**Consultas y Vistas**

*“Cada día trae una nueva oportunidad”*

**Primera actividad:** Utilizando el ejercicio de la Librería realizado en clase (se adjunta script SQL) realice lo siguiente:

* Complete la información para las tablas autor, libro, cliente, editorial, libro\_cliente, libro\_autor y teléfono\_cliente con al menos (5,20,7,4,10,10, 12) registros respectivamente usando **unicamente** comandos SQL creados por usted.
* realice 5 consultas que me permitan conocer el nombre y la fecha de nacimiento de cada escritor, la cantidad de libros diferentes vendidos, el nombre de su cliente acompañado de su numero teléfonico, el nombre del libro acompañado por su autor o sus autores, el nombre de las editoriales que han logrado vender libros.
* Realice las dos vistas que considere sean las más importantes y explique el motivo de su selección.

**Segunda actividad:**

<https://github.com/Ococho/Hospital-GNECJ.git>

Utilizando el ejercicio del hospital realizado por sus compañeros realice lo siguiente:

* **Convierta el MR en una base de datos en MySQL utilizando sentencias SQL o el diagrama EER.**
* **Complete la información para las tablas realizadas con almenos 5 registros por tabla.**
* realice una consulta que me permita conocer que medicamentos a tomado cada paciente y la dosis suministrada.
* realice una consulta que me permita conocer que enfermeros estuvieron en los procedimientos de los pacientes.
* Realice las tres vistas que considere sean las más importantes y explique el motivo de su selección.

**Condiciones de la entrega:**

1. PDF con la solución donde se evidencie el paso a paso que lo llevo a la solución.
2. Archivos SQL necesarios para replicar la solución incluyendo las sentencias para insertar datos.
3. Excelente uso de los commits.
4. Despúes de realizar el trabajo responda ¿Qué le agregaría al modelo para dar mas información y esa información cúal seria?
5. Una vez realizado el PR se considera como entrega.
6. Hora de entrega 11.55 PM

Tercera actividad (TALLER 6):

Elabore 4 procedimientos almacenados que me permitan agregar, actualizar, consultar y borrar, en una de las tablas de la librería (primera actividad).

Elabore una nueva tabla llamada "control\_de\_cambios\_librería" la cual debe contener 3 columnas (usuario, accion, fecha) y guarde utilizando 2 Triggers el nombre del usuario que agrego o elimino un registro en la tabla seleccionada en el punto anterior.

Elabore 4 procedimientos almacenados que me permitan agregar, actualizar, consultar y borrar, en una de las tablas del Hospital (segunda actividad).

Elabore una nueva tabla llamada "control\_de\_cambios\_hospital" la cual debe contener 3 columnas (usuario, accion, fecha) y guarde utilizando 2 Triggers el nombre del usuario que agrego o elimino un registro en la tabla seleccionada en el punto anterior.

**SOLUCIÓN PRIMER ACTIVIDAD**

Script proporcionado por **Juan Pineda**

-- MySQL Workbench Forward Engineering

**SET** @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=**0**;

**SET** @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=**0**;

**SET** @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION';

-- -----------------------------------------------------

-- Schema LibreriaBuscaLibre

-- -----------------------------------------------------

-- -----------------------------------------------------

-- Schema LibreriaBuscaLibre

-- -----------------------------------------------------

**CREATE** **SCHEMA** IF **NOT** **EXISTS** `LibreriaBuscaLibre` **DEFAULT** CHARACTER **SET** utf8 ;

USE `LibreriaBuscaLibre` ;

-- -----------------------------------------------------

-- Table `LibreriaBuscaLibre`.`Editorial`

-- -----------------------------------------------------

**CREATE** **TABLE** IF **NOT** **EXISTS** `LibreriaBuscaLibre`.`Editorial` (

`nombre` VARCHAR(**50**) **NOT** **NULL**,

`ciudad` VARCHAR(**30**) **NOT** **NULL**,

`complemento` VARCHAR(**100**) **NOT** **NULL**,

`Telefono` VARCHAR(**20**) **NOT** **NULL** **DEFAULT** '6013909541',

**PRIMARY** **KEY** (`nombre`),

**UNIQUE** **INDEX** `nombre\_UNIQUE` (`nombre` **ASC**) VISIBLE)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `LibreriaBuscaLibre`.`libro`

-- -----------------------------------------------------

**CREATE** **TABLE** IF **NOT** **EXISTS** `LibreriaBuscaLibre`.`libro` (

`ISBN` VARCHAR(**10**) **NOT** **NULL**,

`titulo` VARCHAR(**45**) **NOT** **NULL**,

`numero\_paginas` VARCHAR(**45**) **NULL**,

`nombre\_editorial` VARCHAR(**50**) **NOT** **NULL**,

**PRIMARY** **KEY** (`ISBN`),

**INDEX** `nombre\_editorial\_idx` (`nombre\_editorial` **ASC**) VISIBLE,

**CONSTRAINT** `nombre\_editorial`

**FOREIGN** **KEY** (`nombre\_editorial`)

**REFERENCES** `LibreriaBuscaLibre`.`Editorial` (`nombre`)

**ON** **DELETE** **NO** ACTION

**ON** **UPDATE** **NO** ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `LibreriaBuscaLibre`.`cliente`

-- -----------------------------------------------------

**CREATE** **TABLE** IF **NOT** **EXISTS** `LibreriaBuscaLibre`.`cliente` (

`cedula` VARCHAR(**10**) **NOT** **NULL**,

`nombre` VARCHAR(**45**) **NULL**,

**PRIMARY** **KEY** (`cedula`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `LibreriaBuscaLibre`.`autor`

-- -----------------------------------------------------

**CREATE** **TABLE** IF **NOT** **EXISTS** `LibreriaBuscaLibre`.`autor` (

`id` VARCHAR(**10**) **NOT** **NULL**,

`fecha de nacimiento` VARCHAR(**45**) **NULL**,

`nacionalidad` VARCHAR(**20**) **NULL**,

`nombre` VARCHAR(**45**) **NULL**,

**PRIMARY** **KEY** (`id`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `LibreriaBuscaLibre`.`libro\_autor`

-- -----------------------------------------------------

**CREATE** **TABLE** IF **NOT** **EXISTS** `LibreriaBuscaLibre`.`libro\_autor` (

`ISBN\_libro` VARCHAR(**10**) **NOT** **NULL**,

`id\_autor` VARCHAR(**10**) **NOT** **NULL**,

**PRIMARY** **KEY** (`ISBN\_libro`, `id\_autor`),

**INDEX** `id\_autor\_idx` (`id\_autor` **ASC**) VISIBLE,

**CONSTRAINT** `id\_autor`

**FOREIGN** **KEY** (`id\_autor`)

**REFERENCES** `LibreriaBuscaLibre`.`autor` (`id`)

**ON** **DELETE** **NO** ACTION

**ON** **UPDATE** **NO** ACTION,

**CONSTRAINT** `ISBN\_libro`

**FOREIGN** **KEY** (`ISBN\_libro`)

**REFERENCES** `LibreriaBuscaLibre`.`libro` (`ISBN`)

**ON** **DELETE** **NO** ACTION

**ON** **UPDATE** **NO** ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `LibreriaBuscaLibre`.`libro\_cliente`

-- -----------------------------------------------------

**CREATE** **TABLE** IF **NOT** **EXISTS** `LibreriaBuscaLibre`.`libro\_cliente` (

`ISBN\_libro\_cliente` VARCHAR(**10**) **NOT** **NULL**,

`id\_cliente` VARCHAR(**10**) **NOT** **NULL**,

**PRIMARY** **KEY** (`ISBN\_libro\_cliente`, `id\_cliente`),

**INDEX** `id\_cliente\_idx` (`id\_cliente` **ASC**) VISIBLE,

**CONSTRAINT** `ISBN\_libro\_cliente`

**FOREIGN** **KEY** (`ISBN\_libro\_cliente`)

**REFERENCES** `LibreriaBuscaLibre`.`libro` (`ISBN`)

**ON** **DELETE** **NO** ACTION

**ON** **UPDATE** **NO** ACTION,

**CONSTRAINT** `id\_cliente`

**FOREIGN** **KEY** (`id\_cliente`)

**REFERENCES** `LibreriaBuscaLibre`.`cliente` (`cedula`)

**ON** **DELETE** **NO** ACTION

**ON** **UPDATE** **NO** ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `LibreriaBuscaLibre`.`telefono\_cliente`

-- -----------------------------------------------------

**CREATE** **TABLE** IF **NOT** **EXISTS** `LibreriaBuscaLibre`.`telefono\_cliente` (

`cedula\_cliente` VARCHAR(**10**) **NOT** **NULL**,

`numero` VARCHAR(**15**) **NOT** **NULL**,

**PRIMARY** **KEY** (`cedula\_cliente`, `numero`),

**CONSTRAINT** `cedula\_cliente`

**FOREIGN** **KEY** (`cedula\_cliente`)

**REFERENCES** `LibreriaBuscaLibre`.`cliente` (`cedula`)

**ON** **DELETE** **NO** ACTION

**ON** **UPDATE** **NO** ACTION)

ENGINE = InnoDB;

**SET** SQL\_MODE=@OLD\_SQL\_MODE;

**SET** FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS;

**SET** UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS;

**INSERCIÓN DE REGISTROS**

-- Tabla Autor

**INSERT** **INTO** `libreriabuscalibre`.`autor` (`id`, `fecha de nacimiento`, `nacionalidad`, `nombre`)

**VALUES**

('1', '12-02-2001', 'Colombiano', 'Pepito Perez'),

("2", "2000-02-22", "Colombiano", "Jesus de Nazaret"),

("3", "2001-08-33", "Venezolano", "Son Goku"),

("4", "2002-05-44", "Peruano", "Bart Simpson"),

("5", "2003-07-55", "Colombiano", "Pedro Alvarez");

-- Tabla cliente

**INSERT** **INTO** `libreriabuscalibre`.`cliente` (`cedula`, `nombre`)

**VALUES**

('1', 'Pepito Jimenez'),

('2', 'Marcela Sepulveda'),

('3', 'Krillin '),

('4', 'Neo Fonseca'),

('5', 'Mr Increible'),

('6', 'Lisa Simpson'),

('7', 'Homer Simpson');

-- Tabla editorial

**INSERT** **INTO** `libreriabuscalibre`.`editorial` (`nombre`, `ciudad`, `complemento`, `Telefono`)

**VALUES**

('Editorial Santy', 'Pereira', 'Calle1', '322911123'),

('Editorial Minuto de DIOS', 'Medellin', 'Calle 5', '3221244'),

('Editorial Letricas', 'Cali', 'Calle 8', '38923874'),

('Editorial Springfield', 'New York', 'Calle 9', '23144444');

-- Tabla libros

**INSERT** **INTO** `libreriabuscalibre`.`libro` (`ISBN`, `titulo`, `numero\_paginas`, `nombre\_editorial`)

**VALUES**

(**1**, "El alquimista", **200**, "Editorial Santy"),

(**2**, "Sueño en el poder", **150**, "Editorial Santy"),

(**3**, "La vida secreta de las abejas", **250**, "Editorial Santy"),

(**4**, "El código da Vinci", **300**, "Editorial Santy"),

(**5**, "Harry Potter y la piedra filosofal", **350**, "Editorial Minuto de DIOS"),

(**6**, "La isla del tesoro", **200**, "Editorial Minuto de DIOS"),

(**7**, "La Odisea", **250**, "Editorial Minuto de DIOS"),

(**8**, "Los hermanos Karamazov", **450**, "Editorial Minuto de DIOS"),

(**9**, "Cien años de soledad", **400**, "Editorial Letricas"),

(**10**, "Matar a un ruiseñor", **300**, "Editorial Letricas"),

(**11**, "Frankenstein o el moderno Prometeo", **250**, "Editorial Letricas"),

(**12**, "Dracula", **350**, "Editorial Letricas"),

(**13**, "El extraño caso del Dr. Jekyll y Mr. Hyde", **150**, "Editorial Letricas"),

(**14**, "El gran Gatsby", **200**, "Editorial Letricas"),

(**15**, "Moby Dick",**300**, "Editorial Springfield"),

(**16**, "El guardián entre el centeno", **200**, "Editorial Springfield"),

(**17**, "La naranja mecánica", **250**, "Editorial Springfield"),

(**18**, "1984", **300**, "Editorial Springfield"),

(**19**, "Un mundo feliz", **150**, "Editorial Springfield"),

(**20**, "El cuento de la criada", **250**, "Editorial Springfield");

-- Tabla LibroCliente

**INSERT** **INTO** `libreriabuscalibre`.`libro\_cliente` (`ISBN\_libro\_cliente`, `id\_cliente`)

**VALUES**

('1', '1'),

('2', '1'),

('3', '1'),

('1', '2'),

('1', '3'),

('1', '4'),

('4', '5'),

('4', '6'),

('5', '6'),

('6', '7');

-- Tabla Libro Autor

**INSERT** **INTO** `libreriabuscalibre`.`libro\_autor` (`ISBN\_libro`, `id\_autor`)

**VALUES**

('1', '1'),

('2', '1'),

('3', '2'),

('4', '2'),

('5', '3'),

('6', '3'),

('7', '4'),

('8', '4'),

('9', '5'),

('10', '5'),

('11', '1'),

('12', '1'),

('13', '1'),

('14', '2'),

('15', '2'),

('16', '3'),

('17', '3'),

('18', '4'),

('19', '4'),

('20', '5');

-- Registros telefono\_cliente

**INSERT** **INTO** `libreriabuscalibre`.`telefono\_cliente` (`cedula\_cliente`, `numero`)

**VALUES**

('1', '1213231'),

('2', '13231321'),

('3', '123233'),

('4', '13231'),

('5', '12313132'),

('6', '132132'),

('7', '12331321'),

('7', '1321321'),

('1', '12123'),

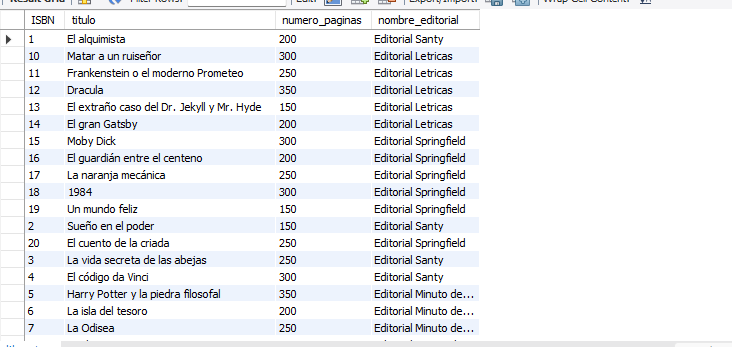
('2', '123123331'),

('1', '132313'),

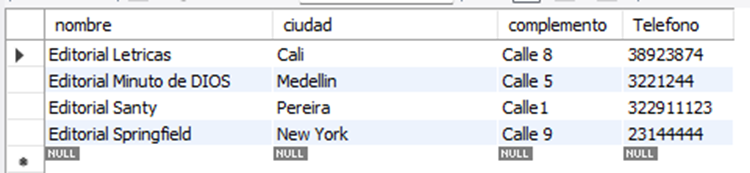
('2', '13213211');

A continuación, se visualizan los registros de dos de las tablas, si se desean ver los demás registros arriba están las inserciones de cada una de las tablas

**Imagen 1:** Registro Libros

****

**Imagen 2:** Registro editorial



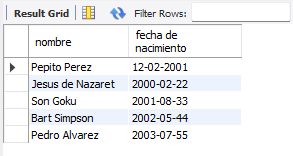
**CONSULTAS**

-- Consulta nombre y fecha de nacimiento de cada escritor

**SELECT** nombre, `fecha de nacimiento`

**FROM** LibreriaBuscaLibre.autor;

**Imagen 3:** Resultado consulta Nombre y Fecha de nacimiento de cada autor



-- Consulta nombre del libro con su autor o autores

**SELECT** titulo, GROUP\_CONCAT(nombre SEPARATOR ', ') **as** autores

**FROM** libro

**JOIN** libro\_autor

**ON** libro.ISBN = libro\_autor.ISBN\_libro

**JOIN** autor

**ON** libro\_autor.id\_autor = autor.id

**GROUP** **BY** titulo;

**Imagen 4:** Resultado Consulta nombre del libro con su autor o autores

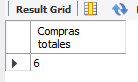
**Interfaz de usuario gráfica, Tabla

Descripción generada automáticamente**

-- Consulta cantidad de libros vendidos

**SELECT** **count**(**DISTINCT** ISBN\_libro\_cliente) **AS** "Contador de compras " **FROM** libro\_cliente;

**Imagen 5:** Resultado consulta cantidad de libros vendidos

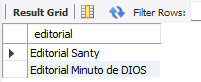
****

-- Nombre editoriales que han vendido libros

**SELECT** **DISTINCT** libro.nombre\_editorial **AS** "editorial"

**FROM** libro **INNER** **JOIN** libro\_cliente **ON** libro.ISBN = libro\_cliente.ISBN\_libro\_cliente;

**Imagen 6:** Resultado consulta editoriales que han vendido libros



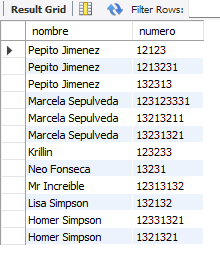
-- El nombre de su cliente acompañado de su número telefónico:

**SELECT** cliente.nombre, telefono\_cliente.numero

**FROM** cliente

**JOIN** telefono\_cliente **ON** cliente.cedula = telefono\_cliente.cedula\_cliente;

**Imagen 7:** Resultado consulta del nombre de los clientes con su número telefónico



**VISTAS**

Considero que una vista muy importante es la de poder visualizar los numeros de telefono de cada cliente, esta vista es importante ya que se puede visualizar de forma inmediata los telefonos de los clientes para enviarles publicidad o contactarlos directamente

-- Vistas TelefonosClienteVisualizar

**CREATE** **VIEW** TelefonoClientesVisualizar **AS**

**SELECT** cliente.nombre, telefono\_cliente.numero

**FROM** cliente

**JOIN** telefono\_cliente **ON** cliente.cedula = telefono\_cliente.cedula\_cliente;

**Imagen 8:** Vista TelefonosClientesVisualizar



Otra vista muy importante es la de tener disponibles el nombre del autor con sus respectivas obras, si por ejemplo una persona viene buscando a un autor y quiere saber que libros ha escrito, puede resultar muy útil para la persona encargada tener acceso rápidamente a estos datos

**CREATE** **VIEW** VistaAutorLibro **AS**

**SELECT** titulo, GROUP\_CONCAT(nombre SEPARATOR ', ') **as** autores

**FROM** libro

**JOIN** libro\_autor

**ON** libro.ISBN = libro\_autor.ISBN\_libro

**JOIN** autor

**ON** libro\_autor.id\_autor = autor.id

**GROUP** **BY** titulo;

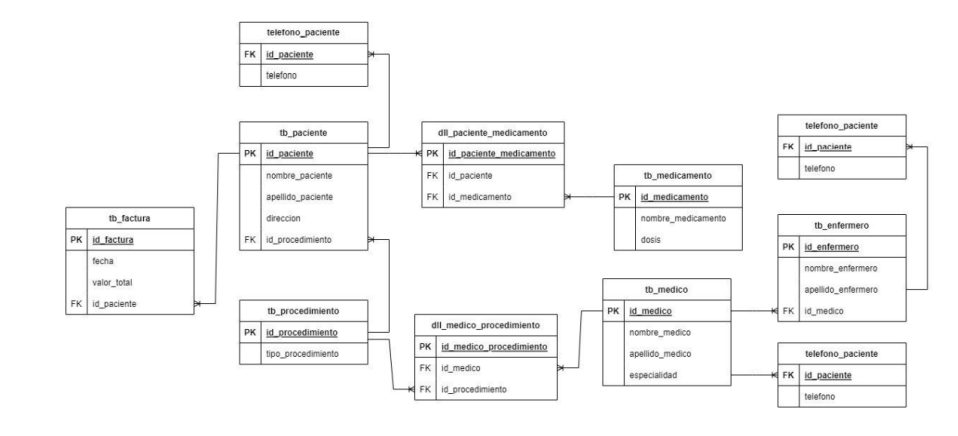
**Imagen 9:** Vista VistaAutorLibro



**SOLUCIÓN ACTIVIDAD 2**

Para la actividad 2 se toma como referencia el diagrama que elaboró el compañero

**Imagen 10:** Modelo relacional Hospital Enfermitas

****

Código de las tablas creadas

**CREATE** **DATABASE** IF **NOT** **EXISTS** HospitalEnfermitas;

USE HospitalEnfermitas;

**Create** **Table** Medico(

idMedico varchar(**20**) **primary** **key**,

nombreMedico varchar(**50**),

apellidoMedico varchar(**50**),

especialidad varchar(**60**)

);

**Create** **Table** TelefonoMedico(

idMedico varchar(**20**),

telefono varchar(**50**),

**primary** **key**(idMedico,telefono),

**foreign** **key**(idMedico) **references** Medico(idMedico)

);

**Create** **Table** Enfermero(

idEnfermero varchar(**20**) **primary** **key**,

nombreEnfermero varchar(**50**),

apellidoEnfermero varchar(**50**),

idMedico varchar(**20**),

**foreign** **key**(idMedico) **references** Medico(idMedico)

);

**Create** **Table** TelefonoEnfermero(

idEnfermero varchar(**20**),

telefono varchar(**50**),

**primary** **key**(idEnfermero,telefono),

**foreign** **key**(idEnfermero) **references** Enfermero(idEnfermero)

);

**Create** **Table** Procedimiento(

idProcedimiento varchar(**20**) **primary** **key**,

tipoProcedimiento varchar(**50**)

);

**Create** **Table** RelacionMedicoProcedimiento(

idMedico varchar(**20**),

idProcedimiento varchar(**20**),

**primary** **key**(idMedico,idProcedimiento),

**foreign** **key**(idMedico) **references** Medico(idMedico),

**foreign** **key**(idProcedimiento) **references** Procedimiento(idProcedimiento)

);

**Create** **Table** Paciente(

idPaciente varchar(**20**) **primary** **key**,

nombrePaciente varchar(**50**),

apellidoPaciente varchar(**50**),

direccion varchar(**50**),

idProcedimiento varchar(**20**),

**foreign** **key**(idProcedimiento) **references** Procedimiento(idProcedimiento)

);

**Create** **Table** TelefonoPaciente(

idPaciente varchar(**20**),

telefono varchar(**50**),

**primary** **key**(idPaciente,telefono),

**foreign** **key**(idPaciente) **references** Paciente(idPaciente)

);

**Create** **Table** Facture(

idFactura varchar(**20**) **primary** **key**,

fecha varchar(**20**),

valorTotal double,

idPaciente varchar(**20**),

**foreign** **key**(idPaciente) **references** Paciente(idPaciente)

);

**Create** **Table** Medicamento(

idMedicamento varchar(**20**) **primary** **key**,

nombreMedicamento varchar(**50**),

dosis varchar(**50**)

);

**Create** **Table** RelacionPacienteMedicamento(

idPaciente varchar(**20**),

idMedicamento varchar(**20**),

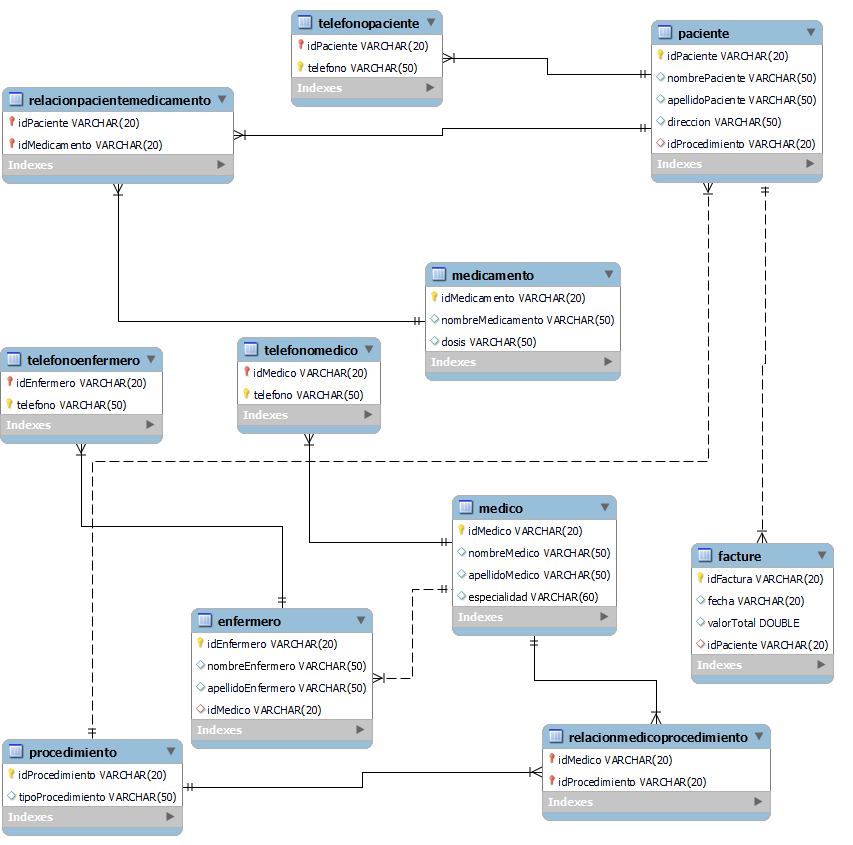
**primary** **key**(idPaciente,idMedicamento),

**foreign** **key**(idPaciente) **references** Paciente(idPaciente),

**foreign** **key**(idMedicamento) **references** Medicamento(idMedicamento)

);

Usando la opción que nos proporciona Workbench de Reverse Engineer obtenemos el siguiente Modelo relacional



**REGISTROS INGRESADOS**

-- Registros Medicos

**INSERT** **INTO** Medico (`idmedico`,`nombreMedico`,`apellidoMedico`,`especialidad`) **values**

("1","Santy","Ramirez","Optamologia"),

("2","Pablo","Castro","Dentista"),

("3","Bart","Simpson","Pediatria"),

("4","Goku","Son","Pedriatria"),

("5","Pedro","Perez","Traumas ");

-- Registros telefono Medico

**insert** **into** telefonoMedico(`idMedico`,`telefono`)

**values**

("1","12345"),

("1","123456"),

("2","23456"),

("3","3456"),

("4","456"),

("5","56");

-- Registros enfermero

**insert** **into** Enfermero (`idEnfermero`,`nombreEnfermero`,`apellidoEnfermero`,`idMedico`)

**values**

("1", "Pepita", "Perez", "1"),

("2", "Elma", "Tillo", "2"),

("3", "Tortu", "Guita", "3"),

("4", "Donald", "Trump", "4"),

("5", "Josefa", "Arenas", "5");

-- Registros telefono Enfermero

**insert** **into** telefonoEnfermero(`idEnfermero`,`telefono`)

**values**

("1","123456789"),

("2","23456789"),

("3","3456789"),

("3","345678910"),

("4","456789"),

("4","45678910"),

("5","56789");

-- Registros Procedimiento

**insert** **into** Procedimiento(`idProcedimiento`,`tipoProcedimiento`)

**values**

("1","Prevencion"),

("2","Cita medica de seguimiento"),

("3","Control diabetes"),

("4","Control reumatologia"),

("5","Intervencion quirurgica");

-- Resgistros relacion Medico Procedimiento

**insert** **into** relacionMedicoProcedimiento(`idMedico`,`idProcedimiento`)

**values**

("1","1"),

("2","2"),

("3","3"),

("4","4"),

("5","5");

**insert** **into** Paciente(`idPaciente`,`nombrePaciente`,`apellidoPaciente`,`direccion`,`idProcedimiento`)

**values**

("1","Chuchito","Nazaret","El cielo","1"),

("2","Matilda","Gonzales"," La granjita","2"),

("3","Karl","Marx","El paraiso","3"),

("4","Homero","Simpson","SpringField","4"),

("5","Naruto","Uzumaki","La aldea de la hoja","5");

-- Registros telefono pacientes

**insert** **into** telefonoPaciente(`idPaciente`,`telefono`)

**values**

("1","123456789"),

("2","23456789"),

("3","3456789"),

("4","456789"),

("5","56789");

-- Regustros Factura

**insert** **into** Facture(`idFactura`,`fecha`,`valorTotal`,`idPaciente`)

**values**

("1","01/04/1954",**0**,"1"),

("2","02/02/2222",**0**,"2"),

("3","03/03/3333",**30000**,"3"),

("4","06/12/3333",**40000**,"4"),

("5","10/10/1998",**50000**,"5");

-- Registros medicamento

**insert** **into** Medicamento(`idMedicamento`,`nombreMedicamento`,`dosis`)

**values**

("1","Acetaminofen"," 4 al dia"),

("2","La bendicion de la abuela","Todos los dias"),

("3","Noraber","1 vez al dia"),

("4","Medicamento para la presion arterial"," 3 veces al dia"),

("5","Gotas para los ojos","1 vez al dia ");

-- Registros relacion paciente medicamento

**insert** **into** relacionPacienteMedicamento(`idPaciente`,`idMedicamento`)

**values**

("1","1"),

("2","2"),

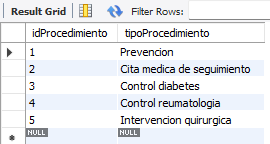
("3","3"),

("4","4"),

("5","5");

A continuación se muestran imágenes de cómo quedaron los registros

**Imagen 11:** Registros de la tabla procedimiento



**Imagen 12:** Registros de la tabla Medico

Tabla

Descripción generada automáticamente

**CONSULTAS**

**Consulta 1**

Consulta que me permita conocer que medicamentos a tomado cada paciente y la dosis suministrada

**select** Paciente.nombrePaciente **as** "Paciente", Medicamento.nombreMedicamento **as** "Medicamento ",Medicamento.dosis **as** "Dosis"

**from** Paciente

**inner** **join** relacionPacienteMedicamento **on** idPaciente=relacionPacienteMedicamento.idPacienteFK

**inner** **join** Medicamento **on** idMedicamento = relacionPacienteMedicamento.idMedicamentoFK;

* Daré una descripción de la lógica de esta consulta

La consulta SQL proporcionada utiliza la cláusula **INNER JOIN** para combinar tres tablas: **Paciente**, **relacionPacienteMedicamento** y **Medicamento**. El resultado final es una lista de nombres de pacientes, nombres de medicamentos y dosis de medicamentos que se han prescrito para los pacientes.

El proceso de unión se realiza en dos etapas, primero se unen las tablas **Paciente** y **relacionPacienteMedicamento** usando la clave primaria **idPaciente** de la tabla **Paciente** y la clave foránea **idPacienteFK** de la tabla **relacionPacienteMedicamento**. Esto permite que la información relacionada con el paciente y su medicación prescrita se combine.

Luego, la tabla **Medicamento** se une a esta combinación a través de la clave primaria **idMedicamento** de la tabla **Medicamento** y la clave foránea **idMedicamentoFK** de la tabla **relacionPacienteMedicamento**. Esto permite que la información relacionada con el medicamento se agregue a la combinación previa de información del paciente y su medicación prescrita.

Por último, la consulta selecciona los campos **nombrePaciente** y **nombreMedicamento** de las tablas **Paciente** y **Medicamento** respectivamente, así como el campo **dosis** de la tabla **Medicamento**. El resultado de la selección se mostrará con los encabezados "Paciente", "Medicamento" y "Dosis" respectivamente.

**Imagen 13:** Consulta que me permita conocer que medicamentos a tomado cada paciente y la dosis suministrada

**Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente**

También se podría realizar de la siguiente forma. Esta consulta utiliza la sintaxis de SQL tradicional que se basa en la cláusula "WHERE" para unir las tablas. Esto devuelve una tabla que muestra el nombre de cada paciente y el medicamento que está tomando, junto con la dosis. Es más optimo usar esta forma de estructurar las consultas si se trata de hacer la unión de muchas tablas, ya que resulta más legible para el programador

**SELECT**

p.nombrePaciente **AS** "Paciente",

m.nombreMedicamento **AS** "Medicamento",

m.dosis **AS** "Dosis"

**FROM**

Paciente p,

Medicamento m,

relacionPacienteMedicamento r

**WHERE**

p.idPaciente = r.idPacienteFK

**AND** m.idMedicamento = r.idMedicamentoFK;

**Consulta 2**

Consulta que me permita conocer que enfermeros estuvieron en los procedimientos de los pacientes

En esta consulta, la cláusula WHERE se utiliza para filtrar los resultados y mostrar solo los registros donde el nombre del enfermero no es nulo. Esto se logra usando la condición "Enfermero.nombreEnfermero IS NOT NULL". Las cláusulas SELECT y FROM se utilizan para seleccionar las columnas y las tablas necesarias para la consulta.

**SELECT** Enfermero.nombreEnfermero **as** "Enfermero", Paciente.nombrePaciente **as** "Paciente", Procedimiento.tipoProcedimiento **as** "Procedimiento"

**FROM** Paciente

**INNER** **JOIN** Procedimiento **ON** idProcedimiento = Procedimiento.idProcedimiento

**INNER** **JOIN** relacionMedicoProcedimiento **ON** idProcedimiento= relacionMedicoProcedimiento.idProcedimientofkRMP

**INNER** **JOIN** Medico **ON** idMedico = relacionMedicoProcedimiento.idMedicofkRMP

**INNER** **JOIN** Enfermero **ON** idMedicofkM = Medico.idMedico

**WHERE** Enfermero.nombreEnfermero **IS** **NOT** **NULL**;

**Imagen 14:** Consulta que me permita conocer que enfermeros estuvieron en los procedimientos de los pacientes

Interfaz de usuario gráfica, Aplicación, Tabla

Descripción generada automáticamente

**Vistas**

Una consulta muy común es la de saber que está tomando cada paciente y en qué dosis lo está tomando, para así saber si los médicos deben recetar una mayor cantidad o aumentar la dosis del medicamento, también puede servir para saber que medicamentos le son de ayuda o no.

-- Vista historia clinica

**SELECT** \* **FROM** HistoriaClinica;

**CREATE** **VIEW** HistoriaClinica **as**

**SELECT** Paciente.nombrePaciente **as** "Paciente", Paciente.direccion **as** "Direccion", Medicamento.nombreMedicamento **as** "Medicamento", Medicamento.dosis **as** "Dosis"

**FROM** Paciente **INNER** **JOIN** relacionPacienteMedicamento **on** idPaciente = relacionPacienteMedicamento.idPacienteFK

**INNER** **JOIN** Medicamento **on** idMedicamento = relacionPacienteMedicamento.idMedicamentoFK;

**Imagen 15:** Vista para el seguimiento del paciente (Historia clinica)

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Vista

Una consulta muy común es saber qué personal está involucrado en un procedimiento, si por ejemplo es una cirugía se necesitan los datos del paciente y del personal a cargo de la cirugía, para que estuviera completa quizás agregaría un atributo llamado fecha para llevar un mejor registro

**CREATE** **VIEW** PersonalInvolucradoCirugia **AS**

**SELECT** Paciente.nombrePaciente **AS** "Paciente", Medico.nombreMedico "Medico", Enfermero.nombreEnfermero **AS** "Enfermero", Procedimiento.tipoProcedimiento **AS** "Procedimiento"

**FROM** Paciente **INNER** **JOIN** Procedimiento **ON** idProcedimientofkPaciente = Procedimiento.idProcedimiento

**INNER** **JOIN** relacionMedicoProcedimiento **ON** idProcedimiento= relacionMedicoProcedimiento.idProcedimientofkRMP

**INNER** **JOIN** Medico **ON** idMedico = relacionMedicoProcedimiento.idMedicofkRMP

**INNER** **JOIN** Enfermero **ON** idMedicofkM = Medico.idMedico;

**Imagen 16:** Vista para el seguimiento del personal a cargo de una cirugía

Tabla

Descripción generada automáticamente

Vista

Una vista muy importante para el personal de contacto es tener a la mano los números de los pacientes para llamarlos y solicitarles información o también para avisarles sobre el agendamiento de citas, reclamar medicamentos etc

-- Vista para llamar a los pacientes

**CREATE** **VIEW** VistaTelefonoPaciente **AS**

**SELECT** Paciente.nombrePaciente, Telefono

**FROM** Paciente

**JOIN** TelefonoPaciente **ON** Paciente.idPaciente = TelefonoPaciente.idPacienteFK;

**Imagen 17:** Vista teléfonos pacientes

Tabla

Descripción generada automáticamente

**Pregunta final**

¿Qué le agregaría al modelo para dar más información y esa información cual sería?

Le agregaría una sección de fechas para los procedimientos como dije anteriormente, esto ayudaría a llevar un mejor registro de cuando se realiza el procedimiento, lo mismo para la tabla medicamentos, agregaría una fecha o mejor, un tiempo de duración en el cual debe usar el medicamento. Si ya se desea llevar a la parte contable pues se ingresaría atributos de precio a los medicamentos, quizás un sueldo para el personal, también un estrato que tenga cada paciente y según el estrato se reducen los gastos de los medicamentos, aunque como se dijo anteriormente no se debe combinar la lógica con los datos como tal.

**TERCER ACTIVIDAD**

Este es un procedimiento almacenado de MySQL llamado "AgregarNuevoCliente" que se utiliza para insertar nuevos clientes en una tabla llamada "cliente". El procedimiento tiene dos parámetros de entrada: "CedulaLocal" y "NombreLocal", que se utilizan para especificar la cédula y el nombre del nuevo cliente que se va a insertar en la tabla "cliente".

-- Procedimiento agregar usuario

**DELIMITER** //

**CREATE** **PROCEDURE** AgregarNuevoCliente ( **IN** CedulaLocal VARCHAR(**10**), **IN** NombreLocal VARCHAR(**45**))

**BEGIN**

**INSERT** **INTO** cliente (cedula, nombre) **VALUES** (CedulaLocal, NombreLocal);

**END**//

**DELIMITER** ;

**CALL** AgregarNuevoCliente("000000", "Santy");

**Imagen 18:** Procedimiento de agregar cliente

**Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente**

Este es un procedimiento almacenado de MySQL llamado "ActualizarCliente" que se utiliza para actualizar el nombre de un cliente existente en una tabla llamada "cliente". El procedimiento tiene dos parámetros de entrada: "cedulaLocal" y "nombreLocal", que se utilizan para especificar la cédula y el nuevo nombre del cliente que se va a actualizar en la tabla "cliente".

**DELIMITER** //

**CREATE** **PROCEDURE** ActualizarCliente (**IN** cedulaLocal VARCHAR(**10**),**IN** nombreLocal VARCHAR(**45**))

**BEGIN**

**UPDATE** cliente **SET** nombre = nombreLocal **WHERE** cedula = cedulaLocal;

**END**//

**DELIMITER** ;

**CALL** ActualizarCliente('000000', 'Tortuguita');;

**Imagen 19:** Procedimiento actualizar cliente

Interfaz de usuario gráfica, Tabla

Descripción generada automáticamente

Procedimiento para eliminar cliente por cedula

**DELIMITER** //

**CREATE** **PROCEDURE** EliminarCliente (**IN** cedulaLocal VARCHAR(**10**))

**BEGIN**

**DELETE** **FROM** cliente **WHERE** cedula = cedulaLocal;

**END**//

**DELIMITER** ;

**CALL** EliminarCliente('000000');

**Imagen 20:** Procedimiento eliminar cliente

Interfaz de usuario gráfica, Aplicación, Tabla

Descripción generada automáticamente

Procedimiento para consultar cliente por cedula

**CREATE** **PROCEDURE** ConsultarCliente (**IN** cedulaLocal VARCHAR(**10**))

**BEGIN**

**SELECT** \* **FROM** cliente **WHERE** cedula = cedulaLocal;

**END**//

**DELIMITER** ;

**CALL** ConsultarCliente('7');

**Imagen 21:** Consular cliente por cedula

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza baja

* Creación de la table ControlCambiosLibreria

**CREATE** **TABLE** IF **NOT** **EXISTS** ControlCambiosLibreria (

usuario VARCHAR(**45**) **NOT** **NULL**,

accion VARCHAR(**10**) **NOT** **NULL**,

fecha DATETIME **NOT** **NULL**,

**PRIMARY** **KEY** (usuario, accion, fecha));

-- Trigger de insertar cliente

**DELIMITER** //

**CREATE** **TRIGGER** TriggerInsertarCliente

**AFTER** **INSERT** **ON** cliente

**FOR** **EACH** **ROW**

**BEGIN**

**INSERT** **INTO** ControlCambiosLibreria (usuario, accion, fecha)

**VALUES** (**USER**(), "Agregar", NOW());

**END**//

**DELIMITER** ;

-- Trigger de eliminar

**DELIMITER** //

**CREATE** **TRIGGER** TriggerEliminarClienye

**AFTER** **DELETE** **ON** cliente

**FOR** **EACH** **ROW**

**BEGIN**

**INSERT** **INTO** ControlCambiosLibreria (usuario, accion, fecha)

**VALUES** (**USER**(), "Eliminar", NOW());

**END**//

**DELIMITER** ;

**Imagen 22:** Historial de agregar cliente y eliminar cliente usando Triggers

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Procedimientos para el Hospital Enfermitas

Procedimiento para agregar

-- Procedimiento para agregar procedimiento medico

**DELIMITER** //

**CREATE** **PROCEDURE** AgregarProcedimiento (**IN** idProcedimientoLocal VARCHAR(**20**),**IN** tipoProcedimientoLocal VARCHAR(**50**))

**BEGIN**

**INSERT** **INTO** Procedimiento (idProcedimiento, tipoProcedimiento) **VALUES** (idProcedimientoLocal, tipoProcedimientoLocal);

**END**//

**DELIMITER** ;

**CALL** AgregarProcedimiento("100000", "Cita de la tercera edad");

**Imagen 23:** Procedimiento agregar procedimiento medico

Tabla

Descripción generada automáticamente

-- Procedimiento actualizar procedimiento medico

**DELIMITER** //

**CREATE** **PROCEDURE** ActualizarProcedimiento (**IN** idProcedimientoLocal VARCHAR(**20**),**IN** tipoProcedimientoLocal VARCHAR(**50**))

**BEGIN**

**UPDATE** Procedimiento **SET** idProcedimiento = idProcedimientoLocal **WHERE** tipoProcedimiento = tipoProcedimientoLocal;

**END**//

**DELIMITER** ;

**CALL** ActualizarProcedimiento('2', 'Operacion de corazon');

**Imagen 22:** Procedimiento actualizar procedimiento medico

Tabla

Descripción generada automáticamente

-- Procedimiento borrar procedimineto medico

**DELIMITER** //

**CREATE** **PROCEDURE** BorrarProcedimiento (**IN** idProcedimientoLocal VARCHAR(**20**))

**BEGIN**

**DELETE** **FROM** Procedimiento **WHERE** idProcedimiento = idProcedimientoLocal ;

**END**//

**DELIMITER** ;

**CALL** BorrarProcedimiento('100000');

**Imagen 23:** Procedimiento eliminar procedimiento medico

Tabla

Descripción generada automáticamente

-- Procedimiento consultar procedimiento medico

**DELIMITER** //

**CREATE** **PROCEDURE** ConsultarProcedimiento (**IN** idProcedimientoLocal VARCHAR(**20**))

**BEGIN**

**SELECT** \* **FROM** Procedimiento **WHERE** idProcedimiento = idProcedimientoLocal ;

**END**//

**DELIMITER** ;

**CALL** ConsultarProcedimiento('1');

**Imagen 24:** Procedimiento consultar procedimiento medico

Tabla

Descripción generada automáticamente

**Tabla para registrar el control de cambios del procedimiento medico**

**CREATE** **TABLE** IF **NOT** **EXISTS** ControlCambiosProcedimientoMedico (

usuario VARCHAR(**45**) **NOT** **NULL**,

accion VARCHAR(**10**) **NOT** **NULL**,

fecha DATETIME **NOT** **NULL**,

**PRIMARY** **KEY** (usuario, accion, fecha)

);

Triggers eliminar y crear en la tabla de procedimientos médicos

-- TRigger para saber quien ingreso nuevos procedimientos medicos

**DELIMITER** //

**CREATE** **TRIGGER** TriggerIngresarProcedimientosMedicos

**AFTER** **INSERT** **ON** Procedimiento

**FOR** **EACH** **ROW**

**BEGIN**

**INSERT** **INTO** TriggerIngresarProcedimientosMedicos (usuario, accion, fecha)

**VALUES** (**USER**(), "Agrego", NOW());

**END**//

**DELIMITER** ;

-- Trigger para saber quien elimino datos en la tabla de procedimientos medicos

**DELIMITER** //

**CREATE** **TRIGGER** TriggerEliminarProcedimientosMedicos

**AFTER** **DELETE** **ON** Procedimiento

**FOR** **EACH** **ROW**

**BEGIN**

**INSERT** **INTO** TriggerEliminarProcedimientosMedicos (usuario, accion, fecha)

**VALUES** (**USER**(), "Elimino", NOW());

**END**//

**DELIMITER** ;

**Imagen 25:** Trigger historial de Agregación de datos y de eliminación en la tabla Procedimiento

Interfaz de usuario gráfica, Aplicación, Tabla

Descripción generada automáticamente