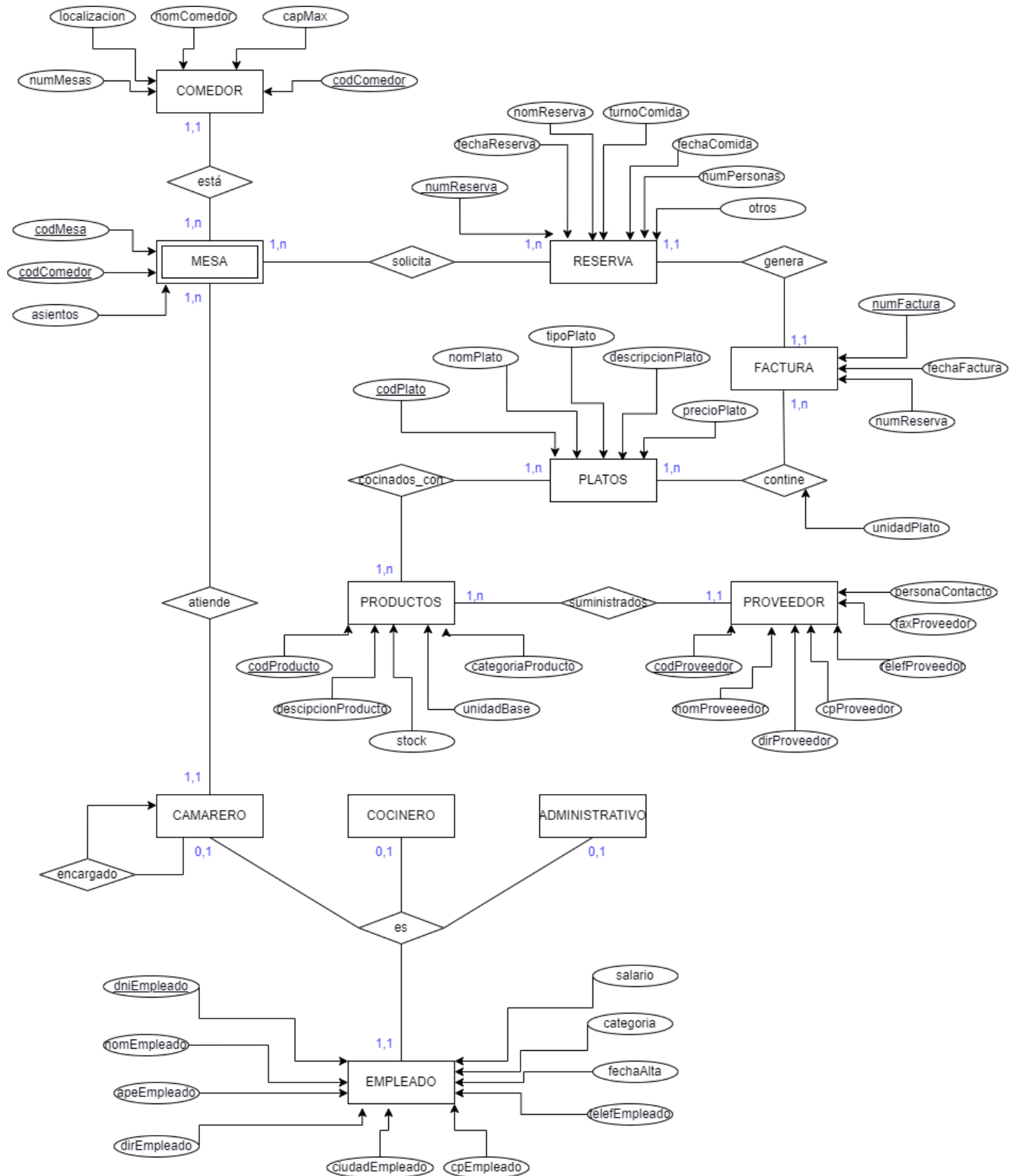


TAREA BASES DE DATOS

1. Modelo Conceptual y Modelo Lógico Relacional

El modelo Entidad-Relación (Diseño Conceptual) que he creado es el siguiente:



Aclaraciones:

- La MESA es una entidad débil de COMEDOR. Para poder definir completamente MESA tengo que saber a qué comedor me estoy refiriendo, puesto que en distintos comedores puede haber mesas con el mismo código.
- Una RESERVA puede estar formada por varias mesas, y a su vez una MESA podrá ser reservada varias veces.
- En la FACTURA hay datos, como el precio de los platos y el precio de la factura total, que se pueden calcular a partir de la información ya almacenada. La FACTURA está formada por los distintos PLATOS, a su vez, un mismo PLATO puede estar en distintas FACTURAS.
- Cada PLATO se elabora con distintos PRODUCTOS, y un mismo PRODUCTO puede formar parte de varios PLATOS distintos.
- Suponemos que cada PRODUCTO es suministrado por un único PROVEEDOR, pero a su vez un mismo PROVEEDOR puede suministrar varios PRODUCTOS.
- Tenemos la superclase EMPLEADO que posee una jerarquía total con CAMARERO, COCINERO y ADMINISTRATIVO. Un empleado solo puede tener uno de estos puestos de trabajo.
- El ENCARGADO del CAMARERO tienen una relación reflexiva, ya que algunos camareros ejercerán de encargados.

A continuación, obtengo el modelo Relacional (Diseño lógico):

EMPLEADO			
NOMBRE	DESCRIPCION	TIPO DE DATO	RESTRICCIONES
dniEmpleado	DNI del empleado	alfanumérico(9)	PK
nomEmpleado	Nombre del empleado	alfanumérico(15)	NOT NULL
apeEmpleado	Apellidos del empleado	alfanumérico(20)	NOT NULL
dirEmpleado	Dirección del empleado	alfanumérico(30)	NOT NULL
ciudadEmpleado	Ciudad del empleado	alfanumérico(15)	NOT NULL
cpEmpleado	CP del empleado	alfanumérico(5)	NOT NULL
telefEmpleado	Teléfono del empleado	alfanumérico(11)	NOT NULL
fechaAlta	Fecha de alta	date	NOT NULL
categoría	Categoría profesional del empleado	alfanumérico(10)	
salario	Salario del empleado	número(6,2)	> 0

CAMARERO			
NOMBRE	DESCRIPCION	TIPO DE DATO	RESTRICCIONES
dniCamarero	DNI del cocinero	alfanumérico(9)	PK FK(dniEmpleado-->EMPLEADO)
turno	Turno de trabajo	alfanumérico	mañana, tarde, noche
añosExperiencia	Años de experiencia	número(2)	
dniEncargado	DNI del encargado	alfanumérico(9)	

COCINERO			
NOMBRE	DESCRIPCION	TIPO DE DATO	RESTRICCIONES
dniCocinero	DNI del cocinero	alfanumérico(9)	PK FK(dniEmpleado-->EMPLEADO)
puesto	Puesto en la cocina	alfanumérico(10)	
especialidad	Especialidad en la cocina	alfanumérico(10)	

ADMINISTRATIVO			
NOMBRE	DESCRIPCION	TIPO DE DATO	RESTRICCIONES
dniAdministrativo	DNI del administrativo	alfanumérico(9)	PK FK(dniEmpleado-->EMPLEADO)
cargo	Cargo que tiene	alfanumérico(10)	

COMEDOR			
NOMBRE	DESCRIPCION	TIPO DE DATO	RESTRICCIONES
codComedor	Código del comedor	alfanumérico(3)	PK
nomComedor	Nombre del comedor	alfanumérico(15)	NOT NULL
capMax	Capacidad máx de comensales	número(3)	NOT NULL / > 0
numMesas	Numero de mesas que tiene	número(2)	NOT NULL / > 0
localizacion	Localización del comedor	alfanumérico(15)	NOT NULL

MESA			
NOMBRE	DESCRIPCION	TIPO DE DATO	RESTRICCIONES
codMesa	Código de la mesa	alfanumérico(5)	PK
codComedor	Código del comedor donde se encuentra la mesa	alfanumérico(5)	PK FK(codComedor-->COMEDOR)
asientos	Asiento tiene la mesa	número(2)	NOT NULL / > 0
dniEmpleado	DNI del empleado que atiende la mesa	alfanumérico(9)	FK(dniEmpleado-->EMPLEADO)

RESERVA			
NOMBRE	DESCRIPCION	TIPO DE DATO	RESTRICCIONES
numReserva	Numero de la reserva	alfanumérico(5)	PK
fechaReserva	Fecha y hora en la que se realiza la reserva	time	
nomReserva	Nombre de la reserva	alfanumérico(15)	NOT NULL
fechaComida	Fecha y hora en la que van a comer	time	NOT NULL
turnoComida	Turno de la comida	alfanumérico	tarde, noche/ NOT NULL
numPersonas	Número de personas que van a comer	número(3)	NOT NULL
otros	Otros datos de interés	alfanumérico(50)	NOT NULL
codComedor	código del comedor donde van a comer	alfanumérico(5)	FK(codComedor-->MESA)
codMesa	código de la mesa	alfanumérico(5)	FK(codMesa-->MESA)

FACTURA			
NOMBRE	DESCRIPCION	TIPO DE DATO	RESTRICCIONES
numFactura	Numero de factura	alfanumérico(5)	PK
fechaFactura	Fecha de la factura	date	NOT NULL
numReserva	Numero de la reserva	alfanumérico(5)	FK(numReserva-->RESERVA)

PLATO			
NOMBRE	DESCRIPCION	TIPO DE DATO	RESTRICCIONES
codPlato	Código del plato	alfanumérico(5)	PK
nomPlato	Nombre del plato	alfanumérico(15)	NOT NULL
tipoPlato	Tipo de plato	alfanumérico(15)	
descripcionPlato	Descripción del plato	alfanumérico(50)	
precioPlato	Precio del plato	number(3,2)	NOT NULL / > 0

PROVEEDOR			
NOMBRE	DESCRIPCION	TIPO DE DATO	RESTRICCIONES
codProveedor	Código del proveedor	alfanumérico(5)	PK
nomProveedor	Nombre del proveedor	alfanumérico(15)	NOT NULL
dirProveedor	Dirección del proveedor	alfanumérico(30)	NOT NULL
cpProveedor	CP del proveedor	alfanumérico(5)	NOT NULL
telefProveedor	Teléfono del proveedor	alfanumérico(11)	NOT NULL
faxProveedor	Fax del proveedor	alfanumérico(11)	
personaContacto	Persona de contacto del proveedor	alfanumérico(20)	NOT NULL

PRODUCTO			
NOMBRE	DESCRIPCION	TIPO DE DATO	RESTRICCIONES
codProducto	Código del producto	alfanumérico(5)	PK
descripcionProducto	Descripción del producto	alfanumérico(50)	
stock	Stock del producto	número(5,3)	NOT NULL / > 0
unidadBase	Unida-base(docenas, kg, litros, etc.)	alfanumérico(10)	NOT NULL
categoriaProducto	Categoría a la que pertenece el producto	alfanumérico(10)	
codProveedor	Código del proveedor	alfanumérico(5)	FK(codProveedor-->PROVEEDOR)

Todas las relaciones N:M se transforman en una tabla

contiene (FACTURA-PLATOS)			
NOMBRE	DESCRIPCION	TIPO DE DATO	RESTRICCIONES
codPlato	Código del plato	alfanumérico(5)	PK
			FK(codPlato-->PLATO)
numFactura	Numero de factura	alfanumérico(5)	PK
			FK(codFactura-->FACTURA)
unidadPlato	Unidades de cada plato	número(2)	NOT NULL / > 0

cocinados_con(PLATOS-PRODUCTOS)			
NOMBRE	DESCRIPCION	TIPO DE DATO	RESTRICCIONES
codPlato	Código del plato	alfanumérico(5)	PK
			FK(codPlato-->PLATO)
codProducto	Código del producto	alfanumérico(5)	PK
			FK(codProducto->PRODUCTO)

solicita (MESAS-RESERVA)			
NOMBRE	DESCRIPCION	TIPO DE DATO	RESTRICCIONES
numReserva	Numero de la reserva	alfanumérico(5)	PK
			FK(numReserva-->RESERVA)
codComedor	Código del comedor donde van a comer	alfanumérico(5)	PK
			FK(codComedor-->MESA)
codMesa	Código de la mesa	alfanumérico(5)	PK
			FK(codMesa-->MESA)

Normalización:

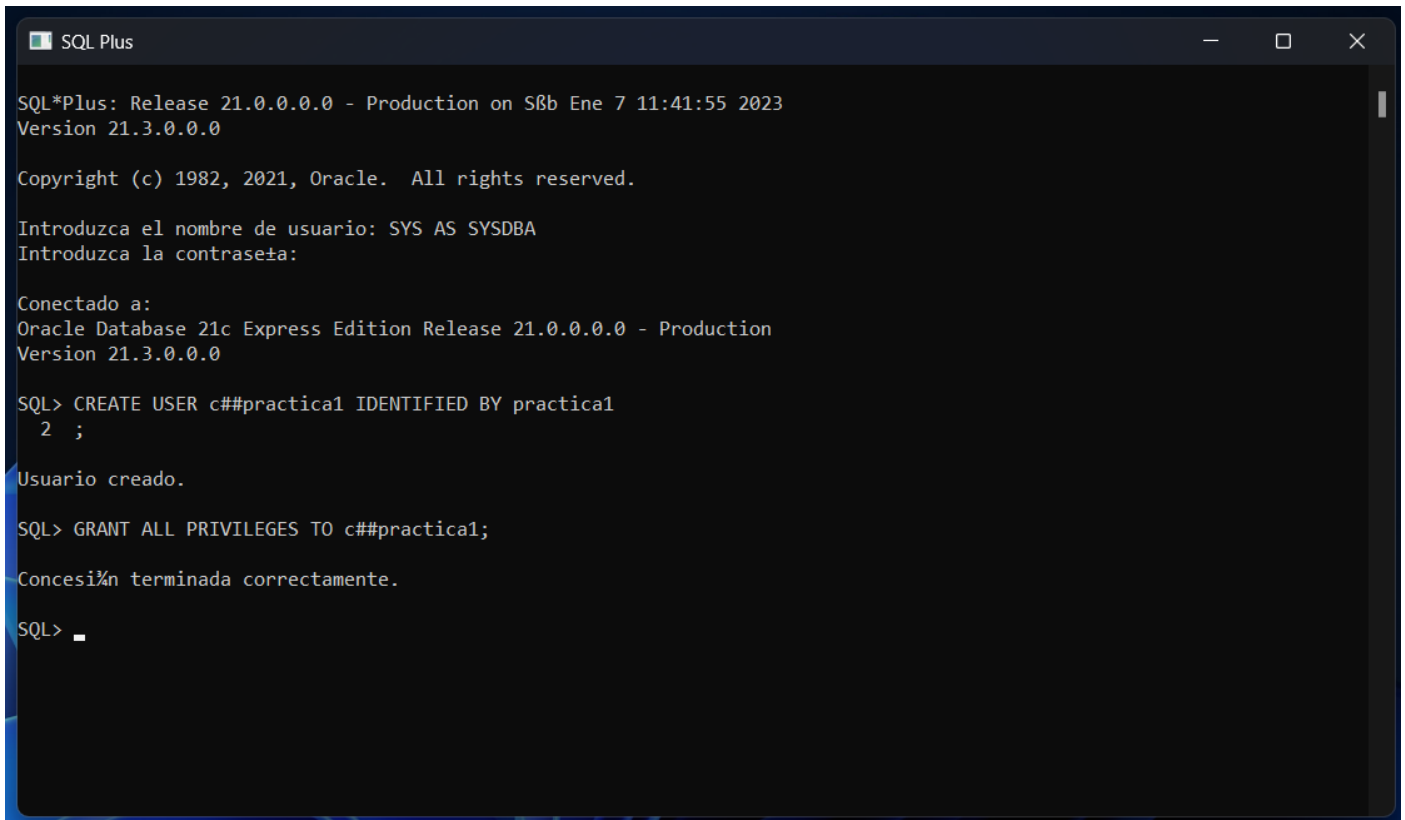
1FN: Todos los atributos son atómicos. Todas las tablas tienen una clave principal no nula.

2FN: En cada tabla, todos los atributos no clave dependen de la clave principal.

3FN: No existe ninguna dependencia transitiva entre los atributos que no son clave.

2. Diseño Físico

Creo un usuario nuevo para trabajar con él.

A screenshot of an SQL*Plus terminal window. The window title is "SQL Plus". The output shows the version information: "SQL*Plus: Release 21.0.0.0.0 - Production on S8b Ene 7 11:41:55 2023" and "Version 21.3.0.0.0". It also displays the copyright notice: "Copyright (c) 1982, 2021, Oracle. All rights reserved." The user is prompted to enter a username and password, and then connected to the Oracle Database 21c Express Edition. The user then enters the command "CREATE USER c##practica1 IDENTIFIED BY practica1 2 ;" and the response is "Usuario creado." followed by "SQL> GRANT ALL PRIVILEGES TO c##practica1;" and the response "Concesión terminada correctamente." The prompt "SQL> " is shown at the end.

```
SQL*Plus: Release 21.0.0.0.0 - Production on S8b Ene 7 11:41:55 2023
Version 21.3.0.0.0

Copyright (c) 1982, 2021, Oracle. All rights reserved.

Introduzca el nombre de usuario: SYS AS SYSDBA
Introduzca la contraseña:

Conectado a:
Oracle Database 21c Express Edition Release 21.0.0.0.0 - Production
Version 21.3.0.0.0

SQL> CREATE USER c##practica1 IDENTIFIED BY practica1
2 ;

Usuario creado.

SQL> GRANT ALL PRIVILEGES TO c##practica1;

Concesión terminada correctamente.

SQL> 
```

Creo todas las tablas en el orden siguiente:

CREATE TABLE EMPLEADO(

dniEmpleado VARCHAR(9) NOT NULL,

nomEmpleado VARCHAR(15) NOT NULL,

apeEmpleado VARCHAR(20) NOT NULL,

dirEmpleado VARCHAR(30) NOT NULL,

ciudadEmpleado VARCHAR(15) NOT NULL,

cpEmpleado VARCHAR(5) NOT NULL,

telefEmpleado VARCHAR(11) NOT NULL,

fechaAlta DATE NOT NULL,

categoria VARCHAR(10),

salario NUMBER(6,2),

CONSTRAINT pk_dniEmpleado PRIMARY KEY(dniEmpleado),

CONSTRAINT chk_salario CHECK(salario > 0));

CREATE TABLE CAMARERO(

```
dniCamarero VARCHAR2(9) NOT NULL,  
turno VARCHAR2(6) NOT NULL,  
añosExperiencia NUMBER(2),  
dniEncargado VARCHAR2(9) NOT NULL,  
CONSTRAINT pk_dniCamarero PRIMARY KEY(dniCamarero),  
CONSTRAINT fk_dniCamarero FOREIGN KEY(dniCamarero) REFERENCES EMPLEADO(dniEmpleado),  
CONSTRAINT fk_dniEncargado FOREIGN KEY(dniEncargado) REFERENCES CAMARERO(dniCamarero),  
CONSTRAINT chk_turno CHECK(turno in ('mañana' || 'tarde' || 'noche')),  
CONSTRAINT chk_añosExperiencia CHECK(añosExperiencia >= 0));
```

CREATE TABLE COCINERO(

```
dniCocinero VARCHAR2(9) NOT NULL,  
puesto VARCHAR2(10),  
especialidad VARCHAR2(10),  
CONSTRAINT pk_dniCocinero PRIMARY KEY(dniCocinero),  
CONSTRAINT fk_dniCocinero FOREIGN KEY (dniCocinero) REFERENCES EMPLEADO(dniEmpleado));
```

CREATE TABLE ADMINISTRATIVO(

```
dniAdministrativo VARCHAR2(9) NOT NULL,  
cargo VARCHAR2(10),  
CONSTRAINT pk_dniAdministrativo PRIMARY KEY(dniAdministrativo),  
CONSTRAINT fk_dniAdministrativo FOREIGN KEY (dniAdministrativo) REFERENCES EMPLEADO(dniEmpleado));
```

CREATE TABLE COMEDOR(

```
codComedor VARCHAR2(5) NOT NULL,  
nomComedor VARCHAR2(15) NOT NULL,  
capMax NUMBER(4) NOT NULL,  
numMesas NUMBER (2) NOT NULL,  
localizacion VARCHAR2(15) NOT NULL,  
CONSTRAINT pk_codComedor PRIMARY KEY(codComedor),  
CONSTRAINT chk_capMax CHECK(capMax > 0),  
CONSTRAINT chk_numMesas CHECK(numMesas > 0));
```


CREATE TABLE MESA(

```
codMesa VARCHAR2(5) NOT NULL,  
codComedor VARCHAR2(5) NOT NULL,  
asientos NUMBER(2) NOT NULL,  
dniEmpleado VARCHAR2(9) NOT NULL,  
CONSTRAINT pk_codMesa_codComedor PRIMARY KEY(codMesa, codComedor),  
CONSTRAINT fk_codComedor_Mesa FOREIGN KEY(codComedor) REFERENCES COMEDOR(codComedor),  
CONSTRAINT fk_dniCamarero_Mesa FOREIGN KEY(dniEmpleado) REFERENCES CAMARERO(dniCamarero));
```

CREATE TABLE RESERVA(

```
numReserva VARCHAR2(5) NOT NULL,  
fechaReserva DATE,  
nomReserva VARCHAR2(15) NOT NULL,  
fechaComida DATE NOT NULL,  
turnoComida VARCHAR2(5) NOT NULL,  
numPersonas NUMBER(3) NOT NULL,  
otros VARCHAR2(50),  
codMesa VARCHAR2(5) NOT NULL,  
codComedor VARCHAR2(5) NOT NULL,  
CONSTRAINT pk_numReserva PRIMARY KEY(numReserva),  
CONSTRAINT fk_mesaComedor_Reserva FOREIGN KEY(codMesa, codComedor) REFERENCES MESA(codMesa, codComedor),  
CONSTRAINT chk_numPersonas CHECK(numPersonas > 0));
```

CREATE TABLE FACTURA(

```
numFactura VARCHAR2(5) NOT NULL,  
fechaFactura DATE,  
numReserva VARCHAR2(5) NOT NULL,  
CONSTRAINT pk_numFactura PRIMARY KEY(numFactura),  
CONSTRAINT fk_numReserva_Factura FOREIGN KEY(numReserva) REFERENCES RESERVA(numReserva));
```

CREATE TABLE PLATO(

```
codPlato VARCHAR2(5) NOT NULL,  
nomPlato VARCHAR2(15) NOT NULL,  
tipoPlato VARCHAR2(15),  
despcripcionPlato VARCHAR2(50),  
precioPlato NUMBER(3,2) NOT NULL,  
CONSTRAINT pk_codPlato PRIMARY KEY(codPlato),  
CONSTRAINT chk_precioPlato CHECK(precioPlato > 0));
```

CREATE TABLE PROVEEDOR(

```
codProveedor VARCHAR2(5) NOT NULL,  
nomProveedor VARCHAR2(15) NOT NULL,  
dirProveedor VARCHAR2(30) NOT NULL,  
cpProveedor VARCHAR2(5) NOT NULL,  
telefProveedor VARCHAR2(11) NOT NULL,  
faxProveedor VARCHAR2(11),  
personaContacto VARCHAR2(20) NOT NULL,  
CONSTRAINT pk_codProveedor PRIMARY KEY(codProveedor));
```

CREATE TABLE PRODUCTO(

```
codProducto VARCHAR2(5) NOT NULL,  
descripcionProducto VARCHAR2(50),  
stock NUMBER(5,3) NOT NULL,  
unidadBase VARCHAR2(10) NOT NULL,  
categoriaProducto VARCHAR2(10),  
codProveedor VARCHAR2(5) NOT NULL,  
CONSTRAINT pk_codProducto PRIMARY KEY(codProducto),  
CONSTRAINT fk_codProveedor_Producto FOREIGN KEY(codProveedor) REFERENCES PROVEEDOR(codProveedor),  
CONSTRAINT chk_stock CHECK(stock > 0));
```

CREATE TABLE contiene(

codPlato VARCHAR2(5) NOT NULL,
numFactura VARCHAR2(5) NOT NULL,
unidadPlato VARCHAR2(5) NOT NULL,
CONSTRAINT pk_codPlato_numFactura PRIMARY KEY(codPlato, numFactura),
CONSTRAINT fk_codPlato_contiene FOREIGN KEY(codPlato) REFERENCES PLATO(codPlato),
CONSTRAINT fk_numFactura_contiene FOREIGN KEY(numFactura) REFERENCES FACTURA(numFactura),
CONSTRAINT chk_unidadPlato CHECK(unidadPlato > 0));

CREATE TABLE cocinados_con(

codPlato VARCHAR2(5) NOT NULL,
codProducto VARCHAR2(5) NOT NULL,
CONSTRAINT pk_codPlato_codProducto PRIMARY KEY(codPlato, codProducto),
CONSTRAINT fk_codPlato_cocinado FOREIGN KEY(codPlato) REFERENCES PLATO(codPlato),
CONSTRAINT fk_codProducto_cocinado FOREIGN KEY(codProducto) REFERENCES PRODUCTO(codProducto));

CREATE TABLE solicita(

numReserva VARCHAR2(5) NOT NULL,
codMesa VARCHAR2(5) NOT NULL,
codComedor VARCHAR2(5) NOT NULL,
CONSTRAINT pk_reserva_mesa_comedor PRIMARY KEY(numReserva, codMesa, codComedor),
CONSTRAINT fk_numReserva FOREIGN KEY(numReserva) REFERENCES RESERVA(numReserva),
CONSTRAINT fk_mesaComedor_solicita FOREIGN KEY(codMesa, codComedor) REFERENCES MESA(codMesa, codComedor));

Modificaciones:

--Tabla EMPLEADO--

-- Añade un índice que facilite búsquedas frecuentes por Apellidos y Nombre sin duplicados--

```
CREATE UNIQUE INDEX index_nombre  
ON EMPLEADO(apeEmpleado, nomEmpleado);
```

-- El Restaurante se ha inaugurado el día 1 de Junio de 2022. Comprueba que la fecha de alta de los empleados no sea anterior a esa fecha --

```
ALTER TABLE EMPLEADO  
ADD CONSTRAINT chk_fechaAlta CHECK(fechaAlta >= '2022-07-01');
```

--Tabla CAMARERO--

-- El turno de trabajo sólo puede tomar 3 valores: mañana, tarde y noche. Añade esa restricción --

--teniendo en cuenta que un camarero puede tener más de un turno (no utilices CHECK) --

--No se me ocurre otra forma--

```
ALTER TABLE CAMARERO  
ADD CONSTRAINT chk_turno CHECK(turno in ('mañana' || 'tarde' || 'noche'));
```

--Tabla PROVEEDOR--

--Añade las columnas Apellidos y Nombre entre el código y la dirección--

--Para poder colocar esas columnas en ese orden, solo se me ha ocurrido borrar las columnas ya existentes y volver a insertarlas de nuevo en el orden requerido--

```
ALTER TABLE PROVEEDOR  
DROP COLUMN personaContacto;  
ALTER TABLE PROVEEDOR  
DROP COLUMN faxProveedor;  
ALTER TABLE PROVEEDOR  
DROP COLUMN telefProveedor;  
ALTER TABLE PROVEEDOR  
DROP COLUMN cpProveedor;  
ALTER TABLE PROVEEDOR
```

DROP COLUMN dirProveedor;

ALTER TABLE PROVEEDOR

DROP COLUMN nomProveedor;

ALTER TABLE PROVEEDOR

ADD nombre VARCHAR2(15) NOT NULL;

ALTER TABLE PROVEEDOR

ADD apellidos VARCHAR2(20) NOT NULL;

ALTER TABLE PROVEEDOR

ADD nomProveedor VARCHAR2(15) NOT NULL;

ALTER TABLE PROVEEDOR

ADD dirProveedor VARCHAR2(30) NOT NULL;

ALTER TABLE PROVEEDOR

ADD cpProveedor VARCHAR2(5) NOT NULL;

ALTER TABLE PROVEEDOR

ADD telefProveedor VARCHAR2(11) NOT NULL;

ALTER TABLE PROVEEDOR

ADD faxProveedor VARCHAR2(11);

ALTER TABLE PROVEEDOR

ADD personaContacto VARCHAR2(20) NOT NULL;

--Tabla PRODUCTO--

--Añade un índice por Categoría--

CREATE UNIQUE INDEX index_categoria

ON PRODUCTO(categoriaProducto);

--Muestra todos los índices de la tabla--

SELECT * FROM all_indexes WHERE TABLE_NAME = 'PRODUCTO';

--Añade una restricción en la tabla, de forma que el Stock sea entero de 4 cifras, sin signo y que no admita nulos--

ALTER TABLE PRODUCTO

MODIFY stock number(4);

--Ya tengo la restricción creada al hacer la table, pero en el caso de no tenerla, se crearía así--

ALTER TABLE PRODUCTO

ADD CONSTRAINT chk_stock CHECK(stock > 0),

--Borra el índice que acabas de crear--

DROP INDEX index_categoria;

--Tabla PLATO--

--Borra la tabla PROVEEDORES. ¿Qué ocurre?. Borra previamente las claves ajenas--

DROP TABLE PROVEEDOR;

--No se puede borrar. Primero tenemos que borrar la FK de la tabla PRODUCTO--

ALTER TABLE PRODUCTO

DROP COLUMN codProveedor;

DROP TABLE PROVEEDOR;

--Base de datos Restaurante Villagarcia de Arriba--

-- Borra la base de datos --

--Borro las tablas en orden--

DROP TABLE solicita;

DROP TABLE cocinados_con;

DROP TABLE contiene;

DROP TABLE PRODUCTO;

DROP TABLE PLATO;

DROP TABLE FACTURA;

DROP TABLE RESERVA;

DROP TABLE MESA;

DROP TABLE COMEDOR;

DROP TABLE ADMINISTRATIVO;

DROP TABLE COCINERO;

DROP TABLE CAMARERO;

DROP TABLE EMPLEADO;