Ejercicio ZOO (Ejercicio B)

El parque zoo Santafé “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 él 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 los hábitats 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.

1. Modelo Entidad Relación

Teniendo en cuenta el planteamiento, se procede a definir las entidades con sus respectivos atributos, con las cuales se genera el Modelo Entidad Relación de la figura 1. El diagrama se encuentra en **diagramas/MER-Zoo**

Entidades:

- Animal:
  - Id (Identificador único)
  - Especie. Puede ser (mamíferos, aves, anfibios)
- Entrenador:
  - Id (Identificador único)
  - Teléfono
  - Correo
  - Nombre
- Reporte:
  - Id (Identificador único)
  - Peso_animal
  - fecha
- Veterinario:
  - Id (Identificador único)
  - Nombre
  - Correo
  - Teléfono
- Empleado_logístico:
  - Id (Identificador único)
  - Correo
  - Nombre
  - Teléfono
- Orden_compra:
  - Id (Identificador único)
  - Fecha
  - unidades
- Proveedor:
  - Id (Identificador único)
  - Nombre
  - Dirección
  - Teléfono
- Factura:
  - Id (Identificador único)
  - Fecha
  - unidades
  - Total
- Alimento:
  - Id (Identificador único)
  - Nombre
  - Tipo
  - cantidad
- Dieta:
  - Id (Identificador único)
  - Nombre
  - Dosis

Entidades como Visitante, Gerente y Cuidador se descartan; ya que no aportan información valiosa a la hora de llevar un registro de los alimentos consumidos por los animales del zoológico.

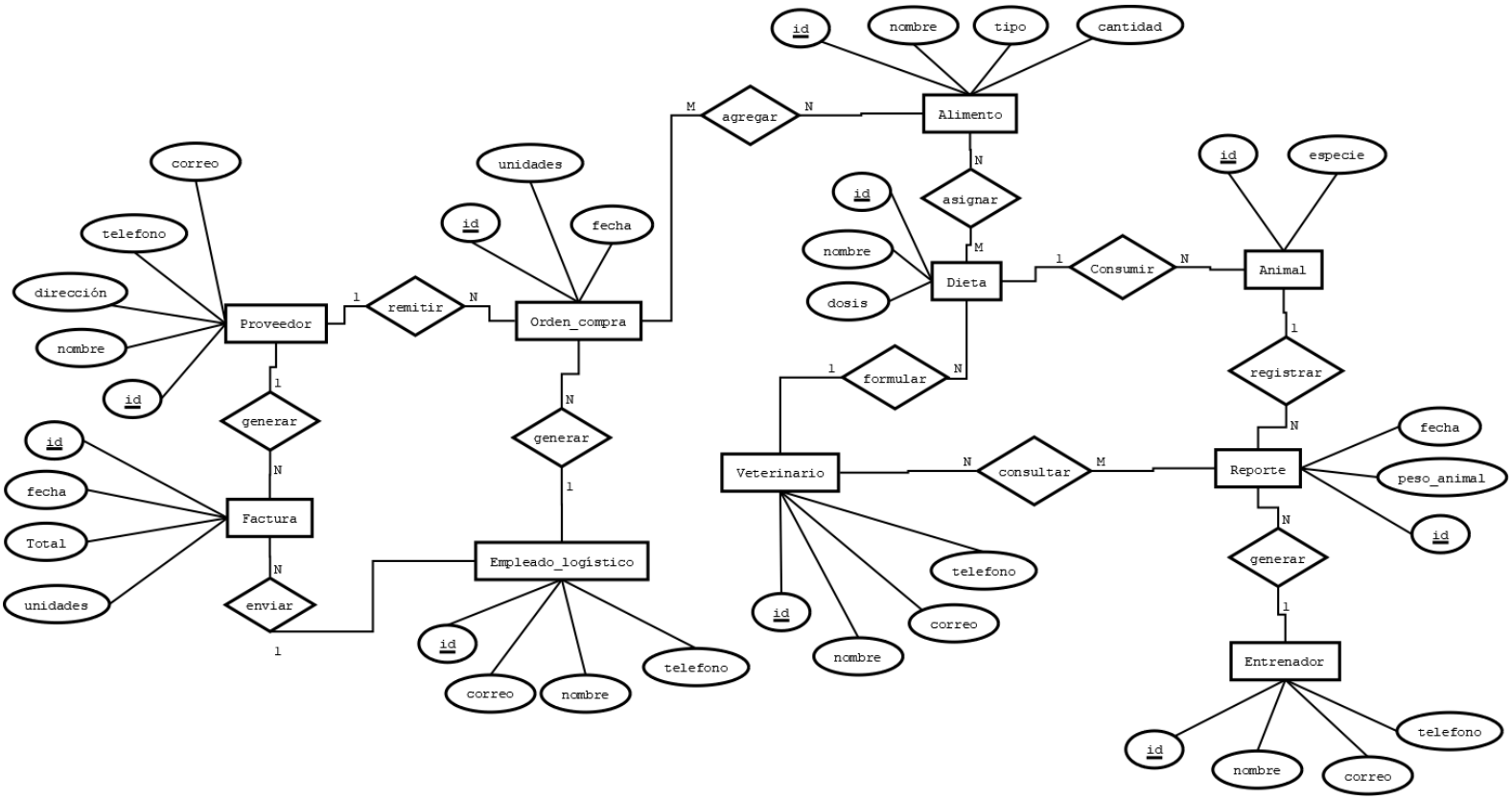

Figura 1: Modelo Entidad Relación ZOO

Cardinalidad:

- **Registrar**: Un Animal puede ser registrado en uno o varios Reportes. Un Reporte registra información de un Animal.
- **Consumir**: Un Animal puede consumir una Dieta. Una Dieta puede ser consumida por uno o varios Animales.
- **Generar**: Un Entrenador puede generar uno o varios Reportes. Un Reporte puede ser generado por un Entrenador.
- **Consultar**: Un Veterinario puede consultar uno o varios Reportes. Un Reporte puede ser consultado por uno o varios Veterinarios.
- **Formular:** Un Veterinario puede formular una o varias Dietas. Una Dieta puede formulada por un Veterinario.
- **Generar:** Un Empleado_logistico puede generar una o varias ordenes_compra. Una orden_compra puede ser generada por un Empleado_logistico.

- **Remitir:** Una Orden_compra puede ser remitida a un Proveedor. Un Proveedor puede ser remitente de una o varias Orden_compra.
- **Generar:** Un Proveedor puede generar una o varias Facturas. Una Factura puede ser generada por un Proveedor.
- **Agregar:** Un Alimento puede ser agregado a una o varias Orden_servicio. Una Orden_servicio puede agregar uno o varios Alimentos.
- **Enviar:** Una Factura puede ser enviada a un Empleado_logistico (quien envió la Orden_compra). Un Empleado_logistico puede recibir una o varias Facturas.
- **Asignar:** Un Alimento puede ser asignado a una Dieta. Una Dieta puede tener asignado uno o varios Alimentos.

Teniendo en cuenta que no se definen relaciones uno a uno, no se toma en cuenta la participación entre las entidades y sus relaciones.

2. Ahora se procede a transformar el MER de la figura 1 para obtener el Modelo Relacional de la figura 2. El MR se encuentra en **diagramas/MR-Zoo**

   Para realizar dicha transformación, se sigue un conjunto de pasos:

   - Primero se convierte cada entidad en una tabla.
   - Cada atributo de la entidad se transforma en una columna de la tabla
   - El identificador único de la entidad se transforma en llave primaria
   - En las relaciones N:M (**Alimento-Orden_compra, Alimento-Dieta y Reporte-Veterinario**), se genera una tabla intermedia cuya clave primaria es la concatenación de los atributos clave de cada entidad que relaciona. De esta transformación surgen las tablas Alimento_Orden_Compra, Alimento_dieta y Reporte_veterinario que se observan en la figura 2.
   - En las relaciones 1: N, la entidad del lado N de la relación añade como llave foránea la llave primaria de la entidad del lado 1. Este es el tipo de relación entre las entidades **Animal-Reporte, Entrenador-Reporte, Veterinario-Dieta, Empleado_logistico-Orden_compra, Orden_compra-Proveedor, Proveedor-Factura y Factura-Empleado_logistico**.

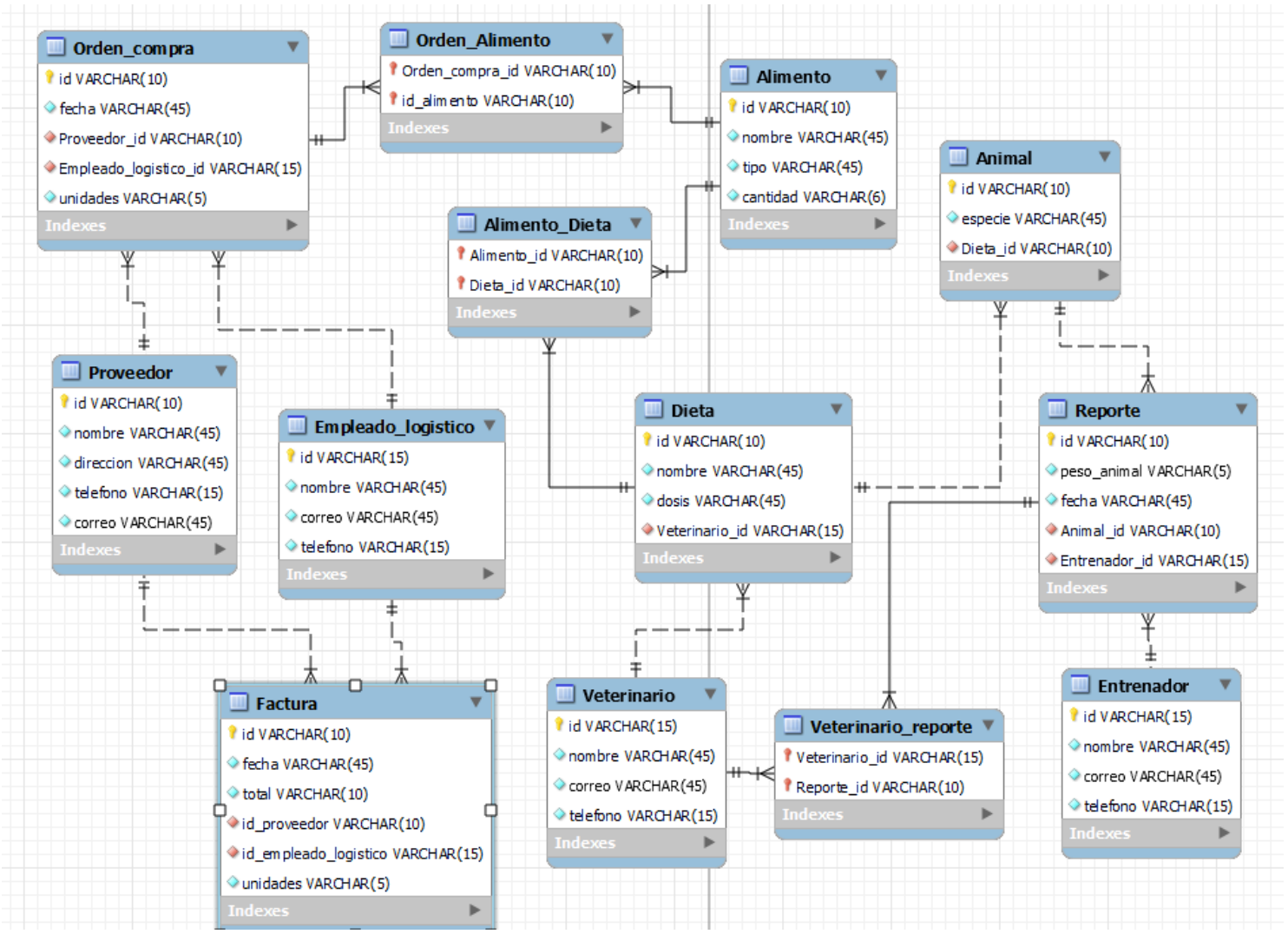

Figura 2: Modelo Relacional Zoo

3. A continuación, se realiza el proceso de normalización.

Tabla 1: Primera Forma Normal

Cada atributo tiene valores atómicos	cumple
No hay atributos multivaluados	cumple
No hay registros duplicados	Cumple
No hay columnas duplicadas	Cumple
Cada tabla tiene definida una clave principal	Cumple

Tabla 2: Segunda Forma Normal

Se cumple la 1FN	Cumple
Los valores de las tablas dependen solo de la llave primaria	Cumple
Las tablas tienen una única llave primaria que las identifica	cumple

Tabla 3: Tercera Forma Normal

Se cumple la 2FN	Cumple
Los atributos no incluidos en la clave primaria no dependen transitivamente de la llave primaria	cumple

4. A continuación, se presenta el script implementado para crear cada una de las tablas de la Base de Datos del zoológico. Este puede ser encontrado en **scripts/script-create-tables-Zoo**

```
1     -- ---------------------------------
2     -- Se crea la tabla Veterinario
3     -- ---------------------------------
4
5     create table zoologico.veterinario (
6         id varchar(15) not null,
7         nombre varchar(45) not null,
8         correo varchar(45) not null,
9         telefono varchar(15) not null,
10        primary key (id));
11
12    -- ---------------------------------------
13    -- Se crea la tabla Dieta
14    -- ---------------------------------------
15
16    create table zoologico.dieta (
17        id varchar(10) not null,
18        nombre varchar(45) not null,
19        dosis varchar(45) not null,
20        veterinario_id varchar(15) not null,
21        primary key (id),
22        foreign key (veterinario_id) references zoologico.veterinario (id));
23
```

```sql
-- ---------------------------------------
-- Se crea la tabla Animal
-- ---------------------------------------

create table zoologico.animal (
    id varchar(10) not null,
    especie varchar(45) not null,
    dieta_id varchar(10) not null,
    primary key (id),
    foreign key (dieta_id) references zoologico.dieta (id));

-- ---------------------------------------
-- Se crea la tabla Entrenador
-- ---------------------------------------

create table zoologico.entrenador (
    id varchar(15) not null,
    nombre varchar(45) not null,
    correo varchar(45) not null,
    telefono varchar(15) not null,
    primary key (id));


-- ---------------------------------------
-- Se crea la tabla Reporte
-- ---------------------------------------

create table zoologico.reporte (
    id varchar(10) not null,
    peso_animal varchar(5) not null,
    fecha varchar(45) not null,
    animal_id varchar(10) not null,
    entrenador_id varchar(15) not null,
    primary key (id),
    foreign key (animal_id) references zoologico.animal (id),
    foreign key (entrenador_id) references zoologico.entrenador (id));

-- ---------------------------------------
-- Se crea la tabla Alimento
-- ---------------------------------------

create table zoologico.alimento (
    id varchar(10) not null,
    nombre varchar(45) not null,
    tipo varchar(45) not null,
    cantidad varchar(6) not null,
    primary key (id));

```

```
71      -- ---------------------------------------
72      -- Se crea la tabla Empleado_logistico
73      -- ---------------------------------------
74
75  ●   create table zoologico.empleado_logistico (
76          id varchar(15) not null,
77          nombre varchar(45) not null,
78          correo varchar(45) not null,
79          telefono varchar(15) not null,
80          primary key (id));
81
82      -- -----------------------------------------
83      -- Se crea la tabla Proveedor
84      -- -----------------------------------------
85
86  ●   create table zoologico.proveedor (
87          id varchar(10) not null,
88          nombre varchar(45) not null,
89          direccion varchar(45) not null,
90          telefono varchar(15) not null,
91          correo varchar(45) not null,
92          primary key (id));
93
98  ●   create table zoologico.orden_compra (
99          id varchar(10) not null,
100         fecha varchar(45) not null,
101         unidades varchar(5) not null,
102         proveedor_id varchar(10) not null,
103         empleado_logistico_id varchar(15) not null,
104         primary key (id),
105         foreign key (proveedor_id) references zoologico.proveedor (id),
106         foreign key (empleado_logistico_id)
107         references zoologico.empleado_logistico (id));
108
109     -- ---------------------------------------
110     -- Se crea la tabla Factura
111     -- ---------------------------------------
112
113 ●   create table zoologico.factura (
114         id varchar(10) not null,
115         fecha varchar(45) not null,
116         unidades varchar(5) not null,
117         total varchar(10) not null,
118         id_proveedor varchar(10) not null,
119         id_empleado_logistico varchar(15) not null,
120         primary key (id),
121         foreign key (id_proveedor) references zoologico.proveedor (id),
122         foreign key (id_empleado_logistico)
123         references zoologico.empleado_logistico (id));
```

```sql
-- ------------------------------------------------
-- Se crea la tabla Veterinario_reporte
-- ------------------------------------------------

create table zoologico.veterinario_reporte (
    veterinario_id varchar(15) not null,
    reporte_id varchar(10) not null,
    primary key (veterinario_id, reporte_id),
    foreign key (veterinario_id) references zoologico.veterinario (id),
    foreign key (reporte_id) references zoologico.reporte (id));

-- -------------------------------------
-- Se crea la tabla Alimento_dieta
-- -------------------------------------

create table zoologico.alimento_dieta (
    alimento_id varchar(10) not null,
    dieta_id varchar(10) not null,
    primary key (alimento_id, dieta_id),
    foreign key (dieta_id) references zoologico.dieta (id),
    foreign key (alimento_id) references zoologico.alimento (id));

-- ------------------------------------------------
-- Se crea la tabla Orden_alimento
-- ------------------------------------------------

create table zoologico.orden_alimento (
    orden_compra_id varchar(10) not null,
    alimento_id varchar(10) not null,
    primary key (orden_compra_id, alimento_id),
    foreign key (orden_compra_id)
    references zoologico.orden_compra (id),
    foreign key (alimento_id)
    references zoologico.alimento (id));
```

5. Ahora se generan 10 consultas que me permitan obtener información valiosa de la base de datos diseñada. Estas consultas se encuentran en scripts/script-consultas

   Consulta 1: se implementa una consulta para visualizar reportes asociados a un animal. Para conocer el historial de reportes de un animal, se debe ingresar su id (llave foránea que relacional las tablas Reporte y Animal). En este caso se toma un reporte.animal_id de ejemplo. En la tabla resultante se observan los valores de los atributos de la tabla Reporte: id, peso_animal, fecha, animal_id y entrenador_id.

```
3      -- -------------------------------------------------------
4      -- Consulta para visualizar Reportes de un Animal
5      -- -------------------------------------------------------
6
7  •   select * from reporte where reporte.animal_id = '001';
8
```

Figura 3: Consulta 1

Consulta 2: consulta que permite visualizar la cantidad disponible de un alimento. En la tabla resultante se pueden observar los valores de la tabla Alimento; nombre y cantidad de alimento disponible.

```
9      -- -------------------------------------------------------
10     -- Consulta para visualizar la cantidad de Alimento
11     -- -------------------------------------------------------
12
13 •   select nombre, cantidad from alimento;
14
```

Figura 4: Consulta 2

Consulta 3: consulta que permite visualizar las ordenes de compra generadas por un empleado logístico. La tabla orden_compra, posee una llave foránea que relacional esta tabla con la de Empleado_logistico. En la tabla resultante se pueden obsevar los atributos de la tabla Orden_compra: id, fecha, unidades, proveedor_id y empleado_logistico_id.

```
15      -- ---------------------------------------------------------
16      -- Consulta para visualizar las ordenes de compras
17      -- generadas por un Empleado_logistico, ingresando su id
18      -- ---------------------------------------------------------
19
20      select * from orden_compra
21      where empleado_logistico_id = "E1";
22
```

Figura 5: consulta 3

Consulta 4: consulta que permite visualizar el proveedor de cada alimento.

Para obtener esta información es necesario realizar 4 inner joins; el primero entre las tablas proveedor y orden_compra (relacionando el id del proveedor con la llave foránea de la tabla orden_compra.proveedor_id), el segundo integra al resultado con la tabla orden_alimento (relacionando el id de la tabla orden_compra con la llave foránea de la tabla orden_alimento.orden_compra_id), y por último, se integra la tabla alimento (relacionando el id de la tabla alimento con la llave foránea orden_alimento.alimento_id). En la tabla resultante se pueden observar los valores de los atributos proveedor.nombre y alimento.nombre.

```
23      -- ---------------------------------------------------------
24      -- Consulta para visualizar el proveedor de cada alimento
25      -- ---------------------------------------------------------
26
27      select proveedor.nombre as proveedor, alimento.nombre as alimento
28      from proveedor
29      inner join orden_compra on proveedor.id = orden_compra.proveedor_id
30      inner join orden_alimento on orden_compra.id = orden_alimento.orden_compra_id
31      inner join alimento on orden_alimento.alimento_id = alimento.id;
```

Figura 6: consulta 4

Consulta 5: consulta que permite visualizar los alimentos contenidos en una dieta. Para obtener esta información es necesario llevar a cabo 2 inner join; el primero entre las tablas alimento y alimento_dieta (relacionando alimento.id y alimento_dieta.alimento_id), y el resultado se integra con la tabla dieta (relacionando alimento_dieta.dieta_id con dieta.id). Teniendo en cuenta que una dieta puede contener múltiples alimentos, se utiliza

group_concat para concatenar los alimentos que conforman una dieta y de esta manera presentarlos en una única fila.

```
33      -- ---------------------------------------------------------------
34      -- Consulta para visualizar los alimentos contenidos en una Dieta
35      -- ---------------------------------------------------------------
36
37  •   select distinct dieta.nombre as dieta, group_concat(distinct alimento.nombre separator ', ') as alimento
38      from alimento
39      inner join alimento_dieta on alimento.id = alimento_dieta.alimento_id
40      inner join dieta on alimento_dieta.dieta_id = dieta.id
41      group by dieta.id;
```

Figura 7: consulta 5

Consulta 6: Consulta que permite visualizar las facturas enviadas por los proveedores. Para obtener esta información es necesario implementar 4 inner joins. El primero entre las tablas alimento y orden alimento (relacionando alimento.id con la llave foránea orden_alimento.alimento_id), despues se agrega la tabla orden_compra (relacionando orden_alimento.orden_compra_id con orden_compra.id), luego se une la tabla proveedor (relacionando orden_compra.proveedor_id con proveedor.id), y por ultimo se agrega la tabla factura (relacionando proveedor.id con factura.id_proveedor). En esta consulta se visualizan en una tabla la fecha, el nombre del alimento, las unidades facturadas y el nombre del proveedor. En la tabla resultante se presentan los valores de los atributos fatura.fecha, alimento.nombre, factura.unidades y proveedor.nombre.

```
43      -- ---------------------------------------------------------------
44      -- Consulta para visualizar las facturas enviadas por los proveedores
45      -- ---------------------------------------------------------------
46
47  •   select factura.fecha, alimento.nombre as producto, factura.unidades, proveedor.nombre proveedor
48      from alimento
49      inner join orden_alimento on alimento.id = orden_alimento.alimento_id
50      inner join orden_compra on orden_alimento.orden_compra_id = orden_compra.id
51      inner join proveedor on orden_compra.proveedor_id = proveedor.id
52      inner join factura on proveedor.id = factura.id_proveedor;
```

Figura 8: consulta 6

Consulta 7: consulta para agregar el nombre del veterinario que formula la dieta a una especie (es decir se realiza un inner join adicional a la consulta 5., relacionando dieta.veterinario_id con veterinario.id). En la tabla resultante

se presentan los valores de los atributos dieta.nombre, y el nombre concatenado de los alimentos asociados a esta.

```
54
55  select distinct dieta.nombre as dieta, group_concat(distinct alimento.nombre separator ', ') as alimento,
56  veterinario.nombre as veterinario
57  from alimento
58  inner join alimento_dieta on alimento.id = alimento_dieta.alimento_id
59  inner join dieta on alimento_dieta.dieta_id = dieta.id
60  inner join veterinario on dieta.veterinario_id = veterinario.id
61  group by dieta.id;
```

Figura 9: consulta 7

Consulta 8: consulta para visualizar los reportes revisados por un veterinario. Se pretende presentar el nombre del animal, su peso (presente en el reporte), el id del animal, asociado al veterinario que realiza la consulta. Para obtener esta información es necesario realizar 3 inner joins; el primero entre veterinario y veterinario_reporte (asociando veterinario.id con veterinario_reporte.reporte_id; el segundo integra la tabla reporte (asociando veterinario_reporte.reporte_id con reporte.id; y por último, se integra la tabla animal (asociando reporte.animal_id con animal.id). En la tabla resultante se presentan los valores de los atributos veterinario.nombre, animal.especie, animal.id, reporte.peso_animal.

```
68  select veterinario.nombre as veterinario, animal.especie as especie, animal.id as id_animal, reporte.peso_animal
69  from veterinario
70  inner join veterinario_reporte on veterinario.id = veterinario_reporte.veterinario_id
71  inner join reporte on veterinario_reporte.reporte_id = reporte.id
72  inner join animal on reporte.animal_id = animal.id;
```

Figura 10: consulta 8

Consulta 9: consulta que permite relacionar el peso de un animal con los alimentos que consume. Para obtener esta iformacion, son necesarios 4 inner join. El primero entre reporte y animal (relacionando reporte.animal_id con animal.id), luego se une la tabla dieta (relacionando animal.dieta_id con dieta.id), después se une la tabla alimento_dieta (relacionando dieta.id con alimento_dieta.dieta_id). La tabla resultante me presenta las columnas animal.id, reporte.peso_animal y animal.especie. Cada alimento se concatena para ser presentado en una única fila

```sql
select animal.id as animal_id, reporte.peso_animal, animal.especie as especie,
group_concat(distinct alimento.nombre separator ', ') as alimento
from reporte
inner join animal on reporte.animal_id = animal.id
inner join dieta on animal.dieta_id = dieta.id
inner join alimento_dieta on dieta.id = alimento_dieta.dieta_id
inner join alimento on alimento_dieta.alimento_id = alimento.id
group by animal.id, reporte.peso_animal;
```

Figura 11: consulta 9

Consulta 10: para esta consulta se parte de la consulta 6 (figura 6), la cual permite visualizar todas las facturas enviadas por los proveedores. Lo que se busca en esta ultima consulta es poder observar las facturas enviadas en el año 2023. Para esto se agrega la sentencia where factura.fecha like '%23', es decir, se pueden ver en la tabla resultante los valores cuyo valor de la columna factura.fecha terminen en 23 (tomando como referencia un formato de fecha dd/mm/aaaa)

```sql
select factura.fecha, alimento.nombre as producto, factura.unidades, proveedor.nombre proveedor
from alimento
inner join orden_alimento on alimento.id = orden_alimento.alimento_id
inner join orden_compra on orden_alimento.orden_compra_id = orden_compra.id
inner join proveedor on orden_compra.proveedor_id = proveedor.id
inner join factura on proveedor.id = factura.id_proveedor
where factura.fecha like '%23';
```

Figura 12: consulta 10

6. Tomando como base las consultas implementadas en el numeral 5, se elijen 4 de ellas para crear vistas que aporten información valiosa en el contexto del zoológico. Las sentencias utilizadas para generar cada vista pueden encontrarse en **scripts/script-views**

   - **Vista 1 (proveedor_alimento_cantidad):** en esta vista se pretende observar la cantidad de alimento disponible asociado al proveedor; información que es valiosa para, por ejemplo, el empleado logístico para generar las ordenes de compra a los proveedores. Ya que es una consulta que es importante y que debe realizarse periódicamente, se decide crear la vista presentada en la figura 13.

```sql
-- ------------------------------------------------------------
-- vista para visualizar la cantidad de alimento disponible
-- asociado a su proveedor
-- ------------------------------------------------------------
create view proveedor_alimento_cantidad as
select proveedor.nombre as proveedor, alimento.nombre as alimento,
alimento.cantidad as cantidad
from proveedor
inner join orden_compra on proveedor.id = orden_compra.proveedor_id
inner join orden_alimento on orden_compra.id = orden_alimento.orden_compra_id
inner join alimento on orden_alimento.alimento_id = alimento.id;
```

Figura 13: script para generar la vista cantidad_alimento

- **Vista 2 (peso_animal_alimentos):** con esta vista se pretende relacionar el peso reportado de cada animal con los alimentos asociados a la dieta que esta consumiendo. Esta es una información valiosa en el contexto del zoológico para el veterinario, ya que él se encarga de ajustar la dieta de cada especie, dependiendo del reporte de su peso. El script para generar la vista se presenta en la figura 13.

```sql
-- ------------------------------------------------------------------------
-- vista donde se relaciona el peso reportado de animal con los alimentos
-- que consume, segun su dieta asignada
-- ------------------------------------------------------------------------
create view peso_animal_alimentos as
select animal.id as animal_id, reporte.peso_animal, animal.especie as especie,
group_concat(distinct alimento.nombre separator ', ') as alimento,
dieta.dosis as dosis
from reporte
inner join animal on reporte.animal_id = animal.id
inner join dieta on animal.dieta_id = dieta.id
inner join alimento_dieta on dieta.id = alimento_dieta.dieta_id
inner join alimento on alimento_dieta.alimento_id = alimento.id
group by animal.id, reporte.peso_animal;
```

Figura 14: script para generar la vista peso_animal_alimentos

- **Vista 3 (facturas_año):** con esta vista se pretende consultar las facturas recibidas en lo corrido del año 2023. Esta es una consulta que es de interés,

por ejemplo, para el empleado logístico a la hora de totalizar la cantidad de alimentos que están consumiendo los animales del zoológico.

```
25    -- ---------------------------------------------------------------------
26    -- vista para visualizar las facturas recibidas en lo corrido del 2023
27    -- ---------------------------------------------------------------------
28
29    create view facturas_año as
30    select factura.fecha, alimento.nombre as producto, factura.unidades, proveedor.nombre proveedor
31    from alimento
32    inner join orden_alimento on alimento.id = orden_alimento.alimento_id
33    inner join orden_compra on orden_alimento.orden_compra_id = orden_compra.id
34    inner join proveedor on orden_compra.proveedor_id = proveedor.id
35    inner join factura on proveedor.id = factura.id_proveedor
36    where factura.fecha like '%23';
```

Figura 15: script para generar vista factura_año

- **Vista 4 (orden_compra_view):** en este caso se pretende tener una vista que me permita obtener toda la información de las ordenes de servicio. Ya que estas son generadas por el empleado logístico, esta es una información importante que permite llevar control sobre los alimentos que se están ordenando para el consumo de los animales.

```
44    -- ---------------------------------------------------------------------
45    -- vista para visualizar las ordenes de compra generadas
46    -- ---------------------------------------------------------------------
47
48    create view orden_compra_view as
49    select orden_compra.fecha as fecha_orden, alimento.nombre as producto,
50    orden_compra.unidades as unidades_solicitadas
51    from alimento
52    inner join orden_alimento on alimento.id = orden_alimento.alimento_id
53    inner join orden_compra on orden_alimento.orden_compra_id = orden_compra.id;
```

Figura 16: script para generar la vista orden_compra_view

7. Ahora se procede a generar 4 procedimientos almacenados que permitan ejecutar acciones sobre las tablas de la base de datos. Las sentencias para generar los procedimientos se pueden encontrar en **scripts/script-procedimientos**

- **Procedimiento 1 (agregar_alimento):** Este procedimiento almacenado permite agregar registros en la tabla Alimento.

```
3   -- ---------------------------------------------
4   -- procedimiento para insertar un alimento
5   -- ---------------------------------------------
6
7   DELIMITER //
8   create procedure agregar_alimento(in id_alimento varchar(10),
9   nombre_alimento varchar(45),
10  tipo_alimento varchar(45),
11  cantidad_alimento varchar(6))
12  begin
13      insert into alimento values (id_alimento, nombre_alimento, tipo_alimento, cantidad_alimento);
14  end //
15  DELIMITER ;
```

Figura 17: procedimiento agregar_alimento

- **Procedimiento 2 (eliminar_alimento):** este procedimiento se implementa para eliminar un registro en la tabla Alimento. Este procedimiento puede ser usado cuando se elimina por completo este alimento de la dieta de todas las especies del zoológico.

```
22  DELIMITER //
23  create procedure eliminar_alimento(in id_alimento varchar(10))
24  begin
25      delete from alimento where alimento.id = id_alimento;
26  end //
27  DELIMITER ;
```

Figura 18: procedimiento eliminar_alimento

- **Procedimiento 3 (actualizar_alimento):** este procedimiento se implementa para actualizar, por ejemplo, la cantidad de alimento disponible, luego de verificar el stock.

```
33      DELIMITER //
34      create procedure actualizar_alimento(in id_alimento varchar(10),
35      nombre_alimento varchar(45),
36      tipo_alimento varchar(45),
37      cantidad_alimento varchar(6))
38      begin
39          update alimento
40          set alimento.nombre = nombre_alimento, alimento.tipo = tipo_alimento,
41              alimento.cantidad = cantidad_alimento
42          where alimento.id = id_alimento;
43      end //
44      DELIMITER ;
```

Figura 19: procedimiento actualizar_alimento

- **Procedimiento 4 (consultar_cantidad_alimento):** Este procedimiento se implementa para consultar la cantidad de un alimento disponible.

```
46      -- -----------------------------------------------------------------
47      -- procedimiento para consultar la cantidad de alimento disponible
48      -- -----------------------------------------------------------------
49
50      DELIMITER //
51      create procedure consultar_cantidad_alimento(in id_alimento varchar(10))
52      begin
53          select alimento.nombre, alimento.cantidad from alimento
54          where alimento.id = id_alimento;
55      end //
56      DELIMITER ;
```

Figura 20: procedimiento consultar_cantidad_alimento

8. Se procede a generar 4 triggers. Lo primero que se hace es crear la tabla control_cambios_zoologico, la cual contiene 3 columnas: usuario, acción y fecha. Para crear la tabla control_cambios_zoologico se ejecuta la sentencia de la figura 21.

```
3     -- -----------------------------------------------------
4     -- sentencia para crear la tabla control_cambios_zoologico
5     -- -----------------------------------------------------
6
7     create table control_cambios_zoologico (
8         usuario varchar(45) not null,
9         accion varchar(45) not null,
10        fecha datetime default current_timestamp
11        );
12
```

Figura 21: sentencia para crear tabla control_cambios_zoologico

- **Trigger 1 (insert_libro):** en la figura 22 se presenta el script con el cual se crea el trigger insert_libro, el cual busca que se dispare cuando un usuario agregue un registro en la tabla Alimento.

```
13    -- ---------------------------------------------------------------------
14    -- sentencia para crear un trigger que se active cuando un usuario
15    -- agregue un registro en la tabla alimento
16    -- ---------------------------------------------------------------------
17
18    DELIMITER //
19    create trigger insert_alimento after insert on alimento
20    for each row
21    begin
22        insert into control_cambios_zoologico
23        values (user(), "Agregó alimento", now());
24    end //
25    DELIMITER ;
26
```

Figura 22: script para generar el trigger insert_libro

- **Trigger 2 (update_alimento):** este trigger se implementa para guardar un registro en la tabla control_cambios_zoologico cuando un usuario actualiza un registro en la tabla alimento.

```
32    DELIMITER //
33    create trigger update_alimento after update on alimento
34    for each row
35    begin
36        insert into control_cambios_zoologico
37        values (user(), "Actualizó alimento", now());
38    end //
39    DELIMITER ;
```

Figura 23: script para generar trigger update_alimento

- **Trigger 3 (delete_alimento**): trigger que se implementa para guardar un registro en la tabla control_cambios_zoologico cuando un usuario borra un registro en la tabla alimento.

```
46    DELIMITER //
47    create trigger delete_alimento after delete on alimento
48    for each row
49    begin
50        insert into control_cambios_zoologico
51        values (user(), "Eliminó alimento", now());
52    end //
53    DELIMITER ;
```

Figura 24: script para generar trigger delete_alimento

- **Trigger 4 (insert_reporte):** trigger que se implementa para guardar un registro en la tabla control_cambios_zoologico cuando un usuario inserta un nuevo reporte en la tabla reporte.

```
60      DELIMITER //
61      create trigger insert_reporte after insert on reporte
62      for each row
63      begin
64          insert into control_cambio_zoologico
65          values (user(), "Agregó reporte", now());
66      end //
67      DELIMITER ;
```

Figura 25: script para generar trigger insert_reporte

9. Ahora se procede a poblar cada una de las tablas de la base de datos definida, utilizando Java. Para esto se utiliza como gestor de dependencias Grade. Las dependencias utilizadas son javafaker y el conector Mysql, el cual permite conectarse a la base de datos desde Java. se presenta el archivo build.gradle en la figura 26.

```
plugins {
    id 'java'
}

group 'com.sofkau'
version '1.0-SNAPSHOT'

repositories {
    mavenCentral()
}

dependencies {
    testImplementation 'org.junit.jupiter:junit-jupiter-api:5.8.1'
    testRuntimeOnly 'org.junit.jupiter:junit-jupiter-engine:5.8.1'
    // https://mvnrepository.com/artifact/com.mysql/mysql-connector-j
    implementation 'com.mysql:mysql-connector-j:8.0.32'
    // https://mvnrepository.com/artifact/com.github.javafaker/javafaker
    implementation 'com.github.javafaker:javafaker:1.0.2'

}

test {
    useJUnitPlatform()
}
```

Figura 26: Dependencias utilizadas para poblar las tablas de la base de datos

Estructura carpetas:

Main/java/com/sofkau/integration/database/mysql

Main/java/com/sofkau/integration/database/models

Main/java/com/sofkau/main

- En el paquete mysql:
  - se encuentran la clase MysqlConstant, donde se definen la url para la conexión a la base de datos y se especifica el Driver para realizar dicha conexión.
  - También se tiene la clase MySqlOperation. En esta clase se implementan los métodos definidos en la interfaz IDatabase. Dichos métodos se encargan de abrir, cerrar la conexión con la base de datos, definir los statements que posteriormente son ejecutados.
- Paquete models:
  - En este paquete se encuentran todas las clases que serán mapeadas con las tablas de la base de datos. Es decir, son clases que toman los mismos atributos que las columnas de las tablas de la base de datos. En estas clases se definen los métodos get y set para cada atributo y el constructor para instanciarlas.
- Paquete sofkau:
  - Aquí se puede encontrar el método main. Allí se instancian los modelos y se implementan los métodos que permiten poblar las tablas de la base de datos.

Por ultimo se presentan las imágenes de cada una de las tablas de la base de datos.

	id	nombre	tipo	cantidad
▶	A1	Zanahoria	fruta	300
	A2	cuido	seco	100
	A4	lechuga	legumbre	20
	Ali 0	Dragonfruit	quibusdam	055
	Ali 1	Tomatoes	molestiae	009
	Ali 10	Nori	nemo	702
	Ali 11	Miso	ullam	712
	Ali 12	Mountain Bread	pariatur	998
	Ali 13	Nuts	sunt	483
	Ali 14	Curry Leaves	sunt	019
	Ali 15	Nashi Pear	blanditiis	305
	Ali 16	Molasses	aliquid	466
	Ali 17	Asparagus	recusandae	074
	Ali 18	Cumquat	sed	272
	Ali 19	Mustard Seed	adipisci	643
	Ali 2	Szechuan Pepp...	dignissimos	417
	Ali 20	Cardamom	voluptatibus	773
	Ali 21	Borlotti Beans	aut	952
	Ali 22	Berries	magni	981
	Ali 23	Ricotta	quis	516
	Ali 24	Dill	id	056
	Ali 25	Parmesan Cheese	enim	737
	Ali 26	Tabasco	omnis	959
	Ali 27	Asian Noodles	aliquid	945
	Ali 28	Pine Nut	minima	002
	Ali 29	White Wine	ab	441

Figura 27: Tabla Alimento

alimento_id	dieta_id
A1	D1
A2	D1
Ali 0	Dieta-0
Alimento 0	Dieta-0
Ali 1	Dieta-1
Alimento 1	Dieta-1
Ali 10	Dieta-10
Ali 11	Dieta-11
Ali 12	Dieta-12
Ali 13	Dieta-13
Ali 14	Dieta-14
Ali 15	Dieta-15
Ali 16	Dieta-16
Ali 17	Dieta-17
Ali 18	Dieta-18
Ali 19	Dieta-19
Ali 2	Dieta-2
Alimento 2	Dieta-2
Ali 20	Dieta-20
Ali 21	Dieta-21
Ali 22	Dieta-22
Ali 23	Dieta-23
Ali 24	Dieta-24
Ali 25	Dieta-25
Ali 26	Dieta-26
Ali 27	Dieta-27