Fundamentos de Bases de Datos
Grupo 7077

Proyecto. Taco Riendo

EQUIPO: mmm xd
317088296 - Demian Oswaldo Garcia Toxqui.
317042522 - Erick Bernal Márquez.
317180321 - Karen Cristóbal Morales.
317061521 - Rosa María Robles Huerta.
317205776 - Rubén Acosta Arzate.

15 de junio de 2022

1. MODELO ENTIDAD RELACIÓN

Caso de uso

1.1 Captura de requerimientos

- La carta de sus locales incluye **tacos, burritos, quesadillas, gringas, tortas, platillos mexicanos, salsas** y **bebidas** (se adjuntan algunos ejemplos de menús de taquerías, para que se den alguna idea), los precios se suelen colocar con etiquetas fluorescentes y en caso de que deban ser actualizados, se pega una nueva etiqueta encima. A don Pepe le interesa poder llevar un **registro histórico** de los precios de todos sus productos y de los insumos de los mismos, ya que le interesa saber en qué momento puede sacar alguna promoción.
- Se desea poder otorgar **tickets por cada consumo** (puede ser por comensal o mesa), el cual debe tener la siguiente información: **fecha**, **sucursal**, **mesero que atendió**, **detalle de los productos solicitados** con el subtotal en cada caso y el **total del consumo**.
- Se necesita identificar la **cantidad aproximada** de **porciones de ingredientes** necesaria para **cada una de las preparaciones** que se realizan en la taquería, esto con el fin de **minimizar** los desperdicios generados (los datos no tienen que ser reales, ten en cuenta que es solo una simulación, solo trata que tengan algún sentido.
- Se desea crear un programa llamado **Taquero de corazón**, el cual otorgará el **10% del consumo en puntos**, lo cuales podrán ser intercambiados por productos **dentro de la taquería**, cada punto equivale a **un peso**.
- De igual forma, suele manejar promociones como: **"Lunes de 2x1 en tacos al pastor"**, miércoles de **"Taco Chinito"** en el que **todos los tacos** cuestan **$10.00** o **"Viernes de tortuga"** (2x1 en tortas).
- La taquería ofrece **servicio a domicilio** por lo que se necesitarán los datos de los clientes (nombre completo, CURP, dirección, teléfono, correo electrónico); con este servicio, los clientes podrán pagar solo con **efectivo o tarjeta de crédito/débito**.
- Se desea tener un **control de pago a empleados** a través del pago **vía nómina** y **otorgar seguridad social**; para lograr esto, se requiere la información personal de los empleados (nombre completo, dirección, RFC, edad, número de seguridad social, antigüedad, dirección, teléfono y correo electrónico). Adicionalmente se busca crear un **programa de bonos para empleados**, con el que cada empleado que cumpla **2 años** laborando en la taquería se le agregará a su pago de nómina **$1,500.00**. Los clientes también pueden ser alguno de los empleados, en cuyo caso, el **servicio será gratuito**.
- Se debe considerar una solución que pueda ser escalable para poder eventualmente controlar sucursales, ya que don Pepe tiene pensando abrir al **menos una en los próximos meses** y posteriormente, abrir **una en cada alcaldía de la CDMX** y más adelante, **una en cada uno de los estados** de la República Mexicana.
- Derivado del punto anterior, **los empleados sólo podrán estar ligados a una sucursal**; en el caso de los clientes, **no se debe duplicar** su información en la base de datos.
- Se busca tener un **control de inventarios** especificando el **día de compra**, **fecha de caducidad**, **cantidad**, **marca del producto adquirido**, **precio de compra** (ingredientes, mesas, sillas, bancos, platos, servilleteros, etcétera).
- También interesa almacenar **información de los proveedores**, ya que en un futuro interesará hacer alianzas comerciales, a fin de obtener precios más competitivos.
- Un rubro importante es lo relacionado a las **salsas**, ya que las de este lugar son muy famosas y se desea presentarlas en una sección detallando: **ingredientes**, **nivel de picor** (dulce, bajo, medio, alto, extremo) y **recomendación de platillo** a acompañar. De igual forma, derivado del éxito de sus salsas, los clientes pueden obtenerlas para sus eventos, **las presentaciones que se tienen son**: frasco de 250 mg., de medio kilo o bien por litro.
- Es posible que un cliente **no desee dar sus datos** (por cuestiones del aviso de privacidad), en este caso, no se le niega el servicio y se debe mantener una imagen que llamaremos **"cliente default"** al cual se ligará el servicio. Este cliente existe por cada sucursal. Los datos en este caso son: nombre completo (**sucursal**), CURP (dejaremos el **RFC de la taquería**), dirección, teléfono, correo electrónico (**estos son los que corresponden a la taquería**). Este tipo de cliente no puede hacer pedido a domicilio.
- Existen **5 tipos de empleados**: parrilleros, taqueros, meseros, cajeros, tortilleros, repartidores.
- En el caso de los **repartidores**, además de la información definida para los empleados, se requiere del **número de licencia** (si aplica) y si cuenta o no con transporte (**motocicleta** o **bicicleta**). En el caso de contar con transporte, se requieren los siguientes datos: **marca**, **modelo**, **tipo** (motocicleta/bicicleta).
- Por cuestiones de mercadotecnia, se necesita saber las **distintas formas** en que los clientes pagan los servicios: **efectivo**, **tarjeta de crédito**, o **puntos** etc.

1.2 Identificación de Entidades, Atributos y Relaciones

- **ENTIDADES Y SUS ATRIBUTOS**

1. **Insumo:** idInsumo, nombre, registroPrecioInsumo(precio, fechaCambio).
2. **Producto:** idProducto, nombre, descripcion, registroPrecioProducto(precio, fechaCambio), precioActual.
3. **Bebida:** Los mismos que producto añadiendo cantidad y sabor.
4. **Salsa:** Los mismos que producto añadiendo nivelPicor, presentacion.
5. **Comida:** Los mismos que producto.
6. **Promocion:** idPromocion, diaPromo, descripcion, descuento, nombre.
7. **Proveedor:** rfc, marca, telefono, nombreC (nombre, paterno, materno), direccion (calle, numero, estado, cp), email.
8. **Inventario:** idInventario.
9. **Sucursal:** rfc, nombre, telefono, email, direccion(calle, numero, estado, cp, alcaldia).
10. **Cliente:** curp, telefono, nombreC (nombre, paterno, materno), direccion (calle, numero, estado, cp), email.
11. **Empleado:** Los mismos que Cliente añadiendo antiguedad, numSegSocial, edad, nacimiento, rfc, salario.
12. **Parrillero:** Los mismos que Empleado.
13. **Taquero:** Los mismos que Empleado.
14. **Mesero:** Los mismos que Empleado.
15. **Repartidor:** Los mismos que Empleado añadiendo numLicencia.
16. **Tortillero:** Los mismos que Empleado.
17. **Cajero:** Los mismos que Empleado.
18. **Transporte:** idTransporte, tipo, marca, modelo.
19. **MetodoPago:** idPago, cantidad.
20. **Efectivo:** Los mismos que MetodoPago.
21. **Puntos:** Los mismos que MetodoPago.
22. **Credito:** Los mismos que MetodoPago añadiendo nombreTitular.
23. **Debito:** Los mismos que Credito.
24. **Ticket:** idTicket, nombreSucursal, fecha, cuentaTotal, nombreCliente, nombreMesero.

- **RELACIONES**

1. Insumo con Producto a través de la relación **PreparарProduc.**
   Los productos pueden tener más de un insumo en su preparación y varios insumos podrían ser parte de varios productos (cardinalidad muchos a muchos). También tenemos que no todos los insumos se necesitan para preparar un producto, pero un producto necesita de insumos para ser preparado (participación total de lado de Producto y parcial de lado de Insumo).

2. Proveedor con Insumo a través de la relación **Proveer.**
   Los insumos pueden ser provistos por varios proveedores y un proveedor puede proveer mas de un insumo (muchos a muchos). También tenemos que un Insumo no necesariamente es provisto por un Proveedor (se puede comprar limones en la verdulería de la esquina), también no es necesario que un proveedor provee todo tipo de insumo (participación parcial en ambos lados).

3. Inventario con Insumo a través de la relación **TenerIns.**
   Los Inventarios pueden tener varios Insumos y varios Insumos pueden pertenecer a varios Inventarios (cardinalidad muchos a muchos). También no es necesario que un inventario deba de tener a todos los insumos y no es necesario que todos los insumos deben de pertecencer a un inventario (participacion parcial de ambos lados).

4. Comida con Promocion a través de la relación **TenerPromo.**
   Una comida puede tener varias promociones y una promoción puede ser aplicada a varias comidas (cardinalidad muchos a muchos). También tenemos que no toda comida aplica promociones, pero la existencia de una promoción depende de la existencia de una comida (participación parcial de lado de Comida y total de lado de Promocion).

5. Salsa con Comida a través de la relación **Recomendar.**
   Las salsas pueden ser recomendadas para acompañar varios tipos de comida y a varios tipos de comida se les pueden recomendar varias salsas distintas (cardinalidad muchos a muchos). También puede haber un platillo al que no se le recomiende una salsa en particular y no todas las salsas son recomendadas a un platillo (participación parcial de ambos lados).

6. Inventario con Sucursal a través de la relación **TenerInv.**
   Una sucursal tiene un inventario y un inventario pertenece a una sucursal (cardinalidad uno a uno). También es necesario que una sucursal tenga un inventario, pero no es necesario que un inventario específico sea parte de una sucursal ya que va cambiando (participación total de lado de Sucursal y parcial de lado de Inventario).

7. Cliente con MetodoPago a través de la relación **TenerPa.**
   Un cliente puede tener varios métodos de pago y varios métodos de pago pueden ser de un cliente (cardinalidad uno a muchos). También es necesario que un cliente tenga al menos un método de pago para poder pagar lo que consume en la sucursal, pero no es necesario que todos los métodos de pago deba de tenerlos un sólo cliente (participacion total de lado de Cliente y parcial de lado de MetodoPago).

8. Repartidor con Transporte a través de la relación **TenerTr.**
   Un repartidor podría tener dos medios de transporte y varios medios de transportes pueden ser tenidos por un repartidor (cardinalidad uno a muchos). También no es necesario que los repartidores cuenten con un transporte pero sí es necesario que un transporte lo tenga un repartidor (participación parcial de lado de Repartidar y total de lado de Transporte).

9. Producto con Ticket a través de la relación **Registrar.**
   Los productos pueden ser registrado en varios tickets y en varios tickets pueden ser registrados varios productos (cardinalidad muchos a muchos). También no todos los productos tienen que estar registrados en un ticket y no en todos los tickets se van a registrar todos los productos (participacion parcial de ambos lados).

10. Sucursal con Ticket a través de la relación **Generar.**
    En una Sucursal se general varios tickets y varios ticket son generados en una Sucursal (cardinalidad uno a muchos). También no es necesario que una Sucursal genere tickets (ya que la Sucursal podría ser nueva), y no es necesario que los tickets sean generados por una sola Sucursal ya que podría haber más sucursales (participación parcial de ambos lados).

11. Sucursal con Empleado a través de la relación **Trabajar.**
    En una Sucursal trabajan varios empleados y varios empleados trabajan en una sola sucur-

sal (cardinalidad uno a muchos). También no es necesario que en una Sucursal trabajen empleados ya que la Sucursal podría ser nueva, por lo anterior entonces no es necesario que haya empleados trabajando en la sucursal (participación parcial de ambos lados).

12. Ticket con Mesero a través de la relación **Atender.**
    Un mesero puede atender varios tickets de clientes y varios tickets de clientes pueden ser atendidos por un mismo mesero (cardinalidad uno a muchos). También no es necesario que todos los tickets sean atendidos por un mesero específico y no es necesario que un mesero atienda todos los tickets (participación parcial en ambos lados).

13. Ticket con Cliente a través de la relación **Dar.**
    A un cliente se le pueden dar varios tickets y varios tickes pueden ser dados a un mismo cliente (cardinalidad uno a muchos). También no es necesario que todos los tickets sean dados a un cliente específico y no es necesario que a un cliente se le den todos los tickets (participación parcial en ambos lados).

1.3 Consideraciones, decisiones y justificaciones.

Como consideraciones obtuvimos lo siguiente

1. Especificación y creación de id's (para entidades que no especifican un identificador):
   - La llave primaria de un Proveedor es su rfc.
   - La llave primaria de un Insumo es idInsumo.
   - La llave primaria de un Inventario es idInventario.
   - La llave primaria de una Sucursal es su rfc.
   - La llave primaria de un Producto es idProducto.
   - La llave primaria de una Promocion es idPromocion.
   - La llave primaria de un Ticket es idTicket.
   - La llave primaria de un Cliente es su curp.
   - La llave primaria de un Empleado es su curp.

   Con esto aseguramos que estas entidades tengan una llave para su posible acceso.

2. En la entidad **Sucursal** agregamos el atributo "alcaldia" debido a que se nos menciona que el dueño quiere expandir sus sucursales.

3. En la entidad **Bebida** agregamos los atributos "cantidad" (el cual estará representada en mililitros) y "sabor" el cual indica el sabor de la bebida.

4. En la entidad **Promocion** agregamos los atributos "diaPromo" (el cual hace referencia al día en que aplica una promoción) y "descripción" el cual es una pequeña descripción de la promoción.

5. En la entidad **Ticket** el atributo "cuentaTotal" es calculado. Obtendremos su valor al sumar el precio de los productos que consumió un cliente.

6. En la entidad **Empleado** el atributo "edad" es calculado. Obtendremos su valor al obtener la fecha de nacimiento del empleado, por lo tanto añadímos este atributo a la entidad.

7. En la entidad **Proveedor** agregamos un atributo llamado "marca" el cual hace referencia a la empresa donde trabaja el proveedor.

8. Debido a que se nos pide un registro histórico de los precios de los Insumos, agregamos un atributo multivaluado a la entidad **Insumo**, este atributo además está compuesto por los atributos "precio"

y “fechaCambio” los cuales nos ayudarán a guardar la información sobre la actualización de un precio en los insumos.

9. Debido a que se nos pide un registro histórico de los precios de los Productos, agregamos un atributo multivaluado a la entidad **Producto**, este atributo además está compuesto por los atributos “precio” y “fechaCambio” los cuales nos ayudarán a guardar la información sobre la actualización de un precio en los productos.

10. En una sucursal los transportes tienen un id único, sin embargo puede ocurrir que en dos diferentes sucursales haya transportes con el mismo id, es por eso que decidimos hacer a la entidad **Transporte** débil ya que no puede ser unívocamente identificada por sus atributos y por ello requiere del id del repartidor.

11. Para entidades que tuvieran atributos en común decidimos agruparlas mediante el uso de herencia creando una superentidad en casos como:

    - Producto hereda a Bebida, Salsa y Comida.
      Creamos a la superentidad Producto debido a que las entidades Bebida, Salsa y Comida tienen atributos en común. En la herencia decidimos usar especialización total para obligar a que sólo haya bebidas, salsas y comida (y no productos en sí). También usamos disyunción para que un producto NO pueda ser bebida, salsa y comida al mismo tiempo (sólo debe ser una cosa).

    - Cliente hereda a Empleado.
      Hicimos superentidad a Cliente debido a que la entidad Empleado tiene varios atributos en común. En la herencia decidimos usar especialización parcial debido a que podemos tener Entidades de ambos tipos (clientes y empleados). También usamos traslape ya que un cliente podría ser también un empleado.

    - Empleado hereda a Parrillero, Taquero, Mesero, Repartidor, Tortillero y Cajero.
      Creamos a la superentidad Empleado para una mejor organización debido a que las entidades Parrillero, Taquero, Mesero, Repartidor, Tortillero y Cajero tienen atributos en común. En la herencia decidimos usar especialización total para obligar a que sólo haya parrilleros, taqueros, meseros, repartidores, tortilleros y cajeros (y no empleados en sí). También usamos disyunción ya que un empleado no puede hacer varias cosas a la vez (por ejemplo ser repartidor y cocinero), sólo debe de ser un tipo de empleado el cual realizará una sóla tarea.

    - MetodoPago hereda a Efectivo, Puntos, Credito y Debito.
      Creamos a la superentidad MetodoPago para tener una mejor organización entre los distintos métodos de pago que puede tener un cliente. En la herencia decidimos usar especialización parcial porque en un futuro podría haber otro tipo de método de pago. También usamos traslape ya que un cliente podría pagar con más de un método de pago.

12. Respecto a ciertas relaciones:

    - En la relación **TenerIns** entre las entidades Inventario e Insumo, se nos hizo más conveniente almacenar en la relación los atributos que nos ayudarán a llevar el registro de los Insumos, esto debido a que podría haber muchos inventarios en los que se tendrán muchos Insumos, por lo que quisiéramos tener esta información en una tabla aparte.

    - En la relación **PrepararProduc** entre las entidades Insumo y Producto, se decidió colocar el atributo “cantidad” porque es más conveniente tenerlo en la relación a tenerlo en alguna de las entidades de la relación.

    - En la relación **Registrar** entre las entidades Ticket y Producto, se decidió colocar el atributo “nombre” y “precioActual” para que sea más fácil acceder al nombre y precio del producto consumido.

    - En la relación **Trabajar** entre las entidades Sucursal y Empleado, se decidió colocar el atributo “fechaInicio” porque es más conveniente tenerlo en la relación a tenerlo en alguna de las

entidades de la relación. Este atributo guardará la fecha en la que un empleado comenzó a trabajar en la sucursal, esto nos ayudará a calcular el atributo “antiguedad” de la entidad Empleado.

2. TRADUCCIÓN AL MODELO RELACIONAL.

2.1 Restricciones y especificaciones de los dominios de los atributos. Llaves foráneas, compuestas y primarias.

***NOTA:** Para llaves primarias usamos el **color rojo**, para llaves foraneas el **color naranja**, para el resto de atributos el negro.

Proveedor

- **rfc**: Cadena de longitud 13 que se compone de caracteres alfanuméricos.
- nombre: Cadena de máximo 50 caracteres.
- paterno: Cadena de máximo 50 caracteres.
- materno: Cadena de máximo 50 caracteres.
- marca: Cadena de máximo 50 caracteres.
- telefono: Cadena de 10 dígitos.
- calle: Cadena de máximo 50 caracteres.
- numero: Cadena de 4 dígitos.
- estado: Cadena de máximo 50 caracteres.
- cp: Cadena de 5 dígitos.
- email: Cadena de máximo 50 caracteres.

Insumo

- **idInsumo**: Cadena de longitud 10 donde el primer caracter es 'I', los siguientes 4 caracteres son los primeros 4 caracteres del nombre del Insumo y los 5 caracteres restantes son dígitos.
- nombre: Cadena de máximo 50 caracteres.

Salsa

- **idSalsa**: Cadena de longitud 10 donde el primer caracter es 'S', los siguientes 4 caracteres son letras y los 5 caracteres restantes son dígitos.
- nombre: Cadena de máximo 50 caracteres.
- descripcion: Cadena de máximo 100 caracteres.
- presentacion: Cadena de máximo 50 caracteres.
- nivelPicor: Los niveles de picor están representados por un entero del 1 al 5.
- precioActual: Float no negativo y de longitud variable.

Comida

- **idComida**: Cadena de longitud 10 donde el primer caracter es 'C', los siguientes 4 caracteres son letras y los 5 caracteres restantes son dígitos.
- nombre: Cadena de máximo 50 caracteres.
- descripcion: Cadena de máximo 100 caracteres.

- precioActual: Float no negativo y de longitud variable.

Bebida

- **idBebida**: Cadena de longitud 10 donde el primer caracter es 'B', los siguientes 4 caracteres son letras y los 5 caracteres restantes son dígitos.
- nombre: Cadena de máximo 50 caracteres.
- descripcion: Cadena de máximo 100 caracteres.
- sabor: Cadena de máximo 20 caracteres.
- cantidadML: Es un número entero positivo.
- precioActual: Float no negativo y de longitud variable.

Inventario

- **idInventario**: Cadena de longitud 10 donde los primeros dos caracteres son 'IN' y los 8 caracteres restantes son dígitos.

Cliente

- **curp**: Cadena de longitud 18 que se compone de caracteres alfanuméricos.
- nombre: Cadena de máximo 50 caracteres.
- paterno: Cadena de máximo 50 caracteres.
- materno: Cadena de máximo 50 caracteres.
- telefono: Cadena de 10 dígitos.
- calle: Cadena de máximo 50 caracteres.
- numero: Cadena de 4 dígitos.
- estado: Cadena de máximo 50 caracteres.
- cp: Cadena de 5 dígitos.
- email: Cadena de máximo 50 caracteres.
- rfc: Cadena de longitud 13 que se compone de caracteres alfanuméricos.
- nSS: Cadena de longitud 11 que se compone de dígitos.
- nacimiento: Fecha con formato DD/MM/AAAA, donde DD es un entero de dos dıgitos del 01 al 31, MM es un entero de dos dıgitos del 01 al 12, AAAA es un entero de cuatro dıgitos de 1950 a 2004.
- salario: Float no negativo y de longitud variable que indica la cantidad que se le pagará.
- fechaInicio: Fecha con formato DD/MM/AAAA, donde DD es un entero de dos dıgitos del 01 al 31, MM es un entero de dos dıgitos del 01 al 12, AAAA es un entero de cuatro dıgitos de 2000 a 2022.
- esEmpleado: Booleano con valores 'true' y 'false'.

RegistroPrecioSalsa

- **idSalsa, registroPrecioSalsa**: Cada atributo que compone a la llave es una cadena de alfanumérica de longitud 10.
- precio: Float no negativo de longitud variable.
- fechaDeCambio: Fecha con formato DD/MM/AAAA, donde DD es un entero de dos dıgitos del 01 al 31, MM es un entero de dos dıgitos del 01 al 12, AAAA es un entero de cuatro dıgitos de 2000 a 2022.

RegistroPrecioComida

- **idComida, registroPrecioComida**: Cada atributo que compone a la llave es una cadena de alfanumérica de longitud 10.
- precio: Float no negativo de longitud variable.
- fechaDeCambio: Fecha con formato DD/MM/AAAA, donde DD es un entero de dos dıgitos del 01 al 31, MM es un entero de dos dıgitos del 01 al 12, AAAA es un entero de cuatro dıgitos de 2000 a 2022.

RegistroPrecioBebida

- **idBebida, registroPrecioBebida**: Cada atributo que compone a la llave es una cadena de alfanumérica de longitud 10.
- precio: Float no negativo de longitud variable.
- fechaDeCambio: Fecha con formato DD/MM/AAAA, donde DD es un entero de dos dıgitos del 01 al 31, MM es un entero de dos dıgitos del 01 al 12, AAAA es un entero de cuatro dıgitos de 2000 a 2022.

RegistroPrecioInsumo

- **idInsumo, registroPrecioInsumo**: Cada atributo que compone a la llave es una cadena de alfanumérica de longitud 10.
- precio: Float no negativo de longitud variable.
- fechaDeCambio: Fecha con formato DD/MM/AAAA, donde DD es un entero de dos dıgitos del 01 al 31, MM es un entero de dos dıgitos del 01 al 12, AAAA es un entero de cuatro dıgitos de 2000 a 2022.

Transporte

- **curp, idTransporte**: curp hace referencia a la llave primaria del repartidor, idTransporte es una cadena de longitud 10.
- tipo: Cadena de máximo 50 caracteres.
- marca: Cadena de máximo 50 caracteres.
- modelo: Cadena de máximo 50 caracteres.

Promocion

- **idPromocion**: Cadena de longitud 10 donde los dos primeros caracteres son 'PR' y los 8 caracteres restantes son dígitos.

- nombre: Cadena de máximo 50 caracteres.
- descuento: Float no negativo de longitud variable.
- diaPromo: Cadena de máximo 10 caracteres.
- descripcion: Cadena de máximo 50 caracteres.

Ticket

- **idTicket**: Cadena de longitud 10 donde los primeros dos caracteres son 'TI' y los 8 caracteres restantes son dígitos.
- **rfc**: hace referncia a la llave primaria de la sucursal.
- **curpCliente**: hace referncia a la llave primaria del cliente.
- **curpMesero**: hace referncia a la llave primaria del mesero.
- nombreSucursal: Cadena de máximo 50 caracteres.
- nombreCliente: Cadena de máximo 50 caracteres.
- nombreMesero: Cadena de máximo 50 caracteres.
- fecha: Fecha con formato DD/MM/AAAA, donde DD es un entero de dos dıgitos del 01 al 31, MM es un entero de dos dıgitos del 01 al 12, AAAA es un entero de cuatro dıgitos de 2000 a 2022.

Sucursal

- **rfc**: Cadena de longitud 13 que se compone de caracteres alfanuméricos.
- **idInventario**: Hace referenia a la llave primaria de un Inventario.
- nombre: Cadena de máximo 50 caracteres.
- calle: Cadena de máximo 50 caracteres.
- alcaldia: Cadena de máximo 50 caracteres.
- cp: Cadena de 5 dígitos.
- estado: Cadena de máximo 50 caracteres.
- numero: Cadena de 4 dígitos.
- email: Cadena de máximo 50 caracteres.
- telefono: Cadena de 10 dígitos.

MetodoDePago

- **idPago**: Cadena de longitud 10 que se compone de caracteres alfanuméricos.
- **curp**: Hace referenia a la llave primaria de un Cliente.
- cantidad: Float no negativo de longitud variable.
- nombreTitular: Cadena de máximo 50 caracteres.
- esEfectivo: Booleano con valores 'true' y 'false'.
- esPuntos: Booleano con valores 'true' y 'false'.
- esDebito: Booleano con valores 'true' y 'false'.

- esEmpleado: Booleano con valores 'true' y 'false'.

Proveer

- **idInsumo**: Hace referenia a la llave primaria de un Insumo.
- **rfc**: Hace referenia a la llave primaria de un Proveedor.
- nombre: Cadena de máximo 50 caracteres.

TenerInsumo

- **idInventario**: Hace referenia a la llave primaria de un Inventario.
- **idInsumo**: Hace referenia a la llave primaria de un Insumo.
- nombre: Cadena de máximo 50 caracteres.
- precio: Float no negativo de longitud variable.
- cantidad: Entero no negativo.
- caducidad: Fecha con formato DD/MM/AAAA mayor a 01/01/2022.
- diaCompra: Fecha con formato DD/MM/AAAA, donde DD es un entero de dos dıgitos del 01 al 31, MM es un entero de dos dıgitos del 01 al 12, AAAA es un entero de cuatro dıgitos de 2000 a 2022.
- marcaProducto: Cadena de máximo 50 caracteres.

PrepararSalsa

- **idInsumo**: Hace referenia a la llave primaria de un Insumo.
- **idSalsa**: Hace referenia a la llave primaria de una Salsa.
- cantidad: Entero no negativo.
- precioActual: Float no negativo y de longitud variable.

PrepararComida

- **idInsumo**: Hace referenia a la llave primaria de un Insumo.
- **idComida**: Hace referenia a la llave primaria de una Comida.
- cantidad: Entero no negativo.
- precioActual: Float no negativo y de longitud variable.

PrepararBebida

- **idInsumo**: Hace referenia a la llave primaria de un Insumo.
- **idBebida**: Hace referenia a la llave primaria de una Bebida.
- cantidad: Entero no negativo.
- precioActual: Float no negativo y de longitud variable.

RegistrarSalsa

- **idTicket**: Hace referenia a la llave primaria de un Ticket.
- **idSalsa**: Hace referenia a la llave primaria de una Salsa.
- nombre: Cadena de máximo 50 caracteres.

RegistrarComida

- **idTicket**: Hace referenia a la llave primaria de un Ticket.
- **idComida**: Hace referenia a la llave primaria de una Comida.
- nombre: Cadena de máximo 50 caracteres.

RegistrarBebida

- **idTicket**: Hace referenia a la llave primaria de un Ticket.
- **idBebida**: Hace referenia a la llave primaria de una Bebida.
- nombre: Cadena de máximo 50 caracteres.

Recomendar

- **idSalsa**: Hace referenia a la llave primaria de una Salsa.
- **idComida**: Hace referenia a la llave primaria de una Comida.

TenerPromo

- **idPromocion**: Hace referenia a la llave primaria de una Promocion.
- **idComida**: Hace referenia a la llave primaria de una Comida.

Mesero

- **curp**: Hace referenia a la llave primaria de un Cliente.
- **rfc**: Hace referenia a la llave primaria de una Sucursal.

Parrillero

- **curp**: Hace referenia a la llave primaria de un Cliente.
- **rfc**: Hace referenia a la llave primaria de una Sucursal.

Taquero

- **curp**: Hace referenia a la llave primaria de un Cliente.
- **rfc**: Hace referenia a la llave primaria de una Sucursal.

Repartidor

- **curp**: Hace referenia a la llave primaria de un Cliente.
- **rfc**: Hace referenia a la llave primaria de una Sucursal.

- numLicencia: cadena de longitud 10 cuyos caracteres son dígitos.
- transporte: Es una cadena de máximo 50 caracteres la cual especifica el transporte que usa el repartidor para hacer las entregas.

Tortillero

- **curp**: Hace referenia a la llave primaria de un Cliente.
- **rfc**: Hace referenia a la llave primaria de una Sucursal.

Cajero

- **curp**: Hace referenia a la llave primaria de un Cliente.
- **rfc**: Hace referenia a la llave primaria de una Sucursal.

3. CONSTRUCCIÓN DE LA BASE DE DATOS

3.1 DDL (Data Definition Language)

Consideraciones al momento de escoger el tipo de dato y las restricciones de dominio:

Tabla Proveedor

```
CREATE TABLE Proveedor(
        rfc CHAR(13) NOT NULL CHECK(rfc <> '') UNIQUE,
        nombre VARCHAR(50) NOT NULL CHECK(nombre <> ''),
        paterno VARCHAR(50) NOT NULL CHECK(paterno <> ''),
        materno VARCHAR(50) NOT NULL CHECK(materno <> ''),
        marca VARCHAR(50) NOT NULL CHECK(marca <> ''),
        telefono CHAR(10) NOT NULL CHECK(telefono SIMILAR TO '[0-9]{10}'),
        calle VARCHAR(50) NOT NULL CHECK(calle <> ''),
        numero CHAR(4) NOT NULL CHECK(numero SIMILAR TO '[0-9]{4}'),
        estado VARCHAR NOT NULL CHECK(estado <> ''),
        cp CHAR(5) NOT NULL CHECK(cp SIMILAR TO '[0-9]{5}'),
        email VARCHAR(50) NOT NULL CHECK(email <> '')
);
```

rfc: Es la llave primaria de la tabla Proveedor.

- No aceptamos valor null puesto que es la llave primaria.
- El rfc está obligado a tener una longitud de 13 caracteres.
- El rfc es único para cada proveedor, por ello lo limitamos a que fuera único.

nombre: Es el nombre del Proveedor.

- Debido a que son muy variados establecimos su tipo de dato como varchar donde cada nombre debe tener máximo 50 caracteres.
- El nombre no debe de ser una cadena vacía ya que no tendría sentido tener proveedores sin nombre, por esa misma razón no permitimos que tome el valor null.

paterno: Es el apellido paterno del Proveedor.

- Debido a que son muy variados establecimos su tipo de dato como varchar donde cada apellido debe tener máximo 50 caracteres.
- El apellido no debe de ser una cadena vacía y no se permite que tome el valor null.

maternoSup: Es el apellido materno del Proveedor.

- Debido a que son muy variados establecimos su tipo de dato como varchar donde cada apellido debe tener máximo 50 caracteres.
- El apellido no debe de ser una cadena vacía y no se permite que tome el valor null.

marca: Es la empresa donde trabaja el Proveedor.

- Debido a que los nombres de empresas son variados establecimos su tipo de dato como varchar donde se debe de tener máximo 50 caracteres.

- La marca no debe de ser una cadena vacía y no se permite que tome el valor null.

telefono: Es el número telefónico del Proveedor.

- El telefono está obligado a tener una logitud de 10 caracteres. Se verifica que estos caracteres sean dígitos.
- Deseamos que se especifique este dato por ello no permitimos que sea una cadena vacia o null.

calle: Es la calle donde vive el Proveedor.

- Debido a que las calles son muy variadas establecimos su tipo de dato como varchar donde se debe de tener máximo 50 caracteres.
- Deseamos que se especifique este dato por ello no permitimos que sea una cadena vacia o null.

numero: Es el número donde vive el Proveedor.

- El número está obligado a tener una logitud de 4 caracteres. Se verifica que estos caracteres sean dígitos.
- Deseamos que se especifique este dato por ello no permitimos que sea una cadena vacia o null.

estado: Es la Estado donde vive el Proveedor.

- Debido a que los Estados son muy variados establecimos su tipo de dato como varchar (no especificamos límite de caracteres).
- Deseamos que se especifique este dato por ello no permitimos que sea una cadena vacia o null.

cp: Es el código postal donde vive el Proveedor.

- El código postal está obligado a tener una logitud de 5 caracteres. Se verifica que estos caracteres sean dígitos.
- Deseamos que se especifique este dato por ello no permitimos que sea una cadena vacia o null.

email: Es el correo electrónico del Proveedor.

- Debido a que el email puede ser muy variado establecimos su tipo de dato como varchar donde debe tener máximo 50 caracteres.
- Deseamos que se especifique este dato por ello no permitimos que sea una cadena vacia o null.

Tabla Insumo

```
CREATE TABLE Insumo(
        idInsumo CHAR(10) NOT NULL CHECK(idInsumo <> '') UNIQUE,
        nombre VARCHAR(50) NOT NULL CHECK(nombre <> '')
);
```

idInsumo: Es la llave primaria de la tabla Insumo.

- No aceptamos valor null puesto que es la llave primaria.

- El id está obligado a tener una longitud de 10 caracteres.
- El id es único para cada insumo, por ello lo limitamos a que fuera único.

nombre: Es el nombre del Insumo.

- Debido a que son muy variados establecimos su tipo de dato como varchar donde cada nombre debe tener máximo 50 caracteres.
- Deseamos que se especifique este dato por ello no permitimos que sea una cadena vacia o null.

Tabla Salsa

```
CREATE TABLE Salsa(
        idSalsa CHAR(10) NOT NULL CHECK (idSalsa<>'') UNIQUE,
        nombre VARCHAR(50) NOT NULL CHECK(nombre <> ''),
        descripcion VARCHAR(100) NOT NULL CHECK(descripcion <> ''),
        presentacion VARCHAR(50) NOT NULL CHECK(presentacion <> ''),
        nivelPicor INT NOT NULL CHECK(nivelPicor >=1 AND nivelPicor <=5),
        precioActual FLOAT NOT NULL CHECK(precioActual > 0)
);
```

idSalsa: Es la llave primaria de la tabla Salsa.

- No aceptamos valor null puesto que es la llave primaria.
- El id está obligado a tener una longitud de 10 caracteres.
- El id es único para cada salsa, por ello lo limitamos a que fuera único.

nombre: Es el nombre de la Salsa.

- Debido a que son muy variados establecimos su tipo de dato como varchar donde cada nombre debe tener máximo 50 caracteres.
- Deseamos que se especifique este dato por ello no permitimos que sea una cadena vacia o null.

descripcion: Es la descripción de la Salsa.

- Debido a que la descripción suele ser variada establecimos su tipo de dato como varchar donde se debe de tener máximo 100 caracteres.
- La descripción no debe de ser una cadena vacía y no se permite que tome el valor null.

presentacion: Es la presentación de la Salsa (frasco, litros, etc).

- Debido a que la presentación suele ser variada establecimos su tipo de dato como varchar donde se debe de tener máximo 50 caracteres.
- Deseamos que se especifique este dato por ello no permitimos que sea una cadena vacia o null.

nivelPicor: Es el nivel de picor que tiene la Salsa.

- Este dato lo decidimos representar de tipo INT, se verifica que el nivel de picor sea un número entre 1 y 5.
- Deseamos que se especifique este dato por ello no permitimos que sea una cadena vacia o null.

precioActual: Es el precio actual de la Salsa.

- Este dato es de tipo FLOAT debido a que a veces el precio se maneja con centavos.
- Verificamos que el precio sea mayor a cero.
- Deseamos que se especifique este dato por ello no permitimos que sea una cadena vacia o null.

Tabla Comida

```
CREATE TABLE Comida(
        idComida CHAR(10) NOT NULL CHECK (idComida<>'') UNIQUE,
        nombre VARCHAR(50) NOT NULL CHECK(nombre <> ''),
        descripcion VARCHAR(100) NOT NULL CHECK(descripcion <> ''),
        precioActual FLOAT NOT NULL CHECK(precioActual > 0)
);
```

idComida: Es la llave primaria de la tabla Comida.

- No aceptamos valor null puesto que es la llave primaria.
- El id está obligado a tener una longitud de 10 caracteres.
- El id es único para cada comida, por ello lo limitamos a que fuera único.

nombre: Es el nombre de la Comida.

- Debido a que son muy variados establecimos su tipo de dato como varchar donde cada nombre debe tener máximo 50 caracteres.
- Deseamos que se especifique este dato por ello no permitimos que sea una cadena vacia o null.

descripcion: Es la descripción de la Comida.

- Debido a que la descripción suele ser variada establecimos su tipo de dato como varchar donde se debe de tener máximo 100 caracteres.
- La descripción no debe de ser una cadena vacía y no se permite que tome el valor null.

precioActual: Es el precio actual de la Comida.

- Este dato es de tipo FLOAT debido a que a veces el precio se maneja con centavos.
- Verificamos que el precio sea mayor a cero.
- Deseamos que se especifique este dato por ello no permitimos que sea una cadena vacia o null.

Tabla Bebida

```
CREATE TABLE Bebida(
        idBebida CHAR(10) NOT NULL CHECK (idBebida<>'') UNIQUE,
        nombre VARCHAR(50) NOT NULL CHECK(nombre <> ''),
        descripcion VARCHAR(100) NOT NULL CHECK(descripcion <> ''),
        sabor VARCHAR(20) NOT NULL CHECK(sabor <> ''),
        cantidadML VARCHAR(10) NOT NULL CHECK(cantidadML <> ''),
        precioActual FLOAT NOT NULL CHECK(precioActual > 0)
);
```

idBebida: Es la llave primaria de la tabla Bebida.

- No aceptamos valor null puesto que es la llave primaria.
- El id está obligado a tener una longitud de 10 caracteres.
- El id es único para cada bebida, por ello lo limitamos a que fuera único.

nombre: Es el nombre de la Bebida.

- Debido a que son muy variados establecimos su tipo de dato como varchar donde cada nombre debe tener máximo 50 caracteres.
- Deseamos que se especifique este dato por ello no permitimos que sea una cadena vacia o null.

descripcion: Es la descripción de la Bebida.

- Debido a que la descripción suele ser variada establecimos su tipo de dato como varchar donde se debe de tener máximo 100 caracteres.
- La descripción no debe de ser una cadena vacía y no se permite que tome el valor null.

sabor: Es el sabor que tendrá la Bebida.

- Debido a que son muy variados establecimos su tipo de dato como varchar donde cada sabor debe tener máximo 20 caracteres.
- Deseamos que se especifique este dato por ello no permitimos que sea una cadena vacia o null.

cantidadML: Es la cantidad en mililitros de la Bebida.

- Decidimos expresar este valor como una cadena la cual debe de tener máximo 20 caracteres.
- Deseamos que se especifique este dato por ello no permitimos que sea una cadena vacia o null.

precioActual: Es el precio actual de la Bebida.

- Este dato es de tipo FLOAT debido a que a veces el precio se maneja con centavos.
- Verificamos que el precio sea mayor a cero.
- Deseamos que se especifique este dato por ello no permitimos que sea una cadena vacia o null.

Tabla Cliente

```
CREATE TABLE Cliente(
        curp CHAR(18) NOT NULL CHECK(CHAR_LENGTH(curp) = 18) CHECK(curp SIMILAR TO '[A-Z]{4}[0-9]{6}[A-Z]{8}') UNIQUE,
        nombre VARCHAR(50) NOT NULL CHECK(nombre <> ''),
        paterno VARCHAR(50) NOT NULL CHECK(paterno <> ''),
        materno VARCHAR(50) NOT NULL CHECK(materno <> ''),
        telefono CHAR(10) NOT NULL CHECK(telefono SIMILAR TO '[0-9]{10}'),
        calle VARCHAR(50) NOT NULL CHECK(calle <> ''),
        numero CHAR(4) NOT NULL CHECK(numero SIMILAR TO '[0-9]{4}'),
        estado VARCHAR NOT NULL CHECK(estado <> ''),
        cp CHAR(5) NOT NULL CHECK(cp SIMILAR TO '[0-9]{5}'),
        email VARCHAR(50) NOT NULL CHECK(email <> ''),
        rfc CHAR(13) CHECK(rfc <> ''),
        nSS CHAR(11) CHECK(nSS SIMILAR TO '[0-9]{11}'),
        nacimiento DATE NOT NULL CHECK(nacimiento > '1950-01-01' AND nacimiento < '2004-01-01'),
        salario FLOAT CHECK(salario>0),
        fechaInicio DATE CHECK(fechaInicio > '2000-01-01' AND fechaInicio < '2022-01-01'),
        esEmpleado BOOLEAN NOT NULL

);
```

curp: Es la llave primaria de la tabla Cliente.

- No aceptamos valor null puesto que es la llave primaria.
- La curp está obligada a tener una longitud de 18 caracteres, donde verificamos que los primeros 4 caracteres sean letras, los siguientes 6 caracteres dígitos y los 8 caracteres restantes letras.
- La curp es única para cada cliente, por ello lo limitamos a que fuera única.

nombre: Es el nombre del Cliente.

- Debido a que son muy variados establecimos su tipo de dato como varchar donde cada nombre debe tener máximo 50 caracteres.
- El nombre no debe de ser una cadena vacía ya que no tendría sentido tener clientes sin nombre, por esa misma razón no permitimos que tome el valor null o vacio.

paterno: Es el apellido paterno del Cliente.

- Debido a que son muy variados establecimos su tipo de dato como varchar donde cada apellido debe tener máximo 50 caracteres.
- El apellido no debe de ser una cadena vacía y no se permite que tome el valor null.

maternoSup: Es el apellido materno del Cliente.

- Debido a que son muy variados establecimos su tipo de dato como varchar donde cada apellido debe tener máximo 50 caracteres.
- El apellido no debe de ser una cadena vacía y no se permite que tome el valor null.

telefono: Es el número telefónico del Cliente.

- El telefono está obligado a tener una logitud de 10 caracteres. Se verifica que estos caracteres sean dígitos.
- Deseamos que se especifique este dato por ello no permitimos que sea una cadena vacia o null.

calle: Es la calle donde vive el Cliente.

- Debido a que las calles son muy variadas establecimos su tipo de dato como varchar donde se debe de tener máximo 50 caracteres.
- Deseamos que se especifique este dato por ello no permitimos que sea una cadena vacia o null.

numero: Es el número donde vive el Cliente.

- El número está obligado a tener una logitud de 4 caracteres. Se verifica que estos caracteres sean dígitos.
- Deseamos que se especifique este dato por ello no permitimos que sea una cadena vacia o null.

estado: Es la Estado donde vive el Cliente.

- Debido a que los Estados son muy variados establecimos su tipo de dato como varchar (no especificamos límite de caracteres).
- Deseamos que se especifique este dato por ello no permitimos que sea una cadena vacia o null.

cp: Es el código postal donde vive el Cliente.

- El código postal está obligado a tener una logitud de 5 caracteres. Se verifica que estos caracteres sean dígitos.
- Deseamos que se especifique este dato por ello no permitimos que sea una cadena vacia o null.

email: Es el correo electrónico del Cliente.

- Debido a que el email puede ser muy variado establecimos su tipo de dato como varchar donde debe tener máximo 50 caracteres.
- Deseamos que se especifique este dato por ello no permitimos que sea una cadena vacia o null.

rfc: Es el RFC del Cliente/Empleado.

- El rfc está obligado a tener una longitud de 13 caracteres.

nSS: Es el número de seguridad social del Cliente/Empleado.

- Este dato está obligado a tener 11 caracteres, los cuales se verifica que sean dígitos.

nacimiento: Es la fecha de nacimiento del Cliente.

- Este dato es de tipo DATE, donde se verifica que las personas hayan nacido a partir del año de 1950 hasta el año 2004.
- Deseamos que se especifique este dato por ello no permitimos que sea una cadena vacia o null.

salario: Es el salario del Cliente/Empleado.

- Este dato es de tipo FLOAT debido a que un salario esta sujeto a impuestos, bonos, inflación, etc. y no siempre será un entero.
- El salario debe de ser mayor a cero.

fechaInicio: Es la fecha en la que empezó a trabajar el Cliente/Empleado.

- Este dato es de tipo DATE donde se verifica que el año esté entre 2000 y 2022.

esEmpleado: Nos dice si un cliente es un empleado.

- Este dato es de tipo Boolean, donde los posibles valores a tomar serán 'true' y 'false'.

- Deseamos que se especifique este dato por ello no permitimos que sea una cadena vacia o null.

Tabla Inventario

```
CREATE TABLE Inventario(
        idInventario CHAR(10) NOT NULL CHECK(idInventario <> '') UNIQUE
);
```

idInventario: Es la llave primaria de la tabla Inventario.

- No aceptamos valor null puesto que es la llave primaria.
- El id está obligado a tener una longitud de 10 caracteres.
- El id es único para cada inventario, por ello lo limitamos a que fuera único.

Tabla Promocion

```
CREATE TABLE Promocion(
        idPromocion CHAR(10) NOT NULL CHECK(idPromocion <> ''),
        nombre VARCHAR(50) NOT NULL CHECK(nombre <> ''),
        descuento FLOAT NOT NULL CHECK (descuento > 0),
        diaPromo VARCHAR(10) NOT NULL CHECK(diaPromo <> ''),
        descripcion VARCHAR(100) NOT NULL CHECK(descripcion <> '')
);
```

idPromocion: Es la llave primaria de la tabla Promocion.

- No aceptamos valor null puesto que es la llave primaria.
- El id está obligado a tener una longitud de 10 caracteres.
- El id es único para cada promoción, por ello lo limitamos a que fuera único.

nombre: Es el nombre de la Promocion.

- Debido a que son muy variados establecimos su tipo de dato como varchar donde cada nombre debe tener máximo 50 caracteres.
- Deseamos que se especifique este dato por ello no permitimos que sea una cadena vacia o null.

precioActual: Es el descuento que aplica la Promocion.

- Este dato es de tipo FLOAT debido a que a veces el descuento se maneja con centavos.
- Verificamos que el descuento sea mayor a cero.
- Deseamos que se especifique este dato por ello no permitimos que sea una cadena vacia o null.

diaPromo: Es el día cuando aplica la Promocion.

- Este dato representará los días de la semana, por ello usamos el dato Varchar donde la cadena pueden tener como máximo una longitud de 10 caracteres.
- Deseamos que se especifique este dato por ello no permitimos que sea una cadena vacia o null.

descripcion: Es la descripción de la Promocion.

- Debido a que la descripción suele ser variada establecimos su tipo de dato como varchar donde se debe de tener máximo 100 caracteres.
- La descripción no debe de ser una cadena vacía y no se permite que tome el valor null.

Tabla RegistroPrecioInsumo

```
CREATE TABLE RegistroPrecioInsumo(
        idInsumo CHAR(10) NOT NULL CHECK (idInsumo<>''),
        registroPrecioInsumo CHAR(10) NOT NULL CHECK (registroPrecioInsumo<>'') UNIQUE,
        precio FLOAT NOT NULL CHECK (precio > 0),
        fechaDeCambio DATE NOT NULL CHECK(fechaDeCambio > '2000-01-01' AND fechaDeCambio < '2022-01-01')
);
```

idInsumo: Es la llave primaria de la entidad "Insumo". Sus restricciones son las mismas que se definieron para "idInsumo" en la tabla **Insumo**.

registroPrecioInsumo: Es la llave primaria de la tabla RegistroPrecioInsumo.

- No aceptamos valor null puesto que es la llave primaria.
- El dato está obligado a tener una longitud de 10 caracteres.
- El dato es único para cada registro de insumo, por ello lo limitamos a que fuera único.

precio: Es el precio que tiene o tuvo el Insumo.

- Este dato es de tipo FLOAT debido a que a veces el precio se maneja con centavos.
- Verificamos que el precio sea mayor a cero.
- Deseamos que se especifique este dato por ello no permitimos que sea una cadena vacia o null.

fechaCambio: Es la fecha en la que se actualizó el precio de un Insumo.

- Este dato es de tipo DATE donde se verifica que el año esté entre 2000 y 2022.
- Deseamos que se especifique este dato por ello no permitimos que sea null.

Tabla RegistroPrecioSalsa

```
CREATE TABLE RegistroPrecioSalsa(
        idSalsa CHAR(10) NOT NULL CHECK (idSalsa<>''),
        registroPrecioSalsa CHAR(10) NOT NULL CHECK (registroPrecioSalsa<>''),
        precio FLOAT NOT NULL CHECK (precio > 0),
        fechaDeCambio DATE NOT NULL CHECK(fechaDeCambio > '2000-01-01' AND fechaDeCambio < '2022-01-01')
);
```

idSalsa: Es la llave primaria de la entidad "Salsa". Sus restricciones son las mismas que se definieron para "idSalsa" en la tabla **Salsa**.

registroPrecioSalsa: Es la llave primaria de la tabla RegistroPrecioSalsa.

- No aceptamos valor null puesto que es la llave primaria.
- El dato está obligado a tener una longitud de 10 caracteres.
- El dato es único para cada registro de salsa, por ello lo limitamos a que fuera único.

precio: Es el precio que tiene o tuvo la Salsa.

- Este dato es de tipo FLOAT debido a que a veces el precio se maneja con centavos.
- Verificamos que el precio sea mayor a cero.
- Deseamos que se especifique este dato por ello no permitimos que sea una cadena vacia o null.

fechaCambio: Es la fecha en la que se actualizó el precio de una Salsa.

- Este dato es de tipo DATE donde se verifica que el año esté entre 2000 y 2022.
- Deseamos que se especifique este dato por ello no permitimos que sea null.

Tabla RegistroPrecioComida

```
CREATE TABLE RegistroPrecioComida(
        idComida CHAR(10) NOT NULL CHECK (idComida<>''),
        registroPrecioComida CHAR(10) NOT NULL CHECK (registroPrecioComida<>''),
        precio FLOAT NOT NULL CHECK (precio > 0),
        fechaDeCambio DATE NOT NULL CHECK(fechaDeCambio > '2000-01-01' AND fechaDeCambio < '2022-01-01')
);
```

idComida: Es la llave primaria de la entidad "Comida". Sus restricciones son las mismas que se definieron para "idComida" en la tabla **Comida**.

registroPrecioComida: Es la llave primaria de la tabla RegistroPrecioComida.

- No aceptamos valor null puesto que es la llave primaria.
- El dato está obligado a tener una longitud de 10 caracteres.
- El dato es único para cada registro de comida, por ello lo limitamos a que fuera único.

precio: Es el precio que tiene o tuvo una Comida.

- Este dato es de tipo FLOAT debido a que a veces el precio se maneja con centavos.
- Verificamos que el precio sea mayor a cero.
- Deseamos que se especifique este dato por ello no permitimos que sea una cadena vacia o null.

fechaCambio: Es la fecha en la que se actualizó el precio de una Comida.

- Este dato es de tipo DATE donde se verifica que el año esté entre 2000 y 2022.
- Deseamos que se especifique este dato por ello no permitimos que sea null.

Tabla RegistroPrecioBebida

```
CREATE TABLE RegistroPrecioBebida(
        idBebida CHAR(10) NOT NULL CHECK (idBebida<>''),
        registroPrecioBebida CHAR(10) NOT NULL CHECK (registroPrecioBebida<>''),
        precio FLOAT NOT NULL CHECK (precio > 0),
        fechaDeCambio DATE NOT NULL CHECK(fechaDeCambio > '2000-01-01' AND fechaDeCambio < '2022-01-01')
);
```

idBebida: Es la llave primaria de la entidad "Bebida". Sus restricciones son las mismas que se definieron para "idBebida" en la tabla **Bebida**.

registroPrecioBebida: Es la llave primaria de la tabla RegistroPrecioBebida.

- No aceptamos valor null puesto que es la llave primaria.
- El dato está obligado a tener una longitud de 10 caracteres.
- El dato es único para cada registro de bebida, por ello lo limitamos a que fuera único.

precio: Es el precio que tiene o tuvo una Bebida.

- Este dato es de tipo FLOAT debido a que a veces el precio se maneja con centavos.
- Verificamos que el precio sea mayor a cero.
- Deseamos que se especifique este dato por ello no permitimos que sea una cadena vacia o null.

fechaCambio: Es la fecha en la que se actualizó el precio de una Bebida.

- Este dato es de tipo DATE donde se verifica que el año esté entre 2000 y 2022.
- Deseamos que se especifique este dato por ello no permitimos que sea null.

Tabla Transporte

```
CREATE TABLE Transporte(
        curp CHAR(18) NOT NULL CHECK (curp<>'') UNIQUE,
        idTransporte CHAR(10) NOT NULL CHECK (idTransporte<>'') UNIQUE,
        tipo VARCHAR(50) NOT NULL CHECK (tipo<>''),
        marca VARCHAR(50) NOT NULL CHECK (marca<>''),
        modelo VARCHAR(50) NOT NULL CHECK (modelo<>'')
);
```

curp: Es la llave primaria de la entidad "Repartidor". Sus restricciones son las mismas que se definieron para "curp" en la tabla **Repartidor**.

idTransporte: Es la llave primaria de la tabla Transporte.

- No aceptamos valor null puesto que es la llave primaria.
- El id está obligado a tener una longitud de 10 caracteres.
- El id es único para cada transporte, por ello lo limitamos a que fuera único.

tipo: Es el tipo de transporte (motocicleta o bicicleta).

- Debido a que el tipo puede variar, establecimos su tipo de dato como varchar donde máximo puede tener 50 caracteres.
- El tipo no debe de ser una cadena vacía y no se permite que tome el valor null.

marca: Es la marca del Transporte.

- Debido a que la marca puede variar, establecimos su tipo de dato como varchar donde máximo puede tener 50 caracteres.
- La marca no debe de ser una cadena vacía y no se permite que tome el valor null.

modelo: Es el modelo del Transporte.

- Debido a que el modelo puede variar, establecimos su tipo de dato como varchar donde máximo puede tener 50 caracteres.
- El modelo no debe de ser una cadena vacía y no se permite que tome el valor null.

Tabla Ticket

```
CREATE TABLE Ticket(
        idTicket CHAR(10) NOT NULL CHECK(idTicket <> '') UNIQUE,
        rfc CHAR(13) NOT NULL,
        curpCliente CHAR(18) NOT NULL,
        curpMesero CHAR(18) NOT NULL,
        nombreSucursal VARCHAR(50) NOT NULL CHECK(nombreSucursal <> ''),
        nombreCliente VARCHAR(50) NOT NULL CHECK(nombreCliente <> ''),
        nombreMesero VARCHAR(50) NOT NULL CHECK(nombreMesero <> ''),
        fecha DATE NOT NULL CHECK(fecha > '2000-01-01' AND fecha < '2022-01-01')
);
```

idTicket: Es la llave primaria de la tabla Ticket.

- No aceptamos valor null puesto que es la llave primaria.
- El id está obligado a tener una longitud de 10 caracteres.
- El id es único para cada ticket, por ello lo limitamos a que fuera único.

rfc: Es la llave primaria de la entidad "Sucursal". Sus restricciones son las mismas que se definieron para "rfc" en la tabla **Sucursal**.

curpCliente: Es la llave primaria de la entidad "Cliente". Sus restricciones son las mismas que se definieron para "curp" en la tabla **Cliente**.

curpMesero: Es la llave primaria de la entidad "Mesero". Sus restricciones son las mismas que se definieron para "curp" en la tabla **Mesero**.

nombreSucursal: Es un atributo de la entidad "Sucursal". Sus restricciones son las mismas que se definieron para "nombre" en la tabla **Sucursal**.

nombreCliente: Es un atributo de la entidad "Cliente". Sus restricciones son las mismas que se definieron para "nombre" en la tabla **Cliente**.

nombreMesero: Es un atributo de la entidad "Mesero". Sus restricciones son las mismas que se definieron para "nombre" en la tabla **Mesero**.

fecha: Es la fecha cuando se genera un Ticket.

- Este dato es de tipo DATE donde se verifica que el año esté entre 2000 y 2022.
- Deseamos que se especifique este dato por ello no permitimos que sea null.

Tabla Sucursal

```
CREATE TABLE Sucursal(
        rfc CHAR(13) NOT NULL CHECK(rfc <> '') UNIQUE,
        idInventario CHAR(10) NOT NULL,
        nombre VARCHAR(50) NOT NULL CHECK(nombre <> ''),
        calle VARCHAR(50) NOT NULL CHECK(calle <> ''),
        alcaldia VARCHAR(50) NOT NULL CHECK(alcaldia <> ''),
        cp CHAR(5) NOT NULL CHECK(cp <> '[0-9]{5}'),
        estado VARCHAR(50) NOT NULL CHECK(estado <> ''),
        numero CHAR(4) NOT NULL CHECK(numero SIMILAR TO '[0-9]{4}'),
        email VARCHAR(50) NOT NULL CHECK(email <> ''),
        telefono CHAR(10) NOT NULL CHECK(telefono SIMILAR TO '[0-9]{10}')
);
```

rfc: Es la llave primaria de la tabla Sucursal.

- No aceptamos valor null puesto que es la llave primaria.
- El rfc está obligado a tener una longitud de 13 caracteres.
- El rfc es único para cada sucursal, por ello lo limitamos a que fuera único.

idInventario: Es la llave primaria de la entidad "Inventario". Sus restricciones son las mismas que se definieron para "idInventario" en la tabla **Inventario**.

nombre: Es el nombre de la Sucursal.

- Debido a que son muy variados establecimos su tipo de dato como varchar donde cada nombre debe tener máximo 50 caracteres.
- El nombre no debe de ser una cadena vacía ya que no tendría sentido tener sucursales sin nombre, por esa misma razón no permitimos que tome el valor null o vacio.

calle: Es la calle donde se ubica la Sucursal.

- Debido a que las calles son muy variadas establecimos su tipo de dato como varchar donde se debe de tener máximo 50 caracteres.
- Deseamos que se especifique este dato por ello no permitimos que sea una cadena vacia o null.

alcaldia: Es la alcaldía donde se ubica la Sucursal.

- Debido a que las alcaldías son muy variadas establecimos su tipo de dato como varchar donde se debe de tener máximo 50 caracteres.
- Deseamos que se especifique este dato por ello no permitimos que sea una cadena vacia o null.

cp: Es el código postal donde se ubica la Sucursal.

- El código postal está obligado a tener una logitud de 5 caracteres. Se verifica que estos caracteres sean dígitos.
- Deseamos que se especifique este dato por ello no permitimos que sea una cadena vacia o null.

estado: Es la Estado donde se ubica la Sucursal.

- Debido a que los Estados son muy variados establecimos su tipo de dato como varchar donde se debe de tener máximo 50 caracteres.
- Deseamos que se especifique este dato por ello no permitimos que sea una cadena vacia o null.

numero: Es el número donde se ubica la Sucursal.

- El número está obligado a tener una logitud de 4 caracteres. Se verifica que estos caracteres sean dígitos.
- Deseamos que se especifique este dato por ello no permitimos que sea una cadena vacia o null.

email: Es el correo electrónico de la Sucursal.

- Debido a que el email puede ser muy variado establecimos su tipo de dato como varchar donde debe tener máximo 50 caracteres.
- Deseamos que se especifique este dato por ello no permitimos que sea una cadena vacia o null.

telefono: Es el número telefónico de la Sucursal.

- El telefono está obligado a tener una logitud de 10 caracteres. Se verifica que estos caracteres sean dígitos.
- Deseamos que se especifique este dato por ello no permitimos que sea una cadena vacia o null.

Tabla MetodoDePago

```
CREATE TABLE MetodoPago(
        idPago CHAR(10) NOT NULL CHECK(idPago <> '') UNIQUE,
        curp CHAR(18) NOT NULL,
        cantidad FLOAT NOT NULL CHECK(cantidad>0),
        nombreTitular VARCHAR(50) NOT NULL CHECK (esEfectivo <> true AND esPuntos <> true),
        esEfectivo BOOLEAN NOT NULL,
        esPuntos BOOLEAN NOT NULL,
        esDebito BOOLEAN NOT NULL,
        esCredito BOOLEAN NOT NULL

);
```

idPago: Es la llave primaria de la tabla MetodoDePago.

- No aceptamos valor null puesto que es la llave primaria.
- El id está obligado a tener una longitud de 10 caracteres.
- El id es único para cada método de pago, por ello lo limitamos a que fuera único.

curp: Es la llave primaria de la entidad “Cliente”. Sus restricciones son las mismas que se definieron para “curp” en la tabla **Cliente**.

cantidad: Es la cantidad de dinero que poseé un método de pago.

- Este dato es de tipo FLOAT debido a que a veces la cantidad de dinero se maneja con centavos.
- Verificamos que la cantidad sea mayor a cero.
- Deseamos que se especifique este dato por ello no permitimos que sea una cadena vacia o null.

nombreTitular: Es un atributo de la entidad "Cliente". Sus restricciones son las mismas que se definieron para "nombre" en la tabla **Cliente**.

esEfectivo: Nos dice si un método de pago es en efectivo.

- Este dato es de tipo Boolean, donde los posibles valores a tomar serán 'true' y 'false'.
- Deseamos que se especifique este dato por ello no permitimos que sea una cadena vacia o null.

esPuntos: Nos dice si un método de pago es por medio de puntos.

- Este dato es de tipo Boolean, donde los posibles valores a tomar serán 'true' y 'false'.
- Deseamos que se especifique este dato por ello no permitimos que sea una cadena vacia o null.

esDebito: Nos dice si un método de pago es por medio de tarjeta de débito.

- Este dato es de tipo Boolean, donde los posibles valores a tomar serán 'true' y 'false'.
- Deseamos que se especifique este dato por ello no permitimos que sea una cadena vacia o null.

esCredito: Nos dice si un método de pago es por medio de tarjeta de crédito.

- Este dato es de tipo Boolean, donde los posibles valores a tomar serán 'true' y 'false'.
- Deseamos que se especifique este dato por ello no permitimos que sea una cadena vacia o null.

Tabla Proveer

```
CREATE TABLE Proveer(
        idInsumo CHAR(10) NOT NULL CHECK (idInsumo <>''),
        rfc CHAR(13) NOT NULL CHECK (rfc<>''),
        nombre VARCHAR(50) NOT NULL CHECK(nombre <> '')
);
```

idInsumo: Es la llave primaria de la entidad "Insumo". Sus restricciones son las mismas que se definieron para "idInsumo" en la tabla **Insumo**.

rfc: Es la llave primaria de la entidad "Proveedor". Sus restricciones son las mismas que se definieron para "rfc" en la tabla **Proveedor**.

nombre: Es el nombre del Insumo.

- Debido a que son muy variados establecimos su tipo de dato como varchar donde cada nombre debe tener máximo 50 caracteres.
- Deseamos que se especifique este dato por ello no permitimos que sea una cadena vacia o null.

Tabla TenerInsumo

```
CREATE TABLE TenerInsumo(
        idInventario CHAR(10) NOT NULL CHECK(idInventario <> ''),
        idInsumo CHAR(10) NOT NULL,
        nombre VARCHAR(50) NOT NULL CHECK(nombre <> ''),
        precio FLOAT NOT NULL CHECK(precio>0),
        cantidad INT NOT NULL CHECK(cantidad>0),
        caducidad DATE CHECK(caducidad > '2022-01-01'),
        diaCompra DATE NOT NULL CHECK(diaCompra > '2000-01-01' AND diaCompra < '2022-01-01'),
        marcaProducto VARCHAR(50) NOT NULL CHECK(marcaProducto <> '')
);
```

idInventario: Es la llave primaria de la entidad "Inventario". Sus restricciones son las mismas que se definieron para "idInventario" en la tabla **Inventario**.

idInsumo: Es la llave primaria de la entidad "Insumo". Sus restricciones son las mismas que se definieron para "idInsumo" en la tabla **Insumo**.

nombre: Es el nombre del Insumo.

- Debido a que son muy variados establecimos su tipo de dato como varchar donde cada nombre debe tener máximo 50 caracteres.
- Deseamos que se especifique este dato por ello no permitimos que sea una cadena vacia o null.

precio: Es el precio que tiene el Insumo.

- Este dato es de tipo FLOAT debido a que a veces el precio se maneja con centavos.
- Verificamos que el precio sea mayor a cero.
- Deseamos que se especifique este dato por ello no permitimos que sea una cadena vacia o null.

cantidad: Es la cantidad de cada insumo que tendrá un Inventario.

- Este valor es de tipo INT, donde se verifica que sea mayor a cero.
- Deseamos que se especifique este dato por ello no permitimos que sea una cadena vacia o null.

caducidad: Es la fecha en la que un insumo caduca.

- Este dato es de tipo DATE donde se verifica que el año sea mayor a 2021.
- Debido a que hay insumo materiales como mesas, servilletas, etc. puede aceptar valores nulos ya que estos no caducan.

diaCompra: Es la fecha en la que se adquirió un insumo.

- Este dato es de tipo DATE donde se verifica que el año esté entre 2000 y 2022.
- Deseamos que se especifique este dato por ello no permitimos que sea una cadena vacia o null.

marcaProducto: Es la marca del Isumo.

- Debido a que son muy variadas establecimos su tipo de dato como varchar donde cada marca debe tener máximo 50 caracteres.
- La marca no debe de ser una cadena vacía y no se permite que tome el valor null.

Tabla PrepararSalsa

```
CREATE TABLE PrepararSalsa(
        idInsumo CHAR(10) NOT NULL CHECK (idInsumo<>''),
        idSalsa CHAR(10) NOT NULL CHECK (idSalsa<>''),
        cantidad INT NOT NULL CHECK(cantidad > 0)
);
```

idInsumo: Es la llave primaria de la entidad "Insumo". Sus restricciones son las mismas que se definieron para "idInsumo" en la tabla **Insumo**.

idSalsa: Es la llave primaria de la entidad "Salsa". Sus restricciones son las mismas que se definieron para "idSalsa" en la tabla **Salsa**.

cantidad: Es la cantidad que se necesita de un insumo para poder preparar la Salsa.

- Este valor es de tipo INT, donde se verifica que sea mayor a cero.
- Deseamos que se especifique este dato por ello no permitimos que sea una cadena vacia o null.

Tabla PrepararComida

```
CREATE TABLE PrepararComida(
        idInsumo CHAR(10) NOT NULL CHECK (idInsumo<>''),
        idComida CHAR(10) NOT NULL CHECK (idComida<>''),
        cantidad INT NOT NULL CHECK(cantidad > 0)
);
```

idInsumo: Es la llave primaria de la entidad "Insumo". Sus restricciones son las mismas que se definieron para "idInsumo" en la tabla **Insumo**.

idComida: Es la llave primaria de la entidad "Comida". Sus restricciones son las mismas que se definieron para "idComida" en la tabla **Comida**.

cantidad: Es la cantidad que se necesita de un insumo para poder preparar una Comida.

- Este valor es de tipo INT, donde se verifica que sea mayor a cero.
- Deseamos que se especifique este dato por ello no permitimos que sea una cadena vacia o null.

Tabla PrepararBebida

```
CREATE TABLE PrepararBebida(
        idInsumo CHAR(10) NOT NULL CHECK (idInsumo<>''),
        idBebida CHAR(10) NOT NULL CHECK (idBebida<>''),
        cantidad INT NOT NULL CHECK(cantidad > 0)
);
```

idInsumo: Es la llave primaria de la entidad "Insumo". Sus restricciones son las mismas que se definieron para "idInsumo" en la tabla **Insumo**.

idBebida: Es la llave primaria de la entidad "Bebida". Sus restricciones son las mismas que se definieron para "idBebida" en la tabla **Bebida**.

marcaProducto: Es la marca del Isumo.

- Debido a que son muy variadas establecimos su tipo de dato como varchar donde cada marca debe tener máximo 50 caracteres.
- La marca no debe de ser una cadena vacía y no se permite que tome el valor null.

cantidad: Es la cantidad que se necesita de un insumo para poder preparar una Bebida.

- Este valor es de tipo INT, donde se verifica que sea mayor a cero.
- Deseamos que se especifique este dato por ello no permitimos que sea una cadena vacia o null.

Tabla TenerPromo

```
CREATE TABLE TenerPromo(
        idPromocion CHAR(10) NOT NULL CHECK(idPromocion <> ''),
        idComida CHAR(10) NOT NULL CHECK(idComida <> '')
);
```

idPromocion: Es la llave primaria de la entidad "Promocion". Sus restricciones son las mismas que se definieron para "idPromocion" en la tabla **Promocion**.

idComida: Es la llave primaria de la entidad "Comida". Sus restricciones son las mismas que se definieron para "idComida" en la tabla **Comida**.

Tabla RegistrarSalsa

```
CREATE TABLE RegistrarSalsa(
        idTicket CHAR(10) NOT NULL CHECK (idTicket<>''),
        idSalsa CHAR(10) NOT NULL CHECK (idSalsa<>''),
        nombre VARCHAR(50) NOT NULL CHECK(nombre <> ''),
        precioActual FLOAT NOT NULL CHECK(precioActual > 0)
);
```

idTicket: Es la llave primaria de la entidad "Ticket". Sus restricciones son las mismas que se definieron para "idTicket" en la tabla **Ticket**.

idSalsa: Es la llave primaria de la entidad "Salsa". Sus restricciones son las mismas que se definieron para "idSalsa" en la tabla **Salsa**.

nombre: Es el nombre de la Salsa que se compró.

- Debido a que son muy variados establecimos su tipo de dato como varchar donde cada nombre debe tener máximo 50 caracteres.
- Deseamos que se especifique este dato por ello no permitimos que sea una cadena vacia o null.

precioActual: Es el precio actual de la Salsa que se compró.

- Este dato es de tipo FLOAT debido a que a veces el precio se maneja con centavos.
- Verificamos que el precio sea mayor a cero.
- Deseamos que se especifique este dato por ello no permitimos que sea una cadena vacia o null.

Tabla RegistrarComida

```
CREATE TABLE RegistrarComida(
        idTicket CHAR(10) NOT NULL CHECK (idTicket<>''),
        idComida CHAR(10) NOT NULL CHECK (idComida<>''),
        nombre VARCHAR(50) NOT NULL CHECK(nombre <> ''),
        precioActual FLOAT NOT NULL CHECK(precioActual > 0)
);
```

idTicket: Es la llave primaria de la entidad "Ticket". Sus restricciones son las mismas que se definieron para "idTicket" en la tabla **Ticket**.

idComida: Es la llave primaria de la entidad "idComida". Sus restricciones son las mismas que se definieron para "idComida" en la tabla **Comida.**

nombre: Es el nombre de la Comida que se compró.

- Debido a que son muy variados establecimos su tipo de dato como varchar donde cada nombre debe tener máximo 50 caracteres.

- Deseamos que se especifique este dato por ello no permitimos que sea una cadena vacia o null.

precioActual: Es el precio actual de la Comida que se compró.

- Este dato es de tipo FLOAT debido a que a veces el precio se maneja con centavos.
- Verificamos que el precio sea mayor a cero.
- Deseamos que se especifique este dato por ello no permitimos que sea una cadena vacia o null.

Tabla RegistrarBebida

```
CREATE TABLE RegistrarBebida(
        idTicket CHAR(10) NOT NULL CHECK (idTicket<>''),
        idBebida CHAR(10) NOT NULL CHECK (idBebida<>''),
        nombre VARCHAR(50) NOT NULL CHECK(nombre <> ''),
        precioActual FLOAT NOT NULL CHECK(precioActual > 0)
);
```

idTicket: Es la llave primaria de la entidad "Ticket". Sus restricciones son las mismas que se definieron para "idTicket" en la tabla **Ticket**.

idBebida: Es la llave primaria de la entidad "idBebida". Sus restricciones son las mismas que se definieron para "idBebida" en la tabla **Bebida.**

nombre: Es el nombre de la Bebida que se compró.

- Debido a que son muy variados establecimos su tipo de dato como varchar donde cada nombre debe tener máximo 50 caracteres.
- Deseamos que se especifique este dato por ello no permitimos que sea una cadena vacia o null.

precioActual: Es el precio actual de la Bebida que se compró.

- Este dato es de tipo FLOAT debido a que a veces el precio se maneja con centavos.
- Verificamos que el precio sea mayor a cero.
- Deseamos que se especifique este dato por ello no permitimos que sea una cadena vacia o null.

Tabla Recomendar

```
CREATE TABLE Recomendar(
        idSalsa CHAR(10) NOT NULL CHECK (idSalsa <>''),
        idComida CHAR(10) NOT NULL CHECK (idComida <>'')
);
```

idSalsa: Es la llave primaria de la entidad "Salsa". Sus restricciones son las mismas que se definieron para "idSalsa" en la tabla **Salsa**.

idComida: Es la llave primaria de la entidad "Comida". Sus restricciones son las mismas que se definieron para "idComida" en la tabla **Comida**.

Tabla Mesero

```
CREATE TABLE Mesero(
        curp CHAR(18) NOT NULL CHECK(CHAR_LENGTH(curp) = 18) CHECK(curp SIMILAR TO '[A-Z]{4}[0-9]{6}[A-Z]{8}') UNIQUE,
        rfc CHAR(13) NOT NULL CHECK(rfc <> '')
);
```

curp: Es la llave primaria de la entidad "Cliente". Sus restricciones son las mismas que se definieron para "curp" en la tabla **Cliente**.

rfc: Es la llave primaria de la entidad "Sucursal". Sus restricciones son las mismas que se definieron para "rfc" en la tabla **Sucursal**.

Tabla Parrillero

```
CREATE TABLE Parrillero(
        curp CHAR(18) NOT NULL CHECK(CHAR_LENGTH(curp) = 18) CHECK(curp SIMILAR TO '[A-Z]{4}[0-9]{6}[A-Z]{8}') UNIQUE,
        rfc CHAR(13) NOT NULL CHECK(rfc <> '')
);
```

curp: Es la llave primaria de la entidad "Cliente". Sus restricciones son las mismas que se definieron para "curp" en la tabla **Cliente**.

rfc: Es la llave primaria de la entidad "Sucursal". Sus restricciones son las mismas que se definieron para "rfc" en la tabla **Sucursal**.

Tabla Taquero

```
CREATE TABLE Taquero(
        curp CHAR(18) NOT NULL CHECK(CHAR_LENGTH(curp) = 18) CHECK(curp SIMILAR TO '[A-Z]{4}[0-9]{6}[A-Z]{8}') UNIQUE,
        rfc CHAR(13) NOT NULL CHECK(rfc <> '')
);
```

curp: Es la llave primaria de la entidad "Cliente". Sus restricciones son las mismas que se definieron para "curp" en la tabla **Cliente**.

rfc: Es la llave primaria de la entidad "Sucursal". Sus restricciones son las mismas que se definieron para "rfc" en la tabla **Sucursal**.

Tabla Repartidor

```
CREATE TABLE Repartidor(
    curp CHAR(18) NOT NULL CHECK(CHAR_LENGTH(curp) = 18) CHECK(curp SIMILAR TO '[A-Z]{4}[0-9]{6}[A-Z]{8}') UNIQUE,
    rfc CHAR(13) NOT NULL CHECK(rfc <> ''),
    numLicencia CHAR(10) CHECK(numLicencia <> ''),
    transporte VARCHAR(20) CHECK(transporte = 'Bicicleta' OR transporte = 'Motocicleta')
);
```

curp: Es la llave primaria de la entidad "Cliente". Sus restricciones son las mismas que se definieron para "curp" en la tabla **Cliente**.

rfc: Es la llave primaria de la entidad "Sucursal". Sus restricciones son las mismas que se definieron para "rfc" en la tabla **Sucursal**.

numLicencia: Es el número de lincencia del Repartidor.

- El número de lincencia está obligado a tener una logitud de 10 caracteres.
- Deseamos que se especifique este dato por ello no permitimos que sea una cadena vacia o null.

transporte: Es el transporte con el cual el repartidor hace las entregas.

- Establecimos su tipo de dato como varchar con máximo 20 caracteres.
- Debido a que no todos los repartidores cuentan con un transporte, este dato acepta null.

Tabla Tortillero

```
CREATE TABLE Tortillero(
        curp CHAR(18) NOT NULL CHECK(CHAR_LENGTH(curp) = 18) CHECK(curp SIMILAR TO '[A-Z]{4}[0-9]{6}[A-Z]{8}') UNIQUE,
        rfc CHAR(13) NOT NULL CHECK(rfc <> '')
);
```

curp: Es la llave primaria de la entidad “Cliente”. Sus restricciones son las mismas que se definieron para “curp” en la tabla **Cliente**.

rfc: Es la llave primaria de la entidad “Sucursal”. Sus restricciones son las mismas que se definieron para “rfc” en la tabla **Sucursal**.

Tabla Cajero

```
CREATE TABLE Cajero(
        curp CHAR(18) NOT NULL CHECK(CHAR_LENGTH(curp) = 18) CHECK(curp SIMILAR TO '[A-Z]{4}[0-9]{6}[A-Z]{8}') UNIQUE,
        rfc CHAR(13) NOT NULL CHECK(rfc <> '')
);
```

curp: Es la llave primaria de la entidad “Cliente”. Sus restricciones son las mismas que se definieron para “curp” en la tabla **Cliente**.

rfc: Es la llave primaria de la entidad “Sucursal”. Sus restricciones son las mismas que se definieron para “rfc” en la tabla **Sucursal**.

3.2 Integridad Referencial

La integridad referencial nos ayuda a mantener consistencia entre las tuplas de dos relaciones. Esto significa que la llave externa (FK) de una tabla de referencia siempre debe aludir a una fila válida de la tabla a la que se haga referencia. La integridad referencial garantiza que la relación entre dos tablas permanezca sincronizada durante las operaciones de actualización y eliminación.

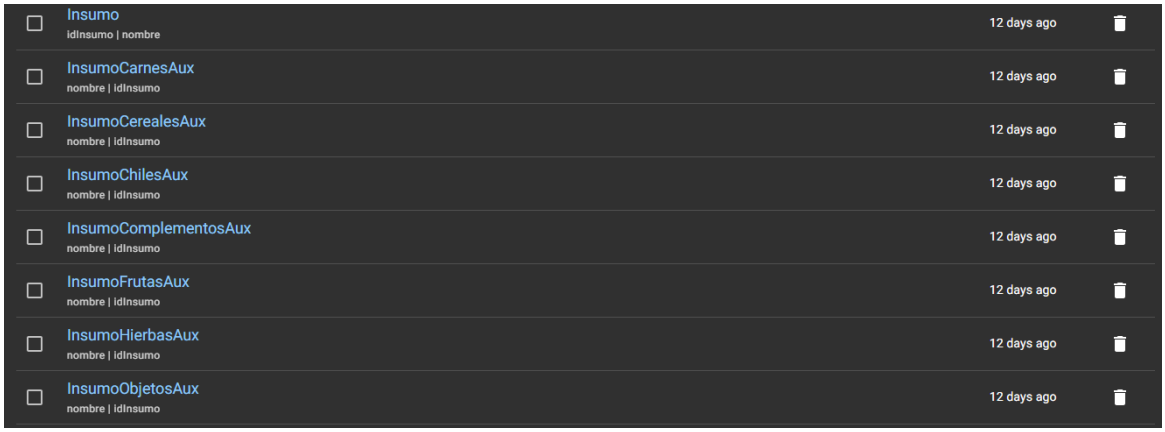
Anteriormente hicimos explicitas las restricciones para las llaves primarias (PK) de nuestras tablas, donde todas deben de cumplir no ser nulas.
Al final de nuestro archivo DDL.sql se puede ver de manera más clara cómo aplicamos la interidad referencial. Es importante mencionar que la política de mantenimiento para llaves foráneas que decidimos usar fue la política de cascada, la razón es porque tenemos en cuenta que al realizar acciones en cascada afectará a todo tipo de tablas dentro de nuestro sistema que tengan referencia a nuestras llaves foráneas, si quisiéramos borrar un dato del que dependan otros datos en específico, todas las referencias a este dato se borrarían del sistema, lo que mantendría la consistencia dentro del mismo. Además, actualizar las referencias de las llaves es sencillo, pues lo hace automáticamente, lo que mantiene la integridad y consistencia de la base de datos.

4. PROBLAMIENTO DE LA BASE DE DATOS

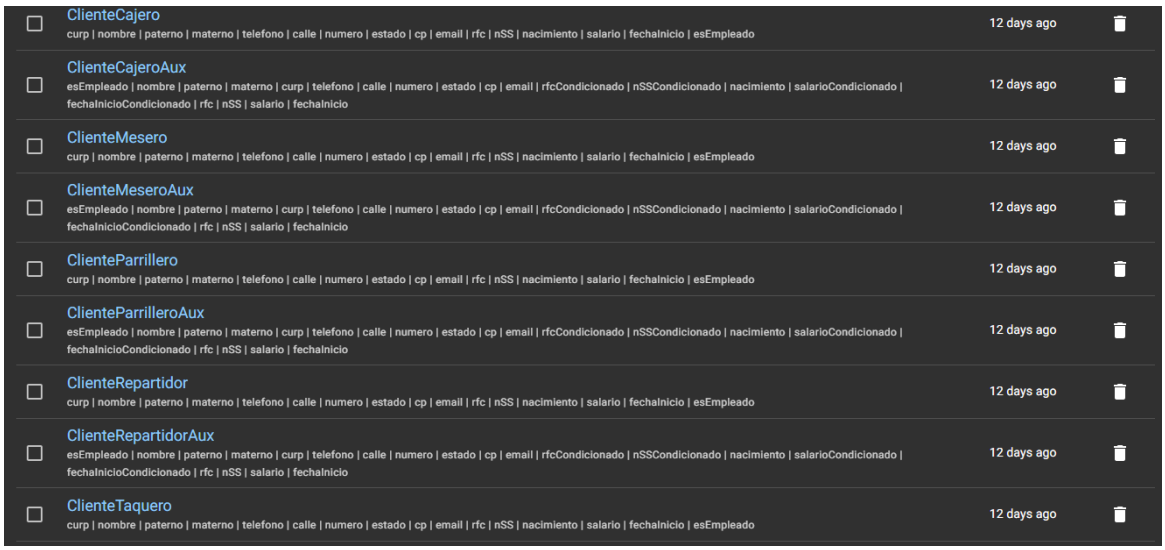
4.1 DML (Data Manipulation Language)

Para realizar esta tarea nos ayudamos de la herramienta **Mockaroo**. La poblacion de la base de datos tiene las siguientes consideraciones.

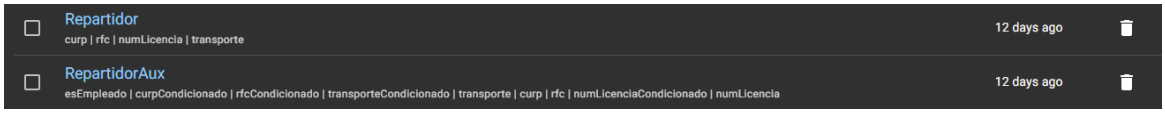
- Para las tabla **Insumo** generamos diferentes tipos de insumo entre los que están las categorías de carnes (tipo de carne empleada en los productos), cereales (tipo de alimento de esta categoría como lo pueden ser bolillos, tortillas etc), chiles (para la elaboración de salsas), frutas (para la elaboración de comidas, bebidas y salsas), hierbas (para la elaboración de comidas y salsas) y objetos (los objetos que pedía el caso de uso como mesas, servilleteros, etc) éstas a su vez serán manejados en un inventario para verificar si son o no perecederos y asignar una fecha adecuada de caducidad en caso de serlo.



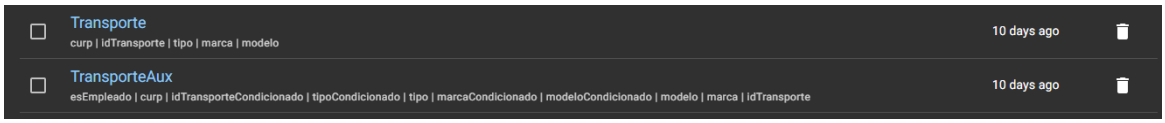
- Para la tabla **Cliente** usamos un sistema que generaba los posibles diferentes empleados de cierto tipo en específico, como el caso de uso explica que no es válido repetir información, alteramos la información para poder asignar ciertos roles mediante el uso de curps modificados, cada cliente puede ser un tipo de empleado por lo que los curps de todos los clientes tienen la terminación de las primeras 3 letras que representan la posibilidad de dicho cliente de ser o no un Mesero, Parrillero, Repartidor, Taquero, Tortillero o Cajero, de esta forma si un cliente a su vez trabaja en un local se va directo a la tabla correspondiente a su cargo, ejemplo: CURP123456ABCDE**MES** este curp pertenece a un cliente que puede ser o no un Mesero, más que nada es para identificar de manera más fácil los tipos de empleado que hay, si un cliente no es empleado (esto se verifica con su valor booleano esEmpleado) entonces los valores nSS (úumero de Seguridad Social), rfc, fecha Inicio y Salario son asignados con el valor NULL.



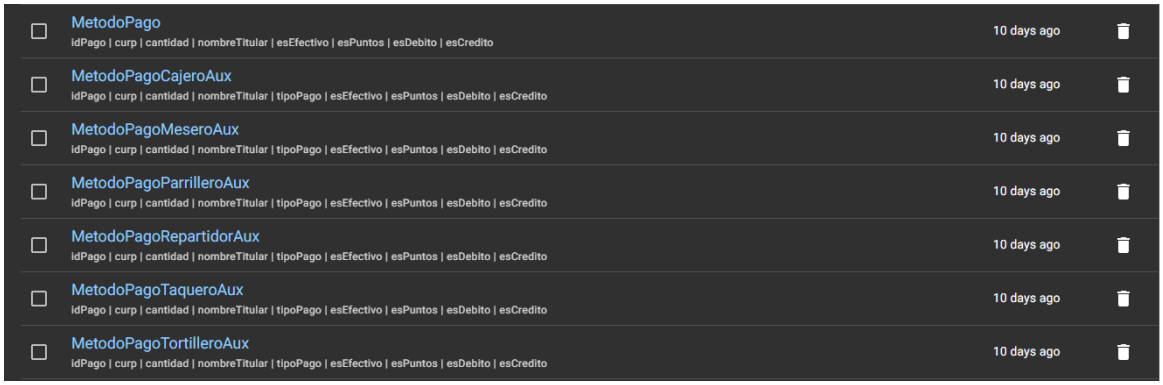
- Para la tabla **Repartidor** generamos al azar el tener una licencia de manejo, la licencia de manejo implica que el repartidor tiene motocicleta como transporte por lo que de no tener licencia tiene dos opciones, o tiene una bicicleta o no tiene transporte, de ser el primer caso en la licencia se asigna el valor NULL, de no tener transporte en ambos casos se asigna el valor NULL.



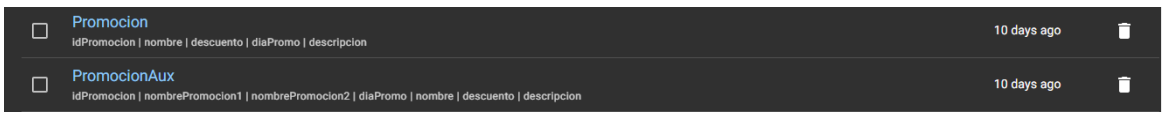
- Para la tabla **Transporte** generamos una condición en la que si el repartidor no tiene una licencia de conducir y no tiene transporte propio se le asigne una bicicleta como medio de transporte, por que como un repartidor que no tiene vehículo no es de utilidad entonces al prestarle una bicicleta no afectamos el uso de licencia y el repartidor a su vez tiene una utilidad en el local, por cada repartidor que hay en el local hay un transporte que esta encargado a dicho repartidor (sea que tenga o no medio de transporte propio).



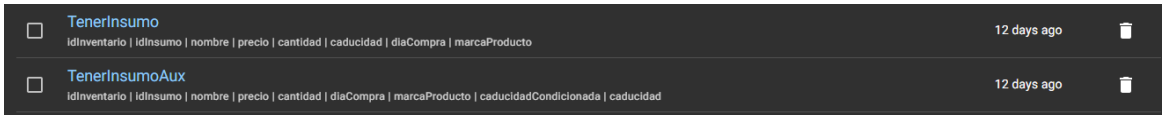
- Para la tabla **MetodoPago** tenemos cuatro atributos que muestran si el método de pago es en efectivo, puntos, por tarjeta de crédito o de débito, estos tienen un valor de tipo booleano, el cual generando un número del 1 al 4 aleatoreamente revisa que coincida con alguno de los campos que tenemos, de ser así le asigna un valor de verdadero, de lo contrario le asigna falso.



- Para la tabla **Promocion** tomamos como consideración el porcentaje a descontar en los productos mediante un valor de tipo flotante para que cuando una promoción sea válida solamente se haga la multiplicación del descuento por el precio del producto al que se le asigna dicho producto.

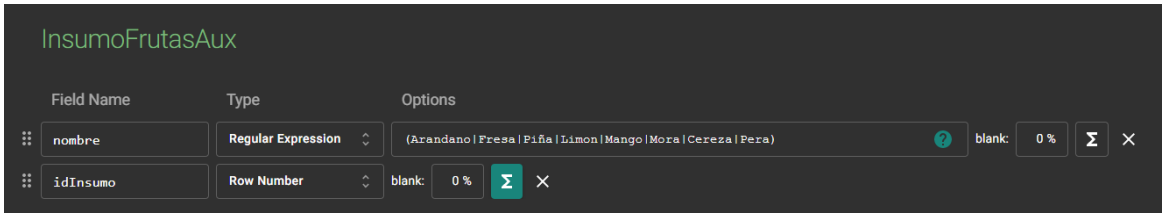


- Para la tabla **TenerInsumo** mediante la verificación del id del insumo que previamente fue modificado a voluntad (la primera letra siendo una I y las siguientes cuatro siendo las primeras cuatro letras del nombre del insumo, por ejemplo IPERA12345) checamos si es o no un insumo perecedero para asignarle una fecha de caducidad, si viene de la familia de objetos (previamente definida en la tabla Insumo) entonces la fecha es un NULL, de lo contrario se asigna una fecha.

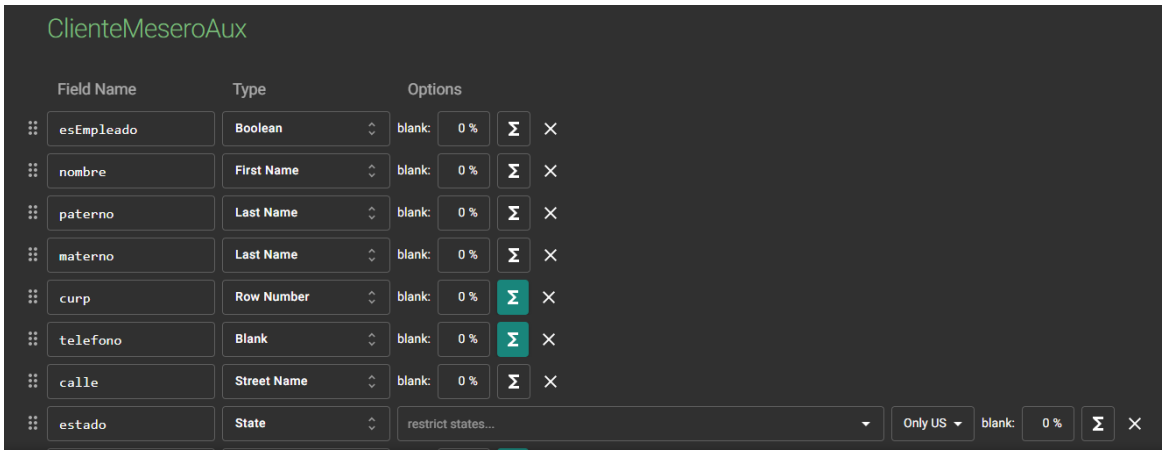


A grandes rasgos estas son las consideraciones más importantes sobre el poblamiento de tablas,

el resto de tablas hasta cierto punto se generaron de manera secuencial gracias a que se definió anteriormente sus cualidades de cada una de ellas, por lo que no hace falta explicarlas, lo que sí se puede tomar en cuenta es que se hizo expresiones regulares que generaran datos que tengan sentido, como por ejemplo, frutas para las bebidas, chiles para las salsas, tipos de carne para las comidas, etc. Se hubieran agregado más si la herramienta Mockaroo no fuese tan ineficiente a la hora de tomar cosas al azar, una disculpa de antemano.



Todos los estados asó como direcciones son de Estados Unidos, para poder preservar una mayor aleatoriedad, esto es meramente para mostrar el funcionamiento adecuado de la base, pero sirve para los Estados de México de igual forma.

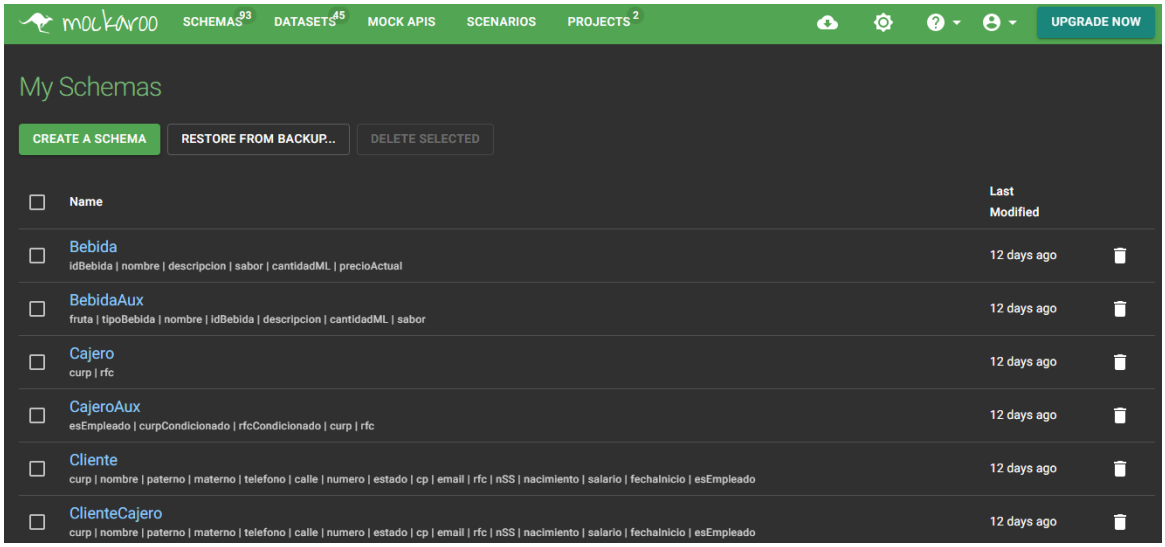


Se generaron alrededor de 60,000 registros para poblar la base de datos, consideramos importante estos hechos para el uso de la base:

- Hay un máximo de 100 Sucursales.
- Hay 6,000 clientes dentro de los que hay un aproximado entre 3,000 empleados y 3,000 que no lo son.
- Hay alrededor de 500 Empleados de cada Tipo (500 meseros, 500 cajeros, 500 repartidores, 500 tortilleros, 500 parrilleros, 500 taqueros).
- Hay 100 tipos de insumos diferentes.
- Hay 100 tipos de bebidas diferentes.
- Hay 100 tipos de comidas diferentes.
- Hay 100 tipos de salsas diferentes.
- Están registrados 12,000 productos de tipo bebida dentro de los tickets.
- Están registrados 12,000 productos de tipo comida dentro de los tickets.
- Están registrados 12,000 productos de tipo salsa dentro de los tickets.
- Hay 6,000 tickets diferentes.
- Cada cliente tiene registrado su método de pago por lo que hay 6,000 métodos de pago.

- El resto de tablas son los registros de los precios de los productos, así como la preparación de los mismos, promociones o recomendaciones.

  La herramienta Mockaroo nos ayudó a generar 93 Schemas y 45 DataSets que permitieron poblar la base de datos en su totalidad.



5. PROCEDIMIENTOS ALMACENADOS Y DISPARADORES

5.1 Procedimientos

***NOTA**: El archivo "Procedures.sql"es el que contiene nuestros procedimientos que diseñamos para la Base de Datos. Para que nuestros procedimientos funcionen correctamente es indispensable ejecutar los bloques de código como van apareciendo en orden.

- **PROCEDIMIENTO 1: bonosEmpleado**

  Su objetivo es aumentar 1,500 al salario de un empleado siempre y cuando haya laborado por dos años o más en una sucursal.

  Éste procedimiento recibe la curp del empleado, la cual nos ayudará a acceder a su atributo "fechaInicio"(representa la fecha cuando el empleado comenzó a trabajar en una sucursal). Creamos una función auxiliar llamada "aniosAntiguedad" la cual toma a la fecha actual y al atributo fechaInicio y los resta, de dicho resultado obtenemos sólo el año para poder saber cuántos años de antigüedad tiene el empleado. El procedimiento toma este valor y verifica que sea mayor o igual a 2, si la condición se cumple entonces se le aumenta el salario al empleado.

  Quisimos hacer este procedimiento robusto, por lo tando si se ingresa un id que no existe manda un error. También debido a que un cliente puede ser un empleado (y los datos de clientes y empleados se guardan en la misma tabla) verificamos que el id ingresado pertenezca a un empleado y no a un cliente.

- **PROCEDIMIENTO 2: taquero_Corazon**

  Su objetivo es registra en una tabla todos los puntos que ha acumulado un cliente a partir de sus compras en la sucursal. Recordemos que Taquero de Corazón es un programa el cual otorga el 10% del consumo en puntos, donde cada punto equivale a un peso.

  Creamos una función auxiliar llamada "cuentaTotalTicket" la cual recibe la curp del cliente y a partir de ahí accedemos a sus tickets generados al comprar salsas, comida y bebidas para posteriormente sumar todo y regresar este valor el cual es tomado por el procedimiento y lo multiplica por 0.10 para así obtener sus puntos que representan el 10% del consumo, después insertamos la curp del cliente y sus puntos en la tabla "PuntosCliente".

  También quisimos hacer este procedimientos robusto, por lo tando si se ingresa una curp que no existe se mandará un error.

5.2 Disparadores

***NOTA**: El archivo "Triggers"es el que contiene nuestros disparadores que diseñamos para la Base de Datos. Para que nuestros disparadores funcionen correctamente es indispensable ejecutar los bloques de código como van apareciendo en orden.

- **DISPARADOR 1: registra_Empleado**

  El objetivo es registrar a un nuevo empleado y también registrar en una tabla llamada "Registros"cuándo y quién insertó a dicho empleado.

  Quisimos hacerlo robusto, por lo que creamos dos funciones auxiliares "checaIdSucursal"y "checaNombres". La primera función verifica si existe una sucursal dado su id (la función fue creada ya que para insertar a un empleado necesitamos de una sucursal ya existente). La segunda función verifica que no se ingresen números o caracteres especiales en el nombre completo del empleado (nombre, apellido paterno y materno). Creamos también un Procedimiento llamado "insertEmpleado"el cual hace la tarea de insertar al empleado en sí. Posteriormente creamos el disparador el cual reaccionará cuando se quiera insertar a un nuevo empleado registrando cuándo y quién lo agregó.

- **DISPARADOR 2: check_Operaciones_Sucursal**

  El objetivo es verificar que una sucursal no sea registrada en la misma alcaldía ya que por alcaldía sólo se quiere abrir una única sucursal. También queremos que no se elimine la información de una Sucursal debido a que es información muy importante.

  Creamos una función auxiliar llamada "existeAlcaldia" a la cual se le pasa como parámetro la alcaldía donde se quiere registrar una nueva sucursal, nos regresa true si en dicha alcaldía ya hay una sucursal. Posteriormente creamos el disparador el cual reaccionará mandando un error cuando se quiera insertar una nueva sucursal con una alcaldía ya registrada. También se mandará a error si se quiere eliminar información de una sucursal.

6. CONSULTAS SQL

Todas nuestras consultas se encuentran registradas en los archivos llamados “Consultas.pdf” (se encuentra dentro de la carpeta **Docs**) y “Consultas.sql” (se encuentra dentro de la carpeta **SQL**).