ESTRUCTURA DE BASE DE DATOS PARA GRUPO CONCESIONARIO

Alumno: Fernando Brun Conesa

Comisión: #59425

Profesor: Nancy Elizabeth Villena Reines

Etapa del Proyecto: 1era Pre-entrega

Presentación de la compañía

Importante grupo concesionario necesita el desarrollo y estructura de una base de datos que permita optimizar informes sobre las reposiciones de ruedas a compañías de seguros. Requiere el tratado de información relevante sobre los siniestros; más específicamente sobre los vehículos más afectados y la segmentación de cada seguro a fin de poder mantener un inventario adecuado a las circunstancias.

Para tal fin, Trabajaremos con el objetivo de identificar los grupos de mayor relevancia para el proyecto, los mismos se separan en 5 segmentos:

- Siniestros: necesitamos una base de datos que permita registrar los casos según siniestros, cantidad de ruedas a reponer, compañías aseguradoras y vehículos involucrados.
- Seguros: una base de datos de compañías es crucial para este proyecto. La misma debe contener información precisa y completa de cada seguro y licitador asociado, ya que serán los principales clientes y requerirán diversas provisiones según el target de cada uno.
- Pólizas: requeriremos almacenar campos puntuales sobre los tipos de pólizas y detalles de coberturas ya que cada siniestro puede diferir su porcentaje dependiendo si es total o parcial. También precisaremos información de contacto de los asegurados.
- Vehículos: es importante considerar la descripción de marca, modelo y utilidad de cada vehículo, ya que las ruedas de equipamiento original varían según dichos campos y debemos asegurarnos de tener el conocimiento previo a la reposición.
- Facturas: mediante la facturación podremos mantener los registros de inventarios y cuentas, llevando así el control de stock relacionado directamente a los montos de las ruedas, actualización de precios y cuentas corrientes a las compañías de seguros.

Tablas

La elección del formato de entidades se centra en 2 tablas de hechos (siniestros y facturas) desde las cuales se comienzan a separar los datos potencialmente categóricos en nuevas tablas con la idea de optimizar el uso y facilitar las consultas. Cada tabla tiene en el nombre de la mayoría de sus atributos una referencia inicial al nombre de la tabla a la que pertenecen, de esta manera lograremos simplificar las consultas externas.

A continuación, se enumeran las tablas y se agrega una breve descripción:

- **SINIESTROS**
  - Tabla de hechos principal, contiene información de cada siniestro, fecha y cantidad de ruedas a reponer, así como referencias FK que conectan al resto de tablas dimensionales.
  - Atributos:
    - siniestro_id (PK)
    - fecha
    - siniestro_tipo (FK)
    - cantidad_ruedas
    - seguro_cia (FK)
    - poliza_nro (FK)
    - licitador (FK)
    - vehiculo(FK)
    - observaciones

- **TIPOS_SINIESTROS**
  - Describe el tipo de siniestro, si fuera rueda de posición, auxilio u otros detalles específicos del tipo de llanta que fuera equipo original.
  - Atributos:
    - siniestro_tipo_id (PK)
    - siniestro_tipo_descripcion

- **SEGUROS**
  - Posee información puntual sobre la compañía de seguro a la que pertenece el caso, datos de contacto y la ubicación matriz, así como también un atributo específico llamado ALIAS que simplifica el nombre ya que muchos seguros tienen razones sociales demasiado extensas, las cuales se describen en NOMBRE.
  - Atributos:
    - seguro_id (PK)

        - seguro_nombre
        - seguro_alias
        - seguro_ciudad (FK)
        - seguro_provincia (FK)
        - seguro_web
        - seguro_telefono
        - seguro_mail

- **CIUDADES**
    - Detalla ID y nombre de la ciudad donde se ubica la casa matriz o central.
    - Atributos:
        - ciudad_id (PK)
        - ciudad_nombre

- **PROVINCIAS**
    - Detalla ID y nombre de la provincia donde se ubica la casa matriz o central.
    - Atributos:
        - provincia_id (PK)
        - provincia_nombre

- **POLIZAS**
    - Informa datos exclusivos de las pólizas, tipo, porcentaje de cobertura y FK que conecta a tabla de asegurados.
    - Atributos:
        - poliza_id (PK)
        - poliza_tipo
        - cobertura
        - asegurado (FK)

- **ASEGURADOS**
    - Describe nombre y datos de contacto del/la titular de póliza.
    - Atributos:
        - asegurado_id (PK)
        - asegurado_nombre
        - asegurado_apellido
        - asegurado_telefono
        - asegurado_mail

- **LICITADORES**
    - Especifica el ente o compañía encargada de las licitaciones de los siniestros.
    - Atributos:

        - licitador_id (PK)
        - licitador_nombre
        - licitador_web

- **VEHICULOS**
    - Contiene información primordial de los vehículos afectados, al ser datos categóricos, dicha tabla es un puente o nexo que conecta a otras 3 que almacenan menor cantidad de información.
    - Atributos:
        - vehiculo_id (PK)
        - vehiculo_marca (FK)
        - vehiculo_modelo (FK)
        - vehiculo_utilidad (FK)

- **MARCAS_VEH**
    - Detalla ID y nombre de la marca fabricante del vehículo.
    - Atributos:
        - marca_id (PK)
        - marca_nombre

- **MODELOS**
    - Detalla ID y descripción del modelo de vehículo.
    - Atributos:
        - modelo_id (PK)
        - modelo_descripcion

- **UTILIDADES**
    - Detalla ID y descripción de la utilidad del vehículo.
    - Atributos:
        - utilidad_id (PK)
        - utilidad_descripcion

- **FACTURAS**
    - Tabla de hechos adicional, describe los datos de facturación y la numeración, así como también las ruedas entregadas.
    - Atributos:
        - factura_id (PK)
        - factura_tipo
        - factura_fecha
        - factura_pdv
        - factura_numero
        - rueda_item (FK)
        - rueda_precio
        - rueda_cantidad

        - factura_precio

- **FACTURAS_TIPOS**
    - Especifica el tipo de emisión de factura según monto y cuit del cliente.
    - Atributos:
        - factura_tipo_id (PK)
        - factura_tipo_descripción

- **RUEDAS**
    - Muestra una breve descripción de la rueda, llanta, rodado y marca de cubierta.
    - Atributos:
        - rueda_id (PK)
        - rueda_descripcion
        - cubierta_marca
        - rodado_llanta

- **MARCAS_CUB**
    - Describe solamente la marca de la cubierta, ya que las llantas son por defecto equipo original.
    - Atributos:
        - marca_id (PK)
        - marca_descripcion

- **LINK_FACTURAS_RUEDAS**
    - Para evitar una relación de muchos a muchos, se crea una tabla vínculo entre FACTURAS y RUEDAS.
    - Asignamos en constraint, que en caso de eliminar y/o modificar registros, sea modificación en cascada.
    - Atributos:
        - id_facturas (PK)
        - id_ruedas (PK)
        - cantidad
- **LOG**
    - Tabla creada para registrar las modificaciones DML (UPDATE, INSERT, DELETE) realizadas y los usuarios responsables.
    - Dicha tabla no requiere constraint y por ahora se asigna, mediante triggers, a la tabla 'siniestros' únicamente a modo de ejemplo.
    - Atributos:
        - id_log (PK)
        - tabla
        - id_pk
        - usuario

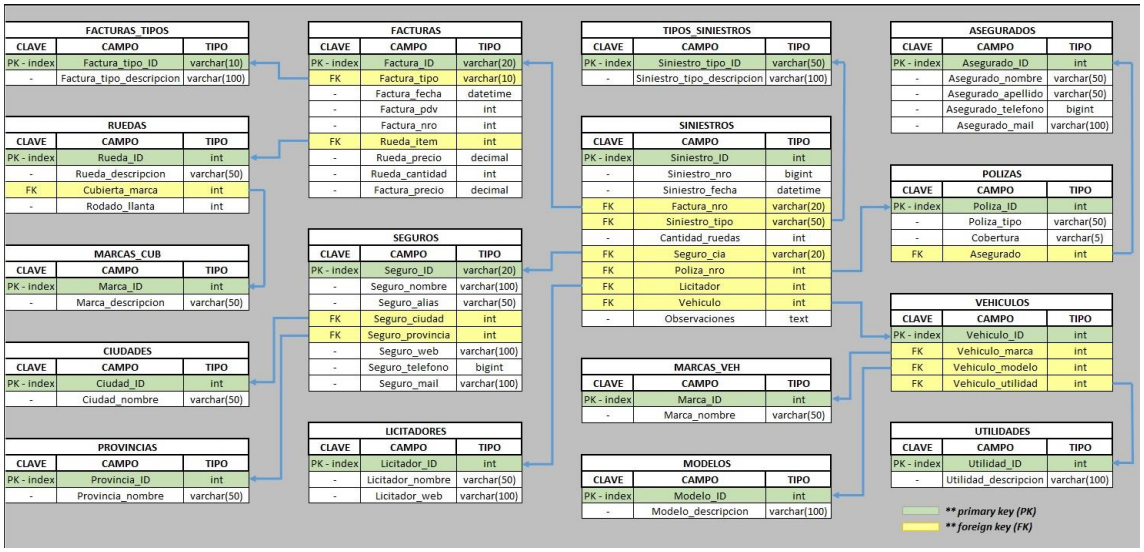
- fecha
- operación

Conexión de las Tablas

Los 'CONSTRAINTS' o restricciones entre llaves foráneas (FK), se construyen de manera tal que no puedan eliminarse registros de una tabla cuyos valores exista en otra relacionada, pero se sentencia que sí se puedan modificar dichas FK y la modificación sea en 'cascada', de manera tal que se actualizará en el resto de tablas comprometidas.

En la siguiente imagen de puede ver las estructuras de tablas, la definición de PK y FK, la conexión entre las mismas y los tipos de valores designados en cada campo.

Se organiza de manera calculada, la tabla de hechos principal (SINIESTROS) al centro y el resto alrededor, las vinculaciones entre tablas externas se pueden notar fácilmente ya que no se cruza ninguna flecha, todo ello a fin de que sea visualmente prolija y comprensible.



FACTURAS_TIPOS

CLAVE	CAMPO	TIPO
PK - index	Factura_tipo_ID	varchar(10)
-	Factura_tipo_descripcion	varchar(100)

FACTURAS

CLAVE	CAMPO	TIPO
PK - index	Factura_ID	varchar(20)
FK	Factura_tipo	varchar(10)
-	Factura_fecha	datetime
-	Factura_pdv	int
-	Factura_nro	int
FK	Rueda_item	int
-	Rueda_precio	decimal
-	Rueda_cantidad	int
-	Factura_precio	decimal

TIPOS_SINIESTROS

CLAVE	CAMPO	TIPO
PK - index	Siniestro_tipo_ID	varchar(50)
-	Siniestro_tipo_descripcion	varchar(100)

ASEGURADOS

CLAVE	CAMPO	TIPO
PK - index	Asegurado_ID	int
-	Asegurado_nombre	varchar(50)
-	Asegurado_apellido	varchar(50)
-	Asegurado_telefono	bigint
-	Asegurado_mail	varchar(100)

RUEDAS

CLAVE	CAMPO	TIPO
PK - index	Rueda_ID	int
-	Rueda_descripcion	varchar(50)
FK	Cubierta_marca	int
-	Rodado_llanta	int

SINIESTROS

CLAVE	CAMPO	TIPO
PK - index	Siniestro_ID	int
-	Siniestro_nro	bigint
-	Siniestro_fecha	datetime
FK	Factura_nro	varchar(20)
FK	Siniestro_tipo	varchar(50)
-	Cantidad_ruedas	int
FK	Seguro_cia	varchar(20)
FK	Poliza_nro	int
FK	Licitador	int
FK	Vehiculo	int
-	Observaciones	text

POLIZAS

CLAVE	CAMPO	TIPO
PK - index	Poliza_ID	int
-	Poliza_tipo	varchar(50)
-	Cobertura	varchar(5)
FK	Asegurado	int

MARCAS_CUB

CLAVE	CAMPO	TIPO
PK - index	Marca_ID	int
-	Marca_descripcion	varchar(50)

SEGUROS

CLAVE	CAMPO	TIPO
PK - index	Seguro_ID	varchar(20)
-	Seguro_nombre	varchar(100)
-	Seguro_alias	varchar(50)
FK	Seguro_ciudad	int
FK	Seguro_provincia	int
-	Seguro_web	varchar(100)
-	Seguro_telefono	bigint
-	Seguro_mail	varchar(100)

MARCAS_VEH

CLAVE	CAMPO	TIPO
PK - index	Marca_ID	int
-	Marca_nombre	varchar(50)

VEHICULOS

CLAVE	CAMPO	TIPO
PK - index	Vehiculo_ID	int
FK	Vehiculo_marca	int
FK	Vehiculo_modelo	int
FK	Vehiculo_utilidad	int

CIUDADES

CLAVE	CAMPO	TIPO
PK - index	Ciudad_ID	int
-	Ciudad_nombre	varchar(50)

LICITADORES

CLAVE	CAMPO	TIPO
PK - index	Licitador_ID	int
-	Licitador_nombre	varchar(50)
-	Licitador_web	varchar(100)

UTILIDADES

CLAVE	CAMPO	TIPO
PK - index	Utilidad_ID	int
-	Utilidad_descripcion	varchar(100)

PROVINCIAS

CLAVE	CAMPO	TIPO
PK - index	Provincia_ID	int
-	Provincia_nombre	varchar(50)

MODELOS

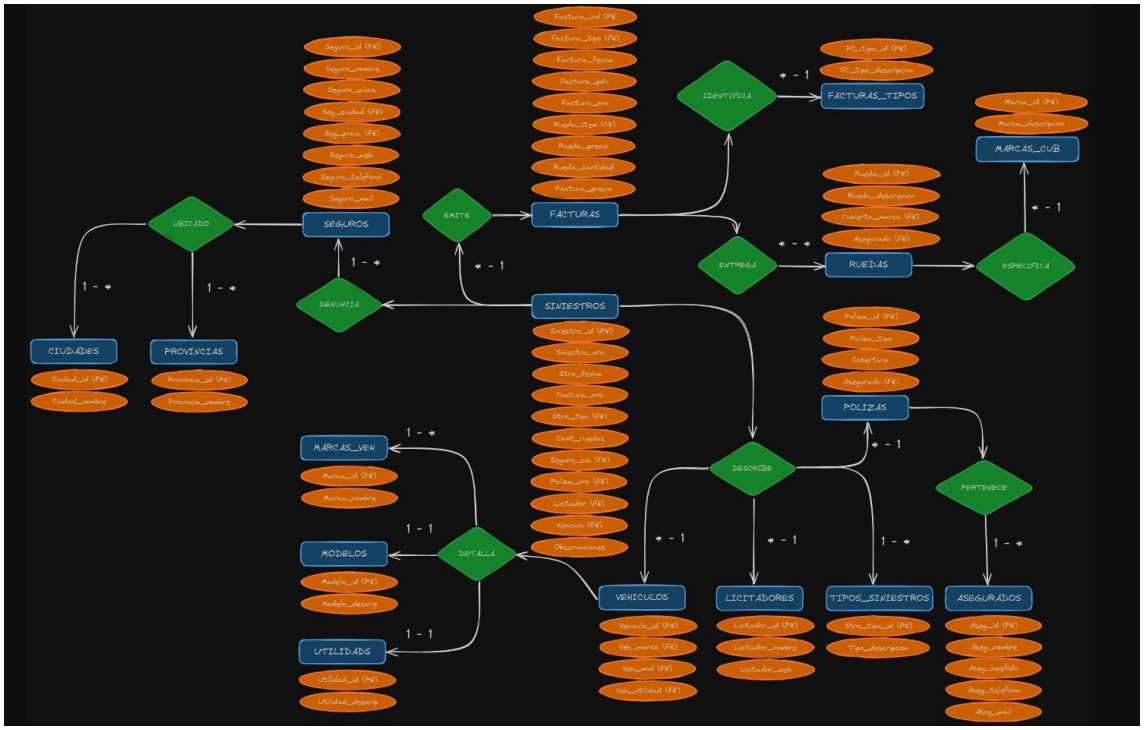
CLAVE	CAMPO	TIPO
PK - index	Modelo_ID	int
-	Modelo_descripcion	varchar(100)

*** primary key (PK)*

*** foreign key (FK)*

Diagrama de Entidad de Relación (DER)

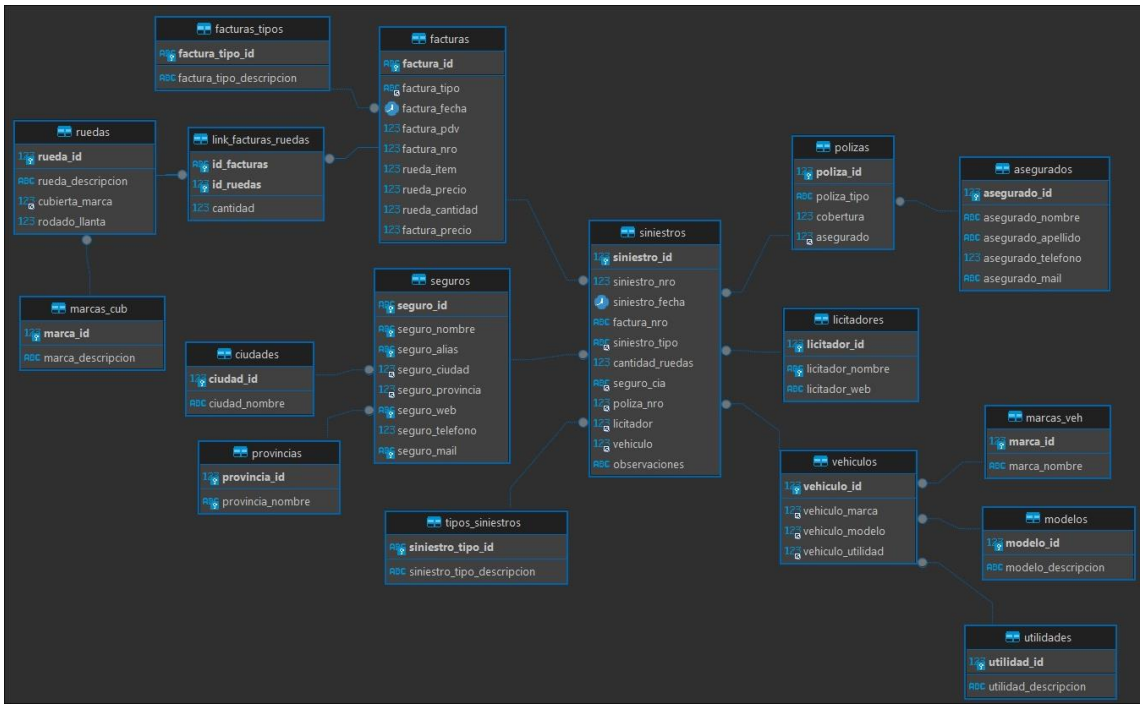
El diagrama muestra en rombos la relación que conecta las entidades y el tipo de conexión.



VALIDACIÓN

Print del DER resultante una vez creada la base de datos en SQL.

En el mismo se puede ver la conexión a la tabla vínculo que no figura en las imágenes anteriores.


facturas_tipos
factura_tipo_id
factura_tipo_descripcion
facturas
factura_id
factura_tipo
factura_fecha
factura_pdv
factura_nro
rueda_item
rueda_precio
rueda_cantidad
factura_precio
ruedas
rueda_id
rueda_descripcion
cubierta_marca
rodado_llanta
link_facturas_ruedas
id_facturas
id_ruedas
cantidad
polizas
poliza_id
poliza_tipo
cobertura
asegurado
asegurados
asegurado_id
asegurado_nombre
asegurado_apellido
asegurado_telefono
asegurado_mail
siniestros
siniestro_id
siniestro_nro
siniestro_fecha
factura_nro
siniestro_tipo
cantidad_ruedas
seguro_cia
poliza_nro
licitador
vehiculo
observaciones
seguros
seguro_id
seguro_nombre
seguro_alias
seguro_ciudad
seguro_provincia
seguro_web
seguro_telefono
seguro_mail
marcas_cub
marca_id
marca_descripcion
ciudades
ciudad_id
ciudad_nombre
provincias
provincia_id
provincia_nombre
licitadores
licitador_id
licitador_nombre
licitador_web
marcas_veh
marca_id
marca_nombre
vehiculos
vehiculo_id
vehiculo_marca
vehiculo_modelo
vehiculo_utilidad
modelos
modelo_id
modelo_descripcion
tipos_siniestros
siniestro_tipo_id
siniestro_tipo_descripcion
utilidades
utilidad_id
utilidad_descripcion

Importación de Datos

Para la correcta importación de datos desde archivos planos, es necesario previamente habilitar los permisos en servidor y cliente, así como también la edición del archivo 'my.ini' o 'my.cnf', insertando el comando local_infile=1 debajo de una terminación específica:

```
[mysqld]
local_infile=1
```

Los permisos fueron activados mediante 2 comandos adicionales:

```
SET GLOBAL local_infile = TRUE;
(en el script)
```

```
local-infile=1
(compando posterior a los datos de inicio en bash -terminal-)
ejemplo de comando completo
mysql -u root -p --host localhost --port 3306 --local-infile=1
```

También fue necesario desactivar temporalmente la verificación de claves foráneas durante la importación, reactivándola posteriormente. Ejemplo de estructura:

```
SET foreign_key_checks = 0;
...
(importación de datos)
...
SET foreign_key_checks = 1;
```

Se ejecutan las importaciones locales ejecutando un comando por cada tabla y en el orden de relevancia según los constraints. Ejemplo de formato habitual de los comandos:

```
LOAD DATA LOCAL INFILE 'ruta/al/archivo/nombre_archivo.csv'
INTO TABLE nombre_tabla
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\r\n'
IGNORE 1 LINES;
```

En los casos de las tablas de hechos (*siniestros* / *facturas*) fué necesario setear los formatos de fecha y posibles existencias de nulos, a fin de evitar inconvenientes con la terminal. Ejemplo del comando modificado:

```
LOAD DATA LOCAL INFILE '/sql_project/data_csv/siniestros.csv'
INTO TABLE siniestros
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\r\n'
IGNORE 1 ROWS
(siniestro_id, siniestro_nro, @siniestro_fecha, factura_nro,
siniestro_tipo, cantidad_ruedas, seguro_cia, poliza_nro,
licitador, vehiculo, @observaciones)
SET siniestro_fecha = STR_TO_DATE(@siniestro_fecha, '%Y-%m-%d'),
      observaciones = NULLIF(@observaciones, '');
```

Objetos de la Base de Datos

A continuación, se detalla una lista con breve descripción de cada objeto creado para la optimización y agilización de la base de datos.

VISTAS

1. **VIEW_TAXES**

   - Vista para determinar el IVA (21%) e IIBB (3%) mensual según monto y provincia.

   - Columnas:
     - Mes
     - IVA 21%
     - IIBB
     - Provincia

   - Ejemplos de uso:

     ```
     SELECT * FROM repositor_ruedas.view_taxes
       ORDER BY mes ASC;
     ```

2. **VIEW_RUEDAS**

   - Vista para determinar el movimiento de stock según cantidad, rodado de llanta y marca de cubierta.

   - Columnas:
     - Cantidad
     - Llanta
     - Marca_cubierta

   - Ejemplos de uso:

     ```
     SELECT * FROM repositor_ruedas.view_ruedas
       ORDER BY Cantidad DESC;
     ```

3. **VIEW_REINCIDENCIAS**

   - Vista para las compañías, ayudará al control de reincidencias por alta siniestralidad.

   - Columnas:
     - Poliza

    - Reincidencias
    - Asegurado

- Ejemplos de uso:

```
SELECT *
  FROM repositor_ruedas.view_reincidencias
  ORDER BY reincidencias DESC
  LIMIT 20;
```

4. **VIEW_SINIESTROS_VEHICULOS**
    - Vista para control de reposiciones, ayudará a determinar las compras a concecionarias oficiales considerando marca y modelo de vehículos.
    - Columnas:
        - Suma_siniestros
        - Modelo
        - Marca
        - Cant_ruedas

    - Ejemplos de uso:

```
SELECT *
  FROM repositor_ruedas.view_siniestros_vehiculos;
  ORDER BY cant_ruedas DESC
  LIMIT 20;
```

5. **VIEW_CIA_PROM**

    - Vista para llevar control del promedio de órdenes que asigna cada seguro. A fines prácticos, considera solamente el último mes de participación.

    - Columnas:
        - Compania
        - Promedio_orden
        - Ultimo_mes

    - Ejemplos de uso:

```
SELECT * FROM repositor_ruedas.view_cia_prom;
```

6. **VIEW_VEHICULOS**

   - Vista para consultar las descripciones de vehículos registrados.

   - Implica joins en las 4 tablas de vehículos involucradas.

   - Columnas:
     - Marca
     - Modelo
     - Utilidad

   - Ejemplos de uso:

     SELECT * FROM repositor_ruedas.view_vehiculos;

FUNCIONES

1. **GANANCIA_NETA**

   - Calcula la ganancia neta, restando -21% (iva) y -3% (IIBB) al valor de las facturas. Determina la ganancia con un cálculo simple del precio *0,79 *0,03 y concatena el resultado con un símbolo $.

   - Ejemplo de uso:

     ```
     SELECT
       factura_id AS Factura,
       factura_precio AS Precio,
       repositor_ruedas.ganancia_neta(factura_precio) AS
     Ganancia_neta
       FROM
       facturas
       LIMIT 20;
     ```

2. **CANT_X_CIA**

   - Toma como parámetro el alias de las compañías para ejecutar la función, la cual calcula de forma simple el total de ruedas entregadas a cada seguro.

- Ejemplo de uso (unitario simple y en lista, por período):

```
-- simple
SELECT repositor_ruedas.cant_x_cia('ALLIANZ') AS Total_ruedas;

-- por período
SELECT
  se.seguro_alias,
  SUM(si.cantidad_ruedas) AS Total_ruedas
FROM siniestros AS si
JOIN seguros AS se
        ON si.seguro_cia = se.seguro_id
WHERE
  si.siniestro_fecha BETWEEN '2024-06-01' AND '2024-06-30'
GROUP BY se.seguro_alias
ORDER BY Total_ruedas DESC;
```

3. **PORCENT_LICITADOR**

- Determina el índice de participación de cada licitador. Considera el total de los siniestros, cuenta la cantidad de veces que aparece cada ente y transforma el valor a porcentual (cuenta / total * 100), modificando también el resultado a decimal y concatenando un símbolo % al final.

- Ejemplo de uso:

```
SELECT
  licitador_nombre AS Licitador,
  repositor_ruedas.porcent_licitador(licitador_nombre) AS
Participación
FROM
  licitadores
ORDER BY
        Participación DESC;
```

PROCEDIMIENTOS

1. **INGRESO_SINIESTRO**

- Éste procedimiento es TRANSACCIONAL (TCL).
- Posibilita el ingreso de nuevos registros en la tabla 'siniestros', determinando validaciones con mensajes SQLSTATE '45000' para 5 de los 11 atributos.
- Al ingresar un siniestro, se crea un savepoint al cual se redirigirá un rollback en caso de que no se completen las validaciones posteriores.
- Dichas validaciones, corrobora que los datos ingresados en campos FK sean existentes en sus respectivas tablas relacionadas.
- Se determinan los campos a completar y finaliza con una query simple que muestra el último registro.
- La idea es simplificar el proceso y posibilitar un nulo dentro del campo de nro de factura, ya que es FK pero no siempre se factura al mismo momento o en la misma fecha.
- Agiliza el ingreso de registros, ya que 3 de los 11 atributos son valores por defecto:
    - siniestro_id = AUTO_INCREMENT
    - Siniestro_fecha = CURRENT_TIMESTAMP()
    - factura_nro = 'Pendiente'
- Ejemplo de uso:

```
CALL ingreso_siniestro(
  2003506792,            -- siniestro_nro
  'POCH',                -- siniestro_tipo
  4,                     -- cantidad_ruedas
  '30-50004946-0',       -- seguro_cia
  167559,                -- poliza_nro
  2,                     -- licitador
  33,                    -- vehiculo
  NULL                   -- observaciones
  );
```

2. AGREGAR_FACTURA

- Éste procedimiento es TRANSACCIONAL (TCL).
- Optimiza el ingreso de una nueva factura, ya que no sólo permite la inserción en la tabla 'facturas', sino que además considera el nro de siniestro al que corresponde, previamente cargado en su respectiva tabla y actualiza el campo de la tabla 'siniestros', es decir, pasa de estar en FC 'Pendiente' a llevar el nro de FC que estamos asignando.

- Inicialmente valida que el siniestro exista y que tenga estado de 'Pendiente' en FC, caso contrario devuelve un mensaje SQLSTATE '45000' como 'Siniestro inexistente'.
- En caso de que exista, se crea un savepoint al cual se redirigirá un rollback en caso de que no se completen los campos pertinentes de la tabla 'facturas'.
- Agiliza el proceso ya que se automatizan varios atributos:
    - El valor en factura_id (PK) es una concatenación entre factura_tipo, factura_pdv y factura_nro.
    - El valor en factura_precio es un cálculo simple que multiplica rueda_precio * rueda_cantidad.
    - El valor en factura_fecha es por defecto CURRENT_TIMESTAMP().
    - El valor en rueda_cantidad lo asigna según el valor correspondiente en cantidad_ruedas de la tabla siniestros.
- Lo más importante es que además de actualizar el nro de FC en la tabla 'siniestros', actualiza los registros completos en la tabla link que existe entre ambas.
- Por último, también devuelve una query simple que muestra la carga exitosa del registro.
- Ejemplo de uso:

```
CALL agregar_factura(
  1253,     -- siniestro_id
  'FA',     -- factura_tipo
  3,        -- factura_pdv
  69050,    -- factura_nro
  60,       -- rueda_item
  220000,   -- rueda_precio
  );
```

3. AGREGAR_VEHICULO

- Optimiza el ingreso de un nuevo vehículo, ya que con sólo 3 datos se actualizan 4 tablas:
    - Vehículos
    - marcas_veh
    - modelos
    - utilidades
- Si ya existe una marca/modelo/utilidad con ese nombre, se actualiza el registro existente y se obtiene el ID del campo

actualizado utilizando
LAST_INSERT_ID(marca_id/modelo_id/utilidad_id).

- Finalmente se retornan los ID correspondientes en las 4 tablas
- Ejemplo de uso:

```
CALL agregar_vehiculo(
  'Audi',                -- Marca
  'A6',          -- Modelo
  'Particular'  -- Utilidad
  );
```

TRIGGERS

1. VALIDACION_WEB

- Creado sobre tabla 'seguros' para asegurar conexión segura a los portales webs donde se gestionan los siniestros y facturas de cada compañía.
- Ejemplo de uso y mensaje SIGNAL SQLSTATE '45000':

```
INSERT INTO seguros
(seguro_id, seguro_nombre, seguro_alias, seguro_ciudad,
seguro_provincia, seguro_web, seguro_telefono, seguro_mail)
VALUES
('33-12345678-1',
'Answer Cia de Seguros S.A.',
'ANSWER',
4,
1,
'www.answer.com',  -- VALOR ERRÓNEO
1154817808,
'siniestros@answer.com'
);
```

ERROR CODE 1644 (45000): Corroborar falta de 'https://', web podría no ser segura

2. CANT_X_SINIESTRO

- Creado sobre tabla 'siniestros' para evitar errores de tipeo, en éste caso, la cantidad de ruedas máxima de ruedas que pueda poseer

cualquier vehículo del segmento trabajado, el cual no incluye transportes o taras más grandes.

- Ejemplo de uso y mensaje SIGNAL SQLSTATE '45000':

```
CALL ingreso_siniestro(
  2554738,              -- siniestro_nro
  NULL,                 -- siniestro_fecha (default CURRENT)
  'AUPOAL',             -- siniestro_tipo
  6,                    -- VALOR ERRÓNEO
  '30-50004717-4',      -- seguro_cia
  169601,               -- poliza_nro
  2,                    -- licitador
  11,                   -- vehiculo
  NULL                  -- observaciones
  );
```

ERROR CODE (45000): La cantidad de ruedas no puede superar las 5 unidades

3. ASEGURADO_TEL

- Creado sobre tabla 'asegurados' con devolución de mensaje de advertencia o warning en caso que no se haya asignado un número de teléfono de contacto.
- Ejemplo de uso y mensaje SIGNAL SQLSTATE '01000':

```
INSERT INTO asegurados
        (asegurado_id, asegurado_nombre, asegurado_apellido)
VALUES
        (1260, 'Rosario', 'Pileyra');
```

WARNING CODE (1000): Recuerde registrar un contacto telefónico

TRIGGERS DML

Para la siguiente utilidad, creamos una serie de 3 triggers los cuales van a registrar ejecuciones DML sobre la tabla 'siniestros' y se guardarán en la tabla 'LOG'.

La cantidad de 3 es porque SQL solamente permite a un trigger ejecutarse sobre una sola acción DML, por lo cual será 1 trigger para cada acción (INSERT, UPDATE, DELETE).

Se puede aplicar la misma modalidad a todas las tablas, pero requiere 3 triggers adicionales por cada una.

Se omiten los DELIMITER // y DELIMITER ; ya que no son corridos correctamente en bash, pero al enviar el script directamente al servidor MySQL puede interpretar correctamente la estructura del trigger sin necesidad de cambiar el delimitador.

4. **SINIESTROS_INSERT_LOG**

- Registra las acciones de inserción de datos en tabla siniestros.
- Ejemplo de uso:

```
CALL ingreso_siniestro(
  2331984,          -- siniestro_nro
  NULL,             -- siniestro_fecha (DEFAULT
CURRENT_TIMESTAMP)
  'AUCH',           -- siniestro_tipo
  3,                -- cantidad_ruedas
  '30-50001770-4',  -- seguro_cia
  8902726,          -- poliza_nro
  1,                -- licitador
  18,               -- vehiculo
  NULL              -- observaciones (sin datos)
  );
```

5. **SINIESTROS_UPDATE_LOG**

- Registra las acciones de modificación de datos en tabla siniestros.
- Ejemplo de uso:

```
UPDATE siniestros
SET cantidad_ruedas = 2
WHERE siniestro_id = 1262; -- ver el id asignado por el CALL
```

6. **SINIESTROS_DELETE_LOG**

- Registra las acciones de eliminación de datos en tabla siniestros.
- Ejemplo de uso:

```
DELETE FROM siniestros
WHERE siniestro_id = 1262; -- ver el id asignado por el CALL
```

Finalmente, al ejecutar una query de consulta simple sobre la tabla 'LOG', devolverá los 3 DML ejecutados, con sus respectivos atibutos:

SELECT * FROM log;

ROLES Y USUARIOS

Para ésta sección se determina la creación de 4 roles que representan las áreas encargadas:

1. **SISTEMA**

    - Sistema tiene TODOS los permisos sobre la base de datos.
    - Se encargará de todas las gestiones sobre estructura y códigos, corrección, depuración y actualización.
    - Posee conexión desde cualquier host ('%').
    - Contraseña: intentos fallidos = 2 / bloqueo de cuenta = 2 días.
    - Usuarios:
        - ‘LeoDI'@'%'
        - 'JesiB'@'%'

2. **ADMIN**

    - Administración posee todos los permisos para DML sobre 4 tablas específicas:
        - siniestros
        - tipos_siniestros
        - facturas
        - facturas_tipos

    - También podrán ejecutar 2 funciones y 2 procedimientos:
        - funcion - ganancia_neta
        - funcion - porcent_licitador
        - proced - ingreso_siniestro
        - proced - agregar_factura

    - Serán los encargados de ingresos de registros sobre las tablas de hechos, así como actualizaciones y correcciones.
    - Acceso restringido a conexión local ('localhost').
    - Contraseña: intentos fallidos = 2 / bloqueo de cuenta = 5 días.
    - Usuarios:
        - 'AndreC'@'localhost'

    - 'FedeZ'@'localhost'
    - 'HugoQ'@'localhost'

3. DEPOSITO

- Depósito posee todos los permisos para DML sobre 6 tablas específicas:
    - ruedas
    - marcas_cub
    - vehiculos
    - marcas_veh
    - modelos
    - utilidades

- También podrán ejecutar 1 función y 1 procedimiento:
    - funcion - cant_x_cia
    - proced - agregar_vehiculo

- Su rol principal será mantener actualizados los registros que tengan relación a vehículos y ruedas, con el fin de mantener un stock acorde al historial que brinda la base de datos.
- Acceso restringido a conexión local ('localhost').
- Contraseña: intentos fallidos = 2 / bloqueo de cuenta = 5 días.
- Usuarios:
    - 'CrisA'@'localhost'
    - 'ReneB'@'localhost'
    - 'SantiG'@'localhost'
    - 'MatiK'@'localhost'

4. CONTACTO

- Dicho sector será el encargado de manter el contacto tanto con personas, como con entidades.
- Solamente podrá visualizar 4 tablas:
    - seguros
    - licitadores
    - asegurados
    - polizas

- También tendrá acceso a 1 vista:
    - view_reincidencias

- No podrán ejecutar DML, se restringe su acceso a consultas de información de las respectivas tablas.
- Acceso restringido a conexión local ('localhost').
- Contraseña: intentos fallidos = 2 / bloqueo de cuenta = 5 días.
- Usuarios:
    - ‘RubenM'@'localhost'
    - 'LucasN'@'localhost'

Una vez creados los roles, usarios y asignaciones, se activan los roles y se actualizan los privilegios con los siguientes respectivos comandos:

SET DEFAULT ROLE '{ROL}' TO '{USER1}'@'%', '{USER2}@'%';
(1 código por cada rol)

FLUSH PRIVILEGES;

Para cambiar de usuario e iniciar sesión con un usuario específico, los comandos en terminal pueden variar dependiendo si es host local (localhost) o si fuera una conexión remota (%), como ser los usuarios del rol 'SISTEMA' y dichos comandos serían los siguientes:

- Comando 'localhost':

mysql -u {nombre_usuario} -p --host 127.0.0.1 --port 3306

- Comando con host remoto:

mysql -u {nombre_usuario} -p -h {nombre_host} --port 3306

BACKUP | DUMP

Se ejecuta la modalidad 'Export to Self-contained File', que devuelve 1 sólo script completo.
El archivo backup se pueden corroborar en la carpeta backup_dump en el repositorio.
También se puede ejecutar el siguiente paso a paso en terminal:

mysqldump --version

- (corrobora la versión mysql)

mysqldump --help

- (muestra la ayuda y opciones disponibles para el comando dump)

mysqldump -u root -p --host 127.0.0.1 --port 3306 --databases repositor_ruedas > bk.repositor.sql

- (conexión y nombre de archivo dump a exportar)

nvim bk.repositor.sql

- (abre el archivo bk.repositor.sql en el editor de texto nvim=Neovim)

mysql -u root -p --host 127.0.0.1 --port 3306 --databases repositor_ruedas < bk.repositor.sql

- (conexión y nombre de archivo a importar/restaurar)

mysql -u root -p --host 127.0.0.1 --port 3306 --e "SHOW DATABASES"

- (--e ejecuta comandos mysql, en éste caso muestra bases de datos)

CÓMO CORRER EL CÓDIGO

make

Ingresar en la sección codespaces y en la terminal, utilizar los comandos:

- make si te da un error de que no conexion al socket, volver al correr el comando make
- make test-db para mirar los datos de cada tabla
- make access-db para acceder a la base de datos
- make backup-db para realizar un backup de mi base de datos
- make clean-db limpiar la base de datos

ACLARACIÓN:

Quizás haya que ejecutar make varias veces para que importe los datos en las tablas de hecho 'siniestros' y 'facturas'.
En mi caso, más de la mitad de las veces no los importa y a veces importa en una tabla si y la otra no.

Repositorio Github con Script de Creación de Tablas:

https://github.com/Ferbrunc89/Script-Brun-Conesa-V2