Proyecto final de la asignatura de bases de datos

Angeles Estrada Ricardo
Muro León Yoaddan Yokaem
Ramirez Gonzalez Jose Miguel
Vazquez Zavala Oliver Alexis

28 de mayo de 2022

1. Introducción

Para este proyecto trataremos de resolver el siguiente problema, diseñando la base de datos desde cero, analizando los datos y creando los modelos correspondientes, así como las transformaciones pertinentes para obtener una base de datos funcional. Se desarrollará en dos fases la primera consiste en desarrollar y crear una base de datos funcional y la segunda parte consiste en crear una interfaz gráfica vía app móvil o web, que permita manipular y obtener datos de nuestra base que creamos, entonces el problema es el siguiente:

Un restaurante desea digitalizar su forma de operación, para ello se desarrollará un sistema informático que constará de varios módulos. El que corresponde a la implementación de la base de datos deberá atender el siguiente requerimiento: Se debe almacenar el RFC, número de empleado, nombre, fecha de nacimiento, teléfonos, edad, domicilio, sueldo; de los cocineros su especialidad, de los meseros su horario y de los administrativos su rol, así como una foto de los empleados y considerar que un empleado puede tener varios puestos. Es necesario tener registro de los dependientes de los empleados, su CURP, nombre y parentesco. Se debe tener disponible la información de los platillos y bebidas que el restaurante ofrece, una descripción, nombre, su receta, precio y un indicador de disponibilidad, así como el nombre y descripción de la categoría a la que pertenecen (considerar que un platillo o bebida solo pertenece a una categoría). Debe tenerse registro del folio de la orden, fecha, la cantidad total a pagar por la orden y registro del mesero que levanto la orden, así como la cantidad de cada platillo/bebida y precio total a pagar por platillo/bebida contenidos en cada orden. Considerar que es posible que los clientes soliciten factura de su consumo, por lo que debe almacenarse su RFC, nombre, domicilio, razón social, email y fecha de nacimiento.

Bien ahora realizando el análisis del problema se determinó lo siguiente:

Las entidades con sus atributos y subtipos, y estos con sus atributos específicos, son:

- Empleado: RFC, número de empleado, nombre, fecha de nacimiento, teléfonos, edad, domicilio, sueldo, foto del empleado
    - Cocineros: Especialidad.

    - Meseros: Horario.
    - Administrativos: Rol.
- Dependiente: CURP, nombre y parentesco.
- Platillos y Bebidas: descripción, nombre, su receta, precio y un indicador de disponibilidad.
- Categoría: nombre, descripción.
- Orden: Folio, fecha, cantidad total a pagar, registro del mesero que levanto la orden, cantidad por platillo/bebida, y la cantidad a pagar por platillo/bebida.
- Cliente: RFC, nombre, domicilio, razón social, email y fecha de nacimiento.

Esas serian las entidades y atributos que encontramos y que se tomaran en cuenta al desarrollar la base de datos, además se identificaron restricciones que se den de cumplir que son: un empleado puede tener varios puestos y que un platillo o bebida solo pertenece a una categoría.

Una vez que ya tenemos este análisis pasaremos a crear nuestros modelos, Modelo Entidad Relación (MER) y el Modelo Relacional (MR), y después pasar a crear tablas y después pasar resolver cada unos de nuestros objetivos que debe de poder realizar nuestra base de datos
Entonces como objetivos tenemos:

- Cada que se agregue un producto a la orden, debe actualizarse los totales (por producto y venta), así como validar que el producto esté disponible
- Crear al menos, un índice, del tipo que se prefiera y donde se prefiera. Justificar el porqué de la elección en ambos aspectos.
- Dado un número de empleado, mostrar la cantidad de ordenes que ha registrado en el día, así como el total que se ha pagado por dichas órdenes. Si no se trata de un mesero, mostrar un mensaje de error.
- Vista que muestre todos los detalles del platillo más vendido.
- Permitir obtener el nombre de aquellos productos que no estén disponibles.
- De manera automática se genere una vista que contenga información necesaria para asemejarse a una factura de una orden.
- Dada una fecha, o una fecha de inicio y fecha de fin, regresar el total del número de ventas y el monto total por las ventas en ese periodo de tiempo. Nota: Con producto se hace referencia a los alimentos y bebidas.

Y ya al final pasara la creación de la pagina web con la que podemos visualizar y modificar los datos de la base de datos

2. Plan de trabajo

Para esto se piensa realizar el modelo entidad relación, después pasar a la representación intermedia entre el MER y el MR, donde tenemos que considerar las relaciones que va haber entre entidades, para hacer crear las llaves primarias y foráneas, además de que aquí se analiza si algún atributo requiere que su tabla sea separada, esto puede darse por diferentes motivos, pero principalmente por ser multivaluado o por contar con atributos que están compuestos por otros atributos, etc.

Después se haría la creación de tablas en PgSQL, en donde se hará cada una de las tablas que consideramos con sus atributos, a continuación, se verían la inclusión de elementos a cada tabla, se consideró que una de las primeras entidades que nos convenia llenar primero iba a hacer el de la categoría, esto debido a que si primero consideramos loas categorías, después podríamos crear los productos que podrían estar en ellos, ya después se pueden llenar a los empleados , luego a los subtipos y sus atributos, así como la relación entre las tablas de empleados y dependientes, una vez que vemos que las tablas ya están con sus atributos y sus relaciones y que todo ya está correcto, se pasaría a realizar las producciones o funciones que se nos solicitan como objetivos, y una vez que estos ya estén o que veamos que no presentan fallas al momentos realizar alguna modificación, eliminación o actualización de algún elemento de cualquier tabla, debemos asegurarnos de que si un cambio se realiza en alguna tabla, este cambio se vea reflejado en las demás tablas que estén relacionas con la tabla del cambio. Ya una vez visto todo esto, se deberá realizar que la app o página web donde crearemos una interfaz la cual nos permita ver nuestra base de datos, así como que pueda ser modificada por esta página.

Ahora para la organización del proyecto las tareas se repartieron de la siguiente forma:

- Angeles Estrada Ricardo: Creación del documento escrito
- Muro Leon Yoaddan Yokaem: Creación de funciones, triggers y vistas
- Ramírez González José Miguel: Creación de la aplicación web y conexión a la base de datos
- Vázquez Zavala Oliver Alexis: Una parte de MER, Representación intermedia, Modelo Relacional, creación de tablas e inserción de información

Pero entre todos nos apoyamos en la realización de las tareas.

3. Diseño

Tomando en cuenta las entidades y atributos que se establecieron en la introducción vamos a realizar el MER, para ello primeramente vamos a identificar las relaciones que hay entre entidades, entonces pudimos identificar las siguientes:

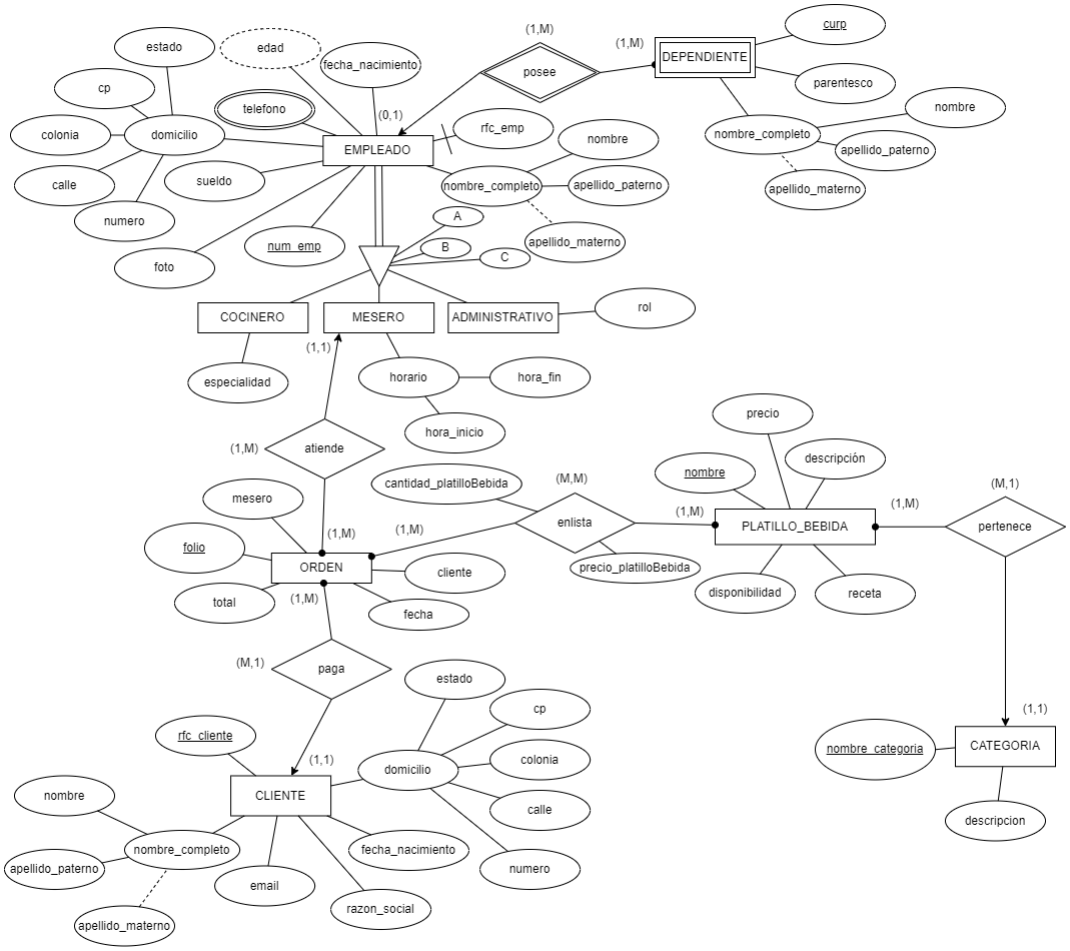
- Un empleado puede tener dependientes.
- Un platillo/bebida pertenece a una categoría.
- Un Mesero puede levantar una orden.
- Una orden debe enlistar los datos de los Platillos/bebidas.

Además de estas relaciones que identificamos, debemos tener en cuenta las restricciones que se nos dan sobre el tipo de relación que hay entre entidades, en nuestro caso tenemos: un empleado puede tener varios puestos y que un platillo o bebida solo pertenece a una categoría.
En el caso de la primera restricción aquí no precisamente estamos hablando de una relación como tal, ya que nosotros identificamos que los puestos de mesero, cocinero y administrativo son especializaciones de la entidad de empleado, ante esto si queremos cumplir con la primera restricción estamos diciendo que la especialización no cuenta con exclusividad, y que se tiene una relación Total. Para el caso de la segunda restricción estamos diciendo que la relación entre platillo/bebida y la categoría es de tipo Muchos a uno.
Después de esto vamos a analizar cada entidad con sus atributos, además de que debemos tomar en cuenta las siguientes consideraciones:

- Donde esté presente el atributo domicilio, está compuesto por estado, código postal, colonia, calle y número.
- Donde esté presente el atributo nombre, está compuesto por nombre, apellido paterno y materno, que este puede ser nulo.

Y ahora identificaremos los atributos con alguna característica que los haga diferentes al resto, ene le caso del empleado podemos decir que su número de empleado es la llave principal y el RFC una llave candidata, en el caso de los teléfonos, nos está hablando en plural, es decir que puede tener varios, por ello este es un atributo multivaluado, en el caso de la edad lo podemos considerar como un atributo calculado. Ahora en el caso de la entidad dependiente el CURP es nuestra llave primaria. En la entidad de Platillos y Bebidas el nombre es la llave primaria, en la entidad Categoría el nombre es la llave primaria, en la entidad Orden el folio es la llave primaria, y el total puede ser calculado y ya en el caso de la entidad cliente el RFC es la llave primaria.

Ahora analizando cada tipo de relación que hay entre las entidades tenemos que el empleado tiene o posee dependientes, en este caso como ya habíamos dicho, tenemos una relación uno a muchos, ahora existe una relación entre el Mesero y una orden, en este caso si analizamos podemos decir que un mesero puede atender muchas ordenes, pero una orden solo puede ser atendida por un mesero, entonces podemos decir que hay una relación de uno a muchos, ahora otra relación hay entre la orden y el cliente, aquí podemos decir que un cliente puede pagar muchas ordenes, pero una orden solo puede ser pagada por un cliente, entonces de igual forma tenemos una relación uno a muchos, la siguiente relación es entre la orden y el platillo/bebida, en este caso podemos decir que una orden enlista muchos platillos/bebidas y muchos platillos/bebidas pueden estar enlistados en una orden, por ello tenemos una relación muchos a muchos y ya por ultimo hay una relación entre los platillos y las categorías, ya que podemos decir que un platillo pertenece a una categoría, pero una categoría le pertenecen muchos platillos, por ello tenemos una relación uno a muchos.
Bien ya habiendo analizado todos estos elementos ya podemos mostrar el MER de este problema el cual es el siguiente:



Ahora pasemos a realizar la representación a MR entonces empecemos checando las entidades que establecimos, bien la primer entidad será la del empleado, recordando habíamos dicho que el número del empleado iba a ser muestra llave primaria, para establecer esto usamos un (PK) luego establecimos el nombre, apellido paterno y en el caso del apellido materno, este puede ser nulo, es decir que al agregar algún valor a la base de datos el usuario pue no ingresar el apellido materno, para hacer esto a la declaración le ponemos una (N) para indicar que puede ser nulo, para el caso del atributo que establecimos como clave candidata, RFC, este se establece usando un (U), a la edad la establecimos como un atributo calculado entonces ese se representa con una (C), ya después solo se agregaron los demás atributos estableciendo el nombre del atributo, el tipo de variable que vamos a estar trabajando y en algunos casos se puede establecer la longitud del atributo.

```
EMPLEADO: {num_emp int (PK), nombre varchar (25), apellido_paterno
varchar(15), apellido_materno varchar (15) (N), rfc_emp char (13)
(U), fecha_nacimiento date, edad int (C), estado varchar (10), cp
int, colonia varchar (25), calle varchar (15), numero_calle int,
sueldo money, foto bytea}
```

Aquí se establece que cuando en alguna entidad tengamos algún atributo multivaluado, se debe de crear una tabla aparte, y si recordamos el atributo de teléfonos en la entidad de empleado es multivaluado, por ello creamos una tabla para los teléfonos, donde se debe establecer que cada número debe ser único y por ello son la llave primaria y además cada número debe de almacenarse en algún cliente por eso es que se la asigna un

numero de empleado, que esta la guardamos como una llave foránea y se establece como (FK) y quedaría de la siguiente forma:

```
EMPLEADO_TELEFONO: {telefono char (10) (PK), num_emp int (FK)}
```

Ahora vamos a ver cómo crear las tablas para el cocinero, administrativo y mesero, para este caso como son Especialidades y por ello estos tienen los mismos atributos que la entidad de empleado entonces aquí solo tenemos que asignar un numero de empleado a los atributos que le correspondan según sea la especialización:

```
COCINERO: {num_emp int (FK)(PK), especialidad varchar (100)}
ADMINISTRATIVO: {num_emp int (FK)(PK), rol varchar (50)}
MESERO: {num_emp int (FK)(PK), hora_inicio time, hora_fin time}
```

En la siguientes ya nada mas se crean las tablas con ayuda de los atributos que ya vimos por entidad, solo hay saber en que tabla se debe colocar la llave foránea, esto dependiendo de la relación que se tenga, por ejemplo sabemos que el dependiente es una entidad débil ya que depende del Empleado, por lo tanto en la entidad dependiente se debe pone la llave foránea, que en este caso seria el numero de empleado, y asi con cada una de la entidades y las relaciones con esto el resto de las tablas quedarían:

```
DEPENDIENTE: {curp char(18) (D) (PK), num_emp int (FK), parentesco
varchar(10), nombre varchar(25), apellido_paterno varchar(15),
apellido_materno varchar(15)(N)}

CLIENTE: {rfc_cliente char (13) (PK), nombre varchar (25),
apellido_paterno varchar (15),apellido_materno varchar (15) (N),
email varchar (50), razón_social varchar (50), fecha_nacimiento
date, estado varchar (15), cp int, colonia varchar(25),  calle
varchar(15), numero int}

ORDEN: {folio char (7) (PK), num_empleado int (FK), rfc_cliente
char (13) (FK), fecha date, total money}

CATEGORIA: {nombre_Categoria varchar (15)(PK), descripción varchar
(150)}

PLATILLO_BEBIDA: {nombre_platilloBebida varchar (25) (PK), precio
money, descripción varchar(200), receta varchar (400),

disponibilidad int, nombre_categoria varchar (15)(FK)}

ENLISTA: {[folio char (7), nombre_platilloBebida varchar (25)](FK)
(PK), cantidad_platiloBebida int, precio_platilloBebida money}
```

4. Implementación

Ahora veamos como pasar de esta representación a la creación de tablas en PgSQL, bien empecemos con la tabla de empleado, pero antes de empezar hay que tomar en cuenta algunas cuestiones, por ejemplo, debemos establecer el nombre de nuestra base de datos

donde vamos a guardar nuestras tablas, a esta base de datos la vamos a llamar restaurantROYJ y para conectarnos a esta base de datos usamos el comando "`\c`"seguido del nombre de la base de datos a la que vamos a conectarnos.

Bien otra cuestión es que como vamos a empezar a construir la tabla para los empleados, si recordamos el atributo de la edad lo colocamos como atributo calculado, por ende, es importante primeramente crear la función que permita calcular la edad. Para este crearemos, o remplazaremos en caso de ser necesario, una función que nos permita regresar la edad, para ello le pediremos que nos pase una fecha de nacimiento, y además le pediremos que el resultado se regrese en el leguaje de plpgsql y que sea inmodificable, después de esto para obtener la edad extraeremos los años de la función "age" donde le pasamos como parámetro la fecha de nacimiento. La función AGE en PostgreSQL calcula la diferencia entre dos fechas devolviendo años, meses y días.

La función AGE () resta el segundo argumento del primero devolviendo un intervalo como resultado. O en nuestro caso como solo le pasamos solo un parámetro entonces resta la fecha actual con la fecha que le pasemos. Entonces con esto lo que hacemos que solo nos regrese los años de la diferencia entre la fecha actual y la fecha de nacimientos, siendo esto la edad. Todo esto se muestra a continuación:

```
--Crear Base de Datos
create database proyectoequipobd;

--Conectar con la Base de Datos
\c proyectoequipobd;


-- Función que permite calcular la edad
create or replace function get_age(fecha_nacimiento date) returns int
language plpgsql immutable
as $CODE$
begin
    return extract (year from age(fecha_nacimiento));
end;
$CODE$;
```

Ahora si pasemos a desarrollar la tabla de empleado, aquí ponemos todos los atributos casi como los pusimos en la representación intermedia, solo que aquí se estableció que en ningún atributo pudiera ser nulo, excepto el apellido materno, además que al construir las tabla ya en PgSql, las llaves foránea y principal ya no se declaran con ayuda del (FK) y (PK), respectivamente, en este caso usaremos el constraint este nos permite establecer un nombre a nuestras llaves, el cual nos ayudara a identificar as rápido de donde viene un error, ya que nos arrojara el nombre que le pongamos en vez de solo mandar el error por defecto de postgres, una vez definido el nombre establecemos que va ser si llave primaria o foránea, después establecemos a que atributo le asignaremos esta propiedad, entonces la tabla empleado nos quedaría:

```sql
-- Tabla Empleado
create table empleado(
    num_empleado int not null,
    nombre varchar(25) not null,
    apellido_pat varchar(15) not null,
    apellido_mat varchar (15) null,
    rfc_emp char(13) not null,
    fecha_nacimiento date not null,
    edad int generated always as (get_age(fecha_nacimiento)) stored,
    estado varchar(10) not null,
    cp int not null,
    colonia varchar(25) not null,
    calle varchar(15) not null,
    numero_calle int not null,
    sueldo money not null,
    foto bytea not null,
    constraint empleado_pk primary key (num_empleado)
);
```

En las siguientes tablas pues casi es lo mismo que la anterior, donde solo casi copiamos lo que se hizo en la representación intermedia, solo que las tablas de `Empleado_Telefono`, cocinero, administrativo, mesero y dependiente, se agregó una propiedad, estos debido a que estas están relacionada con la tabla de empleado, donde en dado caso de que si se elimina o actualiza algún empleado, entonces debería también eliminarse o actualizarse de las otras tablas los datos relacionados a ese empleado, ante esta idea se decido poner la siguiente propiedad:

```
references empleado(num_empleado) on delete cascade on update cascade
```

con esto ya nos aseguramos de que si un numero de empleado en la tabla de empleado se elimina o se actualiza entonces también se modifica en la tabla correspondiente. Entonces las tablas de `Empleado_Telefono`, cocinero, administrativo, mesero y dependiente, quedarían:

```sql
-- Tabla administrativo
create table administrativo (
    num_empleado int not null,
    rol varchar(50) not null,
    constraint administrativo_pk primary key (num_empleado),
    constraint admin_emp_fk foreign key (num_empleado)
    references empleado(num_empleado) on delete cascade on update cascade
);
```

```sql
-- Tabla cocinero
create table cocinero(
    num_empleado int not null,
    especialidad varchar(100) not null,
    constraint cocinero_pk primary key (num_empleado),
    constraint cocinero_emp_fk foreign key (num_empleado)
    references empleado(num_empleado) on delete cascade on update cascade
);
```

```sql
-- Tabla administrativo
create table administrativo (
    num_empleado int not null,
    rol varchar(50) not null,
    constraint administrativo_pk primary key (num_empleado),
    constraint admin_emp_fk foreign key (num_empleado)
    references empleado(num_empleado) on delete cascade on update cascade
);
```

```sql
-- Tabla mesero
create table mesero(
    num_empleado int not null,
    hora_inicio time not null,
    hora_fin time not null,
    constraint empleado_mesero_pk primary key (num_empleado),
    constraint mesero_emp_fk foreign key (num_empleado)
    references empleado(num_empleado) on delete cascade on update cascade
);
```

```sql
-- Tabla dependiente
create table dependiente(
    curp char(18) not null,
    num_empleado int not null,
    parentesco varchar(10) not null,
    nombre varchar(25) not null,
    apellido_pat varchar(15) not null,
    apellido_mat varchar(15) null,
    constraint dependiente_pk primary key (curp),
    constraint depe_emp_fk foreign key (num_empleado)
    references empleado(num_empleado) on delete cascade on update cascade
);
```

Ahora crearemos la tabla de cliente donde se siguen los mismos principios que en la tabla de empleado, entonces nos quedaría:

```sql
-- Tabla cliente
create table cliente(
    rfc_cliente char(13) not null,
    nombre varchar(25) not null,
    apellido_pat varchar(15) not null,
    apellido_mat varchar(15) null,
    email varchar(50) not null,
    razon_social varchar(50) not null,
    fecha_nacimiento date not null,
    estado varchar(15) not null,
    cp int not null,
    colonia varchar(25) not null,
    calle varchar(15) not null,
    numero_calle int not null,
    constraint cliente_pk primary key (rfc_cliente)
);
```

Ahora veremos las tablas de orden, categoría, `platillo_bebida` y enlista, en este caso a estas tablas también se le agrego el references para el caso de eliminación o actualización, ya que estas tienen alguna relación con alguna otra tabla, en el caso de la tabla de orden esta tiene dos references uno para la tabla de empleado, conforme al número de empleado, y otro para la tabla de cliente, conforme al RFC del cliente, ya que se requiere que la orden se vea actualizada en caso recibir un cambio ya sea en el mesero que atiende o del

cliente que pide algo del restaurante.

```
-- Tabla orden (ORD-001)
create table orden(
    folio char(7) not null,
    num_empleado int not null,
    rfc_cliente char(13) not null,
    fecha date default now(),
    total money not null,
    constraint orden_pk primary key (folio),
    constraint orden_emp_fk foreign key (num_empleado)
    references empleado(num_empleado) on delete cascade on update cascade,
    constraint orden_cli_fk foreign key (rfc_cliente)
    references cliente(rfc_cliente) on delete cascade on update cascade
);
```

La tabla de categoría sigue los mismos principios de las tablas de empleado y cliente, y quedaría:

```
-- Tabla categoria
create table categoria(
    nombre_categoria varchar(15) not null,
    descripcion varchar(150) not null,
    constraint categoria_pk primary key (nombre_categoria)
);
```

Ahora para las de `platillo_bebida` se agrega un references para la tabla de categoría, conforme al nombre de la categoría, ya que si se elimina una categoría entonces lo platillos dentro de ella también

```
-- Tabla platillo_bebida
create table platillo_bebida(
    nombre_platilloBebida varchar(25) not null,
    precio money not null,
    descripcion varchar(200) not null,
    receta varchar(400) not null,
    disponibilidad int not null,
    nombre_categoria varchar(15) not null,
    constraint platilloBeb_pk primary key(nombre_platilloBebida),
    constraint platBe_cat_fk foreign key(nombre_categoria)
    references categoria(nombre_categoria) on delete cascade on update cascade
);
```

También para la tabla de enlista se agregaron 2 references una para la orden, conforme al folio, y otro para el `paltillo_bebida`, conforme al nombre del platillo/bebida

```sql
-- Tabla enlista (orden-platillo)
create table enlista(
    folio char(7) not null,
    nombre_platilloBebida varchar(25) not null,
    cantidad_platilloBebida int not null,
    precio_platilloBebida money not null,
    constraint enlista_pk primary key(folio,nombre_platilloBebida),
    constraint enlista_folio foreign key(folio)
    references orden(folio) on delete cascade on update cascade,
    constraint enlista_nomPB foreign key(nombre_platilloBebida)
    references platillo_bebida(nombre_platilloBebida) on delete cascade on update cascade
);
```

Uno de los objetivos era "Permitir obtener el nombre de aquellos productos que no estén disponibles." Para esto realizamos un for para ir registrando el atributo `nombre_platillobebida` de todos los registros que encuentra donde la disponibilidad es 0. El return next es para que continue el for. La función tiene de salida una varchar que ps es el tipo de dato q queda con el nombre, lo de reg es porque es un record que sirve para regresar varios registros.

```sql
--Permitir obtener el nombre de aquellos productos que no esten disponibles.
create or replace function fn_prod_no_disp(producto_no_disponible out varchar)
returns setof character varying as $$
--declaracion de variables
declare
    reg record;
begin
    --REVISA LOS PRODUCTOS CUYA DISPONIBILIDAD ESTA EN 0 Y LOS VA REGISTRANDO
    for reg in select nombre_platillobebida from platillo_bebida
    where disponibilidad = 0 loop
        producto_no_disponible := reg.nombre_platillobebida;
        return next;
    end loop;
    return;
end;
$$
language plpgsql;
```

Otro objetivo es que "Dado un número de empleado, mostrar la cantidad de ordenes que ha registrado en el día, así como el total que se ha pagado por dichas órdenes. Si no se trata de un mesero, mostrar un mensaje de error", para esto primeramente se estableció crear, o remplazar en caso de ya existir, una función para mostrar las ordenes por empleado, a esta función se la va a pasar como parámetro el número del empleado, y esta función debe de regresar una tabla donde se muestre los datos solicitados, entonces dentro de esta función se crearon 3 variables en la cuales podemos almacenar la cantidad de órdenes, total pago en ese día y para comprobar si el empelado es un mesero, después de esto debemos de verificar si existen órdenes del mesero pasado como parámetro, esto con ayuda de un select podamos contar todos los datos donde el número de empleado, este debe de pertenecer tanto a la tabla de empleado como a la del mesero, sea igual al número de empleado que se pasó como parámetro y el valor que regrese esta cuenta se almacena en la variable que creamos para ver si el empleado es un mesero, en este caso si la variable es diferente de 0 entonces es un mesero, si es igual a cero entonces el empleado no es un mesero y se debe de mandar u mensaje de error, ahora bien caso de ser un mesero entonces pasamos contar el número de ordenes que realizo el mesero, esta cuanta se la almacenamos dentro de la variable de cantidad de órdenes, con ayuda de un into, esta cantidad de ordenes se obtiene de juntar la tabla de empelado y la de la orden, en

donde compartan el mismo número de empleado, con esto apenas solo tenemos todos las ordenes de todos los empleados, ahora con un where excluiremos a que solo cuente donde el número de empleado de la tabla de empleado sea igual al que se pasó como parámetro y que además la fecha en la orden sea la fecha actual.

En caso de existir ordenes ese día, se deberá regresa los datos agrupados de la consulta de 2 tablas, la primera es de los datos que pertenezcan a la tabla de empleado y de orden donde el numero de empleado en ambos sean iguales, además debe pertenecer a los datos que regresen de la segunda consulta, la cual consiste en que se regresaran los números de empleado y la suma de la columna de total en la tabla de orden, de donde pertenezcan tanto de empleado como de la orden conforme a donde son iguales el numero de empleado para ambos casos, ahora con un where excluiremos a que solo cuente donde el número de empleado de la tabla de empleado sea igual al que se pasó como parámetro y que además la fecha en la orden sea la fecha actual, y esta consulta 2 se encentra a grupada por el número de empleado. Ahora se debe mostrar los datos que pertenezcan tanto a la consulta 1 como a la 2, en donde los números de empleados de la consulta 2 sean iguales a los números de empleados de la tabla de empleado, entonces nos debe de mostrar los datos donde el número de empleado sean iguales y que se compare la fecha de la orden con la fecha de hoy, todo esto se va a agrupar por el numero de empleado. Después de hacer eso se crea un mensaje en el cual nos dice el número de registro y el total a apagar.

```
--Dado un número de empleado, mostrar la cantidad de órdenes que ha registrado
--en el dia asi como el total que se ha pagado por dichas ordenes.
create or replace function fn_ordenes_empleado(p_num_empleado in numeric)
returns table (num_empleado integer,cant_ordenes bigint,total money) as $fn_ordenes_empleado$
declare
--declaracion de variables
v_cant_ordenes numeric;
v_total_dia numeric;
v_es_mesero numeric;
begin
    --VERIFICA SI HAY REGISTROS EN MESERO CON EL NUMERO DE EMPLEADO PROPORCIONADO
    select count(*) into v_es_mesero from empleado e join mesero m on e.num_empleado=m.num_empleado
    where e.num_empleado=p_num_empleado;

    if v_es_mesero != 0 then
        --VERIFICA LAS ORDENES QUE HA HECHO ESE EMPLEADO EN ESE DIA
        select  count(*) into v_cant_ordenes
        from empleado e join orden o on e.num_empleado=o.num_empleado
        where e.num_empleado=p_num_empleado and o.fecha=current_date;
        select  sum(o.total) into v_total_dia
        from empleado e join orden o on e.num_empleado=o.num_empleado
        where e.num_empleado=p_num_empleado and o.fecha=current_date;
        --VERIFICA SI NO HA HECHO ALGUNA ORDEN ESE DIA
        if v_cant_ordenes = 0 then
            RAISE NOTICE 'El empleado con el numero % ha registrado 0 ordenes hoy.',
            p_num_empleado;
        else
            return query select e.num_empleado,count(*),sb.tot
            from empleado e join orden o on e.num_empleado=o.num_empleado join (select e.num_empleado as nm,sum(o.total) as tot
            from empleado e join orden o on e.num_empleado=o.num_empleado where e.num_empleado=p_num_empleado and o.fecha=current_date
            group by e.num_empleado
            ) sb on sb.nm=e.num_empleado
            where e.num_empleado=p_num_empleado and o.fecha=current_date
            group by e.num_empleado,sb.tot;
            RAISE NOTICE 'El empleado con el numero % ha registrado % orden(es) hoy y su total es %.',
            p_num_empleado,v_cant_ordenes,v_total_dia;
        end if;
    else
        RAISE EXCEPTION 'El empleado con el numero % no es un mesero.', p_num_empleado;
    end if;
end;
$fn_ordenes_empleado$
language plpgsql;
```

Otro objetivo es que "De manera automática se genere una vista que contenga información necesaria para asemejarse a una factura de una orden", para esto creamos una función, o remplazamos en caso de ser necesario, donde solo le pasamos como parámetro el folio de la orden, y esta nos regrese una tabla con los datos de una factura, como el folio de la orden, la fecha, el RFC del cliente, el producto, el precio, la cantidad y el total a pagar. Entonces para que se obtenga esto la función debe realizar una consulta donde primeramente chequemos existen datos que pertenezcan tanto a la tabla de orden y a la

tabla de enlista, bajo la condición que el folio en la orden sea igual al folio en la enlista, además de que también pertenezca a la tabla de `platillo_bebida`, bajo la condición que el nombre del platillo en la tabla de `platillo_bebida` sea igual al nombre en la tabla de enlista, si existen datos que pertenezcan a las 3 tablas entonces ya podemos obtener los datos solicitados donde el folio en la tabla de orden sea igual al folio que se pasó como parámetro. Entonces la funciona escrita para PgSQL quedaria:

```
--De manera automatica se genere una vista que contenga
--informacion necesaria para asemejarse a una factura de una orden
create or replace function fn_vista_factura_orden(p_folio in char(7)) returns table(
    folio char(7),fecha date,rfc_cliente char(13),producto varchar,precio money,
    cantidad integer,total money
) as $$
begin
    return query
    select o.folio,o.fecha,o.rfc_cliente,e.nombre_platillobebida,
    pb.precio,e.cantidad_platillobebida,o.total
    from orden o join enlista e on o.folio=e.folio
    join platillo_bebida pb on pb.nombre_platillobebida=e.nombre_platillobebida
    where o.folio=p_folio;
end;
$$
language 'plpgsql'
```

El siguiente objetivo es que, “Dada una fecha, o una fecha de inicio y fecha de fin, regresar el total del número de ventas y el monto total por las ventas en ese periodo de tiempo. Nota: Con producto se hace referencia a los alimentos y bebidas”, para esto creamos una función donde la pacemos como parámetro las fechas de inicio y fin del periodo de tiempo en que queremos obtener los datos solicitados, dentro ya de la función declaramos 2 variables una para almacenar la cantidad y otra para total. Ya dentro del proceso, la función debe de contar los datos de la tabla de orden, donde los datos tengan las fechas entre la de inicio y la de final y este valor se almacene en la variable que creamos para cantidad, y luego se realiza lo mismo para el total a pagar, donde realizamos la suma del total que se pagaron por órdenes, en el mismo periodo de tiempo.

Si en la variable para la cantidad no contienen algún valor asignado, entonces se dice no hay ordenes entre el rango de fechas pasado, en caso contrario si los hay, entonces debe de regresa el conteo y la suma de totales por orden, en forma de mensaje. Entonces su escritura quedaría de la siguiente manera:

```
--Dada una fecha, o una fecha de inicio y fecha de fin, regresar el total del
--número de ventas y el monto total por las ventas en ese periodo de tiempo.

create or replace function fn_ventas_intervalo(
    p_fecha_inicial in varchar,
    p_fecha_final in varchar
) returns table(cantidad bigint, total money) as $$
declare
--declaracion de variables
v_cantidad numeric;
v_total numeric;
begin
    --CUENTA LAS ORDENES Y EL TOTAL QUE SE GENERARON EN ESAS FECHAS
    select count(*) into v_cantidad from orden
    where fecha between to_date(p_fecha_inicial,'dd-mm-yyyy')
    and to_date(p_fecha_final,'dd-mm-yyyy');

    select sum(orden.total) into v_total from orden
    where fecha between to_date(p_fecha_inicial,'dd-mm-yyyy')
    and to_date(p_fecha_final,'dd-mm-yyyy');
    --IMPRIME LOS VALORES OBTENIDOS
    if v_cantidad = 0 then
        RAISE NOTICE 'No se realizaron ordenes en el intervalo % / % .',
        to_date(p_fecha_inicial,'dd-mm-yyyy'),to_date(p_fecha_final,'dd-mm-yyyy');
    else
        return query select count(*), sum(orden.total) from orden
        where fecha between to_date(p_fecha_inicial,'dd-mm-yyyy')
        and to_date(p_fecha_final,'dd-mm-yyyy');
        RAISE NOTICE 'La cantidad de ordenes en ese intervalo % / % es % y el total es % .',
        to_date(p_fecha_inicial,'dd-mm-yyyy'),to_date(p_fecha_final,'dd-mm-yyyy'),v_cantidad,v_to
    end if;
end;
$$
language plpgsql;
```

Otro objetivo tenemos "Vista que muestre todos los detalles del platillo más vendido", para esto se tendrán en total 3 consultas, la consulta mas interna es una donde nos devuelve la suma de cantidad de platillo que hay y el nombre del platillo, cuyos datos pertenezcan a la `platillo_bebida` y a la enlista, cuyo condición es que el nombre de platillo sea el mismo para `platillo_bebida` y la tabla de enlista y esta consulta se agrupa conforme al nombre del platillo, esta consulta interna le diremos consulta 3, ahora veremos la consulta 2, en ese caso nos regresa el máximo valor en la suma de la consulta 3 y también se regresara el nombre del platillo , y la condición que se debe de cumplir es que el nombre del platillo sea el mismo tanto en el `platillo_bebida` y la tabla de enlista. Ahora la consulta 1 (superior), es para obtener los datos que pertenezcan a la tabla de enlista y el `platillo_bebida`, la condición es que el nombre del platillo para ambas tablas sea igual. Ahora bien, se puede llevar acabo un join entre la consulta 2 como la consulta 3, bajo la condición de que el nombre del platillo sea el mismo entre la consulta 3 y la tabla de `platillo_bebida`, a esto le diéremos consulta 4.

Ahora bien, se puede llevar acabo un join entre la consulta 1 como la consulta 4, este se va a agrupar conforme al nombre del platillo y el valor máximo que regresa la consulta 4, los datos donde la suma de cantidad de platillos debe ser igual a el valor máximo que regresa la consulta 4

```sql
--Vista que muestre todos los detalles del platillo mas vendido
create or replace view vista_platillo_mas_vendido(nombre,precio,descripcion,receta,
disponibilidad,categoria) as
    select pb.nombre_platillobebida,pb.precio,pb.descripcion,pb.receta,
    pb.disponibilidad,pb.nombre_categoria from platillo_bebida pb
    join enlista e on pb.nombre_platillobebida=e.nombre_platillobebida
    natural join (
        select max(sb.suma) as maximo from platillo_bebida pb
        join enlista e on pb.nombre_platillobebida=e.nombre_platillobebida
        join (select sum(cantidad_platillobebida) as suma,
        pb.nombre_platillobebida as nm from platillo_bebida pb
        join enlista e on pb.nombre_platillobebida=e.nombre_platillobebida
        group by pb.nombre_platillobebida) sb on sb.nm=pb.nombre_platillobebida
    ) sbn
    group by pb.nombre_platillobebida,sbn.maximo
    having sum(e.cantidad_platillobebida)=sbn.maximo;
```

5. Presentación

- Actualizar los totales por producto y venta al agregar un producto a una orden

Consultando la tabla `platillo_bebida`:

nombre_platillobebida [PK] character varying (25)	precio money	descripcion character varying (200)	receta character varying (40	disponibilidad integer	nombre_categoria character varying (15)
Arrachera	$150.00	Corte fino de res de 45...	Marinar el corte en...	10	Carnes
Hamburguesa	$100.00	Platillo tipo sÃ¡ndwich ...	Moler y mezclar la...	5	Carnes
Pollo al horno	$130.00	Pollo cocinado en sus ...	Limpiar el pollo y u...	4	Carnes
Chuleta de cordero	$200.00	Pollo cocinado en sus ...	Sazonar la carne c...	10	Carnes
Limonada	$30.00	Bebida refrescante ela...	Exprimir limones, ...	10	Bebidas
Soda	$10.00	Bebida gaseosa de div...	N/A	20	Bebidas
Cerveza	$20.00	Bebida alcohÃ³lica ela...	N/A	30	Bebidas

Consultando la tabla orden:

	folio [PK] character (7)	num_empleado integer	rfc_cliente character (13)	fecha date	total money
1	ORD-001	3	SARV85091183A	2022-05-27	$230.00
2	ORD-002	6	OIRJ870911DFA	2022-05-20	$400.00
3	ORD-003	6	SADE950522HDF	2022-05-27	$270.00
4	ORD-004	7	GATM780118C48	2022-05-21	$230.00
5	ORD-005	7	MANP780319KD8	2022-03-22	$380.00
6	ORD-006	10	GOMS770704S3A	2022-02-17	$700.00
7	ORD-007	10	FESD900810UA8	2022-05-27	$1,200.00
8	ORD-008	10	OIRJ870911DFA	2022-04-03	$150.00
9	ORD-009	10	PEFV980319CXA	2022-03-29	$700.00
10	ORD-010	11	DIHL720925BU5	2022-05-03	$1,630.00
11	ORD-011	16	LOMM961019UI1	2022-05-27	$200.00
12	ORD-012	16	SARV85091183A	2022-03-15	$470.00
13	ORD-013	19	AATM770626V39	2022-01-02	$620.00
14	ORD-014	21	PEYM9012092P4	2022-05-27	$970.00

Ejecutando:

```
insert into enlista values('ORD-001','Cerveza',5,100);
```

Se obtiene en `platillo_bebida`:

Pozole	$100.00	Platillo a base de gra...	Calentar 5 litros agua c...	5	Comida Mexicana
Flan	$50.00	Postre elaborado con...	Verter azÃºcar en una ...	10	Postres
Pastel de Chocolate	$50.00	Rebanada de delicios...	Combinar mantequilla, ...	22	Postres
TiramisÃº	$100.00	Postre que se prepar...	Batir crema a velocidad...	1	Postres
Pay de LimÃ³n	$50.00	Rebanada de delicios...	Para la base, mezclar l...	6	Postres
Cerveza	$20.00	Bebida alcohÃ³lica el...	N/A	25	Bebidas

	folio [PK] character (7)	num_empleado integer	rfc_cliente character (13)	fecha date	total money
1	ORD-001	3	SARV85091183A	2022-05-27	$330.00
2	ORD-002	6	OIRJ870911DFA	2022-05-20	$400.00
3	ORD-003	6	SADE950522HDF	2022-05-27	$270.00
4	ORD-004	7	GATM780118C48	2022-05-21	$230.00
5	ORD-005	7	MANP780319KD8	2022-03-22	$380.00
6	ORD-006	10	GOMS770704S3A	2022-02-17	$700.00
7	ORD-007	10	FESD900810UA8	2022-05-27	$1,200.00
8	ORD-008	10	OIRJ870911DFA	2022-04-03	$150.00
9	ORD-009	10	PEFV980319CXA	2022-03-29	$700.00
10	ORD-010	11	DIHL720925BU5	2022-05-03	$1,630.00
11	ORD-011	16	LOMM961019UI1	2022-05-27	$200.00
12	ORD-012	16	SARV85091183A	2022-03-15	$470.00
13	ORD-013	19	AATM770626V39	2022-01-02	$620.00
14	ORD-014	21	PEYM9012092P4	2022-05-27	$970.00

Actualizando los totales por producto y venta al agregar un producto a una orden

Generar un índice

En este caso se define un índice sobre el atributo "disponibilidad"de la tabla "`platillo_bebida`.[en] este caso se definió el índice de esta manera porque el atributo "disponibilidadçonstantemente es usado en consultas realizadas por diversas funciones como el caso de la función que permite obtener el nombre de los platillos o bebidas que no están disponibles o la función para actualizar los totales por producto y venta al agregar un producto a una orden, por tal motivo se definió el índice de la siguiente manera:

```
create index dispo on platillo_bebida(disponibilidad);
```

En este caso se trata de un índice no agrupado ya que no altera la forma en que los datos se almacenan en la tabla y permite recuperar los resultados de las consultas que emplean al atributo "disponibilidad".

Función con la cual dado un numero de empleado obtener la cantidad de ordenes que ha registrado durante el día, así como el total que se ha pagado por dichas órdenes.

Consultado la tabla orden:

	folio [PK] character (7)	num_empleado integer	rfc_cliente character (13)	fecha date	total money
1	ORD-001	3	SARV85091183A	2022-05-27	$230.00
2	ORD-002	6	OIRJ870911DFA	2022-05-20	$400.00
3	ORD-003	6	SADE950522HDF	2022-05-27	$270.00
4	ORD-004	7	GATM780118C48	2022-05-21	$230.00
5	ORD-005	7	MANP780319KD8	2022-03-22	$380.00
6	ORD-006	10	GOMS770704S3A	2022-02-17	$700.00
7	ORD-007	10	FESD900810UA8	2022-05-27	$1,200.00
8	ORD-008	10	OIRJ870911DFA	2022-04-03	$150.00
9	ORD-009	10	PEFV980319CXA	2022-03-29	$700.00
10	ORD-010	11	DIHL720925BU5	2022-05-03	$1,630.00
11	ORD-011	16	LOMM961019UI1	2022-05-27	$200.00
12	ORD-012	16	SARV85091183A	2022-03-15	$470.00
13	ORD-013	19	AATM770626V39	2022-01-02	$620.00
14	ORD-014	21	PEYM9012092P4	2022-05-27	$970.00

Al llamar la función para el empleado numero 10:

	fn_ordenes_empleado record
1	(10,1,"$1,200.00")

Con lo cual la función retorna para el empleado número 10 que ha registrado una orden en el día con un pago total de $1200, tal y como se mostró en la consulta a la tabla orden.
Vista que muestre los detalles del platillo más vendido Realizando la siguiente consulta:

select `nombre_platillobebida`, sum(`cantidad_platillobebida`) as `total_vendido` from enlista group by `nombre_platillobebida`;
Se tiene:

22	Camarones empanizados	2
23	Jugo de Naranja	4
24	Cafe	8
25	Ensalada Griega	2
26	Cerveza	26

Por lo que sabemos que el producto mas vendido es la cerveza, con 26 unidades vendidas, al consultar la vista de la siguiente manera:

```
select * from vista_platillo_mas_vendido;
```

Se obtiene:

	nombre character varying (25)	precio money	descripcion character varying (200)	receta character varying (400)	disponibilidad integer	categoria character varying (15)
1	Cerveza	$20.00	Bebida alcohÃ³lica elaborada a partir de azÃºcares obtenidas d...	N/A	30	Bebidas

Obtenien
do de esta forma los detalles del platillo o bebida más vendido que es la cerveza tal y

como se esperaba.
Función para obtener el nombre de aquellos productos que no estén disponibles
Al consultar la tabla `platillo_bebida`, obtenemos que los productos no disponibles son:

Ensalada Griega	$80.00	Ensalada elaborada con to...	AÃ±adir tomate cortad...	0	Ensaladas
Gazpacho	$50.00	Sopa refrescante elaborad...	Picar tomates, pepino, ...	0	Sopas

Al llamar la función como se muestra:

select `fn_prod_no_disp`();

Resulta en:

fn_prod_no_disp character varying
Ensalada Griega
Gazpacho

Función que obtiene la información necesaria para asemejarse a la factura de una orden Consultando la tabla orden:

	folio [PK] character (7)	num_empleado integer	rfc_cliente character (13)	fecha date	total money
1	ORD-001	3	SARV85091183A	2022-05-27	$230.00
2	ORD-002	6	OIRJ870911DFA	2022-05-20	$400.00
3	ORD-003	6	SADE950522HDF	2022-05-27	$270.00
4	ORD-004	7	GATM780118C48	2022-05-21	$230.00
5	ORD-005	7	MANP780319KD8	2022-03-22	$380.00
6	ORD-006	10	GOMS770704S3A	2022-02-17	$700.00
7	ORD-007	10	FESD900810UA8	2022-05-27	$1,200.00
8	ORD-008	10	OIRJ870911DFA	2022-04-03	$150.00
9	ORD-009	10	PEFV980319CXA	2022-03-29	$700.00
10	ORD-010	11	DIHL720925BU5	2022-05-03	$1,630.00
11	ORD-011	16	LOMM961019UI1	2022-05-27	$200.00
12	ORD-012	16	SARV85091183A	2022-03-15	$470.00
13	ORD-013	19	AATM770626V39	2022-01-02	$620.00
14	ORD-014	21	PEYM9012092P4	2022-05-27	$970.00

Obteniendo la información para la orden numero 6:
select `fn_vista_factura_orden2`('ORD-006');

	fn_vista_factura_orden record
1	(ORD-006,2022-02-17,GOMS770704S3A,Cerveza,$20.00,1,$700.00)
2	(ORD-006,2022-02-17,GOMS770704S3A,Cafe,$15.00,2,$700.00)
3	(ORD-006,2022-02-17,GOMS770704S3A,"Sopa Miso",$50.00,1,$700.00)
4	(ORD-006,2022-02-17,GOMS770704S3A,"SalmÃ³n al horno",$500.00,1,$700.00)
5	(ORD-006,2022-02-17,GOMS770704S3A,Flan,$50.00,1,$700.00)
6	(ORD-006,2022-02-17,GOMS770704S3A,"Pastel de Chocolate",$50.00,1,$700.00)

Función para obtener el numero de ventas en un periodo de tiempo
Consultando la tabla orden:

	folio [PK] character (7)	num_empleado integer	rfc_cliente character (13)	fecha date	total money
1	ORD-001	3	SARV85091183A	2022-05-27	$230.00
2	ORD-002	6	OIRJ870911DFA	2022-05-20	$400.00
3	ORD-003	6	SADE950522HDF	2022-05-27	$270.00
4	ORD-004	7	GATM780118C48	2022-05-21	$230.00
5	ORD-005	7	MANP780319KD8	2022-03-22	$380.00
6	ORD-006	10	GOMS770704S3A	2022-02-17	$700.00
7	ORD-007	10	FESD900810UA8	2022-05-27	$1,200.00
8	ORD-008	10	OIRJ870911DFA	2022-04-03	$150.00
9	ORD-009	10	PEFV980319CXA	2022-03-29	$700.00
10	ORD-010	11	DIHL720925BU5	2022-05-03	$1,630.00
11	ORD-011	16	LOMM961019UI1	2022-05-27	$200.00
12	ORD-012	16	SARV85091183A	2022-03-15	$470.00
13	ORD-013	19	AATM770626V39	2022-01-02	$620.00
14	ORD-014	21	PEYM9012092P4	2022-05-27	$970.00

Si se quiere saber el numero y monto de las ventas durante el mes de Mayo de 2022 se ejecuta:

select `fn_ventas_intervalo`('01-05-2022','31-05-2022');
Y se obtiene:

	fn_ventas_intervalo record
1	(8,"$5,130.00")

6. Conclusiones

Angeles Estrada Ricardo

En mi conclusión se presentaron muchos problemas que al final pusimos solucionar, se nos dificultaron algunas cosas, pero se pudo salir adelante la programación en postgresql, tiene su chiste, pero me gusta la forma en se pueden realizar la cosas ahí, como pudimos realizar actividades dentro del curso como DDL, DML, DCL, PgSQL, entre otros, e incluso permitió conocer la forma de crear una interfaz gráfica mediante una aplicación web, aprender usar la funciones dentro de los diferente manejadores que usamos a lo largo del semestre. Los retos que se presentaron fue lograr realizar la conexión de la base de datos con la aplicación con la página web, la forma de elegir los datos correctos para funcionar de forma correcta además de que esto fortaleció el conocimiento que teníamos y aclaro las diversas dudas que podíamos tener con respecto la creación de la base de datos y al mismo lo cual es demostrado mediante el diseño y aplicación de los procesos dando como resultado un base de datos funciona, tal vez con sus detalles per que se podría considerar importantes ene el desarrollo del conocimientos sobre las bases de datos

Ramírez González José Miguel

La realización del proyecto se llevó a cabo de manera escalonada, de modo que se comenzó

primeramente con los elementos básicos, después se fue volviendo cada vez más complejo con la incorporación del uso del lenguaje PostgreSQL y posteriormente con la programación. Esto pudo permitir que se tuviera una mejor organización y mayor flujo de trabajo, ya que cada elemento del equipo tenía claro el objetivo del proyecto y las complicaciones pudieron ser resueltas con un tiempo considerable.
El proyecto en cuestión ayudó a reiterar y poner en práctica los temas vistos en el curso, ya que abarco desde las reglas de negocios que se toman en cuenta en un caso de estudios hasta la incorporación de una base de datos en una aplicación web, pasando por el diseño del modelo entidad relación, la representación intermedia, el modelo relacional y el uso del lenguaje PostgreSQL para la creación de la base de datos, la creación de las tablas, inserción de datos y uso de funciones para automatizar procesos. Además de que logró impulsar el conocimiento e interés que se necesitaba para poder formular la aplicación web, ya que esta se logró con el uso de herramientas externas al curso (html, css, php y JavaScript), lo cual considero que es de suma importancia ya que la incorporación de estas herramientas va de la mano con el tema de bases de datos y para un entorno laboral o un proyecto serio se vuelve necesario conocer este tipo de tecnologías. Las dificultades se presentaron en pequeños detalles como la elección de los atributos para cada tabla, la extensión de los campos varchar, la incorporación del índice e incluso la desactualización del manejador PostgreSQL ya que esto limito el correcto funcionamiento de las funciones. Los retos que se presentaron fue lograr la conexión de la base de datos con la aplicación web, la elección de los elementos correctos para la forma estética de la misma y el guardado y recuperación de las fotos en la base de datos. Finalmente se considera que en el desarrollo del proyecto se resolvieron las diversas dudas con respecto al mismo lo cual es demostrado mediante el diseño y aplicación de los procesos dando como resultado un trabajo más que satisfactorio.

Muro León Yoaddan Yokaem

Con la realización del presente trabajo se ha permitido poner en práctica todos los conocimientos estudiados a lo largo del curso, tales como, análisis de requerimientos, modelo entidad relación, modelo entidad relación extendido, representación intermedia, modelo relacional, lenguaje SQL, DDL, DML, DCL, PgSQL, entre otros, e incluso permitió conocer la forma de crear una interfaz gráfica mediante una aplicación web la cual se conecta e interactúa con la base de datos, para la elaboración de este proyecto se trabajaron sobre todas las etapas de diseño de una base de datos iniciando por el análisis de requerimientos, diseño conceptual, diseño lógico y diseño físico, para esto en un principio se planteó el modelo entidad relación extendido en base a los requerimientos solicitados para la base de datos del restaurant, posteriormente se desarrolló la representación intermedia, en donde se aplicaron todas las reglas de transformación estudiadas para todos los elementos del MRE, como lo son la transformación de entidades fuertes, entidades débiles, relaciones con cardinalidad 1:1, 1:M, M:1 y M:M, de igual manera para la transformación de atributos compuestos, atributos multivaluados, entre otros, durante este proceso se definieron las restricciones de llave primaria y foránea, así como los tipos de dato y extensión en algunos casos, de los elementos que conformaron a las tablas finales, seguido de esto se hizo uso de la herramienta pgModeler para la construcción del modelo relacional correspondiente a la base de datos, posteriormente por medio de SQL se crearon todas las tablas correspondientes, se insertaron todos los datos iniciales en ellas y por medio de PgSQL se crearon las funciones, triggers y demás objetos para cumplir con cada uno de los puntos solicitados, lo cual no fue una tarea del todo fácil ya que existen diversas formas de dar solución y se trató de buscar la que mejor se adaptara a la base de datos definida, por ultimo se trabajó sobre la creación de una interfaz gráfica vía aplicación

web que permite realizar ciertas funciones, esto último resultó ser algo laborioso ya que depende bastante de los conocimientos y el dominio que se tenga para su desarrollo, en general uno de los aciertos durante el desarrollo del trabajo fue el plan de trabajo definido ya que se estableció en base a los conocimientos y habilidades de cada integrante del equipo, facilitando con esto tener un mejor avance en el desarrollo del trabajo, incluso permitiendo obtener conocimientos de todos los integrantes sobre las formas de trabajo y solución a los requerimientos en una base de datos, por otro lado una dificultad que se presentó en el desarrollo es referente al control de cambios ya que en algún punto se llegaron a tener múltiples versiones de un mismo archivo y que en conjunto llegaron a crear confusión sobre cuál de ellos correspondía a la versión más reciente, sin embargo con todo lo anterior se logró cumplir de buena forma con el objetivo del presente trabajo, proponiendo una solución a los requerimientos definidos inicialmente, aplicando la gran mayoría de conceptos estudiados en el curso.

Vazquez Zavala Oliver Alexis

Con la realización del presente trabajo se ha permitido poner en práctica todos los conocimientos estudiados a lo largo del curso, tales como, análisis de requerimientos, modelo entidad relación, modelo entidad relación extendido, representación intermedia, modelo relacional, lenguaje SQL, DDL, DML, DCL, PgSQL, entre otros, e incluso permitió conocer la forma de crear una interfaz gráfica mediante una aplicación web la cual se conecta e interactúa con la base de datos, para la elaboración de este proyecto se trabajaron sobre todas las etapas de diseño de una base de datos iniciando por el análisis de requerimientos, diseño conceptual, diseño lógico y diseño físico, para esto en un principio se planteó el modelo entidad relación extendido en base a los requerimientos solicitados para la base de datos del restaurant, posteriormente se desarrolló la representación intermedia, en donde se aplicaron todas las reglas de transformación estudiadas para todos los elementos del MRE, como lo son la transformación de entidades fuertes, entidades débiles, relaciones con cardinalidad 1:1, 1:M, M:1 y M:M, de igual manera para la transformación de atributos compuestos, atributos multivaluados, entre otros, durante este proceso se definieron las restricciones de llave primaria y foránea, así como los tipos de dato y extensión en algunos casos, de los elementos que conformaron a las tablas finales, seguido de esto se hizo uso de la herramienta pgModeler para la construcción del modelo relacional correspondiente a la base de datos, posteriormente por medio de SQL se crearon todas las tablas correspondientes, se insertaron todos los datos iniciales en ellas y por medio de PgSQL se crearon las funciones, triggers y demás objetos para cumplir con cada uno de los puntos solicitados, lo cual no fue una tarea del todo fácil ya que existen diversas formas de dar solución y se trató de buscar la que mejor se adaptara a la base de datos definida, por ultimo se trabajó sobre la creación de una interfaz gráfica vía aplicación web que permite realizar ciertas funciones, esto último resultó ser algo laborioso ya que depende bastante de los conocimientos y el dominio que se tenga para su desarrollo, en general uno de los aciertos durante el desarrollo del trabajo fue el plan de trabajo definido ya que se estableció en base a los conocimientos y habilidades de cada integrante del equipo, facilitando con esto tener un mejor avance en el desarrollo del trabajo, incluso permitiendo obtener conocimientos de todos los integrantes sobre las formas de trabajo y solución a los requerimientos en una base de datos, por otro lado una dificultad que se presentó en el desarrollo es referente al control de cambios ya que en algún punto se llegaron a tener múltiples versiones de un mismo archivo y que en conjunto llegaron a crear confusión sobre cuál de ellos correspondía a la versión más reciente, sin embargo con todo lo anterior se logró cumplir de buena forma con el objetivo del presente trabajo, proponiendo una solución a los requerimientos definidos inicialmente, aplicando la gran

mayoría de conceptos estudiados en el curso.