Universidad Tecnológica Nacional

Facultad Regional Buenos Aires

Gestión de Datos

Trabajo Práctico

2° Cuatrimestre 2021

FRBA – Gestión de Flota

Integrantes: Ezequiel Castiglione, Tomas Rodriguez y Martin Lingeri

ÍNDICE

SCRIPT INICIAL

Entidades

- Viaje: cuenta con una PK, viaje_id de tipo int, para identificarlo unívocamente. Tiene tres atributos FK: viaje_camion de tipo int, teniendo una relación de muchos a uno con la entidad camión, ya que un muchos viajes pueden haber sido realizados por un mismo camión; viaje_chofer de tipo int, teniendo una relación de muchos a uno con la entidad chofer, ya que un muchos viajes pueden haber sido realizados por un mismo chofer; y viaje_recorrido de tipo int, teniendo una relación de muchos a uno con la entidad recorrido, ya que un muchos viajes pueden haber realizado el mismo recorrido. Además, tiene una relación de uno a muchos con paquete.
- Recorrido: esta entidad cuenta con una PK reco_id de tipo entero que funciona para identificarlo de forma unívoca. Tiene el atributo reco_ciudad_origen de tipo int que es FK teniendo una relación de muchos a uno con la entidad ciudad; y tiene otro atributo FK, reco_ciudad_destino, de tipo int, teniendo una relación de muchos a uno con la entidad ciudad.
- Paquete: esta entidad cuenta un una PK pack_id de tipo entero que funciona como un número identificatorio único. Tiene dos FK 's: pack_viaje de tipo int, estableciendo una relación de muchos a uno con viaje; y pack_tipo de tipo int, estableciendo una relación de muchos a uno con tipo_paquete.
- Tipo paquete: tiene el atributo PK tipo_pack_id de tipo int para identificar cada tipo de paquete de forma única. Además tiene una relación de uno a muchos con paquete.
- Ciudad: decidimos realizar una normalización en cuanto a esta entidad ya que tanto el recorrido como el taller contaban con un atributo ciudad, la cual puede ser la misma en ambas. Tiene dos relaciones, ambas de uno a muchos, con recorrido ya que este tiene una ciudad destino y una de origen; y teniendo una relación de uno a muchos con taller). Esta entidad ciudad cuenta con un id de tipo entero que es PK, y su nombre de tipo varchar.
- Taller: esta entidad cuenta con un atributo taller_id de tipo int que es PK para identificar de forma única a cada taller. Tiene un atributo, taller_ciudad de tipo int, que

es FK, estableciendo una relación de muchos a uno con ciudad, ya que muchos talleres pueden estar en una misma ciudad.

- Mecánico: esta entidad cuenta con el atributo meca_nro_legajo de tipo int que es PK para identificar de forma unívoca al mecánico y una FK a taller_id de la tabla Taller. Tiene una relación de uno a muchos con tarea_por_odt.
- Tarea: esta entidad tiene el atributo tarea_codigo de tipo int que es PK cuya función es identificar las tareas de forma única. Tiene el atributo tarea_tipo de tipo int que es FK, teniendo una relación de muchos a uno con tipo_tarea, ya que muchas tareas pueden ser del mismo tipo. Además, tiene una relación de uno a muchos con tarea_por_odt.
- Tarea por ODT: es una entidad que cuenta con tres FK: tarea_id de tipo int, estableciendo una relación de muchos a uno con tareas; odt_id de tipo int, que establece una relación de muchos a uno con orde_de_trabajo; y txo_mecanico que es de tipo int, y relaciona de muchos a uno esta entidad con la entidad mecánico.
- Orden de trabajo: tiene el atributo odt_id de tipo int que es PK cuya función es identificar cada orden de trabajo unívocamente. Tiene el atributo odt_camion de tipo int, estableciendo una relación de muchos a uno con camión, porque muchas órdenes de trabajo pueden tener al mismo camión. Además, tiene una relación de uno a muchos con tarea_por_odt.
- Tipo tarea: cuenta con el atributo tipo_id de tipo int que es PK para identificar de forma única cada tipo de tarea. Tiene una relación de uno a muchos con tareas.
- Herramienta por tarea: es una tabla intermedia que surge del hecho que muchas tareas pueden tener muchas herramientas, y viceversa, además de que se necesita contar con la cantidad de herramientas por cada tarea. Por esto, tiene dos FK: tarea_codigo de tipo int, estableciendo una relación de muchos a uno con tarea; y herra_id de tipo int que establece una relación de muchos a uno con herramienta.
- Herramientas: cuenta con el atributo herra_id de tipo int que es PK, que identifica de forma única cada herramienta. Tiene una relación de uno a muchos con herramienta_por_tarea
- Camión: cuenta con el atributo PK patente, de tipo char, el cual identifica unívocamente cada camión. Tiene un atributo FK cami_modelo de tipo int, que establece una relación de muchos a uno con modelo. Tiene el atributo FK cami_marca de tipo int, que establece una relación de muchos a uno con marca. Además, tiene una relación de uno a muchos con orden de trabajo y una relación de uno a muchos con viaje.
- Modelo: tiene el atributo PK modelo_id de tipo int para identificar cada modelo de forma única. Tiene una relación de uno a muchos con la entidad camión.
- Marca: tiene el atributo marca_id de tipo int que es PK, para identificar cada marca de forma única. Tiene una relación uno a muchos con la entidad camión.

Store Procedures

- Creacion_de_Tablas: con la ejecución de este procedure se crean todas las tablas (sin PK's ni FK's)

- PK_Y_FK: con la ejecución de este procedure se agregan las claves primarias y foráneas de las tablas anteriormente mencionadas.
- Procedures para la realización de las migraciones:
    - Insercion_Tabla_Ciudad
    - Insercion_Tabla_Herramientas
    - Insercion_Tabla_Recorrido
    - Insercion_Tabla_Taller
    - Insercion_Tabla_Chofer
    - Insercion_Tabla_Marca
    - Insercion_Tabla_Modelo
    - Insercion_Tabla_Camion
    - Insercion_Tabla_Mecanico
    - Insercion_Tabla_Tipo_Tarea
    - Insercion_Tabla_Tareas
    - Insercion_Tabla_Viaje
    - Insercion_Tabla_Paquete
    - Insercion_Tabla_Tipo_Paquete
    - Insercion_Tabla_Orden_De_Trabajo
    - Insercion_Tabla_Herramienta_Por_Tarea
    - Insercion_Tabla_Tarea_Por_ODT
- Migración: con la ejecución de este procedure se ejecutan todos los procedures de las migraciones.
- Reseteo_Tablas: con la ejecución de este procedure se eliminan todas las tablas mencionadas al principio.
- Reseteo_Procedures: con la ejecución de este procedure se eliminan todos los procedures anteriores (los de creación de tablas, asignación de pk's y fk's, inserciones de tablas para las migraciones, migración, reseteo de tablas, reseteo, play, y a sí mismo).
- Reseteo: ejecuta los procedures reseteo_tablas y reseteo_procedures, con lo cual se eliminan todas las tablas antes mencionadas y los procedures mencionados.
- Play: sí existe el esquema SQLI ejecutar el procedure de reseteo, y elimina el esquema SQLI, y luego define que va a usar la base de datos GD2C2021, crea el esquema SQLI, ejecuta el procedure de creación de tablas, ejecuta el procedure de agregar PK's y FK's, y ejecuta el procedure para la realización de migraciones.

SCRIPT BI

Entidades

En esta parte del TP, se implementaron dos tablas de hechos para trabajar las vistas. Consideramos esa cantidad debido a que, con esas dos tablas, podemos dividir las vistas pedidas y para también no tener mucha información repetida. Además, se eligieron esos campos como PK (que son en realidad FKs que referencian a las PKs de las dimensiones)

porque es lo único que se utiliza para poder identificar cada viaje/reparación o mantenimiento que exista en el sistema. Estas tablas de hechos son:

- Hechos_Viajes: sirve para trabajar todo lo que tenga que ver con los distintos viajes realizados por la empresa (vistas 6, 7, 8). En esta tabla, tenemos una PK compuesta por tiempo, camión, recorrido, combo de paquete (un id que identifica las distintas combinaciones de paquetes usados por la empresa) y el id del rango etario que le corresponde a cada chofer. A su vez, agregamos otros campos como el precio del recorrido, los litros de combustible consumidos en ese recorrido, la duración del viaje, el costo del mismo, la descripción del rango etario y el costo por hora del chofer. Agregamos estos últimos 4 atributos porque consideramos que debemos tener precalculada esta información importante a la hora de realizar las vistas.

- Hechos_Reparaciones: contiene información acerca de las distintas reparaciones y mantenimientos que se le hicieron a los distintos camiones. La PK es compuesta, conformada por el tiempo, camión, modelo, marca, orden de trabajo, taller, tarea y herramientas. Idem con la situación del rango etario. (Nota: para referirnos a los materiales, nosotros usamos el término de Herramienta. Se aclara ya que en los scripts aparece con ese término). Nos encontramos con un problema: existía un caso donde los campos de la PK se igualan entre sí, prohibiendo que siga la inserción. Como solución, planteamos poner un campo (idRepa) que es un Identity, solucionando el caso. Y para que quede igual en ambas tablas, metimos un idViaje en la otra tabla de hechos. Contiene un campos con valores precalculados como desvio_tarea, que se va a encargar de almacenar la diferencia entre la fecha de inicio planificada y la fecha de ejecución de la tarea, costo materiales, costo_mdo; todos estos valores precalculados son útiles para poder acceder más fácilmente al momento de realizar las vistas correspondientes.

Ahora, para las distintas dimensiones, las únicas que se repiten en ambas tablas son la de Camión y Tiempo. No existen FKs en las dimensiones. Procedemos a explicar cada una:

- Tiempo: contiene un idTiempo que es la PK. Sus atributos son año (denotado año en el script) y cuatrimestre. Cada combinación de año/cuatrimestre se sacó de las distintas fechas de inicio de los viajes en la tabla de Viajes de la entrega pasada y de Tarea por Orden de Trabajo, también de la entrega pasada. Se implementó con un UNION así se podía obtener la mayor cantidad de combinaciones distintas posibles.
- Camión: la PK es la patente. Los otros atributos de esta dimensión son los restantes de la tabla Camión de la entrega pasada menos las FKs. Esta lógica se repite de manera similar en el resto de las dimensiones.
- Marca: contiene un idMarca que es PK y el nombre de la misma.
- Paquete: contiene un idPaquete que es PK y después el resto de los atributos son los de la tabla Paquete combinados con los de Tipo_Paquete de la entrega pasada. Tiene un atributo extra, precio_final, que seria el precio_base * cantidad. Creamos ese atributo ya que lo usamos en la view de Ganancia_X_Camion.
- Modelo: contiene la PK que es idModelo y los mismos atributos que su contraparte de la entrega pasada.
- Taller: idem que Modelo. La única diferencia acá es que, debido a que la dimensión Ciudad no se implementa nunca, el campo de ciudad es un NVARCHAR(255) que joineamos con la tabla Ciudad de la entrega pasada.

- Tarea: su PK es idTarea y contiene el código, la descripción, el tipo de la misma y su tiempo estimado de finalización.
- Recorrido: mismos campos que los creados para la tabla Recorrido de la entrega pasada. Se hizo lo mismo de la ciudad de la dimensión Taller para la ciudad destino y origen.
- Herramienta: Contiene idHerramienta que es la PK, el detalle de la misma, precio y cantidad. Ese valor de cantidad depende de la tabla Herramienta_Por_ODT que se utilizó para la entrega pasada. Si en dos tareas distintas se usa la misma herramienta pero la cantidad usada es distinta, se van a generar dos registros con id distintos.
- ODT: tiene una idODT que es la PK, un campo para el estado de esa orden, una fecha de generación y un campo para la duración de esa ODT, que se implementa en la primera view.
- Rango_Etario: tiene una idRango que es la PK de la Dimensión que identifica de manera unívoca cada uno de los rangos de las edades (18-30, 31-50, +50).

Decidimos no implementar dimensiones para el Chofer y para el Mecánico ya que no considerábamos como piezas importantes en el plan de negocios de la empresa según las vistas que nos proponen.

Store Procedures

- Creacion_Tablas_BI: crea todas las dimensiones y tablas de hechos mencionadas antes, durante el proceso también asigna las PKs donde corresponda
- BI_Migracion: ejecuta los procedures para el llenado de datos en cada dimensión y tabla de hechos. Los procedures que ejecuta son:
    1) Insercion_Dimension_Camion
    2) Insercion_Dimension_Marca
    3) Insercion_Dimension_Modelo
    4) Insercion_Dimension_Taller
    5) Insercion_Dimension_Tarea
    6) Insercion_Dimension_Recorrido
    7) Insercion_Dimension_Tiempo
    8) Insercion_Dimension_Herramienta
    9) Insercion_Dimension_Paquete
    10) Insercion_Dimension_ODT
    11) Insercion_Hechos_Reparaciones
    12) Insercion_Hechos_Viajes

- BI_Reseteo_Tablas: si las tablas de las dimensiones y hechos ya existen, las dropea.
- BI_Reseteo_Procedures: idem que BI_Reseteo_Tablas pero con todos los procedures
- BI_Reseteo_Vistas: idem que BI_Reseteo_Procedures, pero con las vistas
- BI_Reseteo: ejecuta los tres procedures mencionados anteriormente
- BI_Play: es el procedure principal. Si existe el schema que se usa, ejecuta BI_Reseteo y dropea el schema. Si no existe, crea el schema y ejecuta Creacion_Tablas_BI y BI_Migracion.

FE DE ERRATAS

- Se tuvo que modificar el DER corregido de la primera entrega. Tuvimos que crear una relación entre Mecanico y Taller donde meca_taller es una FK a la tabla Taller. Como ven en la screenshot, para una tabla de hechos se relaciona el id del taller con el atributo meca_taller del mecánico. Se aclara ya que en el DER original no estaba esa relación y no nos había corregido ese tema. Los cambios sobre los atributos en el script inicial ya se hicieron.

```
CREATE PROCEDURE Insercion_Hechos_Reparaciones AS
BEGIN
    INSERT INTO [GD2C2021].[SQLI].BI_Hechos_Reparaciones(tiempo, camion, marca, modelo, odt, taller, tarea, legajo_mecanico, herramienta)
    SELECT DISTINCT [GD2C2021].[SQLI].buscarIdDelTiempoSegun(txo.tarea_fecha_inicio), odt_camion, mar.marca_id, mode.modelo_id,
    odt.odt_id, 1 /*ACA IRIA EL TALLER*/, tar.tarea_codigo, mec.meca_nro_legajo, her.herra_id
    FROM [GD2C2021].[SQLI].Orden_De_Trabajo odt
    JOIN [GD2C2021].[SQLI].Tarea_Por_ODT txo on txo.odt_id = odt.odt_id
    JOIN [GD2C2021].[SQLI].Camion cam on cam.cami_id = odt.odt_camion
    JOIN [GD2C2021].[SQLI].Marca mar on mar.marca_id = cam.cami_marca
    JOIN [GD2C2021].[SQLI].Modelo mode on mode.modelo_id = cam.cami_modelo
    JOIN [GD2C2021].[SQLI].Mecanico mec on mec.meca_nro_legajo = txo.tarea_mecanico
--  JOIN [GD2C2021].[SQLI].Taller tal on tal.taller_id = mec.meca_taller
    JOIN [GD2C2021].[SQLI].Tareas tar on tar.tarea_codigo = txo.tarea_id
    JOIN [GD2C2021].[SQLI].Herramienta_Por_Tarea hpt on hpt.tarea_codigo = tar.tarea_codigo
    JOIN [GD2C2021].[SQLI].Herramientas her on her.herra_id = hpt.herra_id
END
GO
```

- En Hechos_Reparaciones, metimos la fecha inicial de la tarea de una OdT porque la fecha de generación de la OdT la tenemos en formato NVARCHAR(255) debido a que figuraba así en la tabla maestra.

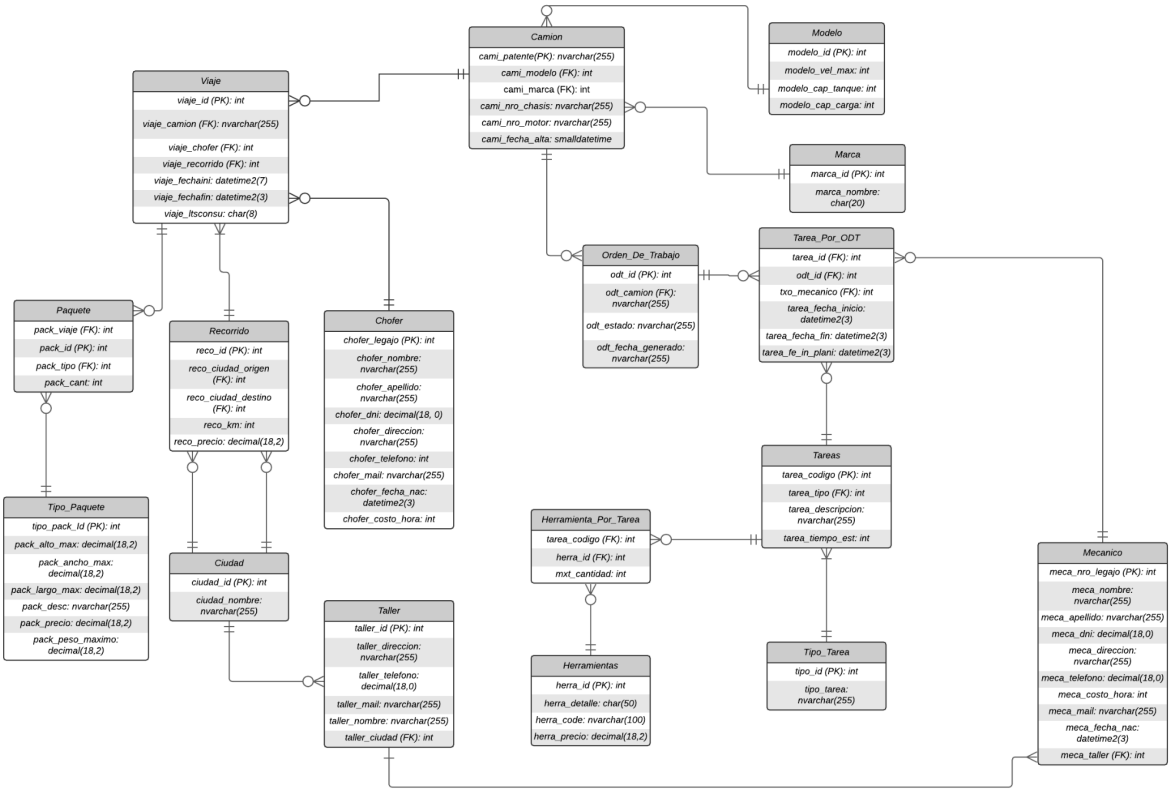
- En el procedure Insercion_Tabla_Herramienta_Por_Tarea no contamos con el atributo MXT_CANTIDAD en la tabla maestra por lo que no podemos realizar la inserción de manera correcta (se colocó un 0, como para que pueda seguir con el flujo del programa).

```
CREATE PROCEDURE Insercion_Tabla_Herramienta_Por_Tarea AS
    INSERT INTO [GD2C2021].[SQLI].Herramienta_Por_Tarea (tarea_codigo, herra_id)--, mxt_cantidad) FE DE ERRATAS mxt_cantidad esta comentado porque no existe la columna en la tabla maestra.
    SELECT      tar.tarea_codigo, her.herra_id --', MXT_CANTIDAD'
    FROM        [GD2C2021].[gd_esquema].Maestra as MASTERTABLE
    join        [GD2C2021].[SQLI].Herramientas her on MASTERTABLE.MATERIAL_COD = her.herra_code and MASTERTABLE.MATERIAL_DESCRIPCION = her.herra_detalle
        and MASTERTABLE.MATERIAL_PRECIO = her.herra_precio
    join [GD2C2021].[SQLI].Tareas tar on MASTERTABLE.TAREA_CODIGO = tar.tarea_codigo
    where (MASTERTABLE.MATERIAL_COD is not null)
    group by tar.tarea_codigo, her.herra_id-- ', MXT_CANTIDAD'
GO
```

- Se modificó el orden de ejecutar y crear las tablas y/o procedures.

DER

DER del modelo relacional



DER del modelo BI