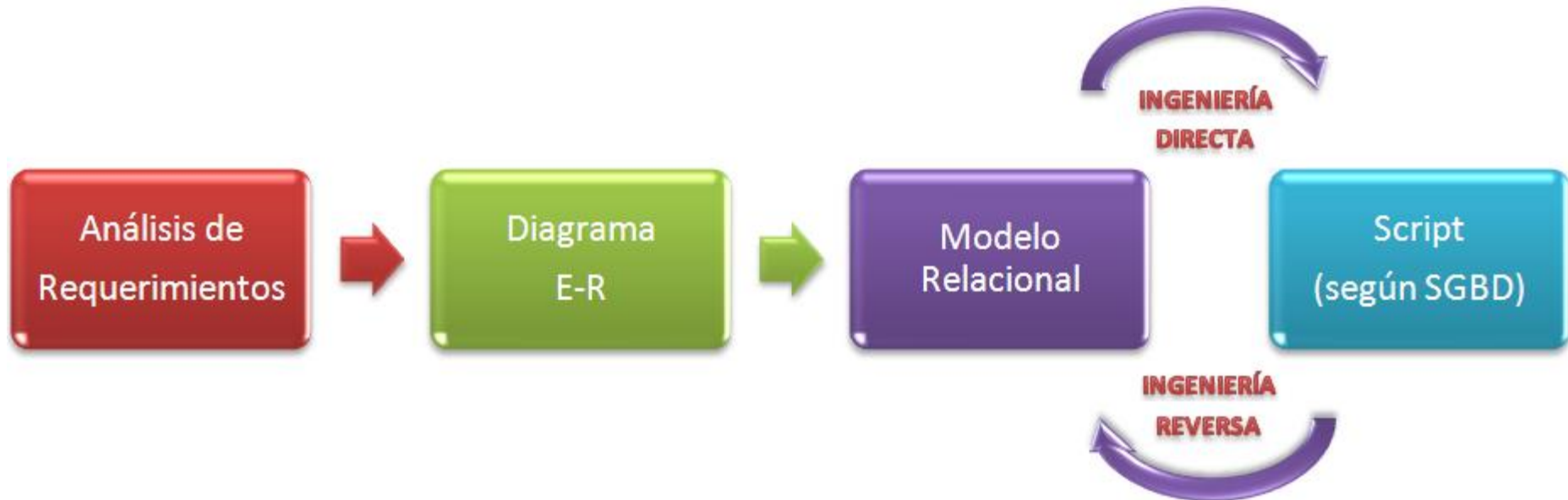


EL MODELADO DE LA BASE DE DATOS

Cómo crear de una manera eficiente una estructura de base de datos que proporcione a nuestra aplicación un rendimiento profesional.

Pasos para el diseño de una base de datos



Análisis de Requerimientos del Sistema

- * Establecer el alcance, los objetivos y requisitos de la BD.
- * Contacto estrecho con el cliente.
- * Identificación de las funciones e interfaces.
- * Especificación del flujo, estructura y asociatividad de la información.
- * Documento formal de los requerimientos.
- * A quien va dirigido el sistema.
- * Que funciones debe tener cada tabla.

Problema (Empresa)

- * Una empresa vende productos a varios clientes.
- * Se necesita conocer los datos personales de los clientes (nombre, apellidos, curp, dirección y fecha de nacimiento).
- * Cada producto tiene un nombre y un código, así como un precio unitario.
- * Un cliente puede comprar varios productos a la empresa, y un mismo producto puede ser comprado por varios clientes.
- * Los productos son suministrados por diferentes proveedores. Se debe tener en cuenta que un producto sólo puede ser suministrado por un proveedor, y que un proveedor puede suministrar diferentes productos.
- * De cada proveedor se desea conocer su rfc, nombre y dirección.

Diseño Conceptual (Modelo Entidad-Relación)

- * Herramienta para los diseñadores de BD.
- * Permite construir el Modelo Conceptual.
- * Utiliza técnica grafica como herramienta de diseño (DER).
- * Es expresada en términos de entidades, relaciones entre entidades y los atributos como propiedades de las entidades.

Conceptos DER

PERSONA

Ejemplo de entidad

PERSONA

Nombre

Ejemplo de atributo

PERSONA



TRABAJO

Ejemplo de relacion

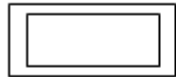
Metodología para construir un DER

- * Identificar posibles entidades.
- * Generar un DER inicial.
- * Identificar los atributos, es decir, los campos que queremos guardar de cada entidad.
- * Buscar la clave.
- * Agregar la cardinalidad.

Representación E-R



Entidad



Entidad débil



Relación



Relación de identificación



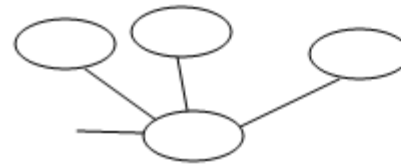
Atributo



Atributo clave



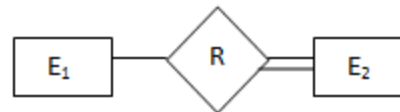
Atributo multivalor



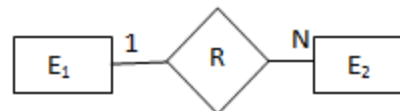
Atributo compuesto




Atributo derivado



Participación total de E_2 en R



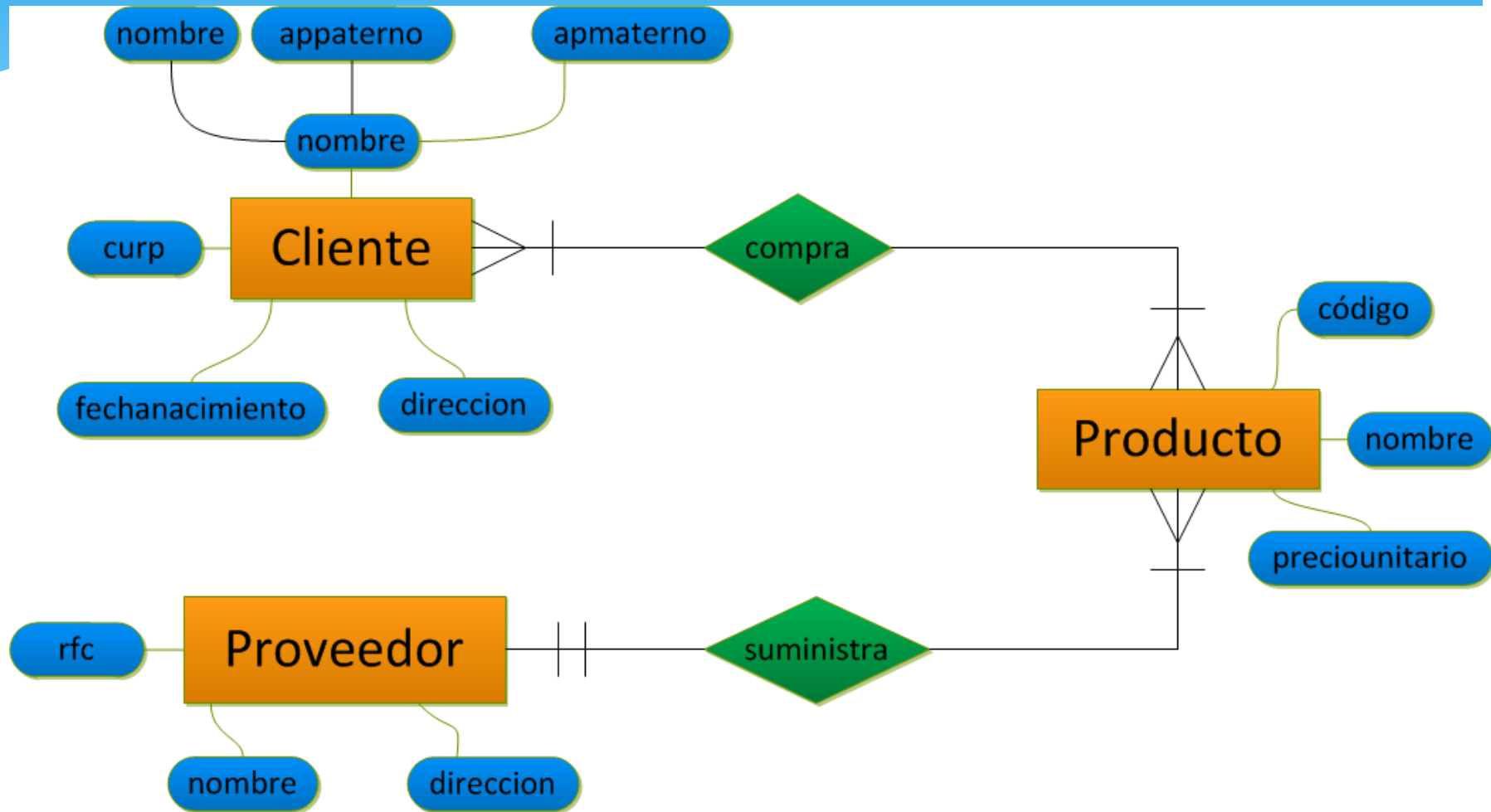
Razón de cardinalidad 1:N para $E_1:E_2$ en R

- 
- ▶ *Entidades independientes (fuertes)*: Las instancias de una entidad pueden existir por sí, sin participar en alguna asociación
 - ▶ *Entidades dependientes (débiles)*: Cada instancia de una entidad tiene que participar en una asociación para existir. Las llaves son importadas de una entidad independiente

Identificadores

- ▶ Identificador de entidad
 - Uno o más atributos que identifican de manera única cada instancia de un tipo de entidad
- ▶ Identificador de asociación
 - Un medio de identificar cada instancia de asociación
 - Usualmente un identificador compuesto consiste de los identificadores de dos o más tipos de entidades que éste asocia.
- ▶ Identificador parcial
 - Es el identificador principal de una entidad dependiente, que en conjunción con el identificador de la entidad independiente, generan la llave primaria.

DER Empresa



Diseño Conceptual (Modelo Relacional)

El objetivo del diseño lógico es convertir los esquemas conceptuales en un esquema lógico que se ajuste al modelo de SGBD sobre el que se vaya a implementar el sistema.

Los modelos de bases de datos más extendidos son el modelo relacional, el modelo de red y el modelo jerárquico. El modelo orientado a objetos es también muy popular, pero no existe un modelo estándar orientado a objetos.

Construyendo el modelo lógico

- * Cada tipo de entidad se transforma en una tabla, con todos sus atributos, el cual uno de ellos es la PK y este será el índice de la tabla.
- * Una entidad débil origina una tabla, con una clave propia de la entidad débil, unida a la clave de la entidad de la que depende.
- * Las relaciones 1:1 originan que una de las dos tablas que se ven involucradas se añada el identificador de la otra

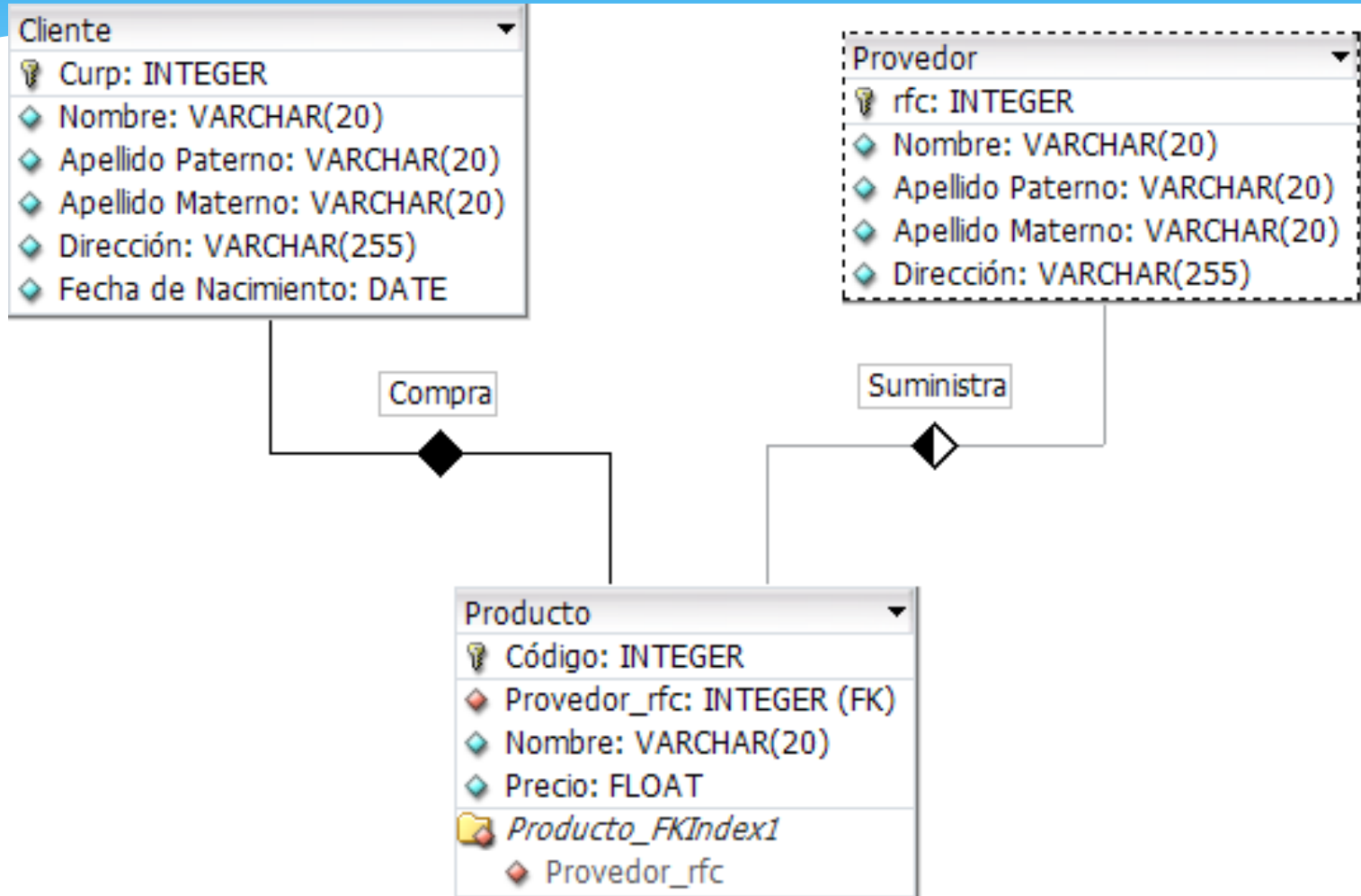
- * Las relaciones 1:N se resuelven introduciendo en la parte N el código identificativo de la parte 1.
- * Las relaciones N:M originan una nueva tabla formada por los atributos propios que puedan tener cada relación, las claves de las dos tablas que entran en juego.

Cliente(curp, dirección, fechaNac, nombre, ApPaterno, ApMaterno)

ClienteProd(curp, codigo)

Producto(código, nombre, precioUnitario, rfc)

Proveedor(rfc, nombre, dirección)



Diseño Físico

Mientras que en el diseño lógico se especifica qué se guarda, en el diseño físico se especifica cómo se guarda. Para ello, el diseñador debe conocer muy bien toda la funcionalidad del SGBD concreto que se vaya a utilizar y también el sistema informático sobre el que éste va a trabajar.

```
CREATE TABLE Cliente (  
  Curp INTEGER UNSIGNED NOT NULL AUTO_INCREMENT, Nombre VARCHAR(20)  
  NULL, Apellido Paterno VARCHAR(20) NULL, Apellido Materno VARCHAR(20)  
  NULL, Dirección VARCHAR(255) NULL, Fecha de Nacimiento DATE NULL,  
  PRIMARY KEY(Curp)  
);
```

```
CREATE TABLE Producto (  
  Código INTEGER UNSIGNED NOT NULL AUTO_INCREMENT, Proveedor_rfc  
  INTEGER UNSIGNED NOT NULL, Nombre VARCHAR(20) NULL, Precio FLOAT  
  NULL, PRIMARY KEY(Código), INDEX Producto_FKIndex1(Proveedor_rfc)  
);
```

```
CREATE TABLE Proveedor (  
  rfc INTEGER UNSIGNED NOT NULL AUTO_INCREMENT, Nombre VARCHAR(20)  
  NULL, Apellido Paterno VARCHAR(20) NULL, Apellido Materno VARCHAR(20)  
  NULL, Dirección VARCHAR(255) NULL, PRIMARY KEY(rfc)  
);
```

Un Ejemplo Modelado

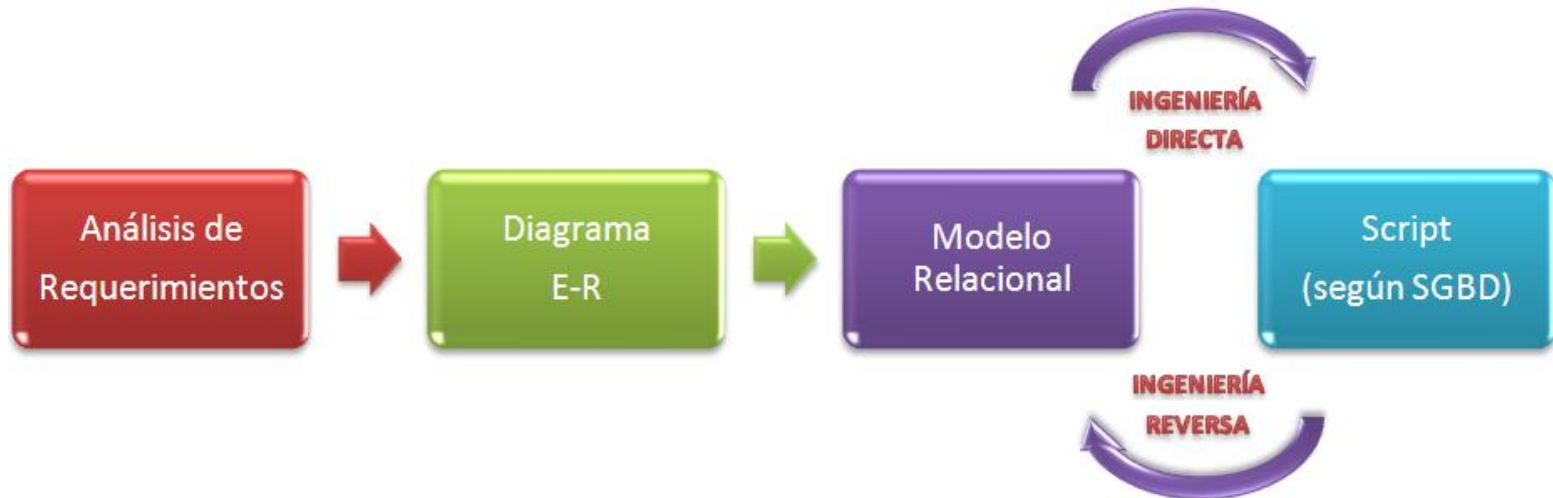


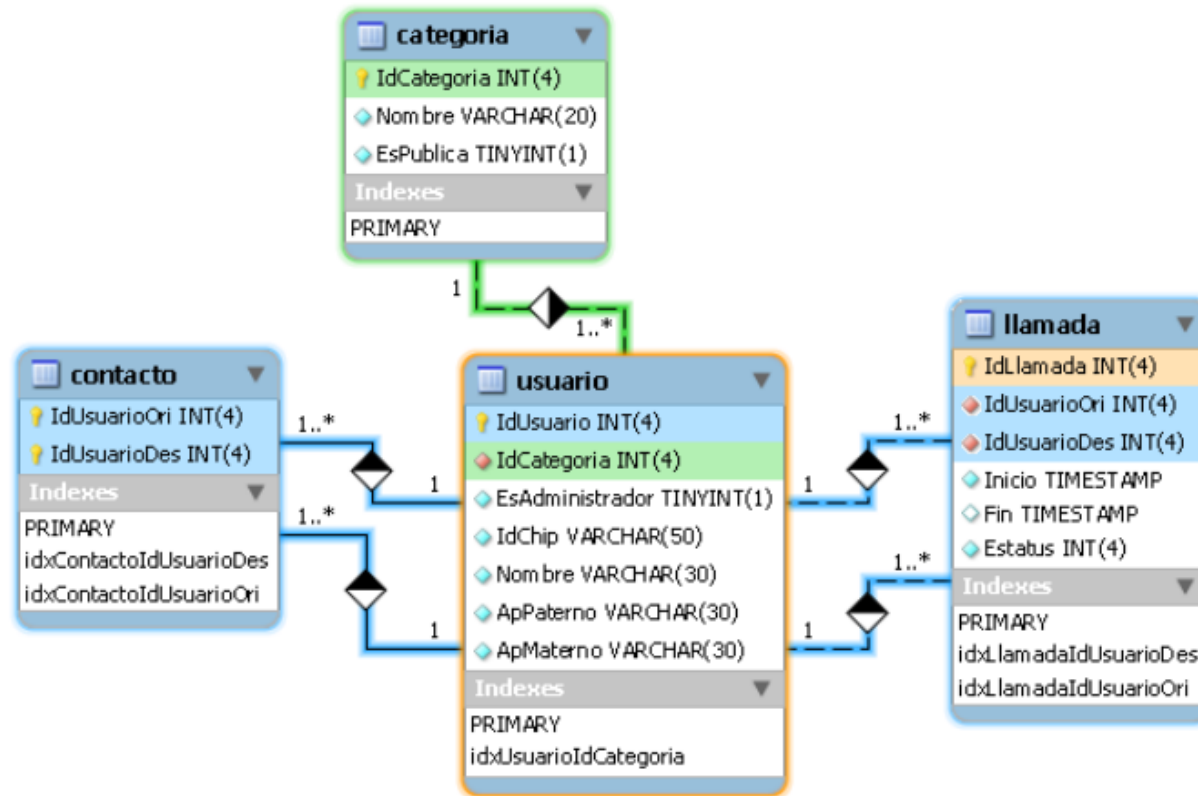
Diagrama Entidad Relación

Diagrama E-R.



Modelo Relacional

Modelo Relacional



Script (categoría, usuario)

```
DROP SCHEMA IF EXISTS `llamadas`;
CREATE SCHEMA IF NOT EXISTS `llamadas` DEFAULT CHARACTER SET utf8 ;
USE `llamadas`;

-----
-- Table `llamadas`.`categoria`
-----

CREATE TABLE IF NOT EXISTS `llamadas`.`categoria` (
  `IdCategoria` INT(4) UNSIGNED NOT NULL AUTO_INCREMENT ,
  `Nombre` VARCHAR(20) NOT NULL DEFAULT '' ,
  `EsPublica` TINYINT(1) UNSIGNED NOT NULL DEFAULT '0' ,
  PRIMARY KEY (`IdCategoria`) )
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;

-----
-- Table `llamadas`.`usuario`
-----

CREATE TABLE IF NOT EXISTS `llamadas`.`usuario` (
  `IdUsuario` INT(4) UNSIGNED NOT NULL AUTO_INCREMENT ,
  `IdCategoria` INT(4) UNSIGNED NOT NULL DEFAULT '0' ,
  `EsAdministrador` TINYINT(1) NOT NULL DEFAULT '0' ,
  `IdChip` VARCHAR(50) NOT NULL DEFAULT '' ,
  `Nombre` VARCHAR(30) NOT NULL DEFAULT '' ,
  `ApPaterno` VARCHAR(30) NOT NULL DEFAULT '' ,
  `ApMaterno` VARCHAR(30) NOT NULL DEFAULT '' ,
  PRIMARY KEY (`IdUsuario`) ,
  INDEX `idxUsuarioIdCategoria` (`IdCategoria` ASC) ,
  CONSTRAINT `idxUsuarioIdCategoria`
    FOREIGN KEY (`IdCategoria`)
    REFERENCES `llamadas`.`categoria` (`IdCategoria`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;
```

Script (contacto)

```
-- Table `llamadas`.`contacto`  
-----  
CREATE TABLE IF NOT EXISTS `llamadas`.`contacto` (  
  `IdUsuarioOri` INT(4) UNSIGNED NOT NULL AUTO_INCREMENT ,  
  `IdUsuarioDes` INT(4) UNSIGNED NOT NULL DEFAULT '0' ,  
  PRIMARY KEY (`IdUsuarioOri`, `IdUsuarioDes`) ,  
  INDEX `idxContactoIdUsuarioDes` (`IdUsuarioDes` ASC) ,  
  INDEX `idxContactoIdUsuarioOri` (`IdUsuarioOri` ASC) ,  
  CONSTRAINT `idxContactoIdUsuarioDes`  
    FOREIGN KEY (`IdUsuarioDes` )  
    REFERENCES `llamadas`.`usuario` (`IdUsuario` )  
    ON DELETE CASCADE  
    ON UPDATE CASCADE ,  
  CONSTRAINT `idxContactoIdUsuarioOri`  
    FOREIGN KEY (`IdUsuarioOri` )  
    REFERENCES `llamadas`.`usuario` (`IdUsuario` )  
    ON DELETE CASCADE  
    ON UPDATE CASCADE)  
  
ENGINE = InnoDB  
DEFAULT CHARACTER SET = utf8;
```


Script (llamada)

```
-- Table `llamadas`.`llamada`  
  
CREATE TABLE IF NOT EXISTS `llamadas`.`llamada` (  
  `IdLlamada` INT(4) UNSIGNED NOT NULL AUTO_INCREMENT ,  
  `IdUsuarioOri` INT(4) UNSIGNED NOT NULL DEFAULT '0' ,  
  `IdUsuarioDes` INT(4) UNSIGNED NOT NULL DEFAULT '0' ,  
  `Inicio` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ,  
  `Fin` TIMESTAMP NULL DEFAULT NULL ,  
  `Estatus` INT(4) UNSIGNED NOT NULL DEFAULT '0' ,  
  PRIMARY KEY (`IdLlamada`) ,  
  INDEX `idxLlamadaIdUsuarioDes` (`IdUsuarioDes` ASC) ,  
  INDEX `idxLlamadaIdUsuarioOri` (`IdUsuarioOri` ASC) ,  
  CONSTRAINT `idxLlamadaIdUsuarioDes`  
    FOREIGN KEY (`IdUsuarioDes` )  
    REFERENCES `llamadas`.`usuario` (`IdUsuario` )  
    ON DELETE CASCADE  
    ON UPDATE CASCADE ,  
  CONSTRAINT `idxLlamadaIdUsuarioOri`  
    FOREIGN KEY (`IdUsuarioOri` )  
    REFERENCES `llamadas`.`usuario` (`IdUsuario` )  
    ON DELETE CASCADE  
    ON UPDATE CASCADE)  
  
ENGINE = InnoDB  
DEFAULT CHARACTER SET = utf8;
```

Normalización

- * Se trata de evitar problemas entre inserciones, actualizaciones y borrado de elementos en las tablas de la BD.

Primera forma normal

- * Separar datos en tablas separadas, de manera que los datos en cada tabla sean de un tipo similar.
- * Dar a cada tabla una clave primaria un identificador o etiqueta única
- * Ejemplo: Se tiene la tabla EMPLEADO con los campos Nombre, Edad, Alojamiento, Responsable, Dirección, Oficio1, Oficio2, Oficio3

Primera forma normal

- * El limitar los oficios a tres posibilidades puede ocasionar problemas.
- * La solución sería situar los oficios en una tabla separada (OFICIO), con una fila por nombre, oficio y descripción

Segunda forma normal

- * Se centra en aislar los datos que sólo dependen de una parte de la clave.
- * En la solución, la larga descripción del oficio sólo aparece una vez, en la de la 1FN las descripciones se repiten para cada trabajador que tenga ese oficio. Cuando el último trabajador con oficio X se marca, la descripción del oficio del herrero se desvanece.
- * Con la solución, se pueden añadir oficios incluso antes de que haya un trabajador con él.

Tercera forma normal

- * Implica deshacerse de cualquier cosa de las tablas que no dependa únicamente de la clave primaria. La información de los trabajadores sobre el alojamiento depende de dónde estén viviendo, pero el patrón de la posada y su dirección son independientes de si el trabajador vive ahí o no.
- * En la solución se utiliza por conveniencia una versión taquigráfica del nombre de la casa de alojamiento como llave primaria

PROBLEMA 2

Se está diseñando una BD para gestionar los pedidos de medicamentos que realizan las farmacias pertenecientes a una red. Los supuestos semánticos que deben contemplarse son:

- a) Un pedido de medicamentos se identifica por un código de pedido (P)
- b) Cada pedido además se realiza en una fecha (D) y se caracteriza por la farmacia que lo efectúa (F)
- c) De cada farmacia se almacena su dirección (A), localidad (L) y NIF del propietario (N).
- d) Un propietario puede poseer varias farmacias.
- e) Cada propietario tiene un nombre (M) que puede no ser único
- f) A cada propietario le corresponde un tipo de cliente (T) que puede ser "A", "B" o "C".
- g) En cada pedido se incluye una serie de productos identificados por su código (C)
- h) Cada producto tiene un nombre (B) de tal manera que no pueden existir dos productos con el mismo nombre.
- i) De cada producto en cada pedido se almacena una cantidad solicitada (Q) a un determinado precio (R). Este precio puede ser distinto en diferentes pedidos.
- j) Los clientes de tipo "A" tienen un descuento global (G) en el importe final de la factura de un 15 %, los de tipo "B" del 10% y los de tipo "C" del 5 %.