Documentación Taller C1-2023-QA-BD-05

Primera actividad

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

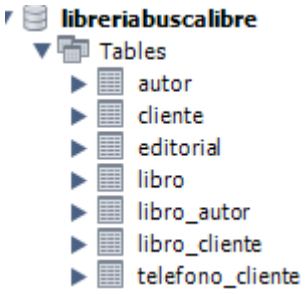
Complete la información para las tablas autor, libro, cliente, editorial, libro_cliente, libro_autor y teléfono_cliente con al menos (5,20,7,4,10,10, 12) registros respectivamente usando unicamente comandos SQL creados por usted.

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.

Solución primera actividad

El archivo sql contiene el código para crear una base de datos llamada *LibreriaBuscaLibre*. En primer lugar, procedo a crear la base datos. La base de datos trae la palabra reservada USE para seleccionar la base de datos que vamos a utilizar. Al final, workbench nos muestra un esquema de la base de datos.



Como se puede ver, trae las siguientes tablas:

- 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;
```

La tabla autor contiene un id como llave primaria, una fecha de nacimiento, una nacionalidad y un nombre. Todos son de tipo VARCHAR.

- 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;
```

La tabla libro contiene un ISBN como llave primaria, un nombre_editorial como llave foránea que se relaciona con la tabla editorial, un titulo y un numero_paginas. También se observa una restricción que indica que el valor nombre_editorial debe tener un nombre existente en la tabla editorial.

- cliente

```
CREATE TABLE IF NOT EXISTS `LibreriaBuscaLibre`.`cliente` (
  `cedula` VARCHAR(10) NOT NULL,
  `nombre` VARCHAR(45) NULL,
  PRIMARY KEY (`cedula`))
ENGINE = InnoDB;
```

La tabla cliente presenta un campo cedula como llave primaria y un nombre. Ambos de tipo varchar.

- 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;
```

La tabla editoral contiene un nombre como llave primaria, una ciudad, un complemento y un teléfono. Todos de tipo VARCHAR.

- 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;
```

La tabla libro_cliente contiene un ISBN_libro_cliente y un id_cliente. La clave primaria de la tabla es una combinación de los dos campos anteriores. ISBN_libro_cliente se refiere a la tabla libro y su campo ISBN.
id_cliente se refiere a la tabla cliente y su campo cedula.

- libro_autor

```sql
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;
```

En la tabla libro_autor contiene un ISBN_libro y un id_autor. Ambos representan la clave primaria de la tabla. Existe dos restricciones indica que los campos anteriores deben de existir en las tablas autor y libro.

- teléfono_cliente

```sql
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;
```

La tabla teléfono_cliente presenta una cedula_cliente y un numero. Ambas son llaves primarias. También hay una restricción que indica que el valor cedula_cliente debe existir en la tabla cliente con el nombre del campo cedula.

Registros

Cada tabla puede ser rellenada por un sinnúmero de registros, no obstante, se estimó una cantidad para cada tabla. En este caso para hacer los registros se usa el comando INSERT INTO el cual se encuentra en el subconjunto DML de SQL. A continuación, se muestran los registros realizados:

- Autor (5 registros)

```
INSERT INTO LibreriaBuscaLibre.autor (id, fecha_de_nacimiento, nacionalidad, nombre)
VALUES
("12", "1990-01-12", "Colombiano", "Camilo Navas"),
("13", "2000-02-22", "Colombiano", "Javier Salas"),
("14", "1990-08-31", "Colombiano", "Sonia Parada"),
("15", "1976-05-25", "Colombiano", "Pedro Pascal"),
("16", "2001-07-26", "Colombiano", "Lina Huerta");
```

- Cliente (7 registros)

```
INSERT INTO libreriabuscalibre.cliente (cedula, nombre)
VALUES
("1140", "Fernando Ballestas"),
("1141", "Jaime Torres"),
("1142", "Tomas Cabal"),
("1143", "Yenni Candela"),
("1144", "Lola Esposito"),
("1145", "Tim Bolaños"),
("1146", "Ruth Chará");
```

- Editorial (4 registros)

```
INSERT INTO libreriabuscalibre.editorial (nombre, ciudad, complemento, Telefono)
VALUES
("Editorial Norma", "Barranquilla", "Calle esquina", "312454"),
("Editorial Nacional", "Bogotá", "Segundo piso", "312321"),
("Editorial Sofka", "Medellin", "Al lado del cai municipal", "311256"),
("Editorial Colombia", "Bogota", "Calle esquina", "301456");
```

- Libro (20 registros)

```sql
INSERT INTO libreriabuscalibre.libro (ISBN, titulo, numero_paginas, nombre_editorial)
VALUES
("I00020", "En la cumbre", "20", "Editorial Norma"),
("I00021", "La cupula", "200", "Editorial Nacional"),
("I00022", "Cumbres borrascosas", "231", "Editorial Sofka"),
("I00023", "Harry Potter 1", "89", "Editorial Nacional"),
("I00024", "Harry Potter 2", "300", "Editorial Nacional"),
("I00025", "Harry Potter 3", "67", "Editorial Norma"),
("I00026", "Harry Potter 4", "89", "EEditorial Sofka"),
("I00027", "Harry Potter 5", "800", "Editorial Nacional"),
("I00028", "Harry Potter 6", "1340", "Editorial Sofka"),
("I00029", "Harry Potter 7", "480", "Editorial Colombia"),
("I00030", "Rojo", "450", "Editorial Nacional"),
("I00031", "Negro", "200", "Editorial Norma"),
("I00032", "Verde", "450", "Editorial Nacional"),
("I00033", "Blanco", "670", "Editorial Norma"),
("I00034", "Resplandor", "210", "Editorial Nacional"),
("I00035", "Claustrofobia", "120", "Editorial Sofka"),
("I00036", "3", "520", "Editorial Nacional"),
("I00037", "Amor y odio", "820", "Editorial Colombia"),
("I00038", "Java para tontos", "20", "Editorial Colombia"),
("I00039", "Sql para tontos", "20", "Editorial Norma"),
("I00040", "Mantenga su casa hermosa", "20", "Editorial Sofka");
```

- Libro autor (10 registros)

```sql
INSERT INTO libreriabuscalibre.libro_autor (ISBN_libro, id_autor)
VALUES
("I00020", "12"),
("I00021", "13"),
("I00022", "14"),
("I00023", "15"),
("I00024", "16"),
("I00025", "12"),
("I00026", "16"),
("I00027", "13"),
("I00028", "12"),
("I00032", "16");
```

- Libro cliente (10 registros)

```
INSERT INTO libreriabuscalibre.libro_cliente (ISBN_libro_cliente, id_cliente)
VALUES
("I00035", "1140"),
("I00036", "1141"),
("I00037", "1142"),
("I00038", "1143"),
("I00031", "1144"),
("I00021", "1145"),
("I00021", "1146"),
("I00022", "1141"),
("I00025", "1140"),
("I00034", "1142");
```

- Teléfono cliente (12 registros)

```
INSERT INTO libreriabuscalibre.telefono_cliente (cedula_cliente, numero)
VALUES
("1140", "313427"),
("1140", "012345"),
("1141", "315908"),
("1141", "002134"),
("1142", "876543"),
("1142", "376432"),
("1143", "102357"),
("1143", "178956"),
("1144", "12053"),
("1144", "31467"),
("1145", "3147927"),
("1145", "319607");
```

Consultas

Las consultas nos permiten traer datos guardados dentro de las tablas. Se usa generalmente el comando SELECT el cual se encuentra en el subconjunto DQL. Los registros solicitados son:

- Nombre y fecha de nacimiento del autor

```
1 •  SELECT nombre, fecha_de_nacimiento from libreriabuscalibre.autor;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

nombre	fecha_de_nacimiento
Camilo Navas	1990-01-12
Javier Salas	2000-02-22
Sonia Parada	1990-08-31
Pedro Pascal	1976-05-25
Lina Huerta	2001-07-26

- Cantidad libros diferente vendidos

```
1 •  SELECT COUNT(DISTINCT ISBN_libro_cliente)
2    FROM libro_cliente;
```

Result Grid | Filter Rows: | Export: | W

COUNT(DISTINCT ISBN_libro_cliente)
9

- Cliente con nombre y teléfono

```
SELECT libreriabuscalibre.cliente.nombre, libreriabuscalibre.telefono_cliente.numero
FROM libreriabuscalibre.cliente
JOIN libreriabuscalibre.telefono_cliente
ON libreriabuscalibre.cliente.cedula = libreriabuscalibre.telefono_cliente.cedula_cliente;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

nombre	numero
Fernando Ballestas	012345
Fernando Ballestas	313427
Jaime Torres	002134
Jaime Torres	315908
Tomas Cabal	376432
Tomas Cabal	876543
Yenni Candela	102357
Yenni Candela	178956
Lola Esposito	12053
Lola Esposito	31467
Tim Bolaños	3147927
Tim Bolaños	319607

- Libro con su autor

```
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;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

titulo	autores
Cumbres borrascosas	Sonia Parada
En la cumbre	Camilo Navas
Harry Potter 1	Pedro Pascal
Harry Potter 2	Lina Huerta
Harry Potter 3	Camilo Navas
Harry Potter 4	Lina Huerta
Harry Potter 5	Javier Salas
Harry Potter 6	Camilo Navas
La cupula	Javier Salas
Verde	Lina Huerta

- **Editoriales que vendieron libros**

```
SELECT DISTINCT nombre_editorial
FROM libro
JOIN libro_cliente
ON libro.ISBN = libro_cliente.ISBN_libro_cliente;
```

Result Grid | Filter Rows: | Export: | Wrap Cell C

nombre_editorial
Editorial Norma
Editorial Sofka
Editorial Nacional
Editorial Colombia

Vistas

- **Libros por editorial**

```
CREATE VIEW libros_por_editorial AS SELECT editorial.nombre AS nombre_editorial, COUNT(*) AS cantidad_libros
FROM libro
JOIN editorial ON libro.nombre_editorial = editorial.nombre
GROUP BY editorial.nombre;
```

Se crea esta vista puesto que parece útil saber cuantos libros están presentes en cada editorial. Para hacer la sentencia se utilizó la cláusula join para unir las tablas libro y editorial. Se usa el group by para agrupar los resultados por el nombre de la editorial. Se usa un count para calcular la cantidad de libros por editorial. El resultado es el siguiente:

```
1 •  select * from libros_por_editorial;
```

Result Grid | Filter Rows: | Export:

nombre_editorial	cantidad_libros
Editorial Colombia	3
Editorial Nacional	8
Editorial Norma	5
Editorial Sofka	4

Se puede observar que la suma de las cantidades da 20, el cual es el número de registros de la tabla libro.

- Detalles libros

```
CREATE VIEW detallesLibros AS SELECT libro.titulo, libro.numero_paginas, autor.nombre as Nombre_autor
FROM libro
INNER JOIN libro_autor ON libro.ISBN = libro_autor.ISBN_libro
INNER JOIN autor ON libro_autor.id_autor = autor.id;
```

En esta vista se puede traer varios detalles de los libros: su título, número de páginas y nombre del autor(es). Parece útil como referencia para futuros clientes o como método para clasificar el libro según su autor. Se usa la cláusula INNER JOIN para combinar las tablas libro y libro_autor por medio de la llave principal ISBN. El resultado es el siguiente:

```
1 •   SELECT * FROM detallesLibros;
```

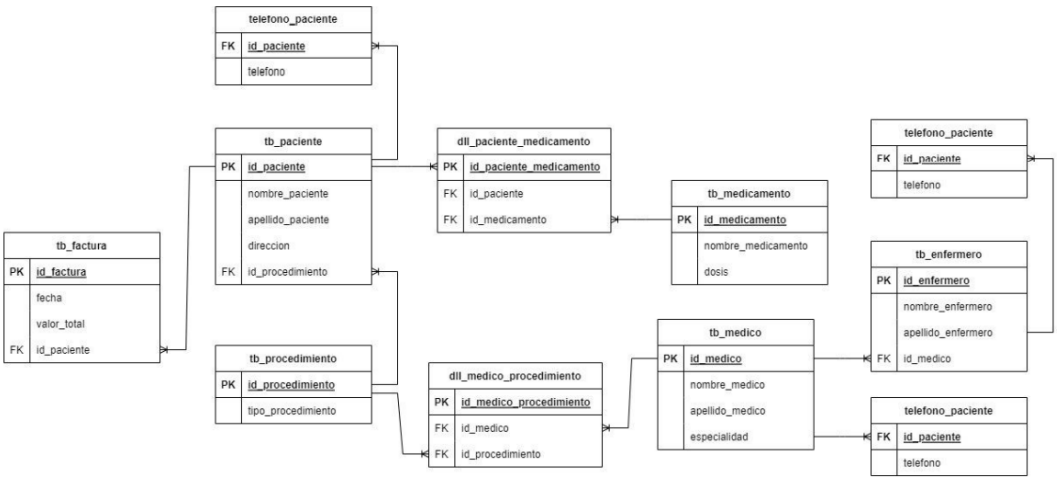
Result Grid | Filter Rows: | Export:

titulo	numero_paginas	Nombre_autor
En la cumbre	20	Camilo Navas
Harry Potter 3	67	Camilo Navas
Harry Potter 6	1340	Camilo Navas
La cupula	200	Javier Salas
Harry Potter 5	800	Javier Salas
Cumbres borrascosas	231	Sonia Parada
Harry Potter 1	89	Pedro Pascal
Harry Potter 2	300	Lina Huerta
Harry Potter 4	89	Lina Huerta
Verde	450	Lina Huerta

Como se puede ver en la imagen anterior, la sentencia permite traer el título de libro, su número de páginas y trae de la tabla libro_autor el nombre del autor. Nótese que varios autores tienen a su nombre -valga la redundancia- varios libros.

Actividad 2

En primer lugar, se descargó la imagen del modelo relacional del repositorio del compañero Jhonatan.



Se pueden evidenciar dos errores en cuanto a las tablas referentes al teléfono de los enfermeros y del doctor. Esos errores se arreglarán en la creación de la base de datos.

Conversión del MR a base de datos con sentencias SQL

Para comenzar a crear la base datos procedemos a usar los comandos CREATE DATABASE y USE para crear y seleccionar la base de datos llamada Hospital. Obsérvese la imagen.

```
CREATE DATABASE Hospital;
USE Hospital;
#Medico---------------------------
CREATE TABLE IF NOT EXISTS medico(
    id_medico VARCHAR(10) PRIMARY KEY,
    nombre_medico VARCHAR(45),
    apellido_medico VARCHAR(45),
    especialidad VARCHAR(45)
);
```

Además, se puede ver la creación de la tabla médico. Esta tabla tiene un id que funciona como llave primaria, un nombre, apellido y una especialidad. Todos de tipo VARCHAR.

De igual forma se creó la tabla teléfono medico

```
#Telefono_medico---------------------------
CREATE TABLE IF NOT EXISTS telefono_medico(
    medico_id VARCHAR(10),
    telefono_medico VARCHAR(15),
    PRIMARY KEY (medico_id,telefono_medico),
    FOREIGN KEY (medico_id) REFERENCES medico(id_medico)
);
```

Se puede observar que tiene un id y un teléfono_medico como llaves primarias las cuales representan la llave primaria de médico.

También se creó la tabla enfermero

```
#Enfermero---------------------------
CREATE TABLE IF NOT EXISTS enfermero(
    id_enfermero VARCHAR(10) PRIMARY KEY,
    nombre_enfermero VARCHAR(50),
    apellido_enfermero VARCHAR(50),
    medicoID VARCHAR(10),
    FOREIGN KEY (medicoID) REFERENCES medico(id_medico)
);
```

La tabla tiene un id, nombre, apellido y un medicoID que funciona como llave foránea la cual se referencia con la llave primaria de la tabla médico.

Tabla teléfono enfermero

```
#Telefono_enfermero---------------------------
CREATE TABLE IF NOT EXISTS telefono_enfermero(
    enfermero_id VARCHAR(10),
    telefono_enfermero VARCHAR(15),
    PRIMARY KEY(enfermero_id,telefono_enfermero),
    FOREIGN KEY(enfermero_id) REFERENCES enfermero(id_enfermero)
);
```

Se puede observar que tiene un id y un teléfono_enfermero como llaves primarias las cuales representan la llave primaria de enfemero y teléfono enfermero.

Tabla procedimiento

```
#Procedimiento---------------------------
CREATE TABLE IF NOT EXISTS procedimiento(
    id_procedimiento VARCHAR(10) PRIMARY KEY,
    tipo_procedimiento VARCHAR(20)
);
```

La tabla procedimiento contiene un id como llave primaria y un tipo de procedimiento. Ambas columnas son de tipo VARCHAR.

Tabla enfermero procedimiento

```
#Dll_enfermero_procedimiento---------------------------
CREATE TABLE IF NOT EXISTS dll_enfermero_procedimiento(
    id_enfermero_procedimiento VARCHAR(10),
    id_procedimiento_enfermero VARCHAR(10),
    PRIMARY KEY( id_enfermero_procedimiento,id_procedimiento_enfermero),
    FOREIGN KEY( id_enfermero_procedimiento)REFERENCES enfermero(id_enfermero),
    FOREIGN KEY(id_procedimiento_enfermero)REFERENCES procedimiento(id_procedimiento)
);
```

En la tabla enfermero procedimiento se aprecia un id_enfermero_procedimiento y un id_procedimiento_enfermero, ambas son llaves primarias y referenciar a las tablas de enfermero y procedimiento respectivamente. Cabe destacar que el compañero Jhonatan realizó en el modelo la relación con la tabla médico. Esto generaría un conflicto al momento de realizar la consulta de procedimiento realizado por enfermero a paciente la cual es una de las búsquedas solicitadas en el taller.

Tabla paciente

```
#Paciente---------------------------
CREATE TABLE IF NOT EXISTS paciente(
    id_paciente VARCHAR(10) PRIMARY KEY,
    nombre_paciente VARCHAR(50),
    apellido_paciente VARCHAR(50),
    direccion VARCHAR(50),
    id_procedimiento_paciente VARCHAR(20),
    FOREIGN KEY(id_procedimiento_paciente) REFERENCES procedimiento(id_procedimiento)
);
```

La tabla paciente contiene un id como llave primaria, un nombre, apellido, dirección y un id_procedimiento_paciente que funciona como llave foránea que se referencia con la tabla procedimientos.

Tabla teléfono paciente

```
#Telefono paciente--------------------------
CREATE TABLE IF NOT EXISTS telefono_paciente(
    id_paciente_telefono VARCHAR(10),
    telefono_paciente VARCHAR(15),
    PRIMARY KEY(id_paciente_telefono,telefono_paciente),
    FOREIGN KEY(id_paciente_telefono) REFERENCES paciente(id_paciente)
);
```

Se puede observar que tiene un id y un teléfono_paciente como llaves primarias. No obstante, el id funciona como llave foránea que se referencia con la tabla paciente.

Tabla factura

```
#Factura--------------------------
CREATE TABLE IF NOT EXISTS factura(
    id_factura VARCHAR(10) PRIMARY KEY,
    fecha VARCHAR(20),
    valor_total double,
    id_paciente_factura varchar(10),
    FOREIGN KEY(id_paciente_factura) REFERENCES paciente(id_paciente)
);
```

La tabla factura contiene un id como llave primaria, una fecha, un valor de tipo double y un id_paciente_factura que funciona como llave foránea que referencia a la tabla paciente.

Tabla medicamento

```
#Medicamento--------------------------
CREATE TABLE IF NOT EXISTS medicamento(
    id_medicamento VARCHAR(10) PRIMARY KEY,
    nombre_medicamento VARCHAR(50),
    dosis VARCHAR(50)
);
```

La tabla medicamento contiene un id como llave primaria, un nombre y una dosis. Todos de tipo VARCHAR.

Tabla paciente medicamento

```
#Dll_paciente_medicamento--------------------------
CREATE TABLE  IF NOT EXISTS dll_paciente_medicamento(
    id_paciente_medicamento VARCHAR(10),
    id_medicamento_paciente VARCHAR(10),
    PRIMARY KEY(id_paciente_medicamento,id_medicamento_paciente),
    FOREIGN KEY(id_paciente_medicamento) REFERENCES paciente(id_paciente),
    FOREIGN KEY(id_medicamento_paciente) REFERENCES medicamento(id_medicamento)
);
```

La tabla paciente medicamento contiene un id_paciente_medicamento, un id_medicamento_paciente que funcionan como llaves primarias. También se convierten en llaves foráneas para poder referenciarse con las tablas paciente y medicamento respectivamente.

Registros

Los registros para cada unas de las tablas son los siguientes:

Medico

```
INSERT INTO medico (id_medico,nombre_medico,apellido_medico,especialidad) VALUES
("1122","Grace","Anatomy","Oftamología"),
("1123","Shaun","Porter","Otorrinolaringología"),
("1124","Pepo","Perez","Dermatologo"),
("1125","Maria","Lin","General"),
("1126","Doctor","Sueño","Psiquiatra");
```

Teléfono medico

```
INSERT INTO telefono_medico(medico_id,telefono_medico) VALUES
("1122","312345"),
("1124","313254"),
("1123","314654"),
("1125","315789"),
("1126","316908");
```

Enfermero

```
INSERT INTO enfermero (id_enfermero,nombre_enfermero,apellido_enfermero,medicoID) VALUES
("3344", "Poncho", "Tevez", "1122"),
("3345", "Lolo", "Torres", "1123"),
("3346", "Juan", "Reyes", "1122"),
("3347", "Morty", "Smith", "1125"),
("3348", "Rick", "Sanchez", "1126");
```

Teléfono enfermero

```
INSERT INTO telefono_enfermero(enfermero_id,telefono_enfermero) VALUES
("3344","212345"),
("3346","123456"),
("3344","543212"),
("3345","612345"),
("3347","712324");
```

Procedimiento

```
INSERT INTO procedimiento(id_procedimiento,tipo_procedimiento) VALUES
("PRO1","Revision ojos"),
("PRO2","Revisión nariz"),
("PRO3","Biopsia"),
("PRO4","Revision"),
("PRO5","Lobotomía");
```

Medico procedimiento

```
INSERT INTO dll_medico_procedimiento(id_medico_procedimiento,id_procedimiento_medico) VALUES
("1122","PRO1"),
("1124","PRO2"),
("1123","PRO3"),
("1125","PRO4"),
("1126","PRO5");
```

Pacientes

```
INSERT INTO paciente(id_paciente,nombre_paciente,apellido_paciente,direccion, id_procedimiento_paciente) VALUES
("5566","Lionel","Messi","carrera 2","PRO1"),
("5567","Elio","Hoyos","calle 34","PRO2"),
("5568","Pepa","Pig","Calle Inglaterra","PRO3"),
("5569","Mario","Casas","calle madrid","PRO4"),
("5570","Tito","Martinez","carrera 34","PRO5");
```

Pacientes teléfono

```
INSERT INTO telefono_paciente(id_paciente_telefono,telefono_paciente) VALUES
("5566","312"),
("5566","313"),
("5568","314"),
("5570","315"),
("5568","316");
```

Factura

```
INSERT INTO factura(id_factura,fecha,valor_total, id_paciente_factura) VALUES
("FOR1","1963-22-11",120000,"5566"),
("FOR2","2021-12-10",360000,"5567"),
("FOR3","2021-11-11",252000,"5566"),
("FOR4","2024-14-11",140000,"5570"),
("FOR5","2023-02-11",520000,"5569");
```

Medicamento

```
INSERT INTO medicamento(id_medicamento,nombre_medicamento,dosis) VALUES
("ME1","Paracetamol","1 cada 8 horas"),
("ME2","Acetaminofen","1 cada 8 horas"),
("M3","Aspirina","1 cada 16 horas"),
("ME4","Chiquitolina","1 cada 8 horas"),
("ME5","Quetiapina","1 cada 16 horas");
```

Paciente medicamento

```
INSERT INTO dll_paciente_medicamento(id_paciente_medicamento,  id_medicamento_paciente) VALUES
("5567","ME1"),
("5566","ME4"),
("5568","ME1"),
("5570","ME2"),
("5569","ME1");
```

Consultas

Las consultas solicitadas son las siguientes:

- Medicamentos tomados pacientes

```
SELECT paciente.nombre_paciente, medicamento.nombre_medicamento, medicamento.dosis
FROM paciente
INNER JOIN dll_paciente_medicamento ON paciente.id_paciente = dll_paciente_medicamento.id_paciente_medicamento
INNER JOIN medicamento ON dll_paciente_medicamento.id_medicamento_paciente = medicamento.id_medicamento;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

nombre_paciente	nombre_medicamento	dosis
Lionel	Chiquitolina	1 cada 8 horas
Elio	Paracetamol	1 cada 8 horas
Pepa	Paracetamol	1 cada 8 horas
Mario	Paracetamol	1 cada 8 horas
Tito	Acetaminofen	1 cada 8 horas

Esta consulta utiliza las tablas paciente, dll_paciente_medicamento y medicamento. Se utiliza la cláusula INNER JOIN para unir las tablas a partir de sus claves primarias y foráneas correspondientes. Luego, se seleccionan los campos nombre_paciente, nombre_medicamento y dosis de las tablas unidas.

- Procedimientos realizados por los enfermeros a los pacientes

```
SELECT e.nombre_enfermero, p.tipo_procedimiento, pa.nombre_paciente
FROM enfermero e
JOIN dll_enfermero_procedimiento de ON e.id_enfermero = de.id_enfermero_procedimiento
JOIN procedimiento p ON de.id_procedimiento_enfermero = p.id_procedimiento
JOIN paciente pa ON pa.id_procedimiento_paciente = p.id_procedimiento;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

nombre_enfermero	tipo_procedimiento	nombre_paciente
Poncho	Revision ojos	Lionel
Poncho	Biopsia	Pepa
Lolo	Revisión nariz	Elio
Morty	Lobotomía	Tito
Rick	Revision	Mario

Esta consulta realiza una unión entre las tablas enfermero, dll_enfermero_procedimiento, procedimiento y paciente para obtener la información requerida. En la tabla inferior se muestra el nombre del enfermero, el nombre del procedimiento y el nombre del paciente.

Vistas

- **Numero procedimientos**

```
CREATE VIEW numero_procedimientos AS SELECT tipo_procedimiento, COUNT(id_procedimiento) AS cantidad
FROM paciente
JOIN procedimiento ON paciente.id_procedimiento_paciente = procedimiento.id_procedimiento
GROUP BY tipo_procedimiento
ORDER BY cantidad DESC
LIMIT 1;
```

Esta consulta puede ser útil para indicar cual es el procedimiento que más se realiza. Esto puede servir para crear un historial de procedimiento. El esquema queda así

```
1 •   SELECT * FROM numero_procedimientos;
```

esult Grid | Filter Rows: | Export:

tipo_procedimiento	cantidad
Revision ojos	1

- Total facturas

```
CREATE VIEW total_facturas AS SELECT COUNT(*) AS total_facturas FROM factura;
```

Saber el total de facturas existentes puede ser muy útil para la contabilidad. Como también proporciona un informe que puede estudiar el comportamiento de los pacientes que buscan los medicamentos y procedimientos que se ven facturados. La vista se muestra de la siguiente forma.

```
1   SELECT * FROM total_facturas;
```

Result Grid | Filter Rows:

total_facturas
5

- Pagaré

  Suministra un inventario de cuanto debe cada paciente, parece útil para poder mantener los pagos realizados y hacer un presupuesto de esto. La imagen anterior muestra la vista

```
CREATE VIEW pagaré AS SELECT factura.id_factura, factura.valor_total, paciente.nombre_paciente, paciente.apellido_paciente
FROM factura
JOIN paciente ON factura.id_paciente_factura = paciente.id_paciente;
```

Al final la vista se ve de esta forma:

```
1    SELECT * FROM pagaré;
```

Result Grid | Filter Rows: | Export:

id_factura	valor_total	nombre_paciente	apellido_paciente
FOR1	120000	Lionel	Messi
FOR2	360000	Elio	Hoyos
FOR3	252000	Lionel	Messi
FOR4	140000	Tito	Martinez
FOR5	520000	Mario	Casas

Como se ve, aparece el id de la factura, y cada uno de los pagos cada paciente, termina siendo muy útil tener el registro de pagos.

Taller 6

Procedimientos y trigger para la base de datos libreriabuscalibre

El taller 6 consiste en realizar unos procedimientos que realicen un CRUD dentro de una tabla de la base de datos libreriabuscalibre. En primer lugar abro Workbench y utilizo la palabra reservada USE para seleccionar la base de datos que voy a utilizar.

```
use libreriabuscalibre;
```

Luego procedemos a crear el primer procedimiento. Este va a buscar un libro por medio de su llave primaria ISBN. En este caso I00020.

```
DELIMITER //

CREATE PROCEDURE buscarLibro (in identificadorLibro VARCHAR (10))
BEGIN
SELECT * FROM libro WHERE ISBN = identificadorLibro;
END //

DELIMITER ;

CALL buscarLibro("I00020");
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ISBN	titulo	numero_paginas	nombre_editorial
I00020	En la cumbre	20	Editorial Norma

Como se puede ver en la imagen el procedimiento trae la el libro con el ISBN declarado. Luego procedemos a actualizar el mismo registro, usamos de nuevo su llave primaria para extraerlo y usar el comando UPDATE para cambiar el titulo del libro.

```
DELIMITER //

CREATE PROCEDURE actualizarLibro (in libroActualizar VARCHAR (10))
BEGIN
UPDATE libro SET titulo = "Libro actualizado" WHERE ISBN = libroActualizar;
END //

DELIMITER ;

CALL actualizarLibro("I00020");
```

tion Output

	Time	Action	Message
1	14:04:58	CALL buscarLibro("I00020")	1 row(s) returned

Al final el procedimiento cambiar el titulo del libro y gracias al procedimiento de actualizar.

ISBN	titulo	numero_paginas	nombre_editorial
I00020	Libro actualizado	20	Editorial Norma

Luego procedemos a realizar el siguiente procedimiento, en esta caso vamos a crear un nuevo registro usando la función agregarLibro .

```
DELIMITER //

CREATE PROCEDURE agregarLibro (in agregarLibro VARCHAR (10))
BEGIN
INSERT INTO libro VALUES ("I00041", "Libro para eliminar" , "300", "Editorial Sofka");
END //

DELIMITER ;

CALL agregarLibro("I00041");
```

Como puede verse no es necesario traer los atributos de la tabla para insertarlos, siempre y cuando completemos cada y uno de los campos de la tabla. Si bien se crea el libro que posteriormente vamos a eliminar.

```
1 •  SELECT * FROM libreriabuscalibre.libro;
```

esult Grid | Filter Rows: | Edit: | Export/Import:

ISBN	titulo	numero_paginas	nombre_editorial
I00039	Sql para tontos	20	Editorial Norma
I00040	Mantenga su casa hermosa	20	Editorial Sofka
I00041	Libro para eliminar	300	Editorial Sofka
NULL	NULL	NULL	NULL

Como se ve en la anterior imagen, el registro es un éxito y trae los campos. Observen el registro arriba del libro creado para verificar que posteriormente lo vamos a eliminar.

En la imagen de abajo creamos el procedimiento para eliminar el libro creado anteriormente. Usamos su llave primaria para poder seleccionarlo y posteriormente eliminarlo.

```
DELIMITER //

CREATE PROCEDURE eliminarLibro (in libroEliminado VARCHAR (10))
BEGIN
DELETE FROM libro WHERE ISBN = libroEliminado;
END //

DELIMITER ;

CALL eliminarLibro("I00041");
```

Como se ve en la imagen de abajo, ya no aparece el libro con ISBN I00041, por lo cual podemos observar que el procedimiento se ejecutó correctamente

ISBN	titulo	numero_paginas	nombre_editorial
I00038	Java para tontos	20	Editorial Colombia
I00039	Sql para tontos	20	Editorial Norma
I00040	Mantenga su casa hermosa	20	Editorial Sofka
NULL	NULL	NULL	NULL

libro 1

Vista después de eliminar el libro.

Triggers

En primer lugar, procedemos a crear la tabla control de cambios.

```
CREATE TABLE IF NOT EXISTS libreriaBuscaLibre.control_de_cambios_librería (
  usuario VARCHAR(50) NOT NULL,
  accion VARCHAR(30) NOT NULL,
  fecha DATETIME DEFAULT CURRENT_TIMESTAMP);
```

Creamos el primer trigger para añadir libros, después de insertar un nuevo libro se crea un registro en la tabla recién creada arrojando un user, un mensaje de acción y una fecha, marcando de forma predeterminada la fecha actual de la creación del registro.

```
DELIMITER //
CREATE TRIGGER añadirLibro AFTER INSERT ON libro
    FOR EACH ROW
    BEGIN
    INSERT INTO libreriabuscalibre.control_de_cambios_librería VALUES
    (user(),"Agrega un nuevo libro", now());
    END;
// DELIMITER ;
```

Hacemos la prueba con insertando un nuevo libro

```
INSERT INTO libro (ISBN, titulo, numero_paginas, nombre_editorial)
VALUES
("I00090", "Vivir al 100", "400", "Editorial Norma");
```

Después de crearlo aparece un registro de acción denotando que el trigger funcionó

```
1 •  SELECT * FROM libreriabuscalibre.control_de_cambios_librería;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

usuario	accion	fecha
root@localhost	Agrega un nuevo libro	2023-02-14 19:16:50

Se comprueba de que el libro en realidad haya sido agregado.

ISBN	titulo	numero_paginas	nombre_editorial
I00039	Sql para tontos	20	Editorial Norma
I00040	Mantenga su casa hermosa	20	Editorial Sofka
I00090	Vivir al 100	400	Editorial Norma
NULL	NULL	NULL	NULL

Procedemos a hacer un segundo trigger que arroje un registro después de eliminar un libro, en este caso se eliminará el libro recién creado.

```
DELIMITER //
CREATE TRIGGER borrarLibro AFTER DELETE ON libro
    FOR EACH ROW
    BEGIN
    INSERT INTO libreriabuscalibre.control_de_cambios_librería VALUES
    (user(),"Eliminó un  libro", now());
    END;
// DELIMITER ;
```

Se comprueba si al borrar un libro traiga un registro de la acción del trigger y resulta ser un éxito.

```
1 •  DELETE FROM libreriabuscalibre.libro WHERE ISBN = "I00090";
2
3 •  SELECT * FROM libreriabuscalibre.control_de_cambios_librería;
4
5
```

esult Grid | Filter Rows: | Export: | Wrap Cell Content:

usuario	accion	fecha
root@localhost	Agrega un nuevo libro	2023-02-14 19:16:50
root@localhost	Eliminó un libro	2023-02-14 19:29:24

Se comprueba si en realidad el libro fue eliminado. El comando se realizó con éxito.

I00038	Java para tontos	20	Editorial Colombia
I00039	Sql para tontos	20	Editorial Norma
I00040	Mantenga su casa hermosa	20	Editorial Sofka
NULL	NULL	NULL	NULL

Procedimientos y triggers en la base de datos Hospital

Se procede a seleccionar la base de datos hospital

```
use hospital;
```

Se procede a crear el procedimiento de crear procedimiento

```
DELIMITER //
CREATE PROCEDURE añadirProcedimiento (in agregarProcedimiento VARCHAR (10))
BEGIN
INSERT INTO hospital.procedimiento VALUES ("PRO6", "Sacar sangre");
END //
DELIMITER ;

CALL añadirProcedimiento("PRO6");
```

Se crean los parámetros y comando insert para añadir un procedimiento.

id_procedimiento	tipo_procedimiento
PRO4	Revision
PRO5	Lobotomía
PRO6	Sacar sangre
NULL	NULL

Se comprueba que todo funcionó. Ahora pasamos al procedimiento de actualizar

```
DELIMITER //
CREATE PROCEDURE actualizarProcedimiento (in actualizarProcedimiento VARCHAR (10))
BEGIN
UPDATE hospital.procedimiento SET tipo_procedimiento = "Sacar plaquetas" WHERE id_procedimiento = actualizarProcedimiento;
END //
DELIMITER ;

CALL actualizarProcedimiento("PRO6");
```

Se comprueba que el procedimiento fue efectivamente actualizado

PRO4	Revision
PRO5	Lobotomía
PRO6	Sacar plaquetas
NULL	NULL

Efectivamente fue un éxito. Procedemos a hacer el procedimiento para ver procedimientos

```
DELIMITER //
CREATE PROCEDURE señalarProcedimiento (in señalarProcedimiento VARCHAR (10))
BEGIN
SELECT * FROM hospital.procedimiento WHERE id_procedimiento = señalarProcedimiento;
END //
DELIMITER ;

CALL señalarProcedimiento("PRO6");
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

id_procedimiento	tipo_procedimiento
PRO6	Sacar plaquetas

Se comprobó que fue un éxito. Ahora pasamos a el procedimiento eliminar. Eliminaremos el procedimiento protagonista de estas pruebas.

```
DELIMITER //
CREATE PROCEDURE eliminarProcedimiento (in eliminarProcedimiento VARCHAR (10))
BEGIN
DELETE FROM hospital.procedimiento WHERE id_procedimiento = eliminarProcedimiento;
END //
DELIMITER ;

CALL eliminarProcedimiento("PRO6");
```

Procedemos a revisar si funcionó

id_procedimiento	tipo_procedimiento
PRO3	Biopsia
PRO4	Revision
PRO5	Lobotomía
NULL	NULL

En efecto, funcionó.

Triggers

Se procede a crear la tabla solicitada

```
CREATE TABLE IF NOT EXISTS hospital.control_de_cambios_hospital (
  usuario VARCHAR(50) NOT NULL,
  accion VARCHAR(30) NOT NULL,
  fecha DATETIME DEFAULT CURRENT_TIMESTAMP);
```

Maneja un usuario, una acción y una fecha. Se crea el primer trigger de agregar

```
DELIMITER //
CREATE TRIGGER añadirProcedimiento AFTER INSERT ON hospital.procedimiento
    FOR EACH ROW
    BEGIN
    INSERT INTO hospital.control_de_cambios_hospital VALUES
    (user(),"Agregó un  procedimiento", now());
    END;
// DELIMITER ;
```

Se comprueba que todo se cumpla.

```
INSERT INTO hospital.procedimiento VALUES
("PRO7", "Sacar una muela");
SELECT * FROM hospital.control_de_cambios_hospital;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

usuario	accion	fecha
root@localhost	Agregó un procedimiento	2023-02-14 20:27:37

Se revisa si se creó

Result Grid | Filter Rows:

id_procedimiento	tipo_procedimiento
PRO4	Revision
PRO5	Lobotomía
PRO7	Sacar una muela
NULL	NULL

Fue un éxito.

Procedemos a hacer el trigger de eliminar. Crea un registro que se realiza después de eliminar un procedimiento

```
DELIMITER //
CREATE TRIGGER borrarProcedimiento AFTER DELETE ON hospital.procedimiento
    FOR EACH ROW
    BEGIN
    INSERT INTO hospital.control_de_cambios_hospital VALUES
    (user(),"Eliminó un  procedimiento", now());
    END;
// DELIMITER ;
```

Se observan los campos de usuario, acción y fecha. Procedemos a verificar.

```
1  DELETE FROM hospital.procedimiento WHERE id_procedimiento = "PRO7";
2  SELECT * FROM hospital.control_de_cambios_hospital;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

usuario	accion	fecha
root@localhost	Agregó un procedimiento	2023-02-14 20:27:37
root@localhost	Eliminó un procedimiento	2023-02-14 20:34:02

Se pueden ver ambos registros tanto de crear y eliminar. Lo cual nos permite afirmar que el trigger funciona con éxito.