Facultad de Ciencias Exactas

Tecnicatura Universitaria en Desarrollo de Aplicaciones Informáticas

Trabajo Práctico Especial

Base de Datos

"Reservas de departamentos Premium"

Meliendrez Agustín
Santos Luciano

Tandil, Junio 2018

1. Introducción

A partir de la cátedra "Base de Datos", perteneciente la carrera "Tecnicatura Universitaria en Desarrollo de Aplicaciones Informáticas" (TUDAI), de la Facultad de Ciencias Exactas, perteneciente a la Universidad Nacional del Centro de la Provincia de Buenos Aires, se planteo la resolución de un conjunto de controles y servicios sobre una base de datos que mantiene un sistema de "Reservas de Departamentos Premiun" según las una serie de consignas planteadas.

2. Desarrollo

En relación al problema planteado, se necesita hacer un sistema sobre una base de datos existente de un sistema de Gestión de servicios de hospedaje. El sistema pertenece a un servicio de alquileres de departamentos premium disponible para ejecutivos que viajan a EEUU.

Hay dueños de casas o departamentos que los proponen para alquiler temporario. La administración consiste en el control y seguimiento completo de las reservas, cada una debe tener el tipo de departamento, fechas, preferencias, etc. El periodo mínimo que se puede reservar un departamento es un día.

Por su parte, el sistema debe llevar control del mantenimiento del departamento (es decir si fue limpiado y/o si se encontró algún problema en el mismo)

A su vez, las habitaciones de los departamentos pueden cambiar su configuración de una reserva a otra, respecto de la cantidad o tipo de camas y/o comodidades (televisor, sillones, frigobar, escritorios, mesas, etc).

Para la resolución de dicho problema se crearon 3 scripts que cumple con las funcionalidades requeridas.

2.1 Creacion sql

Para la conformación del script "Creacion.sql", se utilizo la plataforma "Vertabelo", para cargar un archivo "xml", el cual contenía la estructura general de la base de datos necesaria para la resolución de la problemática.

A partir del resultado obtenido, se modificaron el nombre de las tablas cumpliendo con el requisito de "pautas de nomenclatura".

En este caso especifico, al ser el ***Grupo 2***, se realizo la siguiente denominación de las tablas, claves primarias, y claves foráneas.

- Tablas: **GR02_<<Nombre de la tabla>>.**
- Claves primarias: **PK_GR02_<<Columna referenciada>>.**
- Claves foráneas: **FK_GR02_<<Referenciante>>_<<Referenciada>>.**

A su vez, para este script se agregó la columna "**ciudad**" dentro de la tabla "GR02_Departamento", ya que iba a ser requerido en posteriores requisitos.

Por otra parte, se creó la tabla "**GR02_Tipo_doc**", la cual era necesaria para que lo utilice la tabla "**GR02_Persona**" (***tipo_doc),*** teniendo en cuenta que pueden existir diferentes tipos de documentos de identidad (Pasaporte, DNI, Partida de Nacimiento, etc.), y lo cual no estaba plasmado en el esquema inicial.

Por último, se realizaron las inserciones de información a las diferentes tablas, cumpliendo con el requisito de insertar, como mínimo, 5 filas por tabla.

2.2 Cambios sql

Para la creación del script "Cambios.sql", se realizaron controles, servicios y vistas requeridas.

Controles:

a. En el primer caso, se realizo un check, el cual controla que *las fechas de las reservas sean consistentes, es decir, que la fecha de inicio de la reserva sea menor que la fecha de finalización.* Se utilizó un check, ya que el control se realizaba en una sola tabla.

```
ALTER TABLE gr02_reserva ADD CONSTRAINT chk_fecha CHECK(
  fecha_desde < fecha_hasta
);
```

Para comprobar que el check funciona correctamente se utilizaron las siguientes sentencias:

- **INSERT INTO GR02_Reserva (id_reserva, fecha_reserva, fecha_desde, fecha_hasta, tipo, id_dpto, valor_noche, usa_limpieza, tipo_doc, nro_doc) VALUES (8, '2017-12-01', '2018-07-01', '2018-04-01' ,'Telefonica', 1, 800, 1, 1, 26243466);**

Dicha sentencia da como resultado: **el nuevo registro para la relación «gr02_reserva» viola la restricción «check» «chk_fecha»**

- **UPDATE GR02_Reserva SET fecha_desde = '2018-02-02' WHERE id_reserva = 1**

Dicha sentencia da como resultado: **el nuevo registro para la relación «gr02_reserva» viola la restricción «check» «chk_fecha».** Esto es asi ya que la fecha de fin de reserva es '2018-02-01'.

b. Para el segundo caso, se realizo un assertion que realiza el control de que el detalle de las habitaciones sea consistente con el tipo de departamento, es decir que si el tipo de departamento es de 2 habitaciones, en el detalle se consideren como máximo 2 habitaciones. En este caso, se utilizo un assertion, ya que se hacía uso de diferentes tablas diferentes.
En su forma declarativa, posee la siguiente estructura:

```
CREATE ASSERTION CK_Cantidad_Habitaciones
CHECK (NOT EXISTS
(SELECT 1
FROM gr02_departamento d
        JOIN gr02_tipo_dpto td ON (td.id_tipo_depto = d.id_tipo_depto)
        WHERE td.cant_habitaciones < (
                                SELECT COUNT(*)
                                FROM    gr02_habitacion h
                                WHERE h.id_dpto = d.id_dpto)));
```

- **INSERT INTO GR02_Habitacion (id_dpto, id_habitacion, posib_camas_simples, posib_camas_dobles, posib_camas_kind, tv, sillon, frigobar, mesa, sillas, cocina) VALUES(1,3,0,1,0,true,1,true,false,1,false);**

El departamento ya tiene el máximo de habitaciones permitidas. La sentencia muestra dicho mensaje, ya que el id_dpto 1 es de tipo 1 y puede tener máximo 2 habitaciones, las cuales ya tiene.

- **UPDATE GR02_Habitacion SET id_dpto = 1 WHERE id_dpto = 2 and id_habitacion = 3**

El departamento ya tiene el máximo de habitaciones permitidas.

- **UPDATE GR02_Departamento SET id_tipo_depto = 5 WHERE id_dpto = 1**

El departamento ya tiene el máximo de habitaciones permitidas. Da ese error porque se quiere cambiar el tipo de departamento a uno que tiene 1 habitación y el departamento ya tiene 2 habitaciones asociadas.

- **UPDATE GR02_Tipo_Dpto SET cant_habitaciones = 1 WHERE id_tipo_depto = 1**

Hay un departamento con más habitaciones que el máximo. Da ese error porque se le quiere poner al id_tipo_depto 1, 1 sola habitación, y ya existe un departamento que posee 2 habitaciones asociadas.

c. En tercer lugar, se realizo un control para que, tanto la persona que realiza la reserva, como los huéspedes, no sea el propietario del departamento reservado. Para ello, se realizo un assertion el cual es:

```
CREATE ASSERTION CK_Cantidad_Habitaciones
CHECK (NOT EXISTS
        (SELECT 1
        FROM gr02_reserva r
        JOIN gr02_huesped_reserva hr ON (hr.id_reserva = r.id_reserva)
        WHERE EXISTS (SELECT 1
                        FROM gr02_departamento d
                        WHERE (d.id_dpto = r.id_dpto)
                        AND
                        ((d.tipo_doc = r.tipo_doc and d.nro_doc = r.nro_doc)
                        OR
                        (d.tipo_doc = hr.tipo_doc and d.nro_doc = hr.nro_doc)))));
```

- **INSERT INTO GR02_Reserva (id_reserva, fecha_reserva, fecha_desde, fecha_hasta, tipo, id_dpto, valor_noche, usa_limpieza, tipo_doc, nro_doc) VALUES(8,'2017-12-01','2018-07-01','2018-08-01','Telefonica',1,800,1,1,36626800);**

Huésped es propietario. Da ese error porque el huésped que intenta hacer la reserva es el propietario del departemento con id_dpto 1.

- **UPDATE GR02_Reserva SET nro_doc = 36626800 WHERE id_reserva = 1**

Huésped es propietario. El huésped con el nro_doc 36626800 no puede ser el que hizo la reserva porque es el propietario de ese departamento.

- **INSERT INTO GR02_Huesped_Reserva (tipo_doc, nro_doc, id_reserva) VALUES (1, 22334502, 4).**

Huésped es propietario. Da ese error porque el huésped que se quiere agregar a la reserva es el propietario del departamento de esa reserva.

- **UPDATE gr02_huesped_reserva SET nro_doc = 19243466 WHERE id_reserva = 6 AND nro_doc = 26243466**

Huésped es propietario. ese error se debe a que se quiere actualizar el huésped que hizo la reserva y el que se quiere colocar ya es propietario

d. En cuarto lugar, se realizo un assertion para controlar que la cantidad de huéspedes no supere la cantidad máxima de personas permitidas para una reserva.

```
CREATE ASSERTION CK_Cantidad_Huespedes
CHECK (NOT EXISTS
        (SELECT r.*
        FROM gr02_reserva r
        JOIN gr02_departamento d ON (r.id_dpto = d.id_dpto)
        JOIN gr02_tipo_dpto td ON (td.id_tipo_depto = d.id_tipo_depto)
        WHERE td.cant_max_huespedes <
                        (SELECT count(*)
                        FROM gr02_huesped_reserva hr
                        WHERE hr.id_reserva = r.id_reserva)));
```

- **INSERT INTO GR02_Huesped_Reserva (tipo_doc, nro_doc, id_reserva) VALUES (1, 22334502, 7);**

La cantidad de huéspedes excede el máximo de la habitación. La reserva ya tiene dos huéspedes y el máximo de ese departamento es 2

- **UPDATE GR02_Huesped_Reserva SET id_reserva = 7 WHERE id_reserva = 2 and nro_doc = 32243466**

La cantidad de huéspedes excede el máximo de la habitación. Se quiere pasar un huésped de la reserva 2 a la 7 pero como la reserva 7 ya tiene el máximo permitido muestra ese mensaje de error.

- **UPDATE GR02_Reserva SET id_dpto = 5 WHERE id_reserva = 1**

La cantidad de huéspedes excede el máximo de la habitación. Se quiere cambiar el departamento, pero como la reserva 1 tiene 4 huéspedes y el departamento 5 acepta un máximo de dos huéspedes, muestra ese mensaje de error.

- **UPDATE GR02_Departamento SET id_tipo_depto = 1 WHERE id_dpto = 5**
- **INSERT INTO GR02_Huesped_Reserva(tipo_doc,nro_doc,id_reserva) VALUES(1,22334502,5)**
- **UPDATE GR02_Departamento SET id_tipo_depto = 5 WHERE id_dpto = 5**

Se actualiza el tipo del departamento 5 que acepta 4 huéspedes. Luego se agrega un huésped a la reserva. En el caso de querer modificar nuevamente el tipo de departamento, para que sea del tipo 5, esto no es posible ya que este ultimo acepta solo 2 huéspedes, y la cantidad de huéspedes de dicha reserva excede a esa cantidad.

- **UPDATE GR02_Tipo_Dpto SET cant_max_huespedes = 2 WHERE id_tipo_depto = 1**

La cantidad de huéspedes de una reserva excede el máximo. Como una reserva para un departamento de ese tipo tiene más de 2 huéspedes, muestra una excepción al querer ponerle que el máximo es 2.

Cabe destacar que, en el script "Cambios.sql", están presentadas tanto la forma declarativa de los assertion (presentada en los anteriores ítems) las cuales están comentadas, como así también la forma que es soportada por postgres.

Servicios

1. El primer servicio solicitaba que, por cada departamento en el sistema, obtener el estado en una fecha determinada. Es decir, realizar una consulta la cual responde si el departamento está Ocupado o Libre en una fecha determinada.
   Para la resolución de dicho servicio, se realizo una función la cual devuelve una tabla, y la cual indica el estado del departamento, según la fecha pasada por parámetro.
   Para ello, se debe realizar la llamada a la función de la siguiente manera:

   SELECT * FROM FN_GR02_Departamento_Estado('2018-04-01');

2. El segundo servicio solicitado, requería que dado un rango de fechas y una ciudad, devuelva una lista de departamentos disponibles.
   Para ello, al igual que en el servicio anterior, se hizo uso de una función la cual, solicita por parámetro a la fecha de inicio, fecha final, y una ciudad determinada.

SELECT * FROM FN_GR02_Departamentos_Disponibles('2018-02-02','2018-02-08','Mar del Plata');

Vistas

En tercer y último lugar, se realizo la implementación de las vistas las cuales debían implementarse, respondiendo a la funcionalidad de que:

1. Devuelva un listado de todos los departamentos del sistema junto con la recaudación de los mismos en los últimos 6 meses.
2. Devuelva un listado con los departamentos ordenados por ciudad y por mejor rating (estrellas).

En ambos casos, y en base a la implementación realizada, ninguna de las dos vistas son actualizables, ya que contienen información derivada de una función a partir de los datos obtenidos. En el primer caso se utiliza la función "**SUM(p.importe)**", para obtener la recaudación de cada departamento.
Por su parte, en el segundo caso se utiliza la función "**avg(c.estrellas)**", con el objetivo de obtener un promedio entre las valoraciones de estrellas en cada departamento, para sacar un rating de los mismos.
A su vez, dichas vistas no son actualizables ya que, las funciones utilizadas, se están realizando a columnas que se encuentran en otras tablas.

2.3 Borrado sql

Por último, se realizo un script denominado "Borrado.sql", el cual es el encargado de borrar, tanto los datos insertados, como las tablas y funciones creadas, volviendo al estado original, antes de comenzar con el primer script.