

Proyecto Final Musify

Administración de Bases de Datos

**Facultad de Ingeniería Informática**

**II Cuatrimestre 2016**

**Profesor:**

Gabriel Alcázar Jiménez

**Estudiante:**

Arnaldo Porras Corrales

San José, Costa Rica

Urbanización Turnón

19 de agosto, 2016

Tabla de contenido

Objetivo General 4

Objetivos específicos 4

Alcances 5

Limitaciones 5

Análisis 6

Descripción del entorno de negocio 6

Identificación de reglas de negocio a nivel de datos. 7

Inventario de Datos y Definición de Reglas de Integridad 8

Inventario de entidades 10

Inventario de Interrelaciones 10

Definición de disparadores 10

Identificación de reglas de negocio a nivel de aplicación. 11

Paquetes y procedimientos 11

Transacciones 12

Definición de Transacciones 12

Identificación de permisos 13

Definición de Roles y Esquema de Seguridad 13

Concurrencia 14

Definición e Implementación de Aspectos de Concurrencia 14

Modelo Conceptual 15

Modelo Lógico 15

Modelo relacional 16

Modelo Físico 16

Exclusión de tablas 16

Creación de Tablas 16

Inserción de Datos 18

Definición de Datos 19

Triggers 21

Secuencia de datos 21

Paquetes y Procedimientos de Datos 22

Paquete Administrador 22

Paquete Cliente 25

Paquete Presentador 27

Seguridad 28

Roles 28

Introducción

Para este proyecto se pretende efectuar todas las especificaciones planteadas para el mismo, por ende, se estará realizando un análisis para una idea de negocio que no se ha desarrollado. Esta consta de una aplicación web para clientes que buscan contratar músicos para eventos. Se creará un prototipo y un modelo de datos que cumpla con las necesidades de esta idea de negocios. Además se creará una base de datos mediante las direcciones necesarias para así poder utilizarlo en este sistema web que ayudará a clientes y músicos a poder conectarse de una manera más sencilla y rápida que se amolda a la idea de negocio.

Objetivos

# Objetivo General

Con este trabajo se trata de plantear y desarrollar una primera etapa de una posible aplicación web en la cual clientes puedan contratar a músicos o bandas para participar en eventos. La idea es crear un prototipo completamente funcional de una plataforma web en la que músicos y clientes puedan interactuar de una manera más sencilla.

# Objetivos específicos

* Crear un prototipo funcional de una aplicación web para el manejo de datos.
* Desarrollar una base de datos relacional que pueda estar conectada a una aplicación por medio de servicios web.
* Desarrollar un modelo lógico y uno relacional que permita generar los comandos adecuados para la creación del modelo de la bases de datos.
* Aplicar el proceso de normalización del modelo de base de datos creado, con el fin de evitar problemas en el funcionamiento de la base de datos y optimizarlo al máximo.
* Crear el modelo físico en Oracle SQL para asegurar que el modelo de la base de datos sea completamente funcional.

Alcances y limitaciones

# Alcances

Se está considerando una aplicación web ficticia. Solo es están contemplando las posibilidades de músicos y bandas junto con un cliente que es el encargado de crear eventos.

La base de datos que se creará es para uso de la aplicación web (sin nombre todavía), por lo que se pretende que esta pueda servir en las siguientes fases del desarrollo de la aplicación.

# Limitaciones

Por ser una aplicación ficticia, no es posible conocer completamente los necesidades que se tengan al momento del desarrollo final. Sin embargo, se tiene como punto de partida la idea inicial de la aplicación junto con las reglas de negocio básicas para el funcionamiento de un prototipo.

Puede ocurrir que el prototipo acá planteado sea cambiado conforme el avance en la idea de negocio, debido a que las necesidades de los involucrados cambien teniendo que modificar el modelo acá planteado para adaptarse a nuevas necesidades.

Capitulo I – Definición de entorno

# Análisis

El modelo acá planteado está basado en un modelo ficticio. La idea está pensada para así poder desarrollarla como un modelo real en un futuro a mediano plazo. Entonces, el negocio no existe más que en una idea de negocio por lo cual se crea un modelo ficticio.

# Descripción del entorno de negocio

La idea de negocio es crear una plataforma en la que personas puedan crear eventos en los que se solicita los servicios de un músico o de una banda (inicialmente). De esta forma tanto músicos como bandas tienen un medio en el que pueden buscar eventos dependiendo del estilo de música que ellos desempeñen. Dando la oportunidad de generar nuevas conexiones y oportunidades de negocios conectando a clientes con músicos o bandas.

Para los clientes es una plataforma para solucionar el problema que podrían encontrar a la hora de crear un evento en el que se quiera contratar a un músico o una banda. Por lo general los organizadores de eventos podrían tener un conocimiento reducido de la cantidad de bandas disponibles que quieran participar en un evento o si una banda que tal vez ellos les gusta toquen en el evento que ellos quieren organizar.

Para los músicos y bandas les da la oportunidad de generar nuevas oportunidades de negocios, aumentar su exposición o simplemente darse a conocer entre las personas, tanto las que los buscan para el evento como para las persona que eventualmente podrán asistir al evento.

El cliente va a crear un evento con la información sobre este y se ocupa un músico o una banda. Los eventos se van a mostrar en una página donde están todos los eventos y se muestra su información y detalles. La página podrá desplegar para el cliente tanto los eventos para músicos o seleccionar los que son de bandas. El cliente puede llevar un registro de los eventos que ha creado.

Por su lado, las bandas y músicos no pueden crear eventos, pero pueden ver los eventos que están disponibles dependiendo de su categoría. Eso quiere decir que los músicos solo podrán ver los eventos para músicos y de igual manera para las bandas.

# Identificación de reglas de negocio a nivel de datos.

1. Un cliente puede crear un evento, en el que se especifica la hora del evento (13:00), el día (13/10/16) y el lugar del evento (Hotel Herradura). Además de esto, el cliente debe escoger qué tipo de presentador necesita, si necesita un ‘Músico’ o una ‘Banda’ y el género del evento.
2. Un evento tiene el estatus de ‘Abierto’, ‘Pendiente’ o ‘Cerrado’. *Abierto* se refiere a que no tiene presentador asignado, *pendiente* a que está a la espera de aprobación del cliente y *cerrado* que ya tiene asignado un presentador,
3. Un cliente tiene que estar registrado con el número de cédula (100000000), su nombre (Juan), su primer apellido (Pérez) y opcional el segundo apellido (Vargas). Además se tiene que brindar un número de teléfono (88888888) y opcionalmente un correo electrónico (nombre@ejemplo.com).
4. El cliente tiene estado de ‘Activo’ o ‘Inactivo’, que dependerá si ha creado eventos.
5. El tipo de presentador se refiere al tipo de necesidad, sea un ‘Músico’ o una ‘Banda’. Este tendrá un identificador único (01).
6. Un Presentador tiene que estar registrado con su número de cédula (100000001), su nombre artístico (Marta y los del barrio), y un teléfono (83838383). Opcionalmente un correo electrónico (musico@ejemplo.com).
7. Cada presentador tiene un raiting (calificación) que será cero (0) a la hora que abran la cuenta y conforme tengan eventos irá cambiando
8. Los instrumentos son identificados por un código (1001) y de un nombre que tiene que ser único (guitarra).
9. Los presentadores pueden tener un solo instrumento asociado, pero no es obligatorio a menos que sea músico. Este se representará en la tabla de presentador por medio del código del instrumento (1001).
10. El género musical se debe de registrar por medio de un código (101). Además se identificará por medio de un nombre que ha de ser único (Rock).
11. Todos los presentadores tanto músicos como bandas, deben de tener al menos un género musical. Pueden tener varios géneros musicales que van a ser representados en su respectiva tabla por medio del código del género (101).

# Inventario de Datos y Definición de Reglas de Integridad

Cliente

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Atributo | Tipo | Intensión | Extensión | Supuesto | Validaciones |
| Cédula cliente | A.I.P | Number | 9 | 3 | Not Null, igual a 9 |
| Nombre cliente |  | Varchar2 | 25 | 3 | Not Null |
| Apellido1 |  | Varchar2 | 25 | 3 | Not Null |
| Apellido2 |  | Varchar2 | 25 | 3 |  |
| Teléfono |  | Number | 8 | 3 | Not Null, igual a 8 |
| email |  | Varchar2 | 60 | 3 |  |
| Estado |  | Varchar2 | 25 | 3 | Check in: ‘Activo’, ‘Inactivo’ |

Tipo Presentador

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Atributo | Tipo | Intensión | Extensión | Supuesto | Validaciones |
| Id tipo | A.I.P | Number | 10 | 5 | Not Null |
| Nombre tipo |  | Varchar2 | 10 | 5 | Not Null |

Evento

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Atributo | Tipo | Intensión | Extensión | Supuesto | Validaciones |
| Fecha | A.I.P | Date |  | 1 | Not Null |
| Hora | A.I.P | Timestamp |  | 1 | Not Null |
| Lugar |  | Varchar2 | 50 | 1 | Not Null |
| Status |  | Varchar2 | 25 | 1 | Not Null |
| Cédula presentador | A.I.F | Number | 9 | 1 | Not Null |
| Id tipo | A.I.F | Number | 10 | 1 | Not Null |
| Cédula cliente | A.I.F | Number | 9 | 1 | Not Null |
| Id género | A.I.F | Number | 10 | 1 | Not Null |

Instrumento

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Atributo | Tipo | Intensión | Extensión | Supuesto | Validaciones |
| Id instrumento | A.I.P | Number | 10 | 8 | Not Null |
| Nombre instrumento |  | Varchar2 | 20 | 8 | Not Null, Unique |

Presentador

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Atributo | Tipo | Intensión | Extensión | Supuesto | Validaciones |
| Cédula presentador | A.I.P | Number | 9 | 6 | Not Null, igual a 9 |
| Nombre artístico |  | Varchar2 | 25 | 6 | Not Null |
| Teléfono |  | Number | 8 | 6 | Not Null, igual a 8 |
| email |  | Varchar2 | 60 | 6 |  |
| Rating |  | Varchar2 | 2 | 6 | Not Null |
| Id tipo | A.I.F | Number | 10 | 6 | Not Null |
| Id Instrumento | A.I.F | Number | 10 | 6 | Not Null |

Género

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Atributo | Tipo | Intensión | Extensión | Supuesto | Validaciones |
| Id género | A.I.P | Number | 10 | 10 | Not Null |
| Nombre género |  | Varchar2 | 20 | 10 | Not Null, Unique |

Género presentador

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Atributo | Tipo | Intensión | Extensión | Supuesto | Validaciones |
| Músico cédula músico | A.I.F | Number | 9 | 11 | Not Null |
| Género id género | A.I.F | Number | 10 | 11 | Not Null |

# Inventario de entidades

|  |  |  |
| --- | --- | --- |
| Columna | Número de supuesto | Entidad |
| 1 | 1, 3, 4 | Cliente |
| 2 | 10, 11 | Género |
| 3 | 8, 9 | Instrumento |
| 4 | 5 | Tipo Presentador |
| 5 | 6, 7, 9, 11 | Presentador |
| 6 | 1, 2 | Evento |

# Inventario de Interrelaciones

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Supuesto | Entidades que involucra | | Nombre Relación | Cardinalidad de la interrelación | Dependencia |
| Supuesto 1 | Cliente | Evento | Crea | 1:N | E |
| Tipo Presentador | Género | Crea |
| Supuesto 5 | Evento | Tipo Presentador | Tiene | 1:N | E |
| Tiene |
| Supuesto 9 | Presentador | Instrumento | Toca | 1:N | E |
| Toca |
| Supuesto 11 | Presentador | Género | Selecciona | N:M | ID |
| Selecciona |

# Definición de disparadores

Los siguientes disparadores se crearon para el presente modelo:

* Si el presentador quiere aplicar para un evento, este tiene que tener estado ‘Abierto’, si está ‘Pendiente’ o ‘Cerrado’ no podrá aplicar para dicho evento.
* Si el presentador es tipo ‘Banda’ este no puede ingresar un tipo de instrumento.
* Cuando un presentador quiere aplicar para un evento, alguno de los géneros del presentador tiene que coincidir con el género para ese evento. Un evento que tiene como género ‘Clásico’ una banda que no tiene este género no puede aplicar.

# Identificación de reglas de negocio a nivel de aplicación.

1. En la página de “Eventos” se muestran los eventos disponibles. Si se es un cliente se muestran los eventos creados por el cliente; si se es músico o una banda solo se muestran los eventos con su respectiva clasificación.
2. Cuando un músico/banda selecciona un evento este se bloquea. Si el cliente acepta la propuesta y la acepta; el evento se elimina de los eventos disponibles. Si la oferta es rechazada, el evento vuelve a estar disponible.
3. Las bandas y los músicos, tienen una página donde pueden ver sus eventos próximos o pendientes. Además pueden ver un historial de los eventos que ya han tenido.
4. Cuando la propuesta del músico o banda es aceptada, el evento pasa a estar en su página de eventos pendientes.
5. Un músico o banda pueden tener varios eventos en la misma fecha pero no en la misma hora.
6. Cada músico y banda, tienen un perfil personal con su información general como el género que tocan. En el caso de los músicos, además se mostrará el instrumento que tocan.

# Paquetes y procedimientos

Se crearán los siguiente procedimientos almacenados para el proyecto:

* En el paquete para los presentadores contendrá procedimientos para mostrar sus eventos dependiendo de su tipo de presentador y otro procedimiento para actualizar el estado del evento.
* En el paquete para clientes, incluirá los procedimientos para mostrar los eventos de ese cliente, crear un evento, actualizar el estado de un evento, otro para actualizar la información del evento y uno para borrar algún evento que ya no se llevará a cabo.
* En el paquete de administrador estarán los procedimiento para insertar y modificar género e instrumento junto con mostrar, eliminar a presentador, cliente, instrumento o género.

# Transacciones

## Definición de Transacciones

Las transacciones que se van a utilizar son las siguientes:

* Ingresa nuevo cliente.
* Ingresa nuevo presentador
* Crear evento.
* Selecciona evento.
* Rechaza/Acepta Evento.
* Ingresa nuevo género.
* Ingresa nuevo instrumento.

Acá se muestran con sus respectivos permisos para cada tabla:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Tablas | Ingresa nuevo  cliente | Ingresar nuevo  Presentador | Crea evento | Selecciona  evento | Rechaza/Acepta  evento | Ingresa nuevo  género | Ingresa nuevo  instrumento |
| Cliente | 1. Insert |  | 1. Select |  |  |  |  |
| Género |  | 3. Insert | 3. Select |  |  | 1. Insert |  |
| Instrumento |  | 3. Insert |  |  |  |  | 1. Insert |
| Tipo Presentador |  | 2. Insert | 2. Select |  |  |  |  |
| Presentador |  | 1. Insert |  |  |  |  |  |
| Evento |  |  | 4. Insert | 1. Select,  2. Update | 1. Select, 2. Update |  |  |
| Genero presentador |  | 4. Insert |  |  |  |  |  |

# Identificación de permisos

## Definición de Roles y Esquema de Seguridad

Se crearon 3 tipos de roles para el modelo, cada uno con diferentes grados de permisos para permitirles hacer diferentes acciones. Los roles son:

1. Rol de Administrador: El rol de administrador se refiere al rol de administrador de aplicación. Este tiene permiso para realizar las acciones de SELECT, INSERT, DELETE y UPDATE en todas las tablas del modelo. Es el encargado además de Ingresar Nuevo Instrumento y de Ingresar Nuevo Género.
2. Rol de Cliente: El rol de cliente está dirigido a todo cliente que se dedique a crear nuevos eventos. Este puede Ingresar Nuevo Cliente, Crear Evento y Rechazar Evento.
3. Rol de Presentador: El rol de presentador está definido para cada presentador que se cree bajo los parámetros de este. Este puede Ingresar nuevo Presentador y Seleccionar un evento.

Con el uso de roles se evita que otros usuarios sin los permisos necesarios puedan modificar datos a los cuales no están autorizados. De igual manera con el uso de procedimientos se reduce el riesgo que se le den permisos a otros usuarios que no los tienen.

La idea que el rol de administrador tenga permiso de modificar todas las tablas, da la oportunidad de que este rol tenga los permisos para eliminar a un usuario si hace uso indebido de la aplicación. De igual manera tiene acceso restringido a las áreas de las tablas con información sensible.

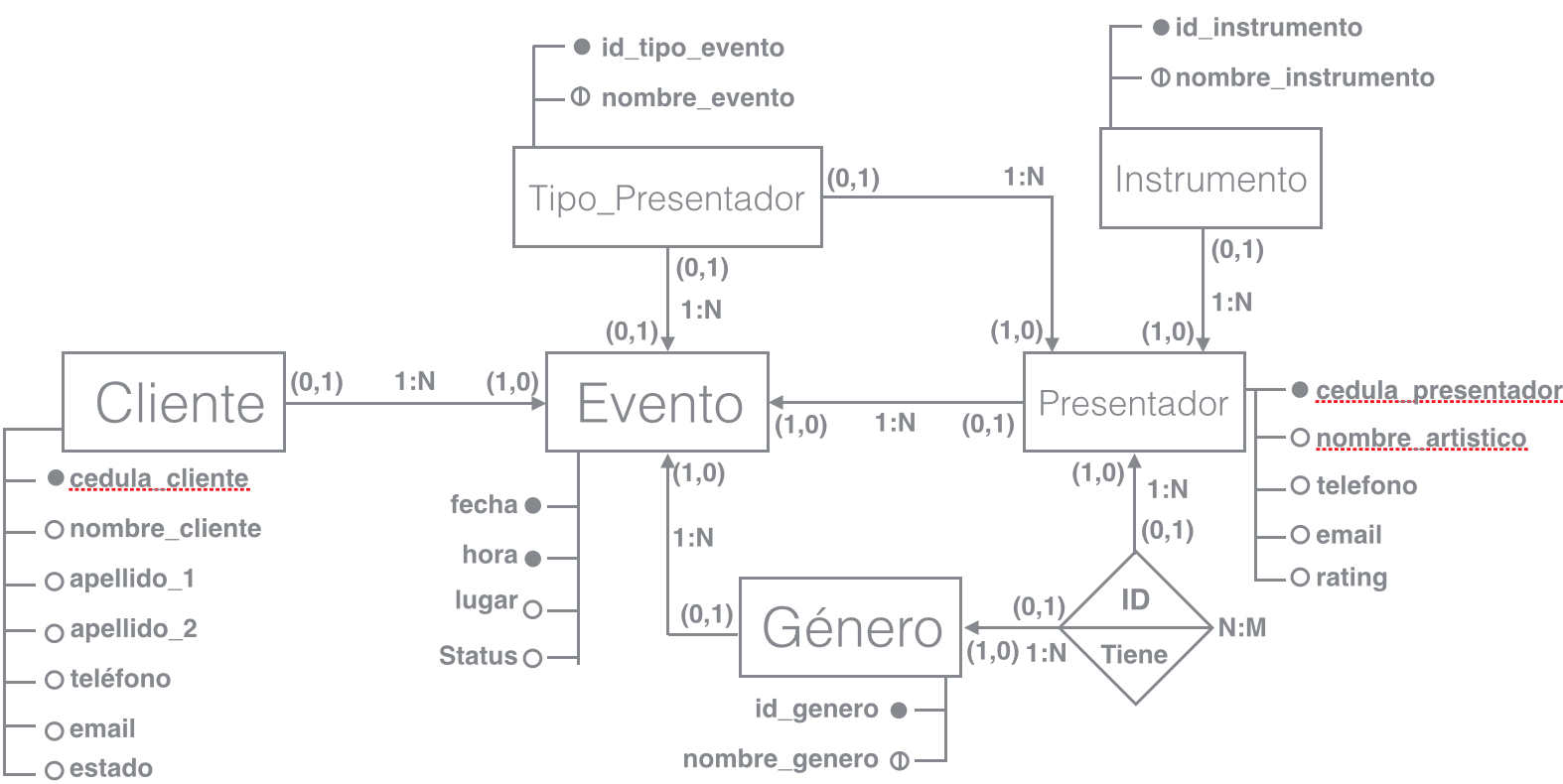
# Concurrencia

## Definición e Implementación de Aspectos de Concurrencia

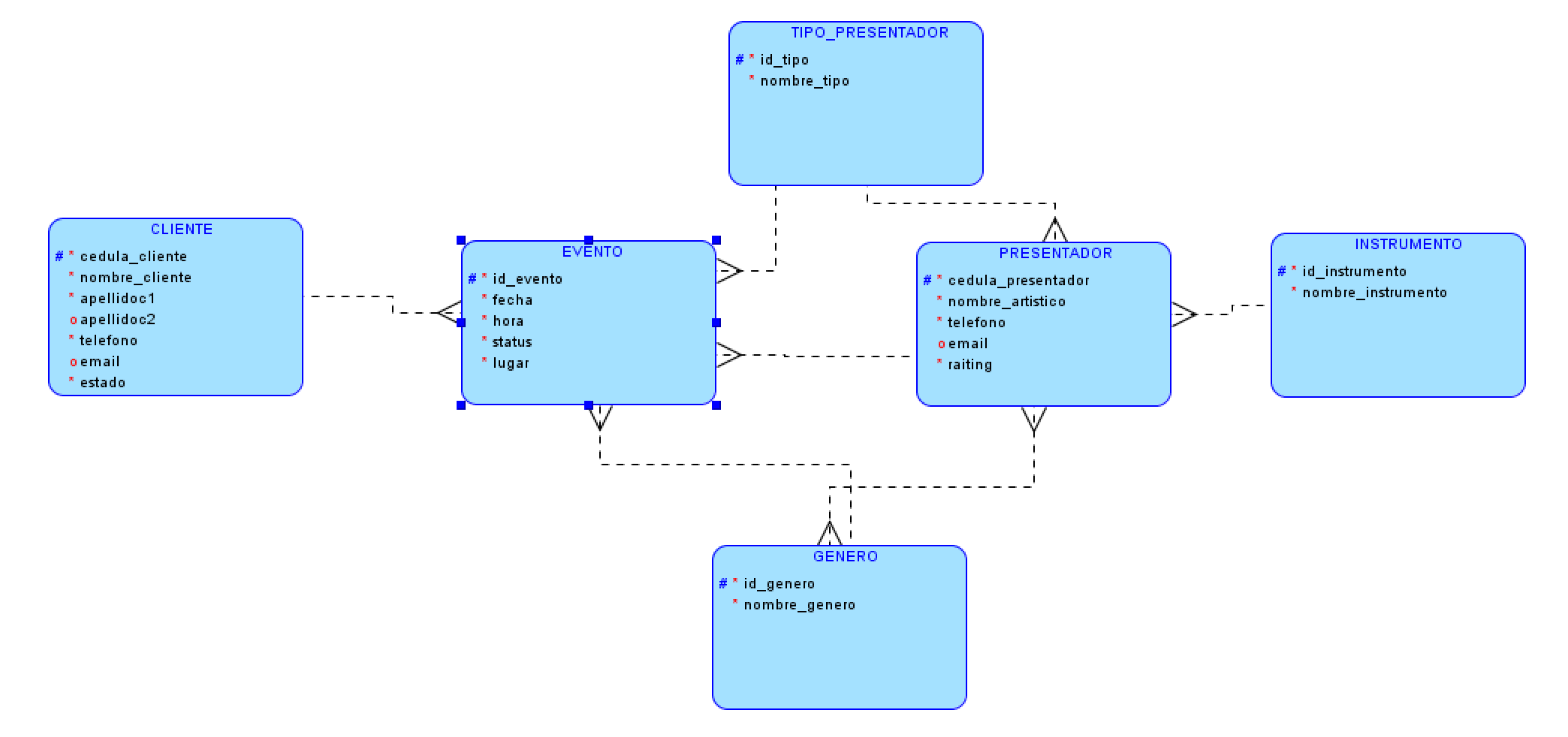
Cuando un músico o una banda seleccione un evento. Este evento quedará bloqueado, pero se seguirá mostrando con la leyenda que está “*En revisión*”. Si otro músico o banda intentara seleccionarlo no va a poder hasta que el cliente acepte o rechace la oferta del músico/banda. Si el evento es aceptado, este se eliminará de los eventos disponibles. Si el evento es rechazado, este volverá a estar disponible para los demás ofertantes.

Capítulo II - Diseño del Modelo

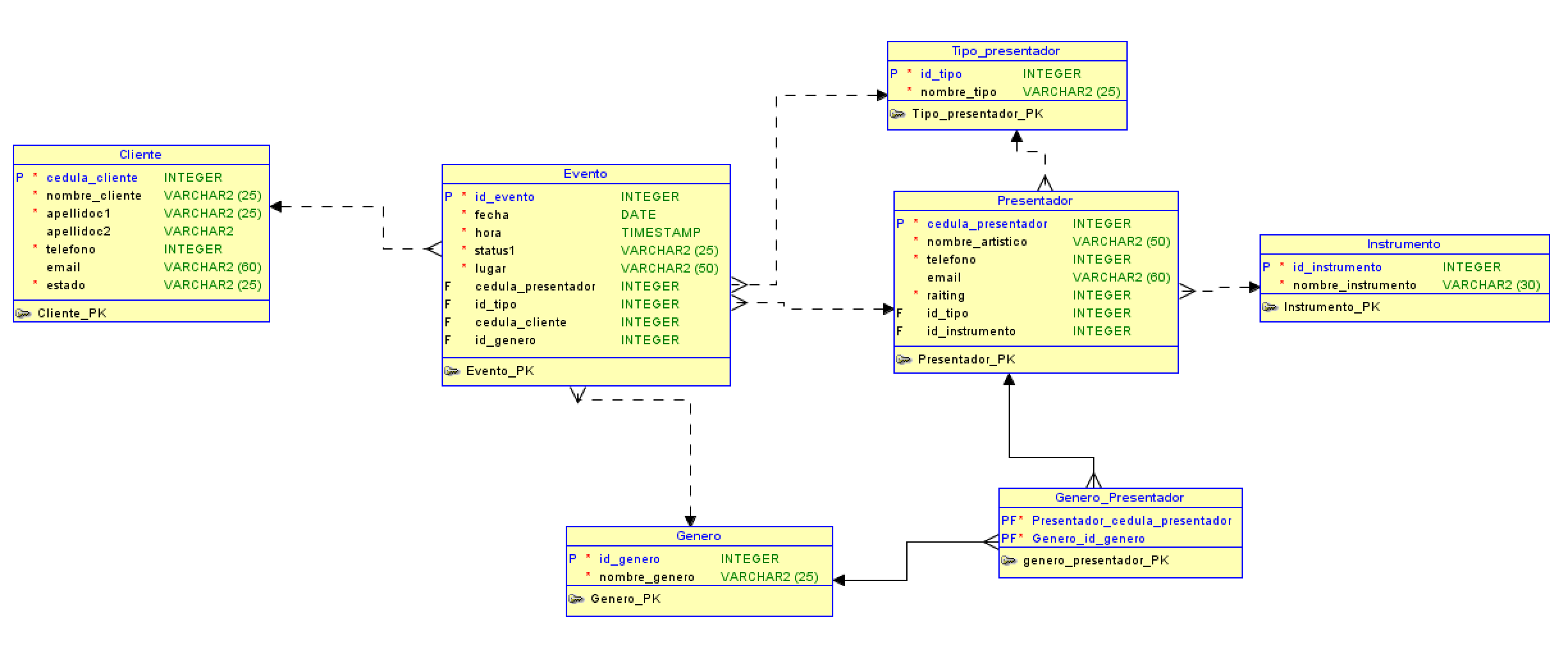
# Modelo Conceptual



# Modelo Lógico



# Modelo relacional



# Modelo Físico

## Exclusión de tablas

drop table CLIENTE cascade constraint;

drop table GENERO cascade constraint;

drop table INSTRUMENTO cascade constraint;

drop table TIPO\_PRESENTADOR cascade constraint;

drop table PRESENTADOR cascade constraint;

drop table GENERO\_PRESENTADOR cascade constraint;

drop table EVENTO cascade constraint;

## Creación de Tablas

CREATE TABLE CLIENTE(

cedula\_cliente NUMBER(9) NOT NULL,

nombre\_cliente VARCHAR2(25) NOT NULL,

apellidoc1 VARCHAR2(25) NOT NULL,

apellidoc2 VARCHAR2(25),

telefono NUMBER(8) NOT NULL,

email VARCHAR2(60),

estado VARCHAR2(25) default 'Activo' NOT NULL,

CONSTRAINT cliente\_PK PRIMARY KEY (cedula\_cliente),

CONSTRAINT ck\_estado CHECK(estado IN ('Activo', 'Inactivo'))

);

CREATE TABLE GENERO(

id\_genero NUMBER(10) NOT NULL,

nombre\_genero VARCHAR2(25) NOT NULL,

CONSTRAINT genero\_PK PRIMARY KEY (id\_genero),

CONSTRAINT uk\_nombre\_genero UNIQUE (nombre\_genero)

);

CREATE TABLE INSTRUMENTO(

id\_instrumento NUMBER(10) NOT NULL,

nombre\_instrumento VARCHAR2(25) NOT NULL,

CONSTRAINT instrumento\_PK PRIMARY KEY (id\_instrumento),

CHECK(nombre\_instrumento = initcap(nombre\_instrumento))

);

CREATE TABLE TIPO\_PRESENTADOR(

id\_tipo NUMBER(10) NOT NULL,

nombre\_tipo VARCHAR2(25) NOT NULL,

CONSTRAINT tipo\_presentador\_PK PRIMARY KEY (id\_tipo),

CONSTRAINT ck\_nombre\_tipo CHECK(nombre\_tipo IN ('Musico', 'Banda'))

);

CREATE TABLE PRESENTADOR(

cedula\_presentador NUMBER(9) NOT NULL,

nombre\_artistico VARCHAR2(60) NOT NULL,

telefono NUMBER(8) NOT NULL,

email VARCHAR2(60),

rating NUMBER(2) DEFAULT 0 NOT NULL,

id\_tipo NUMBER(10),

id\_instrumento NUMBER(10),

CONSTRAINT presentador\_PK PRIMARY KEY (cedula\_presentador),

CONSTRAINT fk\_id\_tipo\_presentador FOREIGN KEY (id\_tipo) REFERENCES TIPO\_PRESENTADOR(id\_tipo) ON DELETE SET NULL,

CONSTRAINT fk\_id\_instrumento FOREIGN KEY (id\_instrumento) REFERENCES INSTRUMENTO(id\_instrumento) ON DELETE SET NULL

);

CREATE TABLE GENERO\_PRESENTADOR(

presentador\_cedula\_presentador NUMBER(9) NOT NULL,

genero\_id\_genero NUMBER(10) NOT NULL,

CONSTRAINT genero\_presentador\_PK PRIMARY KEY (presentador\_cedula\_presentador, genero\_id\_genero),

CONSTRAINT fk\_gen\_pres\_cedula\_presentador FOREIGN KEY (presentador\_cedula\_presentador) REFERENCES PRESENTADOR(cedula\_presentador) ON DELETE CASCADE,

CONSTRAINT fk\_gen\_pres\_id\_genero FOREIGN KEY (genero\_id\_genero) REFERENCES GENERO(id\_genero) ON DELETE CASCADE

);

CREATE TABLE EVENTO(

id\_evento NUMBER(10) NOT NULL,

fecha DATE NOT NULL,

hora TIMESTAMP NOT NULL,

lugar VARCHAR2(50) NOT NULL,

status VARCHAR2(25) DEFAULT 'Abierto' NOT NULL,

cedula\_presentador NUMBER(9),

id\_tipo NUMBER(10),

cedula\_cliente NUMBER(9),

id\_genero NUMBER(10),

CONSTRAINT evento\_PK PRIMARY KEY (id\_evento),

CONSTRAINT fk\_cedula\_presentador\_evento FOREIGN KEY (cedula\_presentador) REFERENCES PRESENTADOR(cedula\_presentador) ON DELETE SET NULL,

CONSTRAINT fk\_id\_tipo\_evento FOREIGN KEY (id\_tipo) REFERENCES TIPO\_PRESENTADOR(id\_tipo) ON DELETE SET NULL,

CONSTRAINT fk\_cedula\_cliente\_evento FOREIGN KEY (cedula\_cliente) REFERENCES CLIENTE(cedula\_cliente) ON DELETE SET NULL,

CONSTRAINT fk\_id\_genero\_evento FOREIGN KEY (id\_genero) REFERENCES GENERO(id\_genero) ON DELETE SET NULL,

CONSTRAINT ck\_status CHECK(status IN ('Abierto', 'Pendiente', 'Cerrado'))

);

## Inserción de Datos

--CLIENTE

INSERT ALL

--cedula\_cliente, nombre\_cliente, apellidoc1, apellidoc2, telefono, email, estado

INTO CLIENTE VALUES (578654433, 'Michael', 'Robinson', NULL, 88553300, 'mrobinson@ejemplo.com', 'Inactivo')

INTO CLIENTE VALUES (112345432, 'Jose', 'Perez', 'Rodriguez', 88776655, 'jperez@ejemplo.com', 'Activo')

INTO CLIENTE VALUES (111480744, 'Raquel', 'Torres', 'Flores', 88990077, 'jflores@ejemplo.com', 'Activo')

SELECT \* FROM dual;

--GENERO

INSERT ALL

--id\_genero, nombre\_genero

INTO GENERO VALUES (102, 'Clasica')

INTO GENERO VALUES (101, 'Trova')

INTO GENERO VALUES (100, 'Rock')

SELECT \* FROM dual;

--INSTRUMENTO

INSERT ALL

--id\_instrumento, nombre\_instrumento

INTO INSTRUMENTO VALUES (1003, 'Saxofon')

INTO INSTRUMENTO VALUES (1001, 'Piano')

INTO INSTRUMENTO VALUES (1000, 'Guitarra')

SELECT \* FROM dual;

--TIPO\_PRESENTADOR

INSERT ALL

--id\_tipo, nombre\_tipo

INTO TIPO\_PRESENTADOR VALUES (02, 'Banda')

INTO TIPO\_PRESENTADOR VALUES (01, 'Musico')

SELECT \* FROM dual;

--PRESENTADOR

INSERT ALL

--cedula\_presentador, nombre\_artistico, telefono, email, raiting, id\_tipo, id\_instrumento

INTO PRESENTADOR VALUES (982341670, 'Marta y los del barrio', 63120543, 'losdelbarrio@ejemplo.com', 10, 02, NULL)

INTO PRESENTADOR VALUES (543217390, 'Josue Padilla', 73890044, 'ppadilla@ejemplo.com', 0, 01, 1000)

INTO PRESENTADOR VALUES (134980909, 'La Pandilla', 88773311, 'lapandilla@ejemplo.com', 8, 01, NULL)

SELECT \* FROM dual;

--EVENTO

INSERT ALL

--id\_evento, fecha, hora, lugar, status, cedula\_presentador, id\_tipo, cedula\_cliente, id\_genero

INTO EVENTO VALUES (10000, '01-10-2016', TO\_TIMESTAMP('2016/10/01 8:30:00', 'YYYY/MM/DD HH24:MI:SS'),

'Tibas', 'Abierto', 982341670, 01, 111480744, 100)

INTO EVENTO VALUES (10001, '01-08-2016', TO\_TIMESTAMP('2016/08/01 16:30:00', 'YYYY/MM/DD HH24:MI:SS'),

'Alajuela', 'Pendiente', 134980909, 02, 112345432, 101)

INTO EVENTO VALUES (10002, '01-10-2016', TO\_TIMESTAMP('2016/10/01 20:30:00', 'YYYY/MM/DD HH24:MI:SS'),

'Escazu', 'Cerrado', 543217390, 01, 578654433, 102)

SELECT \* FROM dual;

--GENERO\_PRESENTADOR

INSERT ALL

--presentador\_cedula\_presentador, genero\_id\_genero

INTO GENERO\_PRESENTADOR VALUES (982341670, 100)

INTO GENERO\_PRESENTADOR VALUES (543217390, 102)

INTO GENERO\_PRESENTADOR VALUES (134980909, 101)

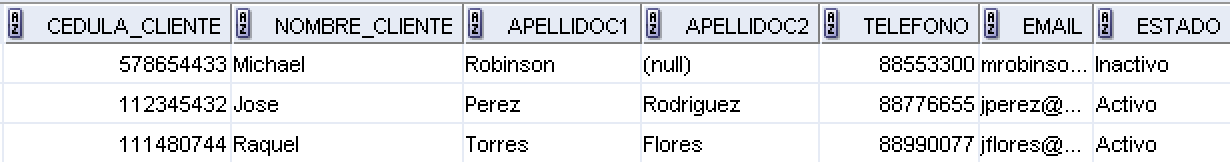
INTO GENERO\_PRESENTADOR VALUES (134980909, 102)

INTO GENERO\_PRESENTADOR VALUES (134980909, 100)

SELECT \* FROM dual;

## Definición de Datos

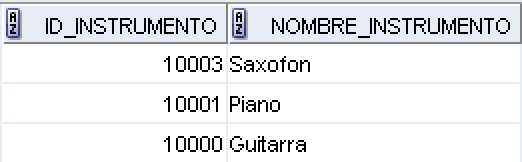
Cliente



Género



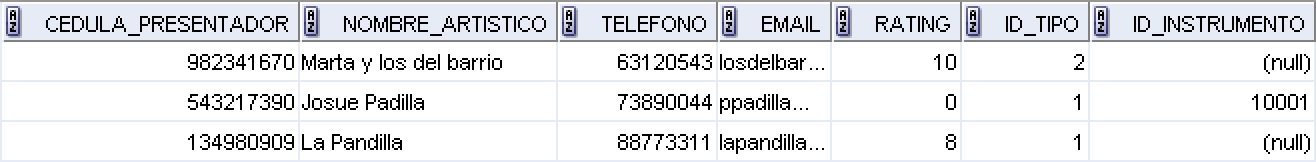
Instrumento



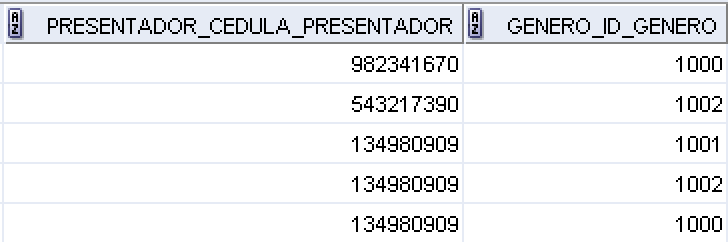
Tipo Presentador



Presentador



Género Presentador



Evento



## Triggers

create OR replace trigger tr\_update\_evento

before UPDATE

ON EVENTO

for each row

DECLARE

v\_status VARCHAR2;

BEGIN

v\_status := :old.status;

if v\_status != 'Abierto' THEN

raise\_application\_error( -20001, 'No puede aplicar para este evento');

end if;

END;

/

show errors

create OR replace trigger tr\_insert\_instrumento

before INSERT

ON PRESENTADOR

for each row

DECLARE

v\_id\_tipo number;

v\_id\_instrumento number;

BEGIN

v\_id\_tipo := :new.id\_tipo;

v\_id\_instrumento := :new.id\_instrumento;

if v\_id\_tipo = 02 AND v\_id\_instrumento IS NOT NULL THEN

raise\_application\_error( -20001, 'Bandas no pueden registrar instrumentos');

end if;

END;

/

show errors

## Secuencia de datos

DROP SEQUENCE sq\_id\_instrumento;

CREATE SEQUENCE sq\_id\_instrumento START WITH 1000 INCREMENT BY 1;

SELECT sq\_id\_instrumento.nextval FROM DUAL;

DROP SEQUENCE sq\_id\_genero;

CREATE SEQUENCE sq\_id\_genero START WITH 100 INCREMENT BY 1;

SELECT sq\_id\_genero.nextval FROM DUAL;

DROP SEQUENCE sq\_id\_evento;

CREATE SEQUENCE sq\_id\_evento START WITH 10000 INCREMENT BY 1;

SELECT sq\_id\_evento.nextval FROM DUAL;

## Paquetes y Procedimientos de Datos

### Paquete Administrador

CREATE OR REPLACE PACKAGE pck\_admin IS

PROCEDURE consInstrumento(

p\_id\_instrumento IN OUT INSTRUMENTO.ID\_INSTRUMENTO%TYPE,

p\_nombre\_instrumento OUT INSTRUMENTO.NOMBRE\_INSTRUMENTO%TYPE);

PROCEDURE insInstrumento(

p\_nombre\_instrumento IN INSTRUMENTO.NOMBRE\_INSTRUMENTO%TYPE);

PROCEDURE delInstrumento(

p\_id\_instrumento IN INSTRUMENTO.ID\_INSTRUMENTO%TYPE);

PROCEDURE updInstrumento(

p\_id\_instrumento IN INSTRUMENTO.ID\_INSTRUMENTO%TYPE,

p\_nombre\_instrumento IN INSTRUMENTO.NOMBRE\_INSTRUMENTO%TYPE);

PROCEDURE consGenero(

p\_id\_genero IN OUT GENERO.ID\_GENERO%TYPE,

p\_nombre\_genero OUT GENERO.NOMBRE\_GENERO%TYPE);

PROCEDURE insGenero(

p\_nombre\_genero IN GENERO.NOMBRE\_GENERO%TYPE);

PROCEDURE delGenero(

p\_id\_genero IN GENERO.ID\_GENERO%TYPE);

PROCEDURE updGenero(

p\_id\_genero IN GENERO.ID\_GENERO%TYPE,

p\_nombre\_genero IN GENERO.NOMBRE\_GENERO%TYPE);

PROCEDURE consPresentador(

p\_cedula\_presentador IN OUT PRESENTADOR.CEDULA\_PRESENTADOR%TYPE,

p\_nombre\_artistico OUT PRESENTADOR.NOMBRE\_ARTISTICO%TYPE,

p\_telefono OUT PRESENTADOR.TELEFONO%TYPE,

p\_email OUT PRESENTADOR.EMAIL%TYPE,

p\_rating OUT PRESENTADOR.RATING%TYPE,

p\_id\_tipo OUT PRESENTADOR.ID\_TIPO%TYPE,

p\_id\_instrumento OUT PRESENTADOR.ID\_INSTRUMENTO%TYPE);

PROCEDURE delPresentador(

p\_cedula\_presentador IN PRESENTADOR.CEDULA\_PRESENTADOR%TYPE);

PROCEDURE consCliente(

p\_cedula\_cliente IN OUT CLIENTE.CEDULA\_CLIENTE%TYPE,

p\_nombre\_cliente OUT CLIENTE.NOMBRE\_CLIENTE%TYPE,

p\_apellidoc1 OUT CLIENTE.APELLIDOC1%TYPE,

p\_apellidoc2 OUT CLIENTE.APELLIDOC2%TYPE,

p\_telefono OUT CLIENTE.TELEFONO%TYPE,

p\_email OUT CLIENTE.EMAIL%TYPE,

p\_estado OUT CLIENTE.ESTADO%TYPE);

PROCEDURE delCliente(

p\_cedula\_cliente IN CLIENTE.CEDULA\_CLIENTE%TYPE);

END pck\_admin;

/

CREATE OR REPLACE PACKAGE BODY pck\_admin AS

-----------INSTRUMENTO------------

--select

PROCEDURE consInstrumento(

p\_id\_instrumento IN OUT INSTRUMENTO.ID\_INSTRUMENTO%TYPE,

p\_nombre\_instrumento OUT INSTRUMENTO.NOMBRE\_INSTRUMENTO%TYPE) IS

BEGIN

SELECT \*

INTO p\_id\_instrumento, p\_nombre\_instrumento

FROM INSTRUMENTO WHERE id\_instrumento = p\_id\_instrumento;

END consInstrumento;

--insert

PROCEDURE insInstrumento(

p\_nombre\_instrumento IN INSTRUMENTO.NOMBRE\_INSTRUMENTO%TYPE) IS

BEGIN

INSERT INTO INSTRUMENTO

VALUES(sq\_id\_instrumento.nextval, p\_nombre\_instrumento);

END insInstrumento;

--delete

PROCEDURE delInstrumento(

p\_id\_instrumento IN INSTRUMENTO.ID\_INSTRUMENTO%TYPE) IS

BEGIN

DELETE INSTRUMENTO i

WHERE i.id\_instrumento = p\_id\_instrumento;

END delInstrumento;

--update

PROCEDURE updInstrumento(

p\_id\_instrumento IN INSTRUMENTO.ID\_INSTRUMENTO%TYPE,

p\_nombre\_instrumento IN INSTRUMENTO.NOMBRE\_INSTRUMENTO%TYPE) IS

BEGIN

UPDATE INSTRUMENTO i

SET i.nombre\_instrumento = p\_nombre\_instrumento

WHERE i.id\_instrumento = p\_id\_instrumento;

END updInstrumento;

-----------GENERO------------

--select

PROCEDURE consGenero(

p\_id\_genero IN OUT GENERO.ID\_GENERO%TYPE,

p\_nombre\_genero OUT GENERO.NOMBRE\_GENERO%TYPE) IS

BEGIN

SELECT \*

INTO p\_id\_genero, p\_nombre\_genero

FROM GENERO WHERE id\_genero = p\_id\_genero;

END consGenero;

--insert

PROCEDURE insGenero(

p\_nombre\_genero IN GENERO.NOMBRE\_GENERO%TYPE) IS

BEGIN

INSERT INTO INSTRUMENTO

VALUES(sq\_id\_genero.nextval, p\_nombre\_genero);

END insGenero;

--delete

PROCEDURE delGenero(

p\_id\_genero IN GENERO.ID\_GENERO%TYPE) IS

BEGIN

DELETE GENERO g

WHERE g.id\_genero = p\_id\_genero;

END delGenero;

--update

PROCEDURE updGenero(

p\_id\_genero IN GENERO.ID\_GENERO%TYPE,

p\_nombre\_genero IN GENERO.NOMBRE\_GENERO%TYPE) IS

BEGIN

UPDATE GENERO g

SET g.nombre\_genero = p\_nombre\_genero

WHERE g.id\_genero = p\_id\_genero;

END updGenero;

-----------PRESENTADOR------------

--select

PROCEDURE consPresentador(

p\_cedula\_presentador IN OUT PRESENTADOR.CEDULA\_PRESENTADOR%TYPE,

p\_nombre\_artistico OUT PRESENTADOR.NOMBRE\_ARTISTICO%TYPE,

p\_telefono OUT PRESENTADOR.TELEFONO%TYPE,

p\_email OUT PRESENTADOR.EMAIL%TYPE,

p\_rating OUT PRESENTADOR.RATING%TYPE,

p\_id\_tipo OUT PRESENTADOR.ID\_TIPO%TYPE,

p\_id\_instrumento OUT PRESENTADOR.ID\_INSTRUMENTO%TYPE) IS

BEGIN

SELECT \*

INTO p\_cedula\_presentador, p\_nombre\_artistico, p\_telefono,

p\_email, p\_rating, p\_id\_tipo, p\_id\_instrumento

FROM PRESENTADOR WHERE cedula\_presentador = p\_cedula\_presentador;

END consPresentador;

--delete

PROCEDURE delPresentador(

p\_cedula\_presentador IN PRESENTADOR.CEDULA\_PRESENTADOR%TYPE) IS

BEGIN

DELETE PRESENTADOR p

WHERE p.cedula\_presentador = p\_cedula\_presentador;

END delPresentador;

-----------CLIENTE------------

--select

PROCEDURE consCliente(

p\_cedula\_cliente IN OUT CLIENTE.CEDULA\_CLIENTE%TYPE,

p\_nombre\_cliente OUT CLIENTE.NOMBRE\_CLIENTE%TYPE,

p\_apellidoc1 OUT CLIENTE.APELLIDOC1%TYPE,

p\_apellidoc2 OUT CLIENTE.APELLIDOC2%TYPE,

p\_telefono OUT CLIENTE.TELEFONO%TYPE,

p\_email OUT CLIENTE.EMAIL%TYPE,

p\_estado OUT CLIENTE.ESTADO%TYPE) IS

BEGIN

SELECT \*

INTO p\_cedula\_cliente, p\_nombre\_cliente, p\_apellidoc1, p\_apellidoc2,

p\_telefono, p\_email, p\_estado

FROM CLIENTE WHERE cedula\_cliente = p\_cedula\_cliente;

END consCliente;

--delete

PROCEDURE delCliente(

p\_cedula\_cliente IN CLIENTE.CEDULA\_CLIENTE%TYPE) IS

BEGIN

DELETE CLIENTE c

WHERE c.cedula\_cliente = p\_cedula\_cliente;

END delCliente;

END pck\_admin;

/

show errors

### Paquete Cliente

CREATE OR REPLACE PACKAGE pck\_cliente IS

PROCEDURE consEventoCliente(

p\_cedula\_cliente IN OUT EVENTO.CEDULA\_CLIENTE%TYPE,

p\_fecha OUT NOCOPY EVENTO.FECHA%TYPE,

p\_hora OUT NOCOPY EVENTO.HORA%TYPE,

p\_lugar OUT NOCOPY EVENTO.LUGAR%TYPE,

p\_status OUT NOCOPY EVENTO.STATUS%TYPE);

PROCEDURE creaEventoCliente(

p\_id\_evento IN EVENTO.ID\_EVENTO%TYPE,

p\_fecha IN EVENTO.FECHA%TYPE,

p\_hora IN EVENTO.HORA%TYPE,

p\_lugar IN EVENTO.LUGAR%TYPE,

p\_id\_tipo IN EVENTO.ID\_TIPO%TYPE,

p\_cedula\_cliente IN EVENTO.CEDULA\_CLIENTE%TYPE,

p\_id\_genero IN EVENTO.ID\_GENERO%TYPE);

PROCEDURE updEventoStatusCliente(

p\_id\_evento IN EVENTO.ID\_EVENTO%TYPE,

p\_status IN EVENTO.STATUS%TYPE);

PROCEDURE updEventoCliente(

p\_id\_evento IN EVENTO.ID\_EVENTO%TYPE,

p\_fecha IN EVENTO.FECHA%TYPE,

p\_hora IN EVENTO.HORA%TYPE,

p\_lugar IN EVENTO.LUGAR%TYPE,

p\_id\_tipo IN EVENTO.ID\_TIPO%TYPE,

p\_id\_genero IN EVENTO.ID\_GENERO%TYPE);

PROCEDURE delEventoCliente(

p\_id\_evento IN EVENTO.ID\_EVENTO%TYPE);

END pck\_cliente;

/

CREATE OR REPLACE PACKAGE BODY pck\_cliente AS

--select

PROCEDURE consEventoCliente(

p\_cedula\_cliente IN OUT EVENTO.CEDULA\_CLIENTE%TYPE,

p\_fecha OUT NOCOPY EVENTO.FECHA%TYPE,

p\_hora OUT NOCOPY EVENTO.HORA%TYPE,

p\_lugar OUT NOCOPY EVENTO.LUGAR%TYPE,

p\_status OUT NOCOPY EVENTO.STATUS%TYPE) IS

BEGIN

SELECT fecha, hora, lugar, status

INTO p\_fecha, p\_hora, p\_lugar, p\_status

FROM EVENTO

WHERE cedula\_cliente = p\_cedula\_cliente;

END consEventoCliente;

--insert

PROCEDURE creaEventoCliente(

p\_id\_evento IN EVENTO.ID\_EVENTO%TYPE,

p\_fecha IN EVENTO.FECHA%TYPE,

p\_hora IN EVENTO.HORA%TYPE,

p\_lugar IN EVENTO.LUGAR%TYPE,

p\_id\_tipo IN EVENTO.ID\_TIPO%TYPE,

p\_cedula\_cliente IN EVENTO.CEDULA\_CLIENTE%TYPE,

p\_id\_genero IN EVENTO.ID\_GENERO%TYPE) IS

BEGIN

INSERT INTO EVENTO

VALUES(sq\_id\_evento.nextval, p\_fecha, p\_hora, p\_lugar, 'Abierto',

NULL, p\_id\_tipo, p\_cedula\_cliente, p\_id\_genero);

END creaEventoCliente;

--update status

PROCEDURE updEventoStatusCliente(

p\_id\_evento IN EVENTO.ID\_EVENTO%TYPE,

p\_status IN EVENTO.STATUS%TYPE) IS

stat varchar2(10);

BEGIN

SELECT STATUS

INTO stat

FROM EVENTO;

if stat = 'Pendiente' then

UPDATE EVENTO e

SET e.STATUS = p\_status

WHERE e.ID\_EVENTO = p\_id\_evento;

end if;

END updEventoStatusCliente;

--update evento

PROCEDURE updEventoCliente(

p\_id\_evento IN EVENTO.ID\_EVENTO%TYPE,

p\_fecha IN EVENTO.FECHA%TYPE,

p\_hora IN EVENTO.HORA%TYPE,

p\_lugar IN EVENTO.LUGAR%TYPE,

p\_id\_tipo IN EVENTO.ID\_TIPO%TYPE,

p\_id\_genero IN EVENTO.ID\_GENERO%TYPE) IS

BEGIN

UPDATE EVENTO e

SET e.FECHA = p\_fecha, e.HORA = p\_hora, e.LUGAR = p\_lugar,

e.ID\_TIPO = p\_id\_tipo, e.ID\_GENERO = p\_id\_genero

WHERE e.ID\_EVENTO = p\_id\_evento;

END updEventoCliente;

--delete evento

PROCEDURE delEventoCliente(

p\_id\_evento IN EVENTO.ID\_EVENTO%TYPE) IS

BEGIN

DELETE EVENTO e

WHERE e.ID\_EVENTO = p\_id\_evento;

END delEventoCliente;

END pck\_cliente;

/

show errors

### Paquete Presentador

CREATE OR REPLACE PACKAGE pck\_presentador IS

PROCEDURE getEventoTipoPresentador(

p\_id\_tipo IN EVENTO.ID\_TIPO%TYPE,

p\_fecha OUT NOCOPY EVENTO.FECHA%TYPE,

p\_hora OUT NOCOPY EVENTO.HORA%TYPE,

p\_lugar OUT NOCOPY EVENTO.LUGAR%TYPE);

PROCEDURE updStatusEventoPresentador(

p\_id\_evento IN EVENTO.ID\_EVENTO%TYPE);

END pck\_presentador;

/

create or replace PACKAGE BODY pck\_presentador AS

--select

PROCEDURE getEventoTipoPresentador(

p\_id\_tipo IN EVENTO.ID\_TIPO%TYPE,

p\_fecha OUT EVENTO.FECHA%TYPE,

p\_hora OUT EVENTO.HORA%TYPE,

p\_lugar OUT EVENTO.LUGAR%TYPE) IS

BEGIN

for c\_evento IN (

SELECT fecha, hora, lugar

FROM EVENTO WHERE id\_tipo = p\_id\_tipo

)

loop

p\_fecha := c\_evento.fecha;

p\_hora := c\_evento.hora;

p\_lugar := c\_evento.lugar;

end loop;

END getEventoTipoPresentador;

--update status

PROCEDURE updStatusEventoPresentador(

p\_id\_evento IN EVENTO.ID\_EVENTO%TYPE) IS

stat varchar2(10);

BEGIN

SELECT status

INTO stat

FROM EVENTO

FOR UPDATE of status nowait;

if stat = 'Abierto' then

UPDATE EVENTO e

SET e.STATUS = 'Pendiente'

WHERE e.ID\_EVENTO = p\_id\_evento;

end if;

END updStatusEventoPresentador;

END pck\_presentador;

/

show errors;

# Seguridad

## Roles

create role CLIENTE;

create role PRESENTADOR;

create role ADMIN;

GRANT EXECUTE on PCK\_CLIENTE to CLIENTE;

GRANT EXECUTE on PCK\_PRESENTADOR to PRESENTADOR;

GRANT EXECUTE on PCK\_ADMIN to ADMIN;

CREATE user musico IDENTIFIED BY musico;

GRANT CREATE SESSION, PRESENTADOR to musico;

CREATE user banda IDENTIFIED BY banda;

GRANT CREATE SESSION, PRESENTADOR to banda;

CREATE user cliente1 IDENTIFIED BY cliente1;

GRANT CREATE SESSION, CLIENTE to cliente1;

CREATE user admin1 IDENTIFIED BY admin1;

GRANT create session, ADMIN to admin1;

CREATE OR REPLACE PUBLIC SYNONYM PCK\_CLIENTE FOR proyecto.PCK\_CLIENTE;

CREATE OR REPLACE PUBLIC SYNONYM PCK\_PRESENTADOR FOR proyecto.PCK\_PRESENTADOR;

CREATE OR REPLACE PUBLIC SYNONYM PCK\_ADMIN FOR proyecto.PCK\_ADMIN;

COMMIT;

Conclusiones y Recomendaciones

Después de haber finalizado el presente trabajo sobre la creación de una base de datos se puede ver el trabajo y conocimiento que se necesita para llevarlo a cabo. Con respecto a los requerimientos, puedo decir que cumple con las características establecidas para el prototipo.

A la hora del diseño y la creación de una base de datos, es importante ver cuál es la mejor manera para organizar la información en los disparadores, paquetes y sus respectivos procedimientos. Esto que viene a representar todas las relaciones entre datos y modelos de la base de datos.

Anotaciones del profesor