Departamento de Cs. e Ingeniería de la Computación

Universidad Nacional del Sur

Bases de Datos

Segundo Cuatrimestre de 2019

Proyecto N$^{\underline{o}}$ 2

Implementación de un sistema con base de datos

Ejercicios

1. Implemente una base de datos en MySQL respetando el modelo relacional que acompaña este enunciado (ver apéndice A). El nombre de dicha base de datos deberá ser *"vuelos"* y los esquemas de las tablas deberán respetar los esquemas propuestos por dicho modelo relacional. Se deberán respetar los nombres de las relaciones y los atributos, así como las restricciones de llaves primarias y foráneas. **No se aceptarán bases de datos que no respeten estas convenciones (ver apéndice B.1)**.

   Además deberá crear los siguientes usuarios:

   - *admin*: Este usuario estará encargado de administrar la base de datos *"vuelos"* por lo tanto deberá tener acceso total sobre todas las tablas, con la opción de crear usuarios y otorgar privilegios sobre las mismas. Para no comprometer la seguridad se restringirá que el acceso de este usuario se realice solo desde la máquina local donde se encuentra el servidor MySQL. El password de este usuario deberá ser *admin*.
   - *empleado*: Este usuario estará destinado a las consultas y reservas de vuelos. Para las consultas necesitará acceso de lectura sobre todas las tablas de la b.d. *vuelos*. Para las reservas necesitará además privilegios para ingresar, modificar y borrar datos sobre las tablas *reservas*, *pasajeros* y *reserva_vuelo_clase* . Dado que la empresa cuenta con varios puestos de atención al público distribuidos en diferentes sucursales, este usuario deberá poder conectarse desde cualquier dominio. El password del *empleado* deberá ser *empleado*. **Importante:** Recuerde eliminar el usuario *vacío* (`drop user ''@localhost`) para poder conectarse con el usuario *empleado* desde localhost.
   - *cliente*: Este usuario está destinado a permitir un acceso público para satisfacer las consultas sobre vuelos de los clientes de la empresa, ya sean tanto personas como agencias de viajes. El *cliente* tendrá una visión restringida de la base de datos que solamente le permita ver la disponibilidad de vuelos ocultando la estructura de la base de datos. A tal efecto, se deberá crear una *vista* con el nombre *vuelos_disponibles* que contenga la siguiente información sobre cada instancia de vuelo:
     - Número de vuelo, modelo de avión, fecha, día, hora de salida, hora de llegada y tiempo estimado de vuelo (puede asumir que no hay vuelos que duren más de 24 hs.)
     - Código, nombre, ciudad, estado y país del aeropuerto tanto de salida como de llegada.
     - Precio del pasaje y cantidad de asientos disponibles en cada clase brindada por el vuelo. Para calcular el número de asientos disponibles deberá restar la cantidad total de reservas hechas para el vuelo y la clase, a la suma de la cantidad de asientos que brinda el vuelo en esa clase más el porcentaje asociado a la clase.

     El usuario *cliente* solo tendrá privilegio de lectura sobre la vista *vuelos_disponibles*, deberá poder conectarse desde cualquier dominio y su password deberá ser *cliente*.

   Se deberá entregar un archivo de texto con el nombre "vuelos.sql" (en formato digital, no impreso) con la secuencia de sentencias para la creación de la base de datos, las tablas, la vista,

los usuarios con nombre, password y privilegios correspondientes. Ademas deberá entregar un archivo de texto (en formato digital, no impreso) con el nombre *"datos.sql"* con una carga inicial de datos de prueba adecuados como para poder realizar consultas significativas sobre ellos, probar la vista y la aplicación (ejercicio 2).

2. Implemente una aplicación en el lenguaje Java que se comunique con la base de datos *"vuelos"* y provea las siguientes funcionalidades:

   a) **Realizar consultas a la base de datos ingresando sentencias SQL.**
   Esta parte del sistema estará disponible sólo para el administrador del sistema por lo cual se requerirá el password correspondiente de forma oculta (ver `JPasswordField`).

   En caso que el password sea incorrecto se deberá mostrar un mensaje de error adecuado. Si el password es el correcto, la interfaz de usuario deberá mostrar un área de texto (`JTextArea`) que permita introducir sentencias SQL arbitrarias (select, insert, delete, update, etc) mostrando el resultado (en el caso que corresponda) de la ejecución de dicha consulta en una tabla (`Jtable`). En caso que la consulta produzca un error, deberá mostrarse un mensaje explicando el error.

   Además deberá mostrar una lista (Por ejemplo `JList`) que contenga los nombres de todas las tablas presentes en la B.D. *"vuelos"*. Cuando se seleccione un ítem (nombre de una tabla) de esta lista, se mostrará en otra lista los nombres de todos los atributos de la tabla seleccionada (Ver sentencias `SHOW TABLES` y `DESCRIBE <nombre_tabla>` en el manual de MySQL).

   b) **Consultas sobre disponibilidad de vuelos.**
   La aplicación deberá permitir ingresar el número de legajo de un empleado y el password de forma oculta, controlando que el password se corresponda con el número de legajo ingresado. En caso de que el password sea incorrecto deberá mostrarse un mensaje adecuado y permitir ingresar el password nuevamente (para realizar este control la aplicación deberá conectarse con el servidor MySQL a través del usuario *empleado* definido en el ejercicio 1). Una vez ingresados un número de legajo y un password correctos, la interface deberá permitir seleccionar:

   - Ciudad origen y destino del vuelo
   - Si se desea consultar por:
     1) Vuelos para el viaje de ida solamente
     2) Vuelos para la ida y vuelos para la vuelta
   - Fecha deseada para el vuelo de ida
   - Fecha deseada para el vuelo de vuelta en caso que corresponda.

   Luego se deberá mostrar información asociada a todos los vuelos disponibles que satisfagan los datos proporcionados. Tanto para los vuelos de ida como para los vuelos de vuelta se deberá mostrar una tabla con los siguientes campos:

   - Número de vuelo
   - Aeropuerto salida
   - Hora salida
   - Aeropuerto llegada
   - Hora llegada
   - Modelo de avión
   - Tiempo estimado

   Cuando se seleccione un vuelo se deberá mostrar otra tabla con las clases disponibles en ese vuelo, la cantidad de asientos disponibles para reservar en cada clase y el precio del pasaje asociado a cada clase.

Nota: El formato de la fecha y hora recuperado desde JAVA a través de JDBC puede variar según el formato de fecha que esté configurado en Windows. **La aplicación entregada deberá funcionar correctamente independientemente de la configuración de fecha y hora de Windows** (ver apéndice B.2)

Fechas y condiciones de entrega

- **Fechas límite de entrega:**
  - **Ejercicio 1 - 17 de Septiembre de 2019**: a través del curso **Moodle** de la materia (`https://moodle.uns.edu.ar/moodle`) y utilizando la tarea correspondiente habilitada para tal fin, se deberán subir los archivos de texto *"banco.sql"* y *"datos.sql"* solicitados en el **ejercicio 1**
  - **Ejercicio 2 - 8 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: los archivos fuentes de la aplicación solicitada en el **ejercicio 2**, el archivo JAR ejecutable correspondiente y los archivos *"banco.sql"* y *"datos.sql"* solicitados en el ejercicio 1.
- **Comisiones:** Los proyectos deben realizarse en comisiones de *dos alumnos* cada una. Las comisiones deberán ser *las mismas* para todas las entregas.
- **Importante:** La entrega en *tiempo y forma* de este proyecto es *condición de cursado* de la materia.

A. Modelo Relacional de la base de datos *"vuelos"*

- **vuelos_programados**(numero, aeropuerto_salida, aeropuerto_llegada)
  numero es una cadena de caracteres; aeropuerto_salida y aeropuerto_llegada corresponden a codigos de aeropuertos

- **salidas**(vuelo, dia, hora_sale, hora_llega, modelo_avion)
  vuelo corresponde a un numero de vuelo programado; dia = {'Do','Lu','Ma','Mi','Ju','Vi','Sa'}; modelo_avion corresponde a un modelo de modelos_avion

- **instancias_vuelo**(vuelo, fecha, dia, estado)
  vuelo y dia corresponden a una salida; estado es una cadena de caracteres

- **aeropuertos**(codigo, nombre, telefono, direccion, pais, estado, ciudad)
  codigo, nombre, telefono y direccion son cadenas de caracteres; pais, estado y ciudad corresponden a una ubicación

- **ubicaciones**(pais, estado, ciudad, huso)
  Pais, estado y ciudad son cadenas de caracteres; huso es un entero en el intervalo [-12,12]

- **modelos_avion**(modelo, fabricante, cabinas, cant_asientos)
  modelo y fabricante son cadenas de caracteres; cabinas y cant_asientos son enteros positivos

- **clases**(nombre, porcentaje)
  nombre es una cadena de caracteres y porcentaje es un numero real en el intervalo [0, 0.99]

- **comodidades**(codigo, descripcion)
  codigo es un número natural y descripcion es una cadena de caracteres

- **pasajeros**(doc_tipo, doc_nro, apellido, nombre, direccion, telefono, nacionalidad)
  doc_nro es un numero natural; el resto de los atributos son cadenas de caracteres

- **empleados**(legajo, password, doc_tipo, doc_nro, apellido, nombre, direccion, telefono)
  doc_nro y legajo son numeros naturales, el resto de los atributos son cadenas de caracteres. El campo password debe ser una cadena de 32 caracteres, para poder almacenarlo de forma segura utilizando la función de hash `MD5` *provista por MySQL(ver sección B.3)*

- **reservas**(numero, fecha, vencimiento, estado, doc_tipo, doc_nro, legajo)
  numero es un natural; fecha y vencimiento son fechas; estado es una cadena de caracteres; doc_tipo, doc_numero corresponden a un pasajero; legajo corresponde a un empleado

- **brinda**(vuelo, dia, clase, precio, cant_asientos)
  vuelo y dia corresponden a una salida; clase corresponde al nombre de una clase; precio es un real positivo con 2 dígitos decimales y 5 dígitos en la parte entera; cant_asientos es un natural

- **posee**(clase, comodidad)
  clase corresponde al nombre de una clase y comodidad corresponde al codigo de una comodidad

- **reserva_vuelo_clase**(numero, vuelo, fecha_vuelo, clase)
  numero corresponde a un numero de reserva; vuelo y fecha_vuelo corresponden a una instancia de vuelo; clase corresponde al atributo nombre de clases

B. Consideraciones generales

B.1. Verificación de la Base de Datos

En el la sección DOWNLOADS/PROYECTOS de la página web, estará disponible para bajar un programa llamado *verificar*. Este programa realiza una verificación sobre la estructura de la base de datos *ya creada*, y muestra un listado con los errores (si los tuviera) que esta presenta con respecto al modelo relacional propuesto en el apendiceA . Este programa debe ejecutarse con el servidor de MySQL corriendo, una vez creada la base de datos. Se recomienda verificar su base de datos con este programa antes de empezar a desarrollar la aplicación. **No se aceptarán proyectos que no pasen correctamente la verificación realizada por este programa**.

B.2. Funciones de MySQL para el manejo de fechas y tiempo

MySQL provee funciones para manipular fechas y tiempo que resultan muy útiles para el desarrollo de este proyecto. En general estas funciones se pueden consultar al servidor mediante la sentencia SQL select y devuelven una tabla con una única celda que contiene el resultado (más información ver sección 12.7 de refman-8.0-en.a4.pdf `https://dev.mysql.com/doc/refman/8.0/en/date-and-time-functions.html` o sección 12.5 de refman-5.0-es.a4.pdf). A continuación se muestran ejemplos de algunas de las funciones disponibles.

Importante: los ejemplos que se muestran son el resultado de invocar a las funciones desde el cliente *mysql.exe*. El formato de la fecha y hora recuperado desde JAVA a través de JDBC puede variar según el formato de fecha que esté configurado en Windows. **La aplicación entregada deberá funcionar correctamente independientemente de la configuración de fecha y hora de Windows**. Para esto puede utilizar las funciones provistas en la clase *Fechas* contenida en el archivo "*Fechas.java*" de "*ejemplo java-MySQL.zip*" que estará disponible en la página web de la materia.

- ```
  mysql> SELECT CURDATE();
  +------------+
  | CURDATE()  |
  +------------+
  | 2005-09-27 |
  +------------+
  1 row in set (0.03 sec)
  ```

  Devuelve la fecha actual en la forma: año-mes-día

- ```
  mysql> SELECT CURTIME();
  +-----------+
  | CURTIME() |
  +-----------+
  | 13:09:37  |
  +-----------+
  1 row in set (0.00 sec)
  ```

  Devuelve la hora actual .

- ```
  mysql> SELECT NOW();
  +---------------------+
  | NOW()               |
  +---------------------+
  | 2005-09-27 13:09:45 |
  +---------------------+
  1 row in set (0.00 sec)
  ```

  Devuelve la fecha y hora actual .
  ```

- ```
  mysql> SELECT DAYOFWEEK('2005-09-26');
  +-------------------------+
  | DAYOFWEEK('2005-09-26') |
  +-------------------------+
  |                       2 |
  +-------------------------+
  1 row in set (0.02 sec)
  ```

  El número devuelto representa el día de la semana correspondiente a la fecha dada. 1 = domingo, 2 = lunes, 3 = martes, ... , 7 = sábado.

- ```
  mysql> SELECT TIMEDIFF('13:40:00', '08:05:00');
  +---------------------------------+
  | TIMEDIFF('13:40:00', '08:05:00') |
  +---------------------------------+
  | 05:35:00                        |
  +---------------------------------+
  1 row in set (0.00 sec)
  ```

  `TIMEDIFF(h1, h2)` devuelve la diferencia de tiempo entre dos horas `h1` y `h2`. `h1` debe ser mayor que `h2`, sino devolverá un número negativo.

  `TIMEDIFF` también se puede usar para calcular la diferencia de tiempo entre dos fechas:

  ```
  mysql> SELECT TIMEDIFF('2005-09-27 01:01:01','2005-09-26 23:59:59');
  +-----------------------------------------------------+
  | TIMEDIFF('2005-09-27 01:01:01','2005-09-26 23:59:59') |
  +-----------------------------------------------------+
  | 01:01:02                                            |
  +-----------------------------------------------------+
  1 row in set (0.00 sec)
  ```

- ```
  mysql> SELECT TIME_TO_SEC('1:00:00');
  +------------------------+
  | TIME_TO_SEC('1:00:00') |
  +------------------------+
  |                   3600 |
  +------------------------+
  1 row in set (0.00 sec)
  ```

  Devuelve la cantidad de segundos que representa el tiempo dado.

B.3. Funciones en MySQL para cifrado de datos

Supongamos que en una base de datos creamos la siguiente tabla para almacenar los usuarios junto con su contraseña o password, y así poder controlar el ingreso al sistema.

```
create table usuarios(
  usuario VARCHAR(30) not null,
  password CHAR(32),
  primary key (usuario)
);
```

Si las contraseñas se almacenan en texto plano y alguien logra acceder a la base de datos, podría recuperar las contraseñas de todos lo usuarios y acceder al sistema.
```

MySQL provee varias funciones para el cifrado de datos, que permiten almacenar este tipo de información sensible de manera segura (más información ver sección 12.13 del manual refman-8.0-en.a4.pdf `https://dev.mysql.com/doc/refman/8.0/en/encryption-functions.html` o sección 12.9.2 de refman-5.0-es.a4.pdf). Nosotros utilizaremos la función hash `md5`. Esta función toma como entrada una cadena de texto de cualquier longitud y devuelve una cadena de texto cifrada de 32 caracteres hexadecimales, utilizando el algoritmo MD5 (Message-Digest Algorithm 5). Lo interesante de este algoritmo es que su proceso es irreversible, es decir, a partir de una cadena cifrada no es posible obtener la cadena de texto original.

Por ejemplo, si quisiéramos almacenar un usuario 'u1' con password 'pw1' podríamos hacerlo de la siguiente forma:

```
insert into usuarios values('u1', md5('pw1'))
```

Luego, si consultamos la tabla usuarios podemos ver que el password se almacenó de forma cifrada:

```
mysql> select * from  usuarios;
+---------+----------------------------------+
| usuario | password                         |
+---------+----------------------------------+
| u1      | 6e6fdf956d04289354dcf1619e28fe77 |
+---------+----------------------------------+
```

por lo tanto, si alguien logra acceder a la la base de datos no podrá obtener el password original. Si desde una aplicación queremos validar el ingreso de un usuario al sistema, simplemente tomamos el pasword ingresado por el usuario, le aplicamos la función `md5` y comparamos el resultado con el valor almacenado en la base de datos. Por ejemplo:

```
mysql> select * from  usuarios where usuario='u1' and password=md5('pw1');
+---------+----------------------------------+
| usuario | password                         |
+---------+----------------------------------+
| u1      | 6e6fdf956d04289354dcf1619e28fe77 |
+---------+----------------------------------+
```

si el password introducido por el usuario '`u1`' es incorrecto (distinto de '`pw1`') la consulta anterior no devolverá ningún resultado y de esta forma podemos controlar la autenticidad del mismo.