Cuarto trabajo de bases de datos 2 (20%)

En *blockchain* uno de los temas principales es la **visualización de datos** con el fin de identificar tendencias y patrones. Cientos de herramientas se dedican a este tema...

En esta parte del trabajo se visualizarán algunos datos de un **día** de trabajo de Ethereum.
Considere como ejemplo los archivos de bloques y transacciones enviados (le serán enviados o compartidos en el trascurso de hoy 23 de octubre).

Observe los campos (columnas) mostrados en cada uno de los archivos.
Nota: Estos archivos son de **muestra**, para las pruebas se trabajará con archivos de un día **con la misma estructura** (con los **mismas** columnas y tipos de datos) pero con datos diferentes.

Punto 1. (10%) Cargue los datos de ambos archivos en Oracle. Para la carga puede usar la herramienta que desee, se recomienda el SQL*Loader. Todos los datos de cada archivo (así algunos de ellos no se terminen usando para las visualizaciones...) deben quedar cada uno (cada archivo) en una tabla, es decir, una tabla de bloques (llamada, por ejemplo, BLOQUE) y una tabla de transacciones (llamada, por ejemplo, TRANSACCION). Defina los tipos de datos adecuados para cargar cada columna.
Pregunta frecuente: ¿Qué nombre y tipo de datos se deben usar para las columnas de las tablas?
Rta. Usted elige, lo que importa es que todos los datos de los archivos queden cargados.

Punto 2 (40%) Luego de cargar los datos de los archivos en la base de datos. Haga una interfaz en Java que se conecte a la base de datos y haga lo siguiente:
Primero se va a trabajar con los datos de las transacciones.

La interfaz solicita los siguientes datos:
Se va a trabajar en un marco de 100 X 100.

- Hora y minuto inicial (ejemplo: 19:08)
- Hora y minuto final (ejemplo: 21:19)
- Tamaño del lado de la cuadrícula (se aceptan solo números múltiplos de 5. Ejemplo, 25).
  Nota: Puede ocurrir que la última cuadrícula de cada fila (y columna) quede de tamaño menor, si la división del lado no es exacta con respecto al tamaño del lado marco (100).

Se van a requerir una escala de colores para el número de transacciones.

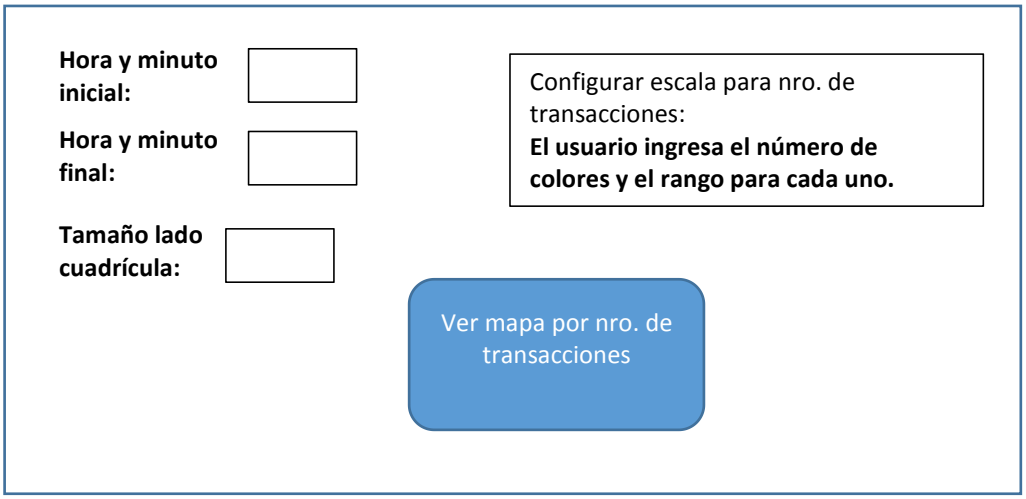
Para **la** escala de colores, en la interfaz se pide lo siguiente (ver ejemplo más abajo):

- Número de colores: un valor entre 1 y 10
- Rango para cada color.

La interfaz tiene un botón llamado:

- Ver mapa por nro. de transacciones

A continuación se muestra un bosquejo de la interfaz.
Pregunta frecuente: ¿Se califica la “belleza” de las interfaces?
Rta. No.

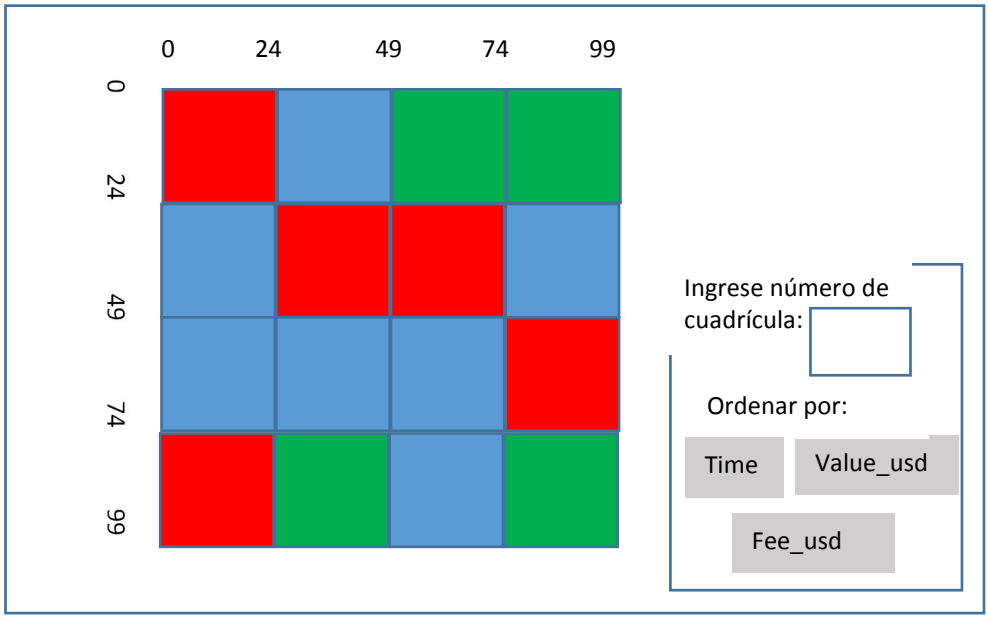


El programa debe entonces graficar lo siguiente cuando el usuario **presiona** el botón Ver mapa por nro. de transacciones.
Vamos a suponer para este ejemplo **que el usuario definió** la escala de colores para el número de transacciones así:
3 colores (usted elige los colores, lo importante es que sean 3 colores diferentes, por ejemplo, azul, verde y rojo y que a cada uno se le asigne un rango (los límites de cada rango los ingresa el usuario)).

- Rango para el color rojo: de 0 a 4
- Rango para el color azul: de 5 a 11
- Rango para el color verde: de 12 a infinito

Por supuesto, los rangos ingresados por el usuario deben ser excluyentes y el fin de uno es el inicio del otro + una unidad. El primer rango inicia en cero y el último termina en infinito.

Al presionar el botón Ver mapa por nro. de transacciones se debe graficar lo siguiente. Suponiendo para este ejemplo que el usuario ingresó un lado de cuadrícula = **25**.



Cada cuadrícula se muestra del color correspondiente según el número de transacciones que hayan sido emitidas desde la cuadrícula correspondiente (columnas x y y en el archivo de transacciones)

durante el rango horario dado (para este ejemplo, el rango horario es [19:08, 21:19]). Así, para este ejemplo, una cuadrícula es roja si emitieron desde ella entre 0 y 4 transacciones.
Además, al interior de cada cuadrícula se debe mostrar el número total de transacciones emitidas desde dicha cuadrícula (para el ejemplo, si es roja, será un número entre 0 y 4 para este ejemplo en particular), la sumatoria del valor del campo value_usd y la sumatoria del valor del campo fee_usd.

Cada cuadrícula debe **aparecer** enumerada (en el ejemplo anterior como hay 16 cuadrículas, entonces van enumeradas del 1 al 16, usted define el orden de enumeración). Cuando el usuario ingrese un número de cuadrícula (en el cuadro donde dice "Ingrese número de cuadrícula"), se deben mostrar en pantalla, en el formato que usted desee, los siguientes datos de **cada una** de las transacciones pertenecientes a dicha cuadrícula:

block_id, x, y, sender, recipient, value_usd, fee_usd y time.

Nota: Para mostrar las transacciones de una cuadrícula se le deben dar tres opciones al usuario: ordenarlas ascentemente por value_usd, por fee_usd o por time (ver los botones grises en la figura anterior).

Pregunta frecuente:
Puede ocurrir que una transacción haya sido emitida desde un punto (x, y) que esté en el borde de dos cuadrículas o en una esquina compartida, por ejemplo, por 4 cuadrículas, ¿qué hacer en este caso?
Rta. Si esto ocurre, asigne la transacción a alguna de las cuadrículas compartidas (pero solo a una de ellas).

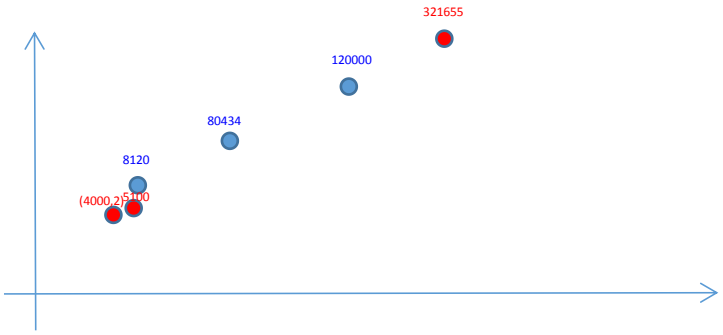
Nota: Recuerde que Java grafica los ejes "al revés". Puede dejarlos así. Puede aumentar la escala multiplicando por algún factor para facilitar la visualización.

Punto 3. (10%) Haga una interfaz que haga lo siguiente:

En esta interfaz el usuario ingresa el código de dos *miners*, una hora y minuto inicial y una hora y minuto final.

Hay un botón llamado Graficar comparación creciente *value_usd*. El programa debe entonces hacer lo siguiente:
Se deben buscar todos los bloques que los dos *miners* han ganado durante el rango horario ingresado (en la tabla BLOQUE, ver el atributo *time* de esa tabla). A continuación se deben obtener todos los *value_ usd* > 0 de todas las transacciones asociados con los bloques de cada uno (están en la tabla TRANSACCION) y debe aparecer en pantalla una gráfica como la siguiente.
Ejemplo.
Supongamos que se ingresa el primer *miner* con código = 0x1e8839fead6924ad004c2560e90804164900ccc y el segundo *miner* con código = 0x1e8877fead6924ad004c2560e90804164999ddd.
Supongamos que el primer *miner* tiene estos valores (*value_usd*): 8120, 80434 y 120000.
Supongamos que el segundo *miner* tiene estos valores (*value_usd*): 4000, 4000, 5100 y 321655.

En el eje X se manejan los *value_usd* y en el eje Y la altura de dichos valores (*value_usd*).



En caso de valores (*value_usd*) repetidos para un mismo *miner* (ejemplo, para el círculo rojo con valor 4000) imprima sin repetidos, pero indique cuantos hay repetidos como se muestra en la figura: (4000, 2).

Pregunta frecuente:
¿Que ocurre si uno o dos de los *miners* ingresados no existe?
Rta. No grafique.

La siguiente parte del trabajo es sobre Multichain, ha sido preparada por el monitor Daniel Sánchez, a quien expreso mis agradecimientos.

Punto 4 (5%)

Mediante comandos, genere una *blockchain* llamada **bdchain**. Adicionalmente, genere un *asset* llamado **bdcoin** (con 1.000.000 unidades en existencia y divisible en 100 partes) en la única dirección que tiene **bdchain**. Toda esta información será usada en el resto del trabajo.

Punto 5 (17.5%)

Haga una ventana en Java con dos botones: “iniciar sesión” y “registrarse”. La ventana debe ser lo primero que se muestre al ejecutar el programa y cada botón debe redirigir al usuario al formulario correspondiente.

El formulario de registro debe permitirle a un usuario registrarse en el sistema. Se le debe pedir al usuario ingresar un nombre de usuario (clave primaria) y una contraseña. Después de obtener estos datos el programa de Java debe generar internamente una dirección en **bdchain**, y esta se le asociará al usuario recién creado; el programa también debe brindarle a esta dirección permisos para conectarse a **bdchain**, enviar activos y recibir activos. Por aparte, en Oracle debe tener una tabla llamada **usuario** que contenga tres columnas:

- **nombre_usuario:** El nombre de usuario (clave primaria, máximo 20 caracteres).
- **contraseña:** La contraseña del usuario (máximo 20 caracteres). **NOTA: no es recomendable almacenar la contraseña en texto directamente en una aplicación real, pero para propósitos del trabajo es suficiente.**
- **direccion:** La dirección del usuario en **bdchain** (los caracteres máximos los deben deducir a partir de cómo están formadas las direcciones).

Tras obtener los datos del usuario y generar la dirección, el formulario debe guardar los datos creados en la tabla y redirigir al usuario a la ventana original.

Previamente, debe insertar en la tabla un usuario con nombre de usuario “admin”, contraseña de su elección y la dirección original de **bdchain**.

El formulario de inicio de sesión debe permitirle al usuario iniciar sesión. Se le debe pedir al usuario ingresar su nombre de usuario y contraseña. Si el nombre de usuario no existe o la contraseña es incorrecta, se debe mostrar una ventana con un mensaje de error. Si los datos son correctos, se debe redirigir al usuario a la ventana del punto 3.

Punto 6 (17.5)

Haga una ventana con tres botones: “consultar saldo”, “pagar” y “cerrar sesión”. Cada botón debe redirigir al usuario al formulario correspondiente.

El formulario de consulta de saldo simplemente debe mostrar el saldo disponible del usuario, es decir, la cantidad de **bdcoin** que tiene la dirección del usuario.

El formulario de pago debe permitirle a un usuario pagarle a otro. Se le debe pedir al usuario el nombre de usuario del usuario a quién se le quiere pagar y la cantidad que se le quiere pagar. Luego el usuario debe poder presionar el botón “pagar”; al presionarlo, el programa debe enviar la cantidad de **bdcoin** especificada por el usuario al receptor especificado.

Al presionar el botón de “cerrar sesión”, se debe cerrar la sesión actual y se debe redirigir al usuario a la ventana inicial del punto 2.

Notas adicionales:

- Sus soluciones deben funcionar para cualquier cantidad de filas que tengan las tablas.
- Puede usar todas las estructuras de datos y todas las tablas auxiliares que desee.
- Para entregar por email a *fjmoreno@unal.edu.co*, el lunes 9 de noviembre **hasta las 11 am**. Solo se califican trabajos enviados a ese correo.
- **No se reciben trabajos en hora posterior**. No se reciben versiones “mejoradas”. No se califican trabajos enviados “por accidente” a otros correos.
- **Se debe incluir un informe donde se describa cómo se solucionó cada punto.** Este informe hace parte de la calificación del trabajo (máximo 5 hojas). **No enviar los datos de prueba que usted usó para probar sus códigos.**
- Grupos de **tres** personas.
- Los trabajos deben ser independientes entre los grupos. Trabajos copiados **así sea en un SOLO punto** se califican con 0 (cero) en su totalidad para todos los integrantes. Las soluciones presentadas deben ser originales. El trabajo debe ser desarrollado por los integrantes del grupo no por personas ajenas a él.
- El monitor les puede ayudar con aspectos técnicos pero su función **no** es hacerles la práctica **ni está autorizado** para **cambiar las condiciones del trabajo**. Sin embargo, en este trabajo él si tiene la potestad de aclarar dudas sobre las condiciones de la parte de Multichain.
- **Si trabaja con otras herramientas**, así su trabajo funcione y sea “espectacular”, el trabajo **NO** será calificado.
- **No** se califica la “belleza” de los formularios, se califica la funcionalidad.
- Cualquier duda consultarla con el profesor.