Departamento de Cs. e Ingeniería de la Computación

Universidad Nacional del Sur

Bases de Datos

Segundo Cuatrimestre de 2020

Proyecto Nº 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 *"parquimetros"* 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 trabajos que no respeten estas convenciones (ver apéndice B.1)**.

   Además deberá crear los siguientes usuarios que definen diferentes tipos de acceso al Servidor MySQL:

   - *admin*: Este usuario se utilizará para administrar la base de datos *"parquimetros"* 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 sólo desde la máquina local donde se encuentra el servidor MySQL. El password de este usuario deberá ser: *admin*.
   - *venta*: Este usuario está destinado a permitir el acceso de la aplicación de venta de tarjetas y deberá tener privilegios *mínimos* para cargar una nueva tarjeta con un saldo inicial y de cualquier tipo. El password de este usuario deberá ser: *venta*.
   - *inspector*: Este usuario estará destinado a permitir el acceso de las unidades personales que utilizan los inspectores para controlar los estacionamientos. Con el objetivo de ocultar la estructura de la base de datos, el usuario *inspector* tendrá una visión restringida de la misma que le permita ver la patente de todos los autos estacionados en cada ubicación. A tal efecto, se deberá crear una *vista* con el nombre *estacionados* que contenga la siguiente información para cada ubicación:
     - calle y altura de la ubicación
     - patente de todos los autos que tienen un estacionamiento *abierto* en dicha ubicación.

     Un estacionamiento se considera *abierto* si tiene fecha y hora de entrada pero no tiene fecha y hora de salida (valores nulos).

     El usuario *inspector* deberá tener privilegios *mínimos* para:
     - validar número de legajo y password de un inspector (ver Ejercicio 2*b*),
     - Dado un parquímetro, obtener las patentes de los automóviles con un estacionamiento abierto en la ubicación del parquímetro,
     - cargar multas, y
     - registrar accesos a parquímetros.

     El password de este usuario deberá ser: *inspector*.

   Se **deberá entregar** un archivo de texto (en formato digital, no impreso) con el nombre ***"parquimetros.sql"*** 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. Además 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. Diseñe una aplicación en el lenguaje Java que se comunique con la base de datos *"parquimetros"* 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 (password=*******, 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. *"parquimetros "*. 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) **Simular la unidad personal de un inspector**.
   Cada inspector cuenta con una unidad personal que utiliza para controlar las ubicaciones que tiene asignadas. Las unidades personales permiten cargar un número de legajo y password del inspector que las utiliza. De esta forma cuando una unidad se conecta a un parquímetro se puede identificar el inspector que la esta utilizando.

   La aplicación deberá permitir ingresar el número de legajo de un inspector y el password de forma oculta (password=*******) controlando que el password se corresponda con el número de legajo ingresado (para almacenar y recuperar los password de forma segura utilice la función hash `md5` de MySQL, ver sección B.3). En caso de que el legajo o el password sean incorrectos deberá mostrarse un mensaje adecuado y permitir ingresarlos nuevamente.

   Para controlar una ubicación, el inspector comienza recorriendo la ubicación a pie y carga en su unidad la patente de cada auto estacionado. Cuando termina de recorrer la ubicación conecta su unidad a un parquímetro para comparar las patentes cargadas con los estacionamientos abiertos en la ubicación donde se encuentra el parquímetro. En este momento se generan automáticamente las multas para aquellos autos que no tienen un estacionamiento abierto en dicha ubicación.

   Una vez ingresados un número de legajo y un password correctos, la aplicación deberá permitir lo siguiente:

   1) Ingresar una lista patentes.
   2) Seleccionar una ubicación y un parquímetro de la misma.

   Luego se simulará la conexión de la unidad al parquímetro seleccionado:

   - Registrando el acceso del inspector al parquímetro,
   - Generando las multas correspondientes, y
   - Mostrando un listado de las multas labradas con la siguiente información: `número de multa, fecha, hora, calle, altura, patente del auto y legajo del inspector.`

Se deberá controlar que el inspector tenga asociada la ubicación correspondiente al parquímetro seleccionado, para el dia y hora en que se realiza la conexión. De no ser así, no se generarán las multas y se deberá mostrar un mensaje indicando que el inspector no esta autorizado para labrar multas en esa ubicación. Considere que el turno mañana es de 8 a 13:59 hs. y el turno tarde es de 14 a 20 hs.

Fechas y condiciones de entrega

- **Fechas límite de entrega:**
  - **Ejercicio 1 - 9 de octubre de 2020**: a través del curso **Moodle** de la materia y utilizando la tarea correspondiente habilitada para tal fin, se deberán subir los archivos de texto *"parquimetros.sql"* y *"datos.sql"* solicitados en el **ejercicio 1**
  - **Ejercicio 2 - 30 de Octubre de 2020**: 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 (proyecto de eclipse importable) solicitada en el **ejercicio 2**, el archivo JAR ejecutable correspondiente y los archivos *"parquimetros.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. Para las entregas a través de las tareas de Moodle, **solo es necesario que uno de los integrantes de cada comisión suba los archivos solicitados.**
- **Importante:** La entrega en *tiempo y forma* de este proyecto es *condición de cursado* de la materia.

A. Modelo E-R y Modelo Relacional

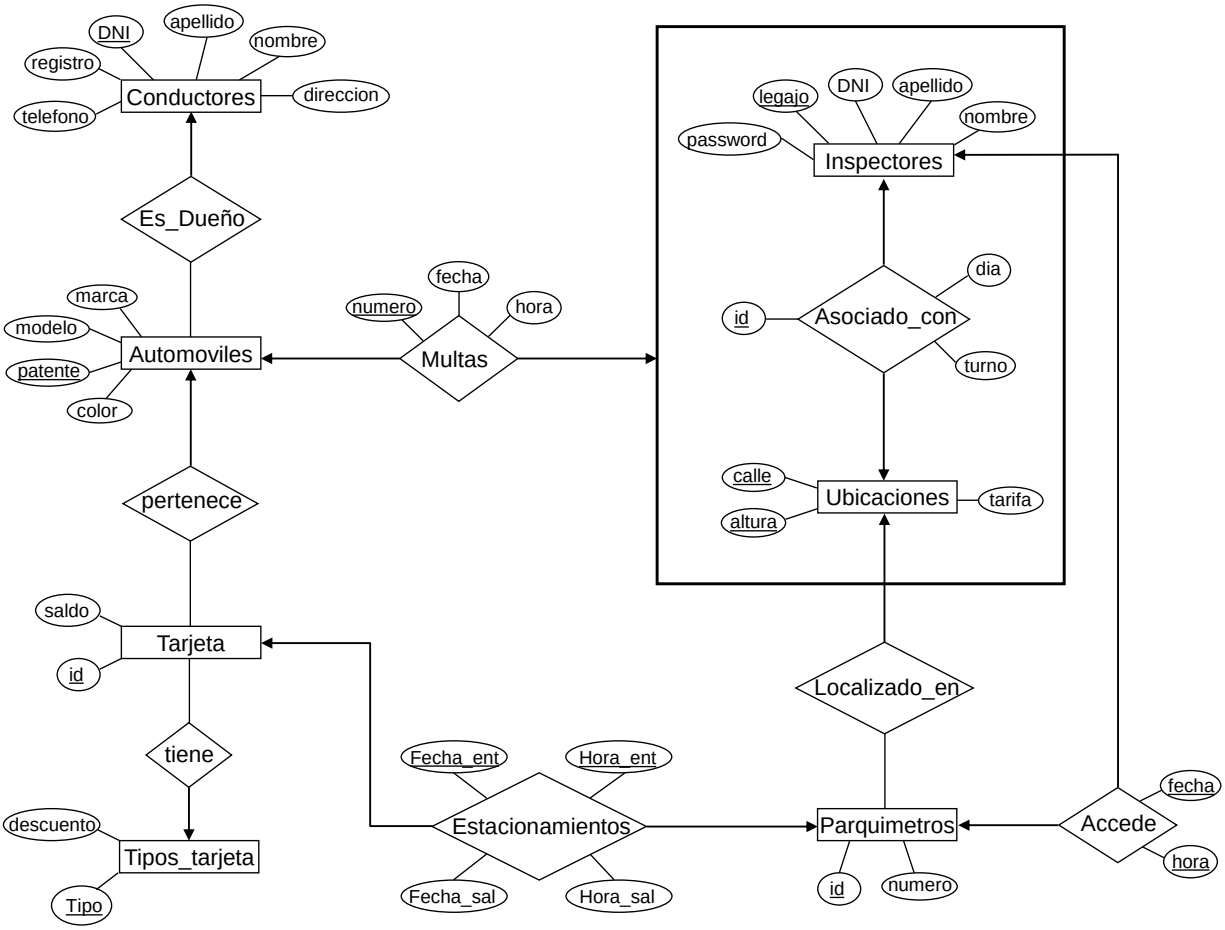

Figura 1: Modelo Entidad-Relación del sistema de parquímetros

- **Conductores** (dni, nombre, apellido, direccion, telefono, registro)
  dni y registro son números naturales; nombre, apellido, dirección y teléfono son cadenas de caracteres.

- **Automoviles**(patente, marca, modelo, color, dni)
  patente es una cadena de 6 caracteres; marca, modelo y color son cadenas de caracteres; dni corresponde al dni de un conductor.

- **Tipos_tarjeta**(tipo, descuento)
  tipo es una cadena de caracteres; descuento es un real con dos dígitos decimales en el intervalo $[0, 1]$ *(por ejemplo* 0,15 *corresponde a un* 15 % *de descuento)*

- **Tarjeta**(id_tarjeta, saldo, tipo, patente)
  id_tarjeta es un número natural; saldo es un real con tres dígitos enteros y dos decimales; tipo corresponde a un tipo de tarjeta; patente corresponde a la patente de un automóvil.

- **Inspectores**(legajo, dni, nombre, apellido, password)
  legajo y dni son números naturales; nombre, apellido 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)*

- **Ubicaciones**(calle, altura, tarifa)
  calle es una cadena de caracteres; altura es un número natural; tarifa es un real positivo con tres dígitos enteros y dos decimales.

- **Parquimetros**(id_parq, numero, calle, altura)
  id_parq y numero son números naturales; calle y altura corresponden a una ubicación.

- **Estacionamientos** (id_tarjeta, id_parq, fecha_ent, hora_ent,fecha_sal, hora_sal)
  id_tarjeta corresponde al identificador de una tarjeta; id_parq corresponde al identificador de un parquímetro.

- **Accede**(legajo, id_parq, fecha, hora)
  legajo corresponde al legajo de un inspector; id_parq corresponde al identificador de un parquímetro.

- **Asociado_con**(id_asociado_con, legajo, calle, altura, dia, turno)
  id_asociado_con es un numero natural; legajo corresponde al legajo de un inspector; calle y altura corresponden a una ubicación; dia es una cadena de dos caracteres (Lu,Ma,...); turno es una cadena con un caracter (M por turno mañana ó T por turno tarde).

- **Multa**(numero, fecha, hora, patente, id_asociado_con)
  numero es un número natural; patente corresponde a la patente de un automóvil; id_asociado_con corresponde a la llave de la relación asociado_con.

B. Consideraciones generales

B.1. Verificación de la Base de Datos

En la sección del proyecto 2 del curso Moodle, 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 apendice A . 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 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 el sistema operativo. **La aplicación entregada deberá funcionar correctamente independientemente de la configuración de fecha y hora del sistema operativo**. Para manipular diferentes formatos de fechas puede utilizar las funciones provistas en la clase *Fechas* contenida en el archivo "*Fechas.java*" de "*ejemplo java-MySQL.zip*" que estará disponible en el curso moodle de la materia.

- ```
  mysql> SELECT CURDATE();
  +------------+
  | CURDATE()  |
  +------------+
  | 2020-09-26 |
  +------------+
  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()               |
  +---------------------+
  | 2020-09-26 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 12.14 del manual refman-8.0-en.a4.pdf 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.