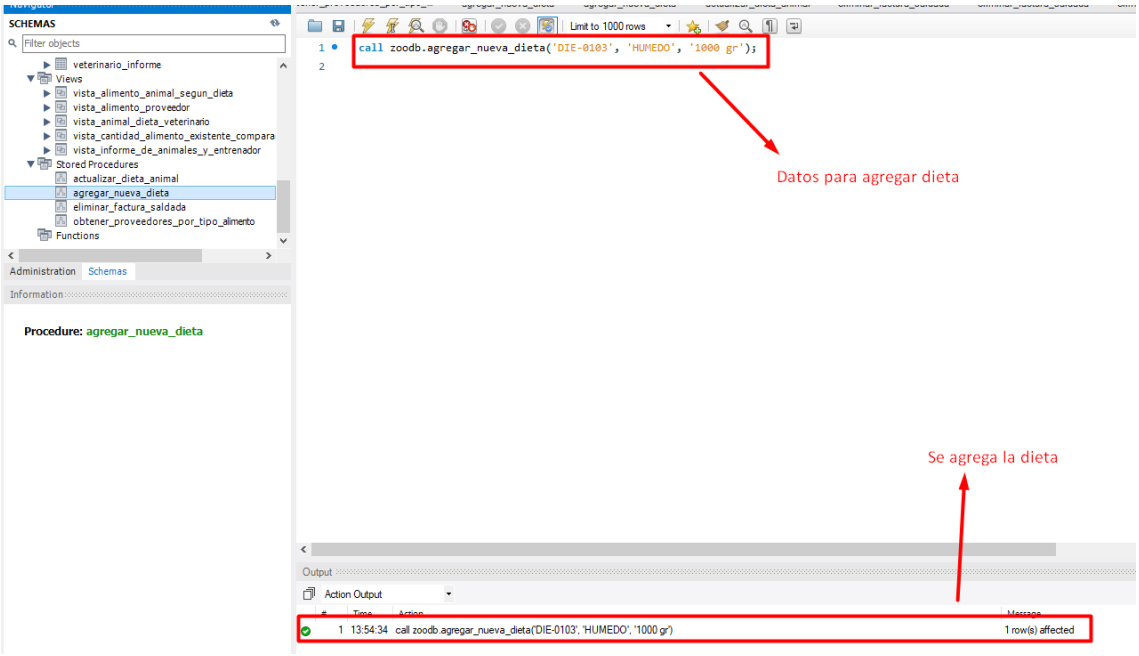
ZOO (Ejercicio B)

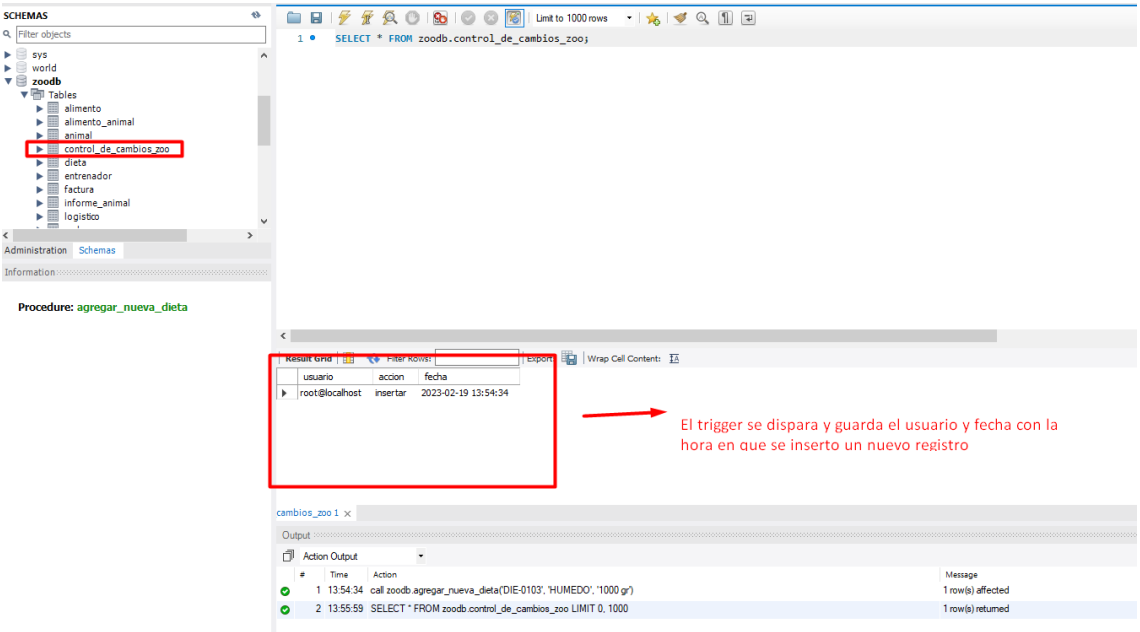


Triggers

```
-- ------------------------------------------------------
-- Creo 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;
//
```

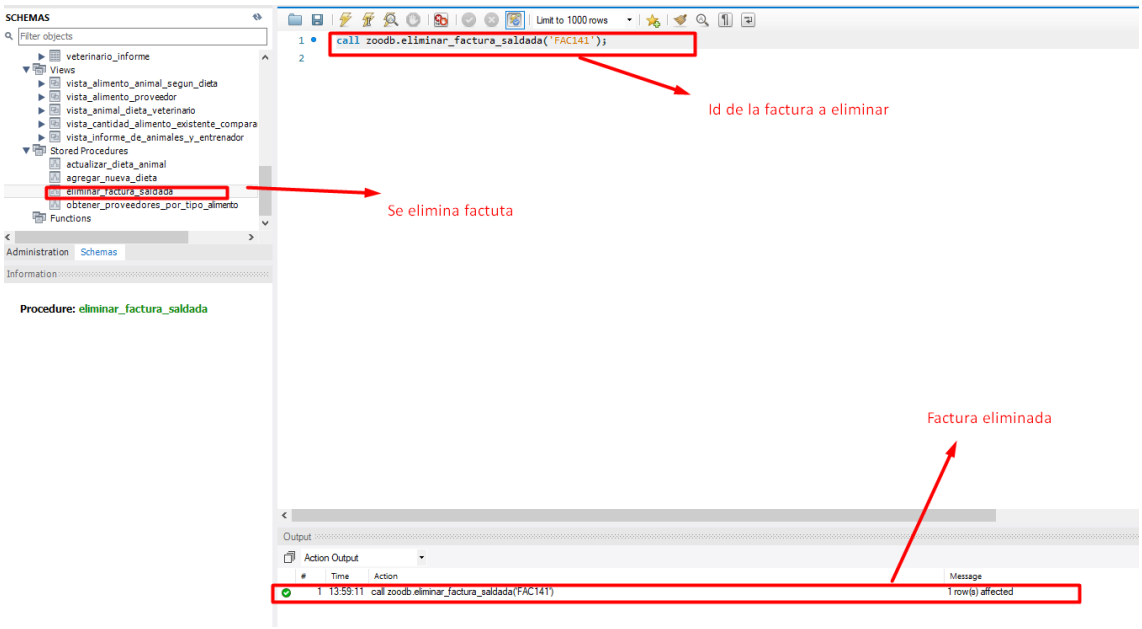
ZOO (Ejercicio B)

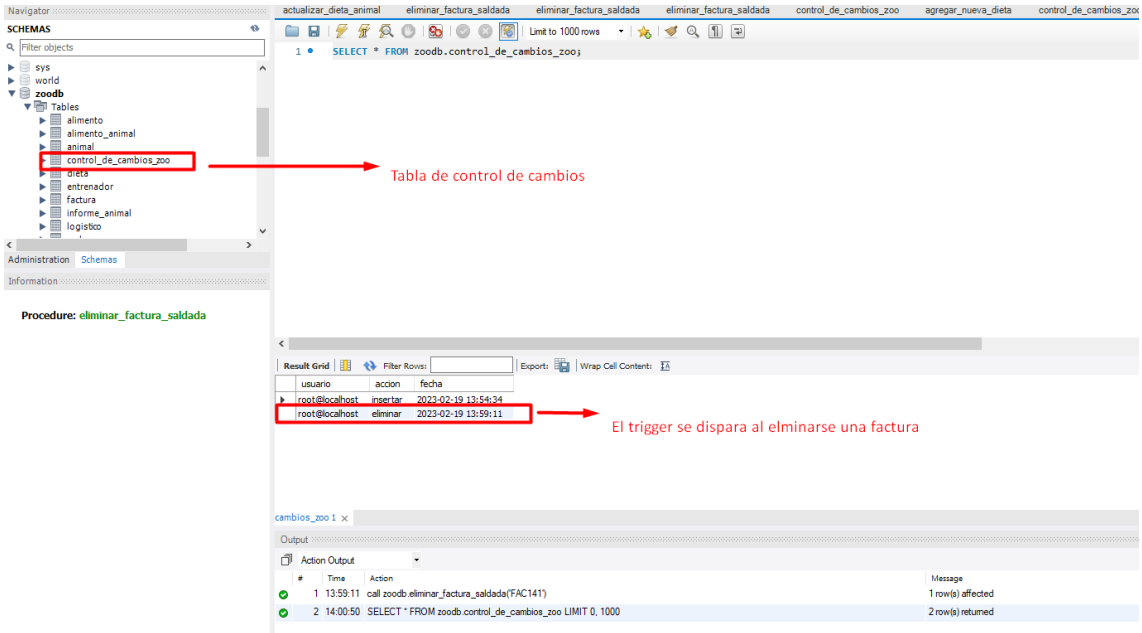





```
-- ---------------------------------------------------------
-- 2. Trigger Eliminar
-- ---------------------------------------------------------
DELIMITER //
CREATE TRIGGER control_zoo_delete_factura
AFTER DELETE ON factura
FOR EACH ROW
BEGIN
INSERT INTO control_de_cambios_zoo (usuario, accion, fecha)
VALUES (user(), 'eliminar', NOW());
END;
//
```

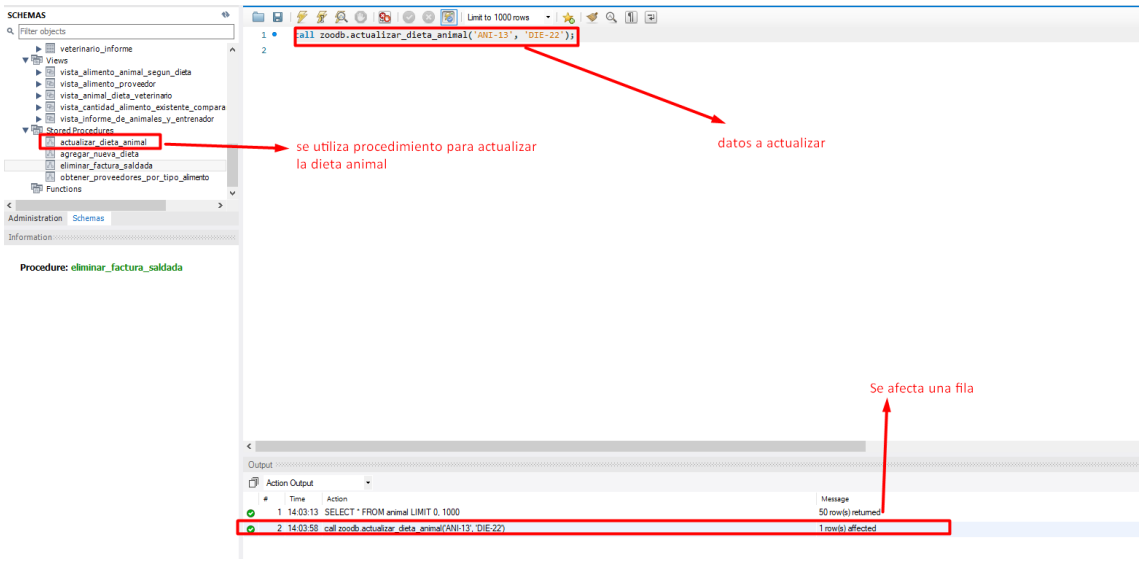
ZOO (Ejercicio B)

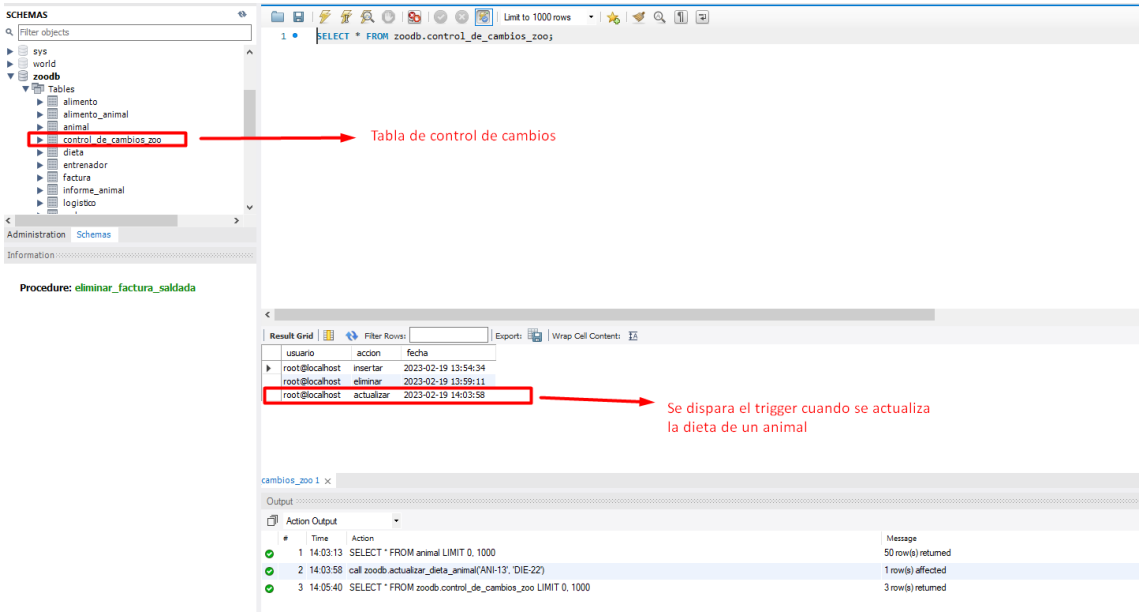





```
-- ------------------------------------------------------
-- 3. Trigger Actualizar Dieta Animal
-- ------------------------------------------------------
DELIMITER //
CREATE TRIGGER control_zoo_actualizar_dieta
AFTER UPDATE ON animal
FOR EACH ROW
BEGIN
INSERT INTO control_de_cambios_zoo (usuario,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

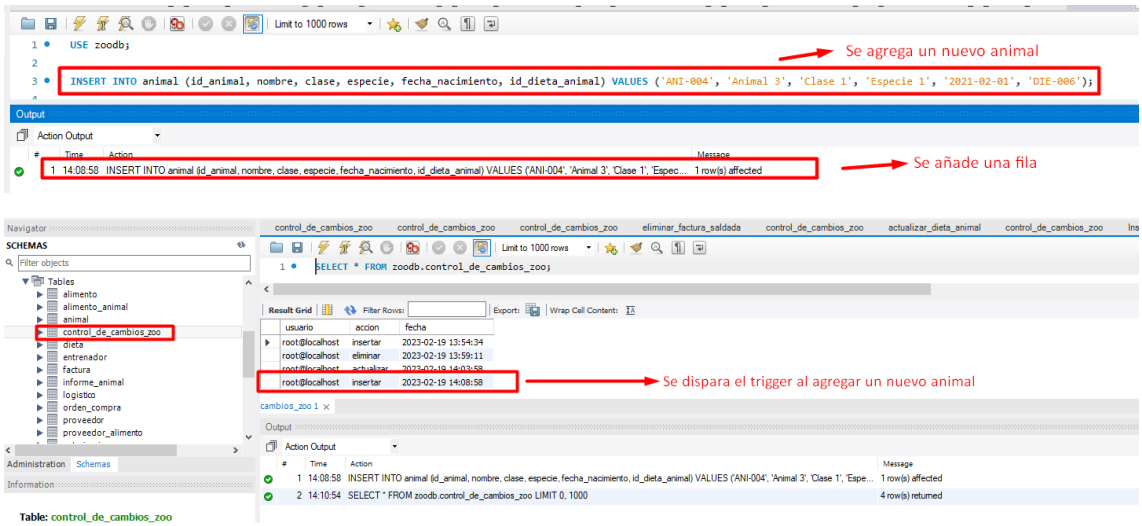
ZOO (Ejercicio B)






```
-- ------------------------------------------------------
-- 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)



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.

Método para poblar tabla de proveedores

ZOO (Ejercicio B)

```java
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( numberString: "##"));
        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


```sql
USE zoodb;

SELECT * FROM Proveedor;
SELECT * FROM factura;
SELECT * FROM logistico;
SELECT * FROM entrenador;
SELECT * FROM orden_compra;
SELECT * FROM alimento;
SELECT * FROM proveedor_alimento;
SELECT * FROM dieta;
SELECT * FROM animal;
SELECT * FROM alimento_animal;
```

Trae la tabla proveedores

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

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.pollich@hotmail.com	Kihn Port

Tabla proveedor

Retorna 50 filas

Proveedor 2

Output

Action Output

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

Método para poblar la tabla facturas

ZOO (Ejercicio B)

```java
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

```sql
USE zoodb;

SELECT * FROM Proveedor;
SELECT * FROM factura;
SELECT * FROM logistico;
SELECT * FROM entrenador;
SELECT * FROM orden_compra;
SELECT * FROM alimento;
SELECT * FROM proveedor_alimento;
SELECT * FROM dieta;
SELECT * FROM animal;
SELECT * FROM alimento_animal;
SELECT * FROM informe_animal;
SELECT * FROM veterinario;
SELECT * FROM veterinario_informe;
```

Trae la tabla factura

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

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

Rertorna filas

Output

Action Output

#	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)

```java
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:


```sql
USE zoodb;

SELECT * FROM Proveedor;
SELECT * FROM factura;
SELECT * FROM logistico;
SELECT * FROM entrenador;
SELECT * FROM orden_compra;
SELECT * FROM alimento;
SELECT * FROM proveedor_alimento;
SELECT * FROM dieta;
SELECT * FROM animal;
SELECT * FROM alimento_animal;
SELECT * FROM informe_animal;
SELECT * FROM veterinario;
SELECT * FROM veterinario_informe;
```

Trae la tabla logistico

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

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

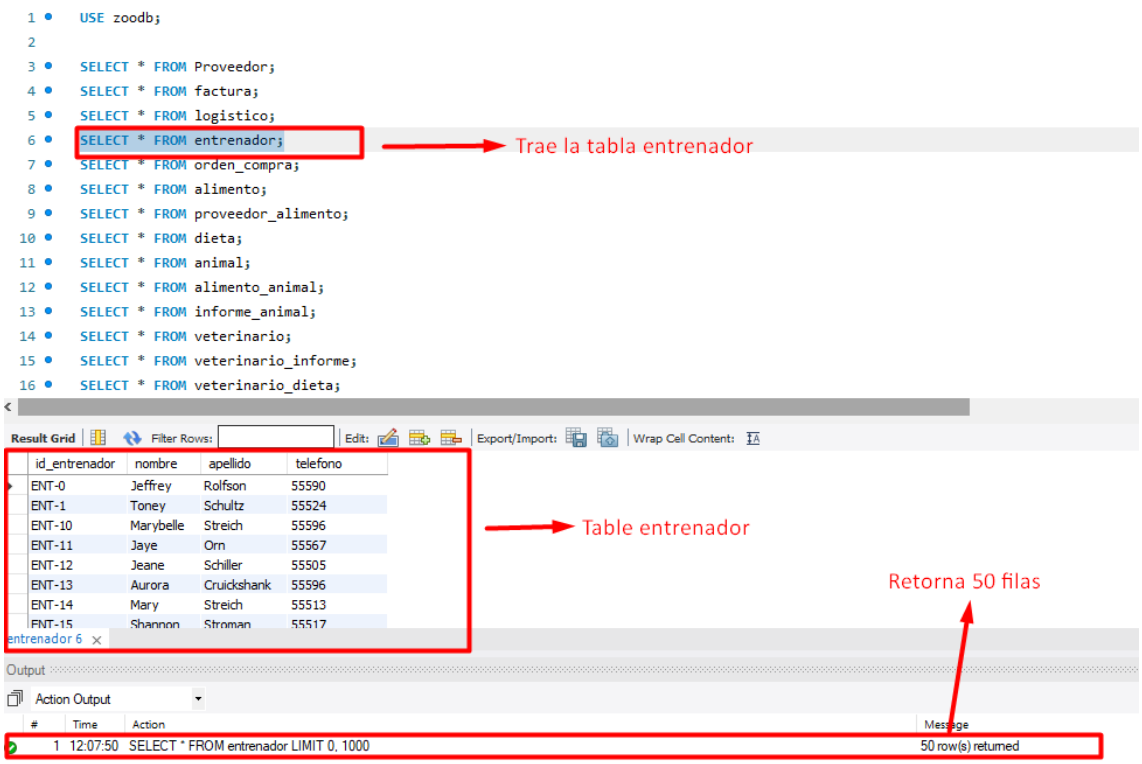
Action Output

#	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( numberString: "##"));

        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;
}
```




```
USE zoodb;

SELECT * FROM Proveedor;
SELECT * FROM factura;
SELECT * FROM logistico;
SELECT * FROM entrenador;
SELECT * FROM orden_compra;
SELECT * FROM alimento;
SELECT * FROM proveedor_alimento;
SELECT * FROM dieta;
SELECT * FROM animal;
SELECT * FROM alimento_animal;
SELECT * FROM informe_animal;
SELECT * FROM veterinario;
SELECT * FROM veterinario_informe;
SELECT * FROM veterinario_dieta;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

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	Cruickshank	55596
ENT-14	Mary	Streich	55513
ENT-15	Shannon	Stroman	55517

entrenador 6

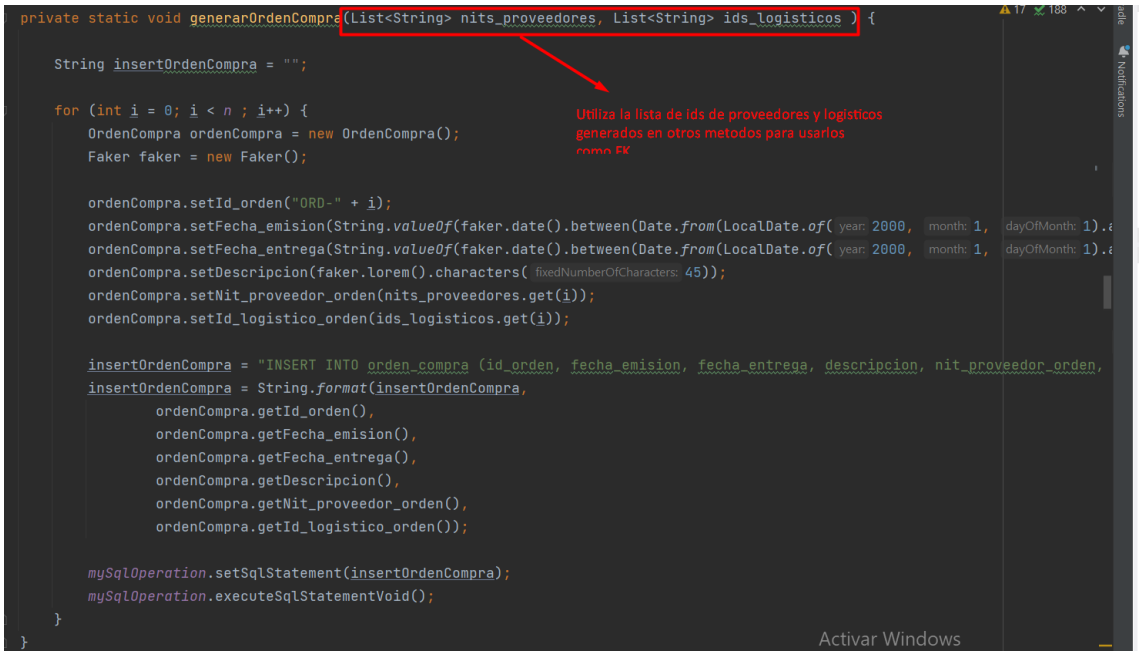
Output

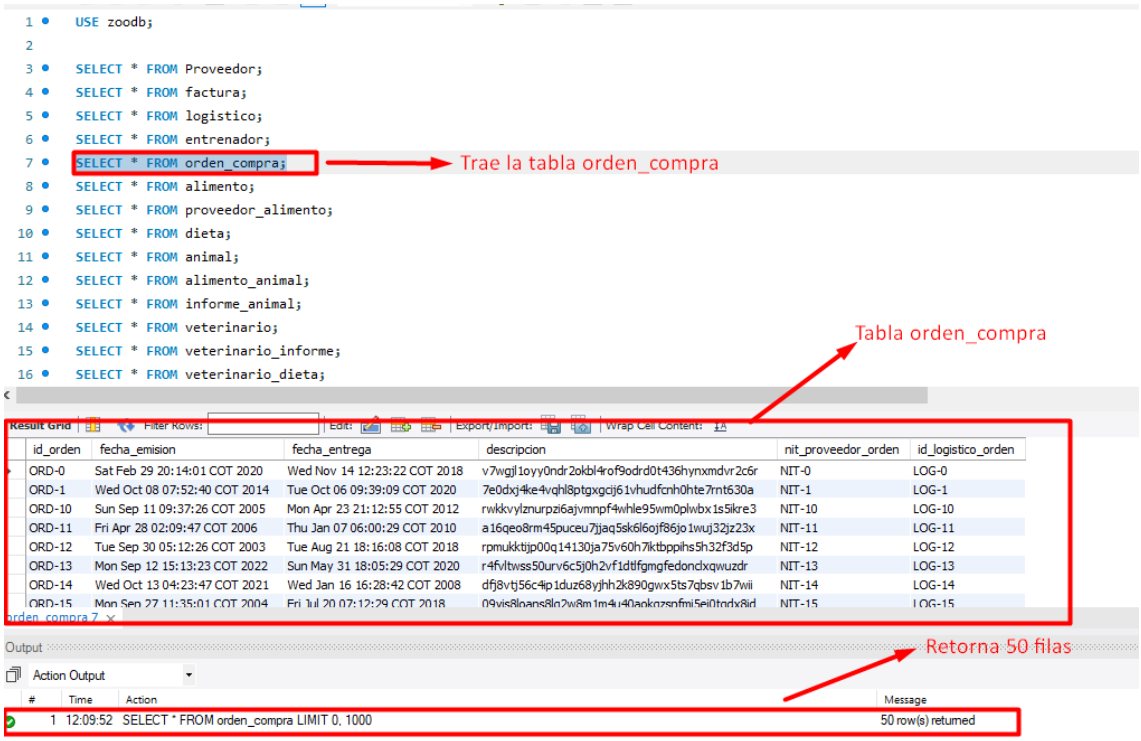
Action Output

#	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)





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	7e0dxj4ke4vqhl8ptgxgcij61vhudfcnh0hte7rnt630a	NIT-1	LOG-1
ORD-10	Sun Sep 11 09:37:26 COT 2005	Mon Apr 23 21:12:55 COT 2012	rwkkvylznurpzi6ajvmnpf4whle95wm0plwbx1s5ikre3	NIT-10	LOG-10
ORD-11	Fri Apr 28 02:09:47 COT 2006	Thu Jan 07 06:00:29 COT 2010	a16qeo8rm45puceu7jjaq5sk6l6ojf86jo1wuj32jz23x	NIT-11	LOG-11
ORD-12	Tue Sep 30 05:12:26 COT 2003	Tue Aug 21 18:16:08 COT 2018	rpmukktjjp00q14130ja75v60h7iktbppihs5h32f3d5p	NIT-12	LOG-12
ORD-13	Mon Sep 12 15:13:23 COT 2022	Sun May 31 18:05:29 COT 2020	r4fvltwss50urv6c5j0h2vf1dtlfgmgfedonclxqwuzdr	NIT-13	LOG-13
ORD-14	Wed Oct 13 04:23:47 COT 2021	Wed Jan 16 16:28:42 COT 2008	dfj8vtj56c4ip1duz68yjhh2k890gwx5ts7qbsv1b7wii	NIT-14	LOG-14
ORD-15	Mon Sep 27 11:35:01 COT 2004	Fri Jul 20 07:12:29 COT 2018	09vis8loans8lo2w8m1m4u40aokqzsnfmi5ei0tqdx8id	NIT-15	LOG-15

Método para poblar tabla alimento.

ZOO (Ejercicio B)

```java
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( obj: faker.number().numberBetween(10, 5000)+ " kg"));

        insertAlimento = "INSERT INTO alimento (id_alimento, nombre_alimento, tipo_alimento, cantidad)  VALUES('%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

```sql
USE zoodb;

SELECT * FROM Proveedor;
SELECT * FROM factura;
SELECT * FROM logistico;
SELECT * FROM entrenador;
SELECT * FROM orden_compra;
SELECT * FROM alimento;
SELECT * FROM proveedor_alimento;
SELECT * FROM dieta;
SELECT * FROM animal;
SELECT * FROM alimento_animal;
SELECT * FROM informe_animal;
SELECT * FROM veterinario;
SELECT * FROM veterinario_informe;
SELECT * FROM veterinario_dieta;
```

Trae la tabla alimento

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

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 kg

Tabla alimento

Retorna 50 filas

Output

Action Output

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

Método para poblar tabla proveedor_alimento

ZOO (Ejercicio B)

```java
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

```sql
USE zoodb;

SELECT * FROM Proveedor;
SELECT * FROM factura;
SELECT * FROM logistico;
SELECT * FROM entrenador;
SELECT * FROM orden_compra;
SELECT * FROM alimento;
SELECT * FROM proveedor_alimento;
SELECT * FROM dieta;
SELECT * FROM animal;
SELECT * FROM alimento_animal;
SELECT * FROM informe_animal;
SELECT * FROM veterinario;
SELECT * FROM veterinario_informe;
SELECT * FROM veterinario_dieta;
```

Trae la tabla proveedor_alimento

Result Grid | Filter Rows: | Edit | Export/Import: | Wrap Cell Content:

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

proveedor_alimento 9

Output

Action Output

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

Retorna 50 filas

Método para poblar la tabla dieta.

ZOO (Ejercicio B)

```java
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

```sql
USE zoodb;

SELECT * FROM Proveedor;
SELECT * FROM factura;
SELECT * FROM logistico;
SELECT * FROM entrenador;
SELECT * FROM orden_compra;
SELECT * FROM alimento;
SELECT * FROM proveedor_alimento;
SELECT * FROM dieta;
SELECT * FROM animal;
SELECT * FROM alimento_animal;
SELECT * FROM informe_animal;
SELECT * FROM veterinario;
SELECT * FROM veterinario_informe;
SELECT * FROM veterinario_dieta;
```

Trae tabla dieta

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

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

Action 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)

```java
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)
        animal.setId_dieta_animal(id_dietas.get(i));

        insertAnimal = "INSERT INTO animal (id_animal, nombre, clase, especie, fecha_nacimiento, id_dieta_animal) VALUES ('%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 tabl


```sql
USE zoodb;

SELECT * FROM Proveedor;
SELECT * FROM factura;
SELECT * FROM logistico;
SELECT * FROM entrenador;
SELECT * FROM orden_compra;
SELECT * FROM alimento;
SELECT * FROM proveedor_alimento;
SELECT * FROM dieta;
SELECT * FROM animal;
SELECT * FROM alimento_animal;
SELECT * FROM informe_animal;
SELECT * FROM veterinario;
SELECT * FROM veterinario_informe;
SELECT * FROM veterinario_dieta;
```

Trae tabla animal

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

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	porpoise	hamster	sardine	Sun Oct 12 23:15:36 COT 2014	DIE-15

Tabla animal

Retorna 50 filas

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)

```java
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 tabl

```sql
USE zoodb;

SELECT * FROM Proveedor;
SELECT * FROM factura;
SELECT * FROM logistico;
SELECT * FROM entrenador;
SELECT * FROM orden_compra;
SELECT * FROM alimento;
SELECT * FROM proveedor_alimento;
SELECT * FROM dieta;
SELECT * FROM animal;
SELECT * FROM alimento_animal;
SELECT * FROM informe_animal;
SELECT * FROM veterinario;
SELECT * FROM veterinario_informe;
SELECT * FROM veterinario_dieta;
```

Trae tabla alimento_animal

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

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

Output

Action Output

#	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)

```java
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,
        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
        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

Activar Windows

```sql
USE zoodb;

SELECT * FROM Proveedor;
SELECT * FROM factura;
SELECT * FROM logistico;
SELECT * FROM entrenador;
SELECT * FROM orden_compra;
SELECT * FROM alimento;
SELECT * FROM proveedor_alimento;
SELECT * FROM dieta;
SELECT * FROM animal;
SELECT * FROM alimento_animal;
SELECT * FROM informe_animal;
SELECT * FROM veterinario;
SELECT * FROM veterinario_informe;
SELECT * FROM veterinario_dieta;
```

Trae tabla informe_animal

Tabla informe_animal

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

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	5jftxy6ozw7wxt7vpxwco0zd7rvvub9tic0vlnojjh...	ANI-0	ENT-0
INF-1	Mon Feb 13 11:21:00 COT 2023	4043 kg	8p0a5pvhmkaakioxyzz1lgmbkt4flydzg9ty6atl6b...	ANI-1	ENT-1
INF-10	Tue Dec 13 07:42:24 COT 2022	2847 kg	rsdzesiw03r040dt4dei6hfu8qg70dbn2en5yywm...	ANI-10	ENT-10
INF-11	Fri Dec 23 05:03:11 COT 2022	8578 kg	23elfhv9kj5ni2tmhxppus5lzlt65tc4rmitc4ccb6d4w	ANI-11	ENT-11
INF-12	Thu May 05 05:45:42 COT 2022	8451 kg	vlyd9kirtgxd5szx8yyd67srjba7e5wiwsmgugd04...	ANI-12	ENT-12
INF-13	Sat Dec 31 14:21:59 COT 2022	4001 kg	q7h0d3s2inosj3p1c7wcg2kag5kb03mhsagrsvg4...	ANI-13	ENT-13
INF-14	Wed Nov 30 06:59:51 COT 2022	5211 kg	gv64gm37erj8s94s6rxdzcdrqqw9ud4fytu4h4nr...	ANI-14	ENT-14
INF-15	Wed Nov 02 21:05:05 COT 2022	6342 kg	d7bhz9c54cds9ctu8mdlzdlzq1mca3ddut31kw99...	ANI-15	ENT-15

informe_animal 13

Output

Action Output

Retorna 50 filas

#	Time	Action	Message
1	12:20:13	SELECT * FROM informe_animal LIMIT 0, 1000	50 row(s) returned

Método para poblar tabla veterinario.

ZOO (Ejercicio B)

```java
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


```sql
USE zoodb;

SELECT * FROM Proveedor;
SELECT * FROM factura;
SELECT * FROM logistico;
SELECT * FROM entrenador;
SELECT * FROM orden_compra;
SELECT * FROM alimento;
SELECT * FROM proveedor_alimento;
SELECT * FROM dieta;
SELECT * FROM animal;
SELECT * FROM alimento_animal;
SELECT * FROM informe_animal;
SELECT * FROM veterinario;
SELECT * FROM veterinario_informe;
SELECT * FROM veterinario_dieta;
```

Trae tabla veterinaio

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

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	Sammie	Grady	55575

Tabla veterinario

veterinario 14

Output

Action Output

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

Retorna 50 filas

Método para poblar tabla veterinario_informe

ZOO (Ejercicio B)

```java
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


```sql
USE zoodb;

SELECT * FROM Proveedor;
SELECT * FROM factura;
SELECT * FROM logistico;
SELECT * FROM entrenador;
SELECT * FROM orden_compra;
SELECT * FROM alimento;
SELECT * FROM proveedor_alimento;
SELECT * FROM dieta;
SELECT * FROM animal;
SELECT * FROM alimento_animal;
SELECT * FROM informe_animal;
SELECT * FROM veterinario;
SELECT * FROM veterinario_informe;
SELECT * FROM veterinario_dieta;
```

Trae tabla veterinario_informe

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

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_infome LIMIT 0, 1000	50 row(s) returned

Método para poblar tabla veterinario_dieta

ZOO (Ejercicio B)

```java
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

```sql
USE zoodb;

SELECT * FROM Proveedor;
SELECT * FROM factura;
SELECT * FROM logistico;
SELECT * FROM entrenador;
SELECT * FROM orden_compra;
SELECT * FROM alimento;
SELECT * FROM proveedor_alimento;
SELECT * FROM dieta;
SELECT * FROM animal;
SELECT * FROM alimento_animal;
SELECT * FROM informe_animal;
SELECT * FROM veterinario;
SELECT * FROM veterinario_informe;
SELECT * FROM veterinario_dieta;
```

Trae tabla veterinario_dieta

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

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

Output

Action Output

#	Time	Action	Message
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.