Informe Base de Datos 2

Integrantes:

Ignacio Capra
Emiliano Gonzalez
Rodrigo Lame
Joaquin Sosa
Alejandro Rodriguez

Para la tarea de BASE DE DATOS 2 realizamos un único programa llamado (JavaApplication3.java), que cuando este es ejecutado se puede ver un Menú con 5 opciones (es una función llamada menuGeneral();):

1- Conectarse a la base de datos manejadorbd con el usuario “ALEJANDRO”;
2- Al seleccionar este punto ingresamos en un Menú de la parte 4 (Mejor explicado adelante en dicho punto).
3- En este punto se realiza la Parte 6 de la tarea.
4- En este punto se realiza la Parte 8 de la tarea
0- Salir, osea finaliza dicha ejecución.

Parte 1) Instalamos el DBMS relacional "PostgreSQL", la versión 11.3. Luego instalamos el NetBeans para poder codificar los siguientes puntos

Parte II) Primero nos conectamos con el SUPERUSER postgres para crear los usuarios para la base de datos local, ahí creamos los usuarios con el siguiente comando:

- CREATE USER (nombre del usuario) PASSWORD ‘123456’, usamos la misma para todos los usuarios.

Después de haber creado los usuarios necesarios, le agregamos los permisos solicitados para cada usuario.

ALEJANDRO (DBA):

- ALTER USER alejandro SUPERUSER;
- ALTER USER alejandro CREATEDB;
- ALTER USER alejandro CREATEROLE;

EMILIANO Y IGNACIO (CREATE):

- ALTER USER emiliano CREATEDB;
- ALTER USER emiliano CREATEROLE;
- ALTER USER ignacio CREATEDB;
- ALTER USER ignacio CREATEROLE;

JOAQUÍN, JUAN Y PEDRO (SELECT):

- GRANT SELECT ON ALL TABLES IN SCHEMA PUBLIC TO joaquin;
- GRANT SELECT ON ALL TABLES IN SCHEMA PUBLIC TO juan;
- GRANT SELECT ON ALL TABLES IN SCHEMA PUBLIC TO pedro;

RODRIGO Y SERGIO (INSERT, UPDATE Y DELETE):

- GRANT INSERT, UPDATE, DELETE ON ALL TABLES IN SCHEMA PUBLIC TO rodrigo;
- GRANT INSERT, UPDATE, DELETE ON ALL TABLES IN SCHEMA PUBLIC TO sergio;
- GRANT INSERT, UPDATE, DELETE ON aerolinea_idaerolinea_seq, ciudad_idciudad_seq, cliente_idc_seq, empleado_ide_seq, formadepago_idpago_seq, viajevendido_idviajevendido_seq, vuelo_idvuelo_seq TO rodrigo;
- GRANT INSERT, UPDATE, DELETE ON aerolinea_idaerolinea_seq, ciudad_idciudad_seq, cliente_idc_seq, empleado_ide_seq, formadepago_idpago_seq, viajevendido_idviajevendido_seq, vuelo_idvuelo_seq TO sergio;

Parte III) Agregamos las tablas, con los atributos que consideramos necesarios para realizar dicha tarea:

```
CREATE TABLE Empleado(
        idE SERIAL PRIMARY KEY,
        ciE varchar unique NOT NULL,
        nombre varchar(15) NOT NULL,
        apellido varchar(25) NOT NULL,
        fechaNac date NOT NULL
);

CREATE TABLE Cliente(
        idC SERIAL PRIMARY KEY,
        ciC varchar unique NOT NULL,
        nombre varchar(15) NOT NULL,
        apellido varchar(25) NOT NULL,
        fechaNac date NOT NULL
);

CREATE TABLE Aerolinea(
        idAerolinea SERIAL,
        nombre varchar(25) unique NOT NULL,
        PRIMARY KEY(idAerolinea,nombre)
);

CREATE TABLE Ciudad(
        idCiudad SERIAL,
        nombre varchar(25) unique NOT NULL,
```

```
        PRIMARY KEY(idCiudad)
);

CREATE TABLE Vuelo(
        idVuelo SERIAL,
        idVueloOrigen int,
        idVueloDestino int,
        origen varchar(45) NOT NULL,
        destino varchar(45) NOT NULL,
        horario_partida time NOT NULL,
        horario_llegada time NOT NULL,
        precio real NOT NULL,
        PRIMARY KEY (idVuelo,idVueloOrigen,idVueloDestino),
        FOREIGN KEY (origen) references Ciudad(nombre),
        FOREIGN KEY (destino) references Ciudad(nombre),
        FOREIGN KEY (idVueloOrigen) references Ciudad(idCiudad),
        FOREIGN KEY (idVueloDestino) references Ciudad(idCiudad)
);


CREATE TABLE FormadePago(
        idPago SERIAL PRIMARY KEY,
        ciCliente varchar NOT NULL,
        debito boolean NOT NULL,
        credito boolean NOT NULL,
        FOREIGN KEY (ciCliente) references Cliente(ciC)
);
```

```
CREATE TABLE ViajeVendido(
        idViajeVendido SERIAL,
        idVueloOrigen int,
        idVueloDestino int,
        origenVuelo varchar(45) NOT NULL,
        destinoVuelo varchar(45) NOT NULL,
        horario_partida time NOT NULL,
        horario_llegada time NOT NULL,
        ciEmpleado varchar(8) NOT NULL,
        ciCliente varchar(8) NOT NULL,
        escala boolean NOT NULL,
        montoVenta real NOT NULL,
        fechaCompra date NOT NULL,
        nombreAero varchar(25) NOT NULL,
        cantEscala int NOT NULL,
        PRIMARY KEY (idViajeVendido,idVueloOrigen,idVueloDestino),
        FOREIGN KEY (origenVuelo) references Ciudad(nombre),
        FOREIGN KEY (destinoVuelo) references Ciudad(nombre),
        FOREIGN KEY (ciEmpleado) references Empleado(ciE),
        FOREIGN KEY (nombreAero) references Aerolinea(nombre),
        FOREIGN KEY (ciCliente) references Cliente (ciC),
        FOREIGN KEY (idVueloOrigen) references Ciudad(idCiudad),
        FOREIGN KEY (idVueloDestino) references Ciudad(idCiudad)
);

CREATE TABLE VueloVV (
        idViajeVen int,
        idV int,
        PRIMARY KEY(idV,idViajeVen),
        FOREIGN KEY (idV) references Vuelo(idVuelo),
        FOREIGN KEY (idViajeVen) references ViajeVendido(idViajeVendido)
);
```

A todas estas tablas les agregamos valores hechos por nosotros mismos que se encuentran en el archivo adjunto llamado “Datos Tablas BD2.sql”

Parte IV) En este punto se llama una función (menuPARTE4();), que este contiene 13 opciones diferentes:

1. (Parte A)
2. (Parte B)
3. (Parte C)
4. (Parte D)
5. (Parte E)
6. (Parte F)

7. (Parte G)
8. (Parte H)
9. (Parte I)
10. (Parte J)
11. (Parte K)
12. (Parte L)
0. Salir, o sea volver al menuGeneral();.

Cada opción del menú anterior se detalla en las correspondientes opciones más adelante.

Parte A) Se puede encontrar en el código fuente, en una función llamada "conectarCrearBase()"; En esta función me conecto al SuperUsuario postgres, y creo la base de datos "manejadorbd", la que vamos a usar a lo largo de la tarea.

Parte B) Se puede encontrar en el código fuente, en una función llamada "createTable();" que al ejecutar dicha función crea las 7 tablas diseñadas en PARTE III de la tarea.

Parte C) Se puede encontrar en el código fuente, en esta parte cramos 4 funciones, una para cada sentencia solicitada, "insert();", "update();", "delete()"; y "select()"; lo que hicimos fue diseñar un Menú (hicimos una función llamada "menu3"), para poder elegir que parte ejecutamos primero y para poder ver los cambios, así podemos insertar datos("insert();"), ver los datos insertados("select();"), actualizar los datos("update();"), después de haber actualizado los datos los podemos ver de nuevo si seleccionamos la opción select en el menú, y también podemos borrar los datos con la función (delete();). Estas sentencias se invocan sobre las tablas Ciudad, Cliente, Aerolinea y Empleado. Antes de probar esta parte, es necesaria llamar a la parte H.

Parte D) Se puede encontrar en el código fuente una función llamada "sentenciasConError();" esta parte que es las sentencias con error las realizamos en las tablas Cliente, Aerolinea, FormaDePago, ViajeVendido y para cada sentencia con error, se despliega el error adecuado. Esta bueno aclarar, que el error va a aparecer si solo si no hay datos en las tablas,en el caso de que haya, no habrá tal error.

Parte E) Se encuentra en el código fuente una función llamada "parteE();", ingresamos 3 operaciones de modificación (Update,Delete,Alter Table) y los errores obtenidos son los que esperábamos, es decir, como el usuario no tiene permiso para realizar dichas modificaciones, no se realizan.
Aparece el siguiente error en consola,

- ERROR: permiso denegado a la relación cliente.
- ERROR: permiso denegado a la relación viajevendido.
- ERROR: debe ser dueño de la relación cliente.

Parte F) Se encuentra en el código fuente una función llamada "parteF();" que inserta 5 viajes vendidos y elimina dos de esos insertados.

Al ejecutar en dos terminales distintas notamos que en la TerminalA inserta todo bien, pero en la TerminalB ya existen los datos.

Parte G) Se encuentra en el código fuente una función llamada "parteGTransacciones();" esta hace lo mismo que la parte anterior pero con transacciones. En esta parte ingresamos de a una transacción los valores, para que cada sentencia pueda ser ejecutada exitosamente. A su vez, cada sentencia cumple con la definicion de transaccion y por eso es que se puede agregar esos valores a las tablas. Igual, en caso de que no se pueda insertar, agregamos el rollback para que nos muestre igual una simulación de dichos valores que se tendrían que haber agregado a las tablas.

Al ejecutar en dos terminales distintas notamos que en la TerminalA inserta todo bien, pero en la TerminalB ya existen los datos.

Parte H) Se encuentra en el código fuente una función llamada menu8();, este Menú consta de 8 opciones:

1. Ingresar 200 clientes

```
La funcion a demorado 70 milisegundos
Clientes agregados...
```

2. Ingresar 50 empleados

```
La funcion a demorado 40 milisegundos
Empleados agregados...
```

3. Ingresar 50 aerolíneas

```
La funcion a demorado 51 milisegundos
Aerolineas agregadas...
```

4. Ingresar 100 ciudades

```
La funcion a demorado 63 milisegundos
Ciudades agregadas...
```

5. Ingresar 15 formas de pago

```
La funcion a demorado 52 milisegundos
Formas de pagos agregadas...
```

6. Ingresar 360 viajes vendidos

```
La funcion a demorado 290 milisegundos
Viajes vendidos agregados...
```

7. Modificar las ciudades de destino y los clientes de la mitad de los viajes y elimina la otra mitad de los viajes que no fueron modificados.

```
La funcion a demorado 114 milisegundos
Modificar destinos...
```

0. Salir al "menuPARTE4();".

Parte I) Se encuentra en el código fuente una función llamada "menu9();", este Menú consta de las misma 8 opciones pero con transacciones.

1. Ingresar 200 clientes

```
La funcion a demorado 73 milisegundos
Clientes agregados...
```

2. Ingresar 50 empleados

```
La funcion a demorado 44 milisegundos
Empleados agregados...
```

3. Ingresar 50 aerolíneas

```
La funcion a demorado 57 milisegundos
Aerolineas agregadas...
```

4. Ingresar 100 ciudades

```
La funcion a demorado 55 milisegundos
Ciudades agregadas...
```

5. Ingresar 15 formas de pago

```
La funcion a demorado 51 milisegundos
Formas de pagos agregadas...
```

6. Ingresar 360 viajes vendidos

```
La funcion a demorado 289 milisegundos
Viajes vendidos agregados...
```

7. Modificar las ciudades de destino y los clientes de la mitad de los viajes y elimina la otra mitad de los viajes que no fueron modificados.

```
La funcion a demorado 381 milisegundos
Modificar destinos...
```

0. Salir al “menuPARTE4();”.

Parte J) Se encuentra en el código fuente una función “tiempodeEjecucionParteJ();”, ésta invoca la función “parteJ();” y devuelve el tiempo de ejecución de dicha función.
La función “parteJ();”, hace un SELECT a la tabla ViajeVendido, ordenado por el nombre de ciudad de origen de forma descendente, como dichos datos fuernos insertados por un bucle (FOR) de forma ascendente, al terminar esta función se imprime en pantalla el tiempo de ejecución.

```
La funcion a demorado 553 milisegundos
Select parte J...
```

Parte K)

Primary index: es creado por defecto en las tablas al usar la sentencia PRIMARY KEY
Common index: acelera la búsqueda para el campo/s sobre el que se crea el índice, en este campo los valores no necesariamente son únicos y aceptan valores nulos.
Unique index: acelera la búsqueda para el campo/s sobre el que se crea el índice, en este campo/s los valores deben ser únicos y diferentes, aunque permite valores nulos y pueden definirse varios por tabla.

En nuestro código ya estaba por defecto el Primary Index, pero creamos otro con

“origenVuelo” que fue el que hicimos consulta.

“CREATE INDEX viajesv_idx ON ViajeVendido(origenVuelo);"

Parte L) Se encuentra en el código fuente una función “tiempodeEjecucionParteL();”, ésta invoca la función “parteL();” y devuelve el tiempo de ejecución de dicha función, la función “parteL();” crea un índice para la función “parteJ();”, y nos devuelve un select de la tabla “ViajeVendido”, ordenado por el nombre de ciudad de origen de forma descendente con el el tiempo que llevo en ejecutar la función, al terminar la función se imprime en pantalla el tiempo de ejecución.

```
La funcion a demorado 553 milisegundos
Select parte J...
```

Como vemos usando índices se hace mas rapido la obtención de datos.

Parte V) A continuación agregamos la Store Procedure correspondiente a lo solicitado:

```
CREATE OR REPLACE PROCEDURE par5 ()
--RETURNS DOUBLE PRECISION AS $$
AS $$
DECLARE
        cont integer := 0;
        ciaux bigint :=0;
        idVuelo integer :=000;
        nomaux varchar :='nombre';
        apeaux varchar :='apellido';
        defaultdate date;
        inicio timestamptz;
        fin timestamptz;
        total double precision;
BEGIN
                inicio:= clock_timestamp();

                --EMPLEADO

                cont:=1;
                LOOP
                        EXIT WHEN cont<=50;
                        INSERT INTO Empleado (ciE, nombre, apellido, fechaNac) VALUES
(" || cont,'nombre'|| cont,'apellido'|| cont,'05/06/2018');
                        cont := cont+1;
                END LOOP;

                --CLIENTE
                cont:=1;
                LOOP
```

```
                        EXIT WHEN cont<=200;
                        ciaux:=ciaux+cont;
                        INSERT INTO Cliente (ciC, nombre, apellido, fechaNac) VALUES ('' ||
cont,'nombre' || cont,'apellido' || cont,'05/06/2018');
                        cont := cont+1;
                END LOOP;

                --AEROLINEA

                cont:=1;
                LOOP
                        EXIT WHEN cont<=50;
                        INSERT INTO Aerolinea (nombre) VALUES ('aerolinea'||cont);
                        cont := cont+1;
                END LOOP;

                --CIUDAD

                cont:=1;
                LOOP
                        EXIT WHEN cont<=100;
                        INSERT INTO Ciudad (nombre) VALUES ('ciudad'|| cont);
                        cont := cont+1;
                END LOOP;

                --FORMA DE PAGO

                cont:=1;
                LOOP
                        EXIT WHEN cont<=15;
                        ciaux:=ciaux+cont;
                        INSERT INTO FormadePago (ciCliente, debito, credito) VALUES ('' ||
cont,true,false);
                        cont := cont+1;
                END LOOP;
                cont:=1;

                --VIAJE VENDIDO 0 escala

                LOOP
                        EXIT WHEN cont=50;
                        ciaux:=ciaux+cont;
                        idVuelo:=idVuelo+cont;
                        INSERT INTO ViajeVendido
(idVueloOrigen,idVueloDestino,origenVuelo,destinoVuelo,horario_partida,horario_llegada,ci
Empleado,ciCliente,escala,montoVenta,fechaCompra,nombreAero,cantEscala) VALUES
(cont,cont,'ciudad'|| cont,'ciudad' || cont,'11:00:00','13:45:00',''||cont,'' ||
cont,false,2400,'05/06/2018','aerolinea' || cont,0);
```

```
		cont := cont+1;
	END LOOP;
	cont:=1;
	LOOP
		EXIT WHEN cont=30;
		ciaux:=ciaux+cont;
		idVuelo:=idVuelo+cont;
		INSERT INTO ViajeVendido
(idVueloOrigen,idVueloDestino,origenVuelo,destinoVuelo,horario_partida,horario_llegada,ci
Empleado,ciCliente,escala,montoVenta,fechaCompra,nombreAero,cantEscala) VALUES
(cont,cont,'ciudad'|| cont,'ciudad' || cont,'11:00:00','13:45:00',''||cont,'' ||
cont,false,2400,'05/06/2018','aerolinea' || cont,0);
		cont := cont+1;
	END LOOP;
	cont:=1;

	--VIAJE VENDIDO 1 escala

	LOOP
		EXIT WHEN cont<=50;
		ciaux:=ciaux+cont;
		idVuelo:=idVuelo+cont;
		INSERT INTO ViajeVendido
(idVueloOrigen,idVueloDestino,origenVuelo,destinoVuelo,horario_partida,horario_llegada,ci
Empleado,ciCliente,escala,montoVenta,fechaCompra,nombreAero,cantEscala) VALUES
(cont,cont,'ciudad' || cont,'ciudad' || cont,'11:00:00','13:45:00',''||cont,'' ||
cont,true,2400,'05/06/2018','aerolinea'||cont,1);
		cont := cont+1;
	END LOOP;
	cont:=1;
	LOOP
		EXIT WHEN cont<=50;
		ciaux:=ciaux+cont;
		idVuelo:=idVuelo+cont;
		INSERT INTO ViajeVendido
(idVueloOrigen,idVueloDestino,origenVuelo,destinoVuelo,horario_partida,horario_llegada,ci
Empleado,ciCliente,escala,montoVenta,fechaCompra,nombreAero,cantEscala) VALUES
(cont,cont,'ciudad' || cont,'ciudad' || cont,'11:00:00','13:45:00',''||cont,'' ||
cont,true,2400,'05/06/2018','aerolinea'||cont,1);
		cont := cont+1;
	END LOOP;
	cont:=1;

	--VIAJE VENDIDO 2 escala

	LOOP
		EXIT WHEN cont<=50;
		ciaux:=ciaux+cont;
```

```
                idVuelo:=idVuelo+cont;
                INSERT INTO ViajeVendido
(idVueloOrigen,idVueloDestino,origenVuelo,destinoVuelo,horario_partida,horario_llegada,ci
Empleado,ciCliente,escala,montoVenta,fechaCompra,nombreAero,cantEscala) VALUES
(cont,cont,'ciudad' || cont,'ciudad' || cont,'11:00:00','13:45:00',''||cont,'' ||
cont,true,2400,'05/06/2018','aerolinea'||cont,2);
                cont := cont+1;
        END LOOP;
        cont:=1;
        LOOP
                EXIT WHEN cont<=50;
                ciaux:=ciaux+cont;
                idVuelo:=idVuelo+cont;
                INSERT INTO ViajeVendido
(idVueloOrigen,idVueloDestino,origenVuelo,destinoVuelo,horario_partida,horario_llegada,ci
Empleado,ciCliente,escala,montoVenta,fechaCompra,nombreAero,cantEscala) VALUES
(cont,cont,'ciudad' || cont,'ciudad' || cont,'11:00:00','13:45:00',''||cont,'' ||
cont,true,2400,'05/06/2018','aerolinea'||cont,2);
                cont := cont+1;
        END LOOP;
        cont:=1;

        --VIAJE VENDIDO 3 escala

        LOOP
                EXIT WHEN cont<=50;
                ciaux:=ciaux+cont;
                idVuelo:=idVuelo+cont;
                INSERT INTO ViajeVendido
(idVueloOrigen,idVueloDestino,origenVuelo,destinoVuelo,horario_partida,horario_llegada,ci
Empleado,ciCliente,escala,montoVenta,fechaCompra,nombreAero,cantEscala) VALUES
(cont,cont,'ciudad' || cont,'ciudad' || cont,'11:00:00','13:45:00',''||cont,'' ||
cont,true,2400,'05/06/2018','aerolinea'||cont,3);
                cont := cont+1;
        END LOOP;
        cont:=1;
        LOOP
                EXIT WHEN cont<=30;
                ciaux:=ciaux+cont;
                idVuelo:=idVuelo+cont;
                INSERT INTO ViajeVendido
(idVueloOrigen,idVueloDestino,origenVuelo,destinoVuelo,horario_partida,horario_llegada,ci
Empleado,ciCliente,escala,montoVenta,fechaCompra,nombreAero,cantEscala) VALUES
(cont,cont,'ciudad' || cont,'ciudad' || cont,'11:00:00','13:45:00',''||cont,'' ||
cont,true,2400,'05/06/2018','aerolinea'||cont,3);
                cont := cont+1;
        END LOOP;
        fin:= clock_timestamp();
```

```
        total := 1000 * ( extract(epoch from fin) - extract(epoch from inicio) );
        RAISE NOTICE 'Duracion: =%', total;



END;
$$ LANGUAGE plpgsql;
```

Para esta parte no supimos como utlizar transacciones ya que siempre nos tiraba algun error, pero se ejecuta bien de todas formas.

Parte VI) Se puede encontrar en el código fuente, en una función llamada tiempodeEjecucionParte6();, que llama a la función llamada parte6(); que esta llama a la sentencia SQL ingresada en el punto anterior. Al finalizar la función se imprime el tiempo de ejecución en la pantalla.

```
La funcion a demorado 55 milisegundos
Store Procedure exitoso...!
```

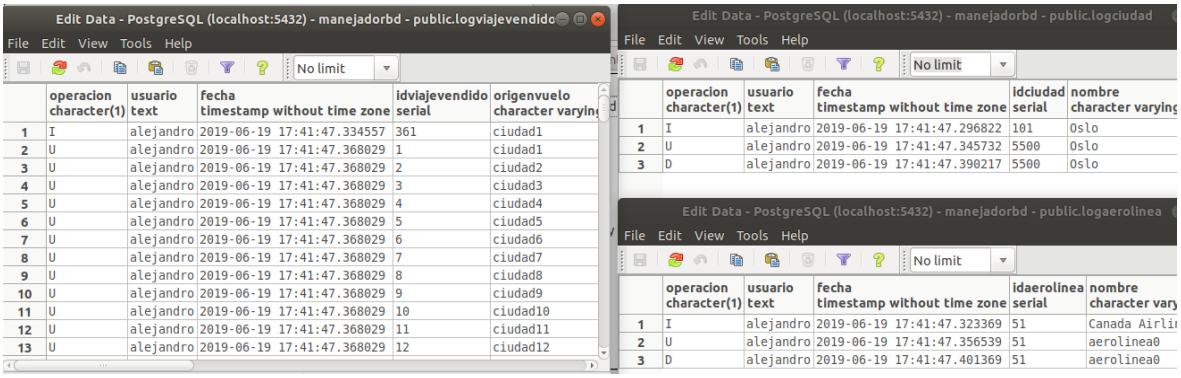
Parte VII) Agregamos los TRIGGERS correspondientes a la tarea:

```
CREATE TABLE logCiudad (
        operacion char(1) not null,
        usuario text not null,
        fecha TIMESTAMP not null,
        idCiudad serial NOT NULL,
        nombre varchar(45) NOT NULL
);

CREATE OR REPLACE FUNCTION trigger_Ciudad()RETURNS trigger AS $$
BEGIN
        IF (TG_OP = 'INSERT') THEN  INSERT INTO logCiudad (operacion, usuario, fecha,
idCiudad, nombre) VALUES ('I', user, now(), new.idCiudad, new.nombre);
        RETURN new;

        ELSEIF (TG_OP = 'UPDATE') THEN INSERT INTO logCiudad (operacion, usuario,
fecha, idCiudad, nombre) VALUES ('U', user, now(), new.idCiudad, new.nombre);
        RETURN new;

        ELSEIF (TG_OP = 'DELETE') THEN INSERT INTO logCiudad (operacion, usuario,
fecha, idCiudad, nombre) VALUES ('D', user, now(), old.idCiudad, old.nombre);
        RETURN old;

        END IF;
        RETURN null;
```

```
END;
$$ language plpgsql;

CREATE TRIGGER trigger_Ciudad AFTER INSERT OR UPDATE OR DELETE ON ciudad
FOR EACH ROW EXECUTE PROCEDURE trigger_Ciudad();

CREATE TABLE logAerolinea (
        operacion char(1) not null,
        usuario text not null,
        fecha TIMESTAMP not null,
        idAerolinea serial,
        nombre varchar(25) NOT NULL
);
CREATE OR REPLACE FUNCTION trigger_Aerolinea()RETURNS trigger AS $$
BEGIN
        IF (TG_OP = 'INSERT') THEN INSERT INTO logAerolinea (operacion, usuario,
fecha, idAerolinea, nombre) VALUES ('I', user, now(), new.idAerolinea, new.nombre);
        RETURN new;

        ELSEIF (TG_OP = 'UPDATE') THEN INSERT INTO logAerolinea (operacion,
usuario, fecha, idAerolinea, nombre) VALUES ('U', user, now(), new.idAerolinea,
new.nombre);
        RETURN new;

        ELSEIF (TG_OP = 'DELETE') THEN INSERT INTO logAerolinea (operacion, usuario,
fecha, idAerolinea, nombre) VALUES ('D', user, now(), old.idAerolinea, old.nombre);
        RETURN old;

        END IF;
        RETURN null;
END;
$$ language plpgsql;

CREATE TRIGGER trigger_Aerolinea AFTER INSERT OR UPDATE OR DELETE ON
aerolinea FOR EACH ROW EXECUTE PROCEDURE trigger_aerolinea();

CREATE TABLE logViajeVendido (
        operacion char(1) not null,
        usuario text not null,
        fecha TIMESTAMP not null,
        idViajeVendido serial,
        origenVuelo varchar(45) NOT NULL,
        destinoVuelo varchar(45) NOT NULL,
        horario_partida time NOT NULL,
        horario_llegada time NOT NULL,
        ciEmpleado varchar(8) NOT NULL,
        ciCliente varchar(8) NOT NULL,
        escala boolean not null,
```

```
	montoVenta real NOT NULL,
	fechaCompra date NOT NULL,
	nombreAero varchar(25) NOT NULL,
	cantEscala int not null
);
CREATE OR REPLACE FUNCTION trigger_ViajeVendido()RETURNS trigger AS $$
BEGIN
	IF (TG_OP = 'INSERT') THEN INSERT INTO logViajeVendido (operacion, usuario,
fecha, idViajevendido, origenVuelo, destinoVuelo, horario_partida, horario_llegada,
ciEmpleado, ciCliente, escala, montoVenta, fechaCompra, nombreAero, cantEscala)
VALUES ('I', user, now(), new.idViajevendido, new.origenVuelo, new.destinoVuelo,
new.horario_partida, new.horario_llegada, new.ciEmpleado, new.ciCliente, new.escala,
new.montoVenta, new.fechaCompra, new.nombreAero, new.cantEscala);
	RETURN new;

	ELSEIF (TG_OP = 'UPDATE') THEN INSERT INTO logViajeVendido (operacion,
usuario, fecha, idViajevendido, origenVuelo, destinoVuelo, horario_partida, horario_llegada,
ciEmpleado, ciCliente, escala, montoVenta, fechaCompra, nombreAero, cantEscala)
VALUES ('U', user, now(), new.idViajevendido, new.origenVuelo, new.destinoVuelo,
new.horario_partida, new.horario_llegada, new.ciEmpleado, new.ciCliente, new.escala,
new.montoVenta, new.fechaCompra, new.nombreAero, new.cantEscala);
	RETURN new;

	ELSEIF (TG_OP = 'DELETE') THEN INSERT INTO logViajeVendido (operacion,
usuario, fecha, idViajevendido, origenVuelo, destinoVuelo, horario_partida, horario_llegada,
ciEmpleado, ciCliente, escala, montoVenta, fechaCompra, nombreAero, cantEscala)
VALUES ('D', user, now(), old.idViajevendido, old.origenVuelo, old.destinoVuelo,
old.horario_partida, old.horario_llegada, old.ciEmpleado, old.ciCliente, old.escala,
old.montoVenta, old.fechaCompra, old.nombreAero, old.cantEscala);
	RETURN old;

	END IF;
	RETURN null;
END;
$$ language plpgsql;

CREATE TRIGGER trigger_viajeVendido AFTER INSERT OR UPDATE OR DELETE ON
viajevendido FOR EACH ROW EXECUTE PROCEDURE trigger_viajeVendido();
```

Parte VIII) Se encuentra en el código fuente una función llamada parte8();, que invoca a los TRIGGERS creados en el punto anterior.


Edit Data - PostgreSQL (localhost:5432) - manejadorbd - public.logviajevendido
File Edit View Tools Help
No limit
operacion character(1) | usuario text | fecha timestamp without time zone | idviajevendido serial | origenvuelo character varying
1 I alejandro 2019-06-19 17:41:47.334557 361 ciudad1
2 U alejandro 2019-06-19 17:41:47.368029 1 ciudad1
3 U alejandro 2019-06-19 17:41:47.368029 2 ciudad2
4 U alejandro 2019-06-19 17:41:47.368029 3 ciudad3
5 U alejandro 2019-06-19 17:41:47.368029 4 ciudad4
6 U alejandro 2019-06-19 17:41:47.368029 5 ciudad5
7 U alejandro 2019-06-19 17:41:47.368029 6 ciudad6
8 U alejandro 2019-06-19 17:41:47.368029 7 ciudad7
9 U alejandro 2019-06-19 17:41:47.368029 8 ciudad8
10 U alejandro 2019-06-19 17:41:47.368029 9 ciudad9
11 U alejandro 2019-06-19 17:41:47.368029 10 ciudad10
12 U alejandro 2019-06-19 17:41:47.368029 11 ciudad11
13 U alejandro 2019-06-19 17:41:47.368029 12 ciudad12
Edit Data - PostgreSQL (localhost:5432) - manejadorbd - public.logciudad
File Edit View Tools Help
No limit
operacion character(1) | usuario text | fecha timestamp without time zone | idciudad serial | nombre character varying
1 I alejandro 2019-06-19 17:41:47.296822 101 Oslo
2 U alejandro 2019-06-19 17:41:47.345732 5500 Oslo
3 D alejandro 2019-06-19 17:41:47.390217 5500 Oslo
Edit Data - PostgreSQL (localhost:5432) - manejadorbd - public.logaerolinea
File Edit View Tools Help
No limit
operacion character(1) | usuario text | fecha timestamp without time zone | idaerolinea serial | nombre character vary
1 I alejandro 2019-06-19 17:41:47.323369 51 Canada Airli
2 U alejandro 2019-06-19 17:41:47.356539 51 aerolinea0
3 D alejandro 2019-06-19 17:41:47.401369 51 aerolinea0