Academia Baile Mambolaye

Alumno: Ernesto Vega

Profesor: Miguel Rodas

Tutor: Juan Rabazzi

Descripción Temática:

La Academia Mambolaye es una escuela de baile que se dedica a enseñar diversos estilos de baile como Mambo (Salsa on2), Mambo Lady Style, Bachata Estilo Pacheco y próximamente se irán agregando más estilos.

Mambolaye se fundó en Venezuela y actualmente cuenta con un solo espacio de enseñanza en Buenos Aires, con miras de expansión. El nombre de la academia está compuesto por dos palabras **Mambo** que es el estilo de baile y **Laye** que significa calle, es decir, el estilo de baile con un toque propio de la persona que solo se obtiene fuera de salones de baile.

Descripción de tablas:

alumno (TD)						
Campo	**Tipo**	**Longitud**	**NOT NULL**	**PK**	**FK**	**Descripcion**
id_alumno	int	autoinc	x	x		Datos de los alumnos de la academia
dni	int		x			
nombre_alumno	varchar	50	x			
apellido_alumno	varchar	50	x			
telefono	int		x			
mail	varchar	120	x			
fecha_nacimiento	date	10	x			

tipo_pago (TD)						
Campo	**Tipo**	**Longitud**	**NOT NULL**	**PK**	**FK**	**Descripcion**
id_tipo_pago	int	autoinc	x	x		Informacion de los metodos de pago
descripcion_pago	varchar	20				

Membresia (TT)						
Campo	**Tipo**	**Longitud**	**NOT NULL**	**PK**	**FK**	**Descripcion**
id_membresia	int	autoinc	x	x		Datos de la membresia de cada alumno
id_alumno	int		x		x	
id_tipo_pago	int		x		x	
fecha_registro	Timestamp		x		x	

profesor (TD)						
Campo	**Tipo**	**Longitud**	**NOT NUL**	**PK**	**FK**	**Descripcion**
id_profesor	int	autoinc	x	x		Contiene informacion
dni	int		x			

nombre	varchar	50	x			de los profesores de baile
apellido	varchar	50	x			
mail	varchar	120	x			
telefono	int		x			

genero_musical (TD)						
Campo	**Tipo**	**Longitud**	**NOT NULL**	**PK**	**FK**	**Descripcion**
id_genero	int	autoinc	x	x		Descripcion del genero musical
nombre_genero	varchar	15	x			

estilo (TD)						
Campo	**Tipo**	**Longitud**	**NOT NULL**	**PK**	**FK**	**Descripcion**
id_estilo	int	autoinc	x	x		Estilos de baile relacionados con los generos musicales
nombre_estilo	varchar	20	x			
id_genero	int		x		x	

horario_clase (TD)						
Campos	**Tipo**	**Longitud**	**NOT NULL**	**PK**	**FK**	**Descripcion**
id_horario	int	autoinc	x	x		Tabla de horarios de clases
dia	varchar	20	x			
hora_inicio	Time					
hora_fin	time					

director(TD)						
Campos	**Tipo**	**Longitud**	**NOT NULL**	**PK**	**FK**	**Descripcion**
id_director	int	autoinc	x	x		Datos de los directores de las sedes
dni	int		x			
nombre	varchar	50	x			
apellido	varchar	50	x			

sede (TD)						
Campos	**Tipo**	**Longitud**	**NOT NULL**	**PK**	**FK**	**Descripcion**
id_sede	int	autoinc	x	x		Contiene informacion de la(s) sede(s) de la academia
nombre_sede	varchar	100	x			
direccion	varchar	150	x			
telefono	int					
id_director	int				x	

nivel (TD)						
Campo	**Tipo**	**Longitud**	**NOT NULL**	**PK**	**FK**	**Descripcion**
id_genero	int	autoinc	x	x		Niveles de baile
descripcion	varchar	20	x			

salon (TD)						
Campos	**Tipo**	**Longitud**	**NOT NULL**	**PK**	**FK**	**Descripcion**
id_salon	int	autoinc	x	x		Contiene datos de salon de practica
nombre_salon	varchar	50	x			
id_sede	int		x		x	

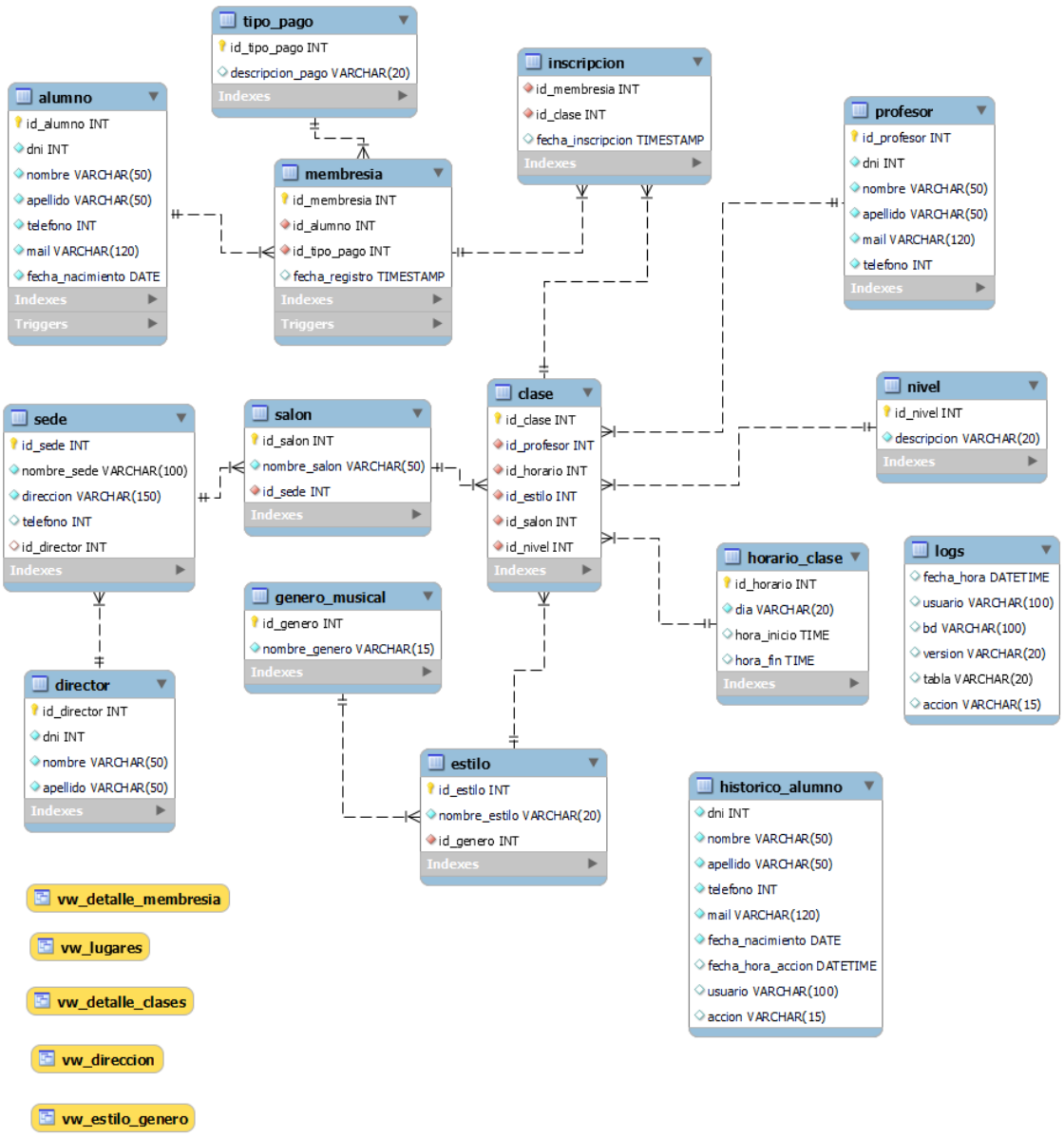
clase (TD)						
Campo	**Tipo**	**Longitud**	**NOT NULL**	**PK**	**FK**	**Descripcion**
id_clase	int	autoinc	x	x		Contiene informacion sobre las clases que brinda la academia
id_profesor	int		x		x	
id_horario	int		x		x	
id_estilo	int		x		x	
id_salon	int		x		x	
id_nivel	int		x		x	

inscripcion (TH)						
Campo	**Tipo**	**Longitud**	**NOT NULL**	**PK**	**FK**	**Descripcion**
id_registro	int	autoinc	x	x		Contiene datos de la inscripcion de cada alumno
fecha_registro	datetime	16	x			
tipo_pago	varchar	20	x			
id_clase	int		x		x	
id_alumno	int		x		x	

logs (TT)						
Campo	**Tipo**	**Longitud**	**NOT NULL**	**PK**	**FK**	**Descripcion**
fecha_hora	datetime	autoinc	x	x		Contiene informacion sobre las clases que brinda la academia
usuario	varchar	100	x		x	
bd	varchar	100	x		x	
version	varchar	20	x		x	
tabla	varchar	20	x		x	
accion	varchar	15	x		x	

historico_alumnos (TT)						
Campo	**Tipo**	**Longitud**	**NOT NULL**	**PK**	**FK**	**Descripcion**
dni	int		x			Contiene informacion sobre las clases que brinda la academia
nombre	varchar	50	x			
apellido	varchar	50	x			
telefono	int		x			
mail	varchar	120	x			
fecha_nacimiento	date		x			
fecha_hora_accion	datetime		x			
usuario	varchar	100	x			
accion	varchar	15	x			

Diagrama ER


tipo_pago
id_tipo_pago INT
descripcion_pago VARCHAR(20)
Indexes
inscripcion
id_membresia INT
id_clase INT
fecha_inscripcion TIMESTAMP
Indexes
alumno
id_alumno INT
dni INT
nombre VARCHAR(50)
apellido VARCHAR(50)
telefono INT
mail VARCHAR(120)
fecha_nacimiento DATE
Indexes
Triggers
membresia
id_membresia INT
id_alumno INT
id_tipo_pago INT
fecha_registro TIMESTAMP
Indexes
Triggers
profesor
id_profesor INT
dni INT
nombre VARCHAR(50)
apellido VARCHAR(50)
mail VARCHAR(120)
telefono INT
Indexes
clase
id_clase INT
id_profesor INT
id_horario INT
id_estilo INT
id_salon INT
id_nivel INT
Indexes
nivel
id_nivel INT
descripcion VARCHAR(20)
Indexes
sede
id_sede INT
nombre_sede VARCHAR(100)
direccion VARCHAR(150)
telefono INT
id_director INT
Indexes
salon
id_salon INT
nombre_salon VARCHAR(50)
id_sede INT
Indexes
horario_clase
id_horario INT
dia VARCHAR(20)
hora_inicio TIME
hora_fin TIME
Indexes
logs
fecha_hora DATETIME
usuario VARCHAR(100)
bd VARCHAR(100)
version VARCHAR(20)
tabla VARCHAR(20)
accion VARCHAR(15)
genero_musical
id_genero INT
nombre_genero VARCHAR(15)
Indexes
director
id_director INT
dni INT
nombre VARCHAR(50)
apellido VARCHAR(50)
Indexes
estilo
id_estilo INT
nombre_estilo VARCHAR(20)
id_genero INT
Indexes
historico_alumno
dni INT
nombre VARCHAR(50)
apellido VARCHAR(50)
telefono INT
mail VARCHAR(120)
fecha_nacimiento DATE
fecha_hora_accion DATETIME
usuario VARCHAR(100)
accion VARCHAR(15)
vw_detalle_membresia
vw_lugares
vw_detalle_clases
vw_direccion
vw_estilo_genero

Script SQL creación Schema:

```
#Habilitar eliminación y actualización en cascada
SET sql_safe_updates=0;

CREATE SCHEMA academia_mambolaye;

USE academia_mambolaye;

-- ALUMNOS
CREATE TABLE IF NOT EXISTS alumno(
id_alumno INT PRIMARY KEY NOT NULL AUTO_INCREMENT,
dni INT NOT NULL,
nombre VARCHAR(50) NOT NULL,
apellido VARCHAR(50) NOT NULL,
telefono INT NOT NULL,
mail VARCHAR(120) NOT NULL,
fecha_nacimiento DATE NOT NULL
);

-- TIPO DE PAGO
CREATE TABLE IF NOT EXISTS tipo_pago(
id_tipo_pago INT PRIMARY KEY NOT NULL AUTO_INCREMENT,
descripcion_pago VARCHAR(20)
);

-- MEMBRESIA
CREATE TABLE IF NOT EXISTS membresia(
id_membresia INT PRIMARY KEY NOT NULL AUTO_INCREMENT,
```

```
id_alumno INT NOT NULL,

id_tipo_pago INT NOT NULL,

fecha_registro TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,

FOREIGN KEY (id_alumno) REFERENCES
academia_mambolaye.alumno(id_alumno) ON DELETE CASCADE,

FOREIGN KEY (id_tipo_pago) REFERENCES
academia_mambolaye.tipo_pago(id_tipo_pago) ON DELETE CASCADE

);


-- PROFESOR

CREATE TABLE IF NOT EXISTS profesor(

id_profesor INT PRIMARY KEY NOT NULL AUTO_INCREMENT,

dni INT NOT NULL,

nombre VARCHAR(50) NOT NULL,

apellido VARCHAR(50) NOT NULL,

mail VARCHAR(120) NOT NULL,

telefono INT NOT NULL

);


-- GENERO MUSICAL

CREATE TABLE IF NOT EXISTS genero_musical(

id_genero INT PRIMARY KEY NOT NULL AUTO_INCREMENT,

nombre_genero VARCHAR(15) NOT NULL

);


-- ESTILO DE BAILE

CREATE TABLE IF NOT EXISTS estilo(

id_estilo INT PRIMARY KEY NOT NULL AUTO_INCREMENT,

nombre_estilo VARCHAR(20) NOT NULL,
```

```
id_genero INT NOT NULL,
FOREIGN KEY (id_genero) REFERENCES
academia_mambolaye.genero_musical(id_genero)
);

-- HORARIOS DE CLASES
CREATE TABLE IF NOT EXISTS horario_clase(
id_horario INT PRIMARY KEY NOT NULL AUTO_INCREMENT,
dia VARCHAR(20) NOT NULL,
hora_inicio TIME,
hora_fin TIME
);

-- DIRECTORES ACADEMIA
CREATE TABLE IF NOT EXISTS director(
id_director INT PRIMARY KEY NOT NULL AUTO_INCREMENT,
dni INT NOT NULL,
nombre VARCHAR(50) NOT NULL,
apellido VARCHAR(50) NOT NULL
);

-- SEDES
CREATE TABLE IF NOT EXISTS sede(
id_sede INT PRIMARY KEY NOT NULL AUTO_INCREMENT,
nombre_sede VARCHAR(100) NOT NULL,
direccion VARCHAR(150) NOT NULL,
telefono INT,
id_director INT,
FOREIGN KEY (id_director) REFERENCES director(id_director)
);
```

```
-- NIVEL DE BAILE
CREATE TABLE IF NOT EXISTS nivel(
id_nivel INT PRIMARY KEY NOT NULL AUTO_INCREMENT,
descripcion VARCHAR(20) NOT NULL
);

-- SALONES
CREATE TABLE IF NOT EXISTS salon(
id_salon INT PRIMARY KEY NOT NULL AUTO_INCREMENT,
nombre_salon VARCHAR(50) NOT NULL,
id_sede INT NOT NULL,
FOREIGN KEY (id_sede) REFERENCES academia_mambolaye.sede(id_sede)
);

-- CLASE
CREATE TABLE IF NOT EXISTS clase(
id_clase INT PRIMARY KEY NOT NULL AUTO_INCREMENT,
id_profesor INT NOT NULL,
id_horario INT NOT NULL,
id_estilo INT NOT NULL,
id_salon INT NOT NULL,
id_nivel INT NOT NULL,
FOREIGN KEY (id_profesor) REFERENCES
academia_mambolaye.profesor(id_profesor),
FOREIGN KEY (id_horario) REFERENCES
academia_mambolaye.horario_clase(id_horario),
FOREIGN KEY (id_estilo) REFERENCES academia_mambolaye.estilo(id_estilo),
FOREIGN KEY (id_salon) REFERENCES academia_mambolaye.salon(id_salon),
FOREIGN KEY (id_nivel) REFERENCES academia_mambolaye.nivel(id_nivel)
```

```
);

-- INSCRIPCION
CREATE TABLE IF NOT EXISTS inscripcion(
id_membresia INT NOT NULL,
id_clase INT NOT NULL,
fecha_inscripcion TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
FOREIGN KEY (id_membresia) REFERENCES academia_mambolaye.membresia(id_membresia) ON DELETE CASCADE,
FOREIGN KEY (id_clase) REFERENCES academia_mambolaye.clase(id_clase)
);

#LOGS
CREATE TABLE IF NOT EXISTS logs(
        fecha_hora DATETIME,
   usuario VARCHAR(100),
   bd VARCHAR(100),
   version VARCHAR(20),
   tabla VARCHAR(20),
   accion VARCHAR(15)
);

#BITACORA_ALUMNOS
CREATE TABLE IF NOT EXISTS historico_alumno(
dni INT NOT NULL,
nombre VARCHAR(50) NOT NULL,
apellido VARCHAR(50) NOT NULL,
telefono INT NOT NULL,
mail VARCHAR(120) NOT NULL,
```

```
fecha_nacimiento DATE NOT NULL,
fecha_hora_accion DATETIME,
usuario VARCHAR(100),
accion VARCHAR(15)
);
```

Vistas de la BD

vw_direccion

Objetivo: observar las sedes de la academia y sus respectivos directores

Tablas: sede y director

Script SQL:

```
create view academia_mambolaye.vw_direccion as
select a.nombre_sede as Sede
,concat(b.nombre,' ', b.apellido) as Director
from sede a
join director b
on (a.id_director = b.id_director);
```

---

vw_ estilo_genero

Objetivo: Debido que los estilos de baile pueden variar para un genero musical, mediante esta vista se puede obtener el genero y sus distintos tipos de estilo

Tablas: estilo y genero_musical

Script SQL:

```
create view academia_mambolaye.vw_estilo_genero as
SELECT est.nombre_estilo as 'Estilo de Baile',
gen.nombre_genero as 'Genero Musical'
```

```
from estilo est
join genero_musical gen
on (est.id_genero = gen.id_genero);
```

vw_detalle_clase

Objetivo: La tabla clase esta conformada por distintos ID que pertenecen a otras tablas, esto a la vista humana es poco entendible, por ende esta view asocia los distintos ID con su respectiva tablas trayendo la descripción de los mismos y así la información sea mas acorde al usuario.

Tablas: clase, estilo, profesor, horario_clas, nivel y salón

Script SQL:

```
create view academia_mambolaye.vw_detalle_clases as
select cla.id_clase
,est.nombre_estilo as 'Estilo Baile'
,concat(pro.nombre, ' ',pro.apellido) as 'Profesor(a)'
,concat(hc.dia, ' ', hc.hora_inicio, ' ', hc.hora_fin) as 'Horario'
,nv.descripcion 'Nivel'
,sl.nombre_salon 'Salon'
from clase cla
join estilo est
on (cla.id_estilo = est.id_estilo)
join profesor pro
on (cla.id_profesor = pro.id_profesor)
join horario_clase hc
on (cla.id_horario = hc.id_horario)
join nivel nv
on (cla.id_nivel = nv.id_nivel)
join salon sl
on (cla.id_salon = sl.id_salon);
```

vw_lugares

Objetivo: Observar cuales son los salones que están en cada sede

Tablas: salón y sede

Script SQL:

```
create view academia_mambolaye.vw_lugares as
select sed.nombre_sede
,sed.direccion
,sed.telefono
,sal.nombre_salon as Salon
from salon sal
join sede sed
on (sal.id_sede = sed.id_sede);
```

---

vw_membresia

Objetivo: Observar toda la información de los alumnos que han obtenido una membresía y también con que método de pago

Tablas: alumno, membresia y tipo_pago

Script SQL:

```
create view academia_mambolaye.vw_detalle_membresia as
select al.*
,pay.descripcion_pago
from alumno al
join membresia mem
on (mem.id_alumno = al.id_alumno)
join tipo_pago pay
on (mem.id_tipo_pago = pay.id_tipo_pago)
order by al.id_alumno asc;
```

Funciones

cant_alumnos_x_clase

Objetivo: Contabilizar cuantos alumnos están inscritos en una clase especifica indicando el id de la clase. Se ingresa un entero que corresponde al ID de la clase a consultar y la función realiza un select count de la tabla inscripción.

Script SQL:

```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` FUNCTION `cant_alumnos_x_clase`(p_id_clase INT) RETURNS int
  DETERMINISTIC
BEGIN
    DECLARE resultado INT;
  set resultado = (select count(*) from inscripcion where id_clase=p_id_clase);
    RETURN resultado;
END$$
DELIMITER ;
```

---

ult_alumno_x_clase

Objetivo: Retornar el nombre del último alumno que se registró a una clase en especifica. Se ingresa un entero que corresponde al ID de la clase a consultar y la función retornar una cadena de texto (varchar) con el nombre de la persona.

Script SQL:

```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` FUNCTION `ult_alumno_x_clase`(p_id_clase INT) RETURNS varchar(255)
READS SQL DATA
BEGIN
```

```
        DECLARE resultado varchar(255);
    set resultado = (select max(al.nombre) from inscripcion ins join membresia
mem on (ins.id_membresia = mem.id_membresia)
        join alumno al on (al.id_alumno = mem.id_alumno) join clase cla on
(ins.id_clase = cla.id_clase) where ins.id_clase = p_id_clase);
    RETURN resultado;
END$$
DELIMITER ;
```

Stored Procedures

sp_orden_x_campo

Objetivo: Mostrar datos ordenados de una tabla indicada por el usuario, esto permitirá al usuario que trabaje directamente en la BD ahorrar tiempo y código de sentencia ya que al llamar al SP solo deberá ingresar el nombre de la tabla, la columna y si es en orden ascendente o descendente.

Script SQL:

```
drop procedure if exists `sp_orden_x_campo`;
DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_orden_x_campo`(IN
p_table VARCHAR(100), IN p_field VARCHAR(100), IN p_order INT)
BEGIN
        IF p_field <> '' THEN
                SET @field = CONCAT(' ORDER BY ', p_field);
        ELSE
                SET @field = CONCAT(' ORDER BY 1');
        END IF;

    IF p_order = 1 THEN
      SET @ord = CONCAT(' ASC');
        ELSE
```

```
        SET @ord = CONCAT(' DESC');
    END IF;

  SET @sentence = CONCAT('SELECT * FROM academia_mambolaye.',p_table,
@field, @ord);
  PREPARE sentenceSQL FROM @sentence;
  EXECUTE sentenceSQL;
  DEALLOCATE PREPARE sentenceSQL;
END$$
DELIMITER ;
```

---

sp_delete_alumno

Objetivo: Eliminar alumno enviando por parámetro el DNI, el SP ayuda a evitar la posibilidad de eliminar registros por equivocación y omisión de la clausula where. Cuenta con una validación de longitud por si el usuario no ingresa por completo la secuencia de números que conforman un DNI.

Script SQL:

```
drop procedure if exists `sp_delete_alumno`;
DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_delete_alumno`(IN
p_dni INT)
BEGIN
    IF length(p_dni) <> 8 then
        SELECT 'ERROR: no se pudo eliminar el registro indicado';
    ELSE
        DELETE FROM academia_mambolaye.alumno WHERE dni
= p_dni;
    END IF;
END$$
DELIMITER ;
```