Departamento de Cs. e Ingeniería de la Computación

Universidad Nacional del Sur

Bases de Datos

Segundo Cuatrimestre de 2019

Proyecto N° 3

Implementación de un sistema con base de datos: Reserva de Pasajes Aéreos

Reserva de Pasajes Aéreos

Para realizar una reserva se necesitan conocer los siguientes datos:

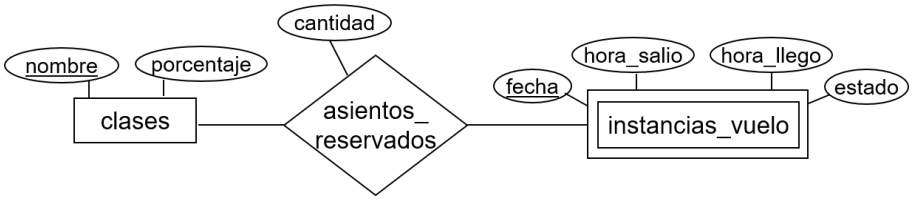
- número, fecha y clase del vuelo que el pasajero desea reservar.
- tipo y número de documento de un pasajero.
- número de legajo del empleado que gestiona la reserva.

Antes de realizar la reserva es necesario verificar que haya lugares disponibles en la clase y vuelo elegidos. Si hay lugares disponibles, la reserva se llevará a cabo insertando los datos correspondientes en las las tablas *reserva* y *reserva_vuelo_clase*. La fecha de la reserva se completará con la fecha actual al momento de realizar la reserva y la fecha de vencimiento será 15 días antes de la fecha de salida del vuelo reservado (el manejo de las reservas vencidas no será desarrollado en el proyecto). El estado de la reserva será calculado en función de la cantidad de reservas existentes al momento de realizar la reserva y la cantidad de asientos que brinda el vuelo en la clase a reservar. Si la cantidad de reservas es menor a la cantidad de asientos que brinda, el estado de la reserva será: *"confirmada"*. En caso contrario el estado de la reserva será: *"en espera"*.

Dado que una reserva involucra modificar dos tablas de la B.D., resulta conveniente que la misma sea implementada como un transacción para garantizar su atomicidad y evitar posibles inconsistencias en caso de fallas.

Además, como la base de datos puede ser accedida concurrentemente por varias aplicaciones debemos asegurarnos que otras reservas que se están realizando concurrentemente no accedan simultáneamente al mismo vuelo y clase, para evitar posibles inconsistencias. Por ejemplo: *supongamos que el vuelo V1 del 10-12-2019 tiene un solo asiento disponible en la clase turista y desde dos puntos de venta se intenta reservar este asiento. Ambos puntos de venta verifican que hay un asiento disponible y podrían realizar la reserva quedando la B.D. en un estado inconsistente.*

Para evitar este tipo de situaciones debemos controlar la concurrencia, permitiendo que solo una reserva logre acceder al vuelo y la clase para verificar los lugares disponibles y las demás reservas sobre ese vuelo y clase deban esperar. Para poder realizar este control, agregaremos a la B.D. una tabla *asientos_reservados* que mantiene la cantidad de asientos reservados para cada clase e instancia de vuelo (ver figura 1).



asientos_reservados (vuelo, fecha, clase, cantidad)

Figura 1: Relacion *asientos_reservados*

La tabla *asientos_reservados* se utilizará para calcular los lugares disponibles en una clase de un vuelo y deberá ser actualizada durante cada reserva. Si varias reservas intentan reservar un asiento en el mismo vuelo y clase, solo una logrará acceder y bloquear la fila correspondiente de la tabla *asientos_reservados* y las demás deberán esperar. Las reservas que se realicen sobre vuelos o clases diferentes se pueden realizar concurrentemente y no deben esperar unas por otras.

Ejercicios

1. Modifique la base de datos *vuelos* según lo planteado anteriormente. Deberá agregar una tabla llamada *asientos_reservados* que respete el esquema planteado en la figura 1. Se deberá respetar los nombres de los atributos, así como las restricciones de llaves primarias y foráneas.

   Se deberá entregar el archivo de texto (en formato digital, no impreso) "vuelos.sql" con las modificaciones planteadas y una carga inicial de datos que se ajuste a estas modificaciones. Verifique la B.D. *vuelos* modificada utilizando el programa `verificar2.jar` disponible en la página web de la materia en la sección: `Downloads>Proyectos`. **No se aceptarán proyectos que no pasen exitosamente la verificación realizada por dicho programa.**

2. Mediante *stored procedures* implemente transacciones para:

   a) realizar una reserva para un vuelo de ida

   b) realizar una reserva para un vuelo de ida y un vuelo de vuelta.

   Ambos *stored procedures* deberán devolver el resultado de la transacción, es decir, si la reserva se realizó con éxito o no, indicando el motivo: por ejemplo no hay lugares disponibles o alguno de los datos (el vuelo, pasajero o empleado) no existe en la base de datos.

   En el caso de tratarse de una reserva para un vuelo de ida y un vuelo de vuelta, la reserva se considerará exitosa solo si se logran reservar ambos vuelos. El número de reserva será uno sólo y estará asociado a ambos vuelos. **Note** que **no** se puede implementar la reserva de 2 vuelos de ida y vuelta invocando 2 veces al stored procedure del inciso 2*a* dado que se generarán dos reservas con números diferentes para cada vuelo.

   El estado de la reserva será *"en espera"* si alguno de los vuelos (ida o vuelta) tiene asientos disponibles pero la capacidad física esta colmada, esto es, la cantidad de asientos reservados en la clase reservada supera la cantidad de asientos que brinda el vuelo en esa clase.

   Importante: Dado que el campo *numero* de la tabla *reservas* es de tipo `AUTO INCREMENT`, deberá utilizar la función `LAST_INSERT_ID()` para recuperar el número de reserva generado automáticamente al insertar una nueva reserva dentro de cada transacción (ver `http://dev.mysql.com/doc/refman/8.0/en/stored-routines-last-insert-id.html`).

   Los *stored procedures* deberán definirse dentro de la base de datos *vuelos*. Las sentencias de creación de dichos procedimientos deberán estar incluidas en el archivo "vuelos.sql".

3. Extienda la aplicación implementada en Java para permitir realizar reservas a los vuelos que se muestran como resultado de una consulta sobre disponibilidad de vuelos (Proyecto 2, ejercicio 2 b)). La aplicación deberá permitir:

   - seleccionar un vuelo y una clase para reservar, o dos vuelos y una clase de cada vuelo en el caso de tratarse de un viaje de ida y vuelta.
   - ingresar el tipo y numero de documento de un pasajero existente en la B.D.
   - para el empleado asociado a la reserva se utilizará el legajo que se halla ingresado para acceder a la aplicación y poder realizar la misma.

Una vez que se tienen todos los datos la aplicación permitirá realizar la reserva invocando al *stored procedure* correspondiente definido en el ejercicio 2 y mostrando a continuación un mensaje que indique si la reserva se realizó con éxito o no y porque.

4. Implemente un trigger para cargar automáticamente las instancias de vuelo asociadas a una nueva salida de un vuelo programado. Dicho trigger deberá activarse cuando se inserta una nueva salida y cargará todas las instancias de vuelo asociadas a dicha salida, por el período de un año a partir de la fecha actual. El campo "estado" de todas las instancias cargadas deberá inicializarse al valor "a tiempo". Además deberá inicializar en 0 la cantidad de asientos reservados (figura 1) para cada clase correspondiente a la instancia de vuelo.

   Para la manipulación de las fechas se recomienda usar las funciones `DAYOFWEEK()` y `DATE_ADD()` (o `SELECT INTERVAL <expresion> DAY <fecha>`) (ver `https://dev.mysql.com/doc/refman/8.0/en/date-and-time-functions.html`)

   Se deberá entregar un archivo de texto llamado "**trigger.sql**" conteniendo la sentencias para crear el trigger dentro de la base de datos *vuelos*. No incluir la creación del trigger dentro del archivo "vuelos.sql" para evitar inconvenientes al cargar los datos de prueba.

Fechas y condiciones de entrega

- **Fecha límite de entrega: martes 29 de octubre de 2019** a través del curso **Moodle** de la materia y utilizando la tarea correspondiente habilitada para tal fin, se deberá subir un archivo comprimido (zip o rar) que contenga:
  - el archivo **"vuelos.sql"** extendido con lo solicitado en los ejercicio 1 y 2, y el archivo **"trigger.sql"** solicitado en el ejercicio 4
  - los archivos fuentes de la aplicación solicitada en el ejercicio 3 junto con el archivo JAR ejecutable correspondiente. **La aplicación deberá incluir las correcciones a todas las observaciones realizadas por la cátedra sobre la entrega del proyecto 2.**
- La entrega en *tiempo y forma* de este proyecto es *condición de cursado* de la materia.