Universidad Autónoma de Baja California
Facultad de Ciencias Químicas e Ingeniería
Ingeniería en Computación
Semestre 2020-2
Gpo.561

Bases de Datos

Profr. Olivia Mendoza Duarte

Proyecto Final:
Base de Datos Musical

Miembros: Matrícula:

- Cruz Sánchez Miriam Fernanda 1253444
- Guerra Cervantes Sergio Enrique 1252819

Tijuana, Baja California, a Lunes 4 de Enero del 2020

Descripción de la problemática

Las disqueras musicales están pasando por una brecha tecnológica desde hace ya varios años, se necesita mantener un listado de sus artistas representados, su música e información, puesto que esta información crece día tras día, dado que surgen nuevos artistas.

Es por estas razones que se desea implementar una base de datos musical, la cual contendrá toda la información relevante lista para agregarse y ser consultada por usuarios de las disqueras. La página web dará un look más actual a la empresa y facilitará el trabajo sin necesidad de un experto en tiempo completo. Contendrá la infraestructura de la base de datos dando una interfaz amigable para realizar búsquedas rápidas y agregar nueva información específica de cada artista.

Descripción Organización-Cliente

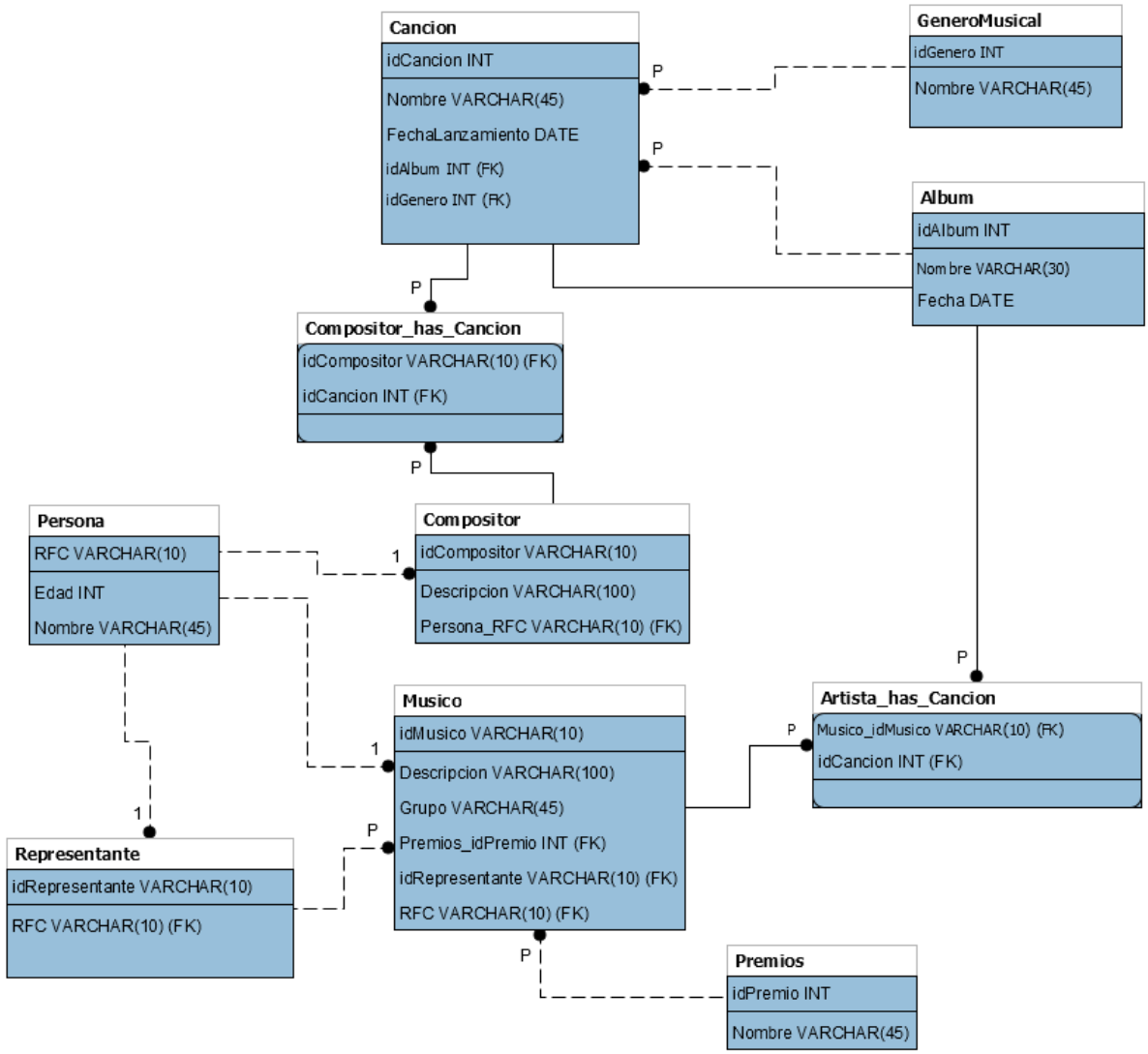
<table>
<tr><td colspan="3">Proyecto:
Base de Datos Musical</td><td>Fecha:
10/11/2020</td><td>Revisión:
1.0</td></tr>
<tr><td rowspan="2">Asociaciones Clave
MySQL Workbench:
Para la creación de los diagramas de la base de datos.

Adminer:
Para almacenar, mantener y actualizar la base de datos.</td><td>Actividades Clave
1. Registrar canciones, artistas y todos sus datos de interés.
2. Consultar datos sobre canciones y artistas.
3. Gestionar y mantener la información.</td><td rowspan="2">Propuesta de Valor
Gestión de información de utilidad para el sector musical.</td><td>Relación con los clientes
Los clientes podrán acceder a la base de datos para la consulta de las canciones, artistas y otros datos de interés, podrán integrar los datos a sus aplicaciones y servicios.</td><td rowspan="2">Segmentos de clientes
1. Empresas musicales.
2. Aplicaciones de música.
3. Estaciones de Radio.</td></tr>
<tr><td>Recursos Clave
1. Canciones.
2. Artistas.
3. Información adicional de soporte.</td><td>Canales de distribución
Todos los procesos de distribución serán llevados a cabo en Internet.</td></tr>
</table>

Historias de usuario

	Enunciado de la historia				Criterios de aceptación			
Identificador de la historia	**Rol**	**Característica / Funcionalidad**	**Razón / Resultado**	**No. de escenario**	**Título del Criterio de aceptación**	**Contexto**	**Evento**	**Resultado / Comportamiento esperado**
P00001	Programador	Realizar un diagrama de E-R, identificando identidades clave y llaves.	Con la finalidad de mantener la información actualizada en la base de datos.	1	Identidad Artista	Identidad Artista con atributos: nombre,grupo, premio	Cuando se registre la información en base de datos.	El sistema podrá realizar consultas acerca de los artistas.
				2	Identidad Representante	Identidad Representante con atributos: nombre y RFC	Cuando se registre la información en base de datos.	El sistema podrá realizar consultas acerca de los Representantes Asociados .
				3	Identidad Premios	Identidad Premios con atributos: nombre,grupo, premio	Cuando se registre la información en base de datos.	El sistema podrá realizar consultas acerca de los premios.
C00001	Cliente	Visualizar una página donde contenga toda la información de los clientes.	Con la finalidad de mantener la información de los clientes (artistas) .	1	Consulta Artistas	Búsqueda que relacione al Artista	Cuando se pueda consultar la información y sea presentada de forma clara.	El sistema podrá realizar consultas acerca de los artistas.
				2	Consulta de canciones	Búsqueda de canciones por género musical.	Cuando se pueda consultar las canciones por género musical	El sistema realizará separación por género musical.
				3	Consulta de Compositores	Listado de los compositores asociados	Cuando se pueda consultar los compositores.	El sistema realizará un listado de compositores.

Diagrama E-R


Cancion
idCancion INT
Nombre VARCHAR(45)
FechaLanzamiento DATE
idAlbum INT (FK)
idGenero INT (FK)
GeneroMusical
idGenero INT
Nombre VARCHAR(45)
Album
idAlbum INT
Nombre VARCHAR(30)
Fecha DATE
Compositor_has_Cancion
idCompositor VARCHAR(10) (FK)
idCancion INT (FK)
Persona
RFC VARCHAR(10)
Edad INT
Nombre VARCHAR(45)
Compositor
idCompositor VARCHAR(10)
Descripcion VARCHAR(100)
Persona_RFC VARCHAR(10) (FK)
Musico
idMusico VARCHAR(10)
Descripcion VARCHAR(100)
Grupo VARCHAR(45)
Premios_idPremio INT (FK)
idRepresentante VARCHAR(10) (FK)
RFC VARCHAR(10) (FK)
Artista_has_Cancion
Musico_idMusico VARCHAR(10) (FK)
idCancion INT (FK)
Representante
idRepresentante VARCHAR(10)
RFC VARCHAR(10) (FK)
Premios
idPremio INT
Nombre VARCHAR(45)
P
1

Sentencias SQL y programas PL/SQL

Procedimientos

- **buscaArtista**

  Justificación

  Es requerimiento del cliente que se pueda buscar el artista que está en su empresa, pero además se agrega el nombre de este cantante y el nombre de los premios para conocer más allá del artista.

  Código

```
DROP PROCEDURE IF EXISTS buscaArtista;

DELIMITER !!
CREATE PROCEDURE buscaArtista(subcadena VARCHAR(45))
BEGIN
    SELECT Musico.idMusico,Persona.Nombre
,Musico.Descripcion,Musico.Grupo,Premios.Nombre as NombrePremiacion FROM
Musico
    INNER JOIN Persona ON Musico.RFC = Persona.RFC
    INNER JOIN Premios ON Musico.Premios_idPremio = Premios.idPremio
    WHERE Persona.Nombre=subcadena;
END!!
```

- **buscaCancion**

  Justificación

  Para conocer la información de la canción además de ser una búsqueda, se implementa y se conoce el nombre del álbum y el nombre del género al que pertenece.

  Código

```
DROP PROCEDURE IF EXISTS buscaCancion;

DELIMITER !!
CREATE PROCEDURE buscaCancion(subcadena VARCHAR(45))
BEGIN
    SELECT Cancion.idCancion,Cancion.Nombre as
NombreCancion,Cancion.FechaLanzamiento,Album.Nombre as
NombreAlbum,GeneroMusical.Nombre as NombreGenero FROM Cancion
    INNER JOIN Album ON Cancion.idAlbum = Album.idAlbum
    INNER JOIN GeneroMusical ON Cancion.idGenero = GeneroMusical.idGenero
    WHERE Cancion.Nombre = subcadena;

END!!
```

- **listaAlbum**

  Justificación

  Se da a conocer el contenido del álbum con el nombre de la canción y su género musical.

  Código

```
DROP PROCEDURE IF EXISTS listaAlbum;

DELIMITER !!
CREATE PROCEDURE listaAlbum(subcadena VARCHAR(45))
BEGIN
    SELECT Album.idAlbum as idAlbum, Album.Nombre as Album, Cancion.Nombre
as Cancion, GeneroMusical.Nombre as Genero
    FROM Cancion
    INNER JOIN Album ON Cancion.idAlbum = Album.idAlbum
    INNER JOIN GeneroMusical ON Cancion.idGenero = GeneroMusical.idGenero
    WHERE Album.Nombre=subcadena;
END!!
```

- **listaGenero**

  Justificación

  Para el cliente la clasificación por género es importante ya que mientras maneje más artistas su base de datos se amplía y con eso facilitará el listado.

  Código

```
DROP PROCEDURE IF EXISTS listaGenero;

DELIMITER !!
CREATE PROCEDURE listaGenero(subcadena VARCHAR(45))
BEGIN
    SELECT GeneroMusical.idGenero as idGenero, GeneroMusical.Nombre as
Genero, Cancion.Nombre as Cancion, Album.Nombre as Album
    FROM Cancion
    INNER JOIN Album ON Cancion.idAlbum = Album.idAlbum
    INNER JOIN GeneroMusical ON Cancion.idGenero = GeneroMusical.idGenero
    WHERE GeneroMusical.Nombre=subcadena;
END!!
```

- **listaArtista**

  Justificación

  Para el cliente la clasificación por Artista es importante ya que mientras maneje más artistas su base de datos se amplía y con eso facilitará encontrar al artista.

  Código

```
DROP PROCEDURE IF EXISTS listaArtista;

DELIMITER !!
CREATE PROCEDURE  listaArtista(subcadena VARCHAR(45))
BEGIN
    SELECT Musico.idMusico as ArtistaID, Persona.Nombre as Artista,
Cancion.Nombre as Cancion
    FROM Artista_has_Cancion
    INNER JOIN Musico ON
Artista_has_Cancion.Musico_idMusico=Musico.idMusico
    INNER JOIN Persona ON Musico.RFC=Persona.RFC
    INNER JOIN Cancion ON Artista_has_Cancion.idCancion=Cancion.idCancion
    WHERE Persona.Nombre=subcadena;
END!!
```

Triggers

- **albumNuevo**

  Justificación

  Si se agrega un nuevo álbum la cantidad de canciones empezará a contar en cero para así asegurar que no se haga un conteo erróneo.

  Código

```
DROP TRIGGER IF EXISTS albumNuevo;

DELIMITER !!
CREATE TRIGGER albumNuevo
BEFORE INSERT ON Album
FOR EACH ROW
BEGIN
    IF NEW.CantidadCanciones<>1 THEN
        SET NEW.CantidadCanciones = 0;
    END IF;
END;!!
```

- **nuevaCancionAlbum**

  Justificación

  Con la implementación de este trigger se incrementa el número de cantidad de canciones siempre y cuando pertenezcan al mismo álbum.

  Código

```
DROP TRIGGER IF EXISTS nuevaCancionAlbum;

DELIMITER !!
CREATE TRIGGER nuevaCancionAlbum
AFTER INSERT ON Cancion
FOR EACH ROW
BEGIN
    UPDATE Album
    SET cantidadCanciones = cantidadCanciones+1
    WHERE idAlbum=NEW.idAlbum;
END;!!
```

Funciones

- **clasificaPersona**

  Justificación

  Con solo el RFC de una persona se puede conocer si una persona es cantante o representante sólo conociendo sus 4 primeros caracteres. Con la función recibimos el RFC completo y le quitamos el resto que no nos interesa, retornamos si es cantante o representante para que se conozca más rápido el paradero de la persona.

  Código

```
-- FUNCION COMPARACION PARA SABER SI UNA PERSONA ES CANTANTE O COMPOSITOR
DELIMITER !!
CREATE FUNCTION clasificaPersona(RFC VARCHAR(10))
RETURNS VARCHAR(13)
BEGIN
    DECLARE AUX VARCHAR(4);
    DECLARE tipo VARCHAR(13);
    SET AUX = SUBSTRING(RFC,1,4);
    IF strcmp(AUX,'CANT') = 0
        THEN SET tipo = 'CANTANTE';
    ELSE
        SET tipo = 'REPRESENTANTE';
    END IF;
    RETURN tipo;
END;!!
DELIMITER ;
```

- **cancionesAnuales**

  Justificación

  La separación y el manejo de la información es importante es por esto que la clasificación de canciones por año nos da a conocer el movimiento de la industria y de los artistas a lo largo de los años.

  Código

```
DROP FUNCTION IF EXISTS cancionesAnuales;
DELIMITER !!
CREATE FUNCTION cancionesAnuales(year INT)
RETURNS INT
BEGIN
    DECLARE res INT;
    SET res=0;
    SELECT SUM(res+1) AS CancionesAnuales INTO res
    FROM Cancion
    WHERE year=YEAR(Cancion.FechaLanzamiento);
    RETURN res;
END;!!
```

Bibliografía consultada

- ORACLE. (2021). MySQL :: MySQL 8.0 Reference Manual :: 13.6.5.2 IF Statement. MySQL. https://dev.mysql.com/doc/refman/8.0/en/if.html
- coffee cup Color Palette. (s. f.). color-hex. Recuperado 4 de enero de 2021, de https://www.color-hex.com/color-palette/200