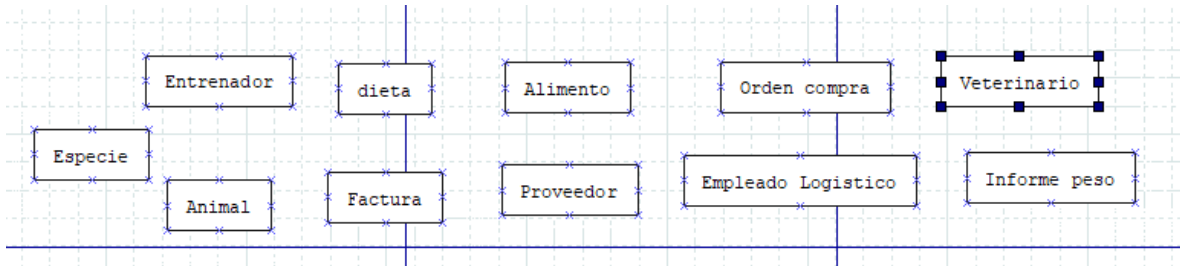


Reto C1-2023-QA-BD-RF

Zoo Santafé

En primer lugar, se crea el diagrama E-R. Partimos creando las entidades:



- Entrenador
- Informe peso
- Animal
- Especie
- Veterinario
- Dieta
- Factura
- Alimento
- Empleado logístico
- Orden compra
- Proveedor

Luego, se empiezan a crear las relaciones y sus respectivas cardinalidades:

Un animal solo tiene una especie, pero una especie puede tener muchos animales.

Un animal solo puede consumir solo un tipo de dieta, pero una dieta puede ser consumida por muchos animales.

Un informe solo puede ser generado por un entrenador, pero un entrenador puede generar muchos informes.

Un informe solo puede ser consultado por un veterinario, pero un veterinario puede consultar uno o muchos informes.

Un veterinario puede diseñar muchas dietas, y muchas dietas pueden ser diseñadas por uno o muchos veterinarios.

Un informe puede registrar solo un animal, pero un animal puede tener muchos registros.

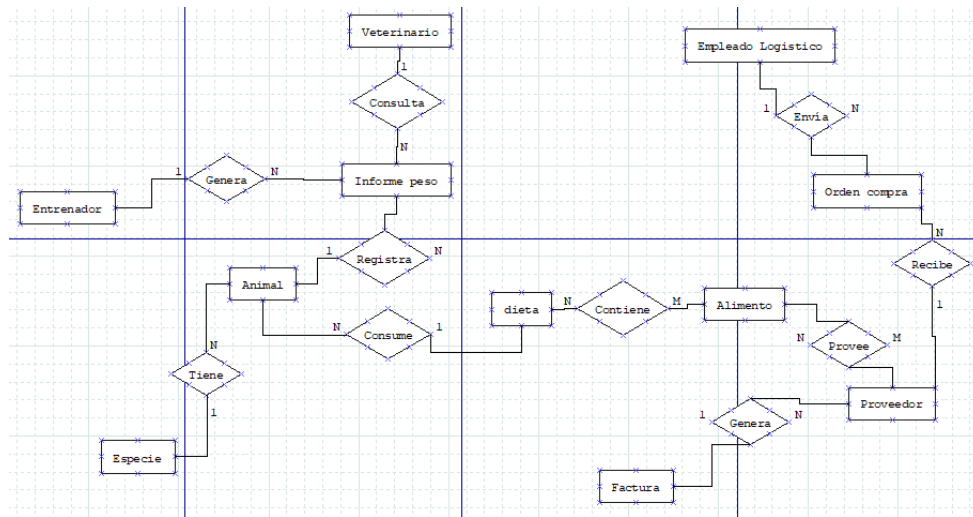
Una factura puede ser generada por solo un proveedor, pero un proveedor puede generar muchas facturas.

Un proveedor puede proveer muchos alimentos, y muchos alimentos pueden ser proveídos por uno o más proveedores.

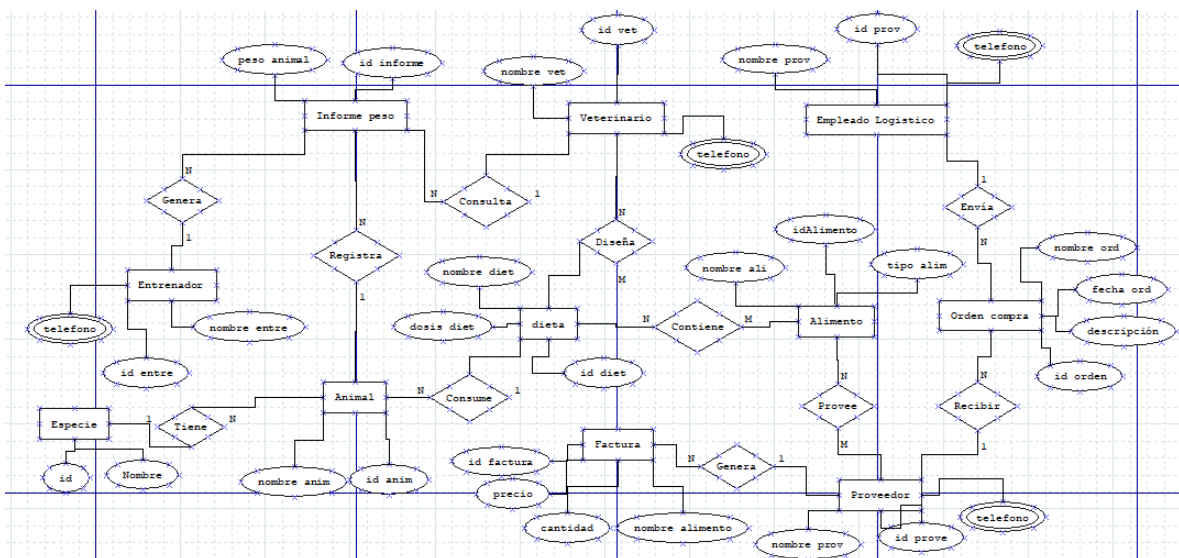
Una orden de compra puede solo ser recibida por un proveedor, pero un proveedor puede recibir muchas órdenes de compra.

Una orden de compra solo puede ser enviada por un logístico, pero un logístico puede enviar muchas órdenes de compra.

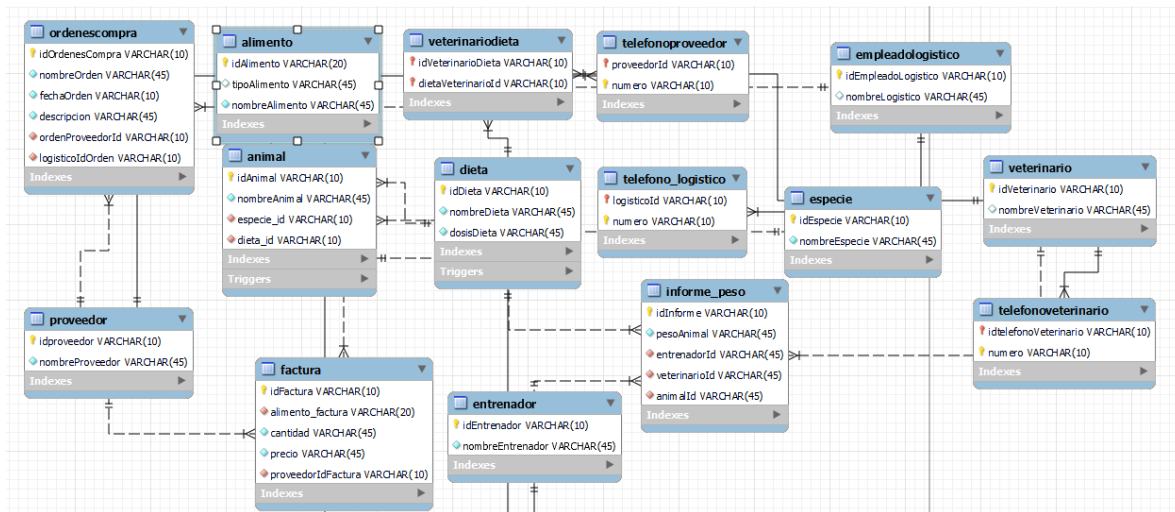
Una dieta puede contener muchos alimentos, y un alimento puede estar contenido en muchas dietas.



Finalmente procedemos a añadir atributos a cada entidad: Obsérvese que las entidades que representan a personas tienen un atributo multivaluado el cual es teléfono



Después de realizar el diagrama E-R, se procedió a crear el modelo relacional en Workbench. El modelo recrea las anteriores entidades y las convierte en tablas, las relaciones se convierten en tablas relacionales y las tablas que contienen un valor multivaluado se conectan con una tabla que representa dicho valor (teléfono).



Después aplicamos ingeniería hacia adelante y creamos la base de datos zoologico con las siguientes tablas:

- Zoologico.animal

```
CREATE TABLE IF NOT EXISTS `zoologico`.`Animal` (
  `idAnimal` VARCHAR(10) NOT NULL,
  `nombreAnimal` VARCHAR(45) NOT NULL,
  `especie_id` VARCHAR(10) NOT NULL,
  `dieta_id` VARCHAR(10) NOT NULL,
  PRIMARY KEY (`idAnimal`),
  INDEX `especie_id_idx` (`especie_id` ASC) VISIBLE,
  CONSTRAINT `especie_id`
    FOREIGN KEY (`especie_id`)
      REFERENCES `zoologico`.`especie` (`idEspecie`),
  INDEX `dieta_id_idx` (`dieta_id` ASC) VISIBLE,
  CONSTRAINT `dieta_id`
    FOREIGN KEY (`dieta_id`)
      REFERENCES `zoologico`.`Dieta` (`idDieta`)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

- Zoologico.dieta

```
• CREATE TABLE IF NOT EXISTS `zoologico`.`Dieta` (  
  `idDieta` VARCHAR(10) NOT NULL,  
  `nombreDieta` VARCHAR(45) NOT NULL,  
  `dosisDieta` VARCHAR(45) NOT NULL,  
  PRIMARY KEY (`idDieta`))  
ENGINE = InnoDB;
```

- Zoologico.entrenador y teléfono entrenador

```
• CREATE TABLE IF NOT EXISTS `zoologico`.`Entrenador` (  
  `idEntrenador` VARCHAR(10) NOT NULL,  
  `nombreEntrenador` VARCHAR(45) NOT NULL,  
  PRIMARY KEY (`idEntrenador`))  
ENGINE = InnoDB;
```

```
--  
-- Table `zoologico`.`telefonoEntrenador`  
--
```

```
• CREATE TABLE IF NOT EXISTS `zoologico`.`telefonoEntrenador` (  
  `idtelefonoEntrenador` VARCHAR(10) NOT NULL,  
  `numero` VARCHAR(10) NOT NULL,  
  PRIMARY KEY (`idtelefonoEntrenador`, `numero`),  
  CONSTRAINT `idtelefonoEntrenador`  
    FOREIGN KEY (`idtelefonoEntrenador`)  
    REFERENCES `zoologico`.`Entrenador` (`idEntrenador`)  
    ON DELETE NO ACTION  
    ON UPDATE NO ACTION)  
ENGINE = InnoDB;
```

- Zoologico.especie

```
• CREATE TABLE IF NOT EXISTS `zoologico`.`especie` (  
  `idEspecie` VARCHAR(10) NOT NULL,  
  `nombreEspecie` VARCHAR(45) NOT NULL,  
  PRIMARY KEY (`idEspecie`))  
ENGINE = InnoDB;
```

- Zoologico.veterinario y teléfono veterinario

```
-----  
CREATE TABLE IF NOT EXISTS `zoologico`.`Veterinario` (  
  `idVeterinario` VARCHAR(10) NOT NULL,  
  `nombreVeterinario` VARCHAR(45) NULL,  
  PRIMARY KEY (`idVeterinario`))  
ENGINE = InnoDB;  
  
-----  
-- Table `zoologico`.`telefonoVeterinario`  
-----  
CREATE TABLE IF NOT EXISTS `zoologico`.`telefonoVeterinario` (  
  `idtelefonoVeterinario` VARCHAR(10) NOT NULL,  
  `numero` VARCHAR(10) NOT NULL,  
  PRIMARY KEY (`idtelefonoVeterinario`, `numero`),  
  CONSTRAINT `idtelefonoVeterinario`  
    FOREIGN KEY (`idtelefonoVeterinario`)  
      REFERENCES `zoologico`.`Veterinario` (`idVeterinario`)  
    ON DELETE NO ACTION  
    ON UPDATE NO ACTION)  
ENGINE = InnoDB;
```

- Zoologico.veterinarioDieta

```
CREATE TABLE IF NOT EXISTS `zoologico`.`VeterinarioDieta` (  
  `idVeterinarioDieta` VARCHAR(10) NOT NULL,  
  `dietaVeterinarioId` VARCHAR(10) NOT NULL,  
  PRIMARY KEY (`idVeterinarioDieta`, `dietaVeterinarioId`),  
  INDEX `dietaVeterinarioId_idx` (`dietaVeterinarioId` ASC) VISIBLE,  
  CONSTRAINT `idVeterinarioDieta`  
    FOREIGN KEY (`idVeterinarioDieta`)  
      REFERENCES `zoologico`.`Veterinario` (`idVeterinario`)  
    ON DELETE NO ACTION  
    ON UPDATE NO ACTION,  
  CONSTRAINT `dietaVeterinarioId`  
    FOREIGN KEY (`dietaVeterinarioId`)  
      REFERENCES `zoologico`.`Dieta` (`idDieta`)  
    ON DELETE NO ACTION  
    ON UPDATE NO ACTION)  
ENGINE = InnoDB;
```

- Zoologico.informe_peso

```
CREATE TABLE IF NOT EXISTS `zoologico`.`Informe_peso` (  
  `idInforme` VARCHAR(10) NOT NULL,  
  `pesoAnimal` VARCHAR(45) NOT NULL,  
  `entrenadorId` VARCHAR(45) NOT NULL,  
  `veterinarioId` VARCHAR(45) NOT NULL,  
  `animalId` VARCHAR(45) NOT NULL,  
  PRIMARY KEY (`idInforme`),  
  INDEX `entrenadorId_idx` (`entrenadorId` ASC) VISIBLE,  
  INDEX `veterinarioId_idx` (`veterinarioId` ASC) VISIBLE,  
  INDEX `animalId_idx` (`animalId` ASC) VISIBLE,  
  CONSTRAINT `entrenadorId`  
    FOREIGN KEY (`entrenadorId`)  
    REFERENCES `zoologico`.`Entrenador` (`idEntrenador`)  
    ON DELETE NO ACTION  
    ON UPDATE NO ACTION,  
  CONSTRAINT `veterinarioId`  
    FOREIGN KEY (`veterinarioId`)  
    REFERENCES `zoologico`.`Veterinario` (`idVeterinario`)  
    ON DELETE NO ACTION  
    ON UPDATE NO ACTION,  
  CONSTRAINT `animalId`  
    FOREIGN KEY (`animalId`)  
    REFERENCES `zoologico`.`Animal` (`idAnimal`)  
    ON DELETE NO ACTION  
    ON UPDATE NO ACTION)  
ENGINE = InnoDB;
```

- Zoologico.Alimento

```
CREATE TABLE IF NOT EXISTS `zoologico`.`Alimento` (  
  `nombreAlimento` VARCHAR(20) NOT NULL,  
  `tipoAlimento` VARCHAR(45) NULL,  
  PRIMARY KEY (`nombreAlimento`))  
ENGINE = InnoDB;
```

- Zoologico.Alimento_dieta

```
CREATE TABLE IF NOT EXISTS `zoologico`.`Alimento_Dieta` (  
  `dietaAlimentoId` VARCHAR(10) NOT NULL,  
  `Alimento_Dieta_Nombre` VARCHAR(20) NOT NULL,  
  PRIMARY KEY (`dietaAlimentoId`, `Alimento_Dieta_Nombre`),  
  INDEX `Alimento_Dieta_Nombre_idx` (`Alimento_Dieta_Nombre` ASC) VISIBLE,  
  CONSTRAINT `dietaAlimentoId`  
    FOREIGN KEY (`dietaAlimentoId`)  
    REFERENCES `zoologico`.`Dieta` (`idDieta`)  
    ON DELETE NO ACTION  
    ON UPDATE NO ACTION,  
  CONSTRAINT `Alimento_Dieta_Nombre`  
    FOREIGN KEY (`Alimento_Dieta_Nombre`)  
    REFERENCES `zoologico`.`Alimento` (`nombreAlimento`)  
    ON DELETE NO ACTION  
    ON UPDATE NO ACTION)  
ENGINE = InnoDB;
```

- Zoologico.empleadoLogistico y telefonoLogistico

```
> CREATE TABLE IF NOT EXISTS `zoologico`.`EmpleadoLogistico` (  
  `idEmpleadoLogistico` VARCHAR(10) NOT NULL,  
  `nombreLogistico` VARCHAR(45) NULL,  
  PRIMARY KEY (`idEmpleadoLogistico`))  
ENGINE = InnoDB;  
  
-----  
-- Table `zoologico`.`telefono_logistico`  
-----  
> CREATE TABLE IF NOT EXISTS `zoologico`.`telefono_logistico` (  
  `logisticoId` VARCHAR(10) NOT NULL,  
  `numero` VARCHAR(10) NOT NULL,  
  PRIMARY KEY (`logisticoId`, `numero`),  
  CONSTRAINT `logisticoId`  
    FOREIGN KEY (`logisticoId`)  
      REFERENCES `zoologico`.`EmpleadoLogistico` (`idEmpleadoLogistico`)  
    ON DELETE NO ACTION  
    ON UPDATE NO ACTION)  
ENGINE = InnoDB;
```

- Zoologico.proveedor y telefonoProveedor

```
-----  
CREATE TABLE IF NOT EXISTS `zoologico`.`Proveedor` (  
  `idproveedor` VARCHAR(10) NOT NULL,  
  `nombreProveedor` VARCHAR(45) NOT NULL,  
  PRIMARY KEY (`idproveedor`))  
ENGINE = InnoDB;  
  
-----  
-- Table `zoologico`.`telefonoProveedor`  
-----  
CREATE TABLE IF NOT EXISTS `zoologico`.`telefonoProveedor` (  
  `proveedorId` VARCHAR(10) NOT NULL,  
  `numero` VARCHAR(10) NOT NULL,  
  PRIMARY KEY (`proveedorId`, `numero`),  
  CONSTRAINT `proveedorId`  
    FOREIGN KEY (`proveedorId`)  
      REFERENCES `zoologico`.`Proveedor` (`idproveedor`)  
    ON DELETE NO ACTION  
    ON UPDATE NO ACTION)  
ENGINE = InnoDB;
```

- Zoologico.factura

```

) CREATE TABLE IF NOT EXISTS `zoologico`.`Factura` (
  `idFactura` VARCHAR(10) NOT NULL,
  `id_alimento_factura` VARCHAR(10) NOT NULL,
  `cantidad` VARCHAR(45) NOT NULL,
  `precio` VARCHAR(45) NOT NULL,
  `proveedorIdFactura` VARCHAR(10) NOT NULL,
  PRIMARY KEY (`idFactura`),
  INDEX `proveedorIdFactura_idx` (`proveedorIdFactura` ASC) VISIBLE,
  CONSTRAINT `proveedorIdFactura`
    FOREIGN KEY (`proveedorIdFactura`)
    REFERENCES `zoologico`.`Proveedor` (`idproveedor`),
  INDEX `id_alimento_factura_idx` (`id_alimento_factura` ASC) VISIBLE,
  CONSTRAINT `id_alimento_factura`
    FOREIGN KEY (`id_alimento_factura`)
    REFERENCES `zoologico`.`alimento` (`idAlimento`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

```

- Zoologico ordenes compra

```

) CREATE TABLE IF NOT EXISTS `zoologico`.`OrdenesCompra` (
  `idOrdenesCompra` VARCHAR(10) NOT NULL,
  `nombreOrden` VARCHAR(45) NOT NULL,
  `fechaOrden` VARCHAR(10) NOT NULL,
  `descripcion` VARCHAR(45) NOT NULL,
  `ordenProveedorId` VARCHAR(10) NOT NULL,
  `logisticoIdOrden` VARCHAR(10) NOT NULL,
  PRIMARY KEY (`idOrdenesCompra`),
  INDEX `ordenProveedorId_idx` (`ordenProveedorId` ASC) VISIBLE,
  INDEX `logisticoIdOrden_idx` (`logisticoIdOrden` ASC) VISIBLE,
  CONSTRAINT `ordenProveedorId`
    FOREIGN KEY (`ordenProveedorId`)
    REFERENCES `zoologico`.`Proveedor` (`idproveedor`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `logisticoIdOrden`
    FOREIGN KEY (`logisticoIdOrden`)
    REFERENCES `zoologico`.`EmpleadoLogistico` (`idEmpleadoLogistico`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

```


Después de crear las tablas se crearon las siguientes consultas

- Nombre de las dietas

```
1 #1 Seleccionar los nombres de las dietas de los animales que están en el zoo:
2 • use zoologico;
3 • SELECT d.nombreDieta
4 FROM Dieta d
5 INNER JOIN Animal a ON d.idDieta = a.dieta_id;
```

nombreDieta
Josef Bogan
Ms. Tyron Leuschke
Kelly Luetngen IV
Delta Dibbert
Otto Schowalter II
Mr. Marisol Grimes
Hyon Jast

- Nombre de animales con su especie

```
1 #2 Seleccionar los nombres de los animales y los nombres de sus especies:
2 • SELECT a.nombreAnimal, e.nombreEspecie
3 FROM Animal a
4 INNER JOIN especie e ON a.especie_id = e.idEspecie;
```

nombreAnimal	nombreEspecie
crow	Umbra
alligator	Melusine
spider	Gargoyle
toad	Ghoul
reindeer	Lubberkin
lizard	Shrieker
macaw	Grave Hag
mallard	Fugas
squirrel	Arachas

- Seleccionar los nombres de las dietas que no tienen ningún animal que la esté siguiendo:

```
#3 Seleccionar los nombres de las dietas que no tienen ningún animal que la esté siguiendo:
• SELECT d.nombreDieta
FROM Dieta d
LEFT JOIN Animal a ON d.idDieta = a.dieta_id
WHERE a.idAnimal IS NULL;
```

Al momento la consulta aparece vacía, puesto que todos los animales tienen una dieta. Puede servir esta consulta para verificar si algún animal no tiene seguimiento.

- Seleccionar los nombres de los veterinarios y los números de teléfono

```
1 • SELECT v.nombreVeterinario, tv.numero
2 FROM Veterinario v
3 INNER JOIN telefonoVeterinario tv ON v.idVeterinario = tv.idtelefonoVeterinario;
```

nombreVeterinario	numero
Torri	1-645-846-7392
Torri	1-757-290-4157
David	306-753-7768
David	816-981-7960
Wilford	461.710.6367
Wilford	558-117-6365
Mao	383.124.1172
Mao	645.058.0897

- Seleccionar los nombres de los animales y los nombres de sus dietas, incluyendo aquellos animales que no tengan dieta asignada:

```
1 • SELECT a.nombreAnimal, d.nombreDieta
2 FROM Animal a
3 LEFT JOIN Dieta d ON a.dieta_id = d.idDieta;
```

nombreAnimal	nombreDieta
crow	Josef Bogan
alligator	Ms. Tyron Leuschke
spider	Kelly Luetttgen IV
toad	Delta Dibbert
reindeer	Otto Schowalter II

- Seleccionar los nombres de las dietas y la cantidad de animales que la siguen, incluyendo aquellas dietas que no tienen ningún animal que la esté siguiendo

```

2 • SELECT d.nombreDieta, COUNT(a.idAnimal) AS cantidad
3 FROM Dieta d
4 LEFT JOIN Animal a ON d.idDieta = a.dieta_id
5 GROUP BY d.idDieta;

```

nombreDieta	cantidad
Josef Bogan	1
Ms. Tyron Leuschke	1
Kelly Luetggen IV	1
Delta Dibbert	1
Otto Schmeltzer II	1

- Seleccionar los nombres de las especies y la cantidad de animales que pertenecen a cada especie

```

1 • SELECT esp.nombreEspecie, COUNT(a.idAnimal) AS cantidad
2 FROM Animal a
3 INNER JOIN especie esp ON a.especie_id = esp.idEspecie
4 GROUP BY esp.idEspecie;
5

```

nombreEspecie	cantidad
Umbra	1
Melusine	1
Gargoyle	1

- Mostrar todos los teléfonos de los entrenadores

```

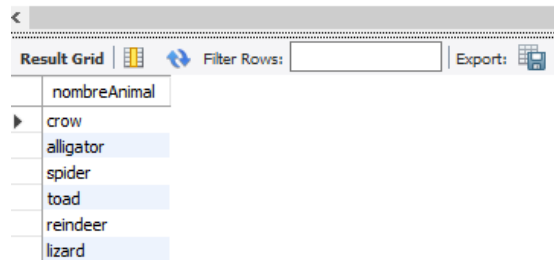
1 • SELECT Entrenador.nombreEntrenador, telefonoEntrenador.numero
2 FROM Entrenador
3 JOIN telefonoEntrenador ON Entrenador.idEntrenador = telefonoEntrenador.idtelefonoEntrenador;

```

nombreEntrenador	numero
Wan	379-110-1166
Wan	562-951-8525
Rosemary	1-656-504-1503
Rosemary	571-704-6825

- Muestra todos los animales

```
1 • SELECT nombreAnimal FROM Animal;
```

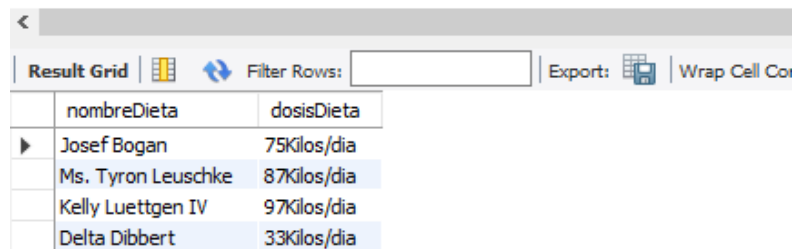


The screenshot shows a database interface with a 'Result Grid' tab. The grid has one column labeled 'nombreAnimal'. Below the header, there is a list of animal names: crow, alligator, spider, toad, reindeer, and lizard. The 'alligator' row is currently selected.

nombreAnimal
crow
alligator
spider
toad
reindeer
lizard

- Mostrar las dietas usadas últimamente

```
1 SELECT nombreDieta, dosisDieta FROM Dieta
2 JOIN Animal ON Dieta.idDieta = Animal.dieta_id;
```



The screenshot shows a database interface with a 'Result Grid' tab. The grid has two columns: 'nombreDieta' and 'dosisDieta'. Below the header, there is a list of diet entries: Josef Bogan (75Kilos/dia), Ms. Tyron Leuschke (87Kilos/dia), Kelly Luetngen IV (97Kilos/dia), and Delta Dibbert (33Kilos/dia). The 'Josef Bogan' row is currently selected.

nombreDieta	dosisDieta
Josef Bogan	75Kilos/dia
Ms. Tyron Leuschke	87Kilos/dia
Kelly Luetngen IV	97Kilos/dia
Delta Dibbert	33Kilos/dia

Después de realizar las consultas procedo a crear 4 triggers;

- Trigger 1: Registra cuando se inserta un animal

-

```
DELIMITER //
CREATE TRIGGER animal_insert_trigger
BEFORE INSERT ON zoologico.animal
FOR EACH ROW
BEGIN
    INSERT INTO zoologico.auditoria (usuario, accion, fecha)
    VALUES (user(), 'Insertó un animal a la hora', NOW());
END;

// DELIMITER ;
```

- Trigger 2: Registra cuando se actualiza un animal

```
DELIMITER //
```

- ```
CREATE TRIGGER animal_update_trigger
BEFORE UPDATE ON zoologico.animal
FOR EACH ROW
BEGIN
 INSERT INTO zoologico.auditoria (usuario, accion, fecha)
 VALUES (user(), 'Actualizó un animal a la hora', NOW());
END;
// DELIMITER ;
```

- Trigger 3: Registra cuando se inserta una nueva dieta

```
DELIMITER //
```

```
CREATE TRIGGER dieta_insert_trigger
BEFORE INSERT ON zoologico.dieta
FOR EACH ROW
BEGIN
 INSERT INTO zoologico.auditoria (usuario, accion, fecha)
 VALUES (user(), 'Insertó una nueva dieta a la hora', NOW());
END;
// DELIMITER ;
```

- Trigger 4: Registra cuando se actualiza una dieta

```
DELIMITER //
```

```
CREATE TRIGGER dieta_update_trigger
BEFORE UPDATE ON zoologico.dieta
FOR EACH ROW
BEGIN
 INSERT INTO zoologico.auditoria (usuario, accion, fecha)
 VALUES (user(), 'Actualizó una dieta a la hora', NOW());
END;
// DELIMITER ;
```

Para hacer validez, se creó una tabla auditoria la cual presenta el uso de los triggers.

1 • SELECT \* FROM zoologico.auditoria;

| usuario        | accion                            | fecha               |
|----------------|-----------------------------------|---------------------|
| root@localhost | Insertó una nueva dieta a la hora | 2023-02-19 15:19:24 |
| root@localhost | Insertó una nueva dieta a la hora | 2023-02-19 15:19:24 |
| root@localhost | Insertó una nueva dieta a la hora | 2023-02-19 15:19:24 |

1 • SELECT \* FROM zoologico.auditoria;

| usuario        | accion                      | fecha               |
|----------------|-----------------------------|---------------------|
| root@localhost | Insertó un animal a la hora | 2023-02-19 15:19:29 |
| root@localhost | Insertó un animal a la hora | 2023-02-19 15:19:29 |
| root@localhost | Insertó un animal a la hora | 2023-02-19 15:19:29 |

Luego de realizar los triggers, se procede a realizar los procedimientos:

- Procedimiento 1: Insertar animal

```
DELIMITER //
CREATE PROCEDURE insertar_animal (IN idAnimal VARCHAR(10), IN nombreAnimal VARCHAR(45), IN especie_id VARCHAR(10), IN dieta_id VARCHAR(10))
BEGIN
 INSERT INTO zoologico.animal (idAnimal, nombreAnimal, especie_id, dieta_id)
 VALUES (idAnimal, nombreAnimal, especie_id, dieta_id);
END;

CALL insertar_animal ('A55', 'Pantera', 'ESP1', 'D1');
```

- Procedimiento 2: Insertar un alimento

```
// DELIMITER ;

DELIMITER //
CREATE PROCEDURE insertar_alimento (IN idAlimento VARCHAR (10), IN nombreAlimento VARCHAR(20), IN tipoAlimento VARCHAR(45))
BEGIN
 INSERT INTO zoologico.alimento (idAlimento, nombreAlimento, tipoAlimento) VALUES (idAlimento, nombreAlimento, tipoAlimento);
END;

CALL insertar_alimento ('A11', 'ALPISTE', 'GRANO');
```

- Procedimiento 3: Insertar un entrenador

```
DELIMITER //
CREATE PROCEDURE insertar_entrenador (IN idEntrenador VARCHAR(10), IN nombreEntrenador VARCHAR(45))
BEGIN
 INSERT INTO zoologico.entrenador (idEntrenador, nombreEntrenador) VALUES (idEntrenador, nombreEntrenador);
END;

CALL insertar_entrenador ('EME10', 'Tin tin');
```

- Procedimiento 4: Insertar un proveedor

```
DELIMITER //
CREATE PROCEDURE insertar_proveedor (IN idproveedor VARCHAR(10), IN nombreProveedor VARCHAR(45))
BEGIN
 INSERT INTO zoologico.proveedor (idproveedor, nombreProveedor) VALUES (idproveedor, nombreProveedor) ;
END;

CALL insertar_proveedor ('PRO10', 'Max Power');

// DELIMITER ;
```

Después de realizar los procedimientos asignados se procede a realizar las vistas:

Vista 1: Muestras los animales por especie. Me parece importante tener una vista de este tipo puesto que permite caracterizar a los animales dentro de su grupo. Esto permite agruparlos en lugares cercanos y proporcionar una guía para el veterinario y el entrenador al momento de generar informes y valorar a un animal.

```
#Vista 1 Animales por especies
CREATE VIEW Cantidad_AnimalesXEspecies AS
SELECT especie.nombreEspecie, COUNT(Animal.especie_id) AS cantidad
FROM especie
JOIN Animal ON especie.idEspecie = Animal.especie_id
GROUP BY especie.nombreEspecie;
```

Vista 2: Mostrar animales, su especie y su dieta: Es una ampliación de la vista anterior, resulta útil al momento de caracterizar los informes. También le facilita el trabajo al veterinario y al entrenador. Puesto que les sirve de guía.

```
CREATE VIEW Especies_X_ANIMALES_X_DIETA AS
SELECT Animal.nombreAnimal, especie.nombreEspecie, Dieta.nombreDieta
FROM Animal
JOIN especie ON Animal.especie_id = especie.idEspecie
JOIN Dieta ON Animal.dieta_id = Dieta.idDieta;
```

Vista 3: Cantidad de animales (numérico) por especie: Parece bastante útil para tener un registro de los animales presentes en cada tipo de hábitad.

```
CREATE VIEW Cantidad_animalesXEspecie AS
SELECT esp.nombreEspecie, COUNT(a.idAnimal) AS cantidad
FROM Animal a
INNER JOIN especie esp ON a.especie_id = esp.idEspecie
GROUP BY esp.idEspecie;
```

Vista 4: Muestra especie con más animales: Resulta útil para evitar una sobrepoblación de un mismo animal. Puede generar un gasto extra mantenerlos.

```
#Vista 4 que muestra la especie con más animales
CREATE VIEW Especie_x_animales AS
SELECT e.nombreEspecie, COUNT(*) AS cantidad_animales
FROM especie e
JOIN Animal a ON e.idEspecie = a.especie_id
GROUP BY e.idEspecie
ORDER BY cantidad_animales DESC
LIMIT 1;
```

Vista 5: Muestra animal, dieta y su dosis: Sirve bastante como guía para el encargado de proporcionar el alimento a los animales.

```
#Vista 5 que muestra el animal, su tipo de dieta y dosis
CREATE VIEW Dieta_y_Dosis_Animales AS
SELECT nombreAnimal, nombreDieta, dosisDieta FROM Animal
JOIN Dieta ON Animal.dieta_id = Dieta.idDieta;
```

Teniendo ya todas las consultas, procedimientos, triggers y vistas terminadas. Pasamos a hacer la inserción con Java. Se procede a mostrar cada método que permite insertar información en cada tabla.

```
public static void main(String[] args) throws SQLException {
 openConnection();
 insertarAlimento();
 insertarDieta();
 insertarEntrenador();
 insertarVeterinario();
 insertarProveedor();
 insertarespecie();
 insertarEmpleadoLogistico();
 insertarAlimento_Dieta();
 insertarAnimal();
 insertarOrdenCompra();
 insertarInforme_peso();
 insertarFactura();
 insertartelefono_logistico();
 insertartelefonoEntrenador();
 insertartelefonoproveedor();
 insertartelefonoVeterinario();
 insertarVeterinarioDieta();
}
```

#### - Alimento

```
private static void insertarAlimento() {
 for (int i = 1; i < 51; i++) {
 Alimento alimento = new Alimento();
 alimento.setIdAlimento("Ali" + i);
 alimento.setNombreAlimento(faker.food().ingredient());
 alimento.setTipoAlimento(faker.food().spice());
 mysqlOperation.setSqlStatement("insert into alimento values(" + alimento.getIdAlimento() + ", " + alimento.getNombreAlimento() +
 ", " + alimento.getTipoAlimento() + ")");
 mysqlOperation.executeUpdate();
 }
}
```



## - Dieta

```
private static void insertarDieta(){
 for (int i = 1; i < 51; i++) {
 Dieta dieta = new Dieta();
 dieta.setIdDieta("D"+ i);
 dieta.setNombreDieta(faker.food().ingredient());
 dieta.setDosisDieta(faker.numerify(numberString: "##"));
 dieta.setDosisDieta(dieta.getDosisDieta() + "Kilos/dia");

 mySqlOperation.setSqlStatement("insert into dieta values('" +dieta.getIdDieta()+"','"+dieta.getNombreDieta()+"','"+
 +dieta.getDosisDieta()+"')");
 mySqlOperation.executeSqlStatementVoid();
 }
}
```

## - Entrenador

```
private static void insertarEntrenador(){
 for (int i = 1; i < 51; i++) {
 Entrenador trainer = new Entrenador();
 Entrenador entrenador = new Entrenador();
 entrenador.setIdEntrenador("EME" + i);
 entrenador.setNombreEntrenador(faker.name().firstName());
 mySqlOperation.setSqlStatement("insert into entrenador values('" +entrenador.getIdEntrenador()+
 "','"+entrenador.getNombreEntrenador()+"')");
 mySqlOperation.executeSqlStatementVoid();
 }
}
```

## - Veterinario

```
private static void insertarVeterinario(){
 for (int i = 1; i < 51; i++) {
 Veterinario vet = new Veterinario();
 Veterinario veterinario = new Veterinario();
 veterinario.setIdVeterinario("V"+ i);
 veterinario.setNombreVeterinario(faker.name().firstName());
 mySqlOperation.setSqlStatement("insert into veterinario values('" +veterinario.getIdVeterinario()+"','"+
 +veterinario.getNombreVeterinario()+"')");
 mySqlOperation.executeSqlStatementVoid();
 }
}
```

## - Proveedor

```
private static void insertarProveedor(){
 for (int i = 1; i < 51; i++) {
 Proveedor prov = new Proveedor();
 Proveedor proveedor = new Proveedor();
 proveedor.setIdproveedor("PRO"+ i);
 proveedor.setNombreProveedor(faker.name().firstName());
 mySqlOperation.setSqlStatement("insert into proveedor values('" +proveedor.getIdproveedor()+"','"+proveedor.getNombreProveedor()+"')");
 mySqlOperation.executeSqlStatementVoid();
 }
}
```

- Especie

```
private static void insertarespecie(){
 for (int i = 1; i < 51; i++) {
 Entrenador trainer = new Entrenador();
 especie especies = new especie();
 especies.setIdEspecie("E" + i);
 especies.setNombreEspecie(faker.witcher().monster());
 mysqlOperation.setSqlStatement("insert into especie values('"+especies.getIdEspecie()+
 "','"+especies.getNombreEspecie()+"')");
 mysqlOperation.executeSqlStatementVoid();
 }
}
```

- Empleado logístico

```
private static void insertarEmpleadoLogistico() {
 for (int i = 1; i < 51; i++) {
 EmpleadoLogistico empleadoLogistico = new EmpleadoLogistico();
 empleadoLogistico.setIdEmpleadoLogistico("EM"+ i);
 empleadoLogistico.setNombreLogistico(faker.name().firstName());
 mysqlOperation.setSqlStatement("INSERT INTO empleadoLogistico VALUES ('" + empleadoLogistico.getIdEmpleadoLogistico()
 + "',' " + empleadoLogistico.getNombreLogistico() + "')");
 mysqlOperation.executeSqlStatementVoid();
 }
}
```

- Alimento con dieta

```
private static void insertarAlimento_Dieta(){
 for (int i = 1; i < 51; i++) {
 String dietaAlimentoId = "D"+ i;
 String alimento_Dieta_ID = "Al"+i;

 mysqlOperation.setSqlStatement("insert into alimento_dieta values('" +dietaAlimentoId+ "',' " +alimento_Dieta_ID+"')");
 mysqlOperation.executeSqlStatementVoid();
 }
}
```

- Animal

```
private static void insertarAnimal(){
 for (int i = 1; i < 51; i++) {
 Animal animal = new Animal();
 animal.setIdAnimal("A" + i);
 animal.setNombreAnimal(faker.animal().name());
 animal.setEspecie_id("E" + i);
 animal.setDieta_id("D" + i);
 mysqlOperation.setSqlStatement("insert into animal values('"+animal.getIdAnimal()+"','"+animal.getNombreAnimal()+
 "','"+animal.getEspecie_id()+"','"+animal.getDieta_id()+"')");
 mysqlOperation.executeSqlStatementVoid();
 }
}
```

## - Orden compra

```
private static void insertarOrdenCompra() {
 for (int i = 1; i < 51; i++) {
 OrdenesCompra ordenesCompra = new OrdenesCompra();
 ordenesCompra.setIdOrdenesCompra("ORD" + i);
 ordenesCompra.setNombreOrden(faker.lorem().sentence());
 ordenesCompra.setFechaOrden(String.valueOf(faker.date().between(Date.from
 (LocalDate.of(year: 2000, month: 1, dayOfMonth: 1).atStartOfDay(ZoneId.systemDefault()).toInstant(), new Date()))));
 ordenesCompra.setDescripcion(faker.lorem().sentence());
 ordenesCompra.setOrdenProveedorId("PRO" + i);
 ordenesCompra.setLogisticoIdOrden("EM" + i);
 mySqlOperation.setSqlStatement("INSERT INTO orden_compra VALUES ('" + ordenesCompra.getIdOrdenesCompra() + "', '" +
 ordenesCompra.getNombreOrden() + "', '" + ordenesCompra.getFechaOrden() + "', '" + ordenesCompra.getDescripcion()
 + "', '" + ordenesCompra.getOrdenProveedorId() + "', '" + ordenesCompra.getLogisticoIdOrden() + "');");
 mySqlOperation.executeSqlStatementVoid();
 }
}
```

## - Informe peso

```
private static void insertarInforme_peso() {
 for (int i = 1; i < 51; i++) {
 Informe_peso informePeso = new Informe_peso();
 informePeso.setIdInforme("IN" + i);
 informePeso.setPesoAnimal(faker.number().numberBetween(10, 50) + " kg");
 String entrenadorId = "EME" + i;
 String veterinarioId = "V" + i;
 String animalId = "A" + i;
 mySqlOperation.setSqlStatement("INSERT INTO informe_peso VALUES ('" + informePeso.getIdInforme() + "', '" + informePeso.getPesoAnimal() +
 "', '" + entrenadorId + "', '" + veterinarioId + "', '" + animalId + "');");
 mySqlOperation.executeSqlStatementVoid();
 }
}
```

## - Factura

```
private static void insertarFactura() {
 for (int i = 1; i < 51; i++) {
 Factura fac = new Factura();
 Factura factura = new Factura();
 factura.setIdFactura("FA" + i);
 factura.setId_alimento_factura("ALI" + i);
 factura.setCantidad(Integer.toString(faker.number().numberBetween(1, 20)));
 factura.setPrecio(Integer.toString(faker.number().numberBetween(100000, 2000000)));
 factura.setProveedorIdFactura("PRO" + i);
 mySqlOperation.setSqlStatement("INSERT INTO factura VALUES ('" + factura.getIdFactura() + "', '" + factura.getId_alimento_factura() +
 "', '" + factura.getCantidad() + "', '" + factura.getPrecio() + "', '" + factura.getProveedorIdFactura() + "');");
 mySqlOperation.executeSqlStatementVoid();
 }
}
```

## - Teléfono logístico

```
private static void insertartelefono_logistico() {
 for (int i = 1; i < 51; i++) {
 String logisticoId = "EM" + i;
 String numero = (faker.phoneNumber().cellPhone().toString());
 mySqlOperation.setSqlStatement("INSERT INTO telefono_logistico VALUES ('" + logisticoId + "', '" + numero + "');");
 mySqlOperation.executeSqlStatementVoid();
 }
}
```

- Telefono entrenador

```
private static void insertartelefonoEntrenador() {
 for (int i = 1; i < 51; i++) {
 String idtelefonoEntrenador = "EME"+ i;
 String numero = (faker.phoneNumber().cellPhone().toString());
 mySqlOperation.setSqlStatement("INSERT INTO telefonoentrenador VALUES ('" +idtelefonoEntrenador+ "', '" +numero+"');");
 mySqlOperation.executeSqlStatementVoid();
 }
}
```

- Telefono proveedor

```
private static void insertartelefonoproveedor() {
 for (int i = 1; i < 51; i++) {
 String proveedorIdTelefono = "PRO"+ i;
 String numero = (faker.phoneNumber().cellPhone().toString());
 mySqlOperation.setSqlStatement("INSERT INTO telefonoproveedor VALUES ('" +proveedorIdTelefono+ "', '" +numero+"');");
 mySqlOperation.executeSqlStatementVoid();
 }
}
```

- Telefono veterinario

```
private static void insertartelefonoVeterinario() {
 for (int i = 1; i < 51; i++) {
 String idtelefonoVeterinario = "V"+ i;
 String numero = (faker.phoneNumber().cellPhone().toString());
 mySqlOperation.setSqlStatement("INSERT INTO zoologico.telefonoveterinario VALUES ('" +idtelefonoVeterinario+ "', '" +numero+"');");
 mySqlOperation.executeSqlStatementVoid();
 }
}
```

- Veterinario y su dieta

```
private static void insertarVeterinarioDieta() {
 for (int i = 1; i < 51; i++) {
 String idVeterinarioDieta = "V" + i;
 String dietaVeterinarioId = "D" + i;

 mySqlOperation.setSqlStatement("INSERT INTO veterinariodieta VALUES ('" +idVeterinarioDieta+ "', '" +dietaVeterinarioId+"');");
 mySqlOperation.executeSqlStatementVoid();
 }
}
```