Solución Taller 5 y 6 BD

Primera Actividad

Para el llenado de las tablas dentro de la BD de la librería Busca-Libre, se realizan las consultas SQL en un script aparte, para tener la cantidad de registros solicitados en cada una. Es necesario correr primero el script de la BD suministrado y luego el script del llenado de registros *FillingTables.sql* para una correcta ejecución. Para no saturar de información repetida, se explicará una consulta por tabla, así.

En la tabla autor, se inserta el autor Juan de Colombia, nacido el 1/1/1980 con identificador único 1.

```
-- ------------------------------------------------------
-- Tabla Autor
-- ------------------------------------------------------
INSERT INTO autor
VALUES
    ('1', '1/1/1980', 'COL', 'Juan'),
```

En la tabla editorial, se inserta la editorial única Planeta de Barcelona, ubicada en la calle 1 # 1 – 1, con teléfono 1111111111.

```
-- ------------------------------------------------------
-- Tabla Editorial
-- ------------------------------------------------------
INSERT INTO editorial
VALUES
    ('Planeta', 'Barcelona', 'Calle 1 # 1 - 1', '1111111111'),
```

En la tabla libro, se inserta el libro de ISBN único 10, titulado Luz, de 40 páginas y publicado por la editorial Planeta.

```
-- ------------------------------------------------------
-- Tabla Libro
-- ------------------------------------------------------
INSERT INTO libro
VALUES
    ('10', 'Luz', '40', 'Planeta'),
```

En la tabla cliente, se inserta un nuevo cliente llamado Pedro, identificado de manera única con su cédula 1111111119.

```
-- ---------------------------------------------------
-- Tabla Cliente
-- ---------------------------------------------------
INSERT INTO cliente
VALUES
    ('1111111119', 'Pedro'),
```

En la tabla de teléfonos de cliente, se inserta el teléfono 5555555555 del cliente identificado con la cédula 1111111113.

```
-- ---------------------------------------------------
-- Tabla Teléfono_Cliente
-- ---------------------------------------------------
INSERT INTO telefono_cliente
VALUES
    ('1111111113', '5555555555'),
```

En la tabla de los autores de libros, se inserta el ISBN 10 de un libro y se asocia con su escritor, es decir, el autor con identificador 1.

```
-- ---------------------------------------------------
-- Tabla Libro_Autor
-- ---------------------------------------------------
INSERT INTO libro_autor
VALUES
    ('10', '1'),
```

Finalmente, en la tabla de los libros comprados por clientes, se inserta el ISBN 11 de un libro que fue comprado por el cliente con cédula 1111111113.

```
-- ---------------------------------------------------
-- Tabla Libro_Cliente
-- ---------------------------------------------------
INSERT INTO libro_cliente
VALUES
    ('11', '1111111113'),
```

Dentro de estas consultas se insertan los datos diferentes para lograr poblar la BD con la cantidad de registros solicitados en la actividad.

Luego, se realizan 5 consultas con el objetivo de visualizar algunos datos en las tablas de la BD. La primera consulta solicitada busca ver el nombre y la fecha de nacimiento de cada

autor. Así, de la tabla de autor, se trae la columna nombre con su alias, y la columna fecha de nacimiento con su alias.

```
-- ------------------------------------------------------
-- Consulta Nombre y Fecha de Nacimiento de Autor
-- ------------------------------------------------------
SELECT nombre AS 'Nombre Autor', `fecha de nacimiento` AS 'Fecha de Nacimiento'
FROM autor;
```

Resultando en:

Nombre Autor	Fecha de Nacimiento
Juan	1/1/1980
Luis	1/2/1980
José	1/3/1980
Mark	1/4/1980
Alek	1/5/1980

La siguiente consulta pide visualizar la cantidad de libros diferentes vendidos. Así, de la tabla libro_cliente se trae cada libro distinto asociado a un cliente, y se cuenta está cantidad, además de asociarle un alias.

```
-- ------------------------------------------------------
-- Consulta cantidad de Libros diferentes vendidos
-- ------------------------------------------------------
SELECT COUNT(DISTINCT ISBN_libro_cliente)
AS 'Cantidad de Libros diferentes Vendidos'
FROM libro_cliente;
```

Cantidad de Libros diferentes Vendidos
8

La tercera consulta solicita ver cada libro con el cliente que lo compró y sus diferentes números telefónicos. Así, se realiza un cruce entre diferentes tablas, comparando las columnas necesarias para alcanzar este objetivo, para luego poder ver cada libro con su comprador, adicionalmente de cada número de teléfono que este tuviera.

```
-- ------------------------------------------------------
-- Consulta Nombre del cliente que compró Libro con su teléfono
-- ------------------------------------------------------
SELECT L.titulo AS 'Título del Libro', C.nombre AS 'Nombre Cliente', T.numero AS 'Teléfono Cliente'
FROM libro AS L
INNER JOIN libro_cliente AS LB
ON L.ISBN = LB.ISBN_libro_cliente
INNER JOIN cliente AS C
ON LB.id_cliente = C.cedula
INNER JOIN telefono_cliente AS T
ON C.cedula = T.cedula_cliente;
```

De la siguiente manera.

Título del Libro	Nombre Cliente	Teléfono Cliente
Viajando al mar	Sara	3033333333
Viajando al mar	Sara	5555555555
Rodolfo, el tripulante	Sara	3033333333
Rodolfo, el tripulante	Sara	5555555555
Viajando al mar	Mateo	4044444444
Viajando al mar	Mateo	6666666666
Cuarenta y tres horas	Mateo	4044444444
Cuarenta y tres horas	Mateo	6666666666
Viajando al mar	Felipe	5055555555
Viajando al mar	Felipe	7777777777
Escribir, arte y pecado	Felipe	5055555555
Escribir, arte y pecado	Felipe	7777777777
El nombre	Andrés	6066666666
El nombre	Andrés	8888888888
Felicidad	Carmen	7077777777
Felicidad	Carmen	9999999999
Despierta YA!	Manuela	1011111111
Español para extranjeros	Pedro	2022222222

Para la cuarta consulta pedida, como solo registraron 10 relaciones en la tabla libro_autor, el resultado contiene menos títulos de los almacenados originalmente, ya que estos aún no cuentan con un autor o autores registrados que los hayan escrito. La petición busca mostrar el nombre del libro, y quién(es) lo escribió. Así, a través de la sentencia INNER JOIN, se juntan 3 tablas, que me permiten acceder a está información y mostrarla de manera ordenada.

```
-- ---------------------------------------------------
-- Consulta Nombre de Libro con su(s) respectivo(s) Autor(es)
-- ---------------------------------------------------
SELECT L.titulo AS 'Título del Libro', A.nombre AS 'Autor'
FROM libro AS L
INNER JOIN libro_autor AS LA
ON L.ISBN = LA.ISBN_libro
INNER JOIN autor AS A
ON LA.id_autor = A.id
ORDER BY L.titulo;
```

Título del Libro	Autor
Cómo ser mejor en...	José
De visita al interior	Juan
Después de la muerte	Alek
El nombre del viento	Juan
La casa de al lado	Alek
Los camellos del sur	José
Luz	Juan
Luz	José
Recuerdos en verso	Alek
Sonidos lejanos	Alek

Finalmente, la quinta consulta, sencilla, pide ver qué editoriales han podido vender al menos un libro. Así, desde la tabla de libros y la relación entre estos y los clientes, se escogen las editoriales de manera distinta que están en la lista de libro_cliente.

```sql
-- ------------------------------------------------------
-- Consulta Editoriales que han vendido algún Libro
-- ------------------------------------------------------
SELECT DISTINCT nombre_editorial AS 'Editorial que ha vendido'
FROM libro AS L, libro_cliente AS LC
WHERE L.ISBN = LC.ISBN_libro_cliente;
```

Resultando en:

Editorial que ha vendido
RM
Planeta
Lumen

Finalmente, se crean dos vistas que se consideran de suma importancia para la librería, ya que con estas tendrá un acceso más limpio y rápido a la información más relevante. La primera vista propuesta es la vista del Stock que tiene la librería, es decir, listar todos los libros en existencia con toda su información, para saber qué productos tienen para ofrecer y resolver todas las dudas del cliente. Así, se utilizan los alias para nombrar las columnas de una mejor manera y se llaman todas desde la tabla libros.

```sql
-- ------------------------------------------------------
-- Stock de libros en la librería con toda su información
-- ------------------------------------------------------
CREATE VIEW stock AS
SELECT ISBN,
       titulo AS 'Título del libro',
       numero_paginas AS 'Número de páginas',
       nombre_editorial AS 'Editorial'
FROM libro;
```

Dejando una vista como la siguiente.

ISBN	Título del libro	Número de páginas	Editorial
10	Luz	40	Planeta
11	Viajando al mar	168	RM
12	Despegando	500	Planeta
13	El nombre	620	Lumen
14	La casa de al lado	461	Debolsillo
15	Felicidad	86	RM
16	Recuerdos en verso	614	Planeta
17	Despierta YA!	163	Planeta
18	Cómo ser mejor en...	613	RM
19	Español para extranjeros	795	Lumen
20	De visita al interior	985	RM
21	Rodolfo, el tripulante	32	Planeta
22	El nombre del viento	963	Debolsillo
23	Cuarenta y tres horas	433	Planeta
24	Los camellos del sur	94	Debolsillo
25	Escribir, arte y pecado	77	Lumen
26	Sonidos lejanos	623	RM
27	Amarte	81	Planeta
28	Después de la muerte	630	Lumen
29	Sentimientos más allá	106	Debolsillo

Luego, la segunda vista se trata de los datos de contacto de las editoriales, es decir, el nombre de la editorial junto con su número telefónico. Esto con el objetivo de que la librería pueda llamar rápidamente a sus proveedores para que puedan renovar el stock en caso de que se agote. Así, se traen las columnas de nombre y teléfono de la tabla editorial con sus correspondientes alias.

```
-- ------------------------------------------------------
-- Teléfonos de las editoriales para tenerlos disponibles rápidamente
-- ------------------------------------------------------
CREATE VIEW contactoEditorial AS
SELECT nombre AS 'Editorial',
       Telefono AS 'Teléfono de la editorial'
FROM editorial;
```

Resultando en una vista como la siguiente.

Editorial	Teléfono de la editorial
Debolsillo	4444444444
Lumen	3333333333
Planeta	1111111111
RM	2222222222

Segunda Actividad

Una vez analizado el MER del Hospital GNECJ, se construyó una BD por dos vías, un script SQL y un diagrama MR en Workbench. Para no extender la documentación, se sugiere dirigirse al archivo SQL *Hospital_GNECJ_BD* para ver en profundidad las consultas realizadas

para crear la BD por este método. En la Figura 1 se presenta el MR creado en Workbench y del que se obtuvo de igual forma la BD.

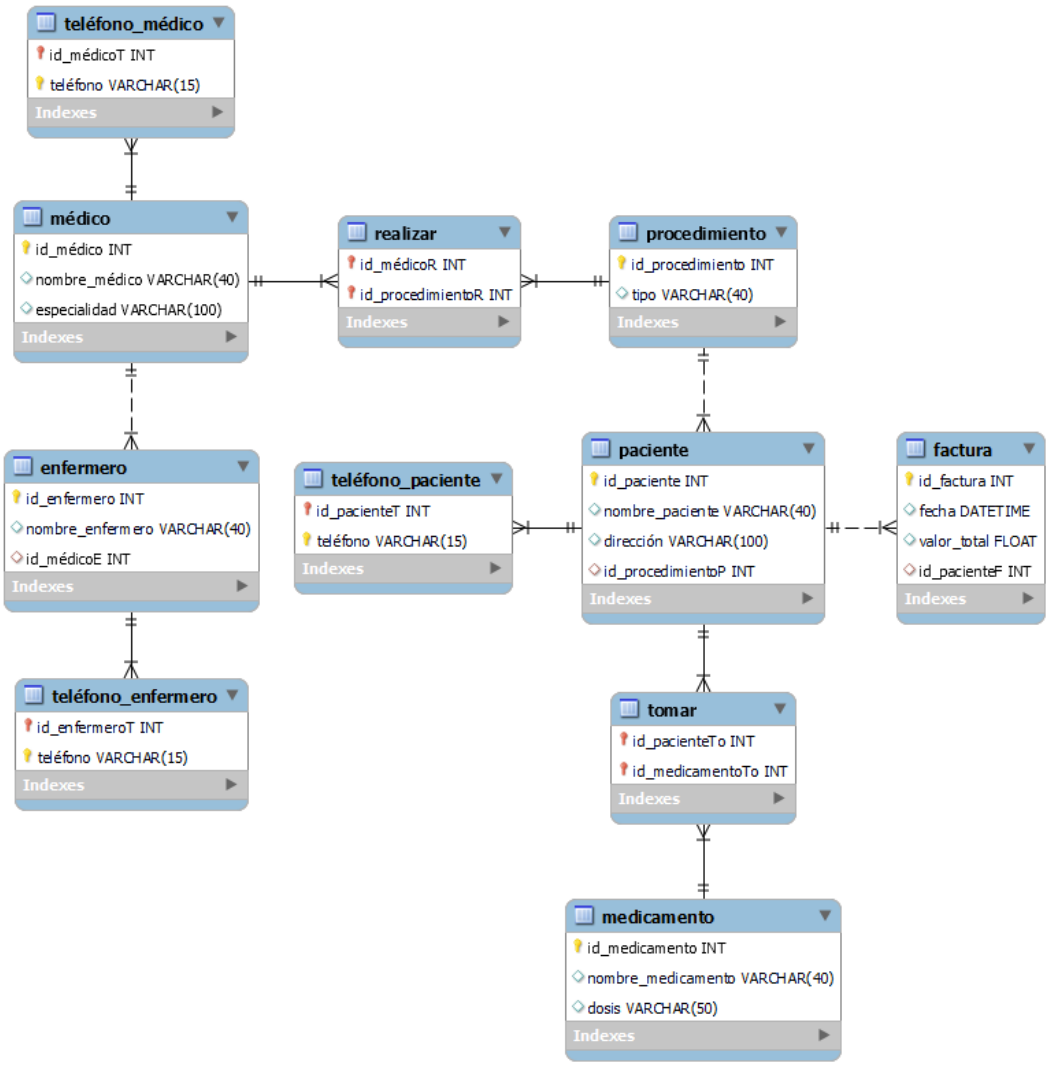

Figura 1 MR Hospital GNECJ.

Luego de tener la BD creada, se procede a llenar los registros, con 5 registros por tabla. Así, se presentan una a una la inserción de un solo registro por tabla para no saturar la documentación.

Se ingresa un médico a la BD del hospital, con su respectivo identificador, nombre y especialidad. En este caso un médico llamado Juan que es especialista cardiovascular. Además de la inserción de los teléfonos de los médicos en su correspondiente tabla.

```sql
-- Tabla Médico
INSERT INTO médico
VALUES
    (1, 'Juan', 'Cardiovascular'),
```

```sql
-- Tabla Teléfono Médico
INSERT INTO teléfono_médico
VALUES
    (1, '1111111111'),
```

De la misma manera para los enfermeros, a los que adicionalmente se les asocia a un médico en particular. En este caso una enfermera llamada Maria que asiste al médico con identificador 5, y su teléfono registrado en la tabla correspondiente.

```sql
-- Tabla Enfermero
INSERT INTO enfermero
VALUES
    (1, 'Maria', 5),
```

```sql
-- Tabla Teléfono Enfermero
INSERT INTO teléfono_enfermero
VALUES
    (1, '1111111110'),
```

En el caso de los procedimientos, se tiene la inserción de un procedimiento de tipo general, con identificador único 1. Además, en la tabla realizar, se muestra como el médico con identificador 1 realiza el procedimiento con identificador 5.

```sql
-- Tabla Procedimiento
INSERT INTO procedimiento
VALUES
    (1, 'General'),
```

```sql
-- Tabla Realizar
INSERT INTO realizar
VALUES
    (1, 5),
```

Luego, la tabla paciente es llenada en este caso con un paciente llamado Ernesto, identificado con el id 1, que vive en la dirección especificada y se realizará el procedimiento de id 5. También cuenta con su propio teléfono en su correspondiente tabla.

```sql
-- Tabla Paciente
INSERT INTO paciente
VALUES
    (1, 'Ernesto', 'Calle 1 # 1 - 1', 5),
```

```sql
-- Tabla Teléfono Paciente
INSERT INTO teléfono_paciente
VALUES
    (1, '1111111100'),
```

La tabla medicamento se le insertan los registros como en el ejemplo, donde la Aspirina se identifica con 1, y su dosis es de 100 mg. Además, en la tabla tomar, se inserta el registro de que el paciente de id 1 debe tomar el medicamento de id 1.

```sql
-- Tabla Medicamento
INSERT INTO medicamento
VALUES
    (1, 'Aspirina', '100 mg'),
```

```sql
-- Tabla Tomar
INSERT INTO tomar
VALUES
    (1, 1),
```

Finalmente, la tabla factura debe ser llenada con su identificador, en este caso 1, la fecha como se muestra, el valor que para el paciente 1, es de $100.

```
-- Tabla Factura
INSERT INTO factura
VALUES
    (1, '2023-01-01 01:00:00', 100, 1),
```

Para ejecutar las consultas solicitadas, se realiza una lógica de conexiones hacia atrás, es decir, conectando las tablas más alejadas en las relaciones entre tablas mediante las sentencias INNER JOIN. Luego, la primera consulta pretende conocer los medicamentos que ha tomado cada paciente y su dosis, entonces se muestran las columnas con su alias de nombre_paciente, nombre_medicamento y dosis. Con 2 INNER JOIN se consigue realizar el cruce entre las tablas y sus identificadores.

```
-- ------------------------------------------------------
-- Consulta Medicamentos que toma cada Paciente y su dosis
-- ------------------------------------------------------
SELECT P.nombre_paciente AS 'Paciente', M.nombre_medicamento AS 'Medicamento', M.dosis AS 'Dosis'
FROM medicamento AS M
INNER JOIN tomar AS T
ON M.id_medicamento = T.id_medicamentoTo
INNER JOIN paciente AS P
ON T.id_pacienteTo = P.id_paciente;
```

Resultando en:

Paciente	Medicamento	Dosis
Ernesto	Aspirina	100 mg
James	Paracetamol	20 mg
Mery	Fluoxetina	30 mg
Barbie	Lumigan	1 gota
Jhon	Omeprazol	20 mg

De la misma manera, para la segunda consulta solicitada, donde el objetivo es ver los enfermeros que asistieron a los procedimientos realizados a los pacientes. Entonces, las columnas seleccionadas son nombre_enfermero, tipo y nombre_paciente con sus respectivos alias. Estas columnas se pueden seleccionar gracias a todos los INNER JOIN que realizan el cruce tabla a tabla entre las diferentes tablas hasta conectar información común entre la tabla pacientes y enfermeros, que es lo que se solicita. Por este motivo fue necesario ir llamando las tablas de realizar y médico, ya que ahí radicaba la información común para relacionar enfermeros con procedimientos realizados a pacientes en específico.

```sql
-- ------------------------------------------------------
-- Consulta Enfermeros en los Procedimientos realizados
-- ------------------------------------------------------
SELECT E.nombre_enfermero AS 'Enfermero', P.tipo AS 'Tipo de procedimiento', PA.nombre_paciente AS 'Paciente'
FROM paciente AS PA
INNER JOIN procedimiento AS P
ON PA.id_procedimientoP = P.id_procedimiento
INNER JOIN realizar AS R
ON P.id_procedimiento = R.id_procedimientoR
INNER JOIN médico AS M
ON R.id_médicoR = M.id_médico
INNER JOIN enfermero AS E
ON M.id_médico = E.id_médicoE;
```

Resultando en:

Enfermero	Tipo de procedimiento	Paciente
Danna	Cardiovascular	Ernesto
Camila	Ortopédico	James
David	Neurológico	Mery
Julian	Estético	Barbie
Maria	General	Jhon

Finalmente, se crean tres vistas que se consideran de suma importancia para el Hospital, ya que con estas tendrá un acceso más limpio y rápido a la información más relevante. La primera vista propuesta es la vista del Stock de medicamentos que tiene el hospital, es decir, listar todos los medicamentos con su dosis recomendada. Así, se utilizan los alias para nombrar las columnas de una mejor manera y se llaman solo las columnas nombre_medicamento y dosis desde la tabla medicamentos.

```sql
-- ------------------------------------------------------
-- Stock de Medicamentos en el Hospital con su dosis recomendada
-- ------------------------------------------------------
CREATE VIEW stock AS
SELECT nombre_medicamento AS 'Medicamento',
       dosis AS 'Dosis recomendada'
FROM medicamento;
```

Medicamento	Dosis recomendada
Aspirina	100 mg
Paracetamol	20 mg
Fluoxetina	30 mg
Lumigan	1 gota
Omeprazol	20 mg

Luego, la segunda vista se trata de los médicos que hay en el hospital, es decir, el nombre del médico junto con su especialidad. Esto con el objetivo de que el hospital identifique qué médicos harían falta para contratar más. Así, se traen las columnas de nombre y especialidad de la tabla médico con sus correspondientes alias.

```sql
-- ------------------------------------------------------
-- Médicos que operan en el Hospital
-- ------------------------------------------------------
CREATE VIEW médicos AS
SELECT nombre_médico AS 'Médico',
       especialidad AS 'Especialidad'
FROM médico;
```

Médico	Especialidad
Juan	Cardiovascular
Luis	Ortopédico
Ana	Neuróloga
Carlos	Estético
Sara	General

Por último, la tercera vista muestra los enfermeros que hay en el hospital, es decir, el nombre del enfermero junto con el o los médicos que asiste. Esto con el objetivo de que el hospital identifique qué enfermeros harían falta para cada médico y así contratar más. Así, se traen las columnas de nombre y especialidad de la tabla médico con sus correspondientes alias.

```sql
-- ------------------------------------------------------
-- Enfermeros que asisten en el Hospital
-- ------------------------------------------------------
CREATE VIEW enfermeros AS
SELECT E.nombre_enfermero AS 'Enfermero',
       M.nombre_médico AS 'Asiste al Médico'
FROM enfermero AS E
INNER JOIN médico AS M
ON E.id_médicoE = M.id_médico;
```

Enfermero	Asiste al Médico
Maria	Sara
Julian	Carlos
David	Ana
Camila	Luis
Danna	Juan

Pregunta

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

Debido a que el modelo propuesto es bastante simple, se le pueden añadir bastantes cosas. Pero con el fin de mantener esta simplicidad, se sugiere agregar una conexión o relación entre procedimiento y factura. Esto haría que en la factura hubiera más detalle sobre el servicio que se está cobrando, incluso saber el doctor que realizó dicho procedimiento y dejarlo claro en la factura.

Tercera Actividad (Taller 6)

En este taller se utilizan las BD generadas en el taller anterior, por lo que es necesario tener cargadas las BD previamente y haber llenado los registros con los scripts SQL *FillingTables*.

Para la BD de la librería, se crean los procedimientos CRUD para la tabla autor. Así, el primero es el de agregar, donde a través de la consulta de insertar, se agrega un nuevo registro con los datos provistos al procedimiento al momento del llamado.

```
-- ------------------------------------------------------
-- Procedimiento Agregar
-- ------------------------------------------------------
DELIMITER //
CREATE PROCEDURE agregar_autor (IN idA VARCHAR(10),
                                IN fecha_nacimiento VARCHAR(45),
                                IN nacionalidadA VARCHAR(20),
                                IN nombreA VARCHAR(45))
BEGIN
INSERT INTO autor
VALUES
    (idA, fecha_nacimiento, nacionalidadA, nombreA);
END;
//
```

Una vez llamado el procedimiento, la tabla autor se ve modificada de la siguiente manera.

id	fecha de nacimiento	nacionalidad	nombre
1	1/1/1980	COL	Juan
2	1/2/1980	ESP	Luis
3	1/3/1980	VEN	José
4	1/4/1980	USA	Mark
5	1/5/1980	RUS	Alek
6	1/6/1980	BOL	Edward

El siguiente procedimiento es el de consultar un autor por su id. En este caso se utiliza la consulta de seleccionar, obteniendo las columnas relevantes del autor y renombrándolas con un alias, esto filtrando por el id que se recibe como parámetro de entrada al procedimiento. El resultado de llamar este procedimiento con un id 4, es el siguiente.

```
-- ------------------------------------------------------
-- Procedimiento Consultar
-- ------------------------------------------------------
DELIMITER //
CREATE PROCEDURE autor_por_id (IN idA VARCHAR(10))
BEGIN
SELECT nombre AS 'Nombre Autor',
       `fecha de nacimiento` AS 'Fecha de Nacimiento',
       nacionalidad AS 'Nacionalidad'
FROM autor
WHERE id = idA;
END;
//
```

Nombre Autor	Fecha de Nacimiento	Nacionalidad
Mark	1/4/1980	USA

Actualizar un campo en una tabla es peligroso si no se utilizan las palabras clave adecuadas, por eso el procedimiento de actualizar filtra por id el autor que se desea actualizar, recibiendo los valores que desea cambiar para todas sus columnas.

```
-- ------------------------------------------------
-- Procedimiento Actualizar
-- ------------------------------------------------
DELIMITER //
CREATE PROCEDURE actualizar_autor (IN idA VARCHAR(10),
                                   IN fecha_nacimiento VARCHAR(45),
                                   IN nacionalidadA VARCHAR(20),
                                   IN nombreA VARCHAR(45))
BEGIN
UPDATE autor
SET `fecha de nacimiento` = fecha_nacimiento,
    nacionalidad = nacionalidadA,
    nombre = nombreA
WHERE id = idA;
END;
//
```

Ahora el autor de id 6 tiene una nacionalidad diferente.

id	fecha de nacimiento	nacionalidad	nombre
1	1/1/1980	COL	Juan
2	1/2/1980	ESP	Luis
3	1/3/1980	VEN	José
4	1/4/1980	USA	Mark
5	1/5/1980	RUS	Alek
6	1/6/1980	CAN	Edward

De la misma manera, el tratar de borrar registros sin definir una condición, puede ser peligroso. Entonces el procedimiento de borrar autor por id se asegura de que se elimina el registro deseado a través de la sentencia DELETE

```
-- ------------------------------------------------
-- Procedimiento Borrar
-- ------------------------------------------------
DELIMITER //
CREATE PROCEDURE borrar_autor_por_id (IN idA VARCHAR(10))
BEGIN
DELETE FROM autor WHERE id = idA;
END;
//
```

Al llamar el procedimiento, e indicándole el id 6, se actualiza la tabla autores, y deja de estar el registro para ese id en concreto.

id	fecha de nacimiento	nacionalidad	nombre
1	1/1/1980	COL	Juan
2	1/2/1980	ESP	Luis
3	1/3/1980	VEN	José
4	1/4/1980	USA	Mark
5	1/5/1980	RUS	Alek

Para la BD del hospital, se crean los procedimientos CRUD para la tabla medicamento. Así, el primero es el de agregar, donde a través de la consulta de insertar, se agrega un nuevo registro con los datos provistos al procedimiento al momento del llamado.

```
-- ------------------------------------------------
-- Procedimiento Agregar
-- ------------------------------------------------
DELIMITER //
CREATE PROCEDURE agregar_medicamento (IN nombreM VARCHAR(40),
                                      IN dosisM VARCHAR(50))
BEGIN
INSERT INTO medicamento (nombre_medicamento, dosis)
VALUES
    (nombreM, dosisM);
END;
//
```

Una vez llamado el procedimiento, la tabla medicamento se ve modificada de la siguiente manera.

id_medicamento	nombre_medicamento	dosis
1	Aspirina	100 mg
2	Paracetamol	20 mg
3	Fluoxetina	30 mg
4	Lumigan	1 gota
5	Omeprazol	20 mg
6	Ibuprofeno	600 mg

El siguiente procedimiento es el de consultar un medicamento por su id. En este caso se utiliza la consulta de seleccionar, obteniendo las columnas relevantes del medicamento y renombrándolas con un alias, esto filtrando por el id que se recibe como parámetro de entrada al procedimiento. El resultado de llamar este procedimiento con un id 6, es el siguiente.

```
-- ------------------------------------------------
-- Procedimiento Consultar
-- ------------------------------------------------
DELIMITER //
CREATE PROCEDURE medicamento_por_id (IN idM INT)
BEGIN
SELECT nombre_medicamento AS 'Medicamento',
        dosis AS 'Dosis recomendada'
FROM medicamento
WHERE id_medicamento = idM;
END;
```

Medicamento	Dosis recomendada
Ibuprofeno	600 mg

Actualizar un campo en una tabla es peligroso si no se utilizan las palabras clave adecuadas, por eso el procedimiento de actualizar filtra por id el medicamento que se desea actualizar, recibiendo los valores que desea cambiar para todas sus columnas.

```
-- ------------------------------------------------
-- Procedimiento Actualizar
-- ------------------------------------------------
DELIMITER //
CREATE PROCEDURE actualizar_medicamento (IN idM INT,
                                         IN nombreM VARCHAR(40),
                                         IN dosisM VARCHAR(50))
BEGIN
UPDATE medicamento
SET nombre_medicamento = nombreM,
    dosis = dosisM
WHERE id_medicamento = idM;
END;
//
```

Ahora el medicamento de id 6 tiene una dosis recomendada diferente.

id_medicamento	nombre_medicamento	dosis
1	Aspirina	100 mg
2	Paracetamol	20 mg
3	Fluoxetina	30 mg
4	Lumigan	1 gota
5	Omeprazol	20 mg
6	Ibuprofeno	400 mg

De la misma manera, el tratar de borrar registros sin definir una condición, puede ser peligroso. Entonces el procedimiento de borrar medicamento por id se asegura de que se elimina el registro deseado a través de la sentencia DELETE

```
-- ------------------------------------------------
-- Procedimiento Borrar
-- ------------------------------------------------
DELIMITER //
CREATE PROCEDURE borrar_medicamento_por_id (IN idM INT)
BEGIN
DELETE FROM medicamento WHERE id_medicamento = idM;
END;
//
```

Al llamar el procedimiento, e indicándole el id 6, se actualiza la tabla medicamentos, y deja de estar el registro para ese id en concreto.

id_medicamento	nombre_medicamento	dosis
1	Aspirina	100 mg
2	Paracetamol	20 mg
3	Fluoxetina	30 mg
4	Lumigan	1 gota
5	Omeprazol	20 mg

Para finalizar, se realiza la construcción de los Triggers para ambas actividades. Básicamente será el mismo comportamiento para ambos ejercicios, donde en la actividad de la librería se almacena en una tabla nueva llamada control_de_cambios_librería, la información de quién agregó o eliminó un registro de la tabla autor gracias a los triggers implementados. Y en la actividad del hospital se almacena en una tabla nueva llamada control_de_cambios_hospital, la información de quién agregó o eliminó un registro de la tabla medicamento gracias a los triggers implementados.

Así, se crea la nueva tabla para la BD de la librería de la siguiente manera.

```
-- ----------------------------------------
-- Creando tabla para registros
-- ----------------------------------------
CREATE TABLE control_de_cambios_librería (
    usuario VARCHAR(100),
    acción VARCHAR(40),
    fecha DATETIME DEFAULT CURRENT_TIMESTAMP
);
```

Luego se crean los triggers de agregar autor y eliminar autor, así.

```
-- ----------------------------------------
-- Creando Trigger para registro Agregar
-- ----------------------------------------
DELIMITER //
CREATE TRIGGER ins_autor
AFTER INSERT ON autor
FOR EACH ROW
BEGIN
INSERT INTO control_de_cambios_librería
VALUES (USER(), 'AGREGAR', NOW());
END;
//
```

```
-- ----------------------------------------
-- Creando Trigger para registro Eliminar
-- ----------------------------------------
DELIMITER //
CREATE TRIGGER del_autor
AFTER DELETE ON autor
FOR EACH ROW
BEGIN
INSERT INTO control_de_cambios_librería
VALUES (USER(), 'ELIMINAR', NOW());
END;
//
```

Luego de insertar un nuevo registro, y posteriormente borrarlo, la nueva tabla queda con los registros que se muestran a continuación.

usuario	acción	fecha
root@localhost	AGREGAR	2023-02-14 23:34:49
root@localhost	ELIMINAR	2023-02-14 23:34:58

Por último, para la BD del hospital, se crea la nueva tabla que llevará los registros de los triggers, de la siguiente manera.

```
-- ----------------------------------------
-- Creando tabla para registros
-- ----------------------------------------
CREATE TABLE control_de_cambios_hospital (
    usuario VARCHAR(100),
    acción VARCHAR(40),
    fecha DATETIME DEFAULT CURRENT_TIMESTAMP
);
```

Luego se crean los triggers de agregar medicamento y eliminar medicamento, así.

```
-- ----------------------------------------
-- Creando Trigger para registro Agregar
-- ----------------------------------------
DELIMITER //
CREATE TRIGGER ins_medicamento
AFTER INSERT ON medicamento
FOR EACH ROW
BEGIN
INSERT INTO control_de_cambios_hospital
VALUES (USER(), 'AGREGAR', NOW());
END;
//
```

```
-- ----------------------------------------
-- Creando Trigger para registro Eliminar
-- ----------------------------------------
DELIMITER //
CREATE TRIGGER del_medicamento
AFTER DELETE ON medicamento
FOR EACH ROW
BEGIN
INSERT INTO control_de_cambios_hospital
VALUES (USER(), 'ELIMINAR', NOW());
END;
//
```

Luego de insertar un nuevo registro, y posteriormente borrarlo, la nueva tabla queda con los registros que se muestran a continuación.

usuario	acción	fecha
root@localhost	AGREGAR	2023-02-14 23:41:07
root@localhost	ELIMINAR	2023-02-14 23:41:15

ESTO PARECE UNA TESIS DE GRADO, ASÍ QUE LE DEDICO ESTE TRABAJO A JUAN PINEDA POR TANTO CONOCIMIENTO QUE NOS HA DADO, ASÍ COMO VARILLA.