UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

GESTIÓN DE DATOS

Año 2022 - 1° Cuatrimestre
Curso: K3151
Grupo: GDD_EXPRESS

Integrantes:

- Selvaggi, Tomás - 1716086
- Frioli, Florencia - 1524203
- Maiolo, Joaquín - 1728489

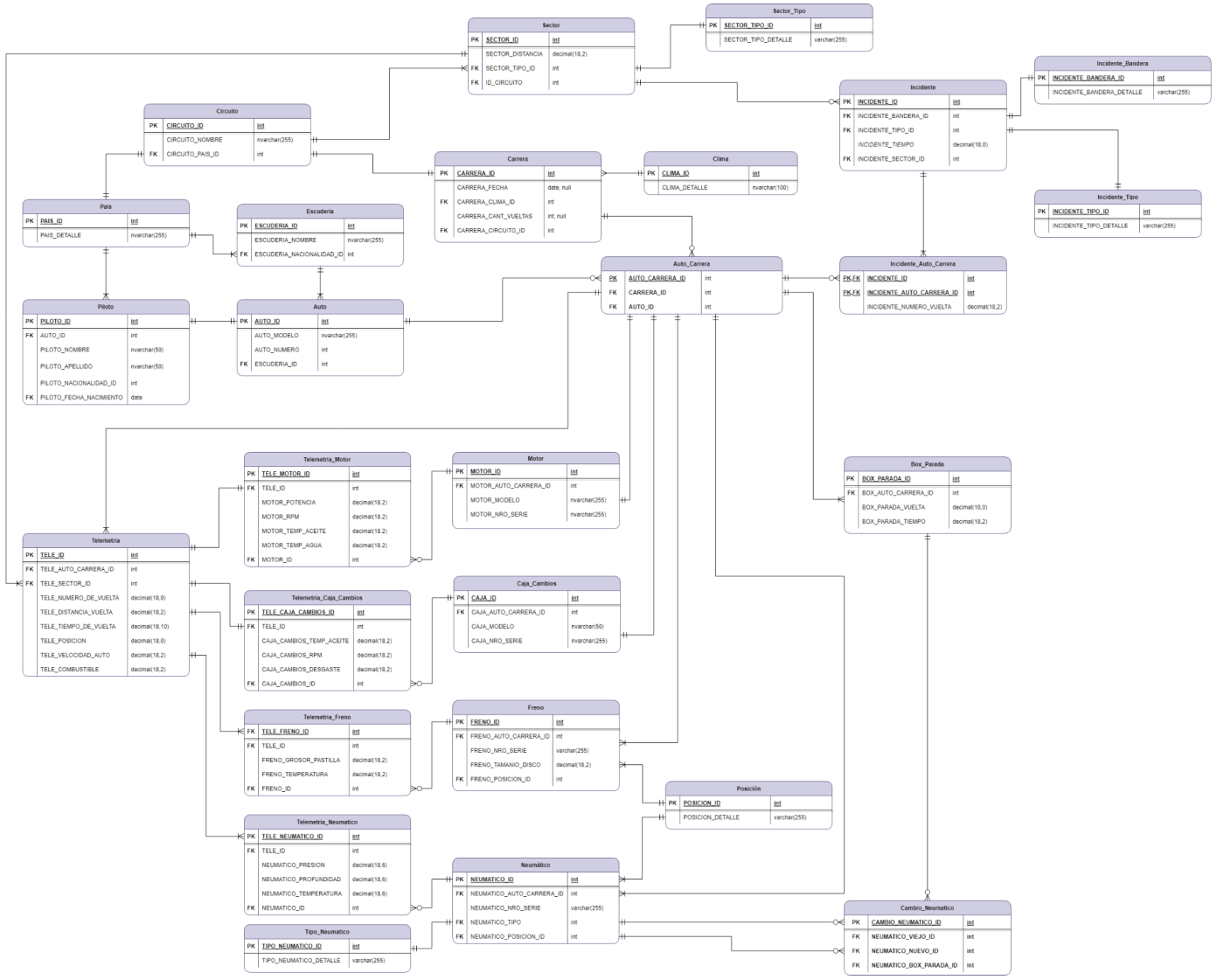
Fechas de entrega:

- Entrega N°1 - DER - 15/05/2022
- Entrega N°2 - Modelo de datos y Migración - 05/06/2022
- Entrega N°3 - Modelo Business Intelligence - 26/06/2022

Índice

1. Diagrama de Entidad - Relación

El siguiente diagrama de entidad - relación representa la estructura del modelo de datos a través del cual se organizan y normalizan los datos de la única tabla provista por la cátedra.
Adjuntamos una copia de este en su resolución original en la entrega cuyo nombre es **“DER.png”**.

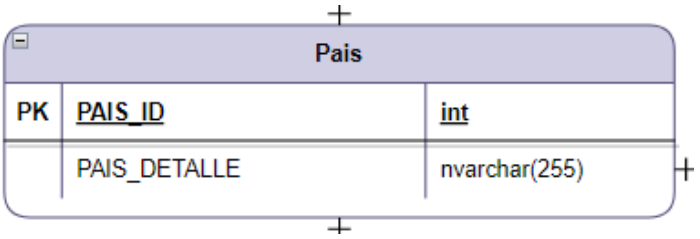


A continuación, procedemos a justificar la creación de las distintas tablas utilizadas en el modelo relacional para la migración de datos de la tabla maestra.

Tabla pais

Decidimos normalizar el atributo **país** que formaba parte tanto de la escudería, como del piloto. También estaba presente en el circuito.

Entonces, luego de crear esta tabla reemplazamos en las tablas mencionadas por una FK esta nueva, la cual contiene únicamente el detalle del país:



En Piloto → FK a País será PILOTO_NACIONALIDAD_ID

En Escuderia → FK a Pais será ESCUDERIA_NACIONALIDAD_ID

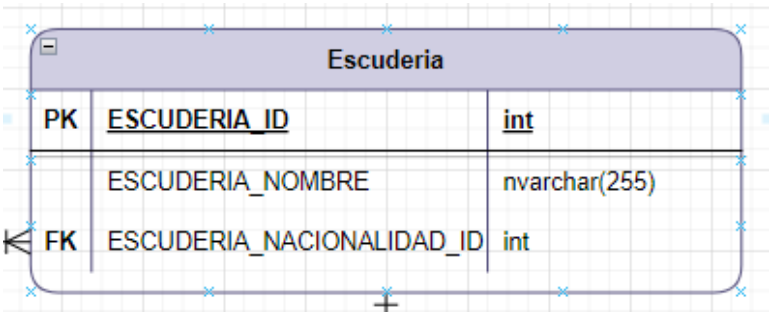
En Circuito → FK a Pais será CIRCUITO_PAIS_ID

Se decide establecer un **pais_id** como **PRIMARY KEY** que será generado por un **IDENTITY.**

```
CREATE TABLE GDD_EXPRESS.Pais
(
PAIS_ID                          int IDENTITY(1,1), --PK
PAIS_DETALLE                     nvarchar(255),
PRIMARY KEY (PAIS_ID)
)
```

Tabla escudería

Debido a que una escudería puede tener uno o dos autos, para evitar redundancias definimos la tabla ESCUDERÍA con la siguiente información, y asociamos al AUTO una FK a la escudería.

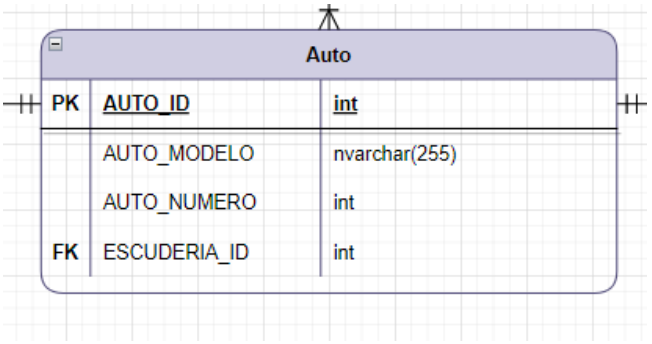


Se decide establecer un **escuderia_id** como **PRIMARY KEY** que será generado por un **IDENTITY.**

```
CREATE TABLE GDD_EXPRESS.Escuderia
(
ESCUDERIA_ID                    int IDENTITY(1,1), --PK
ESCUDERIA_NOMBRE                nvarchar(255),
ESCUDERIA_PAIS_ID               int,
PRIMARY KEY(ESCUDERIA_ID),
FOREIGN KEY (ESCUDERIA_PAIS_ID) REFERENCES GDD_EXPRESS.Pais (PAIS_ID)
)
```

Tabla Auto

Decidimos extraer los datos de un auto como el modelo y el número y definir y crear una tabla AUTO, compuesta por los siguientes campos:

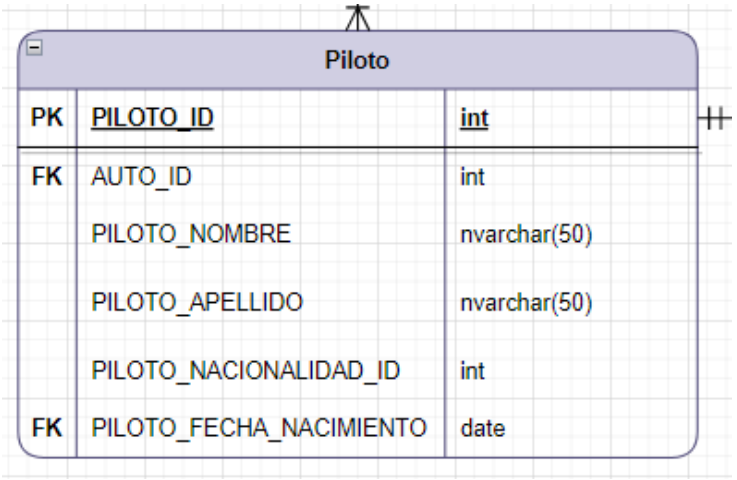


Se decide establecer un **auto_id** como **PRIMARY KEY** que será generado por un **IDENTITY.**

```
CREATE TABLE GDD_EXPRESS.Auto
(
AUTO_ID                         int IDENTITY(1,1), --PK
AUTO_MODELO                     nvarchar(255),
AUTO_ESCUDERIA_ID               int, --FK
AUTO_NUMERO                     int,
PRIMARY KEY (AUTO_ID),
FOREIGN KEY (AUTO_ESCUDERIA_ID) REFERENCES GDD_EXPRESS.Escuderia (ESCUDERIA_ID))
```

Tabla piloto

Decidimos abstraer en una tabla piloto los datos referidos al mismo, quedando de la siguiente manera:



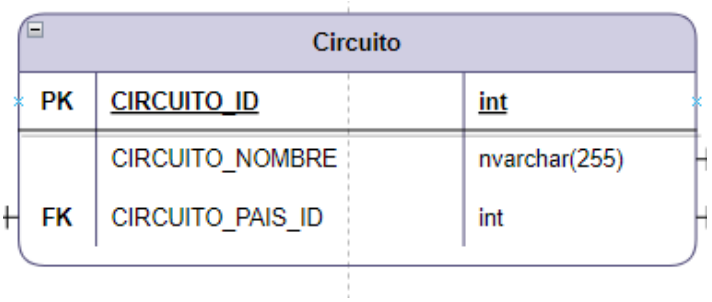
Se asocia un piloto a un auto mediante la **FK AUTO_ID**

Se decide establecer un **piloto_id** como **PRIMARY KEY** que será generado por un **IDENTITY.**

```
CREATE TABLE GDD_EXPRESS.Piloto
(
PILOTO_ID                           int IDENTITY(1,1), --PK
PILOTO_AUTO_ID                      int UNIQUE, --FK
PILOTO_NOMBRE                       nvarchar(50),
PILOTO_APELLIDO                     nvarchar(50),
PILOTO_NACIONALIDAD                 int,
PILOTO_FECHA_NACIMIENTO             date,
PRIMARY KEY (PILOTO_ID),
FOREIGN KEY(PILOTO_AUTO_ID) REFERENCES GDD_EXPRESS.Auto (AUTO_ID)
)
```

Tabla circuito

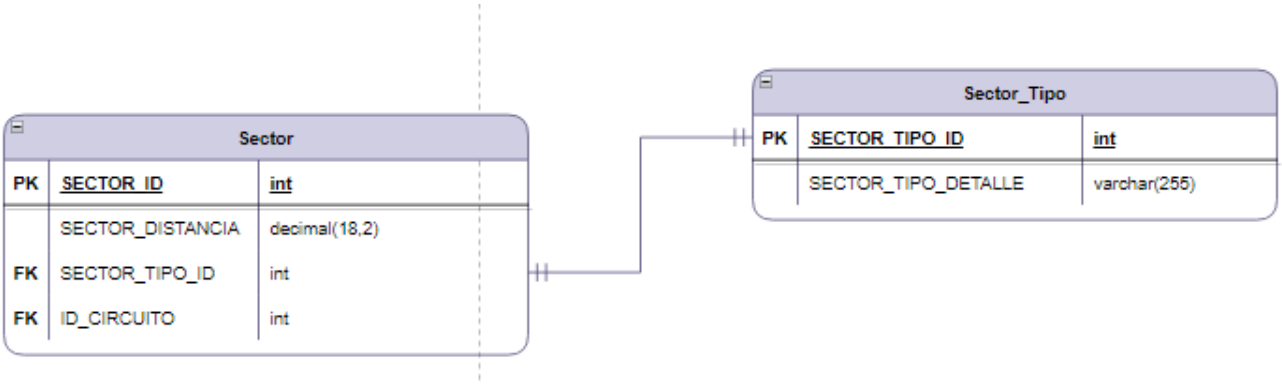
Como se mencionó anteriormente, se normaliza en esta tabla la información del clima, con un **circuito_id** como **PRIMARY KEY** que será generado por un **IDENTITY.**
Esto se referenciará en la tabla carrera mediante una FK CARRERA_CIRCUITO_ID y tendrá una FK a la tabla país mencionada anteriormente.
Asumimos que una carrera únicamente está formada por un circuito, y un circuito solo pertenece a una carrera, por lo tanto se los vincula con una relación 1 a 1.




```
CREATE TABLE GDD_EXPRESS.Circuito
(
CIRCUITO_ID                    int, --PK
CIRCUITO_NOMBRE                      nvarchar(255),
CIRCUITO_PAIS_ID               int, --FK
PRIMARY KEY (CIRCUITO_ID),
FOREIGN KEY (CIRCUITO_PAIS_ID) REFERENCES GDD_EXPRESS.Pais (PAIS_ID)
)
```

Tabla sector

Se crea la tabla sector, dado que un circuito está conformado por muchos sectores con datos propios como distancia y tipo, quedando de la siguiente manera:



Se decide establecer un **sector_id** como **PRIMARY KEY** que será generado por un **IDENTITY.**

A su vez, dado que el tipo de un sector se puede repetir muchas veces entre distintos registros, y para evitar la redundancia, se abstrae el **sector_tipo en una tabla.**

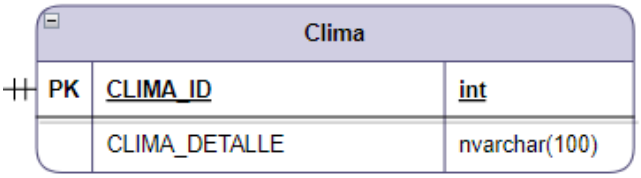
Se decide establecer un **sector_tipo_ id** como **PRIMARY KEY** que será generado por un **IDENTITY** y sector_tipo_detalle contendrá la descripción del tipo (*curva, recta, etc*). No deberán repetirse.

```
CREATE TABLE GDD_EXPRESS.Sector_Tipo
(
SECTOR_TIPO_ID                      int IDENTITY(1,1), --PK
SECTOR_TIPO_DETALLE                 nvarchar(255),
PRIMARY KEY (SECTOR_TIPO_ID))
```

```
CREATE TABLE GDD_EXPRESS.Sector
(
SECTOR_ID                    int, --PK
SECTOR_DISTANCIA             decimal(18,2),
SECTOR_TIPO_ID                     int, --FK
SECTOR_CIRCUITO_ID                 int, --FK
PRIMARY KEY (SECTOR_ID),
FOREIGN KEY (SECTOR_TIPO_ID) REFERENCES GDD_EXPRESS.Sector_Tipo (SECTOR_TIPO_ID),
FOREIGN KEY (SECTOR_CIRCUITO_ID) REFERENCES GDD_EXPRESS.Circuito (CIRCUITO_ID)
)
```

Tabla clima

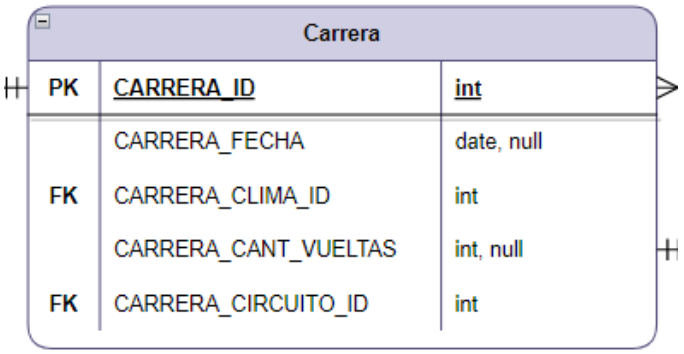
Se normaliza en esta tabla la información del clima, con un **clima_id** como **PRIMARY KEY** que será generado por un **IDENTITY.**
Esto se referenciará en la tabla carrera mediante una FK CARRERA_CLIMA_ID




```
CREATE TABLE GDD_EXPRESS.Clima
(
CLIMA_ID                     int IDENTITY(1,1), --PK
CLIMA_DETALLE                      nvarchar(100),
PRIMARY KEY (CLIMA_ID)
)
```

Tabla Carrera

Se decide generar una tabla carrera para agrupar la información correspondiente a la misma, quedando de la siguiente manera:



Se decide establecer un **carrera_id** como **PRIMARY KEY** que será generado por un **IDENTITY.**

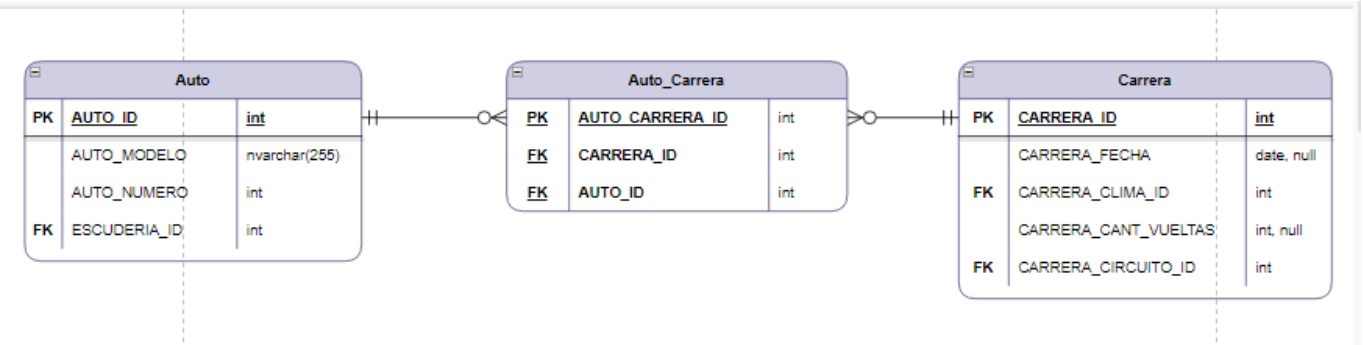
Se decide normalizar los datos del circuito y del clima en otras tablas que serán explicadas a continuación y referenciadas acá con las FK presentes.

El total de la carrera asumimos que es un dato calculable, por lo tanto no se incluirá en la tabla de Carrera.

```
CREATE TABLE GDD_EXPRESS.Carrera
(
CARRERA_ID                    int, --PK
CARRERA_FECHA                       date,
CARRERA_CLIMA_ID              int, --FK
CARRERA_CANT_VUELTAS                int,
CARRERA_CIRCUITO_ID                 int, --FK
PRIMARY KEY (CARRERA_ID),
FOREIGN KEY (CARRERA_CLIMA_ID) REFERENCES GDD_EXPRESS.Clima (CLIMA_ID),
FOREIGN KEY (CARRERA_CIRCUITO_ID) REFERENCES GDD_EXPRESS.Circuito (CIRCUITO_ID)
)
```

Tabla auto_carrera

Dado que un auto puede participar de muchas carreras, y una carrera está conformada por muchos autos, para simplificar esta relación N a N generamos una tabla intermedia **auto_carrera**.
Además decidimos utilizar **auto_carrera_id** como **PRIMARY KEY** que será generado por un **IDENTITY.**
Entonces, para evitar tener registros duplicados en cuanto a combinación de FKs de AUTO_ID y CARRERA_ID decidimos agregar una constraint de uniq a esas dos columnas.

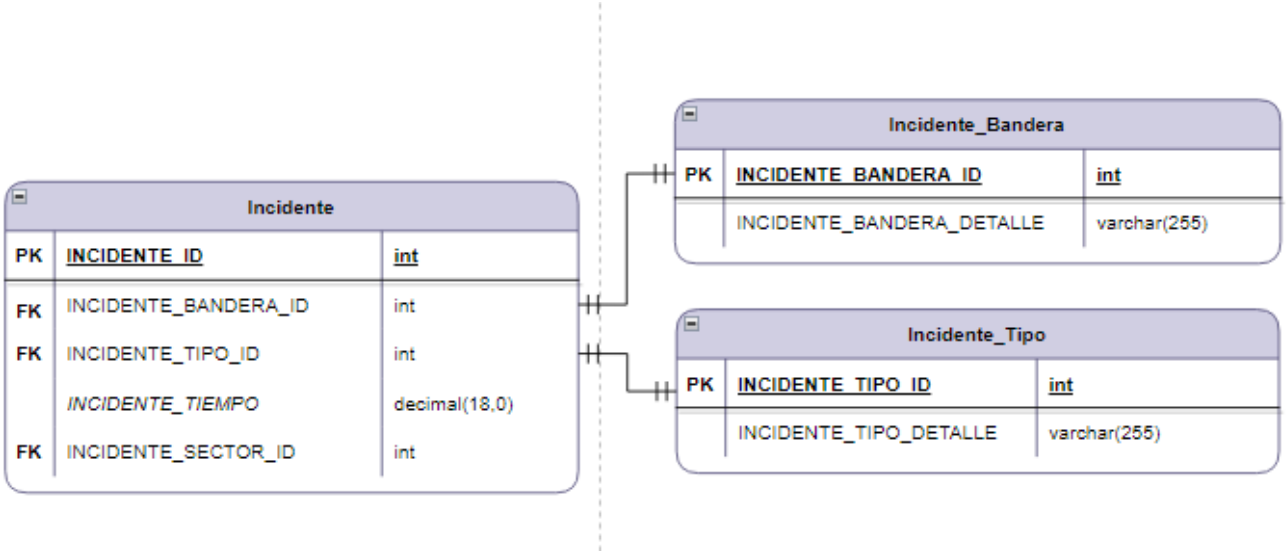


Como ventaja, todas las tablas que referencian a esta tabla Auto Carrera necesitarán de una única FK AUTO_CARRERA_ID

```
CREATE TABLE GDD_EXPRESS.Auto_Carrera
(
AUTO_CARRERA_ID                    int IDENTITY(1,1),
AUTO_ID                            int, --PK, FK
CARRERA_ID                    int, --PK, FK
PRIMARY KEY (AUTO_CARRERA_ID),
UNIQUE (AUTO_ID, CARRERA_ID),
FOREIGN KEY (AUTO_ID) REFERENCES GDD_EXPRESS.Auto (AUTO_ID),
FOREIGN KEY (CARRERA_ID) REFERENCES GDD_EXPRESS.Carrera (CARRERA_ID)
)
```

Tabla incidentes

Generamos una tabla para tener toda la información referida a un incidente, la cual queda de la siguiente manera:



El tiempo del incidente asumimos que es un dato que debería existir y guardarlo en la tabla del incidente.

Al igual que en casos anteriores, se decide establecer un **incidente_id** como **PRIMARY KEY** que será generado por un **IDENTITY.**

Tabla incidente_tipo

Similar a casos anteriores, dado que el tipo de un incidente se puede repetir muchas veces entre distintos registros, y para evitar la redundancia, se abstrae el incidente_tipo en una tabla.

Se decide establecer un **incidente_tipo_id** como **PRIMARY KEY** que será generado por un **IDENTITY** e incidente_tipo_detalle contendrá la descripción del tipo (rotura, choque, etc). No deberán repetirse.

Tabla incidente_bandera

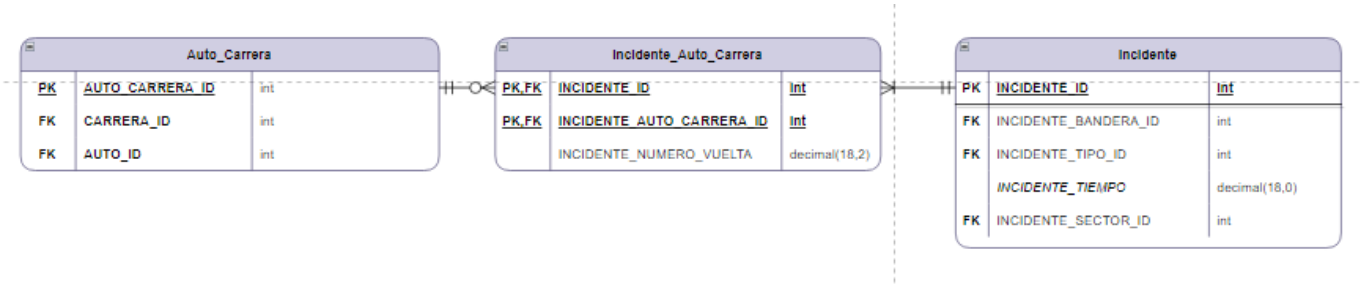
Similar a casos anteriores, dado que la bandera de un incidente se puede repetir muchas veces entre distintos registros, y para evitar la redundancia, se abstrae el incidente_bandera en una tabla.

Se decide establecer un **incidente_bandera_id** como **PRIMARY KEY** que será generado por un **IDENTITY** e incidente_bandera_detalle contendrá la descripción de la bandera (roja, amarilla, etc). No deberán repetirse.

```
CREATE TABLE GDD_EXPRESS.Incidente_Tipo
(
INCIDENTE_TIPO_ID             int IDENTITY(1,1), --PK
INCIDENTE_TIPO_DETALLE              varchar(255),
PRIMARY KEY (INCIDENTE_TIPO_ID)
)

CREATE TABLE GDD_EXPRESS.Incidente_Bandera
(
INCIDENTE_BANDERA_ID                int IDENTITY(1,1), --PK
INCIDENTE_BANDERA_DETALLE           varchar(255),
PRIMARY KEY (INCIDENTE_BANDERA_ID)
)


CREATE TABLE GDD_EXPRESS.Incidente
(
INCIDENTE_ID                        int IDENTITY(1,1), --PK
INCIDENTE_BANDERA_ID                int, --FK
INCIDENTE_TIPO_ID             int, --FK
INCIDENTE_TIEMPO              decimal(18,2),
INCIDENTE_SECTOR_ID                 int, --FK
PRIMARY KEY (INCIDENTE_ID),
FOREIGN KEY (INCIDENTE_BANDERA_ID) REFERENCES GDD_EXPRESS.Incidente_Bandera
(INCIDENTE_BANDERA_ID),
FOREIGN KEY (INCIDENTE_TIPO_ID) REFERENCES GDD_EXPRESS.Incidente_Tipo
(INCIDENTE_TIPO_ID)
)
```

Tabla incidente_auto_carrera

Definimos que en un incidente debe participar AL MENOS un auto o (o muchos). Por otro lado, un auto, puede o no participar de un incidente (o puede tener muchos en varias carreras) por lo tanto, tendríamos una relación MUCHOS A MUCHOS entre auto_carrera e incidente. Entonces generamos una tabla intermedia.



En este caso se decide establecer como **PRIMARY KEY** la combinación de ambas FK.

```
CREATE TABLE GDD_EXPRESS.Incidente_Auto_Carrera
(
INCIDENTE_ID                        int IDENTITY(1,1), --PK, FK
INCIDENTE_AUTO_CARRERA_ID           int, --PK, FK --ACID
INCIDENTE_NUMERO_VUELTA             decimal(18,0),
PRIMARY KEY(INCIDENTE_ID, INCIDENTE_AUTO_CARRERA_ID),
FOREIGN KEY (INCIDENTE_AUTO_CARRERA_ID) REFERENCES GDD_EXPRESS.Auto_Carrera
(AUTO_CARRERA_ID)
)
```

Componentes de un Auto

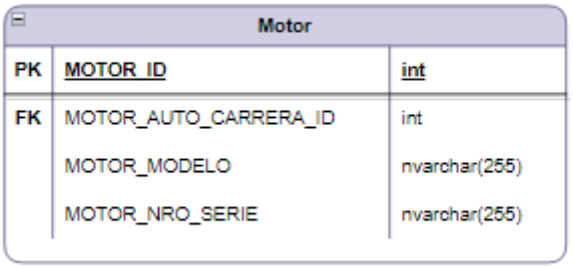
Para evitar la redundancia de datos de los distintos componentes que pertenecen a un auto, definimos y creamos las correspondientes tablas para cada uno de ellos.

Se decide establecer un **“componente”_id** como **PRIMARY KEY** que será generado por un **IDENTITY** para cada uno de los “componentes”. Se descarta el uso del número de serie dado que el mismo es del tipo varchar.

Cada uno de los componentes tendrá como **FOREIGN KEY** a **id_auto_carrera** ya que asumimos que la relación entre los componentes y el auto se corresponden con una carrera en particular, y como se mencionó en el componente anterior, definimos una única PK autogenerada para auto_carrera.

Tabla motor

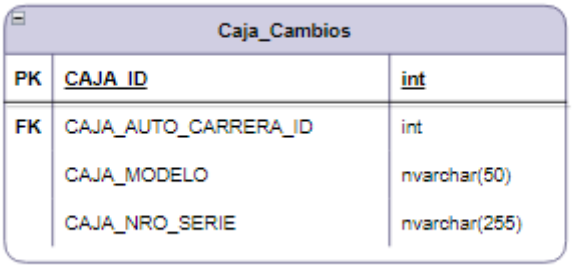
Es una abstracción del motor de un auto, se detallarán modelo y número de serie.




```
CREATE TABLE GDD_EXPRESS.Motor
(
MOTOR_ID                        int IDENTITY(1,1), --PK
MOTOR_AUTO_CARRERA_ID                 int, --FK
MOTOR_MODELO                          nvarchar(255),
MOTOR_NRO_SERIE                       nvarchar(255),
PRIMARY KEY (MOTOR_ID),
FOREIGN KEY (MOTOR_AUTO_CARRERA_ID) REFERENCES GDD_EXPRESS.Auto_Carrera
(AUTO_CARRERA_ID)
)
```

Tabla caja_cambios

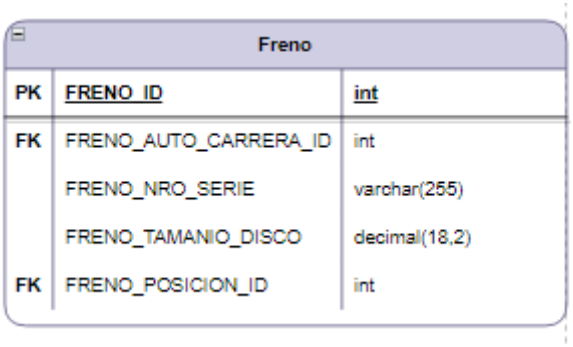
Es una abstracción del motor de un auto, se detallarán modelo y número de serie.




```
CREATE TABLE GDD_EXPRESS.Caja_Cambios
(
CAJA_ID                               int IDENTITY(1,1), --PK
CAJA_AUTO_CARRERA_ID                  int, --FK
CAJA_MODELO                     nvarchar(50),
CAJA_NRO_SERIE                        nvarchar(255),
PRIMARY KEY (CAJA_ID),
FOREIGN KEY (CAJA_AUTO_CARRERA_ID) REFERENCES GDD_EXPRESS.Auto_Carrera
(AUTO_CARRERA_ID)
)
```

Tabla freno

Es una abstracción de un freno de un auto, se detallarán el tamaño del disco, número de serie, y la posición en el auto.

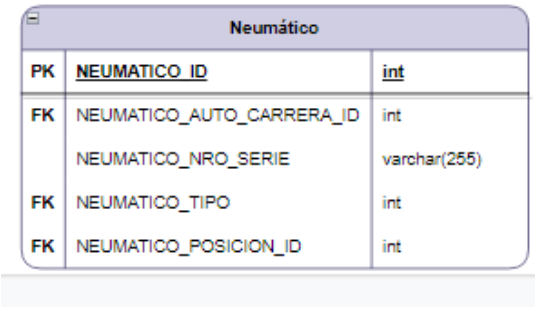



```
CREATE TABLE GDD_EXPRESS.Incidente_Auto_Carrera
(
INCIDENTE_ID                        int IDENTITY(1,1), --PK, FK
INCIDENTE_AUTO_CARRERA_ID           int, --PK, FK --ACID
INCIDENTE_NUMERO_VUELTA             decimal(18,0),
PRIMARY KEY(INCIDENTE_ID, INCIDENTE_AUTO_CARRERA_ID),
FOREIGN KEY (INCIDENTE_AUTO_CARRERA_ID) REFERENCES GDD_EXPRESS.Auto_Carrera
(AUTO_CARRERA_ID)
)
```

Tabla neumático

Es una abstracción de un neumático de un auto, se detallarán el número de serie, el tipo, y la posición en el auto.

Cabe mencionar que el procedimiento es crear primero las tablas posicion y tipo_neumatico, y luego la tabla neumático.

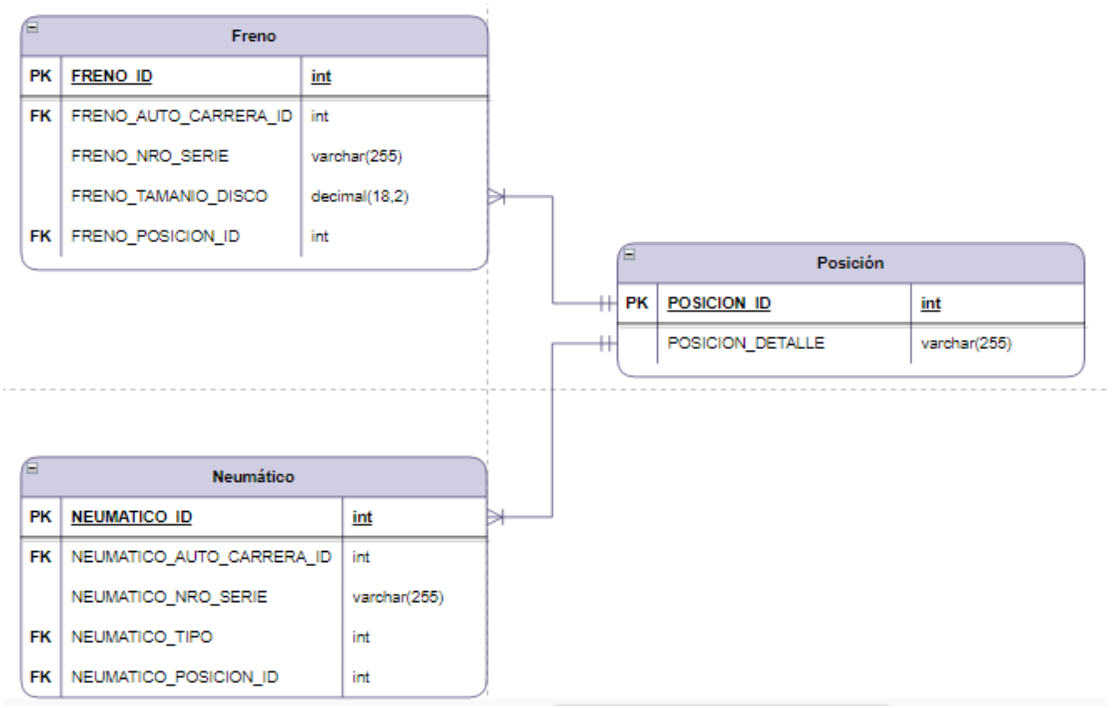



```
CREATE TABLE GDD_EXPRESS.Neumatico
(
NEUMATICO_ID                        int IDENTITY(1,1), --PK
NEUMATICO_AUTO_CARRERA_ID           int, --FK
NEUMATICO_NRO_SERIE                 nvarchar(255),
NEUMATICO_TIPO_ID             int, --FK
```

```
NEUMATICO_POSICION_ID               int, --FK
PRIMARY KEY (NEUMATICO_ID),
FOREIGN KEY (NEUMATICO_AUTO_CARRERA_ID) REFERENCES GDD_EXPRESS.Auto_Carrera
(AUTO_CARRERA_ID),
FOREIGN KEY (NEUMATICO_TIPO_ID) REFERENCES GDD_EXPRESS.Neumatico_Tipo
(NEUMATICO_TIPO_ID),
FOREIGN KEY (NEUMATICO_POSICION_ID) REFERENCES GDD_EXPRESS.Posicion (POSICION_ID)
)
```

Tabla posición

Dado que un auto tiene cuatro neumáticos y cuatro frenos y estos pueden encontrarse en distintas posiciones, para eliminar la repetición de esa posición en varios registros de ambas tablas, se abstrae la posición a una nueva tabla POSICIÓN



Se decide establecer un **posicion_id** como **PRIMARY KEY** que será generado por un **IDENTITY** y posicion_detalle contendrá la descripción de la posición. No deberán repetirse.

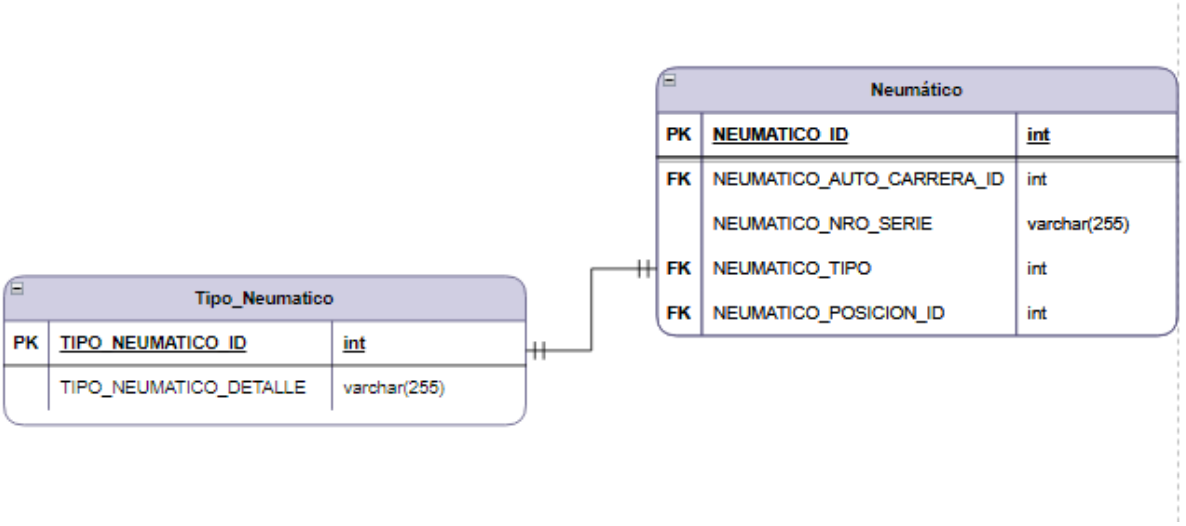
Pudimos relevar que todos los neumáticos que fueron cambiados, también fueron medidos por telemetría. Por lo que concluímos en que todos los neumáticos para la tabla Neumático pueden ser extraídos de la telemetría.

```
CREATE TABLE GDD_EXPRESS.Posicion
(
POSICION_ID                    int IDENTITY(1,1), --PK
POSICION_DETALLE               nvarchar(255),
PRIMARY KEY (POSICION_ID))
```

Tabla tipo_neumatico

Similar a casos anteriores, dado que el tipo de un neumático se puede repetir muchas veces entre distintos registros, y para evitar la redundancia, se abstrae el tipo_neumatico en una tabla.

Se decide establecer un **tipo_neumatico_id** como **PRIMARY KEY** que será generado por un **IDENTITY** y tipo_neumatico_detalle contendrá la descripción del tipo (duro, blando, etc). No deberán repetirse.




```
CREATE TABLE GDD_EXPRESS.Neumatico_Tipo
(
NEUMATICO_TIPO_ID             int IDENTITY(1,1), --PK
NEUMATICO_TIPO_DETALLE              varchar(255),
PRIMARY KEY (NEUMATICO_TIPO_ID)
)
```

Tabla box_parada

Generamos una tabla para tener toda la información referida a un parada en box, la cual queda de la siguiente manera:

Una box_parada hace referencia a un auto en una carrera mediante la FK box_auto_carrera.

De nuevo, se decide establecer un **box_parada_id** como **PRIMARY KEY** que será generado por un **IDENTITY.**

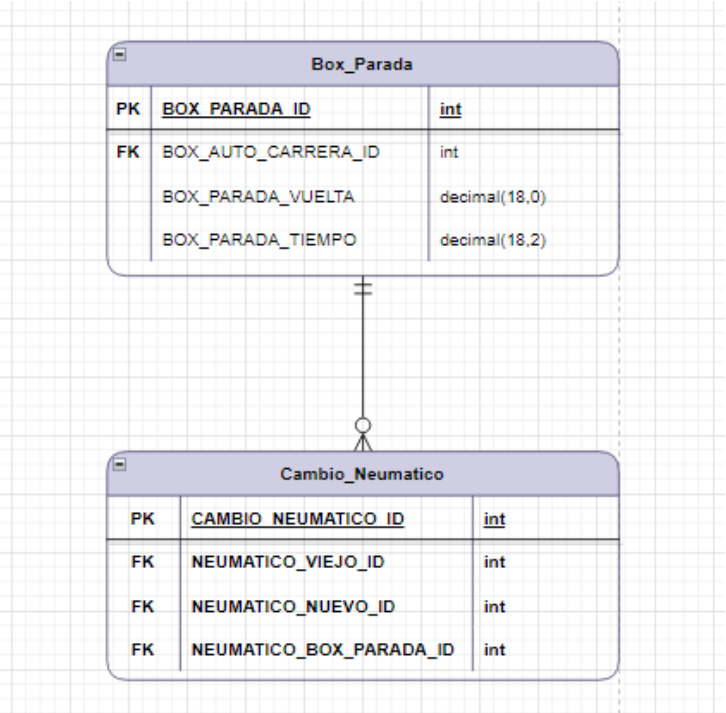


Tabla cambio_neumatico

A su vez, dado que una parada box no siempre tiene un cambio de neumático, se decide modelar esta tabla con una FK a la parada en box, y otra dos FK a los neumáticos que fueron intercambiados, neumatico_viejo_id y neumatico_nuevo_id

De nuevo, se decide establecer un **box_parada_id** como **PRIMARY KEY** que será generado por un **IDENTITY.**

```
CREATE TABLE GDD_EXPRESS.Box_Parada
(
BOX_PARADA_ID                       int IDENTITY(1,1), --PK
BOX_AUTO_CARRERA_ID           int, --FK
BOX_PARADA_VUELTA             decimal(18,0),
BOX_PARADA_TIEMPO             decimal(18,2),
PRIMARY KEY (BOX_PARADA_ID),
FOREIGN KEY (BOX_AUTO_CARRERA_ID) REFERENCES GDD_EXPRESS.Auto_Carrera
(AUTO_CARRERA_ID)
)

CREATE TABLE GDD_EXPRESS.Cambio_Neumatico
(
CAMBIO_NEUMATICO_ID                 int IDENTITY(1,1),
BOX_PARADA_ID                       int, --PK, FK
NEUMATICO_NUEVO_ID                        int, --PK, FK
NEUMATICO_VIEJO_ID                        int,
PRIMARY KEY (CAMBIO_NEUMATICO_ID),
```

```
FOREIGN KEY (BOX_PARADA_ID) REFERENCES GDD_EXPRESS.Box_Parada (BOX_PARADA_ID),
FOREIGN KEY (NEUMATICO_NUEVO_ID) REFERENCES GDD_EXPRESS.Neumatico (NEUMATICO_ID),
FOREIGN KEY (NEUMATICO_VIEJO_ID) REFERENCES GDD_EXPRESS.Neumatico (NEUMATICO_ID))
```

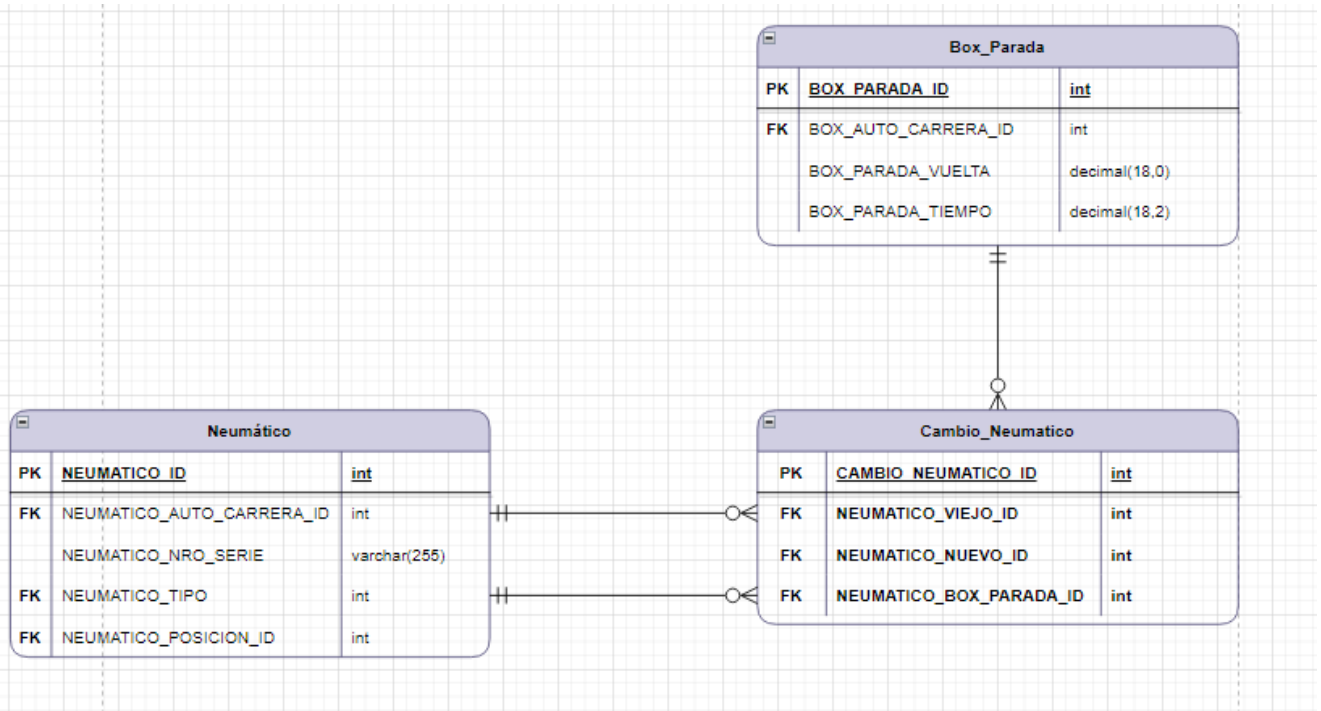
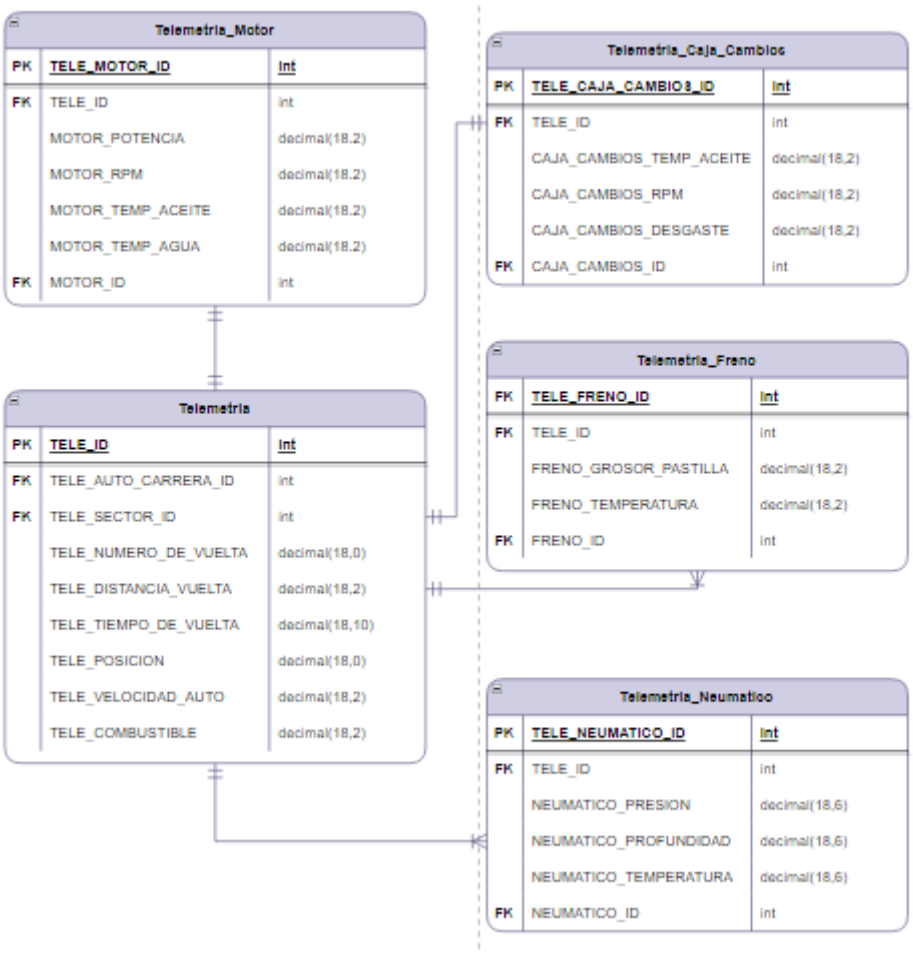


Tabla telemetría

Decidimos normalizar los datos de la telemetría de acuerdo a los componentes del auto. Se tendrá una tabla general de telemetría que quedará de la siguiente manera:
Contiene: el número de vuelta, la distancia de la vuelta, el tiempo de vuelta, la posición,, la velocidad y el combustible del auto en ese momento.
En este caso asumimos que la Distancia Carrera es un dato calculable, por lo que decidimos no incluirlo.
Además contiene una FK al sector donde se ejecutó la medición tele_sector_id, y una FK al auto en la carrera, es decir a auto_carrera con tele_auto_carrera_id.

Al igual que en casos anteriores, se decide establecer un **tele_id** como **PRIMARY KEY** que será generado por un **IDENTITY.** Lo mismo sucede con cada uno de los componentes de la telemetría que se listan a continuación y que ya fueron presentados en el gráfico anterior.




```
CREATE TABLE GDD_EXPRESS.Telemetria
(
TELE_ID                              int, --PK
TELE_AUTO_CARRERA_ID                 int, --FK
TELE_SECTOR_ID                       int, --FK
TELE_NUMERO_DE_VUELTA                decimal(18,0),
TELE_DISTANCIA_VUELTA                decimal(18,2),
TELE_TIEMPO_DE_VUELTA                decimal(18,10),
TELE_POSICION                        decimal(18,0),
TELE_VELOCIDAD_AUTO                  decimal(18,2),
TELE_COMBUSTIBLE               decimal(18,2),
PRIMARY KEY (TELE_ID),
FOREIGN KEY (TELE_AUTO_CARRERA_ID) REFERENCES GDD_EXPRESS.Auto_Carrera
(AUTO_CARRERA_ID),
FOREIGN KEY (TELE_SECTOR_ID) REFERENCES GDD_EXPRESS.Sector (SECTOR_ID)
)
```

Tabla telemetria_motor

Es una tabla que contiene las mediciones relacionadas a un motor, por ejemplo: potencia, rpm, temperatura del aceite, temperatura del agua.

Tiene como referencia el motor medido, mediante la FK motor_id, y además la telemetría a la cual pertenece, mediante la FK tele_id.

```
CREATE TABLE GDD_EXPRESS.Telemetria_Motor
(
TELE_MOTOR_ID                       int, --PK
TELE_ID                             int, --FK
MOTOR_POTENCIA                      decimal(18,2),
MOTOR_RPM                     decimal(18,2),
MOTOR_TEMP_ACEITE             decimal(18,2),
MOTOR_TEMP_AGUA                     decimal(18,2),
MOTOR_ID                      int, --FK
PRIMARY KEY (TELE_MOTOR_ID),
FOREIGN KEY (TELE_ID) REFERENCES GDD_EXPRESS.Telemetria (TELE_ID),
FOREIGN KEY (MOTOR_ID) REFERENCES GDD_EXPRESS.Motor (MOTOR_ID)
)
```

Tabla telemetria_caja_cambios

Es una tabla que contiene las mediciones relacionadas a una caja de cambios, por ejemplo: temperatura del aceite, rpm y desgaste.

Tiene como referencia la caja de cambios medida, mediante la FK caja_cambios_id, y además la telemetría a la cual pertenece, mediante la FK tele_id.

```
CREATE TABLE GDD_EXPRESS.Telemetria_Caja_Cambios
(
TELE_CAJA_ID                        int, --PK
TELE_ID                             int, --FK
CAJA_TEMP_ACEITE              decimal(18,2),
CAJA_RPM                      decimal(18,2),
CAJA_DESGASTE                       decimal(18,2),
CAJA_ID                             int, --FK
PRIMARY KEY (TELE_CAJA_ID),
FOREIGN KEY (TELE_ID) REFERENCES GDD_EXPRESS.Telemetria (TELE_ID),
FOREIGN KEY (CAJA_ID) REFERENCES GDD_EXPRESS.Caja_Cambios (CAJA_ID)
)
```

Tabla telemetria_freno

Es una tabla que contiene las mediciones relacionadas a un freno, por ejemplo: grosor de la pastilla, y la temperatura.

Tiene como referencia el freno medido, mediante la FK freno_id, y además la telemetría a la cual pertenece, mediante la FK tele_id.

Cabe destacar que al ser cuatro los frenos medidos en una telemetría, a diferencia del motor, esta relación es de 1 a muchos (4).

```
CREATE TABLE GDD_EXPRESS.Telemetria_Freno
(
TELE_FRENO_ID                       int, --PK
TELE_ID                             int, --FK
FRENO_TAMANIO_DISCO                 decimal(18,2),
FRENO_TEMPERATURA             decimal(18,2),
FRENO_ID                      int, --FK
PRIMARY KEY (TELE_FRENO_ID),
FOREIGN KEY (TELE_ID) REFERENCES GDD_EXPRESS.Telemetria (TELE_ID),
FOREIGN KEY (FRENO_ID) REFERENCES GDD_EXPRESS.Freno (FRENO_ID)
)
```

Tabla telemetria_neumaticos

Es una tabla que contiene las mediciones relacionadas a un neumático, por ejemplo: presión, profundidad y temperatura.

Tiene como referencia el freno medido, mediante la FK neumatico_id, y además la telemetría a la cual pertenece, mediante la FK tele_id.

Cabe destacar que al ser cuatro los neumáticos, se da la misma situación que con los frenos. Hay cuatro telemetrías de frenos para una telemetría, por lo tanto la relación es de 1 a muchos (4).

```
CREATE TABLE GDD_EXPRESS.Telemetria_Neumatico
(
TELE_NEUMATICO_ID             int, --PK
TELE_ID                             int, --FK
NEUMATICO_PRESION             decimal(18,6),
NEUMATICO_PROFUNDIDAD               decimal(18,6),
NEUMATICO_TEMPERATURA               decimal(18,6),
NEUMATICO_ID                        int, --FK
PRIMARY KEY (TELE_NEUMATICO_ID),
FOREIGN KEY (TELE_ID) REFERENCES GDD_EXPRESS.Telemetria (TELE_ID),
FOREIGN KEY (NEUMATICO_ID) REFERENCES GDD_EXPRESS.Neumatico (NEUMATICO_ID)
)
```

2. Migración - Aspectos Generales

En este apartado, se detallarán las decisiones tomadas en el desarrollo de la migración de datos en orden tal como se encuentra en el script de creación inicial.

2.1 Eliminación de cualquier objeto existente

Con el objeto de poder automatizar la creación de las distintas abstracciones que implementamos mediante estructuras varias, y repetir la operación de creación de dichas estructuras procedemos a eliminar toda aparición de objeto que será generado por el script. Dicha tarea engloba tanto a las tablas, funciones, vistas, procedimientos e índices. Teniendo en cuenta que el último objeto por borrar será el esquema.

Secuencia de Eliminación de tablas

En el caso de las tablas tuvimos en cuenta que la secuencia de borrado de estas era fundamental para evitar tener inconvenientes con la **regla de integridad referencial**, donde entablamos las relaciones mediante las claves foráneas respetando el dominio del problema en cuestión.

2.2 Creación de objetos necesarios para la migración

Creación del esquema

Dado que el enunciado pide que cada objeto sea creado en el esquema cuyo nombre debe corresponder con el del grupo procedemos a crear un **esquema** llamado **“GDD_EXPRESS”.** Cada objeto de los creados a continuación será creado dentro de este esquema.

Creación de las tablas

En esta sección procedemos a crear las tablas que corresponden al diagrama de entidad – relación detallado anteriormente las cuales contendrán los datos migrados de la tabla maestra.
Tuvimos que establecer un orden para la creación de las tablas ya que algunos atributos de estas referencian a través de una FOREING KEY a otras tablas, por lo tanto, primero deben ser creadas las tablas que serán referenciadas y luego las que hacen referencia.

En esta sección se establecen las PRIMARY KEY de cada tabla cuya conformación se encuentra justificada en el apartado de diagrama entidad – relación.

Utilizamos el constraint IDENTITY para establecer que los atributos que conforman PRIMARY KEYS se incrementen en una unidad su valor cada vez que se inserte un registro en una tabla a la hora de realizar la migración de datos.

Creación de funciones auxiliares

A modo de evitar disponer de operaciones repetitivas que podían ser representadas mediante subqueries extensas y en algunos casos quizás hasta complejas, recurrimos a implementar funciones propias que nos permite desarrollar el motor. La utilización de estas funciones auxiliares mejora la performance de la migración.

A continuación, detallamos las funciones auxiliares creadas:

- **fn_id_auto_carrera**: esta función nos permite obtener el número de id de un auto_carrera a partir del número de auto, el modelo y la carrera. Utilizamos esta función para la migración de datos.
- **fn_id_neumatico:** esta función nos permite obtener el número de id de un neumático a partir del número de serie. Utilizamos esta función para la migración de datos.
- **fn_id_pais:** esta función nos permite obtener el id de un país a partir de su detalle. Utilizamos esta función para la migración de datos del piloto, la carrera y la escudería que poseen un país/nacionalidad.
- **fn_id_clima:** esta función nos permite obtener el id del clima a partir de su detalle. Utilizamos esta función para la migración de datos de la carrera.
- **fn_id_sector_tipo:** esta función nos permite obtener el id del tipo de sector a partir de su detalle. Utilizamos esta función para la migración de datos del circuito.
- **fn_id_posicion:** esta función nos permite obtener el id de una posición a partir de su detalle. Utilizamos esta función para la migración de datos de los neumáticos y los frenos.

Creación de Stored Procedures

Detallamos los procedimientos almacenados que creamos para luego ser utilizados para la migración de datos de la tabla maestra al modelo relacional.

La estructura de nuestros stored procedure se compone del bloque principal, y dentro del mismo, para obtener un mejor manejo de errores y debug, realizamos un try/catch para que se notifique en cuál de los módulos fue que ocurrió el error.
Dentro del “try”, abrimos un bloque transaction que en caso de error, se rollbackea.

Migración de Parámetros

Esta es la etapa inicial de toda la migración. Nos ocupamos de dar el alta de las tablas que contienen valores fijos y parametrizables tales como Pais, Sector_Tipo, Incidente_Bandera, etc. El objetivo de estas tablas es favorecer a la normalización y agilizar la posibilidad de la existencia de nuevos valores. Por ejemplo, ante la aparición de un nuevo piloto, con una nacionalidad nueva, es más simple y más centralizado manejarlo mediante una entidad o tabla que guarde todos los países y nacionalidades, la tabla piloto sólo deberá referenciar a la PK de la tabla parámetro.
A nivel funcional realizamos una serie de selects que extraen todos los valores que hoy día tiene la tabla. Le autogeneramos ID con la constraint IDENTITY.

```
INSERT INTO GDD_EXPRESS.Pais (PAIS_DETALLE)
SELECT distinct m.CIRCUITO_PAIS FROM gd_esquema.Maestra m
where m.CIRCUITO_PAIS IS NOT NULL
INSERT INTO GDD_EXPRESS.Pais (PAIS_DETALLE)
SELECT distinct m.ESCUDERIA_NACIONALIDAD FROM gd_esquema.Maestra m
where m.ESCUDERIA_NACIONALIDAD IS NOT NULL
INSERT INTO GDD_EXPRESS.Pais (PAIS_DETALLE)
SELECT distinct m.PILOTO_NACIONALIDAD FROM gd_esquema.Maestra m
where  m.PILOTO_NACIONALIDAD IS NOT NULL
```

Migración de Auto, Escudería y Piloto

En este módulo nos ocupamos de realizar la migración de las entidades Auto y Escudería. La estrategia utilizada fue usar tablas temporales y cursores. Creamos una tabla temporal que contiene todos los datos que representa unívocamente a un auto en la tabla maestra en conjunto con el nombre de la escudería. Entonces una fila de la tabla temporal contiene la relación entre el auto y la escudería. A partir de ese momento lo que queda es migrar los datos para que se adapten a nuestro modelo. Para ello, utilizamos un cursor con el objetivo de crear las escuderías de una en una y utilizar la variable global @@IDENTITY para guardar la key que luego va a utilizar el auto para referenciar a la escudería. Finalmente, mediante un insert damos de alta la tabla Piloto.

```
DECLARE c_escuderia CURSOR FOR
SELECT distinct ESCUDERIA_NOMBRE, ESCUDERIA_NACIONALIDAD FROM gd_esquema.Maestra
WHERE ESCUDERIA_NOMBRE IS NOT NULL


CREATE TABLE #t_autos(
      AUTO_MODELO nvarchar(255),
      AUTO_NUMERO int,
      ESCUDERIA_NOMBRE nvarchar(255)
)
```

Migración de Carreras

Este módulo se ocupa de migrar la entidad Carrera y todas sus relaciones, tales como Sector y Circuito. Utilizamos un cursor que se ocupa de recorrer todo el set de datos perteneciente a una fila Carrera-Sector-Circuito. Recorriendo secuencialmente armamos en orden las tablas.

```
DECLARE c_circuito_carrera CURSOR FOR
SELECT distinct CODIGO_CARRERA, CARRERA_CLIMA, CARRERA_FECHA,
```

```
CARRERA_CANT_VUELTAS, CIRCUITO_CODIGO, CIRCUITO_NOMBRE, CIRCUITO_PAIS FROM
gd_esquema.Maestra
```

Migración de Auto_Carrera

Este store procedure es considerado el troncal de toda la migración. Se ocupa de migrar la entidad Auto_Carrera y consecuente a esto, todos los componentes de un auto en una carrera sumado a sus telemetrías. En principio utilizamos un cursor que va a generar secuencialmente una fila en la entidad Auto_Carrera. Dentro de la iteración del cursor, aprovechamos y damos el alta de todos los componentes del auto utilizando tablas temporales para poder acceder más rápido a los valores y no tener que estar consultando constantemente a la tabla maestra.
Los valores que se relacionan con la entidad Auto_Carrera utilizando las claves que nos deja el cursor: @auto_id, @carrera_id, @auto_carrera_id.
El orden de jerarquía sería: Auto_Carrera, telemetría de ese Auto_Carrera, componentes, telemetría de los componentes.

```
- MIGRACION AUTO_CARRERA
INSERT INTO GDD_EXPRESS.Auto_Carrera(AUTO_ID, CARRERA_ID)
VALUES (@id_auto, @codigo_carrera)

SET @id_auto_carrera = @@IDENTITY

-- MIGRACION TELEMETRIA
INSERT INTO GDD_EXPRESS.Telemetria (TELE_ID, TELE_SECTOR_ID, TELE_COMBUSTIBLE,
TELE_DISTANCIA_VUELTA, TELE_NUMERO_DE_VUELTA, TELE_POSICION,TELE_TIEMPO_DE_VUELTA,
TELE_VELOCIDAD_AUTO, TELE_AUTO_CARRERA_ID)
SELECT t.TELE_AUTO_CODIGO, t.CODIGO_SECTOR, t.TELE_AUTO_COMBUSTIBLE,
t.TELE_AUTO_DISTANCIA_VUELTA, t.TELE_AUTO_NUMERO_VUELTA, t.TELE_AUTO_POSICION,
t.TELE_AUTO_TIEMPO_VUELTA, t.TELE_AUTO_VELOCIDAD, @id_auto_carrera
FROM #t_telemetria t
WHERE t.AUTO_MODELO = @auto_modelo AND t.AUTO_NUMERO = @auto_numero AND
t.CODIGO_CARRERA = @codigo_carrera
```

Migración de Incidentes

En este módulo nos vamos a ocupar de dar de alta las tablas Incidente e Incidente_Auto_Carrera. Utilizamos un cursor para iterar por todos los incidentes que hubieron, con el fin de detectar si en algún incidente hubo más de un auto involucrado.

```
DECLARE c_incidente CURSOR FOR
SELECT distinct i.CODIGO_SECTOR, i.INCIDENTE_BANDERA, i.INCIDENTE_TIEMPO,
i.INCIDENTE_TIPO FROM #t_incidentes i
```

Migración de Parada Box

Esta es la etapa final de la migración. Se tratan las tablas Box_Parada y Cambio_Neumatico. A través de una tabla temporal insertamos las columnas que representan un cambio de neumático y se encuentran dispersas por la tabla para unificarlas en una sola. Luego de obtener la tabla unificada, utilizamos un cursor para iterar sobre las paradas en boxes. Si hubo cambio de neumático, se da el alta y se referencia a la tabla Neumático la key, asociándolo a que es un neumático viejo o uno nuevo.

```
CREATE TABLE #t_neumatico_cambios
(
AUTO_MODELO                   nvarchar(255),
AUTO_NUMERO                   int,
CODIGO_CARRERA                int,
NEUMATICO_NRO_SERIE_NUEVO nvarchar(255),
PARADA_BOX_TIEMPO decimal(18,2),
NEUMATICO_NRO_SERIE_VIEJO nvarchar(255)
)

DECLARE c_box CURSOR FOR SELECT distinct m.AUTO_MODELO, m.AUTO_NUMERO,
m.CODIGO_CARRERA, m.PARADA_BOX_TIEMPO, m.PARADA_BOX_VUELTA FROM gd_esquema.Maestra m
WHERE m.PARADA_BOX_TIEMPO IS NOT NULL
```

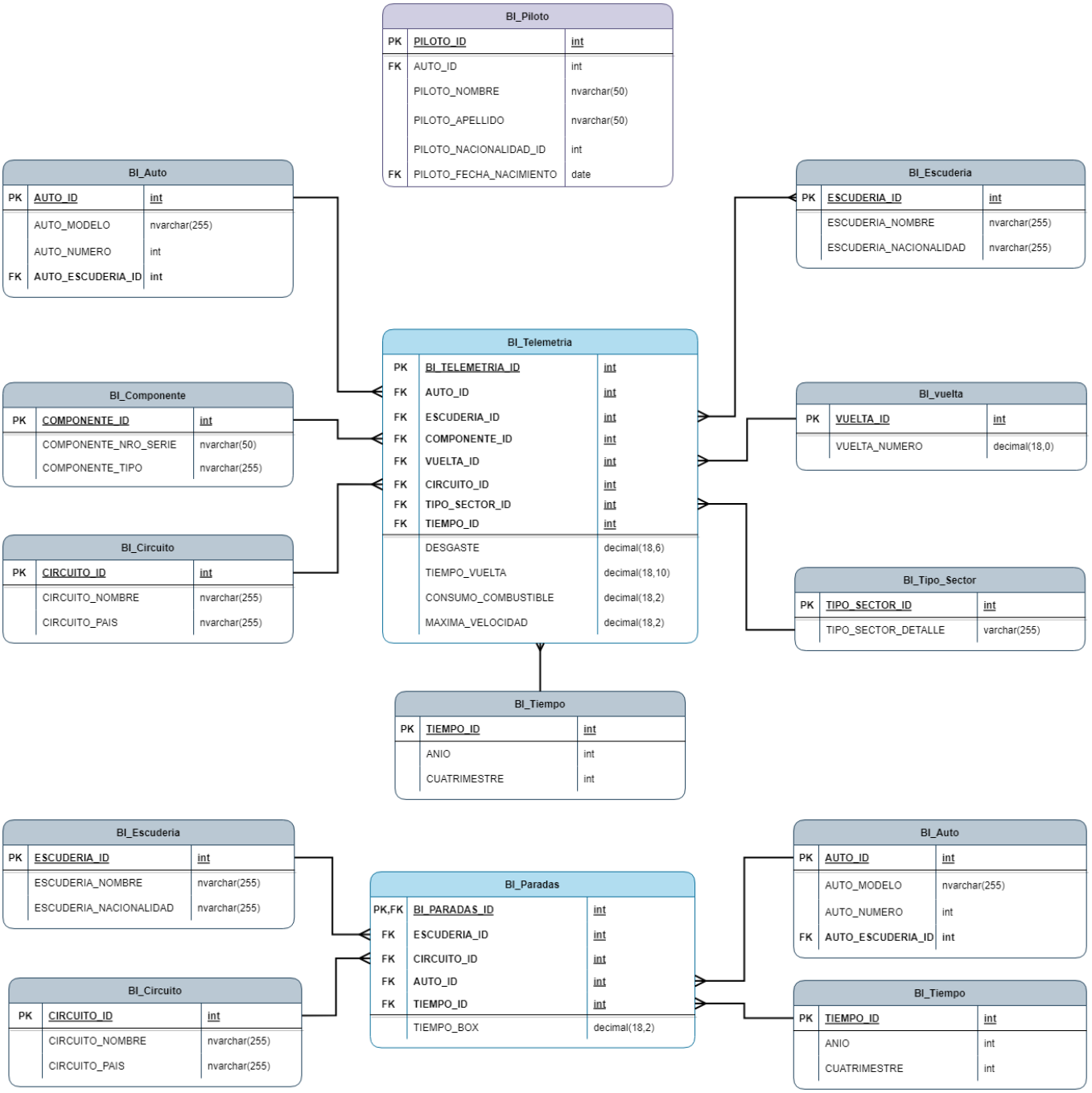
2.3 Ejecución de Stored Procedures para la migración de datos

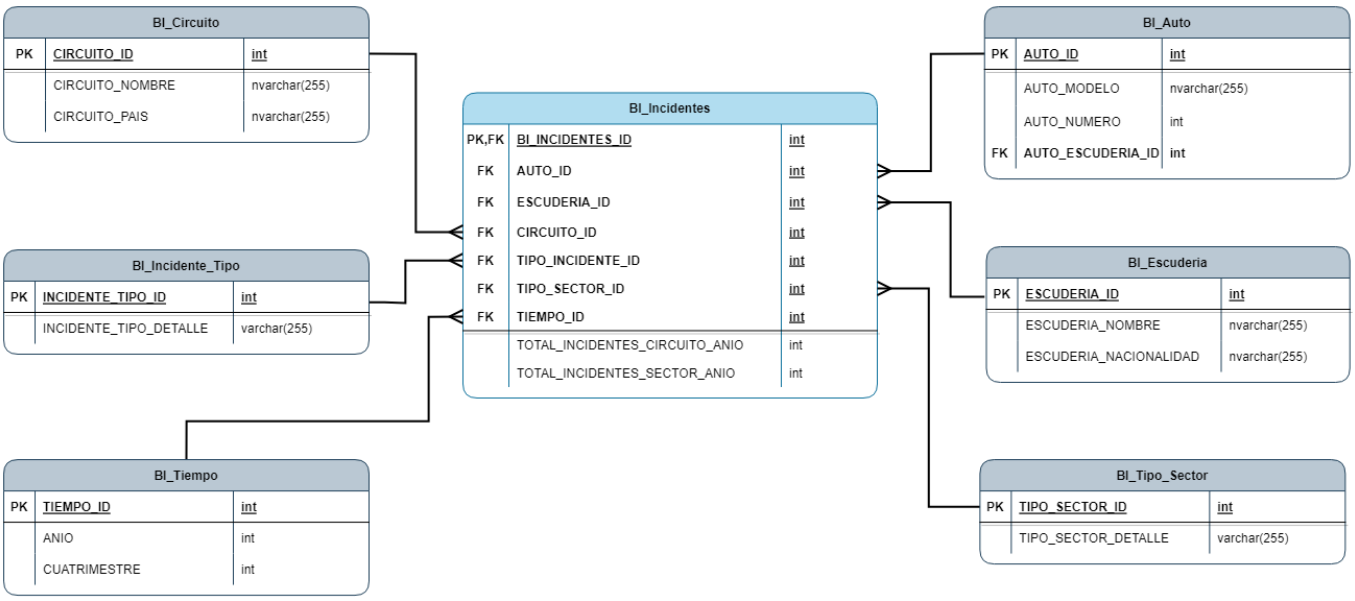
Para realizar la migración de datos de la tabla maestra deben ser ejecutados los stored procedures creados anteriormente respetando el siguiente orden:

```
EXECUTE GDD_EXPRESS.migracion_parametros;
EXECUTE GDD_EXPRESS.migracion_autos_escuderias_pilotos;
EXECUTE GDD_EXPRESS.migracion_carrera;
EXECUTE GDD_EXPRESS.migracion_auto_carrera;
EXECUTE GDD_EXPRESS.migracion_incidentes;
EXECUTE GDD_EXPRESS.migracion_box;
```

3. Modelo de Inteligencia de Negocios (BI)

En este apartado se detalla el procedimiento utilizado para el desarrollo del modelo de Business Intelligence. A continuación, se presenta el DER (separado por partes) correspondiente al Modelo BI:
*Se adjunta igualmente una copia del mismo en la entrega del proyecto, el nombre es **DER-BI.png***





3.1 Borrado Previo

Igual que en la sección anterior, antes de realizar la migración se realiza un borrado de objetos de base de datos que hayan sido creados por este script. La modalidad es la misma y tiene como objetivo evitar conflictos al ejecutar el script en reiteradas ocasiones. El script no funcionará si anteriormente no se ejecuta **script_creación_inicial.sql.**

3.2 Modelo Estrella

Tal y como se vio en clase, se decide utilizar el modelo estrella para confeccionar el modelo de *Business Intelligence.*

3.3 Tablas de hechos (Fact tables)

Para confeccionar el modelo estrella se han definido las siguientes tablas de hechos:

- **BI_Telemetria:** En esta tabla se detallan todas las mediciones que nos van a ser útiles para la realización de las vistas.
- **BI_Paradas**: En esta tabla se detallan las paradas en box que ocurrieron, guardando el tiempo que tomó la misma.
- **BI_Incidentes**: En esta tabla se detallan los incidentes que ocurrieron, registrando su cantidad por sector o circuito.

Para confeccionar las tablas descritas anteriormente se han tenido en cuenta las siguientes dimensiones

BI_Telemetria

- auto
- escudería

- componente
- vuelta
- circuito
- tipo sector
- tiempo

BI_Paradas

- escudería
- circuito
- auto
- tiempo

BI_Incidentes

- escudería
- circuito
- tipo incidente
- tipo sector
- tiempo

Se decide separar en distintas tablas de hechos (distintos Data Marts) debido a que, si fuese una sola tabla de hechos, habría que lidiar con claves nulas, así como también atributos nulos, lo cual complica las operaciones sobre las tablas que son necesarias para confeccionar el modelo.

3.4 Migración hacia el modelo de Business Intelligence

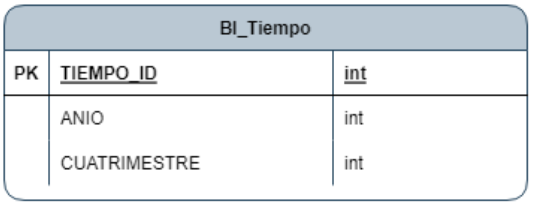
Para poder confeccionar el modelo se decide migrar las tablas necesarias, que pasarán a ser las dimensiones del modelo.

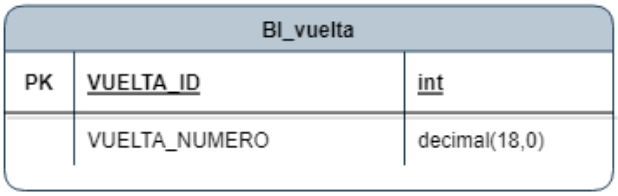
Tablas migradas:

- BI_Auto
- BI_Tipo_Incidente
- BI_Piloto
- BI_Escuderia
- BI_Circuito
- BI_Tipo_Sector

3.5 Tablas confeccionadas para el modelo de Business Intelligence

Para facilitar la confección del modelo, se han creado las siguientes tablas:





BI_Tiempo y BI_Vuelta para representar las dimensiones pedidas y faltantes en la mayoría de los hechos mencionados.

BI_Componente, para tener cada uno de los componentes del auto en la misma tabla, y así evitar migrar por separado Neumáticos, Motor, Caja de Cambios y Frenos.

BI_Componente		
PK	COMPONENTE_ID	int
	COMPONENTE_NRO_SERIE	nvarchar(50)
	COMPONENTE_TIPO	nvarchar(255)

3.6 Proceso de migración hacia el modelo Business Intelligence

Para migrar los datos del modelo transaccional hacia el modelo de Business Intelligence, se han utilizado, de la misma manera que la migración de la entrega anterior, Stored Procedures que serán detallados más adelante.

A continuación, detallaremos aquellas decisiones tomadas acerca de las migraciones

a) **Tipo Neumático**: No hay forma de obtener el tipo de neumático para todos desde la tabla maestra, con lo cual, se decide dejar el atributo en null.

b) **Dimensión Piloto:** No vimos la necesidad de que esta dimensión pueda ser utilizada por los diversos requerimientos de las vistas, pero de igual manera migramos la dimensión ante la posibilidad de aparición de nuevas necesidades.

c) **Telemetría ante accidente:** Notamos que, ante un accidente, se genera una telemetría que tiene como atributo posición en la carrera igual a 0, y algunos otros valores que pueden afectar el análisis final de los datos y resultados en algunas vistas, por lo que decidimos ignorar dichas telemetrías.

Ejemplo de lo mencionado en c):

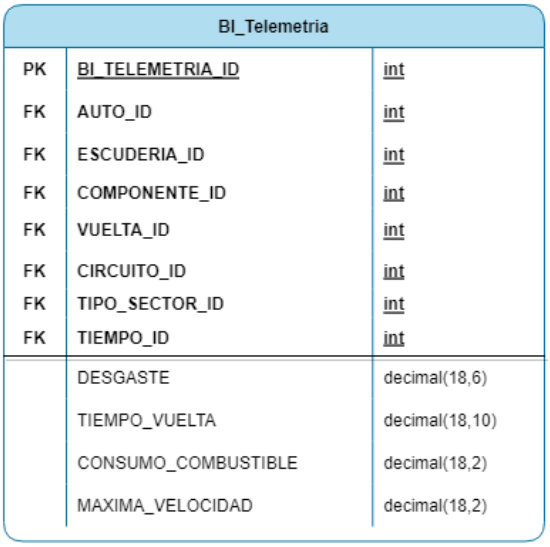
```
SELECT * FROM GDD_EXPRESS.Telemetria
WHERE TELE_AUTO_CARRERA_ID = 90 AND TELE_NUMERO_DE_VUELTA = 3

SELECT * FROM GDD_EXPRESS.Incidente_Auto_Carrera iac WHERE
iac.INCIDENTE_AUTO_CARRERA_ID = 90
```

3.7 Tablas de hechos

Se han creado las siguientes tablas de hechos tal y como se mencionó anteriormente:

BI_Telemetria

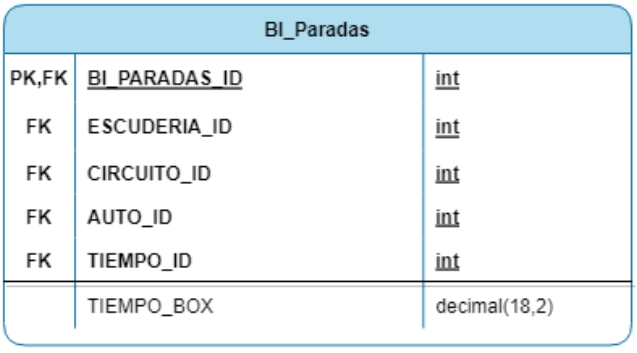
Modelamos como hecho una telemetría, la cual como vemos estará relacionada a un componente de un auto de una escudería, en un sector de una vuelta de un circuito determinado. También sumamos la dimensión tiempo.
Los valores calculados que tenemos aquí son: desgaste del componente, tiempo vuelta, consumo de combustible del auto y máxima velocidad.
Agregamos auto como dimensión al hecho de medición, ya que el nivel de combustible, la velocidad, y el tiempo, son propios de un auto. Por más que no se considere para la consulta en particular.



BI_Telemetria		
PK	BI_TELEMETRIA_ID	int
FK	AUTO_ID	int
FK	ESCUDERIA_ID	int
FK	COMPONENTE_ID	int
FK	VUELTA_ID	int
FK	CIRCUITO_ID	int
FK	TIPO_SECTOR_ID	int
FK	TIEMPO_ID	int
	DESGASTE	decimal(18,6)
	TIEMPO_VUELTA	decimal(18,10)
	CONSUMO_COMBUSTIBLE	decimal(18,2)
	MAXIMA_VELOCIDAD	decimal(18,2)

BI_Paradas

Modelamos como hecho una parada, la cual como vemos estará relacionada a un auto de una escudería, en un circuito determinado. También sumamos la dimensión tiempo.
Los valores calculados que tenemos aquí son: tiempo box que sería el tiempo de parada.

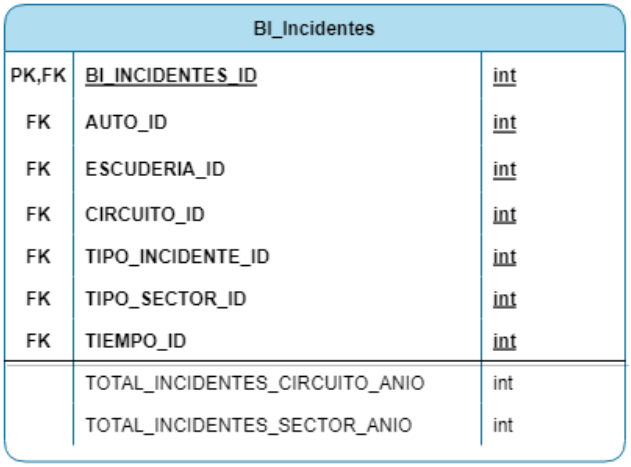


BI_Paradas		
PK,FK	BI_PARADAS_ID	int
FK	ESCUDERIA_ID	int
FK	CIRCUITO_ID	int
FK	AUTO_ID	int
FK	TIEMPO_ID	int
	TIEMPO_BOX	decimal(18,2)

BI_Incidentes

Modelamos como hecho un incidente, el cual como vemos tendrá un tipo que estará relacionado a un auto de una escudería, en un sector de un circuito determinado. Sumamos la dimensión tiempo.
Los valores calculados que tenemos aquí son: los totales de incidentes ocurridos por sector y por circuito.

Consideramos la unicidad de los incidentes. Es decir, si un incidente es producido por un choque, e involucra a más de un auto, el mismo vale como uno sólo.



BI_Incidentes		
PK,FK	BI_INCIDENTES_ID	int
FK	AUTO_ID	int
FK	ESCUDERIA_ID	int
FK	CIRCUITO_ID	int
FK	TIPO_INCIDENTE_ID	int
FK	TIPO_SECTOR_ID	int
FK	TIEMPO_ID	int
	TOTAL_INCIDENTES_CIRCUITO_ANIO	int
	TOTAL_INCIDENTES_SECTOR_ANIO	int

4. Migración hacia el Modelo BI

4.1 Funciones

Implementación de Funciones Adicionales

Desarrollamos las siguientes funciones, que nos facilitan acceder a la información desde distintas consultas. Destacamos cuáles fueron las más importantes con una breve explicación sobre la problemática que soluciona.

- **fn_ cuatrimestre**

Dada una fecha, esta función calcula el cuatrimestre.

- **fn_tiempo_id**
- **fn_vuelta_id**

Ambas facilitan el acceso a las tablas que ya fueron migradas para que las mismas puedan ser referidas por otras. Devuelve el id que refiere a cada tabla mencionada (tiempo y vuelta)

4.2 Stored Procedures

Para poder confeccionar el modelo se decide migrar las tablas necesarias que pasarán a ser las dimensiones del modelo. Realizamos la migración a través de Stored Procedures los cuales cargan los datos en las tablas dimensión y tablas de hechos a partir de las tablas existentes del Modelo Relacional.

Tablas migradas:
Utilizamos los siguientes Stored Procedures:

- **migracion_bi_escuderia**

```
CREATE PROCEDURE GDD_EXPRESS.migracion_bi_escuderia AS
    BEGIN
        INSERT INTO GDD_EXPRESS.BI_Escuderia (escuderia_id, escuderia_nombre,
escuderia_pais)
            SELECT es.ESCUDERIA_ID, es.ESCUDERIA_NOMBRE, pais.PAIS_DETALLE
            FROM GDD_EXPRESS.Escuderia es
            JOIN GDD_EXPRESS.Pais pais ON es.ESCUDERIA_PAIS_ID = pais.PAIS_ID
    END
```

- **migracion_bi_piloto**

```
CREATE PROCEDURE GDD_EXPRESS.migracion_bi_piloto AS
    BEGIN
        INSERT INTO GDD_EXPRESS.BI_Piloto (PILOTO_ID, PILOTO_NOMBRE,
PILOTO_APELLIDO, PILOTO_NACIONALIDAD, PILOTO_FECHA_NACIMIENTO, PILOTO_AUTO_ID)
            SELECT p.PILOTO_ID, p.PILOTO_NOMBRE, p.PILOTO_APELLIDO,
                   pais.PAIS_DETALLE, p.PILOTO_FECHA_NACIMIENTO, p.PILOTO_AUTO_ID
            FROM GDD_EXPRESS.Piloto p
            JOIN GDD_EXPRESS.Pais pais ON p.PILOTO_NACIONALIDAD = pais.PAIS_ID
    END
```

- **migracion_bi_auto**

```
CREATE PROCEDURE GDD_EXPRESS.migracion_bi_auto AS
    BEGIN
        INSERT INTO GDD_EXPRESS.BI_Auto (auto_id, auto_modelo, auto_escuderia_id,
auto_numero)
            SELECT AUTO_ID, AUTO_MODELO, AUTO_ESCUDERIA_ID, AUTO_NUMERO
            FROM GDD_EXPRESS.Auto
    END
```

- **migracion_bi_tipo_incidente**

```
CREATE PROCEDURE GDD_EXPRESS.migracion_bi_tipo_incidente AS
    BEGIN
        INSERT INTO GDD_EXPRESS.BI_Tipo_Incidente(TIPO_INCIDENTE_ID,
TIPO_INCIDENTE_DETALLE)
            SELECT it.INCIDENTE_TIPO_ID, it.INCIDENTE_TIPO_DETALLE
            FROM GDD_EXPRESS.Incidente_Tipo it
    END
```

- **migracion_bi_circuito**

```
CREATE PROCEDURE GDD_EXPRESS.migracion_bi_circuito AS
    BEGIN
        INSERT INTO GDD_EXPRESS.BI_Circuito (circuito_id, circuito_nombre,
circuito_pais)
           SELECT c.CIRCUITO_ID, c.CIRCUITO_NOMBRE, pais.PAIS_DETALLE
           FROM GDD_EXPRESS.Circuito c
           JOIN GDD_EXPRESS.Pais pais ON c.CIRCUITO_PAIS_ID = pais.PAIS_ID
    END
```

- **migracion_bi_tipo_sector**

```
CREATE PROCEDURE GDD_EXPRESS.migracion_bi_tipo_sector AS
    BEGIN
        INSERT INTO GDD_EXPRESS.BI_Tipo_Sector(TIPO_SECTOR_ID,
TIPO_SECTOR_DETALLE)
           SELECT s.SECTOR_TIPO_ID, s.SECTOR_TIPO_DETALLE
           FROM GDD_EXPRESS.Sector_Tipo s
    END
```

Tienen como objetivo principal migrar la integridad de las tablas pertenecientes al modelo relacional, desnormalizado algunos campos tales como país o nacionalidad y otros campos descartados que no aportan al dominio BI.

Tablas confeccionadas para el modelo de Business Intelligence. Para facilitar la confección del modelo, se han creado las siguientes tablas:

- **migracion_bi_tiempo**

```
CREATE PROCEDURE GDD_EXPRESS.migracion_bi_tiempo AS
    BEGIN
        INSERT INTO GDD_EXPRESS.BI_Tiempo(TIEMPO_ANIO, TIEMPO_CUATRIMESTRE)
           SELECT distinct YEAR(CARRERA_FECHA),
                  GDD_EXPRESS.fn_cuatrimestre(CARRERA_FECHA)
            FROM GDD_EXPRESS.Carrera
    END
```

- **migracion_bi_vueltas**

```
CREATE PROCEDURE GDD_EXPRESS.migracion_bi_vueltas AS
    BEGIN
        INSERT INTO GDD_EXPRESS.BI_Vuelta(VUELTA_NUMERO)
           SELECT DISTINCT TELE_NUMERO_DE_VUELTA FROM GDD_EXPRESS.Telemetria
    END
```

Tablas de Hechos:

- **migracion_bi_telemetria**

Sólo se incluye el ejemplo de frenos, el procedure contiene la migración de cada uno de los componentes, la cual puede apreciarse en el script completo:

```
CREATE PROCEDURE GDD_EXPRESS.migracion_telemetria AS
    BEGIN

      declare @bi_componente_id int

      --NEUMATICOS
      INSERT INTO GDD_EXPRESS.BI_Componente(COMPONENTE_TIPO)
      VALUES ('NEUMATICO')
      SET @bi_componente_id = @@IDENTITY

      INSERT INTO GDD_EXPRESS.BI_Telemetria(AUTO_ID, ESCUDERIA_ID, COMPONENTE_ID,
VUELTA_ID, CIRCUITO_ID, TIEMPO_ID, DESGASTE)
      SELECT ac.AUTO_ID, a.AUTO_ESCUDERIA_ID, @bi_componente_id,
GDD_EXPRESS.fn_vuelta_id(t.TELE_NUMERO_DE_VUELTA), c.CARRERA_CIRCUITO_ID ,
GDD_EXPRESS.fn_tiempo_id(c.CARRERA_FECHA),
            MAX(tn.NEUMATICO_PROFUNDIDAD) - MIN(tn.NEUMATICO_PROFUNDIDAD)
            FROM GDD_EXPRESS.Telemetria t
            JOIN GDD_EXPRESS.Telemetria_Neumatico tn on t.TELE_ID = tn.TELE_ID
            JOIN GDD_EXPRESS.Auto_Carrera ac on ac.AUTO_CARRERA_ID =
t.TELE_AUTO_CARRERA_ID
            JOIN GDD_EXPRESS.Carrera c on c.CARRERA_ID = ac.CARRERA_ID
            JOIN GDD_EXPRESS.Auto a on a.AUTO_ID = ac.AUTO_ID
            WHERE t.TELE_POSICION <> 0
            GROUP BY t.TELE_NUMERO_DE_VUELTA, c.CARRERA_CIRCUITO_ID,
c.CARRERA_FECHA, ac.AUTO_ID, a.AUTO_ESCUDERIA_ID, tn.NEUMATICO_ID

       -- PROCEDIMIENTOS SIMILARES PARA FRENOS, CAJA DE CAMBIO Y MOTOR, CAMBIA EL
CÁLCULO DEL DESGASTE.
```

Valor calculado del combustible:

Asumimos que el nivel de combustible es la diferencia entre la primera y última medición, no existe carga de combustible durante la carrera para un auto.

```
      DECLARE c_tiempos_combustible CURSOR FOR
      SELECT a.AUTO_ID, GDD_EXPRESS.fn_vuelta_id(t.TELE_NUMERO_DE_VUELTA),
a.AUTO_ESCUDERIA_ID,
      GDD_EXPRESS.fn_tiempo_id(c.CARRERA_FECHA), c.CARRERA_CIRCUITO_ID,
MAX(t.TELE_TIEMPO_DE_VUELTA),
```

```
      MAX(t.TELE_COMBUSTIBLE) - MIN(t.TELE_COMBUSTIBLE)
      FROM GDD_EXPRESS.Telemetria t
      JOIN GDD_EXPRESS.Sector s on s.SECTOR_ID = t.TELE_SECTOR_ID
      JOIN GDD_EXPRESS.Auto_Carrera ac on ac.AUTO_CARRERA_ID =
t.TELE_AUTO_CARRERA_ID
      JOIN GDD_EXPRESS.Carrera c on c.CARRERA_ID = ac.CARRERA_ID
      JOIN GDD_EXPRESS.Auto a on a.AUTO_ID = ac.AUTO_ID
      WHERE t.TELE_POSICION <> 0
      GROUP BY t.TELE_AUTO_CARRERA_ID, a.AUTO_ID, a.AUTO_ESCUDERIA_ID,
c.CARRERA_FECHA, c.CARRERA_CIRCUITO_ID, t.TELE_NUMERO_DE_VUELTA

      declare @auto_id int
      declare @vuelta_id int
      declare @escuderia_id int
      declare @tiempo_id int
      declare @circ_id int
      declare @tiempo_vuelta decimal(18,10)
      declare @consumo_combustible decimal(18,2)

      OPEN c_tiempos_combustible
      FETCH NEXT FROM c_tiempos_combustible INTO @auto_id, @vuelta_id,
@escuderia_id, @tiempo_id, @circ_id, @tiempo_vuelta, @consumo_combustible
      WHILE (@@FETCH_STATUS = 0)
      BEGIN

            UPDATE GDD_EXPRESS.BI_Telemetria
            SET TIEMPO_VUELTA = @tiempo_vuelta, CONSUMO_COMBUSTIBLE =
@consumo_combustible
            WHERE
            AUTO_ID = @auto_id AND
            ESCUDERIA_ID = @escuderia_id AND
            VUELTA_ID = @vuelta_id AND
            TIEMPO_ID = @tiempo_id AND
            CIRCUITO_ID = @circ_id

            FETCH NEXT FROM c_tiempos_combustible INTO @auto_id, @vuelta_id,
@escuderia_id, @tiempo_id, @circ_id, @tiempo_vuelta, @consumo_combustible
      END

      CLOSE c_tiempos_combustible
      DEALLOCATE c_tiempos_combustible
```

Valor calculado de la máxima velocidad:

```
INSERT INTO GDD_EXPRESS.BI_Telemetria(AUTO_ID, CIRCUITO_ID, TIPO_SECTOR_ID,
TIEMPO_ID, MAXIMA_VELOCIDAD)
```

```
      SELECT a.AUTO_ID, c.CARRERA_CIRCUITO_ID, s.SECTOR_TIPO_ID,
GDD_EXPRESS.fn_tiempo_id(c.CARRERA_FECHA), MAX(t.TELE_VELOCIDAD_AUTO) FROM
GDD_EXPRESS.Telemetria t
      JOIN GDD_EXPRESS.Sector s on s.SECTOR_ID = t.TELE_SECTOR_ID
      JOIN GDD_EXPRESS.Auto_Carrera ac on ac.AUTO_CARRERA_ID =
t.TELE_AUTO_CARRERA_ID
      JOIN GDD_EXPRESS.Carrera c on c.CARRERA_ID = ac.CARRERA_ID
      JOIN GDD_EXPRESS.Auto a on a.AUTO_ID = ac.AUTO_ID
      WHERE t.TELE_POSICION <> 0
      GROUP BY a.AUTO_ID, c.CARRERA_CIRCUITO_ID, s.SECTOR_TIPO_ID, c.CARRERA_FECHA
```

- **migracion_bi_paradas**

```
CREATE PROCEDURE GDD_EXPRESS.migracion_bi_paradas AS
    BEGIN
        INSERT INTO GDD_EXPRESS.BI_Paradas(AUTO_ID, ESCUDERIA_ID, CIRCUITO_ID,
TIEMPO_ID, TIEMPO_BOX)
           SELECT a.AUTO_ID, a.AUTO_ESCUDERIA_ID, c.CARRERA_CIRCUITO_ID,
GDD_EXPRESS.fn_tiempo_id(c.CARRERA_FECHA), b.BOX_PARADA_TIEMPO
           FROM GDD_EXPRESS.Box_Parada b
           JOIN GDD_EXPRESS.Auto_Carrera ac on ac.AUTO_CARRERA_ID =
b.BOX_AUTO_CARRERA_ID
           JOIN GDD_EXPRESS.Carrera c on c.CARRERA_ID = ac.CARRERA_ID
           JOIN GDD_EXPRESS.Auto a on a.AUTO_ID = ac.AUTO_ID
    END
```

- **mirgacion_bi_incidentes**

```
CREATE PROCEDURE GDD_EXPRESS.migracion_bi_incidentes AS
    BEGIN
        INSERT INTO GDD_EXPRESS.BI_Incidentes(AUTO_ID, ESCUDERIA_ID, CIRCUITO_ID,
TIPO_INCIDENTE_ID, TIPO_SECTOR_ID, TIEMPO_ID)
           SELECT a.AUTO_ID, a.AUTO_ESCUDERIA_ID, c.CARRERA_CIRCUITO_ID,
i.INCIDENTE_TIPO_ID, s.SECTOR_TIPO_ID, GDD_EXPRESS.fn_tiempo_id(c.CARRERA_FECHA)
           FROM GDD_EXPRESS.Incidente i
           JOIN GDD_EXPRESS.Incidente_Auto_Carrera iac on i.INCIDENTE_ID =
iac.INCIDENTE_ID
           JOIN GDD_EXPRESS.Auto_Carrera ac on ac.AUTO_CARRERA_ID =
iac.INCIDENTE_AUTO_CARRERA_ID
           JOIN GDD_EXPRESS.Carrera c on c.CARRERA_ID = ac.CARRERA_ID
           JOIN GDD_EXPRESS.Auto a on a.AUTO_ID = ac.AUTO_ID
           JOIN GDD_EXPRESS.Sector s on s.SECTOR_ID = i.INCIDENTE_SECTOR_ID
    END
```

```
      DECLARE c_total_anio_circ CURSOR FOR
      SELECT c.CARRERA_CIRCUITO_ID, YEAR(c.CARRERA_FECHA), count(distinct
i.INCIDENTE_ID)
      FROM GDD_EXPRESS.Incidente i
      JOIN GDD_EXPRESS.Incidente_Auto_Carrera iac on i.INCIDENTE_ID =
iac.INCIDENTE_ID
      JOIN GDD_EXPRESS.Auto_Carrera ac on ac.AUTO_CARRERA_ID =
iac.INCIDENTE_AUTO_CARRERA_ID
      JOIN GDD_EXPRESS.Carrera c on c.CARRERA_ID = ac.CARRERA_ID
      GROUP BY c.CARRERA_CIRCUITO_ID, YEAR(c.CARRERA_FECHA)

      declare @circ_id int
      declare @anio int
      declare @total int

      OPEN c_total_anio_circ
      FETCH NEXT FROM c_total_anio_circ INTO @circ_id, @anio, @total
      WHILE (@@FETCH_STATUS = 0)
      BEGIN

            UPDATE GDD_EXPRESS.BI_Incidentes
            SET TOTAL_INCIDENTES_CIRCUITO_ANIO = @total
            WHERE EXISTS (
                SELECT 1
                FROM GDD_EXPRESS.BI_Tiempo t
                WHERE t.TIEMPO_ANIO = @anio AND t.TIEMPO_ID = TIEMPO_ID
            ) AND CIRCUITO_ID = @circ_id

            FETCH NEXT FROM c_total_anio_circ INTO @circ_id, @anio, @total
      END

      CLOSE c_total_anio_circ
      DEALLOCATE c_total_anio_circ

      INSERT INTO GDD_EXPRESS.BI_Tiempo(TIEMPO_ANIO)
      VALUES (2020)
      INSERT INTO GDD_EXPRESS.BI_Tiempo(TIEMPO_ANIO)
      VALUES (2021)

      INSERT INTO GDD_EXPRESS.BI_Incidentes(ESCUDERIA_ID, TIPO_SECTOR_ID,
TIEMPO_ID, TOTAL_INCIDENTES_SECTOR_ANIO)
      SELECT a.AUTO_ESCUDERIA_ID, s.SECTOR_TIPO_ID, (SELECT t.TIEMPO_ID
      FROM GDD_EXPRESS.BI_Tiempo t
      WHERE t.TIEMPO_ANIO = YEAR(c.CARRERA_FECHA) AND t.TIEMPO_CUATRIMESTRE IS
NULL),
      COUNT(distinct i.INCIDENTE_ID) FROM GDD_EXPRESS.Incidente i
      JOIN GDD_EXPRESS.Incidente_Auto_Carrera iac on i.INCIDENTE_ID =
iac.INCIDENTE_ID
```

```
      JOIN GDD_EXPRESS.Auto_Carrera ac on ac.AUTO_CARRERA_ID =
iac.INCIDENTE_AUTO_CARRERA_ID
      JOIN GDD_EXPRESS.Carrera c on c.CARRERA_ID = ac.CARRERA_ID
      JOIN GDD_EXPRESS.Auto a on a.AUTO_ID = ac.AUTO_ID
      JOIN GDD_EXPRESS.Sector s on s.SECTOR_ID = i.INCIDENTE_SECTOR_ID
      GROUP BY a.AUTO_ESCUDERIA_ID, s.SECTOR_TIPO_ID, YEAR(c.CARRERA_FECHA)
```

5. Vistas

Se desarrollaron las vistas correspondientes para cumplir con los requerimientos solicitados en el enunciado.

5.1 Vista - Desgaste promedio por componente

El nombre de la vista creada es **vw_desgaste_promedio_por_componente**

Esta vista nos permite visualizar el desgaste promedio de cada componente de cada auto por vuelta por circuito.
Las columnas que agregamos para mostrar dicha información

AUTO_MODELO
AUTO_NUMERO
COMPONENTE_TIPO
VUELTA_NUMERO
CIRCUITO_NOMBRE
PROMEDIO_DESGASTE

```
CREATE VIEW GDD_EXPRESS.vw_desgaste_promedio_por_componente AS
    SELECT a.AUTO_MODELO, a.AUTO_NUMERO, c.COMPONENTE_TIPO, v.VUELTA_NUMERO,
cir.CIRCUITO_NOMBRE, avg(d.DESGASTE) as promedio_desgaste FROM
GDD_EXPRESS.BI_Telemetria d
      JOIN GDD_EXPRESS.BI_Componente c on c.COMPONENTE_ID = d.COMPONENTE_ID
      JOIN GDD_EXPRESS.BI_Auto a on a.AUTO_ID = d.AUTO_ID
      JOIN GDD_EXPRESS.BI_Vuelta v on v.VUELTA_ID = d.VUELTA_ID
      JOIN GDD_EXPRESS.BI_Circuito cir on cir.CIRCUITO_ID = d.CIRCUITO_ID
      GROUP BY d.CIRCUITO_ID, d.VUELTA_ID, v.VUELTA_NUMERO, cir.CIRCUITO_NOMBRE,
a.AUTO_MODELO, a.AUTO_NUMERO, c.COMPONENTE_TIPO
```

5.2 Vista - Mejor tiempo de vuelta

El nombre de la vista creada es **vw_mejor_tiempo_vuelta_escuderia_x_circuito_anio**

Esta vista nos permite visualizar el mejor tiempo de vuelta de cada escudería por circuito por año.
El mejor tiempo está dado por el mínimo tiempo en que un auto logra realizar una vuelta de un circuito.

Las columnas que agregamos para mostrar dicha información

VUELTA_ID
ESCUDERIA_ID
CIRCUITO_ID
TIEMPO_ANIO
MEJOR_TIEMPO

```
CREATE VIEW GDD_EXPRESS.vw_mejor_tiempo_vuelta_escuderia_x_circuito_anio AS
    SELECT tele.VUELTA_ID, tele.ESCUDERIA_ID, tele.CIRCUITO_ID, t.TIEMPO_ANIO,
min(tele.TIEMPO_VUELTA) as MEJOR_TIEMPO FROM GDD_EXPRESS.BI_Telemetria tele
      JOIN GDD_EXPRESS.BI_Tiempo t on t.TIEMPO_ID = tele.TIEMPO_ID
      where tele.VUELTA_ID = 1
      GROUP BY tele.VUELTA_ID, tele.ESCUDERIA_ID, tele.CIRCUITO_ID, t.TIEMPO_ANIO
```

5.3 Vista - Circuitos con mayor consumo de combustible

El nombre de la vista creada es **vw_top3_circuitos_consumo_combustible**

Esta vista nos permite visualizar los 3 circuitos con mayor consumo de combustible promedio..

Las columnas que agregamos para mostrar dicha información

CIRCUITO NOMBRE
CONSUMO_PROMEDIO

```
CREATE VIEW GDD_EXPRESS.vw_top3_circuitos_consumo_combustible AS
    SELECT TOP 3 c.CIRCUITO_NOMBRE, avg(t.CONSUMO_COMBUSTIBLE) as CONSUMO_PROMEDIO
FROM GDD_EXPRESS.BI_Telemetria t
      JOIN GDD_EXPRESS.BI_Circuito c on c.CIRCUITO_ID = t.CIRCUITO_ID
      GROUP BY t.CIRCUITO_ID, c.CIRCUITO_NOMBRE
      ORDER BY avg(t.CONSUMO_COMBUSTIBLE) desc
```

5.4 Vista - Máxima velocidad de auto por sector

El nombre de la vista creada es **vw_maxima_velocidad_x_sector_auto**

Esta vista nos permite visualizar la máxima velocidad alcanzada por cada auto en cada tipo de sector de cada circuito.

Las columnas que agregamos para mostrar dicha información

AUTO_ID
CIRCUITO_ID
TIPO_SECTOR_ID
AUTO_MODELO,
AUTO_NUMERO
TIPO_SECTOR_DETALLE

CIRCUITO_NOMBRE
MAX_VELOCIDAD

```
CREATE VIEW GDD_EXPRESS.vw_maxima_velocidad_x_sector_auto AS
    SELECT t.AUTO_ID, t.CIRCUITO_ID, t.TIPO_SECTOR_ID, a.AUTO_MODELO,
a.AUTO_NUMERO, ts.TIPO_SECTOR_DETALLE, c.CIRCUITO_NOMBRE, MAX(t.MAXIMA_VELOCIDAD)
as MAX_VELOCIDAD FROM GDD_EXPRESS.BI_Telemetria t
      JOIN GDD_EXPRESS.BI_Auto a on a.AUTO_ID = t.AUTO_ID
      JOIN GDD_EXPRESS.BI_Tipo_Sector ts on ts.TIPO_SECTOR_ID = t.TIPO_SECTOR_ID
      JOIN GDD_EXPRESS.BI_Circuito c on c.CIRCUITO_ID = t.CIRCUITO_ID
      WHERE t.TIPO_SECTOR_ID IS NOT NULL
      GROUP BY t.AUTO_ID, a.AUTO_NUMERO, t.TIPO_SECTOR_ID, t.CIRCUITO_ID,
ts.TIPO_SECTOR_DETALLE, a.AUTO_MODELO, c.CIRCUITO_NOMBRE
```

5.5 Vista - Tiempo promedio por paradas

El nombre de la vista creada es **vw_tiempo_promedio_cuatrimestral_paradas**

Esta vista nos permite visualizar el tiempo promedio que tardó cada escudería en las paradas por cuatrimestre.

Las columnas que agregamos para mostrar dicha información

TIEMPO_ANIO
TIEMPO_CUATRIMESTRE
ESCUDERIA_NOMBRE
TIEMPO_PROMEDIO

```
CREATE VIEW GDD_EXPRESS.vw_tiempo_promedio_cuatrimestral_paradas AS
    SELECT t.TIEMPO_ANIO, t.TIEMPO_CUATRIMESTRE, e.ESCUDERIA_NOMBRE,
avg(p.TIEMPO_BOX) as TIEMPO_PROMEDIO
    FROM GDD_EXPRESS.BI_Paradas p
      JOIN GDD_EXPRESS.BI_Tiempo t on t.TIEMPO_ID = p.TIEMPO_ID
      JOIN GDD_EXPRESS.BI_Escuderia e on e.ESCUDERIA_ID = p.ESCUDERIA_ID
      group by t.TIEMPO_CUATRIMESTRE, t.TIEMPO_ANIO, p.ESCUDERIA_ID,
e.ESCUDERIA_NOMBRE
```

5.6 Vista - Cantidad de paradas promedio por circuito

El nombre de la vista creada es **vw_cantidad_paradas_anio_x_circuito_escuderia**

Esta vista nos permite visualizar la cantidad de paradas por circuito por escudería por año.

Las columnas que agregamos para mostrar dicha información

TIEMPO_ANIO
ESCUDERIA_NOMBRE

CIRCUITO_NOMBRE
CANTIDAD_PARADAS

```
CREATE VIEW GDD_EXPRESS.vw_cantidad_paradas_anio_x_circuito_escuderia AS
    SELECT t.TIEMPO_ANIO, e.ESCUDERIA_NOMBRE, c.CIRCUITO_NOMBRE, count(*) as
CANTIDAD_PARADAS FROM GDD_EXPRESS.BI_Paradas p
      JOIN GDD_EXPRESS.BI_Tiempo t on t.TIEMPO_ID = p.TIEMPO_ID
      JOIN GDD_EXPRESS.BI_Escuderia e on e.ESCUDERIA_ID = p.ESCUDERIA_ID
      JOIN GDD_EXPRESS.BI_Circuito c on c.CIRCUITO_ID = p.CIRCUITO_ID
      group by t.TIEMPO_ANIO, p.CIRCUITO_ID, c.CIRCUITO_NOMBRE, p.ESCUDERIA_ID,
e.ESCUDERIA_NOMBRE
```

5.7 Vista - Circuitos con mayor tiempo de paradas

El nombre de la vista creada es **vw_top3_circuitos_mayor_tiempo_paradas**

Esta vista nos permite visualizar los 3 circuitos donde se consume mayor cantidad en tiempo de paradas en boxes.

Las columnas que agregamos para mostrar dicha información

CIRCUITO_NOMBRE
TIEMPO_BOX
TOTAL_TIEMPO

```
CREATE VIEW GDD_EXPRESS.vw_top3_circuitos_mayor_tiempo_paradas AS
    SELECT TOP 3 c.CIRCUITO_NOMBRE, sum(p.TIEMPO_BOX) as TOTAL_TIEMPO
    FROM GDD_EXPRESS.BI_Paradas p
      JOIN GDD_EXPRESS.BI_Circuito c on c.CIRCUITO_ID = p.CIRCUITO_ID
      group by p.CIRCUITO_ID, c.CIRCUITO_NOMBRE
      ORDER BY sum(p.TIEMPO_BOX) DESC
```

5.8 Vista - Circuitos más peligrosos

El nombre de la vista creada es **vw_top3_circuitos_mas_peligrosos**

Esta vista nos permite visualizar los 3 circuitos más peligrosos del año, en función mayor cantidad de incidentes

Las columnas que agregamos para mostrar dicha información

CIRCUITO_NOMBRE
TOTAL_INCIDENTES_CIRCUITO_ANIO
TIEMPO_ANIO

```
CREATE VIEW GDD_EXPRESS.vw_top3_circuitos_mas_peligrosos AS
      SELECT distinct TOP 3 c.CIRCUITO_NOMBRE, i.TOTAL_INCIDENTES_CIRCUITO_ANIO,
             t.TIEMPO_ANIO
      FROM GDD_EXPRESS.BI_Incidentes i
      JOIN GDD_EXPRESS.BI_Circuito c on i.CIRCUITO_ID = c.CIRCUITO_ID
      JOIN GDD_EXPRESS.BI_Tiempo t on t.TIEMPO_ID = i.TIEMPO_ID
      ORDER BY i.TOTAL_INCIDENTES_CIRCUITO_ANIO DESC
```

5.9 Vista - Promedio anual de incidentes

El nombre de la vista creada es **vw_promedio_anual_incidentes_x_escuderia**

Esta vista nos permite visualizar el promedio de incidentes que presenta cada escudería por año en los distintos tipos de sectores.

Las columnas que agregamos para mostrar dicha información

ESCUDERIA_NOMBRE
TIPO_SECTOR_DETALLE
PROMEDIO_ANUAL

```
CREATE VIEW GDD_EXPRESS.vw_promedio_anual_incidentes_x_escuderia AS
      SELECT e.ESCUDERIA_NOMBRE, ts.TIPO_SECTOR_DETALLE,
             avg(i.TOTAL_INCIDENTES_SECTOR_ANIO) AS PROMEDIO_ANUAL
      FROM GDD_EXPRESS.BI_Incidentes i
      JOIN GDD_EXPRESS.BI_Tiempo t on t.TIEMPO_ID = i.TIEMPO_ID
      JOIN GDD_EXPRESS.BI_Escuderia e on e.ESCUDERIA_ID = i.ESCUDERIA_ID
      JOIN GDD_EXPRESS.BI_Tipo_Sector ts on ts.TIPO_SECTOR_ID = i.TIPO_SECTOR_ID
      group by i.ESCUDERIA_ID, e.ESCUDERIA_NOMBRE, i.TIPO_SECTOR_ID,
ts.TIPO_SECTOR_DETALLE
```