UTN.BA

UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Estrategia

Gestión de Flota

ALUMNOS:

Domínguez, Francisco – 1715884

Horowitz, Ezequiel - 1715811

Molino, Tomás – 1717716

GRUPO 6 : Brogrammers3de5

Curso: K3521

Índice

Introducción

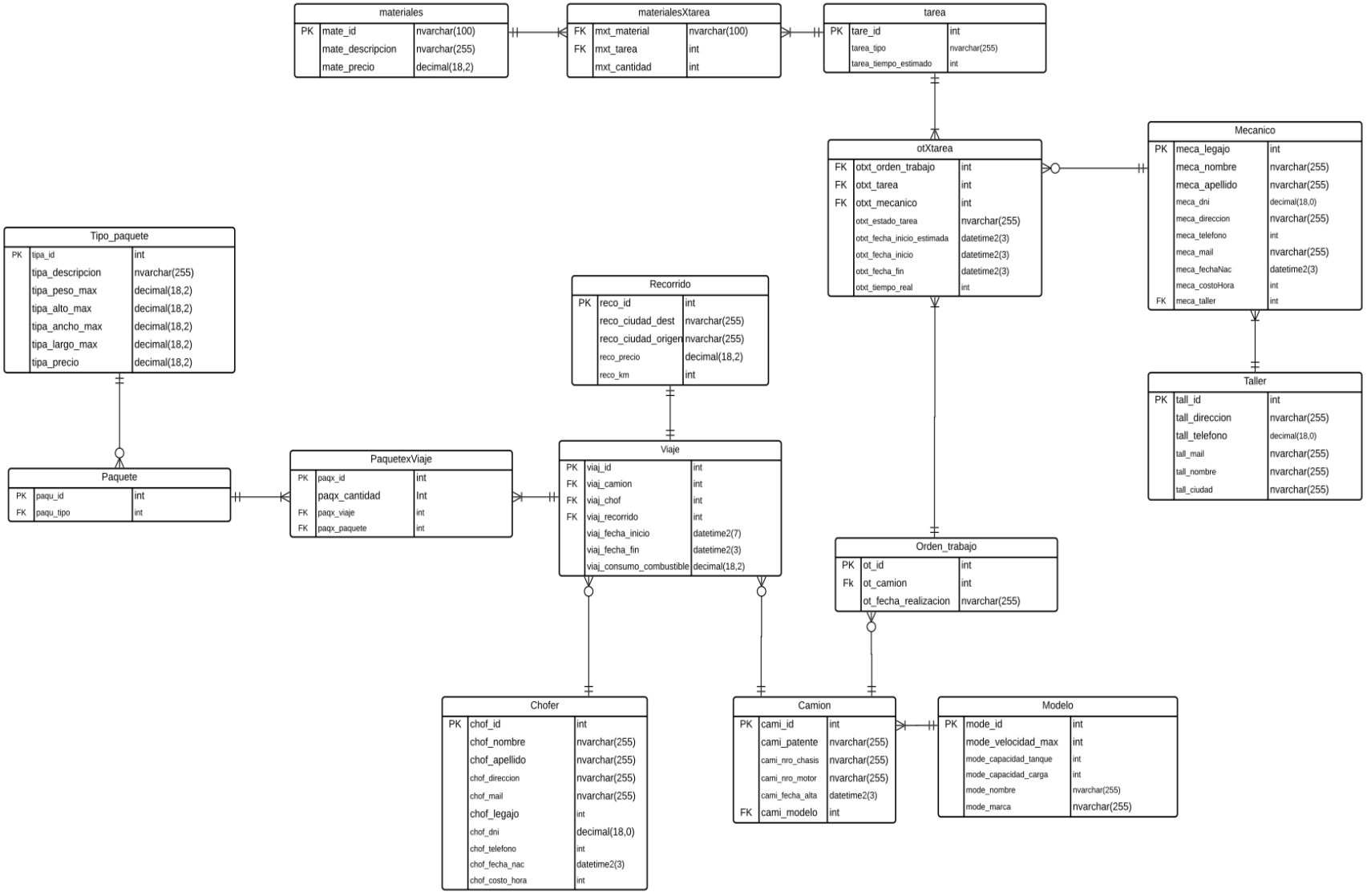
En el presente documento se desarrollará la estrategia utilizada y las decisiones tomadas para desarrollar un nuevo sistema de gestión de viajes y mantenimiento de camiones de una empresa de logística que se encarga de transportar paquetes entre distintas ciudades del país. Siguiendo el Diagrama de Entidad de Relación (DER), crearemos un modelo de datos para organizar y normalizar los datos de la tabla maestra, para una correcta migración de estos e implementación de los nuevos requerimientos a incorporar.

Haremos también una explicación en forma detallada sobre las estrategias y decisiones que tomamos y utilizamos para desarrollar el TP.

UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

GRUPO 6: Brogrammers3de5

Diagrama Entidad - Relación (DER), modelo transaccional



Documentación DER Gestión de Flota

Tras leer el enunciado, decidimos comenzar modelando “Viaje” ya que es una de las entidades principales del modelo, relacionándose con muchas otras del mismo. Tales como, “Camión”, “Chofer”, “Recorrido” y “PaquetesxViaje”, que serán detalladas más adelante.

La clase “Viaje” consta de :

viaj_id	(PK)
viaj_camion	(FK => cami_id)
viaj_chof	(FK => chof_id)
viaj_recorrido	(FK => reco_id)
viaj_fecha_inicio	
viaj_fecha_fin	
viaj_consumo_combustible	

Siguiendo con la lectura, decidimos modelar las clases directamente relacionadas con viaje, las ya nombradas, “Camión”, ”Chofer”, ”Recorrido”.

Arrancaremos por chofer, que se relaciona con viaje para saber cual es el chofer designado para el mismo. Esta clase tiene los siguientes atributos:

chof_legajo	(PK)
chof_nombre	
chof_apellido	
chof_dni	
chof_direccion	
chof_telefono	
chof_mail	
chof_fecha_nac	
chof_costo_hora	

Siguiendo con el camión, este se relaciona con viaje, debido a que cada chofer en cada viaje puede manejar un camión diferente. El camión y el chofer se relacionan entre sí a través del viaje.

cami_id	(PK)
cami_modelo	(FK => mode_id)
cami_patente	
cami_nro_chasis	
cami_nro_motor	
cami_nro_chasis	

La clase “Recorrido” consta de :

reco_id	(PK)
reco_ciudad_origen	
reco_ciudad_dest	
reco_precio	
reco_km	

A su vez, creamos la tabla “Modelo”, la cual nos permite normalizar el modelo y consta de :

mode_id	(PK)
mode_velocidad_max	
mode_capacidad_tanque	
mode_capacidad_carga	
mode_nombre	
mode_marca	

El modelo tiene como PK un ID específico (mode_id), que permite identificar al Modelo de forma unívoca, respectivamente.

Luego continuamos con la siguiente relación del viaje que es la tabla PaquetesxViaje, la cual es una tabla intermedia para separar a “Paquete” de viaje, ya que un paquete podría estar en muchos viajes y un viaje podría tener muchos paquetes. Esta tabla surge de la ruptura de la relación de muchos a muchos. Esta tabla intermedia tiene los siguientes atributos:

paqx_paquete	(PK, FK => paqu_id)
paqx_viaje	(PK,FK => viaje_id)
paqx_cantidad	

La tabla “Paquete” consta de :

paqu_id	(PK)
paqu_tipo	(FK => tipa_id)

Creamos la tabla “Tipo_Paquete”, la cual nos permite normalizar el tipo del paquete, ya que hay solo 3 tipos de paquetes cuyas medidas y especificaciones están estandarizadas. Esta tabla consta de :

tipa_id	(PK)
tipa_descripcion	
tipa_peso_max	
tipa_alto_max	
tipa_ancho_max	
tipa_largo_max	
tipa_precio	

Continuando con el modelado, realizamos la tabla de “Orden_trabajo”, la cual también posee su propio ID como PK y, al relacionarse con el camión, posee el ID del mismo como PK

ot_id	(PK)
ot_camion	(FK => cami_id)
ot_fecha_realizacion	
ot_estado	

Luego para poder incluir la información relacionada a una tarea específica que será realizada en un momento determinado, incluimos la tabla Orden de trabajo X Tarea que tiene como FK, la orden de trabajo (otxt_orden_trabajo), la tarea a realizar(otxt_tarea) y el mecánico encargado de llevar adelante la misma(otxt_mecanico).

otxt_orden_trabajo	(FK => ot_id)
otxt_tarea	(FK => tare_id)
otxt_mecanico	(FK => meca_ilegajo)
otxt_fecha_inicio_estimada	
otxt_fecha_inicio	
otxt_fecha_fin	
otxt_tiempo_real	

A su vez, creamos la tabla “Taller”, la cual nos permite normalizar el modelo y consta de :

tall_id	(PK)
tall_direccion	
tall_telefono	
tall_mail	
tall_nombre	
tall_ciudad	

Luego, modelamos la tabla mecánico para poder incluir la información de los mecánicos. En esta tabla incluimos la FK del taller mecánico en el que trabaja cada uno de los mecánicos (meca_taller).

meca_legajo	(PK)
meca_nombre	
meca_apellillido	
meca_dni	
meca_direccion	
meca_telefono	
meca_mail	
meca_fechaNac	
meca_costoHora	
meca_taller	(FK => tall_id)

Para continuar, creamos la tabla “Materiales”, la cual nos permite normalizar el modelo y consta de :

mate_id	(PK)
mate_descripcion	
mate_precio	

Luego, creamos la tabla “MaterialesXTarea” que nos permite relacionar los materiales con las tareas en las que se los requiere y la cantidad requerida para realizar cada una de las tareas. En la misma contamos con la FK que se relaciona con el material (mxt_material) y la FK que se relaciona con la tarea (mxt_tarea).

mxt_material	(FK => mate_id)
mxt_tarea	(FK => tare_id)
mxt_cantidad	

Por último, creamos la tabla “Tarea”, la cual nos permite normalizar el modelo y consta de :

tare_id	(PK)
tare_desc	
tare_tipo	
tare_tiempo_estimido	

Migración

En el presente documento se detallarán consideraciones pertinentes a algunas de las tablas de la migración puesto que los casos restantes son triviales.

Para comenzar con el proceso de migración decidimos crear todas las tablas con sus correspondientes atributos y tipo de datos asociados. A modo de ejemplo, mostraremos la creación de la tabla “VIAJE”:

```
IF OBJECT_ID ('brog.Viaje', 'U') IS NOT NULL
   DROP TABLE brog.Viaje;
GO
CREATE TABLE [brog].[Viaje] (
   [viaj_id] int identity(1,1),
   [viaj_fecha_inicio] datetime2(7) NOT NULL,
   [viaj_fecha_fin] datetime2(3) NOT NULL,
   [viaj_consumo_combustible] decimal(18,2) NOT NULL,
   [viaj_camion] int NOT NULL,
   [viaj_chof] int NOT NULL,
   [viaj_recorrido] int NOT NULL,
   CONSTRAINT PK_Viaje PRIMARY KEY ([viaj_id])
)
```

Las FK fueron creadas posteriormente para poder evitar problemas a la hora de la migración. Incluimos imagen de la creación de las llaves foráneas de la tabla “VIAJE”.

```
-- FK VIAJE
ALTER TABLE [brog].[Viaje]
ADD
  CONSTRAINT FK_Viaje_camion FOREIGN KEY (viaj_camion) REFERENCES brog.Camion(cami_id),
  CONSTRAINT FK_Viaje_chof FOREIGN KEY (viaj_chof) REFERENCES brog.Chofer(chof_id),
  CONSTRAINT FK_Viaje_recorrido FOREIGN KEY (viaj_recorrido) REFERENCES brog.Recorrido(reco_id)
GO
```

Algunas de las decisiones que tomamos fueron, la PK de la tabla viaje es un autoincremental, al igual que muchas otras tablas, ya que simplifica la creación de estos ids, además de agilizar la búsqueda. Por lo explicado anteriormente, el id no se inserta en la tabla. Los datos de los viajes los sacamos directamente de la tabla maestra con las columnas VIAJE_*.
Luego, para conectar las foreign keys joineamos con las tablas correspondientes ya cargadas anteriormente, excepto el legajo del chofer, que se puede sacar directamente de la tabla maestra.

La tabla de “materialXtarea” fue creada con el propósito de indicar que materiales y sus cantidades deben ser utilizados para la realización de cada tarea. La tabla cuenta con una FK que referencia a la tarea y otra que referencia al material en cuestión. Cada fila de la tabla representa una relación y a la misma se le incorpora el dato de la cantidad del material necesario para la tarea. Para la realización de este select, encaramos diferentes estrategias, teniendo en cada una de ellas otro resultado incorrecto. Luego, entendimos que la solución debía incorporar un subSelect y un count en el mismo y esto nos permitió dar con la solución del problema.

```
create procedure brog.migracionMaterialxTarea
as
begin
    insert into brog.MaterialesXtarea
    SELECT MATERIAL_COD, TAREA_CODIGO,
    COUNT(MATERIAL_COD) / (select count(distinct convert(varchar,TAREA_FECHA_FIN)
                          +convert(varchar, TAREA_FECHA_INICIO)
                          +convert(varchar, TAREA_FECHA_INICIO_PLANIFICADO)
                          +str(TAREA_TIEMPO_ESTIMADO)+ CAMION_PATENTE)
    FROM gd_esquema.Maestra
    WHERE TAREA_CODIGO= G1.TAREA_CODIGO)
    FROM gd_esquema.Maestra G1 where TAREA_DESCRIPCION <> 'null'
    group by  TAREA_CODIGO,MATERIAL_COD,MATERIAL_DESCRIPCION
    order by TAREA_CODIGO
end
GO
```

Otra de las tablas que creemos que vale la pena mencionar es “camión” que cuenta con los atributos que sirven para describir los atributos de los camiones. Los mismos cuentan con un id autoincremental, una patente, un número de chasis, número de motor, fecha de alta, y el modelo. El modelo, es una FK que hace referencia a mode_id de la tabla “modelo”.

```
-- MIGRACION CAMION

IF OBJECT_ID ('brog.migracionCamion', 'P') IS NOT NULL
   DROP PROCEDURE brog.migracionCamion;
GO

create procedure brog.migracionCamion
as
begin
    insert into brog.Camion
    select distinct CAMION_PATENTE, CAMION_NRO_CHASIS, CAMION_NRO_MOTOR,CAMION_FECHA_ALTA, mode_id
    from gd_esquema.Maestra
    join brog.Modelo on (mode_nombre = MODELO_CAMION and MODELO_CAPACIDAD_CARGA = mode_capacidad_carga and
    MODELO_CAPACIDAD_TANQUE = mode_capacidad_tanque and MODELO_VELOCIDAD_MAX = mode_velocidad_max)
    where CAMION_PATENTE <> 'null'
end
GO
```

La tabla modelo, incluye la información con la que contamos acerca de los modelos de camino. La relación que mantienen Camión y modelo es una relación de un modelo a muchos camiones, ya que podrían haber muchos camiones con el mismo modelo. La misma opcional del lado del camión ya que no deben existir obligatoriamente camiones de todos los modelos. Pero si es obligatoria del lado del modelo, ya que no puede existir ningún camión que no tenga modelo.

```
-- MIGRACION MODELO

IF OBJECT_ID ('brog.migracionModelo', 'P') IS NOT NULL
   DROP PROCEDURE brog.migracionModelo;
GO

create procedure brog.migracionModelo
as
begin
    insert into brog.Modelo
    select distinct MODELO_CAMION,
                    MARCA_CAMION_MARCA,
                    MODELO_VELOCIDAD_MAX,
                    MODELO_CAPACIDAD_TANQUE,
                    MODELO_CAPACIDAD_CARGA
    from gd_esquema.Maestra where CHOFER_NOMBRE <> 'null'

end
GO
```