PROYECTO FINAL BASE DE DATOS

LUIS DAVID ANGEL RINCON

S541B

RESUMEN DETALLADO DEL SEMESTRE

BASE DE DATOS 1

GERMAN ANDRES MORALES LEON

INGENIERÍA DE SISTEMAS

ANTONIO JOSE CAMACHO SEDE SUR

2025

Contenido

1. PRIMER CORTE

Para la primera parte se investigó el sistema de gestión de bases de datos.

1.1.1. CONCEPTOS

Se define como una colección, agrupación o conjunto de herramientas blandas y herramientas duras, permiten interactuar entre ellas y al hacerlo permite la interacción entre los datos que llegan a un proceso donde se conservan se controlan y se administran para una posterior toma de decisiones y donde el dato se transforma en información.

Vino la línea del tiempo de las bases de datos donde primero se inicio con las tarjetas perforadas, en el siglo XX el avance de las bases de datos incremento rápidamente, en 1950 se crearon las cintas magnéticas, diez años después llego el disco duro que permitió generar bases de datos almacenadas en sistemas operativos. En 1970 se empezaron a popularizar los modelos relacionales, pocos años después se propuso modelo Entidad-relación. En 1980 nacieron los lenguajes de consulta estructurados (SQL), se convirtió en el lenguaje de consulta estándar, ya que las bases de datos relacionales con sus sistemas de tablas pudieron competir con las bases de datos con las bases jerárquicas y de red, como consecuencia de que su nivel de programación era sencillo y su nivel de programación era relativamente bajo. En 1990 los programas Access y Excel de la compañía Microsoft Office marcan el comienzo de las bases de datos orientadas a objetos. Durante el siglo 21 se crearon y/o actualizaron las bases de datos virtuales tales como: Microsoft SQL, Oracle, IB DBZ, Teradata, entre otros. En 2010 se da lugar a nuevas bases de datos independientes en la nube, como iCloud, Google Drive, Email, etc.

1.1.1. Sistema gestor de base de datos (SGBD): sus objetivos son almacenar y recuperar datos con el propósito de que sea confiable y seguro.

El software y hardware interactúan entre ellos para gestionar datos, procesar datos, interactuar con los datos, tomar decisiones y manejar infor11mación.

1.1.2. Anomalías en las bases de datos: redundancia, duplicidad, inconsistencia, integridad, seguridad, dificulta de acceso, concurrencia, atomicidad, aislamiento.

1.1.3. Visión de datos: vistas, controlador, modelo.

Un sistema de base de datos es una colección de datos interrelacionada y un conjunto de programas que permiten a los usuarios tener acceso a esos datos y modificarlos. Una de las principales finalidades de los sistemas de base de datos es ofrecer a los usuarios una visión abstracta de los datos. Es decir, el sistema oculta ciertos detalles del mundo en que se almacenan y mantienen los datos.

Figura 1.1. Los tres niveles de abstracción de datos.

1.2. MODELO DE LOS DATOS

Se define como el modelo de los datos a una abstracción de elementos orinados del mundo real que basados en conceptos semánticos de los datos y mediante herramientas graficas geométricas nos permitirán realizar un esquema ejemplarizante de una base de datos requerida por personas o empresas con necesidades y requerimientos específicos.

Su fundamentación es optimizar y organizar datos. En el modelo de los datos podemos encontrar tales como el modelo entidad-relación y el diagrama entidad-relación que son construidos por I.S. también nos encontramos con el modelo relacional, que es construido por el motor de base de datos.

Las vistas, la lógica y lo físico (modelado BD), interactúan entre ellas para lograr obtener un buen resultado.

1.3. LEVANTAMIENTO DE DATOS

Para el modelamiento de nuestra base de datos Ukumari, seguimos el modelo Entidad-relación. Este modelo se utiliza para construir bases de datos relacionales. Bajo este modelo debemos tomar la información entregada por la empresa.

Se nos solicita crear el modelo de la base de datos para el servicio de la atención de pacientes veterinarios.

Para este modelo, realizamos el levantamiento de datos. Se registrarán los valores que cumplan con el requerimiento de la empresa.

Para cumplir con los propósitos y los requisitos para esta base de datos, el cliente nos suministró información como su página web, mapas y un video donde se explica el manejo que tiene para proteger la fauna y la flora. Donde encontramos mucha información de todos los animales que se encuentran el bioparque, de ellos podemos reconocer que son una entidad importante para el objetivo de la base de datos, encontramos sus atributos y los datos necesarios para su cuidado, los espacios necesarios y dietas.

También podemos notar que existen diversos espacios en bioparque y también muchas labores, que se tienen que desempeñar con personas selectas para cada cargo.

Detrás de las operaciones que desempeña cada persona existe un historial importante, para llevar el control de sus animales, sus empleados y visitantes, que pueden tomar diferentes roles.

1.4. MODELO ENTIDAD-RELACION (MER)

Es una herramienta para representar gráficamente como las entidades (objetos, personas o concetos) y sus relaciones se organizan dentro de un sistema, como una base de datos. Es un diagrama que ayuda a visualizar como se conectan los diferentes elementos y como la información fluye entre ellos.

1.4.1. Elementos clave dentro del modelo entidad-relación:

- **Entidades:** son los objetos principales del sistema, representados gráficamente como rectángulos. Pueden ser personas, objetos, conceptos o eventos.
- **Atributos:** son las características de las entidades, representados como elipses o óvalos.
- **Relaciones:** son las conexiones entre las entidades, representadas como rombos o líneas que unen las entidades. Indican como las entidades interactúan entre sí.
- **Cardinalidad:** define la cantidad de instancias de una entidad que pueden relacionarse con otra. Por ejemplo, uno a uno, uno a muchos o muchos a muchos

1.4.2. Usos del modelo entidad-relación:

- **Diseño de bases de datos:** se utiliza para visualizar y diseñar la estructura de una base de datos.

- **Ingeniería de software:** ayuda a los desarrolladores a comprender y modelar la estructura de un sistema de software.
- **Sistemas de información empresarial:** permite visualizar las relaciones entre diferentes elementos de la empresa.
- **Educación e investigación:** se utiliza para diseñar y comprender la estructura de los sistemas.

1.4.3. Dentro del modelo de los datos existen las siguientes características:

- Conjunto de atributos.
- Conjunto de asociaciones y cardinalidades.
- Conjunto de líneas de flujo.
- Conjunto de entidades.

Cada una de ellas va a tener una representación gráfica. Las entidades por cuadrados, los atributos por círculos, las asociaciones por rombos y líneas de flujo por líneas

Entidades → 

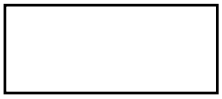
Atributos → 

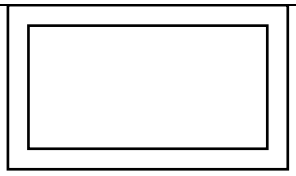
Asociaciones → 

Líneas de flujo →

Se define como entidad a personas, animales, cosas u objetos sacados del mundo real, estas entidades nacerán como también tendrán características que los resaltaran e identificaran acorde al ámbito en que se desenvuelve.

1.4.4. Tipos de entidades:

Primarias	Fuertes		La entidad fuerte se reconoce por que dentro de sus atributos hay uno que se caracteriza por ser de valor único e irrepetible, lo que significa que dentro de esta entidad no podrán habitar datos repetidos.
Secundarias	Débiles		La entidad débil se reconoce porque dentro de sus atributos hay uno que se caracteriza por ser de valor repetible

			duplicado, esto quiere decir que podrá darse la duplicidad de datos dentro de ella, este tipo de atributo tendrá las características derivadas de una entidad fuerte.
	Principales		Este tipo de entidades se caracterizan por tener y compartir un atributo de valor único, propio de la entidad y otro valor que se deriva de una entidad fuerte.

De esta forma podemos identificar que entidades puede haber en nuestra base de datos Ukumari y que tipo de entidades cada una.

1.4.5. Estas son las entidades encontradas y su tipo:

Animal → Entidad secundaria principal: ya que su idAnimal que es su atributo de valor único, se compartirá con entidades como historia clínica y tratamiento. Se derivan otros atributos de entidades como la dieta, el alimento, el hábitat, el país de origen, la raza y la especie.

Veterinario → Entidad secundaria principal: ya que su idVeterinario que es su atributo de valor único, se compartirá con la entidad historia clínica. Se deriva un atributo de la entidad especialidad veterinaria.

Usuario → Entidad secundaria principal: ya que su idUsuario que es su atributo de valor único, se compartirá con la entidad bitácora. Se deriva un atributo de la entidad rol.

Historia clínica → Entidad secundaria principal: ya que su idHistoriaClinica que es su atributo de valor único, se compartirá con la entidad detalle historia clínica. Se derivan otros atributos de la entidades animal y veterinario.

Tratamiento → Entidad secundaria principal: ya que su idTratamiento que es su atributo de valor único, se compartirá con la entidad medicamento. Se deriva un atributo de la entidad animal.

Diagnostico → Entidad secundaria débil: ya que uno de sus atributos se deriva de la entidad enfermedad.

Detalle historia clínica → Entidad secundaria débil: ya que uno de sus atributos se deriva de la entidad historia clínica.

Bitácora → Entidad secundaria débil: ya que uno de sus atributos se deriva de la entidad usuario.

Medicamento → Entidad secundaria débil: ya que uno de sus atributos se deriva de la entidad tratamiento.

País de origen → Entidad primaria fuerte: ya que su identificador único se comparte con la entidad animal y no recibe atributos derivados.

Hábitat → Entidad primaria fuerte: ya que su identificador único se comparte con la entidad animal y no recibe atributos derivados.

Raza → Entidad primaria fuerte: ya que su identificador único se comparte con la entidad animal y no recibe atributos derivados.

Especie → Entidad primaria fuerte: ya que su identificador único se comparte con la entidad animal y no recibe atributos derivados.

Dieta → Entidad primaria fuerte: ya que su identificador único se comparte con la entidad animal y no recibe atributos derivados.

Alimento → Entidad primaria fuerte: ya que su identificador único se comparte con la entidad animal y no recibe atributos derivados.

Rol → Entidad primaria fuerte: ya que su identificador único se comparte con la entidad usuario y no recibe atributos derivados.

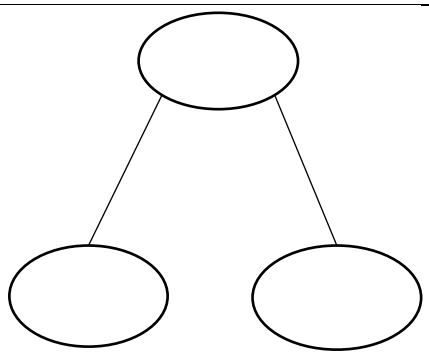
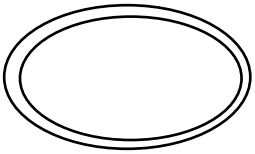
Cuidador → Entidad primaria fuerte: ya que su identificador único se comparte con la entidad historia clínica y no recibe atributos derivados.

Especialidad veterinaria → Entidad primaria fuerte: ya que su identificador único se comparte con la entidad veterinario y no recibe atributos derivados.

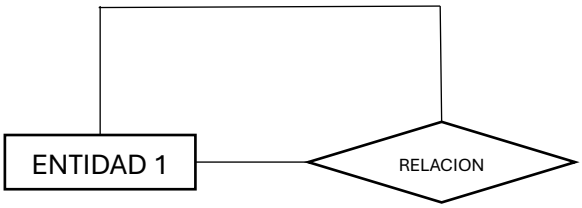
Enfermedad → Entidad primaria fuerte: ya que su identificador único se comparte con la entidad diagnóstico y no recibe atributos derivados.

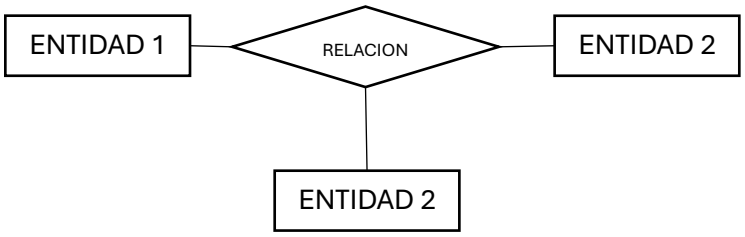
1.4.6. Tipos de atributos:

	Simples		No están divididos en subpartes.

Por estructura	Compuestos		Se pueden dividir en subpartes (es decir, en otros atributos). Por ejemplo, nombre podría estar estructurado o dividido nombre uno, nombre dos, apellido uno y apellido dos
	Derivado		Es el atributo que se genera a partir de otro atributo, por ejemplo, el atributo edad se genera a partir de la fecha de nacimiento.
Por valor	Multivalorado		Es donde el valor del dato puede permitir derivar a otros atributos y su valor nos permite obtener múltiples valores como por ejemplo la fecha de nacimiento permite saber la edad y la generación, otro podría ser la dirección, la cual nos permite derivar la comuna, el estrato u otros valores

Tipos de relaciones:

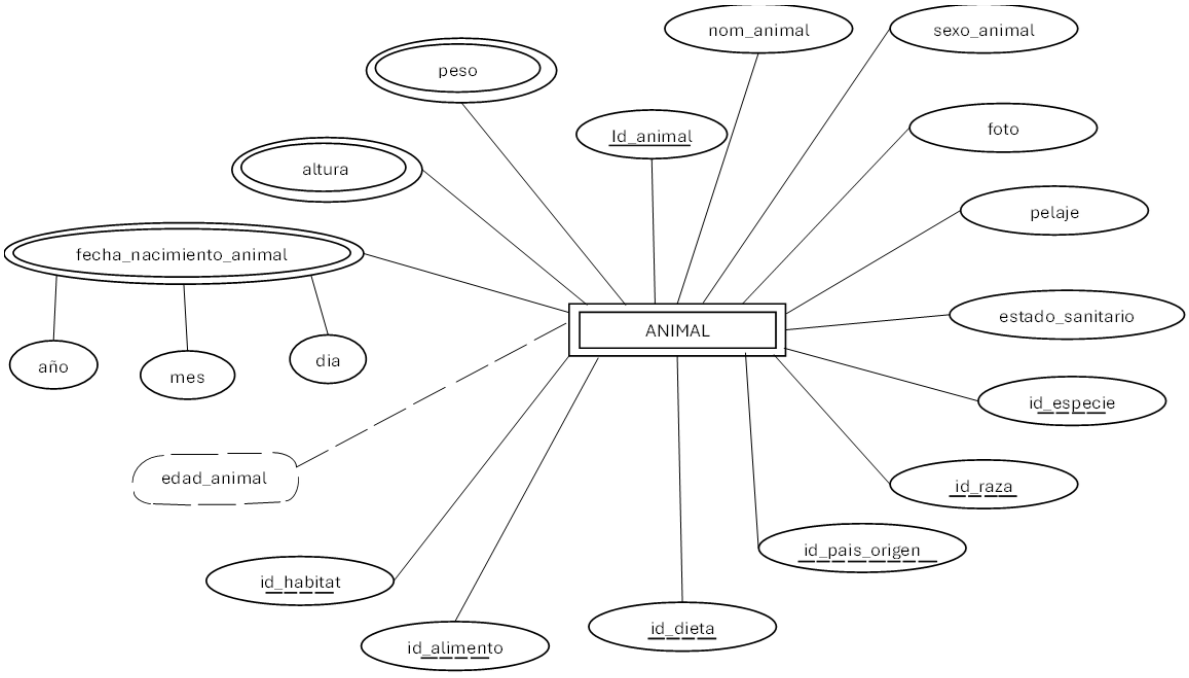
Reflexiva – Unaria – Grado 1		La relación reflexiva es un tipo de relación en la cual la entidad se asocia a si misma cuando tiene que especificar un rol que la entidad; este tipo de relación también se le conoce como relación recursiva.
Binarias		Esta relación existe cuando solo dos entidades se relacionan para compartir datos a través de los atributos, puede ser entre una entidad fuerte y otra débil o entre una entidad fuerte y otra fuerte. Esta relación es la mas común entre las entidades y es la que mayormente se usa para

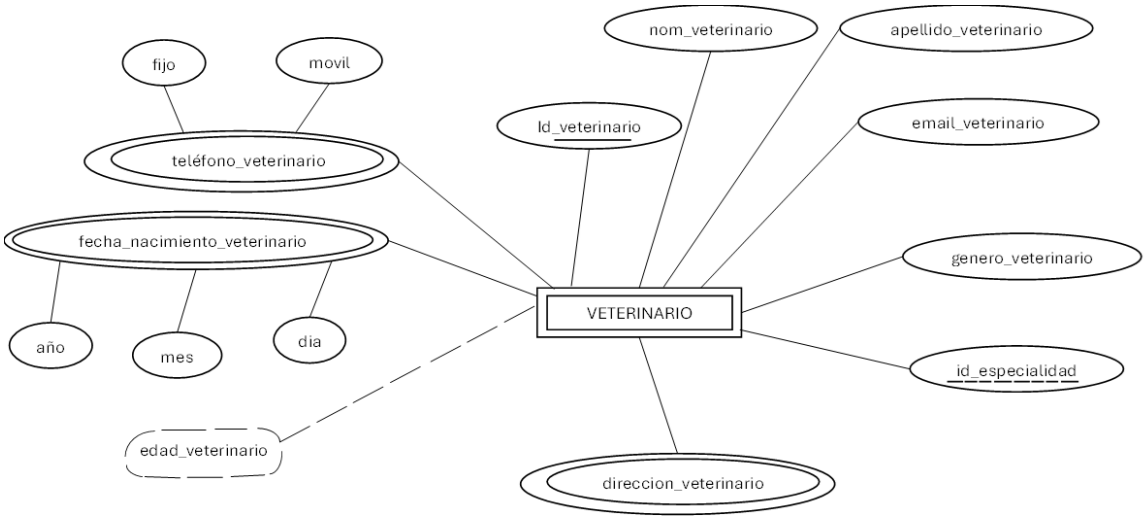
		relacionar las entidades.
Ternarias		Esta relación es cuando tres entidades se relacionan a través de un área común y comparten los mismos datos, a través de los atributos claves de ellas. Pero cuando ocurre este tipo de relación entre las entidades, porque no es común este tipo de relación.

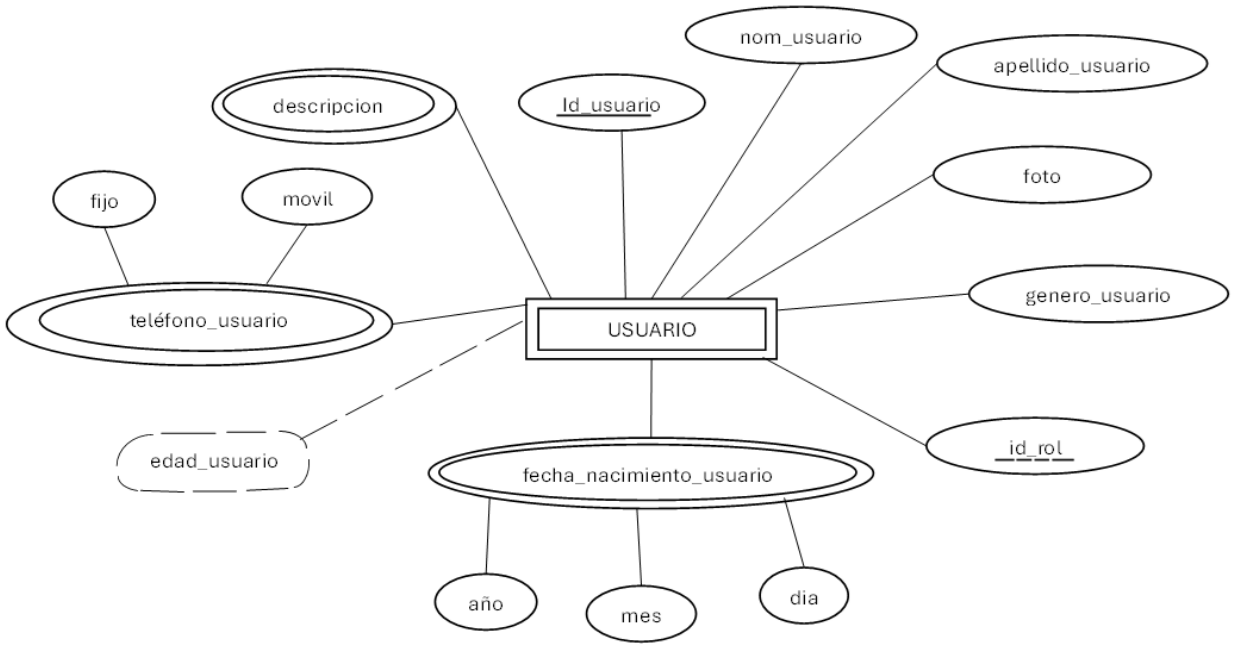
1.5. ENTIDAD Y ATRIBUTO (EyA)

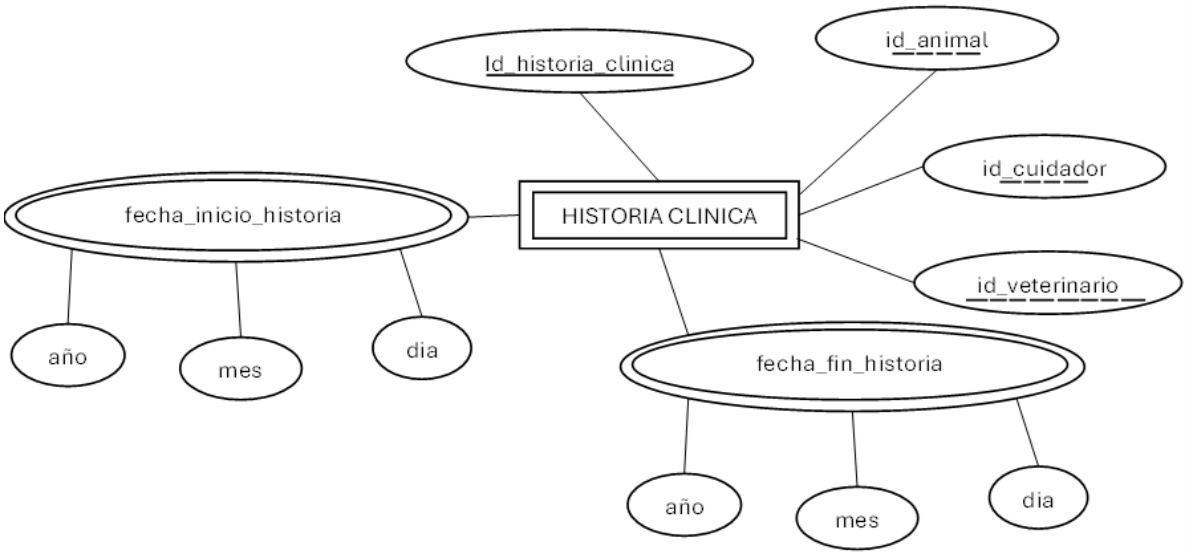
1.5.1. Entidades secundarias principales

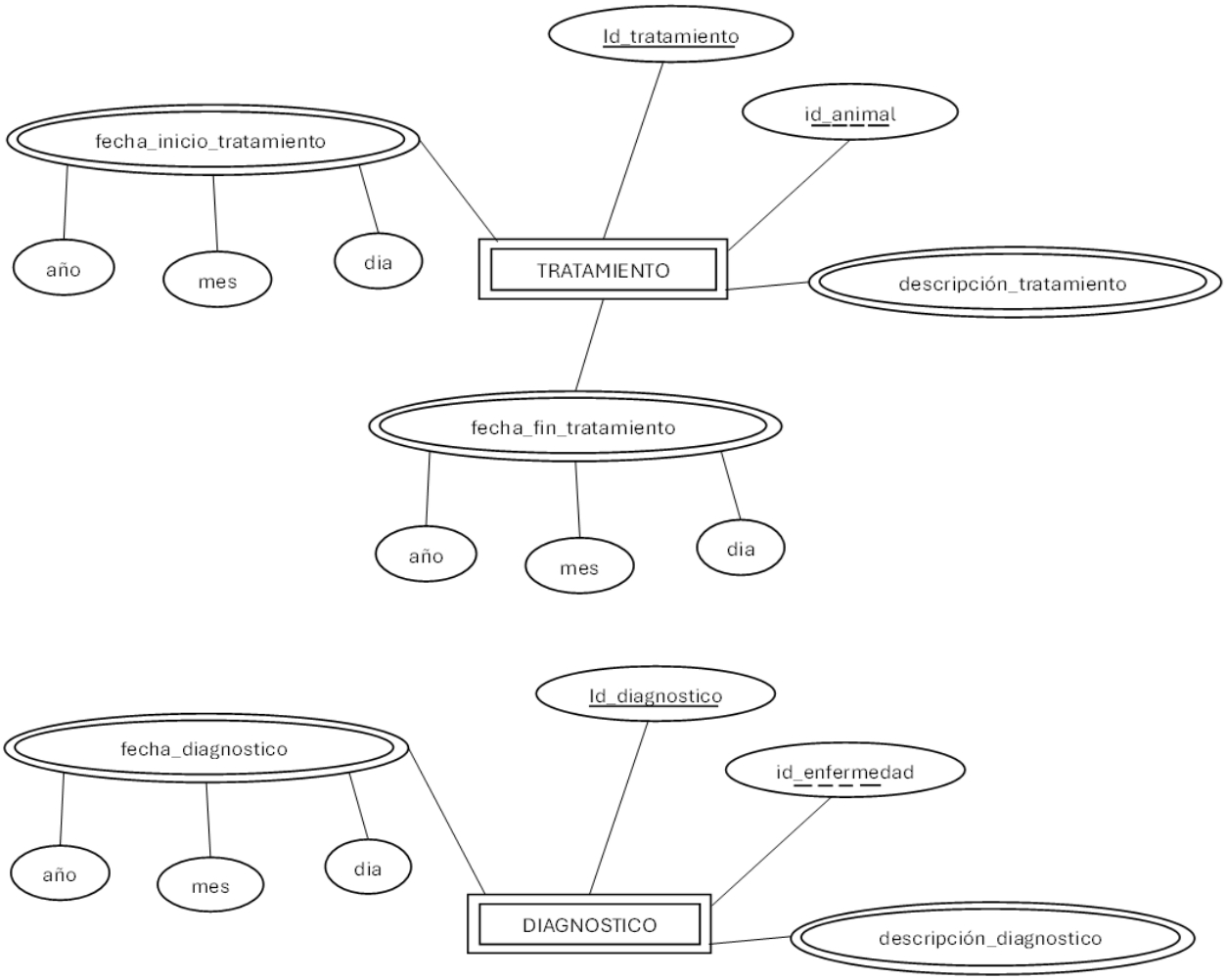
Como sabemos, la entidad animal es una entidad secundaria principal. Como identificador único tenemos el identificador de la entidad animal, esta graficada como un atributo simple y su línea inferior la identifica como el atributo con valores únicos (PK). Para los atributos simples tenemos el nombre del animal, el sexo del animal, la foto del animal, el pelaje del animal y el estado sanitario del animal. Para los atributos simples con línea intermitente inferior, identificándolo como un atributo derivado de otra entidad, tenemos al identificador de la entidad especie, el identificador de la entidad raza, el identificador de la entidad país de origen, el identificador de la entidad dieta, el identificador de la entidad alimento y el identificado de la entidad hábitat. El atributo edad del animal, esta graficado como un atributo derivado, ya que se deriva o se obtiene del atributo que sería la fecha de nacimiento del animal, este atributo esta graficado como un atributo multivalorado. Por último, tenemos como atributos multivalorados a los atributos altura del animal y peso del animal.


nom_animal
sexo_animal
peso
Id_animal
foto
altura
pelaje
fecha_nacimiento_animal
ANIMAL
estado_sanitario
año
mes
dia
id_especie
edad_animal
id_raza
id_pais_origen
id_habitat
id_dieta
id_alimento


nom_veterinario
apellido_veterinario
fijo
movil
Id_veterinario
email_veterinario
teléfono_veterinario
fecha_nacimiento_veterinario
genero_veterinario
VETERINARIO
año
mes
dia
id_especialidad
edad_veterinario
direccion_veterinario

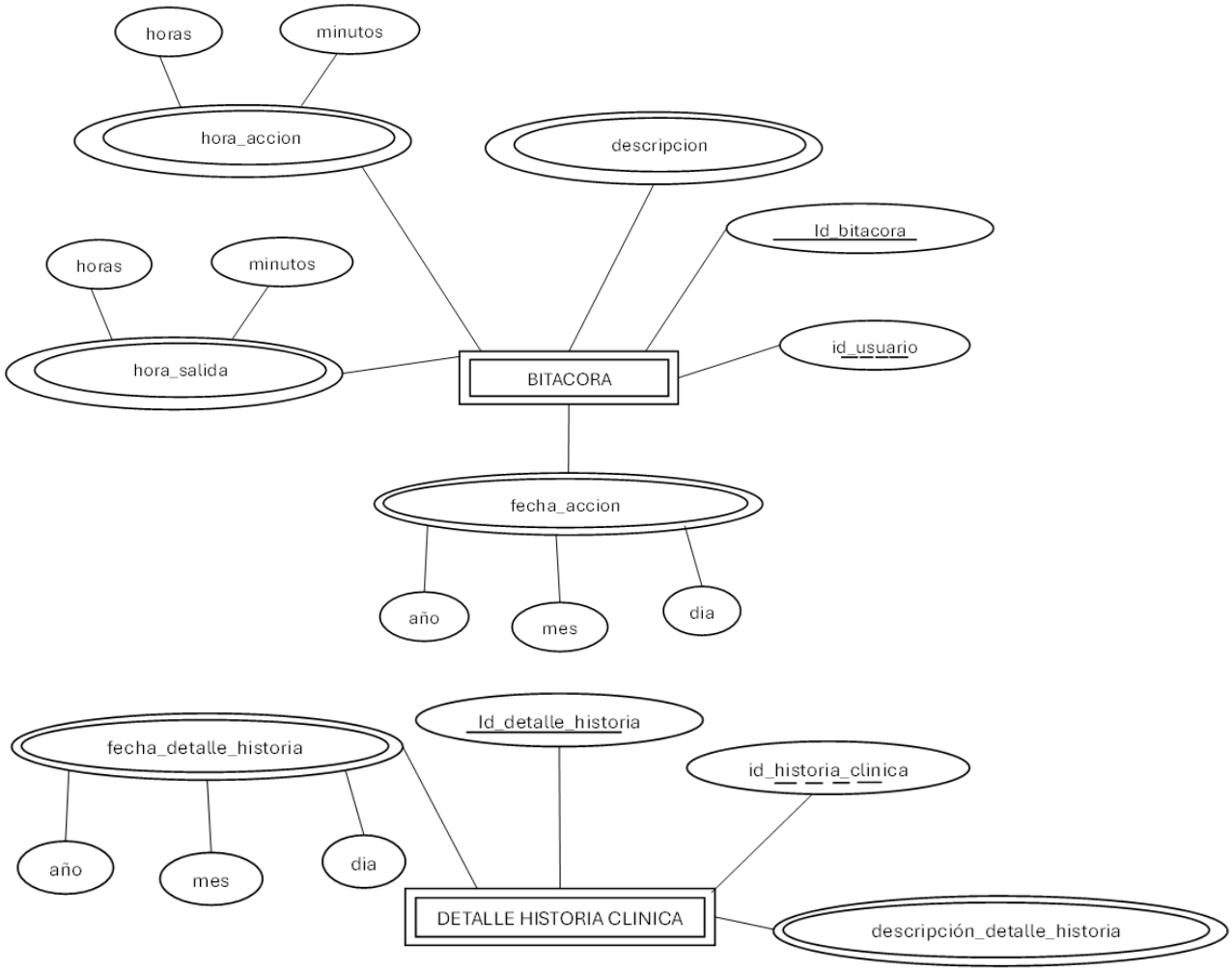

nom_usuario
apellido_usuario
descripcion
Id_usuario
foto
fijo
movil
genero_usuario
teléfono_usuario
USUARIO
edad_usuario
id_rol
fecha_nacimiento_usuario
año
mes
dia

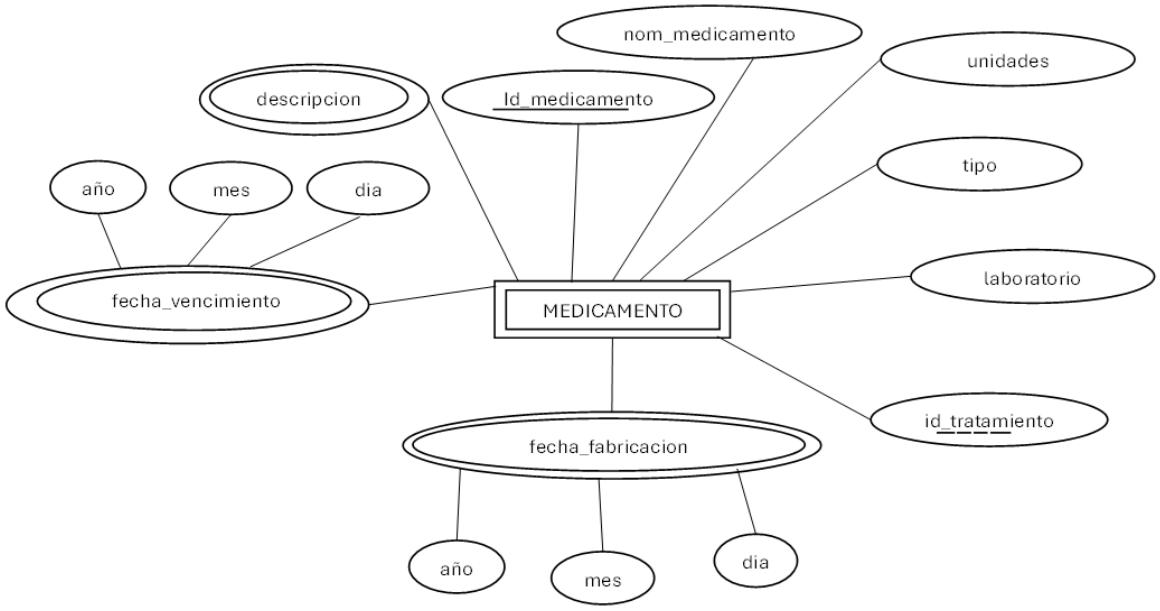

Id_historia_clinica
id_animal
id_cuidador
fecha_inicio_historia
HISTORIA CLINICA
id_veterinario
año
mes
dia
fecha_fin_historia
año
mes
dia



1.5.2. Entidades secundarias débiles

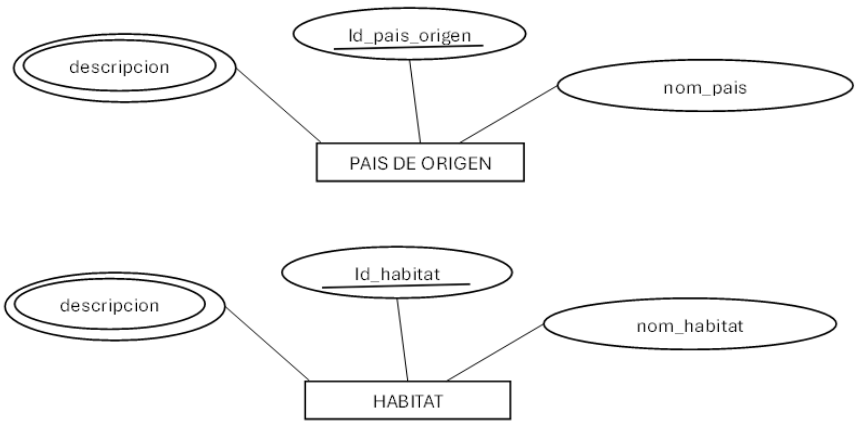
Como sabemos la entidad bitácora es una entidad secundaria débil. Para esta entidad tenemos como atributo derivado de otra entidad al identificador de la entidad usuario. Los atributos multivalorados son fecha de acción, hora de acción, hora de salida y la descripción de la acción.

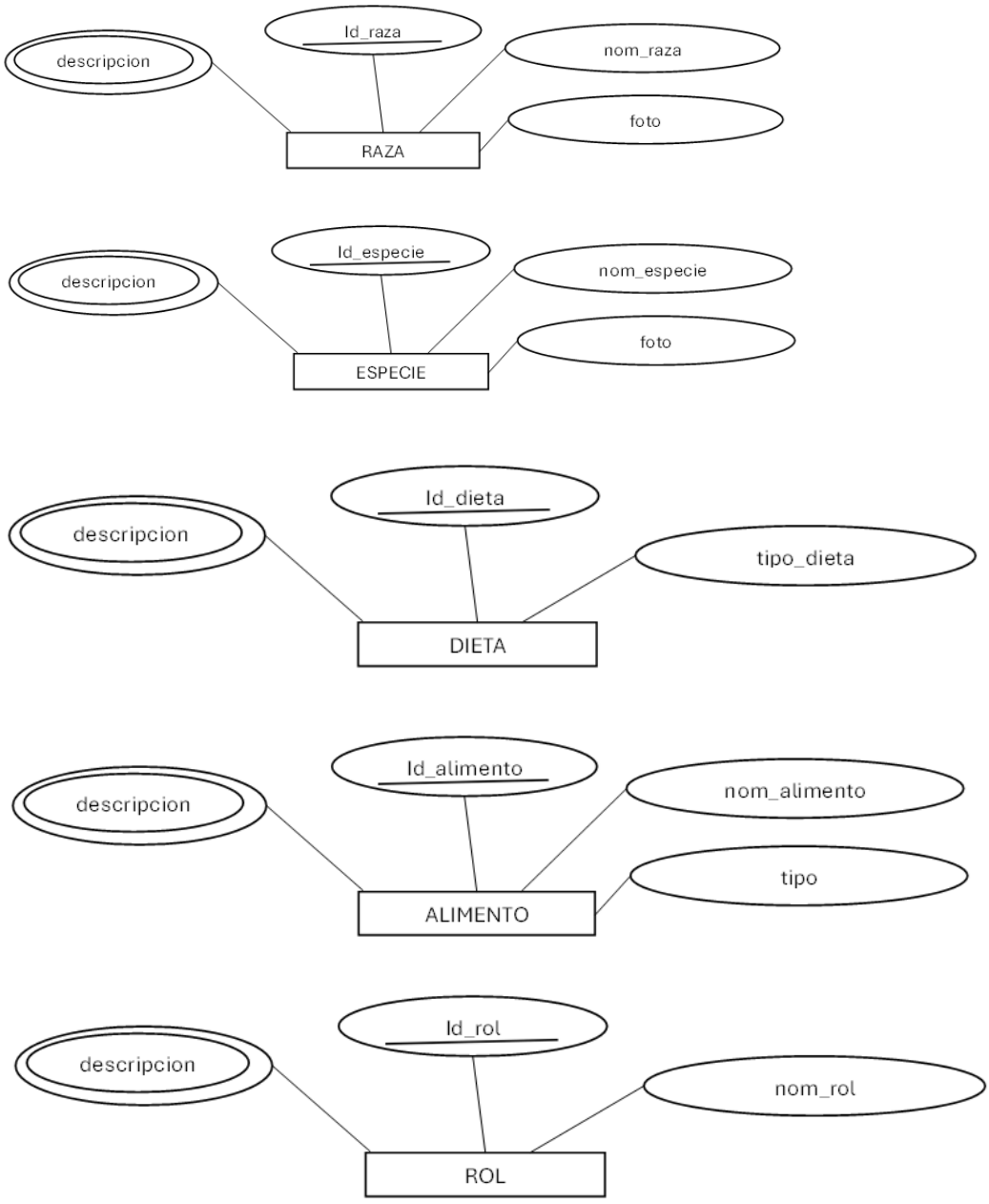

horas
minutos
hora_accion
descripcion
Id_bitacora
horas
minutos
hora_salida
BITACORA
id_usuario
fecha_accion
año
mes
dia
fecha_detalle_historia
Id_detalle_historia
id_historia_clinica
año
mes
dia
DETALLE HISTORIA CLINICA
descripción_detalle_historia

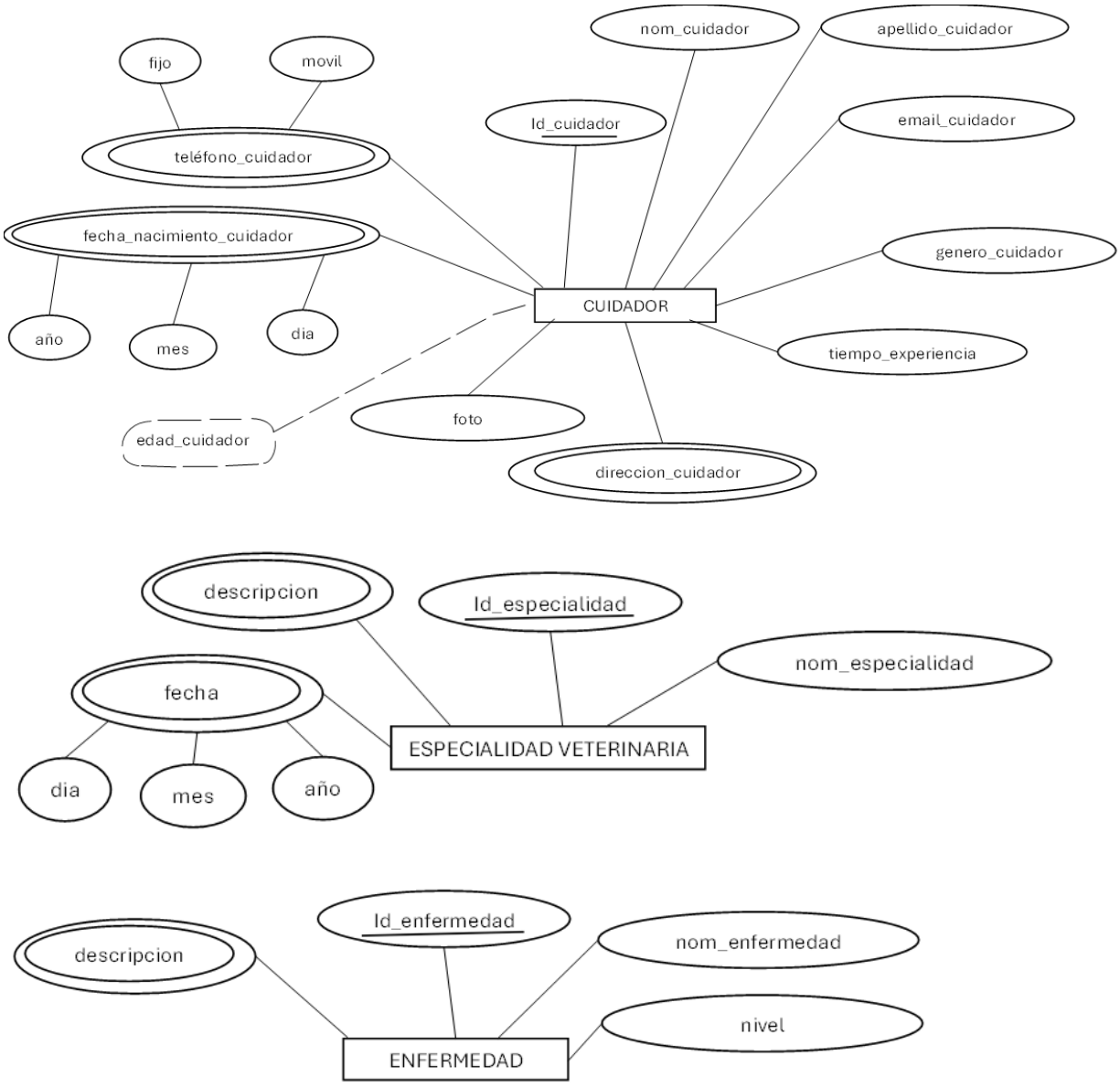


1.5.3. Entidades primarias fuertes

Como sabemos la entidad país de origen es una entidad primaria fuerte. Como atributo simple y con línea inferior, que lo identifica como un atributo con valores únicos y está definido como identificador del país de origen. Como atributo simple está el nombre del país y como atributo multivalorado tenemos la descripción del país.

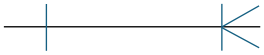
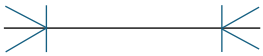



descripcion
Id_raza
nom_raza
foto
RAZA
descripcion
Id_especie
nom_especie
foto
ESPECIE
descripcion
Id_dieta
tipo_dieta
DIETA
descripcion
Id_alimento
nom_alimento
tipo
ALIMENTO
descripcion
Id_rol
nom_rol
ROL

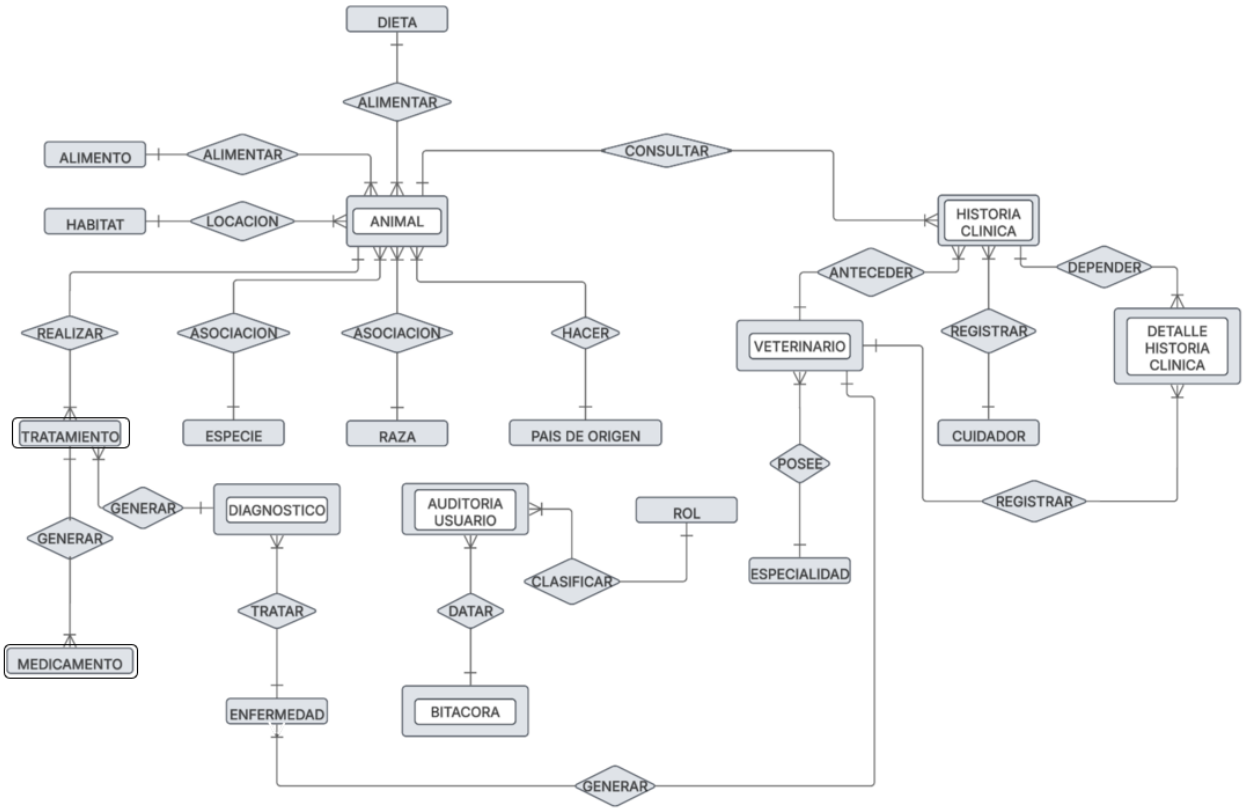


Tipos de relaciones:

Uno a uno		Relaciona dos entidades en función de un único valor común, como un numero de serie o un numero de identificación, una relación uno a uno es un vínculo entre la información de dos

		entidades, donde cada registro en cada entidad solo aparece una vez.
Uno a muchos		En una relación de uno a muchos, un registro de una entidad se puede asociar a uno o varios registros de otra entidad.
Muchos a muchos		Una relación de muchos a muchos se produce cuando varios registros de una entidad se asocian a varios registros de otra entidad.

1.5.3. MER



1.6. CLAVES – INDICES – LLAVES

Se define como llame o índice, dentro de la base de datos, al atributo mas importante dentro de una entidad, este atributo permite usar componentes importantes de los datos de una entidad en una estructura binaria para mejorar la capacidad de búsqueda. Cada registro de datos en la tabla debe estar asociado con datos en la llave.

<table>
<tr><td rowspan="4">Principales</td><td>Primaria</td><td>PK</td><td rowspan="4">Es el atributo más relevante dentro de la entidad, este tipo de clave permite definir el tipo y rol de la entidad dentro de la base de datos.
Su valor debe ser único.
No debe contener valores nulos.
Es el atributo más importante dentro de la entidad.</td></tr>
<tr><td>Única</td><td>UK</td></tr>
<tr><td>Candidata</td><td>CK</td></tr>
<tr><td>Compuesta</td><td>Compuesta</td></tr>
<tr><td>Secundarias</td><td>Foráneas</td><td>FK</td><td>La llave foránea es el atributo que permitirá identificar una entidad débil, esta llave se crea a partir de la llave primaria de la entidad que va a relacionar a la entidad que contiene la clave en mención.
Debe tener el mismo nombre del atributo de la entidad fuerte que vaya a relacionar la entidad que contiene la llave foránea.
Tener el mismo tipo de dato que tiene el atributo de la entidad fuerte que va a relacionar la entidad que la contiene.
Tener la misma longitud de la llave primaria de la entidad fuerte que vaya a relacionar a la entidad que contiene la FK.</td></tr>
</table>

Se define como llave compuesta o superclave, a la unión de dos o mas datos que provienen de dos o mas atributos, ya sea de la misma entidad o de otras entidades.

1.7. TIPOS DE DATOS

1. **Numéricos:**
   - INT, BIGINT, SMALLINT, TINYINT → Números enteros
   - FLOAT, REAL, DOUBLE → Números de coma flotante (menos precisos)
2. **Cadenas de texto:**
   - CHAR, VARCHAR → Textos de longitud fija o variable
   - TEXT, CLOB → Texto largo (depende del motor)
3. **Fechas y tiempos:**
   - DATE → Fecha
   - TIME → Tiempo
   - DATETIME, TIMESTAMP → Fecha y hora
4. **Booleanos:**
   - BOOLEAN (en MySQL se usa TINYINT(1) (TY), en SQL server BIT)
5. **Datos binarios y JSON:**
   - BLOB → Datos binarios grandes (imágenes, archivos)
   - JSON → Para almacenar objetos en formato JSON (PostgreSQL, MySQL)

1.8. TABLAS CON LA APLICACIÓN DEL TIPO DE DATO, LONGITUD, LLAVE, AUTO INCREMENTO, NULOS Y DESCRIPCION

ANIMAL						
Nombre	**Tipo de dato**	**Longitud**	**Key**	**Auto incremento**	**Nulos**	**Descripcion**
id_animal	INT		PK	SI	NO	Identificador unico para cada animal
nom_animal	VARCHAR	20		NO	NO	Nombre del animal
edad_animal	INT			NO	NO	Edad del animal
sexo_animal	VARCHAR	1		NO	NO	Sexo del animal
fecha_nacimiento_animal	DATE			NO	NO	Fecha de nacimiento del animal
id_especie	INT		FK	NO	NO	Identificador unico para cada especie
id_raza	INT		FK	NO	NO	Identificador unico para cada raza
id_pais_origen	INT		FK	NO	NO	Identificador unico para cada pais
id_habitat	INT		FK	NO	NO	Identificador unico para cada habitat
id_alimento	INT		FK	NO	NO	Identificador unico para cada alimento
id_dieta	INT		FK	NO	NO	Identificador unico para cada dieta
foto	BLOB			NO	NO	Foto del animal
pelaje	VARCHAR	20		NO	NO	Pelaje del animal
altura	INT			NO	NO	Altura del animal
peso	INT			NO	NO	Peso del animal
estado_sanitario_animal	VARCHAR	1		NO	NO	Estado sanitario del animal

VETERINARIO						
Nombre	**Tipo de dato**	**Longitud**	**Key**	**Auto incremento**	**Nulos**	**Descripcion**
id_veterinario	INT		PK	SI	NO	Identificadaro unico para cada veterinario
nom_veterinario	VARCHAR	20		NO	NO	Nombres del veterinario
apellido_veterinario	VARCHAR	20		NO	NO	Apellidos del veterinario
id_especialidad	INT		FK	NO	NO	Identificadaro unico para cada especialidad
fecha_nacimiento_veterinario	DATE			NO	NO	Fecha de nacimiento del animal
edad_veterinario	INT	2		NO	NO	Edad del veterinario
genero_veterinario	VARCHAR	1		NO	NO	Genero del veterinario
telefono_veterinario	INT	10		NO	NO	Telefono del veterinario
email_veterinario	VARCHAR	30		NO	NO	Email del veterinario
foto	BLOB			NO	NO	Foto del veterinario
direccion_veterinario	VARCHAR	30		NO	NO	Direccion del veterinario

USUARIO						
Nombre	**Tipo de dato**	**Longitud**	**Key**	**Auto incremento**	**Nulos**	**Descripcion**
id_usuario	INT		PK	SI	NO	Identificador unico para cada usuario
apellidos_usuario	VARCHAR	20		NO	NO	Apellidos del usuario
edad_usuario	INT	2		NO	NO	Edad del usuario
genero_usuario	VARCHAR	1		NO	NO	Genero del usuario
fecha_nacimiento_usuario	DATE			NO	NO	Fecha de nacimiento del usuario
telefono_usuario	INT	10		NO	NO	Telefono del usuario
descripcion	VARCHAR	300		NO	NO	Descripcion del usuario
foto				NO	NO	Foto del usuario
nom_usuario	VARCHAR	20		NO	NO	Nombre del usuario
id_rol	INT		FK	NO	NO	Identificador unico del rol para cada usuario

HISTORIA CLINICA						
Nombre	**Tipo de dato**	**Longitud**	**Key**	**Auto incremento**	**Nulos**	**Descripcion**
id_historia_clinica	INT		PK	SI	NO	Identificador unico para cada historia clinica
id_animal	INT		FK	NO	NO	Identificador unico del animal para la historia clinica
id_veterinario	INT		FK	NO	NO	Identificador unico del veterinario para la historia clinica
id_cuidador	INT		FK	NO	NO	Identificador unico del cuidador para la historia clinica
fecha_inicio_historia	DATE			NO	NO	Fecha de inicio para la historia clinica
fecha_fin_historia	DATE			NO	NO	Fecha de la ultima actualizacion de la historia clinica

TRATAMIENTO						
Nombre	**Tipo de dato**	**Longitud**	**Key**	**Auto incremento**	**Nulos**	**Descripcion**
id_tratamiento	INT		PK	SI	NO	Identificador unico para cada tratamiento
fecha_inicio_tratamiento	DATE			NO	NO	Fecha de inicio del tratamiento
fecha_fin_tratamiento	DATE			NO	NO	Fecha de finalizacion del tratamiento
descripcion_tratamiento	VARCHAR	400		NO	NO	Descripcion del tratamiento
id_animal	INT		FK	NO	NO	Identificador unico del animal para el tratamiento

DIAGNOSTICO						
Nombre	**Tipo de dato**	**Longitud**	**Key**	**Auto incremento**	**Nulos**	**Descripcion**
id_diagnostico	INT		PK	SI	NO	Identificador unico para cada diagnostico
fecha_diagnostico	DATE			NO	NO	Fecha del diagnostico
descripcion_diagnostico	VARCHAR	400		NO	NO	Descripcion del diagnostico
id_enfermedad	INT		FK	NO	NO	Identificador unico de la enfermedad para el diagnostico

DETALLE DE HISTORIA CLINICA						
Nombre	**Tipo de dato**	**Longitud**	**Key**	**Auto incremento**	**Nulos**	**Descripcion**
id_detalle_historia	INT		PK	SI	NO	Identificador unico para el detalle de cada historia clinica
descripcion_detalle_historia	VARCHAR	400		NO	NO	Descripcion del detalle de la historia clinica
fecha_detalle_historia	DATE			NO	NO	Fecha del detalle de la historia clinica
id_historia_clinica	INT		FK	NO	NO	Identificador unico de la historia clinica para el detalle de cada historia clinica

BITACORA						
Nombre	**Tipo de dato**	**Longitud**	**Key**	**Auto incremento**	**Nulos**	**Descripcion**
id_bitacora	INT		PK	SI	NO	Identificador unico para cada bitacora
id_usuario	INT		FK	NO	NO	Identificador unico del usuario para cada bitacora
descripcion_accion	VARCHAR	400		NO	NO	Descripcion de la accion
fecha_accion	DATE			NO	NO	Fecha de la accion
hora_salida	TIME			NO	NO	Hora de la salida
hora_accion	TIME			NO	NO	Hora de la accion

MEDICAMENTO						
Nombre	**Tipo de dato**	**Longitud**	**Key**	**Auto incremento**	**Nulos**	**Descripcion**
id_medicamento	INT		PK	SI	NO	Identificador unico del medicamento
nom_medicamento	VARCHAR	20		NO	NO	Nombre del medicamento
unidades	VARCHAR	30		NO	NO	
fecha_fabricacion	DATE			NO	NO	Fecha de la fabricacion del medicamento
fecha_vencimimiento	DATE			NO	NO	Fecha de vencimiento del medicamento
tipo_medicamento	VARCHAR	20		NO	NO	Tipo de medicamento
laboratorio	VARCHAR	20		NO	NO	Laboratio del medicamento
descripcion_medicamento	VARCHAR	400		NO	NO	Descripcion del medicamento
id_tratamiento	INT		FK	SI	NO	Identificador unico para cada tratamiento

PAIS DE ORIGEN						
Nombre	**Tipo de dato**	**Longitud**	**Key**	**Auto incremento**	**Nulos**	**Descripcion**
id_pais_origen	INT		PK	SI	NO	Identificador unico para cada pais de origen
descripcion	VARCHAR	300		NO	NO	Descripcion del pais
nom_pais	VARCHAR	25		NO	NO	Nombre del pais

HABITAT						
Nombre	**Tipo de dato**	**Longitud**	**Key**	**Auto incremento**	**Nulos**	**Descripcion**
id_habitat	INT		PK	SI	NO	Identificador unico para cada habitat
descripcion	VARCHAR	300		NO	NO	Descripcion del habitat
nombre_habitat	VARCHAR	30		NO	NO	Nombre del habitat

RAZA						
Nombre	**Tipo de dato**	**Longitud**	**Key**	**Auto incremento**	**Nulos**	**Descripcion**
id_raza	INT		PK	SI	NO	Identificador unico para cada raza
foto	BLOB			NO	NO	Foto de la raza
descripcion	VARCHAR	300		NO	NO	Descripcion de la raza
nom_raza	VARCHAR	30		NO	NO	Nombre de la raza

ESPECIE						
Nombre	**Tipo de dato**	**Longitud**	**Key**	**Auto incremento**	**Nulos**	**Descripcion**
id_especie	INT		PK	SI	NO	Identificador unico para cada especie
foto	BLOB			NO	NO	Foto de la especie
descripcion	VARCHAR	300		NO	NO	Descripcion de la especie
nom_especie	VARCHAR	20		NO	NO	Nombre de la especie

DIETA						
Nombre	**Tipo de dato**	**Longitud**	**Key**	**Auto incremento**	**Nulos**	**Descripcion**
id_dieta	INT		PK	SI	NO	Identificador unico para cada dieta
descripcion	VARCHAR	400		NO	NO	Descripcion de la dieta
tipo_dieta	VARCHAR	40		NO	NO	Tipo de alimento

ALIMENTO						
Nombre	**Tipo de dato**	**Longitud**	**Key**	**Auto incremento**	**Nulos**	**Descripcion**
id_alimento	INT		PK	SI	NO	Identificador unico para cada alimento
tipo	VARCHAR	20		NO	NO	Tipo de alimento (vegtales, fruta, proteina, etc.)
descripcion	VARCHAR	300		NO	NO	Descripcion del alimento
nom_alimento	VARCHAR	30		NO	NO	Nombre del alimento

ROL						
Nombre	**Tipo de dato**	**Longitud**	**Key**	**Auto incremento**	**Nulos**	**Descripcion**
id_rol	INT		PK	SI	NO	Identificador unico del rol
descripcion	VARCHAR	400		NO	NO	Descripcion del rol
nom_rol	VARCHAR	25		NO	NO	Nombre del rol

CUIDADOR						
Nombre	**Tipo de dato**	**Longitud**	**Key**	**Auto incremento**	**Nulos**	**Descripcion**
id_cuidador	INT		PK	SI	NO	Identificador unico para cada cuidador
nom_cuidador	VARCHAR	20		NO	NO	Nombre del cuidador
apellidos_cuidador	VARCHAR	20		NO	NO	Apellidos del cuidador
direccion_cuidador	VARCHAR	30		NO	NO	Direccion del cuidador
fecha_nacimiento_cuidador	DATE			NO	NO	Fecha de nacimiento del cuidador
edad_cuidador	INT	2		NO	NO	Edad del cuidador
genero_cuidador	VARCHAR	1		NO	NO	Genero del cuidador
telefono_cuidador	INT	10		NO	NO	Telefono del cuidador
tiempo_experiencia	INT			NO	NO	Tiempo de experiencia del cuidador
foto	BLOB			NO	NO	Foto del cuidador
email_cuidador	VARCHAR	30		NO	NO	Email del cuidador

ESPECIALIDAD VETERINARIA						
Nombre	**Tipo de dato**	**Longitud**	**Key**	**Auto incremento**	**Nulos**	**Descripcion**
id_especialidad	INT		PK	SI	NO	Identificador unico para cada especialidad
nom_especialidad	VARCHAR	20		NO	NO	Nombre de la especialidad
fecha	DATE			NO	NO	Fecha en la que el veterinario se especializo
descripcion	VARCHAR	300		NO	NO	Descripcion de la especialidad

ENFERMEDAD						
Nombre	**Tipo de dato**	**Longitud**	**Key**	**Auto incremento**	**Nulos**	**Descripcion**
id_enfermedad	INT		PK	SI	NO	Identificador unico para cada enfermedad
nom_enfermedad	VARCHAR	20		NO	NO	Nombre de la enfermedad
nivel	VARCHAR	20		NO	NO	Nivel de la enfermedad
descripcion	VARCHAR	400		NO	NO	Descripcion de la enfermedad

Como podemos evidenciar, cada tabla (entidad) tiene su llave primaria (PK), ya que esto nos permite la manipulación de los datos y podemos aplicar las operaciones para crear tablas temporales y las operaciones de la teoría de conjuntos.

Las relaciones están sujetas al concepto de keys, llaves, índices, claves.

1.9. NORMALIZACION

La normalización en base de datos es el proceso de organizar los datos para reducir la redundancia y mejorar la integridad de la base de datos. Esto se logra creando tablas y estableciendo relaciones entre ellas mediante reglas llamadas “formas normales”.

1. **Primera norma formal (1FN):** Elimina los grupos repetidos de datos dentro de una tabla, creando tablas separadas para cada grupo.
2. **Segunda forma normal (2FN):** Elimina las dependencias parciales, asegurando que todos los atributos no clave dependan de la clave primaria.
3. **Tercera forma normal (3FN):** Elimina las dependencias transitivas, asegurando que todos los atributos no clave dependan directamente de la clave primaria y no de otros atributos no clave.

1.9.1. Aplicación:

La aplicación de la normalización la realizamos en hojas de Excel, donde tenemos las tres formas normales. En la primera forma normal tenemos tablas (entidades) con sus atributos.

En la segunda forma normal tenemos las tablas (entidades) con su tipo de dato, longitud, llave, auto incremento, nulos y su descripción. Este proceso es clave para eliminar las dependencias para evidenciar los atributos que de dependen de la llave primaria (PK).

Para la tercera forma normal tenemos el proceso que nos permite eliminar dependencias a partir de sus relaciones, asociando las tablas con datos reales.

Además de estas tres formas normales agregamos la forma normal 0 (**0FN**), donde encontramos el levantamiento de datos, este fue el mismo procedimiento que se describió desde que recibimos los datos y se puede evidenciar al comienzo del documento en el título “**LEVANTAMIENDO DE DATOS**”.

Seguido a añadimos una hoja con **EyA,** donde encontramos los diagramas de Entidad-Atributo, este fue el proceso que se describió en el titulo “**MODELO ENTIDAD RELACION (MER)**”.

Por último, añadimos la hoja **MER**, que contiene el diagrama del Modelo Entidad Relación, el diagrama de este proceso se describió en el título “**MODELO ENTIDAD RELACION**”.

2. SEGUNDO CORTE

2.1. CONCEPTOS

2.1.1. ¿Que son los paradigmas en la programación?

Los paradigmas de programación son enfoques o estilos que determina como se estructura y organiza el código.

A continuación, podemos seguir el paradigma para las bases de datos:

2.1.2. ¿Qué es SQL, para que sirve y cómo funciona?

SQL (Structured Query Language) es un lenguaje de computación estandarizado para gestionar y consultar base de datos relacionales. Sirve para crear, modificar, consultar y administrar datos dentro de una base de datos. Funciona mediante comando que permiten realizar operaciones como insertar, actualizar, eliminar y seleccionar datos.

2.1.3. ¿Para qué sirve SQL?

- **Creación y modificación de base de datos:** SQL permite definir la estructura de las bases de datos, creando tablas, definiendo columnas y estableciendo relaciones entre ellas.
- **Manipulación de datos:** SQL permite realizar operaciones CRUD (créate, read, update, delete) sobre los datos almacenados en las tablas.
- **Consultas:** SQL permite obtener información específica de la base de datos, seleccionando y filtrando datos según ciertos criterios.
- **Administración de usuarios y permisos:** SQL permite controlar el acceso a la base de datos, definiendo quien puede realizar que operaciones.

2.1.4. ¿Cómo funciona SQL?

SQL funciona mediante comandos que se ejecutan en un sistema de gestión de bases de datos relacionales. Los comandos SQL permiten:

- **Definir la estructura de la base de datos:** se utilizan comandos como CREATE TABLE para crear tablas, ALTER TABLE para modificar la estructura de las tablas y DROP TABLE para eliminar tablas.
- **Insertar datos:** se utiliza el comando INSERT INTO para agregar nuevos registros a una tabla.
- **Actualizar datos:** se utiliza el comando UPDATE para modificar los valores de registros existentes.
- **Eliminar datos:** se utiliza el comando DELETE FROM para eliminar registros de una tabla.
- **Consultar datos:** se utiliza el comando SELECT para obtener datos de una nueva o varias tablas.

2.2. COMANDOS SQL

DDL:

- CREATE
- ALTER
- DROP
- TRUNCATE

DML:

- INSERT
- UPDATE
- DELETE

DCL:

- GRANT
- REVOKE

TCL:

- COMMIT
- ROLLBACK
- SAVE
- POINT

DQL:

- SELECT

Para la creación de la base de datos podemos aplicar los siguientes comandos para modificar, mostrar, crear y eliminar tablas y sus atributos:

SHOW DATABASES; → Imprime las bases de datos que existen en la memoria.

CREATE DATABASE nomDatabase; → Crea una base de datos y la guarda en la memoria.

USE nomDatabase; → Nos permite ingresar a la base de datos para realizar las modificaciones necesarias.

CREATE TABLE nomTabla(atributo INT PRIMARY KEY AUTO_INCREMENT NOT NULL, atributo VARCHAR(255), atributo VARCHAR(255)); → Crea una tabla para una entidad fuerte en la base de datos.

CREATE TABLE nomTabla(atributo INT PRIMARY KEY AUTO_INCREMENT NOT NULL, atributo VARCHAR(255), id_ atributo INT,
CONSTRAINT fk_id_atributo FOREIGN KEY(id:atributo)

REFERENCES(id_atributo)); → Crea una tabla para una entidad secundaria en la base de datos.

SHOE TABLES; → Imprime las tablas que existen en la base de datos.

DESC nomTabla; → Imprime la tabla seleccionada y sus atributos.

ALTER TABLE nomTabla ADD campo VARCHAR(255); → Permite adicionar un atributo a una tabla.

ALTER TABLE nomTabla ADD campo VARCHAR(255) AFTER campoExistente; → Permite adicionar un atributo a una tabla después de un atributo ya creado en la tabla.

ALTER TABLE nomTabla DROP campo; → Elimina un atributo de la tabla.

<table>
<tr><th colspan="4">PROCEDIMENTALES</th></tr>
<tr><td rowspan="4">CREATE</td><td colspan="2">DATABASE</td><td rowspan="4">INSERT</td><td>SIMPLE</td></tr>
<tr><td colspan="2">TABLES</td><td rowspan="3">MULTIPLE</td></tr>
<tr><td colspan="2">INDEX</td></tr>
<tr><td colspan="2">VIEW</td></tr>
<tr><td rowspan="2">ALTER</td><td rowspan="2">TABLE</td><td>ADD</td><td rowspan="2">UPDATE</td><td rowspan="2">SET</td></tr>
<tr><td>DROP</td></tr>
<tr><td rowspan="2">DROP</td><td colspan="2">DATABASE</td><td rowspan="2">DELETE</td><td rowspan="2">FROM … WHERE</td></tr>
<tr><td colspan="2">TABLES</td></tr>
</table>

2.3. INTEGRIDAD REFERENCIAL

Es un conjunto de reglas que garantizan la consistencia y la precisión de los datos entre tablas relacionadas. Esto se logra mediante el uso de claves foráneas, que permiten establecer relaciones entre tablas y evitar que se creen registros huérfanos.

Restricción referencial: reglas que se aplican a las claves foráneas para asegurar que las referencias sean validas.

Tabla principal (tabla referenciada): la tabla que contiene la clave primaria (PK) a la que hace referencia.

Tabla secundaria (tabla que referencia): la tabla que contiene la clave foránea que hace referencia a la clave primaria.

Precisión de los datos: garantiza que las relaciones entre datos sean consistentes y validas, evitando errores.

Confiabilidad de los datos: permite mantener la integridad de los datos incluso durante actualizaciones o eliminaciones.

Reducción de errores: ayuda a evitar errores humanos al aplicar restricciones de forma automática.

Mantenimiento de la coherencia: garantiza que los datos sean coherentes entre tablas relacionadas.

Cuando se intenta modificar o eliminar datos que afectarían la integridad referencial, la base de datos puede:

1. **RESTRICT:** impide la operación que violaría la integridad referencial.
2. **CASCADE:** propaga la operación a los registros relacionados.
3. **SET NULL:** establece NULL en la llave foránea de los registros relacionados.
4. **SET DEFAULT:** establece un valor por defecto en la FK.

2.4. Aplicación CRUD:

Sintaxis:

Primeramente, ingresamos a la base de datos, digitando en el prompt el comando **# mysql -u root -h localhost -p.**

Seguido a esto podemos empezar a hacer operaciones. Nuestra primera opción es ver las bases de datos que existen en nuestro almacenamiento:

Setting environment for using XAMPP for Windows.

luisd@LUIS c:\xampp

mysql -u root -h localhost -p

Enter password:

Welcome to the MariaDB monitor. Commands end with ; or \g.

Your MariaDB connection id is 8

Server version: 10.4.32-MariaDB mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> show databases;

+--------------------+

| Database |

+--------------------+

| information_schema |

| mysql |

| odbcorganigrama |

| odbparcial2 |

| performance_schema |

| phpmyadmin |

| test |

+--------------------+

7 rows in set (0.001 sec)

A continuación, utilizamos **CREATE DATABASE** para crear nuestra base de datos Ukumari:

MariaDB [(none)]> CREATE DATABASE odbcUkumari;

Query OK, 1 row affected (0.003 sec)

MariaDB [(none)]> SHOW DATABASES;

+--------------------+

| Database |

+--------------------+

| information_schema |

| mysql |

| odbcorganigrama |

| odbcukumari |

| odbparcial2 |

| performance_schema |

| phpmyadmin |

| test |

+--------------------+

8 rows in set (0.001 sec)

Accedemos a nuestra base de datos a través del comando **USE odbcUkumari;**:

MariaDB [(none)]> USE odbcUkumari;

Database changed

MariaDB [odbcUkumari]>

2.4.1. Entidades primarias

Una vez estemos usando nuestra base de datos, empezamos a crear las tablas del modelamiento de los datos. Con el comando **CREATE TABLE nomTabla(atributo TIPO_DATO KEY AUTO_INCREMENT NOT NULL/NULL, atributo …);**:

Debemos recordar que debemos seguir un orden para la creación de la base de datos en el motor BD. Iniciamos con las entidades **Primarias Fuertes**:

Entidad cuidador

MariaDB [odbcUkumari]> CREATE TABLE tCuidador(id_cuidador INT PRIMARY KEY AUTO_INCREMENT NOT NULL,

-> nom_cuidador VARCHAR(20),apellido_cuidador VARCHAR(20), direccion_cuidador VARCHAR(30), foto BLOB, fecha_nacimiento_cuidador DATE, edad_cuidador INT(2), genero_cuidador VARCHAR(1), telefono_cuidador INT(10), email_cuidador VARCHAR(30), tiempo_expericencia INT);

Query OK, 0 rows affected (0.044 sec)

MariaDB [odbcUkumari]> desc tCuidador;

+--------------------------+-------------+------+-----+---------+----------------+

| Field | Type | Null | Key | Default | Extra |

+--------------------------+-------------+------+-----+---------+----------------+

| id_cuidador | int(11) | NO | PRI | NULL | auto_increment |

nom_cuidador	varchar(20)	YES		NULL	
apellido_cuidador	varchar(20)	YES		NULL	
direccion_cuidador	varchar(30)	YES		NULL	
foto	blob	YES		NULL	
fecha_nacimiento_cuidador	date	YES		NULL	
edad_cuidador	int(2)	YES		NULL	
genero_cuidador	varchar(1)	YES		NULL	
telefono_cuidador	int(10)	YES		NULL	
email_cuidador	varchar(30)	YES		NULL	
tiempo_expericencia	int(11)	YES		NULL	

+---+---+---+---+---+---+

11 rows in set (0.009 sec)

Entidad enfermedad

MariaDB [odbcUkumari]> CREATE TABLE tEnfermedad(id_enfermedad INT PRIMARY KEY AUTO_INCREMENT NOT NULL,

-> nom_enfermedad VARCHAR(20), descripcion VARCHAR(300), nivel VARCHAR(20));

Query OK, 0 rows affected (0.011 sec)

MariaDB [odbcUkumari]> desc tEnfermedad;

+---+---+---+---+---+---+

| Field | Type | Null | Key | Default | Extra |

+---+---+---+---+---+---+

id_enfermedad	int(11)	NO	PRI	NULL	auto_increment
nom_enfermedad	varchar(20)	YES		NULL	
descripcion	varchar(300)	YES		NULL	
nivel	varchar(20)	YES		NULL	

```
+---------------+-------------+------+-----+---------+---------------+
```

4 rows in set (0.019 sec)

Entidad especialidad veterinaria

```
MariaDB [odbcUkumari]> CREATE TABLE
tEspecialidadVeterinaria(id_especialidad INT PRIMARY KEY AUTO_INCREMENT
NOT NULL,
    -> nom_especialidad VARCHAR(20), descripcion VARCHAR(300), fecha DATE);
```

Query OK, 0 rows affected (0.010 sec)

```
MariaDB [odbcUkumari]> desc tEspecialidadVeterinaria;
+-----------------+-------------+------+-----+---------+----------------+
| Field           | Type        | Null | Key | Default | Extra          |
+-----------------+-------------+------+-----+---------+----------------+
| id_especialidad | int(11)     | NO   | PRI | NULL    | auto_increment |
| nom_especialidad | varchar(20) | YES  |     | NULL    |                |
| descripcion     | varchar(300) | YES  |     | NULL    |                |
| fecha           | date        | YES  |     | NULL    |                |
+-----------------+-------------+------+-----+---------+----------------+
```

4 rows in set (0.010 sec)

Entidad rol

```
MariaDB [odbcUkumari]> CREATE TABLE tRol(id_rol INT PRIMARY KEY
AUTO_INCREMENT NOT NULL,
    -> decripcion VARCHAR(300), nom_rol VARCHAR(25));
```

Query OK, 0 rows affected (0.009 sec)

```
MariaDB [odbcUkumari]> desc tRol;
+-----------+-------------+------+-----+---------+---------------+
| Field     | Type        | Null | Key | Default | Extra         |
+-----------+-------------+------+-----+---------+---------------+
| id_rol    | int(11)     | NO   | PRI | NULL    | auto_increment |
| decripcion | varchar(300) | YES  |     | NULL    |               |
| nom_rol   | varchar(25)  | YES  |     | NULL    |               |
+-----------+-------------+------+-----+---------+---------------+
3 rows in set (0.018 sec)
```

Entidad alimento

```
MariaDB [odbcUkumari]> CREATE TABLE tAlimento(id_alimento INT PRIMARY
KEY AUTO_INCREMENT NOT NULL,
    -> nom_alimento VARCHAR(30), tipo VARCHAR(50), descripcion
VARCHAR(300));
Query OK, 0 rows affected (0.013 sec)

MariaDB [odbcUkumari]> desc tAlimento;
+-------------+-------------+------+-----+---------+---------------+
| Field       | Type        | Null | Key | Default | Extra         |
+-------------+-------------+------+-----+---------+---------------+
| id_alimento | int(11)     | NO   | PRI | NULL    | auto_increment |
| nom_alimento | varchar(30) | YES  |     | NULL    |               |
| tipo        | varchar(50) | YES  |     | NULL    |               |
| descripcion | varchar(300) | YES  |     | NULL    |               |
+-------------+-------------+------+-----+---------+---------------+
4 rows in set (0.009 sec)
```

Entidad dieta

```
MariaDB [odbcUkumari]> CREATE TABLE tDieta(id_dieta INT PRIMARY KEY
AUTO_INCREMENT NOT NULL,
    -> tipo_dieta VARCHAR(40), descripcion VARCHAR(300));
Query OK, 0 rows affected (0.010 sec)

MariaDB [odbcUkumari]> desc tDieta;
+-------------+--------------+------+-----+---------+----------------+
| Field       | Type         | Null | Key | Default | Extra          |
+-------------+--------------+------+-----+---------+----------------+
| id_dieta    | int(11)      | NO   | PRI | NULL    | auto_increment |
| tipo_dieta  | varchar(40)  | YES  |     | NULL    |                |
| descripcion | varchar(300) | YES  |     | NULL    |                |
+-------------+--------------+------+-----+---------+----------------+
3 rows in set (0.011 sec)
```

Entidad especie

```
MariaDB [odbcUkumari]> CREATE TABLE tEspecie(id_especie INT PRIMARY
KEY AUTO_INCREMENT NOT NULL,
    -> nom_especie VARCHAR(20), foto BLOB, descripcion VARCHAR(300));
Query OK, 0 rows affected (0.011 sec)

MariaDB [odbcUkumari]> desc tEspecie;
+-------------+--------------+------+-----+---------+----------------+
| Field       | Type         | Null | Key | Default | Extra          |
+-------------+--------------+------+-----+---------+----------------+
| id_especie  | int(11)      | NO   | PRI | NULL    | auto_increment |
```

| nom_especie | varchar(20) | YES | | NULL | |

| foto | blob | YES | | NULL | |

| descripcion | varchar(300) | YES | | NULL | |

+-------------+--------------+------+-----+---------+---------------+

4 rows in set (0.009 sec)

Entidad raza

MariaDB [odbcUkumari]> CREATE TABLE tRaza(id_raza INT PRIMARY KEY AUTO_INCREMENT NOT NULL,

    -> foto BLOB, nom_raza VARCHAR(30), descripcion VARCHAR(300));

Query OK, 0 rows affected (0.009 sec)

MariaDB [odbcUkumari]> desc tRaza;

+-------------+--------------+------+-----+---------+---------------+

| Field | Type | Null | Key | Default | Extra |

+-------------+--------------+------+-----+---------+---------------+

| id_raza | int(11) | NO | PRI | NULL | auto_increment |

| foto | blob | YES | | NULL | |

| nom_raza | varchar(30) | YES | | NULL | |

| descripcion | varchar(300) | YES | | NULL | |

+-------------+--------------+------+-----+---------+---------------+

4 rows in set (0.008 sec)

Entidad hábitat

MariaDB [odbcUkumari]> CREATE TABLE tHabitat(id_habitat INT PRIMARY KEY AUTO_INCREMENT NOT NULL,

    -> nom_habitat VARCHAR(30), descripcion VARCHAR(300));

Query OK, 0 rows affected (0.010 sec)

MariaDB [odbcUkumari]> desc tHabitat;

```
+-------------+-------------+------+-----+---------+----------------+
| Field       | Type        | Null | Key | Default | Extra          |
+-------------+-------------+------+-----+---------+----------------+
| id_habitat  | int(11)     | NO   | PRI | NULL    | auto_increment |
| nom_habitat | varchar(30) | YES  |     | NULL    |                |
| descripcion | varchar(300) | YES  |     | NULL    |                |
+-------------+-------------+------+-----+---------+----------------+
```

3 rows in set (0.009 sec)

Entidad país de origen

MariaDB [odbcUkumari]> CREATE TABLE tPaisDeOrigen(id_pais_origen INT PRIMARY KEY AUTO_INCREMENT NOT NULL,

-> nom_pais_origen VARCHAR(25), descripcion VARCHAR(300));

Query OK, 0 rows affected (0.009 sec)

MariaDB [odbcUkumari]> desc tPaisDeOrigen;

```
+----------------+-------------+------+-----+---------+----------------+
| Field          | Type        | Null | Key | Default | Extra          |
+----------------+-------------+------+-----+---------+----------------+
| id_pais_origen | int(11)     | NO   | PRI | NULL    | auto_increment |
| nom_pais_origen | varchar(25) | YES  |     | NULL    |                |
| descripcion    | varchar(300) | YES  |     | NULL    |                |
+----------------+-------------+------+-----+---------+----------------+
```

3 rows in set (0.008 sec)

2.4.2. Entidades secundarias

Después de haber creado las tablas de las entidades fuertes, podemos continuar con las entidades secundarias, hay que tener en cuenta que tablas están creadas, ya que, si el atributo que es derivado de otra entidad no existe, el motor BD no permite crear la tabla.

Desde este punto iniciamos a aplicar la integridad referencial, ya que en estas entidades vamos a tener **llaves foráneas (FK)**. Para la primera entidad animal referenciamos la llave foránea de las entidades especie, raza, país de origen, hábitat, alimento y dieta. Para todas ella se uso **ON DELETE RESTRICT**, para proteger registros que estén asociados al animal y elimine registros que no tenga referencias, porque no tiene sentido que hayan registros en estas tablas si no hay animales, entonces cada vez que se elimina un animal, los registros en esas tablas que estén asociados al animar también serán eliminados esos registros y se actualizaran todas las referencias con **ON UPDATE CASCADE**:

Entidad animal

MariaDB [odbcUkumari]> CREATE TABLE tAnimal(id_animal INT PRIMARY KEY AUTO_INCREMENT NOT NULL,

-> nom_animal VARCHAR(20), edad_animal INT, sexo_animal VARCHAR(1), fecha_nacimiento_animal DATE, id_especie INT, id_raza INT, id_pais_origen INT, id_habitat INT, id_alimento INT, estado_sanitario VARCHAR(1), id_dieta INT, foto BLOB, pelaje VARCHAR(30),

-> FOREIGN KEY (id_especie) REFERENCES tEspecie(id_especie) ON DELETE RESTRICT ON UPDATE CASCADE,

-> FOREIGN KEY (id_raza) REFERENCES tRaza(id_raza) ON DELETE RESTRICT ON UPDATE CASCADE,

-> FOREIGN KEY (id_pais_origen) REFERENCES tPaisDeOrigen(id_pais_origen) ON DELETE RESTRICT ON UPDATE CASCADE,

-> FOREIGN KEY (id_habitat) REFERENCES tHabitat(id_habitat) ON DELETE RESTRICT ON UPDATE CASCADE,

-> FOREIGN KEY (id_alimento) REFERENCES tAlimento(id_alimento) ON DELETE RESTRICT ON UPDATE CASCADE,

-> FOREIGN KEY (id_dieta) REFERENCES tDieta(id_dieta) ON DELETE RESTRICT ON UPDATE CASCADE);

Query OK, 0 rows affected (0.064 sec)

```
MariaDB [odbcUkumari]> desc tAnimal;
+-------------------------+-------------+------+-----+---------+----------------+
| Field                   | Type        | Null | Key | Default | Extra          |
+-------------------------+-------------+------+-----+---------+----------------+
| id_animal               | int(11)     | NO   | PRI | NULL    | auto_increment |
| nom_animal              | varchar(20) | YES  |     | NULL    |                |
| edad_animal             | int(11)     | YES  |     | NULL    |                |
| sexo_animal             | varchar(1)  | YES  |     | NULL    |                |
| fecha_nacimiento_animal | date        | YES  |     | NULL    |                |
| id_especie              | int(11)     | YES  | MUL | NULL    |                |
| id_raza                 | int(11)     | YES  | MUL | NULL    |                |
| id_pais_origen          | int(11)     | YES  | MUL | NULL    |                |
| id_habitat              | int(11)     | YES  | MUL | NULL    |                |
| id_alimento             | int(11)     | YES  | MUL | NULL    |                |
| estado_sanitario        | varchar(1)  | YES  |     | NULL    |                |
| id_dieta                | int(11)     | YES  | MUL | NULL    |                |
| foto                    | blob        | YES  |     | NULL    |                |
| pelaje                  | varchar(30) | YES  |     | NULL    |                |
+-------------------------+-------------+------+-----+---------+----------------+
14 rows in set (0.009 sec)
```

Entidad veterinaria

```
MariaDB [odbcUkumari]> CREATE TABLE tVeterinario(id_veterinario INT
PRIMARY KEY AUTO_INCREMENT NOT NULL,
    -> nom_veterinario VARCHAR(20), apellidos_veterinario VARCHAR(20),
id_especialidad INT, fecha_naciemiento_veterinario DATE, edad_veterinario
INT(2), genero_veterinario VARCHAR(1), telefono_veterinario INT(10),
email_veterinario VARCHAR(30), foto BLOB, direccion_veterinario VARCHAR(30),
```

-> FOREIGN KEY (id_especialidad) REFERENCES tEspecialidadVeterinaria(id_especialidad) ON DELETE RESTRICT ON UPDATE CASCADE);

Query OK, 0 rows affected (0.045 sec)

MariaDB [odbcUkumari]> desc tVeterinario;

```
+-----------------------------+------------+------+-----+---------+----------------+
| Field                       | Type       | Null | Key | Default | Extra          |
+-----------------------------+------------+------+-----+---------+----------------+
| id_veterinario              | int(11)    | NO   | PRI | NULL    | auto_increment |
| nom_veterinario             | varchar(20) | YES  |     | NULL    |                |
| apellidos_veterinario       | varchar(20) | YES  |     | NULL    |                |
| id_especialidad             | int(11)    | YES  | MUL | NULL    |                |
| fecha_naciemiento_veterinario | date       | YES  |     | NULL    |                |
| edad_veterinario            | int(2)     | YES  |     | NULL    |                |
| genero_veterinario          | varchar(1)  | YES  |     | NULL    |                |
| telefono_veterinario        | int(10)    | YES  |     | NULL    |                |
| email_veterinario           | varchar(30) | YES  |     | NULL    |                |
| foto                        | blob       | YES  |     | NULL    |                |
| direccion_veterinario       | varchar(30) | YES  |     | NULL    |                |
+-----------------------------+------------+------+-----+---------+----------------+
```

11 rows in set (0.008 sec)

Entidad usuario

MariaDB [odbcUkumari]> CREATE TABLE tUsuario(id_usuario INT PRIMARY KEY AUTO_INCREMENT NOT NULL,

-> nom_usuario VARCHAR(20), apellido_usuario VARCHAR(20), fecha_nacimiento_usuario DATE, edad_usuario INT(2), genero_usuario VARCHAR(1), telefono_usuario INT(10), id_rol INT, foto BLOB, descripcion VARCHAR(300),

-> FOREIGN KEY (id_rol) REFERENCES tRol(id_rol) ON DELETE RESTRICT ON UPDATE CASCADE);

Query OK, 0 rows affected (0.038 sec)

MariaDB [odbcUkumari]> desc tUsuario;

```
+-------------------------+-------------+------+-----+---------+----------------+
| Field                   | Type        | Null | Key | Default | Extra          |
+-------------------------+-------------+------+-----+---------+----------------+
| id_usuario              | int(11)     | NO   | PRI | NULL    | auto_increment |
| nom_usuario             | varchar(20) | YES  |     | NULL    |                |
| apellido_usuario        | varchar(20) | YES  |     | NULL    |                |
| fecha_nacimiento_usuario | date       | YES  |     | NULL    |                |
| edad_usuario            | int(2)      | YES  |     | NULL    |                |
| genero_usuario          | varchar(1)  | YES  |     | NULL    |                |
| telefono_usuario        | int(10)     | YES  |     | NULL    |                |
| id_rol                  | int(11)     | YES  | MUL | NULL    |                |
| foto                    | blob        | YES  |     | NULL    |                |
| descripcion             | varchar(300) | YES |     | NULL    |                |
+-------------------------+-------------+------+-----+---------+----------------+
```

10 rows in set (0.009 sec)

Entidad historia clinica

MariaDB [odbcUkumari]> CREATE TABLE tHistoriaClinica(id_historia_clinica INT PRIMARY KEY AUTO_INCREMENT NOT NULL,

-> id_animal INT, id_veterinario INT, id_cuidador INT, fecha_inicio_historia DATE, fecha_fin_historia DATE,

-> FOREIGN KEY (id_animal) REFERENCES tAnimal(id_Animal) ON DELETE RESTRICT ON UPDATE CASCADE,

-> FOREIGN KEY (id_veterinario) REFERENCES tVeterinario(id_veterinario) ON DELETE RESTRICT ON UPDATE CASCADE,

-> FOREIGN KEY (id_cuidador) REFERENCES tCuidador(id_cuidador) ON DELETE RESTRICT ON UPDATE CASCADE);

Query OK, 0 rows affected (0.047 sec)

MariaDB [odbcUkumari]> desc tHistoriaClinica;

```
+----------------------+---------+------+-----+---------+----------------+
| Field                | Type    | Null | Key | Default | Extra          |
+----------------------+---------+------+-----+---------+----------------+
| id_historia_clinica  | int(11) | NO   | PRI | NULL    | auto_increment |
| id_animal            | int(11) | YES  | MUL | NULL    |                |
| id_veterinario       | int(11) | YES  | MUL | NULL    |                |
| id_cuidador          | int(11) | YES  | MUL | NULL    |                |
| fecha_inicio_historia | date   | YES  |     | NULL    |                |
| fecha_fin_historia   | date    | YES  |     | NULL    |                |
+----------------------+---------+------+-----+---------+----------------+
```

6 rows in set (0.009 sec)

Entidad tratamiento

MariaDB [odbcUkumari]> CREATE TABLE tTratamiento(id_tratamiento INT PRIMARY KEY AUTO_INCREMENT NOT NULL,

-> fecha_inicio_tratamiento DATE, fecha_fin_tratamiento DATE, descripcion_tratamiento VARCHAR(300), id_animal INT,

-> FOREIGN KEY (id_animal) REFERENCES tAnimal(id_animal) ON DELETE CASCADE ON UPDATE CASCADE);

Query OK, 0 rows affected (0.046 sec)

MariaDB [odbcUkumari]> desc tTratamiento;

```
+------------------------+-------------+------+-----+---------+----------------+
| Field                  | Type        | Null | Key | Default | Extra          |
+------------------------+-------------+------+-----+---------+----------------+
| id_tratamiento         | int(11)     | NO   | PRI | NULL    | auto_increment |
| fecha_inicio_tratamiento | date      | YES  |     | NULL    |                |
| fecha_fin_tratamiento  | date        | YES  |     | NULL    |                |
| descripcion_tratamiento | varchar(300) | YES |     | NULL    |                |
| id_animal              | int(11)     | YES  | MUL | NULL    |                |
+------------------------+-------------+------+-----+---------+----------------+
5 rows in set (0.007 sec)
```

Entidad diagnostico

```
MariaDB [odbcUkumari]> CREATE TABLE tDiagnostico(id_diagnostico INT
PRIMARY KEY AUTO_INCREMENT NOT NULL,
    -> fecha_diagnostico DATE, descripcion_diagnostico VARCHAR(300),
id_enfermedad INT,
    -> FOREIGN KEY (id_enfermedad) REFERENCES
tEnfermedad(id_enfermedad) ON DELETE RESTRICT ON UPDATE CASCADE);
Query OK, 0 rows affected (0.041 sec)

MariaDB [odbcUkumari]> desc tDiagnostico;
+------------------------+-------------+------+-----+---------+----------------+
| Field                  | Type        | Null | Key | Default | Extra          |
+------------------------+-------------+------+-----+---------+----------------+
| id_diagnostico         | int(11)     | NO   | PRI | NULL    | auto_increment |
| fecha_diagnostico      | date        | YES  |     | NULL    |                |
| descripcion_diagnostico | varchar(300) | YES |     | NULL    |                |
| id_enfermedad          | int(11)     | YES  | MUL | NULL    |                |
```

+-----------------------+-------------+------+-----+---------+----------------+

4 rows in set (0.005 sec)

Entidad detalle historia clínica

MariaDB [odbcUkumari]> CREATE TABLE tDetalleHistoriaClinica(id_detalle_historia INT PRIMARY KEY AUTO_INCREMENT NOT NULL,

-> descripcion_detalle VARCHAR(300), fecha_detalle_historia DATE, id_historia_clinica INT,

-> FOREIGN KEY (id_historia_clinica) REFERENCES tHistoriaClinica(id_historia_clinica) ON DELETE CASCADE ON UPDATE CASCADE);

Query OK, 0 rows affected (0.040 sec)

MariaDB [odbcUkumari]> desc tDetalleHistoriaClinica;

+-----------------------+-------------+------+-----+---------+----------------+

| Field | Type | Null | Key | Default | Extra |

+-----------------------+-------------+------+-----+---------+----------------+

| id_detalle_historia | int(11) | NO | PRI | NULL | auto_increment |

| descripcion_detalle | varchar(300) | YES | | NULL | |

| fecha_detalle_historia | date | YES | | NULL | |

| id_historia_clinica | int(11) | YES | MUL | NULL | |

+-----------------------+-------------+------+-----+---------+----------------+

4 rows in set (0.009 sec)

Entidad bitácora

MariaDB [odbcUkumari]> CREATE TABLE tBitacora(id_bitacora INT PRIMARY KEY AUTO_INCREMENT NOT NULL,

-> id_usuario INT, descripcion_accion VARCHAR(300), fecha_accion DATE, hora_accion TIME, hora_salida TIME,

-> FOREIGN KEY (id_usuario) REFERENCES tUsuario(id_usuario) ON DELETE RESTRICT ON UPDATE CASCADE);

Query OK, 0 rows affected (0.044 sec)

MariaDB [odbcUkumari]> desc tBitacora;

```
+-------------------+-------------+------+-----+---------+----------------+
| Field             | Type        | Null | Key | Default | Extra          |
+-------------------+-------------+------+-----+---------+----------------+
| id_bitacora       | int(11)     | NO   | PRI | NULL    | auto_increment |
| id_usuario        | int(11)     | YES  | MUL | NULL    |                |
| descripcion_accion | varchar(300) | YES  |     | NULL    |                |
| fecha_accion      | date        | YES  |     | NULL    |                |
| hora_accion       | time        | YES  |     | NULL    |                |
| hora_salida       | time        | YES  |     | NULL    |                |
+-------------------+-------------+------+-----+---------+----------------+
```

6 rows in set (0.008 sec)

Entidad medicamento

MariaDB [odbcUkumari]> CREATE TABLE tMedicamento(id_medicamento INT PRIMARY KEY AUTO_INCREMENT NOT NULL,

-> nom_medicamento VARCHAR(20), descripcion_medicamento VARCHAR(300), unidades VARCHAR(30), fecha_fabricacion DATE, fecha_vencimiento DATE, tipo_medicamento VARCHAR(40), laboratorio VARCHAR(40), id_tratamiento int,

-> FOREIGN KEY (id_tratamiento) REFERENCES tTratamiento(id_tratamiento) ON DELETE RESTRICT ON UPDATE CASCADE);

Query OK, 0 rows affected (0.011 sec)

```
MariaDB [odbcUkumari]> desc tMedicamento;
+-----------------------+-------------+------+-----+---------+----------------+
| Field                 | Type        | Null | Key | Default | Extra          |
+-----------------------+-------------+------+-----+---------+----------------+
| id_medicamento        | int(11)     | NO   | PRI | NULL    | auto_increment |
| nom_medicamento       | varchar(20) | YES  |     | NULL    |                |
| descripcion_medicamento | varchar(300) | YES |     | NULL    |                |
| unidades              | varchar(30) | YES  |     | NULL    |                |
| fecha_fabricacion     | date        | YES  |     | NULL    |                |
| fecha_vencimiento     | date        | YES  |     | NULL    |                |
| tipo_medicamento      | varchar(40) | YES  |     | NULL    |                |
| laboratorio           | varchar(40) | YES  |     | NULL    |                |
| id_tratamiento        | int(11)     | YES  | MUL | NULL    |                |
+-----------------------+-------------+------+-----+---------+----------------+
9 rows in set (0.010 sec)

MariaDB [odbcUkumari]> show tables;
+-------------------------+
| Tables_in_odbcukumari   |
+-------------------------+
| talimento               |
| tanimal                 |
| tbitacora               |
| tcuidador               |
| tdetallehistoriaclinica |
| tdiagnostico            |
| tdieta                  |
| tenfermedad             |
```

| tespecialidadveterinaria |

| tespecie |

| thabitat |

| thistoriaclinica |

| tmedicamento |

| tpaisdeorigen |

| traza |

| trol |

| ttratamiento |

| tusuario |

| tveterinario |

+-------------------------+

19 rows in set (0.014 sec)

2.5. MODELO RELACIONAL (MR)

El modelo relacional es una forma estándar de organizar y manipular datos utilizando tablas, también llamadas relaciones. Estas tablas están compuestas por filas y columnas, donde cada columna representa un atributo y cada fila un registro o entrada de datos. El modelo relacional se basa en la teoría de conjuntos y la lógica de predicados.

Características principales:

- **Tablas (relaciones):** los datos se almacenan en tablas que organizan la información en filas y columnas.
- **Atributos:** cada columna representa un atributo de la entidad que se está modelando.
- **Tuplas (filas):** cada fila representa un registro de la tabla, conteniendo los valores de los atributos.
- **Claves:** las claves (primarias y foráneas) permiten establecer relaciones entre diferentes tablas.
- **Normalización:** un proceso que ayuda a reducir la redundancia y mejorar la integridad de los datos.

- **Independencia física y lógica:** la estructura de los datos no depende de la forma en que se almacenan físicamente, ni de como se accede a ellos.

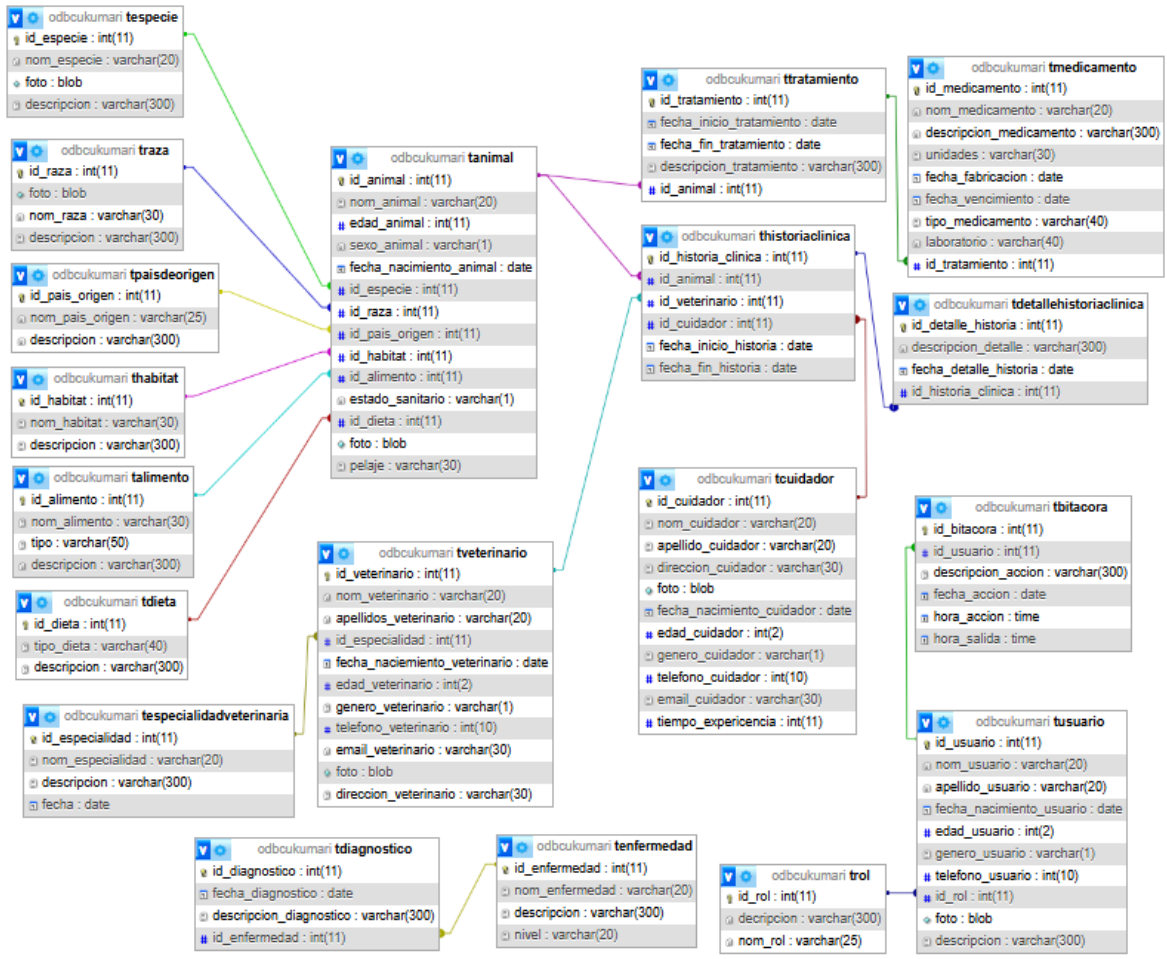
Ventajas:

- **Organización estructurada:** facilita la comprensión y el manejo de los datos.
- **Reducción de redundancia:** minimiza la duplicidad de datos y evita inconsistencias.
- **Facilita la consulta:** permite realizar consultas complejas y eficientes a los datos,
- **Estandarización:** proporciona una forma común y ampliamente utilizada de modelar bases de datos.

El modelo relacional y el modelo entidad-relación están estrechamente relacionados siendo el modelo ER una herramienta de diseño que se utiliza para crear modelos relacionales. El modelo ER define la estructura lógica de la base de datos, mientras que el modelo relacional proporciona la implementación física, organizando los datos en tablas con relaciones entre ellas.

2.5.1. Aplicación:

Una vez creadas las tablas con sus índices y cardinalidades, podemos dirigirnos al administrador de MySQL donde podemos encontrar el **MODELO RELACIONAL (MR)**:



2.6. INSERCION DE DATOS

Existen diversas formas estas son las más conocidas:

Simple:

INSERT INTO nomTabla(campos) VALUES(valores);

INSERT INTO nomTabla VALUES(valores);

Múltiple:

INSERT INTO nomTabla(campos) VALUES(valores), (valores);

INSERT INTO nomTabla VALUES(valores), (valores);

No Procedimentales				
RECUPERACION				
SELECT … FROM				
FN AGRAGADA	PREDICADOS	PALABRAS CLAVE		
MAX(): devuelve el valor máximo. MIN(): devuelve el valor mínimo. SUM(): devuelve el valor de la suma de los valores del campo. COUNT(): devuelve el numero de filas que cumplen la condicion. AVG(): devuelve el promedio de los valores del campo.	BETWEEN … AND: comprueba que el valor esta dentro de un intervalo. LIKE: compara un campo con una cadena alfanumérica. LIKE admite el uso de caracteres comodines. ALL: señala todos los elementos de la selección de la consulta. ANY: indica que la condicion se cumplirá si la comparación es cierta para al menos un elemento del conjunto. EXISTS: devuelve un valor verdadero si el resultado de una subconsulta devuelve resultados. IN: comprueba si un campo se encuentra dentro de un determinado rango. El rango puede ser una sentencia SELECT.	ALL AVG CHECK CREATE DELETE EXISTS FROM IN INTO NOT OR SELECT UNION VALUES AND BEGIN CLOSE CURSOR DESC FETCH GRANT INDEX LIKE NUMERIC ORDER	SET BY COUNT DECIMAL DISTINCT FLOAT GROUP INSERT MAX ON REVOKE SUM UPDATE WHERE ASC CHAR COMMIT DECLARE DEFAULT FOR HAVING INTEGER MIN OPEN ROLLBACK	TABLE USER WITH

2.6.1. Aplicación:

En este caso usamos una inserción menos convencional, ya que nuestros registros se encuentran en hojas de Excel. Para traer los registros a la base de datos, convertimos la hoja de Excel en un archivo csv. De esta manera utilizaremos un comando un poco especial, para que cargar el archivo .csv que se encuentra en memoria, se indica a través del comando la ruta para encontrar u

acceder al archivo, para que encuentre la entidad y obtenga los registros. La sintaxis **LOAD DATA LOCAL INFILE 'C:/TALLER/Ukumari.csv' INTO TABLE nomTabla FIELDS TERMINATED BY ';' OPTIONALLY ENCLOSED BY '"' LINES TERMINATED BY '\r\n' IGNORE 1 LINES;** nos permite realizar esta operación en el mismo orden el que creamos las tablas:

```
MariaDB [odbcUkumari]> LOAD DATA LOCAL INFILE 'C:/TALLER/Ukumari.csv'
INTO TABLE tEnfermedad FIELDS TERMINATED BY ';' OPTIONALLY
ENCLOSED BY '"' LINES TERMINATED BY '\r\n' IGNORE 1 LINES

    -> ;

Query OK, 150 rows affected, 34 warnings (0.015 sec)

Records: 150  Deleted: 0  Skipped: 0  Warnings: 34


MariaDB [odbcUkumari]> SELECT * FROM tEnfermedad;

+---------------+-----------------------+-----------------------------------------------------------------
+-----------------+

| id_enfermedad | nom_enfermedad         | descripcion
| nivel           |

+---------------+-----------------------+-----------------------------------------------------------------
+-----------------+

|             1 | Gastroenteritis        | Gastroenteritis aguda                                            |
Cr├¡tico         |

|             2 | Otitis externa         | Otitis externa bilateral                                         |
Extremo         |

|             3 | Dermatitis             | Dermatitis al├®rgica                                              |
Grave           |

|             4 | Fractura de f├®mur      | Fractura de f├®mur
| Muy grave       |

|             5 | Conjuntivitis          | Conjuntivitis bacteriana                                         |
Severo          |

|             6 | Obesidad               | Obesidad moderada
| Alto            |

|             7 | Infecci├│n urinaria     | Infecci├│n urinaria (ITU)
| Moderado        |

|             8 | Cuerpo extra├▒o g├ístr   | Cuerpo extra├▒o en est├│mago
| Leve            |
```

| 9 | Artritis | Artritis degenerativa | M ├¡nimo |

| 10 | Sarna demodecica | Sarna demod ├®cica | Agudo |

.

.

.

| 146 | S ├¡ndrome de Fanconi | S ├¡ndrome renal con p ├®rdida excesiva de nutrientes | Localizado |

| 147 | Mielofibrosis | Reemplazo de m ├®dula ├│sea por tejido fibroso | Generalizado |

| 148 | Osteodistrofia hiper | Deformaci ├│n ├│sea por alteraci ├│n en el metabolismo del calcio | Sist ├®mico |

| 149 | Amiloidosis | Acumulaci ├│n anormal de prote ├¡na amiloide en tejidos | Focal |

| 150 | Necrosis avascular | Muerte del tejido ├│seo por falta de irrigaci ├│n sangu ├¡nea | Multifocal |

+--------------+----------------------+--+-----------------+

150 rows in set (0.003 sec)

De esta forma completamos los registros para las 19 tablas.

3. TERCER CORTE

3.1. CONCEPTOS

3.1.1. Teoría de conjuntos:

Un conjunto es una colección de objetos distintos (llamados elementos) que comparten una propiedad común, se denotan con letras mayúsculas A, B, C, … ,. Los elementos se escriben entre llaves {}, separadas por comas.

A = {1, 2, 3, 4]

B = {a, b, c, d}

Si un elemento x está en un conjunto A, se escribe **x ∈ A**. si no pertenece, se escribe **x ∉ A**.

Conjunto vacío **(Ø o {})**, conjunto universal **(U)**, conjunto unitario **solo tiene un elemento**.

Unión	∪	∈
Intersección	∩	∉
Diferencia	-	∀
Diferencia simétrica	Δ	∀
Producto cartesiano	×	: / |
Complemento	'	⊂

3.1.2. Operaciones de conjuntos:

Unión: todos los elementos que están en A y en B.

A ∪ B = { x | x ∈ A o x ∈ B }

Interseccion: elementos comunes a ambos conjuntos.

A ∩ B = { x | x ∈ A y x ∈ B }

Diferencia: elementos que están en A pero no en B.

A - B = { x | x ∈ A y x ∉ B }

Diferencia simétrica: elementos que están en A o en B, pero no en ambos.

A Δ B = **(A ∪ B) – (B ∩ A)**

A Δ B = **(A - B) ∪ (B - A)**

Complemento: elementos que no están en A (respecto a un conjunto universal U).

A' = **U - A**

Producto cartesiano: conjunto de todos los pares ordenados (a, b) donde a ∈ A y B ∈ B.

A × B = { (a, b) | a ∈ A, b ∈ B }

En las bases de datos la teoría de conjuntos es fundamental, ya que estas manejan datos organizado en tablas. Las operaciones de conjuntos permiten consultar, filtrar y combinar datos de manera eficiente.

- **Tablas = conjuntos.**
- **Registro (fila) = elemento de un conjunto.**
- **Atributo (columna) = propiedad de los elementos.**

3.1.3. Operaciones de conjuntos en base de datos:

Unión: combina resultados de dos consultas, eliminando duplicados.

A ∪ B

Sintaxis → SELECT campo FROM nomTabla UNION SELECT campo FROM nomTabla;

Intersección: devuelve registros comunes a ambas consultas.

A ∩ B

Sintaxis → SELECT campo FROM nomTabla INTERSECT SELECT campo FROM nomTabla;

Diferencia: registros en la primera consulta que no están en la segunda.

A - B

Sintaxis → SELECT campo FROM nomTabla EXCEPT SELECT campo FROM nomTabla;

Producto cartesiano: combina cada fila de una tabla con todas las filas de otra.

A × B

Sintaxis → SELECT * FROM campo CROSS JOIN campo;

Diferencia simétrica: registros que están en A o en B, pero no en ambos.

A Δ B

Sintaxis → (SELECT * FROM nomTabla EXCEPT SELECT * FROM nomTabla) UNION (SELECT * FROM nomTabla EXCEPT SELECT * FROM nomTabla);

3.1.4. Notación por extensión en base de datos:

Se utiliza para definir tablas, conjuntos de datos o relaciones enumerando explícitamente todos sus elementos (registros). Su propósito es especificar el contenido de una tabla o relación.

D = {campo, campo, campo, …, campo}

3.1.5. Notación por compension en base de datos:

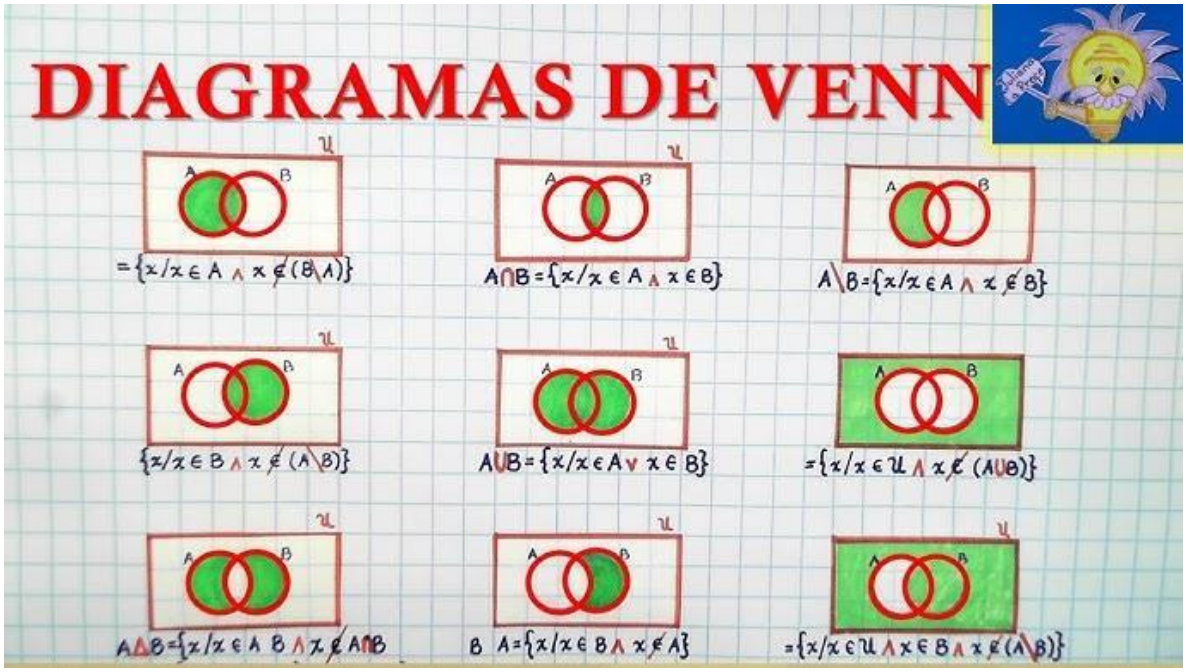
Permite definir conjuntos de datos (tablas, vistas o resultados de consultas) mediante reglas lógicas o condiciones, en lugar de enumerar elementos

explícitamente, su propósito es describir datos basados en propiedades, condiciones o relaciones.

D = {x:x ∀'s x ∈ D}

3.1.6. Diagramas de VENN:

Los diagramas de Venn son representaciones graficas utilizadas en la teoría de conjuntos para visualizar las relaciones entre conjuntos, como uniones, intersecciones, diferencias y complementos.



3.1.7. Tablas temporales en MySQL:

Son una herramienta poderosa que permite almacenar datos intermedio durante la ejecución de sesiones o transacciones.

Sintaxis → CREATE TEMPORARY TABLE [IF NOT EXISTS] nomTabla (campo INT [restricciones], campo INT [restricciones], …,);

CREATE TEMPORARY TABLE nomTabla SELECT * FROM nomTabla WHERE campo sentencia;

CREATE TEMPORARY TABLE nomTabla AS SELECT campo FROM nomTablaOrigen WHERE condicion;

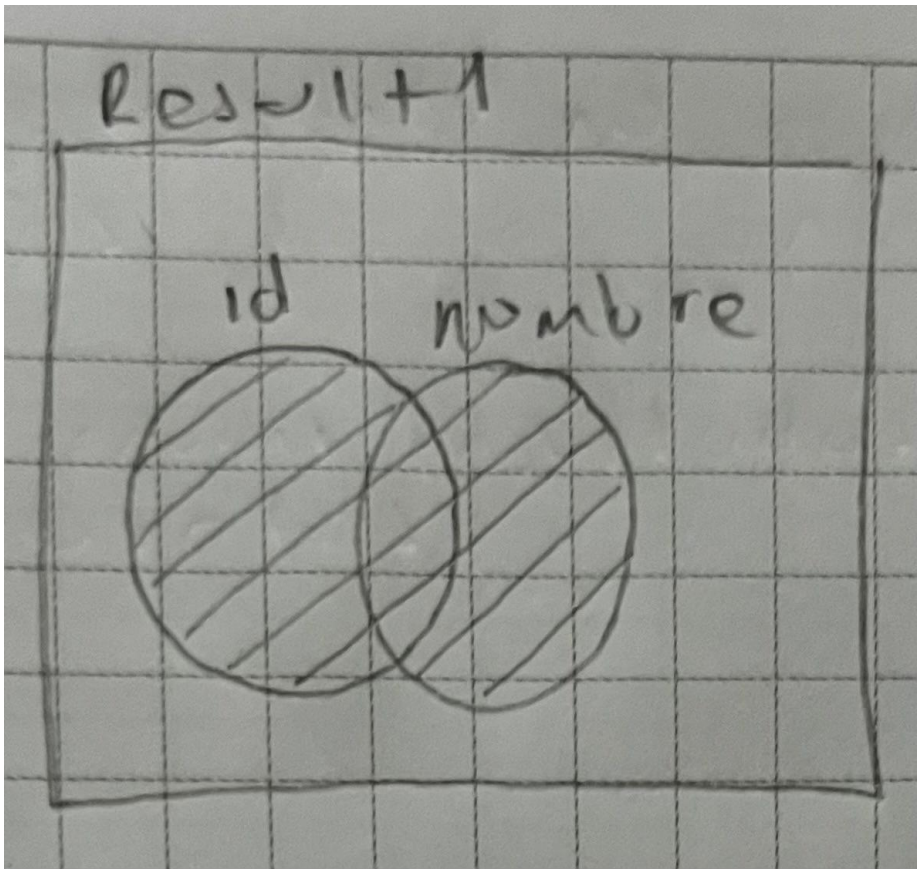
3.1.8. Alias:

SELECT B.* , A.* FROM nomTabla A , nomTabla B;

3.2. Aplicación:

Union: para la unión de conjuntos, se creo una tabla temporal **result1,** su contenido será el resultado de una subconsulta, donde se aplicará la operación entre las tablas **tAnimal** y **tEnfermedad**. Los atributos **id_animal** y **nom_animal** con sus registros, de la entidad **Animal** y los atributos **id_enfermedad** y **nom_enfermedad** con sus registros, de la entidad **Enfermedad** conformaran esta tabla temporal.

Diagrama de Venn:



Notación por extensión:

id U nombre = {1, Leon, Animal, 2, Tigre, Animal, 1, Dermatitis, Enfermedad, 2, Fractura, Enfermedad, …,}

Notación por comprensión:

result1 = {(id_animal, nom_animal, ‘Animal’) | ∀ a ∈ tAnimal) U ((id_enfermedad, nom_enfermedad, ‘Enfermedad’) | ∀ e ∈ tEnfermedad)}

Sintaxis:

```
MariaDB [odbcukumari]> CREATE TEMPORARY TABLE result1 AS SELECT id_animal AS id,
    -> nom_animal AS nombre,
    -> 'Animal' AS tipo FROM tAnimal UNION SELECT id_enfermedad AS id,
    -> nom_enfermedad AS nombre,
    -> 'Enfermedad' AS tipo FROM tEnfermedad;
Query OK, 298 rows affected (0.030 sec)
Records: 298  Duplicates: 0  Warnings: 0

MariaDB [odbcukumari]> SELECT * FROM result1;
+-----+----------------------+------------+
| id  | nombre               | tipo       |
+-----+----------------------+------------+
|   1 | Simba                | Animal     |
|   2 | Nala                 | Animal     |
|   3 | Mufasa               | Animal     |
|   4 | Sarabi               | Animal     |
|   5 | Baloo                | Animal     |
|   6 | Lola                 | Animal     |
|   7 | Misha                | Animal     |
|   8 | Dumbo                | Animal     |
|   9 | Elli                 | Animal     |
|  10 | Tembo                | Animal     |
```

| 142 | S ├¡ndrome de malabsor | Enfermedad |

| 143 | Atrofia progresiva d | Enfermedad |

| 144 | Enfermedad de von Wi | Enfermedad |

| 145 | Poliradiculoneuritis | Enfermedad |

| 146 | S ├¡ndrome de Fanconi | Enfermedad |

| 147 | Mielofibrosis | Enfermedad |

| 148 | Osteodistrofia hiper | Enfermedad |

| 149 | Amiloidosis | Enfermedad |

| 150 | Necrosis avascular | Enfermedad |

+-----+----------------------+------------+

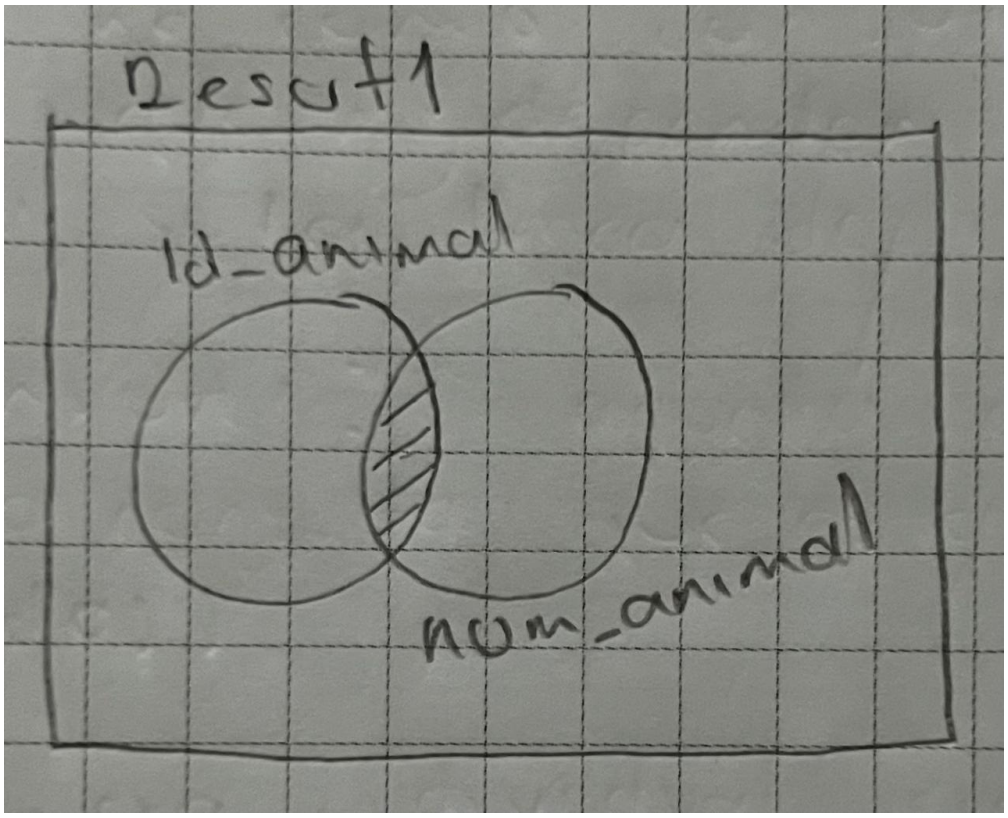
298 rows in set (0.001 sec)

MariaDB [odbcukumari]> DROP TABLE IF EXISTS result1;

Query OK, 0 rows affected (0.013 sec)

intersección: para la intersección de conjuntos, se creó una tabla temporal **result1,** su contenido será el resultado de una subconsulta, donde se aplicará la operación entre las tablas **tAnimal**, **tDiagnostico** y **tHistoriaClinica**, para identificar que animales tienen registros en las entidades diagnostico e historia clinica. Los atributos **id_animal** y **nom_animal** con sus registros, de la entidad **Animal,** conformaran esta tabla temporal.

Diagrama de Venn:



Notación por extencion:

id_animal ∩ nom_animal = {1, Leon, 2, Oso, Tigre, 5, …,}

Notación por comprensión:

result1 = {(a.id_animal, a.nom_animal) | a ∈ tAnimal ∧ a.id_animal ∈ π_id_animal(tDiagnostico) ∧ a,id_animal ∈ π_id_animal(tHistoriaClinica)}

Sintaxis:

MariaDB [odbcukumari]> CREATE TEMPORARY TABLE result1 AS SELECT a.id_animal, a.nom_animal FROM tAnimal a WHERE a.id_animal IN (SELECT id_animal FROM tDiagnostico INTERSECT SELECT id_animal FROM tHistoriaClinica);

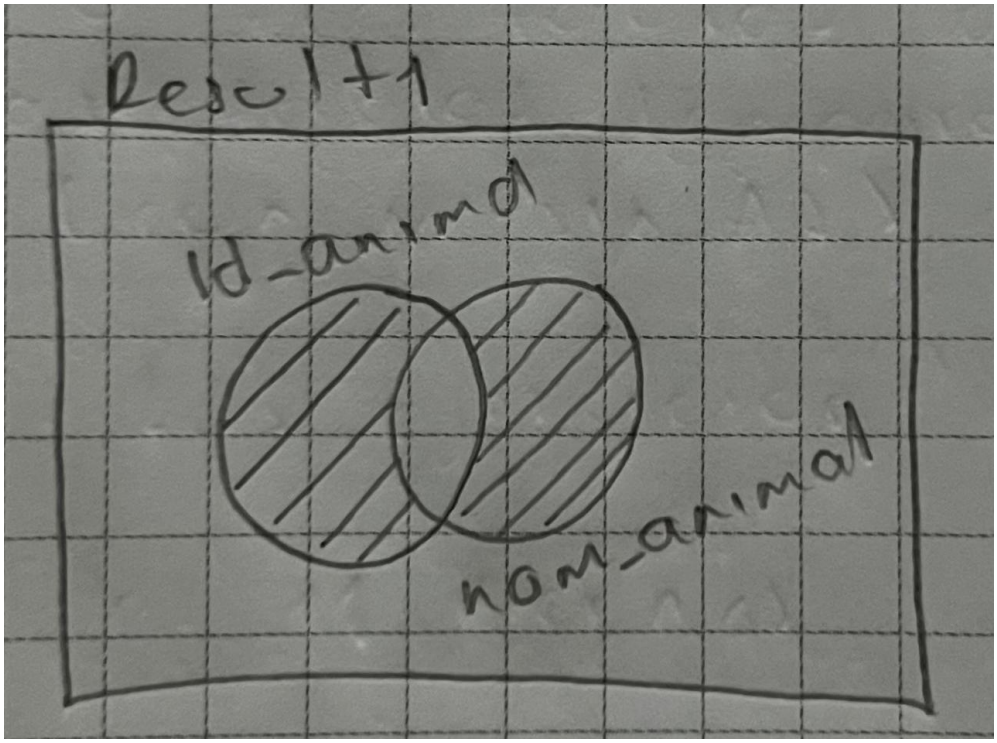
Query OK, 146 rows affected (0.069 sec)

Records: 146 Duplicates: 0 Warnings: 0

```
MariaDB [odbcukumari]> SELECT * FROM result1;
+-----------+------------+
| id_animal | nom_animal |
+-----------+------------+
|         1 | Simba      |
|         2 | Nala       |
|         3 | Mufasa     |
|         4 | Sarabi     |
|         5 | Baloo      |
|         6 | Lola       |
|         7 | Misha      |
|         8 | Dumbo      |
|         9 | Elli       |
|        10 | Tembo      |
|       144 | Haku       |
|       145 | Indra      |
|       146 | Jova       |
|       147 | Kaida      |
|       148 | Lirio      |
|       149 | Mako       |
|       150 | Nixie      |
+-----------+------------+
146 rows in set (0.002 sec)

MariaDB [odbcukumari]> DROP TABLE IF EXISTS result1;
Query OK, 0 rows affected (0.024 sec)
```

Diferencia: para la diferencia de conjuntos, se creó una tabla temporal **result1,** su contenido será el resultado de una subconsulta, donde se aplicará la operación entre las tablas **tAnimal**, **tDiagnostico** y **tHistoriaClinica**, para identificar que animales tienen registros en la entidades diagnostico pero no tienen registros en historia clinica. Los atributos **id_animal** y **nom_animal** con sus registros, de la entidad **Animal,** conformaran esta tabla temporal.

Diagrama de Venn:



Notación por extencion:

id_animal – nom_animal = {2, Elefante, Jirafa, 4, Cebra, 6, …,}

Notación por comprensión:

result1 = {(a.id_animal, a.nom_animal) | a ∈ tAnimal ∧ a.id_animal ∈ π_id_animal(tDiagnostico) ∧ a.id_animal ∉ π_id_animal(tHistoriaClinica)}

Sintaxis:

MariaDB [odbcukumari]> CREATE TEMPORARY TABLE result1 AS SELECT a.id_animal, a.nom_animal FROM tAnimal a WHERE a.id_animal IN (SELECT

id_animal FROM tDiagnostico) AND a.id_animal NOT IN (SELECT id_animal FROM tHistoriaClinica);

Query OK, 2 rows affected (0.055 sec)

Records: 2 Duplicates: 0 Warnings: 0

MariaDB [odbcukumari]> SELECT * FROM result1;

+-----------+------------+

| id_animal | nom_animal |

+-----------+------------+

| 50 | Quetzal |

| 62 | Thor |

+-----------+------------+

2 rows in set (0.002 sec)

MariaDB [odbcukumari]> DROP TABLE IF EXISTS result1;

Query OK, 0 rows affected (0.001 sec)

Producto cartesiano: para el producto cartesiano de conjuntos, se creó una tabla temporal **result1,** su contenido será el resultado de una subconsulta, donde se aplicará la operación entre las tablas **tAnimal**, **tEnfermedad**, para identificar que el producto cartesiano entre animales y enfermedades. Los atributos **id_animal** y **nom_animal** con sus registros, de la entidad **Animal** y los atributos **id_enfermedad** y **nom_enfermedad**, conformaran esta tabla temporal.

Notación por extencion:

id_animal × nom_animal × id_enfermedad × nom_enfermedad = {1, Leon, 1, Gastroenteritis, 1, Leon, 2, Otitis, 2, Tigre, 1, Gastroenteritis, 2, Tigre, 2, Otitis, 3, Oso, 1, Gastroenteritis, 3, Oso, 2, Otitis, …,}

Notación por comprensión:

result1 = {(a.id_animal, a.nom_animal, e.id_enfermedad, e.nom_enfermedad) | a ∈ tAnimal ∧ e ∈ tEnfermedad}

Sintaxis:

MariaDB [odbcukumari]> CREATE TEMPORARY TABLE result1 AS SELECT a.id_animal, a.nom_animal, e.id_enfermedad, e.nom_enfermedad FROM tAnimal a CROSS JOIN tEnfermedad r;

Query OK, 22200 rows affected (0.055 sec)

Records: 22200 Duplicates: 0 Warnings: 0

MariaDB [odbcukumari]> SELECT * FROM result1;

131	Umi	150	Necrosis avascular
132	Vayu	150	Necrosis avascular
133	Wero	150	Necrosis avascular
134	Xolo	150	Necrosis avascular
135	Yavi	150	Necrosis avascular
136	Zola	150	Necrosis avascular
137	Akili	150	Necrosis avascular
138	Brio	150	Necrosis avascular
139	Cira	150	Necrosis avascular
140	Dune	150	Necrosis avascular
141	Elio	150	Necrosis avascular
142	Fennec	150	Necrosis avascular
143	Gadi	150	Necrosis avascular
144	Haku	150	Necrosis avascular
145	Indra	150	Necrosis avascular
146	Jova	150	Necrosis avascular
147	Kaida	150	Necrosis avascular
148	Lirio	150	Necrosis avascular
149	Mako	150	Necrosis avascular
150	Nixie	150	Necrosis avascular

+----------+-----------+--------------+----------------------+

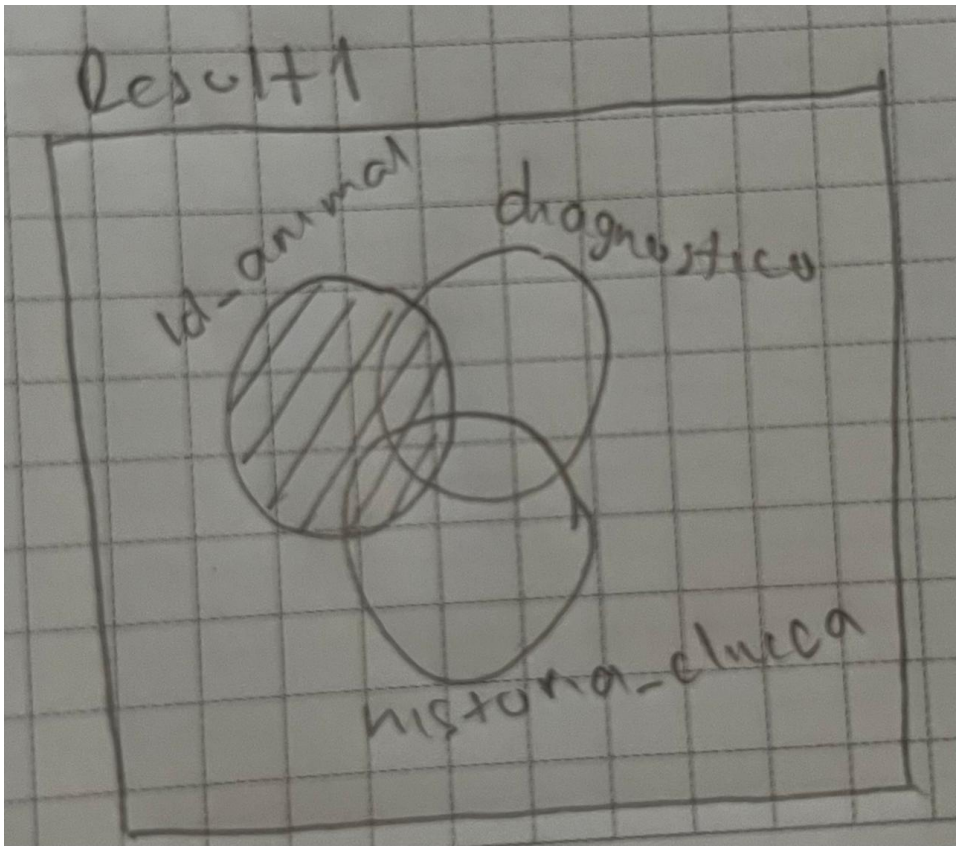
22200 rows in set (0.011 sec)

MariaDB [odbcukumari]> DROP TABLE IF EXISTS result1;

Query OK, 0 rows affected (0.024 sec)

Diferencia simétrica: para la diferencia simetrica de conjuntos, se creó una tabla temporal **result1,** su contenido será el resultado de una subconsulta, donde se aplicará la operación entre las tablas **tAnimal**, **tDiagnostico** y **tHistoriaClinica**, para identificar los animales que tienen diagnóstico, pero no historial. Los atributos **id_animal** y **nom_animal** con sus registros, de la entidad **Animal**, conformaran esta tabla temporal.

Diagrama de Venn:



Notación por extensión:

id_animal = {2, 4, 5, 7, …,}

nom_animal = {Elefante, Jirafa, Rinoceronte, Hipopotamo, …,}

tipo = {Solo diagnóstico, Solo diagnóstico, Solo historial, Solo historial, …,}

Notación por comprensión:

result1 = {(a.id_animal, a.nom_animal, 'Solo diagnostico') | a ∈ tAnimal ∧ a.id_animal ∈ π_id_animal(tDiagnostico) ∧ a.id_animal ∉ π_id_animal(tHistoriaClinica)} U {(a.id_animal, a.nom_animal, 'Solo historia') | a ∈ tAnimal ∧ a.id_animal ∈ π_id_animal(tHistoriaClinica) ∧ a.id_animal ∉ π_id_animal(tDiagnostico)}

Sintaxis:

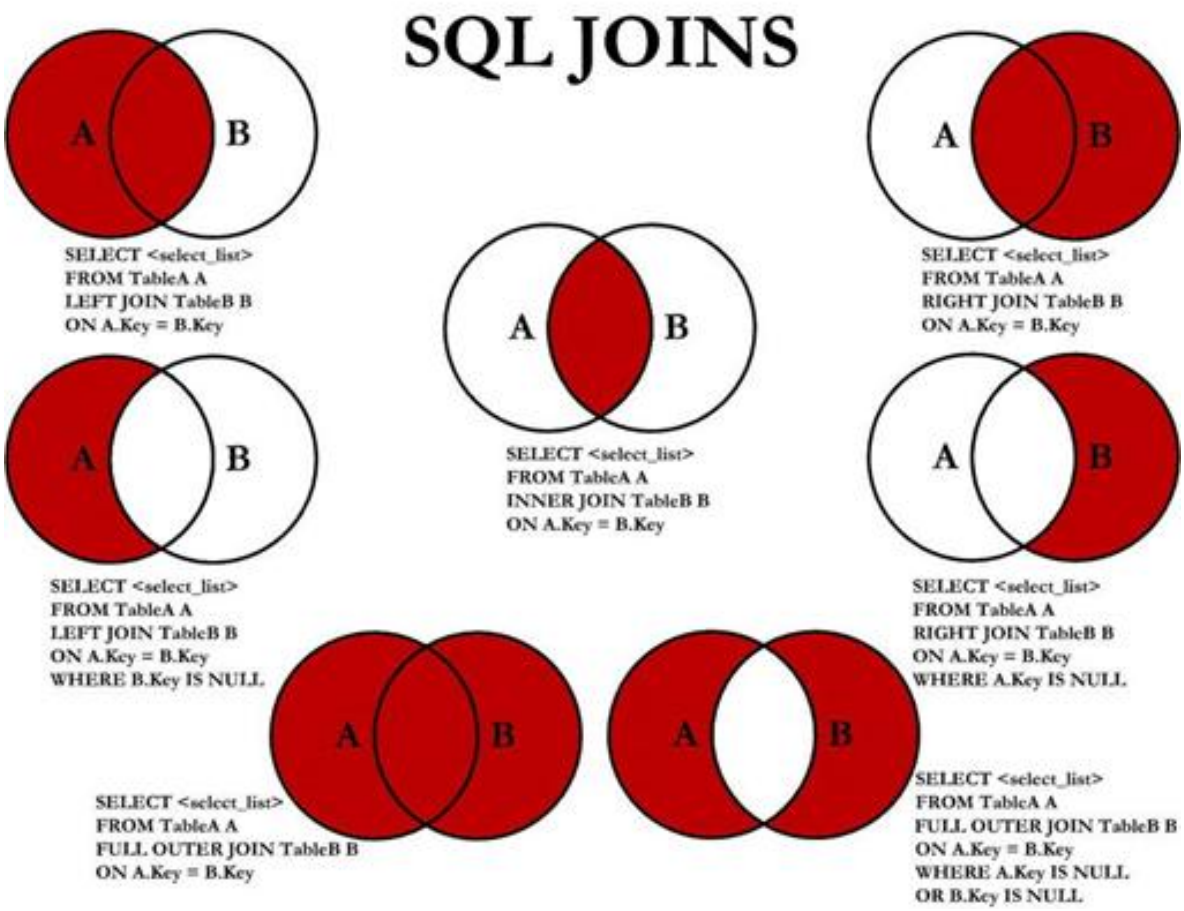
```
MariaDB [odbcukumari]> CREATE TEMPORARY TABLE result1 AS SELECT a.id_animal, a.nom_animal, 'Solo Diagnostico' AS tipo FROM tAnimal a WHERE a.id_animal IN (SELECT id_animal FROM tDiagnostico) AND a.id_animal NOT IN (SELECT id_animal FROM tDiagnostico);

Query OK, 0 rows affected (0.039 sec)

Records: 0  Duplicates: 0  Warnings: 0


MariaDB [odbcukumari]> SELECT * FROM result1;

Empty set (0.002 sec)


MariaDB [odbcukumari]> DROP TABLE IF EXISTS result1;

Query OK, 0 rows affected (0.000 sec)
```

Algebra relacional:

Proporciona un lenguaje formal para describir como manipular y consultar base de datos relacionales. Es una colección de operaciones matemáticas que actúan sobre relaciones para crear nuevas relaciones.

Selección	σ	Filtra tuplas que cumplen una condicion.

Proyeccion	π	Muestra atributos específicos de una relacion.
Union	∪	Devuelve las tuplas de dos relaciones compatibles.
Diferencia	-	Devuelve las tuplas que están en una relacion pero no en otra.
Interseccion	∩	Muestra tuplas comunes en dos relaciones compatibles
Producto cartesiano	×	Combina todas las tuplas de dos relaciones, generando todas las combinaciones posibles.
Renombramiento	ρ	Cambia el nombre de una relacion o sus atributos.
Division	÷	Selecciona el nombre de una relacion que esta en todas las tuplas de otras relaciones.



Aplicación:

σ sexo_animal = ‘M’ (Animal)

Sintaxis:

MariaDB [odbcukumari]> CREATE TEMPORARY TABLE result1 AS SELECT id_animal, nom_animal, sexo_animal FROM tAnimal WHERE sexo_animal = 'M';

Query OK, 87 rows affected (0.010 sec)

Records: 87 Duplicates: 0 Warnings: 0

MariaDB [odbcukumari]> SELECT * FROM result1;

+-----------+------------+-------------+

| id_animal | nom_animal | sexo_animal |

```
+----------+------------+-------------+
|        1 | Simba      | M           |
|        3 | Mufasa     | M           |
|        5 | Baloo      | M           |
|        7 | Misha      | M           |
|        8 | Dumbo      | M           |
|       10 | Tembo      | M           |
|       11 | Shere Khan | M           |
|       13 | Bagheera   | M           |
|      130 | Tamo       | M           |
|      132 | Vayu       | M           |
|      133 | Wero       | M           |
|      134 | Xolo       | M           |
|      135 | Yavi       | M           |
|      138 | Brio       | M           |
|      140 | Dune       | M           |
|      141 | Elio       | M           |
|      142 | Fennec     | M           |
|      143 | Gadi       | M           |
|      144 | Haku       | M           |
|      148 | Lirio      | M           |
|      149 | Mako       | M           |
+----------+------------+-------------+
87 rows in set (0.001 sec)

MariaDB [odbcukumari]> DROP TABLE IF EXISTS result1;
Query OK, 0 rows affected (0.022 sec)
```

π nom_cuidador, apellido_cuidador (tCuidador)

Sintaxis:

MariaDB [odbcukumari]> CREATE TEMPORARY TABLE result1 AS SELECT nom_cuidador, apellido_cuidador FROM tCuidador;

Query OK, 150 rows affected (0.035 sec)

Records: 150 Duplicates: 0 Warnings: 0

MariaDB [odbcukumari]> SELECT * FROM result1;

```
+--------------+------------------+
| nom_cuidador | apellido_cuidador |
+--------------+------------------+
| Liam         | Mart├¡nez         |
| Emma         | Johnson          |
| Noah         | Smith            |
| Olivia       | Garc├¡a           |
| Mateo        | Rodr├¡guez        |
| Sophia       | L├┤pez           |
| Valery       | Ramirez          |
| Andr├®s       | Perez            |
| Lorena       | Torres           |
| Elkin        | Herrera          |
| Juliana      | Lopez            |
| Luis         | Diaz             |
| Johana       | Soto             |
+--------------+------------------+
```

150 rows in set (0.002 sec)

MariaDB [odbcukumari]> DROP TABLE IF EXISTS result1;

Query OK, 0 rows affected (0.001 sec)

π nom_veterinario (σ edad_veterinario = ‘30’ (tVeterinario)) U π nom_veterinario (σ edad_veterinario = ‘38’ (tVeterinario))

Sintaxis:

MariaDB [odbcukumari]> CREATE TEMPORARY TABLE result1 AS SELECT nom_veterinario FROM tveterinario WHERE edad_veterinario = '30' UNION SELECT nom_veterinario FROM tVeterinario WHERE edad_veterinario = '38';

Query OK, 15 rows affected (0.063 sec)

Records: 15 Duplicates: 0 Warnings: 0

MariaDB [odbcukumari]> SELECT * FROM result1;

```
+-----------------+
| nom_veterinario |
+-----------------+
| Catalina        |
| Paula           |
| Mariana         |
| Iv├ín            |
| Viviana         |
| Mauricio        |
| Oriana          |
| Belkis          |
| Juan Carlos     |
| Marcela         |
| Laura           |
| Ana Cristina    |
| Sara            |
| Rosalba         |
| Edelmira        |
```

+----------------+

15 rows in set (0.001 sec)

MariaDB [odbcukumari]> DROP TABLE IF EXISTS result1;

Query OK, 0 rows affected (0.001 sec)

π nom_veterinario (σ edad_veterinario = ‘30’ (tVeterinario)) - π nom_veterinario (σ edad_veterinario = ‘38’ (tVeterinario))

Sintaxis:

MariaDB [odbcukumari]> CREATE TEMPORARY TABLE result1 AS SELECT nom_veterinario FROM tveterinario WHERE edad_veterinario = '30' AND nom_veterinario NOT IN (SELECT nom_veterinario FROM tVeterinario WHERE edad_veterinario = '38');

Query OK, 8 rows affected (0.015 sec)

Records: 8 Duplicates: 0 Warnings: 0

MariaDB [odbcukumari]> SELECT * FROM result1;

+----------------+

| nom_veterinario |

+----------------+

| Catalina |

| Paula |

| Mariana |

| Iv├ín |

| Viviana |

| Mauricio |

| Oriana |

| Belkis |

+----------------+

8 rows in set (0.001 sec)

MariaDB [odbcukumari]> DROP TABLE IF EXISTS result1;

Query OK, 0 rows affected (0.001 sec)

tHistoriaClinica × tAlimento

Sintaxis:

MariaDB [odbcukumari]> CREATE TEMPORARY TABLE result1 AS SELECT * FROM tHistoriaClinica, tAlimento;

Query OK, 22200 rows affected (0.100 sec)

Records: 22200 Duplicates: 0 Warnings: 0

MariaDB [odbcukumari]> SELECT * FROM result1;

```
s ricos en clorofila, enzimas digestivas y nutrientes esenciales.                    |
|              147 |       116 |            68 |          31 | 2020-08-08           | 2024-03-16
|        150 | Alfalfa germinada           | Germinado                  | Alfalfa germinada:
Brotes ricos en clorofila, enzimas digestivas y nutrientes esenciales.
|
|              148 |       142 |             1 |         105 | 2020-08-09           | 2024-03-17
|        150 | Alfalfa germinada           | Germinado                  | Alfalfa germinada:
Brotes ricos en clorofila, enzimas digestivas y nutrientes esenciales.
|
|              149 |       141 |           106 |          52 | 2020-08-10           | 2024-03-18
|        150 | Alfalfa germinada           | Germinado                  | Alfalfa germinada:
Brotes ricos en clorofila, enzimas digestivas y nutrientes esenciales.
|
|              150 |       127 |            59 |           6 | 2020-08-11           | 2024-03-19
|        150 | Alfalfa germinada           | Germinado                  | Alfalfa germinada:
Brotes ricos en clorofila, enzimas digestivas y nutrientes esenciales.
|
+---------------------+-----------+----------------+-------------+-----------------------+-----------------
----+-------------+-----------------------------+----------------------------+---------------------------
----------------------------------------------------------------------------------------+
```

22200 rows in set (0.074 sec)

MariaDB [odbcukumari]> DROP TABLE IF EXISTS result1;

Query OK, 0 rows affected (0.011 sec)

BIBLIOGRAFIAS

1. Wikipedia. (s.f.). Álgebra relacional. Recuperado el 23 de mayo de 2025, de https://es.wikipedia.org/wiki/%C3%81lgebra_relacional

2. Microsoft Support. (s.f.). Mantener la integridad referencial en diagramas de modelo de base de datos. Recuperado el 23 de mayo de 2025, de https://support.microsoft.com/es-es/office/mantener-la-integridad-referencial-en-diagramas-de-modelo-de-base-de-datos-80f60e10-1238-48f7-ab59-2bd31b2f047a

3. Universidad Europea. (s.f.). Lenguaje de programación SQL. Recuperado el 23 de mayo de 2025, de https://universidadeuropea.com/blog/lenguaje-programacion-sql/

4. EBAC. (s.f.). Normalización de bases de datos. Recuperado el 23 de mayo de 2025, de https://ebac.mx/blog/normalizacion-de-bases-de-datos

5. Stack Overflow en español. (2016, octubre 19). Oracle: Clave primaria vs. Clave única vs. Índice único. Recuperado el 23 de mayo de 2025, de https://es.stackoverflow.com/questions/28696/oracle-clave-primaria-vs-clave-%C3%BAnica-vs-%C3%ADndice-%C3%BAnico

6. Lucidchart. (s.f.). ¿Qué es un diagrama entidad-relación?. Recuperado el 23 de mayo de 2025, de https://www.lucidchart.com/pages/es/que-es-un-diagrama-entidad-relacion

7. UNIR México. (s.f.). Modelo entidad-relación. Recuperado el 23 de mayo de 2025, de https://mexico.unir.net/noticias/ingenieria/modelo-entidad-relacion/

8. ILERNA. (s.f.). Modelo entidad-relación en base de datos. Recuperado el 23 de mayo de 2025, de https://www.ilerna.es/blog/modelo-entidad-relacion-base-datos