DEPARTAMENTO DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

Trabajo Práctico 1

Informe

Bases de datos

Primer Cuatriméstre de 2013

Grupo 1

Integrante	LU	Correo electrónico
Pablo Gauna	334/09	gaunapablo@gmail.com
Julián Sackmann	540/09	jsackmann@gmail.com
Manuel Ferreria	199/10	m.ferreria@gmail.com
Juan Pablo Darago	272/10	jpdarago@gmail.com

Facultad de Ciencias Exactas y Naturales

Universidad de Buenos Aires

Ciudad Universitaria - (Pabellón I/Planta Baja)

Intendente Güiraldes 2160 - C1428EGA

Ciudad Autónoma de Buenos Aires - Rep. Argentina

Tel/Fax: (54 11) 4576-3359

http://www.fcen.uba.ar

Índice

1. Introducción

En este trabajo práctico se detalla el modelado realizado para el *backend*, en particular el subsistema de almacenamiento de datos mediante una base de datos relacional, de un sistema online de reserva de pasajes (problema referido en el enunciado bajo el nombre de *booking*). Este problema consiste en mantener un sistema de reservas de pasajes de manera que el sistema en su totalidad (es decir más alla del módulo de base de datos implementado) permita a los usuarios consultar y administrar reservas para pasajes de avión.

En primer lugar se detalla el DER (Diagrama Entidad Relación) realizado para este trabajo, junto con las limitaciones adicionales al modelo expresadas en lenguaje coloquial. Posteriormente se detalla el Modelo Relacional derivado del Diagrama Entidad Relación. Finalmente, se detalla la implementación (mediate SQL) de 3 funcionalidades pedidas en el enunciado del trabajo práctico. Las mismas se detallan acordemente antes de mostrarse su implementación.

2. Diagrama Entidad Relación

2.1. Diagrama en si

A continuación incluimos el Diagrama Entidad Relación (DER) correspondiente al diseño realizado para este trabajo. El mismo se puede ver en la Figura ??

2.2. Restricciones adicionales en lenguaje coloquial

Las siguientes limitaciones adicionales del Diagrama Entidad Relación mostrado anteriormente, y que no se pueden detallar utilizando la notación misma, se incluyen a continuación.

- Un usuario puede tener hasta 3 ciudades favoritas (restricción tomada del enunciado).
- Los índices de escala de un vuelo con escalas son consecutivos. Adicionalmente, se cumple que los aeropuertos de llegada y de salida es encadenan (esta restricción, si bien no es necesaria la tomamos por simplicidad). Por "se encadenan" se entiende que, dada la escala con índice i ocurre que
  - $i = 0$ o el aeropuerto de llegada de la escala $i - 1$ es el aeropuerto de salida de la escala i.
  - $i = n - 1$ con n la cantidad de escalas del viaje con escalas considerado, o el aeropuerto de llegada de la escala i es el aeropuerto de salida de la escala $i + 1$.
- El vuelo esta determinado no solo por el vuelo en si sino también por la clase del asiento. No se hacen otras distinciones entre asientos dentro de una misma clase ni nada por el estilo.
- El precio de un vuelo esta determinado por la clase y el viaje, para cada uno de los viajes que corresponden a las escalas del vuelo.
- La cantidad de reservas que tienen un vuelo bajo una cierta clase dentro de su lista (considerando escalas) no puede superar la cantidad de asientos disponibles para esa clase por la aeronave del vuelo.

2.3. Explicación de las decisiones tomadas

A continuación detallamos el por qué de algunas de las decisiones tomadas para el diseño del Diagrama Entidad Relación.

- Se considero, como se explico anteriormente, que un viaje con escalas consiste de un viaje con varias escalas. El precio del viaje esta determinado por las relación entre los vuelos y clases de los asientos para cada uno los varios vuelos del viaje, considerando además las tasas de llegada de los aeropuertos de llegada y salida.
- Se considero preferible mantener la información de los datos del viajero separado en cada reserva: La motivación de esto es que se ve por que el enunciado dice que una reserva puede ser realizada en nombre de otra persona (sin que esa persona sea un usuario registrado). Sin embargo, se le cobrará a la persona registrada que realizó la reserva (usando para ello los datos de cobro que tiene el usuario registrado). Si bien esto puede introducir redundancia cuando un usuario saca una reserva para si mismo, la otra opción sería mantener valores especiales. Preferimos mantener una duplicación (se puede para ello usar un *trigger* en los inserts) para mantener la simplicidad de los datos (las otras opciones consideradas implicaban mantener posibles nulls lo cual implicaba mantener una lógica implícita independiente de los datos guardados).

- Se decidió que las multiples reservas de un usuario se modelan como reservas separadas. Sin embargo, una reserva puede comprender varios vuelos.
- Algunos de los detalles del enunciado, a falta de una especificación más fina de los datos, se decidió mantenerlos como campos de texto con contenido indeterminado. Ejemplos incluyen: la composición de la tripulación, los datos de llegada y salida de un aeropuerto, los datos de la persona a nombre de cual se hace una reserva, la tasa de un aeropuerto, el modelo de una aeronave, etc.

3. Modelo Relacional

A continuación detallamos el Modelo Relacional derivado del Diagrama Entidad Relación realizado anteriormente. Una aclaración de notación: PK corresponde al término *Primary Key* o *Clave primaria*, CK corresponde a *Candidate Key* o *Clave candidata*, y FK corresponde a *Foreign Key* o *Clave foránea*.

- **Usuario**(idUsuario,username,nombre,apellido,telefono,fechaNacimiento, preferencias, direccion,profesion,email,clavehash,idClase,idPaisNacimiento)
  - *PK:* idUsuario
  - *CK:* idUsuario,username
  - *FK:* idClase,idPaisNacimiento
- **Tarjeta**(idTarjeta,username,empresa,nroTarjeta,codigoSeguridad,direccion)
  - *PK:* idUsuario,idTarjeta
  - *CK:* (idUsuario,idTarjeta), (idUsuario,empresa,nroTarjeta)
  - *FK:* username
- **Reservas**(idReserva,username,tipoPago,fechaCaducidad,datosViajante)
  - *PK = CK:* idReserva
  - *FK:* username
- **ViajesConEscalas**(idViajeConEscalas,idViajePartida,idViajeLlegada)
  - *PK = CK:* idViajeConEscalas
  - *FK:* idViajePartida,idViajeLlegada
- **EsReservaPara**(idReserva,idVueloConEscalas,idClase,costo)
  - *PK = CK:* (idReserva,idVueloConEscalas,idClase)
  - *FK:* idReserva,idVueloConEscalas,idClase
- **Vuelos**(idVuelo,idAeronave,fechaSalida,fechaLlegada,idAeropuertoSalida,idAeropuertoLlegada)
  - *PK = CK:* idVuelo
  - *FK:* idAeronave,idAeropuertoSalida,idAeropuertoLlegada
- **HaceEscalaEn**(idVueloConEscalas,idVuelo,NumeroEscala)
  - *PK = CK:* (idVueloConEscalas,idVuelo)
  - *FK:* idVuelo,idVueloConEscalas
- **Aeropuertos**(idAeropuerto,tasa,opcionesTransporte, nombre, idCiudad)
  - *PK:* idAeropuerto
  - *CK:* nombre
  - *FK:* idPais
- **TelefonoAeropuerto** (idAeropuerto,número)
  - *PK = CK:* (idAeropuerto,número)
  - *FK:* idAeropuerto
- **Aeronaves**(idAeronave,tripulacion,millas,modelo,idPaisOrigen)

- *PK = CK:* idAeronave
- *FK:* idPaisOrigen

- **Paises**(idPais,nombre)
  - *PK:* idPais
  - *CK:* idPais, nombre
- **Ciudades**(idCiudad,nombre)
  - *PK:* idCiudad
  - *CK:* idCiudad, nombre
- **Clases**(idClase,nombre)
  - *PK:* idClase
  - *CK:* idClase, nombre
- **DisponeDeAsientos**(idAeronave,idClase,asientos)
  - *PK = CK:* (idAeronave,idClase)
  - *FK:* idClase, idAeronave

4. Funcionalidad

- Primera funcionalidad:
  - Enunciado: Mediante SQL escribir una consulta para obtener el nombre, apellido y nombre identificatorio de aquellos pasajeros que han viajado a todos los paises cubertis por la línea aérea en los últimos 5 años.
  - Resolución:

```
SELECT nombre, apellido, idUsuario FROM usuarios
    WHERE idUsuario IN (
        SELECT idUsuario FROM usuarios WHERE NOT EXISTS (
            SELECT p.idPais FROM paises WHERE NOT EXISTS (
                SELECT idReserva FROM reservas r
                JOIN esReservaPara ep ON r.idReserva = ep.idReserva
                JOIN viajeConEscalas vce
                    ON vce.idViajeConEscalas = ep.idViajeConEscalas
                JOIN viaje v ON v.idViaje = vce.idViajeLlegada
                JOIN aeropuerto ap ON v.llega = ap.idAeropuerto
                JOIN ciudades c ON c.idCiudad = ap.idCiudad
                    WHERE c.idPais = p.idPais
            )
        )
    )
```

- Segunda funcionalidad:
  - Enunciado: Controlar mediante alguna restricción que un usuario no pueda realizar reservas que se superpongan en el tiempo. La única excepción que se permite consiste en que un usuario puede realizar a lo sumo dos reservas para la misma fecha de viaje entre los mismos aeropuertos de origen y destino, siempre que la fecha de partida no se encuentre dentro de los próximos 7 días.
  - Resultado: Vamos a utilizar un *trigger*:

```
CREATE TRIGGER RestringirReservasUsuario ON Reservas
    BEFORE INSERT
    FOR EACH ROW
    BEGIN

    DECLARE fechaSalida AS DATE
    DECLARE fechaLlegada AS DATE

    -- Conseguimos las fechas de salida y llegada
    SELECT v.fechaSalida INTO fechaSalida
        FROM vuelos v, vuelosConEscalas ve, EsReservaDe er
            WHERE v.idVuelo = ve.idVueloPartida
            AND ve.idVueloConEscalas = er.idVueloConEscalas
            AND er.idReserva = :NEW.idReserva

    SELECT v.fechaLlegada INTO fechaLlegada
    FROM vuelos v, vuelosConEscalas ve, EsReservaDe er
    WHERE v.idVuelo = ve.idVueloLlegada
    AND ve.idVueloConEscalas = er.idVueloConEscalas
    AND er.idReserva = :NEW.idReserva

```

```
22      -- Nos fijamos si una reserva se superpone
23      IF( SELECT COUNT(*) FROM reservas WHERE EXISTS
24          ( SELECT * FROM vuelos v,vuelosConEscala ve,
25              EsReservaPara ep,Reservas r, Usuario u
26              WHERE v.fechaSalida <= fechaSalida
27                  AND v.fechaLlegada >= fechaLlegada
28                  AND v.
29          )
30      > 0)
31          BEGIN
32
33
34              RAISE ERROR ('Restriccion de reservas violada')
35              ROLLBACK TRANSACTION
36          END
37      END
```