# **Consultas, Vistas, Procedimientos y Trigger**

Contents

[Taller 5 1](#_Toc127289365)

[Sentencias para insertar datos en libreriabuscaLibre 1](#_Toc127289366)

[Consultas en libreríaBuscaLibre 4](#_Toc127289367)

[Vista importantes para librería busca libre 6](#_Toc127289368)

[MER hospital 7](#_Toc127289369)

[Script para crear base de datos 8](#_Toc127289370)

[Script para insertar datos en base de datos hospital 11](#_Toc127289371)

[Sentencias para consultas solicitadas 13](#_Toc127289372)

[Vista necesarias en hospital 13](#_Toc127289373)

[Respuesta a pregunta 15](#_Toc127289374)

[Taller 6 15](#_Toc127289375)

[Primera actividad 15](#_Toc127289376)

[Procedimiento para agregar clientes 15](#_Toc127289377)

[Procedimiento para actualizar clientes 15](#_Toc127289378)

[Procedimiento para consultar clientes 16](#_Toc127289379)

[Procedimiento para eliminar clientes por el parámetro cedula. 16](#_Toc127289380)

[Sentencias sql para crear table control\_de\_cambios\_librería 16](#_Toc127289381)

[Trigger para registrar el usuario que registro los datos en la tabla clientes. 17](#_Toc127289382)

[Trigger para registrar el usuario que registro los datos en la tabla clientes. 17](#_Toc127289383)

[Segunda actividad 17](#_Toc127289384)

[Procedimiento para agregar procedimiento medico 17](#_Toc127289385)

[Procedimiento para actualizar procedimiento medico 18](#_Toc127289386)

[Procedimiento para consultar procedimiento medico 18](#_Toc127289387)

[Procedimiento para eliminar procedimiento medico 18](#_Toc127289388)

[Sentencias sql para crear table control\_de\_cambios\_de procedimiento en hospital 19](#_Toc127289389)

[Trigger para control de cambios al insertar procedimientos medicos 19](#_Toc127289390)

[Trigger para control de cambios al eliminar registros de procedimientos medicos 19](#_Toc127289391)

# 

# Taller 5

## Sentencias para insertar datos en libreriabuscaLibre

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

VALUES ("1", "2000-01-01", "colombiano", "Ana Maria Lopez"),

("2", "2000-01-01", "colombiano", "Ana Maria Lopez"),

    ("3", "1998-01-01", "colombiano", "Jenny Hernandez"),

        ("4", "2000-01-01", "colombiano", "Liliana Paz"),

        ("5", "2000-01-01", "colombiano", "Juliana Paguay");

INSERT INTO  `LibreriaBuscaLibre`.`Editorial` ( `nombre`,`ciudad`, `complemento` ,`Telefono` )

VALUES("Vintage", "Cali", "La mejor editorial", "101010"),

    ("zeta", "Cali", "La mejor editorial", "401010"),

        ("Alba", "Bogota", "La mejor editorial de Bogotá", "201010"),

        ("Lectura", "Cali", "Leer es mi cuento", "301010");

INSERT INTO  `LibreriaBuscaLibre`.`cliente`(`cedula`,  `nombre`)

VALUES ("1088888", "Agela Cumbal"),

  ("101010101", "Juliana Rios"),

    ("202020202", "Ivan Lopez"),

    ("303030303", "Aura Lucia Soto"),

    ("404040404", "Jose Pinzon"),

    ("505050505", "Jairo Cultid"),

    ("606060606", "Marcela Paguay");

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

VALUES ("1088888", "317777777"),

("1088888", "317777778"),

("606060606",  "317777776"),

("101010101",  "317777775"),

("101010101",  "317777755"),

("303030303",   "317777776"),

("404040404",   "317777775"),

("505050505",   "317777774");

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

VALUES("123B", "El retrato", "120", "Vintage"),

    ("123C", "Lideres mundiales", "200", "Vintage"),

    ("123D", "El diario de Ana Frank", "30", "Vintage"),

    ("123E", "El alquimista", "180", "Vintage"),

    ("123F", "La republica", "200", "Vintage"),

    ("123G", "Simple", "200", "Vintage"),

    ("123H", "Cocina de tu vida", "200", "zeta"),

    ("123I", "Historia del arte", "500", "zeta"),

    ("123J", "Arte y fotografía", "150", "zeta"),

    ("123k", "Historia de la imágenes", "300", "zeta"),

    ("123L", "Historia del jazz", "90", "Alba"),

    ("123M", "Tu cerebro y la musica", "100", "Alba"),

    ("123N", "Plantas olvidadas", "150", "Alba"),

    ("123O", "Algas marinas", "99", "Alba"),

    ("123P", "Viviendo en el futuro", "100", "Lectura"),

    ("123Q", "Los innovadores", "100", "Lectura"),

    ("123R", "TIC en casa", "110", "Lectura"),

    ("123S", "Fisioterapia en casa", "100", "Lectura"),

    ("123T", "Manualidades en casa", "167", "Lectura"),

    ("123U", "Desarrollo industrial", "190", "Lectura"),

    ("123V", "El enigma y el espejo", "100", "Lectura");

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

VALUES ("123B",1),

("123C", "2"),

("123D","2"),

("123E","1"),

("123F", "3"),

("123F", "4"),

("123G", "4"),

("123G", "5"),

("123H","5"),

("123I", "1"),

("123J", "2"),

("123k", "3"),

("123L", "4"),

("123M", "4"),

("123N", "5"),

("123O", "5"),

("123P", "1"),

("123Q", "1"),

("123R", "2"),

("123S", "3"),

("123T", "4"),

("123U", "4"),

("123V", "3");

INSERT INTO  `LibreriaBuscaLibre`.`cliente`(`cedula`,  `nombre`)

VALUES ("1088888", "Agela Cumbal"),

  ("101010101", "Juliana Rios"),

    ("202020202", "Ivan Lopez"),

    ("303030303", "Aura Lucia Soto"),

    ("404040404", "Jose Pinzon"),

    ("505050505", "Jairo Cultid"),

    ("606060606", "Marcela Paguay");

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

VALUES    ("123J", "1088888" ),

    ("123k", "1088888"),

    ("123L", "101010101"),

    ("123M", "101010101"),

    ("123N", "202020202"),

    ("123O", "202020202"),

    ("123P", "303030303"),

    ("123Q", "303030303"),

    ("123R", "404040404"),

    ("123S", "505050505"),

    ("123T", "606060606"),

    ("123U", "606060606"),

    ("123V", "101010101");

## Consultas en libreríaBuscaLibre

1. conocer el nombre y la fecha de nacimiento de cada escritor.

SELECT nombre, `fecha de nacimiento`

FROM LibreriaBuscaLibre.autor;

Graphical user interface, text, application

Description automatically generated

Figura 1: Autor con fecha de nacimiento

1. la cantidad de libros diferentes vendidos

SELECT COUNT(\*) FROM libro\_cliente;



Figura : cantidad de libros vendidos

1. El nombre de su cliente acompañado de su número telefónico

SELECT cliente.nombre, telefono\_cliente.numero

FROM telefono\_cliente

INNER JOIN cliente ON telefono\_cliente.cedula\_cliente = cliente.cedula

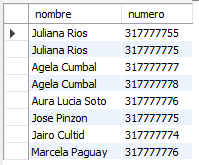


Figura :Telefono y cliente

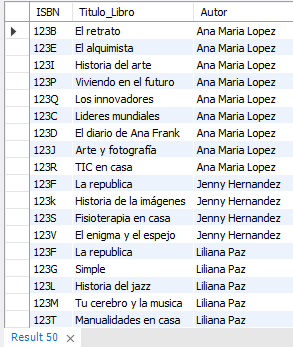
1. El nombre del libro acompañado por su autor o sus autores.

SELECT libro.ISBN, libro.titulo, autor.nombre

FROM libro

INNER JOIN libro\_autor ON libro.ISBN = libro\_autor.ISBN\_libro

INNER JOIN autor ON autor.id = libro\_autor.id\_autor;



Sentencias para agrupar autores por libro

1. El nombre de las editoriales que han logrado vender libros.

SELECT Editorial.nombre AS NombreEditorial, libro.titulo AS TituloLibro, cliente.nombre AS NombreCliente

FROM libro\_cliente

INNER JOIN libro ON libro\_cliente.ISBN\_libro\_cliente = libro.ISBN

INNER JOIN Editorial ON libro.nombre\_editorial = Editorial.nombre

INNER JOIN cliente ON libro\_cliente.id\_cliente = cliente.cedula;

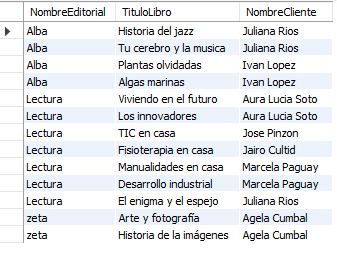


Figura : editorial, libro, cliente

## Vista importantes para librería busca libre

1. Es importante conocer los libros con los respectivos autores, ya que un cliente puede estar interesado en ciertos autores para la respectivacompra de un libro pero teniendo en cuenta el autor.

CREATE VIEW vista\_LibrosAutores AS

SELECT libro.ISBN, libro.titulo as Titulo\_Libro, GROUP\_CONCAT(autor.nombre SEPARATOR ', ') as Autores

FROM libro

INNER JOIN libro\_autor ON libro.ISBN = libro\_autor.ISBN\_libro

INNER JOIN autor ON autor.id = libro\_autor.id\_autor

GROUP BY libro.ISBN;

Graphical user interface, text, application

Description automatically generated

Figura : vista libro y autor

1. Vista telefono\_cliente, es importante esta vista porque genera una tabla de clientes con los números de teléfono y además agrupa los números de teléfono, ya que un cliente puede tener uno o más teléfonos, esto permite tener una información más concisa al momento de que la librería necesite contactar a un cliente.

CREATE VIEW telefono\_cliente AS

SELECT cliente.nombre, GROUP\_CONCAT(telefono\_cliente.numero SEPARATOR ', ') AS telefonos

FROM cliente

INNER JOIN telefono\_cliente ON cliente.cedula = telefono\_cliente.cedula\_cliente

GROUP BY cliente.cedula;

Graphical user interface, text, application, table

Description automatically generated

Figura : vista clientes y contactos

## MER hospital

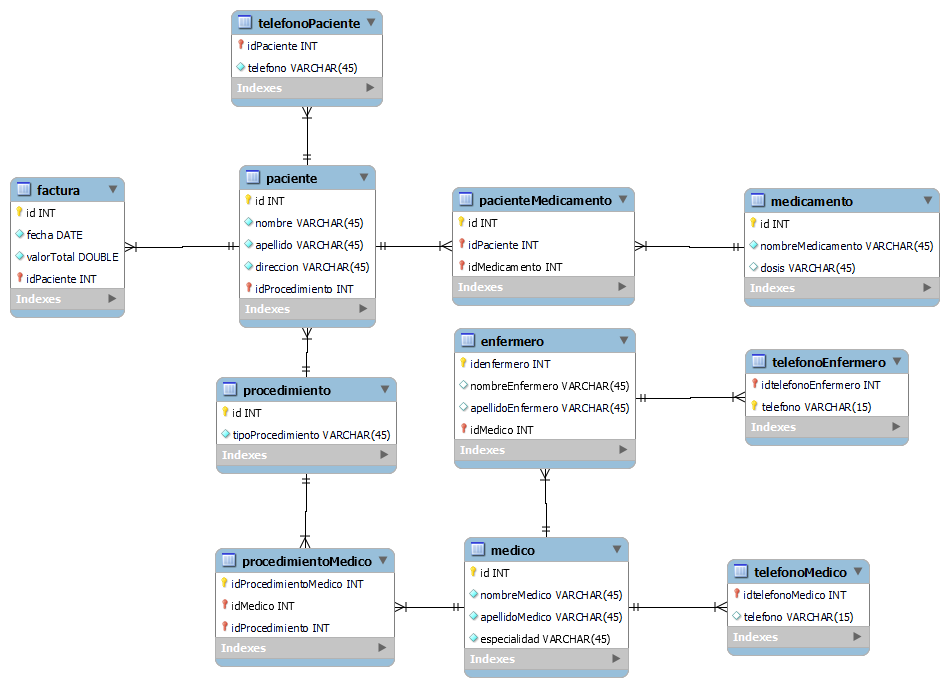


Figura : MER hospital

## Script para crear base de datos

CREATE SCHEMA IF NOT EXISTS `hospital` DEFAULT CHARACTER SET utf8 ;

USE `hospital` ;

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

-- Table `hospital`.`procedimiento`

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

CREATE TABLE IF NOT EXISTS `hospital`.`procedimiento` (

  `id` INT NOT NULL,

  `tipoProcedimiento` VARCHAR(45) NOT NULL,

  PRIMARY KEY (`id`))

ENGINE = InnoDB;

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

-- Table `hospital`.`paciente`

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

CREATE TABLE IF NOT EXISTS `hospital`.`paciente` (

  `id` INT NOT NULL,

  `nombre` VARCHAR(45) NOT NULL,

  `apellido` VARCHAR(45) NOT NULL,

  `direccion` VARCHAR(45) NOT NULL,

  `idProcedimiento` INT NOT NULL,

  PRIMARY KEY (`id`, `idProcedimiento`),

    FOREIGN KEY (`idProcedimiento`) REFERENCES `hospital`.`procedimiento` (`id`)

    )ENGINE = InnoDB;

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

-- Table `hospital`.`telefonoPaciente`

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

CREATE TABLE IF NOT EXISTS `hospital`.`telefonoPaciente` (

  `idPaciente` INT NOT NULL,

  `telefono` VARCHAR(45) NOT NULL,

  PRIMARY KEY (`idPaciente`),

  CONSTRAINT `idPaciente`

    FOREIGN KEY (`idPaciente`) REFERENCES `hospital`.`paciente` (`id`)

    )

ENGINE = InnoDB;

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

-- Table `hospital`.`factura`

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

CREATE TABLE IF NOT EXISTS `hospital`.`factura` (

  `id` INT NOT NULL,

  `fecha` DATE NOT NULL,

  `valorTotal` DOUBLE NOT NULL,

  `idPaciente` INT NOT NULL,

  PRIMARY KEY (`id`, `idPaciente`),

    FOREIGN KEY (`idPaciente`) REFERENCES `hospital`.`paciente` (`id`)

  )

ENGINE = InnoDB;

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

-- Table `hospital`.`medicamento`

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

CREATE TABLE IF NOT EXISTS `hospital`.`medicamento` (

  `id` INT NOT NULL,

  `nombreMedicamento` VARCHAR(45) NOT NULL,

  `dosis` VARCHAR(45) NULL,

  PRIMARY KEY (`id`))

ENGINE = InnoDB;

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

-- Table `hospital`.`pacienteMedicamento`

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

CREATE TABLE IF NOT EXISTS `hospital`.`pacienteMedicamento` (

  `id` INT NOT NULL,

  `idPaciente` INT NOT NULL,

  `idMedicamento` INT NOT NULL,

  PRIMARY KEY (`id`, `idMedicamento`, `idPaciente`),

    FOREIGN KEY (`idPaciente`) REFERENCES `hospital`.`paciente` (`id`),

    FOREIGN KEY (`idMedicamento`)  REFERENCES `hospital`.`medicamento` (`id`)

    )

ENGINE = InnoDB;

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

-- Table `hospital`.`medico`

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

CREATE TABLE IF NOT EXISTS `hospital`.`medico` (

  `id` INT NOT NULL,

  `nombreMedico` VARCHAR(45) NOT NULL,

  `apellidoMedico` VARCHAR(45) NOT NULL,

  `especialidad` VARCHAR(45) NOT NULL,

  PRIMARY KEY (`id`))

ENGINE = InnoDB;

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

-- Table `hospital`.`procedimientoMedico`

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

CREATE TABLE IF NOT EXISTS `hospital`.`procedimientoMedico` (

  `idProcedimientoMedico` INT NOT NULL,

  `idMedico` INT NOT NULL,

  `idProcedimiento` INT NOT NULL,

  PRIMARY KEY (`idProcedimientoMedico`, `idMedico`, `idProcedimiento`),

    FOREIGN KEY (`idProcedimiento`) REFERENCES `hospital`.`procedimiento` (`id`),

    FOREIGN KEY (`idMedico`) REFERENCES `hospital`.`medico` (`id`)

)

ENGINE = InnoDB;

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

-- Table `hospital`.`telefonoMedico`

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

CREATE TABLE IF NOT EXISTS `hospital`.`telefonoMedico` (

  `idtelefonoMedico` INT NOT NULL,

  `telefono` VARCHAR(15) NULL,

  PRIMARY KEY (`idtelefonoMedico`),

    FOREIGN KEY (`idtelefonoMedico`) REFERENCES `hospital`.`medico` (`id`)

  )

ENGINE = InnoDB;

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

-- Table `hospital`.`enfermero`

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

CREATE TABLE IF NOT EXISTS `hospital`.`enfermero` (

  `idenfermero` INT NOT NULL,

  `nombreEnfermero` VARCHAR(45) NULL,

  `apellidoEnfermero` VARCHAR(45) NULL,

  `idMedico` INT NOT NULL,

  PRIMARY KEY (`idenfermero`, `idMedico`),

    FOREIGN KEY (`idMedico`) REFERENCES `hospital`.`medico` (`id`)

)

ENGINE = InnoDB;

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

-- Table `hospital`.`telefonoEnfermero`

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

CREATE TABLE IF NOT EXISTS `hospital`.`telefonoEnfermero` (

  `idtelefonoEnfermero` INT NOT NULL,

  `telefono` VARCHAR(15) NOT NULL,

  PRIMARY KEY (`idtelefonoEnfermero`, `telefono`),

    FOREIGN KEY (`idtelefonoEnfermero`) REFERENCES `hospital`.`enfermero` (`idenfermero`)

)

ENGINE = InnoDB;

## Script para insertar datos en base de datos hospital

# Datos de tabla procedimiento.

INSERT INTO hospital.procedimiento (id, tipoProcedimiento)

VALUES ( 1, "Cirugia"),

    (2,"Odontologia"),

    (3, "Optometria"),

    (4, "Consulta medica"),

    (5, "Control prenatal");

# Datos de tabla paciente

INSERT INTO hospital.paciente( id, nombre, apellido, direccion, idProcedimiento)

VALUES (1, "Omaira", "Chinguad", " cr 2a -4, cali", 5),

    (2, "Lucia", "Moreno", " cr 2a -5, cali", 4),

    (3, "Rosa", "Alpala", " cr 5a -6, cali", 2),

    (4, "Martin", "Parra", " cr 2a -4, cali", 5),

    (5, "Yoli", "Guadir", " cr 5a -4, cali", 3);

# Datos de tabla telefonoPaciente

INSERT INTO hospital.telefonopaciente(idPaciente, telefono)

VALUES ( 1, "202020"),

    ( 2, "303030"),

    ( 4, "404040"),

    ( 3, "505050"),

    (5, "606060");

# Datos de la tabla factura

INSERT INTO hospital.factura(id,fecha,valorTotal, idPaciente)

VALUES( 1, "2022-02-12", 30000, 1),

    ( 2, "2022-01-12", 40000, 2),

    ( 3, "2022-02-02", 90000, 3),

    ( 4, "2022-02-05", 30000, 4),

    ( 5, "2022-02-11", 90000, 5);

#Datos de la tabla medicamento

INSERT INTO hospital.medicamento(id, nombreMedicamento, dosis)

VALUES(1, "medicamento1", "50 gramos por día"),

    (2, "medicamento2", "90 gramos por día"),

    (3, "medicamento3", "40 gramos por día"),

    (4, "medicamento4", "70 gramos por día"),

    (5, "medicamento5", "60 gramos por día");

#Datos de la tabla pacienteMedicamento

INSERT INTO hospital.pacientemedicamento(id, idPaciente,idMedicamento)

VALUES(1,2,3),

    (3,5,1),

    (2,4,5),

    (4,1,2),

    (5,3,4);

# Datos de la tabla medico

INSERT INTO hospital.medico(id, nombreMedico, apellidoMedico, especialidad)

VALUES (1, "Ana", "Mimalchi", "Cirujano"),

    (2, "Laura", "Galindo", "Odontoogo"),

    (3, "Matias", "Moreno", "Optómetra"),

    (4, "Santiago", "Tapie", "Medico general"),

    (5, "Samuel", "Diaz", "Pediatra");

#Datos de la tabla procedimientoMedico

INSERT INTO hospital.procedimientomedico(idProcedimientoMedico, idMedico,idProcedimiento)

VALUES(1,1,1),

(2,2,2),

(3,3,3),

(4,4,4),

(5,5,5);

#Datos telefono medico

INSERT INTO hospital.telefonomedico(idtelefonoMedico, telefono)

VALUES(1,"912345"),

    (2,"922345"),

    (3,"922245"),

    (4,"922225"),

    (5,"922335");

#Datos tabla enfermeros

INSERT INTO hospital.enfermero(idenfermero,nombreEnfermero,apellidoEnfermero,idMedico)

VALUES(1, "Liliana", "Colimba", 1),

    (2, "Viviana", "Iquinas", 2),

    (3, "Jhoana", "Taimal", 3),

    (4, "Josue", "Colimba", 4),

    (5, "Edison", "Perez", 5);

#Datos tabla telefono Enfermero

INSERT INTO hospital.telefonoenfermero(idtelefonoEnfermero, telefono)

VALUES(1, "112345"),

    (2, "122345"),

    (3, "132345"),

    (4, "142345"),

    (5, "152345");

## Sentencias para consultas solicitadas

#Consultas

# medicamentos a tomado cada paciente y la dosis suministrada.

SELECT p.nombre, m.nombreMedicamento, m.dosis

FROM paciente p

INNER JOIN pacienteMedicamento pm ON p.id = pm.idPaciente

INNER JOIN medicamento m ON m.id = pm.idMedicamento;

Table

Description automatically generated

Figura : paciente y dosis

#Enfermeros estuvieron en los procedimientos de los pacientes.

SELECT pr.tipoProcedimiento, e.nombreEnfermero, p.nombre as nombre\_paciente, p.apellido as apellido\_paciente

FROM procedimiento pr

INNER JOIN paciente p ON p.idProcedimiento = pr.id

INNER JOIN procedimientoMedico pm ON pm.idProcedimiento = pr.id

INNER JOIN enfermero e ON e.idMedico = pm.idMedico;

Table

Description automatically generated

Figura : procedimiento, enfermero y paciente

## Vista necesarias en hospital

1. Es importante conocer el nombre del medico y enfermero que atendió al paciente ya que para alguna solicitud o aclaración en cuanto al procedimiento medico por parte del paciente, además es necesario esta consulta para registrar en la historia clínica del paciente.

# Medicos  y enfermeros que asistieron a pacientes

CREATE VIEW enfermero\_medico\_paciente AS

SELECT e.nombreEnfermero, e.apellidoEnfermero, p.nombre as paciente, pm.idProcedimiento, m.nombreMedico, m.apellidoMedico

FROM enfermero e

JOIN procedimientoMedico pm ON e.idMedico = pm.idMedico

JOIN paciente p ON p.idProcedimiento = pm.idProcedimiento

JOIN medico m ON m.id = pm.idMedico;

SELECT\*FROM enfermero\_medico\_paciente;

Graphical user interface, application

Description automatically generated

Figura : vista enfermero, médico y paciente

1. Vista teléfono medico es importante porque es necesario conocer el teléfono del médico ya que un paciente o el jefe de los médicos los puede requerir información de disponibilidad del médico o en cuanto al procedimiento realizado.

# vista telefono medico

CREATE VIEW telefono\_medico AS

SELECT m.nombreMedico, tm.telefono

FROM medico m

JOIN telefonoMedico tm ON m.id = tm.idtelefonoMedico;

Table

Description automatically generated

Figura : vista médico teléfono

1. Vista teléfono enfermero, es importante porque el médico o jefe del hospital puede requerir una información o atención inmediata por parte de un enfermero en especifico y al contar con esta tabla de contacto de enfermeros facilitaría el llamado a cualquier enfermero que sea solicitado.

#Vista telefono enfermero

CREATE VIEW vista\_telefono\_enfermero AS

SELECT e.nombreEnfermero, e.apellidoEnfermero, t.telefono

FROM enfermero e

JOIN telefonoEnfermero t ON e.idenfermero = t.idtelefonoEnfermero;



Figura : vista enfermero teléfono

## Respuesta a pregunta

¿Qué le agregaría al modelo para dar más información y esa información cual sería? En la tabla paciente se puede agregar fecha de nacimiento y sexo (masculino, femenino) de esta manera determinar la edad del paciente, en consecuencia, ayudaría al hospital a tener un registro estimado del tipo de pacientes a los que atiende, por ejemplo, si la mayoría son niños, jóvenes o adultos y de esta manera abrir espacios de atención que estén acordes a los pacientes.

# Taller 6

## Primera actividad

Procedimientos almacenados para agregar, actualizar, consultar y borrar en la tabla clientes.

### Procedimiento para agregar clientes

DELIMITER //

CREATE PROCEDURE sp\_agregar\_cliente (

    IN cedula\_param VARCHAR(10),

    IN nombre\_param VARCHAR(45)

)

BEGIN

    INSERT INTO cliente (cedula, nombre) VALUES (cedula\_param, nombre\_param);

END//

DELIMITER ;

CALL sp\_agregar\_cliente("9898011", "210021");

Graphical user interface, text, application, table, Excel

Description automatically generated

Figura : sp agregar cliente

### Procedimiento para actualizar clientes

#sp para actualizar clientes

DELIMITER //

CREATE PROCEDURE sp\_actualizar\_cliente (

    IN cedula\_param VARCHAR(10),

    IN nombre\_param VARCHAR(45)

)

BEGIN

    UPDATE cliente SET nombre = nombre\_param WHERE cedula = cedula\_param;

END//

DELIMITER ;

CALL sp\_actualizar\_cliente('9898011', 'AlejandraGuadir');

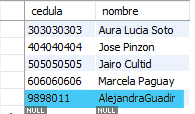


Figura : sp actualizar cliente

### Procedimiento para consultar clientes

#sp para consultar clientes

DELIMITER //

CREATE PROCEDURE sp\_consultar\_cliente (

    IN cedula\_param VARCHAR(10)

)

BEGIN

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

END//

DELIMITER ;

CALL sp\_consultar\_cliente('9898011');



Figura : sp consultar cliente

### Procedimiento para eliminar clientes por el parámetro cedula.

#sp para eliminar cliente

DELIMITER //

CREATE PROCEDURE sp\_borrar\_cliente (

    IN cedula\_param VARCHAR(10)

)

BEGIN

    DELETE FROM cliente WHERE cedula = cedula\_param;

END//

DELIMITER ;

CALL sp\_borrar\_cliente('9898011');

### Sentencias sql para crear table control\_de\_cambios\_librería

# Crear la tabla control\_de\_cambios\_librería

CREATE TABLE IF NOT EXISTS control\_de\_cambios\_librería (

  usuario VARCHAR(45) NOT NULL,

  accion VARCHAR(10) NOT NULL,

  fecha DATETIME NOT NULL,

  PRIMARY KEY (usuario, accion, fecha));

### Trigger para registrar el usuario que registro los datos en la tabla clientes.

DELIMITER //

CREATE TRIGGER tr\_insertar\_cliente

AFTER INSERT ON cliente

FOR EACH ROW

BEGIN

  INSERT INTO control\_de\_cambios\_librería (usuario, accion, fecha)

  VALUES (USER(), "insert", NOW());

END//

DELIMITER ;

### Trigger para registrar el usuario que registro los datos en la tabla clientes.

DELIMITER //

CREATE TRIGGER tr\_eliminar\_cliente

AFTER DELETE ON cliente

FOR EACH ROW

BEGIN

  INSERT INTO control\_de\_cambios\_librería (usuario, accion, fecha)

  VALUES (USER(), "delete", NOW());

END//

DELIMITER ;

Graphical user interface, application, table

Description automatically generated

Figura : control de cambios en la tabla clientes

## Segunda actividad

Procedimientos almacenados para agregar, actualizar, consultar y borrar en la tabla clientes.

### Procedimiento para agregar procedimiento medico

#Sp para agregar procedimiento

DELIMITER //

CREATE PROCEDURE sp\_agregar\_procedimiento (

    IN id\_param INT,

    IN tipoProcedimiento\_param VARCHAR(45)

)

BEGIN

    INSERT INTO procedimiento (id, tipoProcedimiento) VALUES (id\_param, tipoProcedimiento\_param);

END//

DELIMITER ;

CALL sp\_agregar\_procedimiento("11", "Urologia");

SELECT\*FROM procedimiento;

### Procedimiento para actualizar procedimiento medico

#Sp para actualizar procedimiento

DELIMITER //

CREATE PROCEDURE sp\_actualizar\_procedimiento (

    IN id\_param INT,

    IN tipoProcedimiento\_param VARCHAR(45)

)

BEGIN

    UPDATE procedimiento SET id = id\_param WHERE tipoProcedimiento = tipoProcedimiento\_param;

END//

DELIMITER ;

CALL sp\_actualizar\_procedimiento('11', 'Cirugia general');

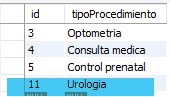


Figura : sp para actualizar procedimiento

### Procedimiento para consultar procedimiento medico

#Sp para consultar procedimiento

DELIMITER //

CREATE PROCEDURE sp\_consultar\_procedimiento (

    IN id\_param  INT

)

BEGIN

    SELECT \* FROM procedimiento WHERE id = id\_param ;

END//

DELIMITER ;

CALL sp\_consultar\_procedimiento('11');



Figura : sp para consultar procedimiento

### Procedimiento para eliminar procedimiento medico

#sp para eliminar procedimiento

DELIMITER //

CREATE PROCEDURE sp\_borrar\_procedimiento (

    IN id\_param   INT

)

BEGIN

    DELETE FROM procedimiento WHERE id = id\_param  ;

END//

DELIMITER ;

CALL sp\_borrar\_procedimiento('11');

### Sentencias sql para crear table control\_de\_cambios\_de procedimiento en hospital

# Crear la tabla control\_de\_cambios\_en procedimientos de hospital

CREATE TABLE IF NOT EXISTS control\_de\_cambios\_procedimientosMedicos (

  usuario VARCHAR(45) NOT NULL,

  accion VARCHAR(10) NOT NULL,

  fecha DATETIME NOT NULL,

  PRIMARY KEY (usuario, accion, fecha)

);

### Trigger para control de cambios al insertar procedimientos medicos

#Trigger para registrar el usuario que registro los datos en la tabla clientes.

DELIMITER //

CREATE TRIGGER tr\_insertar\_procedimientoMedico

AFTER INSERT ON procedimiento

FOR EACH ROW

BEGIN

  INSERT INTO tr\_insertar\_procedimientoMedico (usuario, accion, fecha)

  VALUES (USER(), "insert", NOW());

END//

DELIMITER ;

### Trigger para control de cambios al eliminar registros de procedimientos medicos

#Trigger para registrar el usuario que registro los datos en la tabla clientes.

DELIMITER //

CREATE TRIGGER tr\_eliminar\_procedimientoMedico

AFTER DELETE ON procedimiento

FOR EACH ROW

BEGIN

  INSERT INTO tr\_insertar\_procedimientoMedico (usuario, accion, fecha)

  VALUES (USER(), "delete", NOW());

END//

DELIMITER ;