# Práctica DBD

Diseño de una Base de Datos - Viajes Barceló

5 de Diciembre de 2023 - UPV/EHU

Nicolás Aguado

Daria Pavslaska

Agustín García

# Índice

# Análisis de Requerimientos

Se nos ha encargado realizar un diseño de bases de datos para la agencia de viajes "BARCELÓ". Después de unas provechosas conversaciones con los distintos departamentos de la agencia, se han sacado los siguientes datos en claro.

## Contexto

La empresa "BARCELÓ" comercializará viajes organizados. Estos viajes se venden en sucursales físicas (presentes en ciudades conocidas), por tanto, tendrán una serie de empleados. Cuando un cliente decide contratar un viaje (ya sea en una sucursal o mediante la web, nos dará igual) , se le pregunta información sobre sus vacunas y alergias que tiene, y consecuentemente, se le crea una reserva para el viaje en específico. En un mismo viaje los viajeros podrán visitar varias ciudades hospedándose en distintos alojamientos de los distintos grupos hoteleros. Los viajes siempre tienen una fecha de inicio, fecha de fin y precio fijo. Como es una empresa muy grande, Viajes Barceló tendrá sus propios guías. Éstos serán a la vez empleados de la empresa y contendrán la información de un viajero normal. Ayudarán a los viajeros durante su estancia y hablarán muchos idiomas. Claro, siendo una empresa muy grande, Viajes Barceló tendrá un programa de beneficios. Sus empleados si alcanzan una tasa de ventas, tendrán viajes gratis pagados por la empresa. Esto significa que habrá empleados que tendrán las características de un viajero. Así, estarán incentivados a trabajar más.

## Información a almacenar

Viajes Barceló necesitará almacenar información sobre sus empleados, sus clientes, los viajeros, los viajes que van a realizar, los alojamientos, los grupos hoteleros con los que trabaja, las ciudades que se visitan, los guías y las sucursales que tienen.  Además, hay que tener en cuenta que los grupos hoteleros como son empresas grandes podrán tener varios números de teléfono.

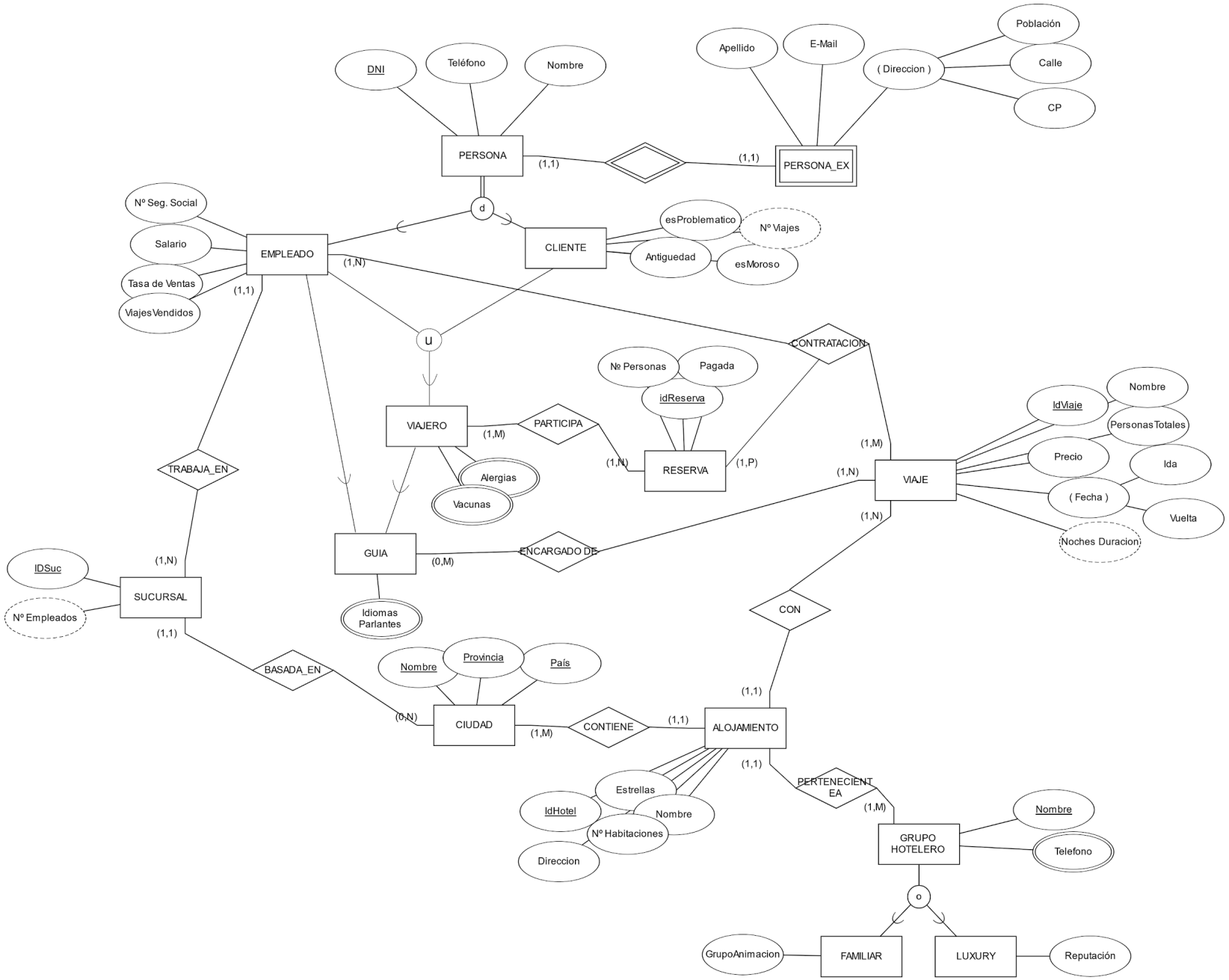
## Límites

Como Viajes Barceló es una empresa especializada, prefiere dedicar su tiempo en ofrecer los mejores viajes a sus clientes. Entonces, tiene la facturación y contabilidad delegada a una subcontrata y no se encargará de nada de gestión de pagos, etc. Además, tampoco se encarga de gestionar el soporte técnico telefónico.

Nos interesará además almacenar la información justa de cada alojamiento. La base de datos no debe rellenarse de información sobrante, sino que un alojamiento sólo tendrá sentido asociado a un grupo hotelero, a un viaje y en una misma ciudad.

# Esquema Conceptual (E-R++)

NICOLAS AGUADO -- AGUSTIN GARCIA -- DARIA PAVSLASKA



# Conversiones

## Entidad-Relación++ a Esquema Relacional

### Esquema

Persona(DNI,Teléfono, Nombre)

Persona_EX(DNI,Apellido, E-Mail, Calle, Población, CP) CE:Persona

Ciudad(Nombre, Provincia, Pais)

Sucursal(idSucursal, NombreCiu, ProvinciaCiu, PaisCiu) CE: Ciudad, CE:Ciudad, CE:Ciudad

Empleado(DNI,idSucursal, ViajesVendidos, Salario, TasaVentas, NSeguridadSocial) CE:Persona, CE: Sucursal

Cliente(DNI, Antiguedad, esMoroso, esProblematico, NViajes) CE: Persona

Viajero(DNI) CE: Empleado ó Cliente

Guia(DNI) CE: Empleado y Viajero

Viaje(idViaje, Precio, fechaIda, fechaVuelta, PersonasTotales, Nombre)

Reserva(idReserva, Pagada, NumPersonas)

Contratacion(DNI, idViaje, idReserva) CE: Empleado, CE: Viaje, CE: Reserva

Participa(DNI, idReserva) CE: Viajero, CE:Reserva

Encargado_De(DNI, idViaje) CE: Guia, CE: Viaje

Viajero_Alergias(DNI, Alergia) CE: Viajero

Viajero_Vacunas(DNI, Vacuna) CE: Viajero

Basada_En(IDSuc, Nombre,Provincia,Pais) CE: Sucursal, CE: Ciudad,CE: Ciudad,CE: Ciudad

Idiomas_Guia(DNI, Idioma) CE: Guia

Encargado_De(DNI, idViaje) CE: Guia, CE: Viaje

Grupo_Hotelero(Nombre)

Grupo_Hotelero_Telefono(Nombre, Telefono) CE: Grupo_Hotelero

Familiar(Nombre, GrupoAnimacion) CE: Grupo_Hotelero

Luxury(Nombre, Reputacion) CE: Grupo_Hotelero

Alojamiento(NombreCiu, ProvinciaCiu, PaisCiu, IdViaje, NombreGHotelero, IdHotel, DireccionHot, EstrellasHot,NHabHotel, NombreHot) CE: Ciudad, CE: Ciudad, CE: Ciudad, CE: Viaje, CE: Grupo_Hotelero

PertenecienteA(idHotel, Nombre) CE: Alojamiento, CE:GrupoHotelero

## Proceso

Para dar el paso al esquema relacional, hemos aplicado una serie de reglas relativas a las especializaciones, a las categorías y a las subclases compartidas:

- Herencia: Como norma general, y como no conocemos los casos de uso del cliente ni pruebas reales de rendimiento aplicamos la regla de herencia simple. Asumimos A como tipo de entidad padre y X,Y como tipos de entidad hijas. Aplicaremos una inserción de la clave de A (como clave) en los hijos, X e Y.

- Categoría: Como Empleado y Cliente tienen la misma clave, bastará con reflejarla como atributo en la categoría Viajero

- Subclase compartida: Como Empleado y Viajero tienen la misma clave, hemos aplicado el método que se haría con una especialización: Insertarla como clave en Guía

## Esquema Relacional Normalizado

Para este apartado, trabajaremos con un esquema un poco más laxo que el resultante de la transformación de E-R++ a relacional.

### DF Parcial

Según el esquema, la tabla Guía está definida de la siguiente forma:

- Guia(DNI, IdiomasParlantes)

Esto no cumple con la primera forma normal, ya que tenemos un atributo multivalor y esto provocaría tuplas repetidas para cada dni.

Lo solucionaremos dividiendo esta tabla guía en dos:

- Guia(DNI)
- idiomas_guia(DNI, Idioma)

Así, la segunda tabla almacenará todas esas tuplas.

### DF Transitiva

Para la dependencia transitiva, vamos a relajarel diseño y a introducir dependencias funcionales parciales y transitivas

Supongamos la tabla reserva, que almacena información de la reserva completa de un cliente, es decir, contiene su id de reserva, su información de vuelo y del hotel.

- reserva(dni, ida, vuelta, precio, idViaje, idhotel, numnoches)

Esto nos genera las siguientes dependencias funcionales:

- {DNI, idViaje} -> {ida,vuelta,precio,idhotel}
- {idViaje} -> {ida,vuelta}
- {idhotel} -> {estrellasHotel}

Este diseño nos presenta una dependencia funcional parcial, ya que hay atributos no-primos que dependen de una parte de la clave.

La solución es llevar esta dependencia a una nueva tabla y dejar en la tabla original la clave y aquellos atributos que dependan totalmente de esta.

Usaremos un atributo idViaje como clave extranjera entre ambas tablas y de esta forma lo pasaremos a segunda forma normal.

- reserva(dni, idViaje, precio, idhotel, estrellasHotel)

    - {idhotel} -> {estrellasHotel}
    - {dni, idViaje} -> {precio, idhotel}
- viaje(idViaje, ida, vuelta)
    - {idViaje} -> {ida,vuelta}

Este diseño sigue presentando un problema, y es que existen atributos no claves dependientes de otros atributos no claves; es decir, tenemos dependencias transitivas.

- {idhotel} -> {estrellasHotel}

Para solucionar esto, creamos otra tabla con idHotel como clave y estrellasHotel como atributo.

El diseño final normalizado nos quedaría así:

- reserva(dni, idViaje, precio, idhotel)
    - {dni, idViaje} -> {precio, idhotel}
- viaje(idViaje, ida, vuelta)
    - {idViaje} -> {ida,vuelta}
- hotel(idHotel, estrellasHotel)
    - {idHotel} -> {estrellasHotel}

# Vistas, Restricciones y Disparadores

## Vistas

### El guía y sus viajes

Desde el punto de vista de los guías, estos necesitarán saber qué viajes tiene asignados cada uno para hacer las planificaciones oportunas. Entonces, como no necesitarán acceso completo a la base de datos, se plantea una vista.

La vista tendrá que tener el DNI y del guía asignado, junto con el nombre del viaje, el número de personas que tienen reserva en el viaje y la fecha de ida y vuelta. Como los guías no son los encargados de la parte de gestión, esta vista será simplemente consultiva.

Definiremos así el SQL

```
CREATE VIEW GuiaInfo AS
SELECT DNI, NombreViaje, PersonasTotales, fechaIda, fechaVuelta
FROM (Encargado_De ED INNER JOIN Guia G ON ED.DNI = G.DNI)INNER JOIN
Viaje V ON ED.idViaje = V.idViaje
with read only;
```

### Hoteles de ciudad

Definiremos una vista para indicar cada destino donde exista al menos un hotel “low cost”, junto con los detalles del mismo. Consideramos low cost como un precio por noche menor a 30€.

```
CREATE VIEW DestinosLowCost
SELECT A.Nombre, A.Direccion, A.Estrellas, C.Nombre, C.Pais
FROM (Ciudad AS C INNER JOIN Contiene AS Cont ON C.Nombre =
Cont.Nombre AND C.Provincia = Cont.Provincia AND C.Pais = Cont.Pais)
INNER JOIN Alojamiento AS A ON Cont.idHotel = A.idHotel
WHERE A.PrecioPorNoche < 30
```

### Mejores empleados

Para premiar a los mejores empleados de todos los departamentos, se decidió hacer una vista de empleados dependiendo de los viajes vendidos en cada sucursal y, en consecuencia, aquellos trabajadores del departamento que vendieron la mayor cantidad de viajes son los mejores.

```
CREATE OR REPLACE VIEW MejoresEmple
SELECT DNI, e.IdSucursal,s.NombreCiudad, e.Salario,
max(e.ViajesVendidos)
FROM Empleado e natural join Sucursal s
GROUP BY s.IdSucursal
ORDER BY max(e.ViajesVendidos) DESC
```

## Restricciones de Integridad

### DNI Correcto

El principal método de identificación en la agencia de viajes “Barceló” es el DNI español. Entonces, este tendrá que ser correcto porque si no se pueden ocasionar muchos problemas. Como administradores del sistema, añadiremos una cláusula que compruebe esto. Los DNIs tendrán que contener 8 números + 1 letra.

```
ALTER TABLE Persona ADD CONSTRAINT DniCHK CHECK(
LENGHT(Persona.DNI)=9) ENABLE VALIDATE;
```

### Fechas correctas

Esta restricción asegura que siempre que se introduzca un viaje, este contará con un par de fechas coherentes; es decir, que la ida nunca sea después de la vuelta.

```
ALTER TABLE Viaje ADD CONSTRAINT dateCHK
CHECK(fechaIda < fechaVuelta)
ENABLE VALIDATE
```

### Salario correcto

Si un empleado vende más de 30 viajes, entonces su salario tiene que  ser superior a 1000.

$(v>30) \rightarrow (s>1000) = \neg (v>30) \vee (s>1000) = \neg ((v>30) \wedge \neg (s>1000))$

```
ALTER TABLE Empleado ADD CONSTRAINT salarioCHK
CHECK( NOT (
(ViajesVendidos > 30)
AND
(NOT(Salario > 1000))
)) ENABLE VALIDATE
```

## Disparadores

### Actualización de Personas en un Viaje

Para llevar la cuenta de cuántas personas se encuentran con una reserva en un viaje, definiremos un trigger. Este atributo, al crear el viaje por el operador del sistema, será inicializado a 0. Entonces, tendremos que escuchar en la tabla “Reserva” para ver cuándo se añaden o eliminan personas.

```
CREATE OR REPLACE TRIGGER PERSONASTRIGGER
AFTER INSERT OR UPDATE OR DELETE OF NumPersonas ON Reserva
FOR EACH ROW BEGIN
IF INSERTING THEN
UPDATE Viaje SET PersonasTotales = PersonasTotales +
:NEW.NumPersonas;
ELSIF UPDATING THEN
UPDATE Viaje SET PersonasTotales =
PersonasTotales + :NEW.NumPersonas - OLD.Numpersonas;
ELSE
UPDATE Viaje SET PersonasTotales =
PersonasTotales - OLD.Numpersonas;
END IF;
END;
```

### Número de noches

Este disparador calculará el número de noches en la tabla viajes, en función de la ida y la vuelta. Damos por supuesto que las fechas tienen coherencia porque habrán sido comprobadas con una de las restricciones de integridad propuestas anteriormente. Este trigger debe ser de tipo “compound” ya que va a modificar la tabla viaje y de no serlo, se daría el problema de tabla mutante.

```
CREATE OR REPLACE TRIGGER SetTripDuration
FOR INSERT ON VIAJE
COMPOUND TRIGGER
AFTER EACH ROW IS
BEGIN
UPDATE Viaje
SET NumeroNoches = DATEDIFF(d, :new.fechaIda, :new.fechaVuelta)
```

```
WHERE viajeId = :new.viajeId;

END AFTER EACH ROW

END;
```

### HotelesComfortables

Para organizar unas buenas vacaciones para los clientes, la empresa decidió cooperar únicamente con hoteles que tengan al menos dos estrellas. Por eso hemos decidido hacer un disparador que va a 'evitar' hoteles con menos de 2 estrellas dando un error.

```
CREATE OR REPLACE TRIGGER HotelesComfortables

BEFORE INSERT OR UPDATE OF Estrellas ON ALOJAMIENTO

referring NEW as nuevafila

FOR EACH ROW

BEGIN

WHEN nuevafila.Estrellas < 2

RAISE_APPLICATION_ERROR(-20002, 'El hotel debe tener al menos dos
estrellas')

END;
```