IMPLEMENTANDO NUESTRA IDEA

DAW 1 – Bases de Datos

Práctica 3

En esta nueva práctica de la unidad 2 vamos a realizar un nuevo modelo de nuestra empresa, donde vamos a mejorar con los últimos comentarios realizados nuestro modelo para pasarlo después al código en MySQL. Después, haremos ingeniería reversa para ver si es fiel a nuestro modelo.

Marco Valiente/ Rubén Calvo / Joaquín Ottonello / Mario Pila

Equipo de DAW1 para empresa TAXI KMRJ

Índice

Contenido

Planteamiento del Problema

Nuestro camino comienza con el modelo anterior que poseíamos, donde vamos a mejorar el modelo relacional añadiendo unas tablas intermedias, en base a los comentarios realizados por su previa evaluación.

a. Identificación de las mejoras

La sugerencia de mejoras es la siguiente:

–

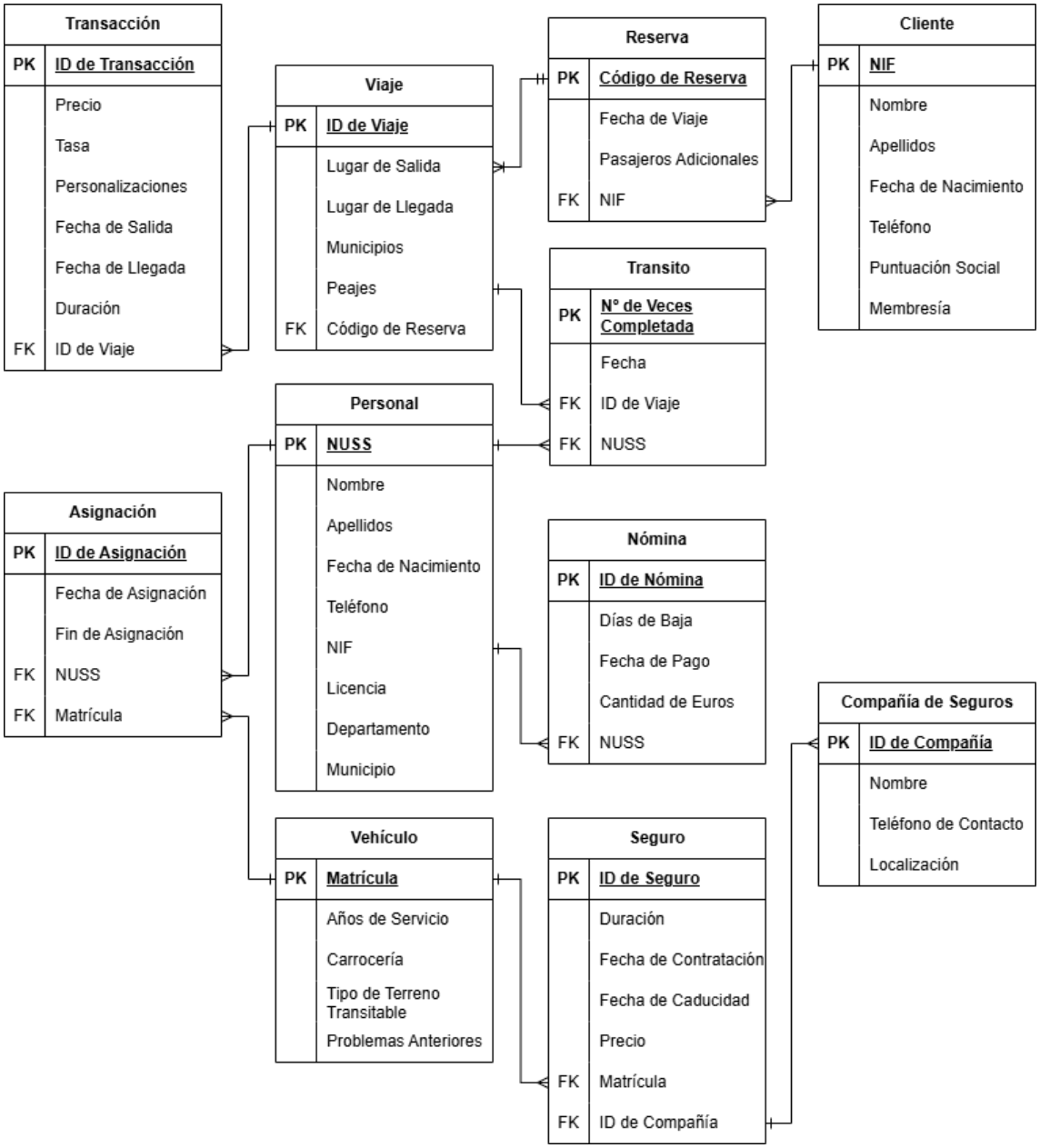
Señores, no deberíamos empezar el trabajo con un logo totalmente pixelado.

El cliente recorre la zona de operación, pero no recogemos los detalles básicos de la "carrera" (el viaje en taxi). Por otra parte, podemos viajar en taxi entre varias zonas, ¿cómo gestionaríamos eso?

Aunque podría ser gestionable el hecho de tener un taxista haciendo varios transites, acaba siendo no ideal. Así mismo, la palabra tarifa no corresponde fielmente a la idea que queremos transmitir.

Veamos ahora, como estas mejoras son implementadas, únicamente al modelo relacional.

b. Modelo Relacional Mejorado



Se han realizado los siguientes cambios. Las zonas de operación ya no existen, sino que ahora son viajes. En las licencias del personal es donde se van a incluir en qué comunidades autónomas se ha completado

c. Implementación en MySQL

Este es el código en MySQL:

```
DROP DATABASE IF EXISTS taxisKMRJ;
CREATE DATABASE IF NOT EXISTS taxisKMRJ;
USE taxisKMRJ;

CREATE TABLE IF NOT EXISTS cliente(
    nif CHAR(9),
    nombre VARCHAR(30),
    apellidos VARCHAR(50),
    fecha_nacimiento DATE,
    numero_telefono CHAR(12),
    puntuacion_social TINYINT,
    membresia BOOLEAN,
    CONSTRAINT pk_cliente PRIMARY KEY (nif)
);
CREATE TABLE IF NOT EXISTS personal(
    nuss CHAR(12),
    nombre VARCHAR(30),
    apellidos VARCHAR(50),
    fecha_nacimiento DATE,
    numero_telefono CHAR(12),
    nif CHAR(9),
    licencia VARCHAR(30),
    departamento VARCHAR(25),
    CONSTRAINT pk_personal PRIMARY KEY (nuss)
);
CREATE TABLE IF NOT EXISTS vehiculo(
    matricula CHAR(7),
    tiempo_servicio TIMESTAMP,
    carroceria VARCHAR(15),
    terreno_transitable VARCHAR(20),
    problemas_anteriores VARCHAR(80),
    CONSTRAINT pk_vehiculo PRIMARY KEY (matricula)
);
CREATE TABLE IF NOT EXISTS compania(
    id CHAR(9),
    nombre VARCHAR(40),
    telefono_contacto CHAR(12),
    localizacion VARCHAR(30),
    CONSTRAINT pk_compania PRIMARY KEY (id)
);
CREATE TABLE IF NOT EXISTS reserva(
    cod CHAR(9),
    fecha_viaje DATE,
    pasajeros_adicionales TINYINT,
```

```
    nif CHAR(9),
    CONSTRAINT pk_reserva PRIMARY KEY (cod),
    CONSTRAINT cliente_fk_reserva FOREIGN KEY (nif) REFERENCES
cliente(nif)
);
CREATE TABLE IF NOT EXISTS nomina(
    id CHAR(5),
    dias_baja TINYINT,
    fecha_pago DATETIME,
    cantidad_euros SMALLINT,
    nuss CHAR(12),
    CONSTRAINT pk_nomina PRIMARY KEY (id),
    CONSTRAINT persona_fk_nomina FOREIGN KEY (nuss) REFERENCES
personal(nuss)
);
CREATE TABLE IF NOT EXISTS viaje(
    id SMALLINT,
    lugar_salida VARCHAR(40),
    lugar_llegada VARCHAR(40),
    municipios VARCHAR(50),
    peajes BOOLEAN,
    cod CHAR(9),
    CONSTRAINT pk_viaje PRIMARY KEY (id),
    CONSTRAINT reserva_fk_viaje FOREIGN KEY (cod) REFERENCES reserva(cod)
);
CREATE TABLE IF NOT EXISTS transaccion(
    id MEDIUMINT,
    precio DECIMAL(5,2),
    tasa DECIMAL(3,2),
    personalizaciones VARCHAR(30),
    fecha_salida DATETIME,
    fecha_llegada DATETIME,
    duracion SMALLINT,
    id_viaje SMALLINT,
    CONSTRAINT pk_transaccion PRIMARY KEY (id),
    CONSTRAINT viaje_fk_transaccion FOREIGN KEY (id_viaje) REFERENCES
viaje(id)
);
CREATE TABLE IF NOT EXISTS asginacion(
    id SMALLINT,
    fecha_asignacion DATE,
    fin_asignacion DATE,
    nuss CHAR(12),
    matricula CHAR(7),
    CONSTRAINT pk_asignacion PRIMARY KEY (id),
    CONSTRAINT vehiculo_fk_asignacion FOREIGN KEY (matricula) REFERENCES
vehiculo(matricula),
    CONSTRAINT personal_fk_asignacion FOREIGN KEY (nuss) REFERENCES
personal(nuss)
```

```
);
CREATE TABLE IF NOT EXISTS transito(
    n_veces_completadas MEDIUMINT,
    fecha DATE,
    id_viaje SMALLINT,
    nuss CHAR(12),
    CONSTRAINT pk_asignacion PRIMARY KEY (n_veces_completadas),
    CONSTRAINT viaje_fk_transito FOREIGN KEY (id_viaje) REFERENCES
viaje(id),
    CONSTRAINT personal_fk_transito FOREIGN KEY (nuss) REFERENCES
personal(nuss)
);
CREATE TABLE IF NOT EXISTS seguro (
    id SMALLINT,
    duracion TINYINT,
    fecha_contratacion DATETIME,
    fecha_caducidad DATETIME,
    precio DECIMAL(5,2),
    matricula CHAR(7),
    id_compania CHAR(9),
    CONSTRAINT pk_seguro PRIMARY KEY (id),
    CONSTRAINT vehiculo_fk_seguro FOREIGN KEY (matricula) REFERENCES
vehiculo(matricula),
    CONSTRAINT compania_fk_seguro FOREIGN KEY (id_compania) REFERENCES
compania(id)
);
```

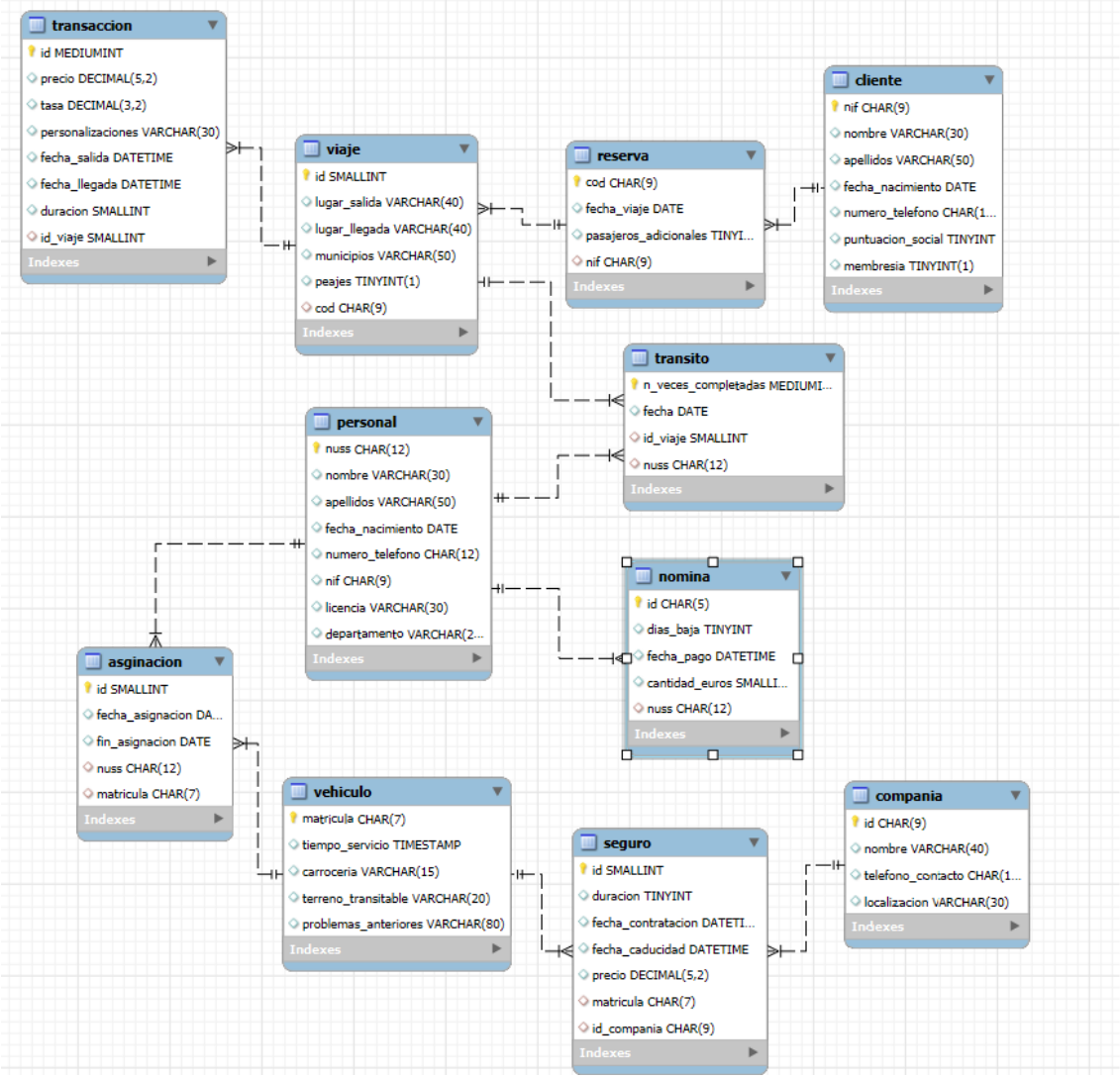
Ahora es interesante aportar una breve explicación para cada elección:

- NIF: Tiene 9 caracteres por norma.
- NUSS: Tiene 12 caracteres por norma.
- Matrícula: Tiene 7 caracteres por norma.
- Compañía (ID): Tiene un CHAR el cual nuestra propia empresa le va a asignar. Puede estar basado en datos que nos den más adelante.
- ID de Nómina: Lo dicho anteriormente, esta va a usar CHARS que asignaremos nosotros.
- IDs de otras tablas: Al ser operaciones, estas van a tener un ID asignado numéricamente a estas mismas, y, por tanto, se usan datos con apertura suficiente para que si nuestro negocio crece, se pueda adaptar a los nuevos datos.
- Uso de DATETIME y DATE: Se usa DATETIME cuando queremos tener los datos precisos de las fechas, y simplemente DATE cuando queramos solo el día, mes, año.
- Booleans: Algunas de estas operaciones son descritas como booleans ya que son opciones que pueden estar activas o no, y esto se refleja a través de un TINYINT de 1.
- TIMESTAMP: Para algunos casos particulares, especialmente duración, podemos usar TIMESTAMP directamente, como el tiempo de servicio que un coche ha servido, marcado con fechas para que aporte más información.
- Longitud: Algunos tienen unas longitudes mayores para poder ser más específicos o incluso añadir varias palabras o relaciones clave que sirvan de utilidadl.

d. Comprobación a través de ingeniería reversa

Tras estas aclaraciones, nos queda solo comprobar que, efectivamente, todo encaje con nuestra reversión.

Hacemos la prueba:



El resultado es favorable, ya que los datos encajan correctamente con nuestro modelo relacional mejorado.

Bibliografía

- Recurso del curso de Desarrollo de Aplicaciones Web, Documento de Introducción y tipos de datos.
- https://auladecroly.com/pluginfile.php/47045/mod_resource/content/4/2.1%20Introducci%C3%B3n%20y%20tipos%20de%20datos.pdf