18-7-2023

Cesar Gaston Petit Martin

CADENA DE GIMNASIOS

Segunda eNTREGA

CODERHOUSE

CURSO SQL

Contenido

[Definición 2](#_Toc140602914)

[Objetivo 2](#_Toc140602915)

[Necesidades 2](#_Toc140602916)

[Diagrama Entidad-Relación (conceptual) 3](#_Toc140602917)

[Diagrama Entidad Relación (DER) 4](#_Toc140602918)

[VISTAS GENERADAS 5](#_Toc140602919)

[VISTA 1 5](#_Toc140602920)

[VISTA 2 7](#_Toc140602921)

[VISTA 3 9](#_Toc140602922)

[VISTA 4 11](#_Toc140602923)

[STORES PROCEDURES 12](#_Toc140602924)

[STORES PROCEDURE 1 12](#_Toc140602925)

[STORES PROCEDURE 2 12](#_Toc140602926)

[FUNCIONES 13](#_Toc140602927)

[FUNCION 1 13](#_Toc140602928)

[TRIGGER 15](#_Toc140602929)

[TRIGGER 1 15](#_Toc140602930)

[TRIGGER 2 16](#_Toc140602931)

# Definición

El modelo que se va a utilizar es el de una cadena de gimnasios fitness, conformado por diversas sucursales, y que a su vez cuenta muchas actividades, en los distintos turnos.

# Objetivo

Crear un sistema de Base De Datos, que no permita llevar el correcto funcionamiento de la cadena de gimnasios y permitir un control de todos los aspectos correspondientes al modelo elegido, a su vez poder llevar una de control y seguimiento.

Necesidades

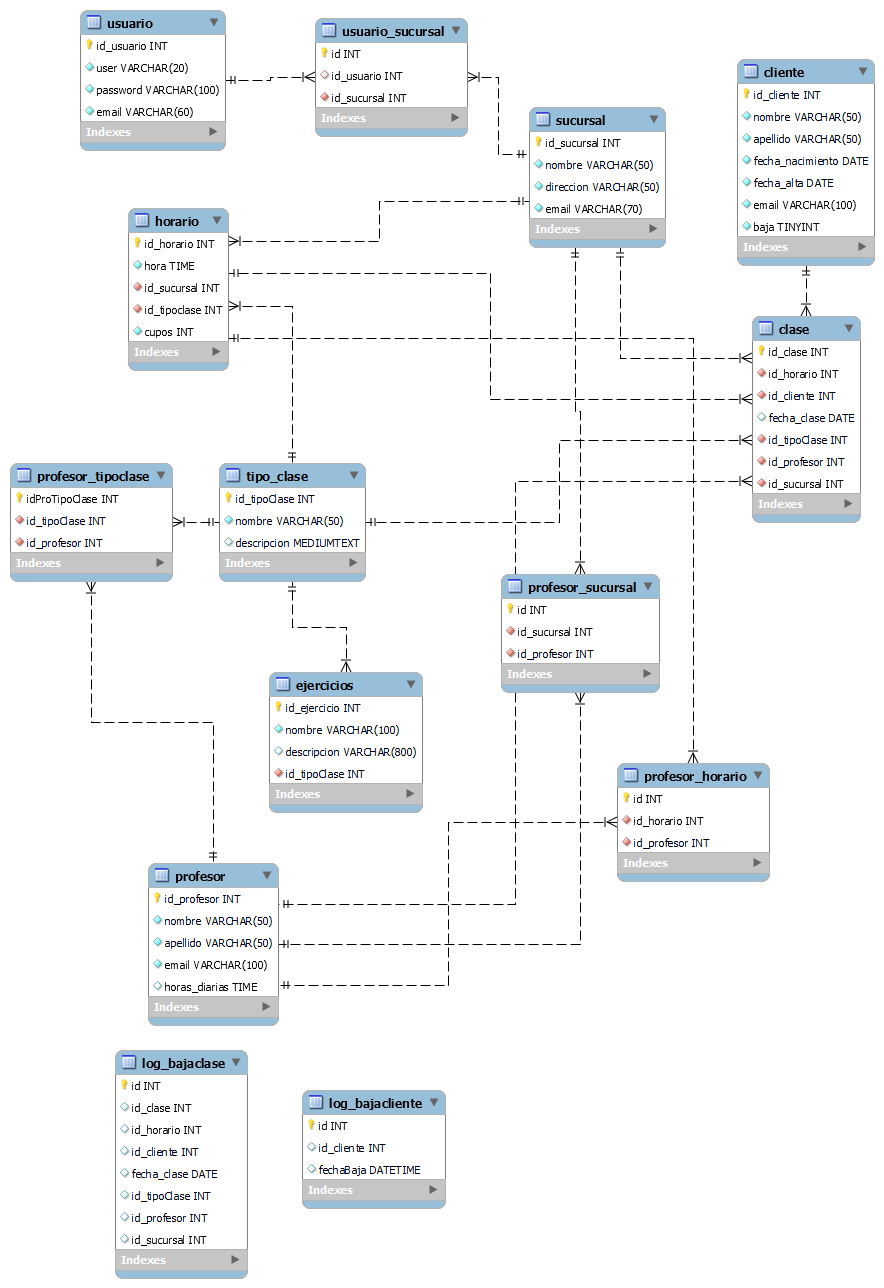
Podes administrar todas la cadenas del gimnasio , poder llevar un correcto control contables de los clientes y poder controlar la evolución de lo empleados que trabajan.

# Diagrama Entidad-Relación (conceptual)

Diagrama

Descripción generada automáticamente

# Diagrama Entidad Relación (DER)



# VISTAS GENERADAS

## VISTA 1

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**NOMBRE**: **v\_clases\_info**

**DEFINICION:** ESTA VISTA MUESTRA LA INFORMACIÓN DE LAS CLASES.

**ESQUEMA:**

CREATE

ALGORITHM = UNDEFINED

DEFINER = `root`@`localhost`

SQL SECURITY DEFINER

VIEW `workshop\_cesarpetit..v\_clases\_info `v` AS

SELECT

`c`.`id\_clase` AS `id\_clase`,

`h`.`hora` AS `hora`,

`c`.`fecha\_clase` AS `fecha\_clase`,

`s`.`nombre` AS `nombre\_sucursal`,

`p`.`nombre` AS `nombre\_profesor`,

`tc`.`nombre` AS `tipo\_clase`

FROM

((((`workshop\_cesarpetit`.`clase` `c`

JOIN `workshop\_cesarpetit`.`horario` `h` ON ((`c`.`id\_horario` = `h`.`id\_horario`)))

JOIN `workshop\_cesarpetit`.`sucursal` `s` ON ((`c`.`id\_sucursal` = `s`.`id\_sucursal`)))

JOIN `workshop\_cesarpetit`.`profesor` `p` ON ((`c`.`id\_profesor` = `p`.`id\_profesor`)))

JOIN `workshop\_cesarpetit`.`tipo\_clase` `tc` ON ((`c`.`id\_tipoClase` = `tc`.`id\_tipoClase`)))

**RESULTADO OBTENIDO:**

Tabla

Descripción generada automáticamente

## VISTA 2

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**NOMBRE:** **v\_clases\_por\_cliente :**

**DEFINICION:** ESTA VISTA MUESTRA LAS CLASES QUE HICIERON LOS CLIENTES

**ESQUEMA:**

CREATE

ALGORITHM = UNDEFINED

DEFINER = `root`@`localhost`

SQL SECURITY DEFINER

VIEW `workshop\_cesarpetit`.`v\_clases\_por\_cliente` AS

SELECT

`c`.`id\_cliente` AS `id\_cliente`,

CONCAT(`c`.`nombre`, ' ', `c`.`apellido`) AS `nombre\_completo`,

COUNT(`cl`.`id\_clase`) AS `total\_clases`

FROM

(`workshop\_cesarpetit`.`cliente` `c`

LEFT JOIN `workshop\_cesarpetit`.`clase` `cl` ON ((`c`.`id\_cliente` = `cl`.`id\_cliente`)))

GROUP BY `c`.`id\_cliente` , `nombre\_completo`

**RESULTADO OBTENIDO:**

**Tabla

Descripción generada automáticamente**

## VISTA 3

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**NOMBRE:**  **v\_cupos\_horario\_sucursal:**

**DEFINICION:** ESTA VISTA MUESTRA LOS CUPOS LIBRES POR CLASE

**ESQUEMA:**

**CREATE**

**ALGORITHM = UNDEFINED**

**DEFINER = `root`@`localhost`**

**SQL SECURITY DEFINER**

**VIEW `workshop\_cesarpetit`.`v\_cupos\_horario\_sucursal` AS**

**SELECT**

**`h`.`id\_horario` AS `id\_horario`,**

**`h`.`hora` AS `hora`,**

**`s`.`nombre` AS `nombre\_sucursal`,**

**`h`.`cupos` AS `total\_cupos`,**

**(`h`.`cupos` - COUNT(`cl`.`id\_clase`)) AS `cupos\_disponibles`**

**FROM**

**((`workshop\_cesarpetit`.`horario` `h`**

**JOIN `workshop\_cesarpetit`.`sucursal` `s` ON ((`h`.`id\_sucursal` = `s`.`id\_sucursal`)))**

**LEFT JOIN `workshop\_cesarpetit`.`clase` `cl` ON ((`h`.`id\_horario` = `cl`.`id\_horario`)))**

**GROUP BY `h`.`id\_horario` , `h`.`hora` , `s`.`nombre` , `h`.`cupos`**

**RESULTADO OBTENIDO:**

**Tabla

Descripción generada automáticamente**

## VISTA 4

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**NOMBRE:**  **vistacanttipoclase:**

**DEFINICION:** ESTA VISTA MUESTRA LA CANTIDAD DE CLASES QUE TIENE EL PROFESOR

**ESQUEMA:**

**CREATE**

**ALGORITHM = UNDEFINED**

**DEFINER = `root`@`localhost`**

**SQL SECURITY DEFINER**

**VIEW `workshop\_cesarpetit`.`vistacanttipoclase` AS**

**SELECT**

**CONCAT(`p`.`nombre`, ' ', `p`.`apellido`) AS `nombre Profesor`,**

**`tc`.`nombre` AS `nombre`,**

**COUNT(0) AS `Cantidad Clase`**

**FROM**

**((`workshop\_cesarpetit`.`clase` `c`**

**JOIN `workshop\_cesarpetit`.`profesor` `p` ON ((`p`.`id\_profesor` = `c`.`id\_profesor`)))**

**JOIN `workshop\_cesarpetit`.`tipo\_clase` `tc` ON ((`tc`.`id\_tipoClase` = `c`.`id\_tipoClase`)))**

**WHERE**

**(`p`.`id\_profesor` = 1)**

**GROUP BY `tc`.`id\_tipoClase`**

**RESULTADO OBTENIDO:**

**Imagen que contiene Aplicación

Descripción generada automáticamente**

# STORES PROCEDURES

## STORES PROCEDURE 1

**NOMBRE:**  **sp\_agregar\_profesor:**

**PARAMETROS:**

**P\_nombre: NOMBRE DEL PROFESOR**

**p\_apellido : APELLIDO DEL PROFESOR**

**p\_email: EMAIL DEL PROFESOR**

**p\_horasDiarias: HORAS DEL PROFESOR**

**DEFINICION:** AGREGA UN NUEVO PROFESOR AL STAFF

**ESQUEMA:**

**CREATE DEFINER=`root`@`localhost` PROCEDURE `sp\_agregar\_profesor`(in p\_nombre varchar(50), p\_apellido varchar(50), in p\_email varchar(100), in p\_horasDiarias time)**

**BEGIN**

**INSERT INTO profesor(nombre,apellido,email,horas\_diarias)**

**values(p\_nombre,p\_apellido,p\_email,p\_horasDiarias);**

**END**

## STORES PROCEDURE 2

**NOMBRE:** **sp\_ordenar\_clase**

**PARAMETROS :**

**Field : NOMBRE DE LA COLUMNA QUE QUIERO FILTRAR,**

**Orden: ELIGO EL ORDENAMIENTO ASC - DESC**

**DEFINICION:** DEVUELVE LA CLASES ORDENADO POR LOS PARAMETROS SELECCIONADOS.

# FUNCIONES

## FUNCION 1

**NOMBRE:** **profesoresCantidadHoras**

**PARAMETROS :**

**p\_horas: PARAMETRO QUE DEFINE LOS PROFESORES QUE TIENENE MAS DE ESAS HORAS.**

**DEFINICION:** DEVUELVE TODOS LO PROFESORES QUE HACEN LAS MISMAS O MAYORES HORAS QUE EL PARAMETRO DE ENTRADA.

**ESQUEMA:**

**CREATE DEFINER=`root`@`localhost` FUNCTION `profesoresCantidadHoras`( p\_horas time) RETURNS int**

**READS SQL DATA**

**BEGIN**

**declare profesoresHoras int ;**

**set profesoresHoras = (select count(\*) cantidadProfesores from profesor where horas\_diarias >=p\_horas);**

**return profesoresHoras;**

**END**

## FUNCION 2

**NOMBRE:** **mayor\_numero**

**PARAMETROS:**

**num1: NUMERO 1 PARA COMPARAR**

**num2: NUMERO 2 PARA COMPARAS**

**DEFINICION:** FUNCION QUE DEVUELVE CUAL ES EL NUMERO MAS GRANDE EN CASO DE IGUALDAD DEVUELVE IGUALES

**ESQUEMA:**

**delimiter $$**

**CREATE FUNCTION mayor\_numero(num1 int,num2 int) RETURNS varchar(50)**

**NO SQL**

**BEGIN**

**declare mayor varchar(50);**

**if (num1 > num2) then**

**set mayor = concat('el mayor numero es:',' ',CONVERT(num1, CHAR(20)));**

**end if;**

**if (num2 > num1) then**

**set mayor = concat('el mayor numero es:',' ',CONVERT(num2, CHAR(20)));**

**end if;**

**if (num1 = num2) then**

**set mayor = concat("Los numeros son iguales!!!!");**

**end if;**

**return mayor;**

**END$$**

**delimiter ;**

**RESULTADO OBTENIDO:**

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

Descripción generada automáticamente**

# 

# TRIGGER

## TRIGGER 1

**NOMBRE:** tr\_bajaUser

**DEFINICION:** SE CREO TRIGGER QUE CUANDO SE DA DE BAJA EL USUARIO DE MANERA LOGICA LO GUARDA EN EL LOG

**TABLAS QUE IMPACTA:** CLIENTE – LOG\_CLIENTE

**ESQUEMA:**

**delimiter $$**

**use workshop\_CesarPetit $$**

**CREATE TRIGGER tr\_bajaUser**

**AFTER UPDATE**

**ON cliente FOR EACH ROW**

**BEGIN**

**IF (OLD.baja <> NEW.baja AND NEW.BAJA = 1) THEN**

**INSERT INTO log\_bajaCLiente (id\_cliente,fechaBaja)**

**VALUES (NEW.id\_cliente,NOW());**

**END IF;**

**END $$**

## TRIGGER 2

**NOMBRE:** clase\_BEFORE\_DELETE

**DEFINICION:**  SE CREO UN TRIGGER QUE AL ELIMINAR LA CLASE DEL CLIENTE LO GUARDA EN EL LOG

**TABLAS QUE IMPACTA:** CLASE – LOG\_BAJACLASE

**ESQUEMA:**

**use workshop\_CesarPetit $$**

**CREATE TRIGGER `workshop\_CesarPetit`.`clase\_BEFORE\_DELETE` BEFORE DELETE ON `clase` FOR EACH ROW**

**BEGIN**

**INSERT INTO log\_bajaClase**

**VALUES(OLD.id\_clase,OLD.id\_horario,OLD.id\_cliente,OLD.fecha\_clase,OLD.id\_tipoClase,OLD.id\_profesor,OLD.id\_sucursal);**

**END$$**

**delimiter ;**