TRABAJO PRÁCTICO ESPECIAL

El Trabajo Práctico Especial (TPE) consiste en la resolución de un conjunto de controles y servicios sobre una base de datos reducida de un Exchange de Criptomonedas según las consignas que se indican en la sección “Pautas de Desarrollo”. El esquema inicial es provisto por la cátedra.

Consideraciones Importantes:

- El TPE se resolverá en grupos (de dos o tres integrantes), que deberán **inscribirse previamente**, a través del formulario correspondiente, hasta el **28/09/2020**.
- Se recomienda **leer el enunciado completo** del trabajo antes de resolverlo.
- Se deberán **respetar las Pautas de entrega**, las **consignas en cada parte del enunciado** y las de nomenclatura indicada en la sección **Reglas de nombres** para todos los objetos del esquema de la base de datos que cada grupo entregue.
  Importante: se descontarán hasta 2 puntos de la nota final por no respetar las pautas.
- El código SQL entregado debe estar **optimizado**, es decir utilizando estrictamente los atributos, tablas y/u operaciones necesarias, de acuerdo a lo requerido.
- Los scripts entregados se probarán y testearán en el DBMS que se usa durante el curso (PostreSQL), vía web utilizando el *PhpPgAdmin* o mediante la aplicación de escritorio *DataGrip (de JetBrains)*. Cada grupo debe **asegurarse que los scripts ejecuten completos sin errores, en el orden estipulado, para que la cátedra pueda proceder a la corrección del TPE**.

Entrega del TPE: El trabajo podrá entregarse hasta el día **9/11/20.** Se deberá entregar a través de la plataforma Moodle, según se especifique oportunamente. En la misma tarea, la cátedra confirmará el día siguiente a la entrega si los scripts corrieron sin errores o, de haber advertido algún inconveniente en su ejecución, se informará en dicha respuesta el primer error encontrado.

Re-entrega del TPE o entrega fuera de término: En caso de detectar errores en la entrega original se aceptará que el grupo re-entregue **-por única vez-** los scripts corregidos hasta el **11/11/20**. Para esto, se deberá realizar una re-entrega en la plataforma Moodle, según se especifique oportunamente. Se descontarán 2 (dos) puntos en la nota final del TPE, la misma pauta se aplicará para quienes entreguen fuera de término. En la misma tarea se confirmará el día siguiente a la re-entrega, si los scripts corrieron sin errores o, de haber advertido algún inconveniente en su ejecución, se informará en dicha respuesta el primer error encontrado. En caso de que la re-entrega aún tuviera algún error de ejecución, el trabajo NO será evaluado.

Introducción:

El trabajo especial consiste en la implementación de la base de datos reducida de un Exchange de Criptomonedas, al estilo de Liquid, Coineal, Conbase, Binance, etc.

El sitio maneja usuarios, cada uno de los cuales tiene una billetera electrónica para cada una de las monedas que maneja el sitio (inicialmente con saldo 0). Se deben manejar todos los Movimientos sobre cada moneda, tanto las entradas como las salidas y se debe llevar el saldo, el cual nunca podrá ser negativo. Cada vez que se realiza un movimiento de Entrada o de Salida, el sitio se queda con el 0.5 % de la cantidad de ese movimiento en concepto de comisión (fee).

Cada Movimiento tiene un tipo asociado. Las entradas y las salidas pueden resultar de la ejecución de órdenes o por Depósitos o Extracciones. Todo Depósito o Extracción hace referencia al número de bloque (blockchain) y una dirección de origen o destino según corresponda. Este número nunca puede hacer referencia a uno menor cronológicamente para una operación de la misma moneda de cualquier usuario (o sea, si hay una operación a las 10:20 en BTC y hace referencia al bloque 12.450, entonces ninguna otra operación en BTC luego de las 10.20 puede hacer referencia a un bloque menor que el 12.450).

Una vez que los fondos se encuentren en la billetera, el usuario los podrá utilizar en la sección de Exchange o Trading (esta sección es donde se pueden intercambiar una moneda por otra a través de órdenes de compra/venta en un mercado determinado, el cual fija la relación entre dos monedas a la vez). Los fondos asignados en órdenes para Trading no podrán ser retirados.

El sitio maneja mercados, siendo cada mercado una asociación entre dos monedas en las cuales se ejecutan órdenes de compra y venta.

Dentro de cada mercado los usuarios colocan órdenes, estas pueden ser de dos tipos: de Compra y de Venta.

A la derecha se encuentra una imagen que muestra las órdenes de compra (las líneas en verde) y las de venta (las líneas en rojo). En particular se muestra el mercado BTC/USDT, que básicamente es el mercado de compra y venta de Bitcoin en dólares estadounidenses.

Cada una de estas órdenes involucra un precio al cual comprar o vender y una cantidad. Como se puede apreciar, en la primer orden de venta se venden 0.008004 BTC (bitcoin de ahora en más) a 11342 dólares, en la siguiente 0.000430 BTC a 11343.85 dólares, y así sucesivamente. Para la compra es igual, únicamente que se ordena en forma ascendente.

Una orden de compra/venta está compuesta por varias órdenes de venta/compra

respectivamente. Entonces suponiendo que uno tuviera 4000 dolares y quisiera comprar Bitcoin sin importar el precio, uno colocaría una orden de compra por 4000 y el sistema lo que haría es ir tomando una a una las órdenes de venta hasta completar el pedido de 4000, en este caso en particular, tomaría la primer orden de venta, luego la segunda, la tercera y parte de la cuarta. Esta última (la de 0.21 a 11370) quedaría incompleta, ya que sólo se necesitaría una parte de dicha orden para completar los 4000 que uno quiere comprar.

Esto funciona así en la realidad, pero **para el caso de este trabajo Práctico Especial, NO se considerará el caso de órdenes incompletas, sino que las órdenes a tomar deberán ser completas; es decir, que puede quedar un remanente pero no quedar una orden incompleta.**

Cada orden tiene un atributo que se llama estado, en el mismo se irán reflejando los distintos estados por los que pasa la orden (por ejemplo, pasará de Nueva a Cumplida).

Las órdenes de compra y venta tienen 3 modalidades: Limit, Market y Stop Loss

Market: Pueden ser de Compra o Venta. Básicamente el usuario especifica el monto a invertir en compra o venta, en el caso en particular que estamos viendo (la imagen anterior), si es compra serán dólares y si es venta serán BTC y de acuerdo a las órdenes más cercanas en monto ésta se va ejecutando hasta que se agoten los fondos asignados a la misma.

Limit: Puede ser de Compra o Venta, específica básicamente una oferta de una cantidad de X moneda (correspondiente al mercado en el que se está operando) a un precio Y. Si existe alguna orden que haga matching con la misma (es decir que el precio sea el mismo), el sistema automáticamente la ejecutará.
Recordar que en la realidad, la orden puede ser completada al 100 % o no. En caso de que no sea completada al 100 %, el resto de la orden quedará incompleta y seguirá con los parámetros iniciales de la orden original. Tener en cuenta que este caso refleja la realidad pero NO debe ser tenido en cuenta a la hora de la realización de este trabajo.

Por ejemplo, siguiendo en el mercado BTC/USDT, un usuario coloca una orden para vender 1 BTC a 11,750.00 USDT. Si otro usuario decide comprar 0.35 BTC a ese precio, entonces el sistema le descontará al usuario comprador 4,112.5 USDT de su billetera y quedará una orden en el mercado por 0.65 BTC a 11,750.00 USDT.

Stop Loss: Este tipo de orden se usa para colocar una orden de Venta en el caso de que se cumplan ciertos parámetros.
Por ejemplo, el usuario tiene 1 BTC y el precio actual es de 11,500 USDT y, como no puede estar monitoreando el mercado las 24 horas, coloca una Orden Stop Loss para que sea el sistema quien monitoree permanentemente el valor del mercado y, si el BTC baja de 10,500 automáticamente coloque una orden para vender a dicho precio.

Precio de Mercado:

El precio de mercado se calculará como el promedio entre el 20 % superior e inferior de la cantidad de la moneda asociada a las órdenes de compra y venta respectivamente, ponderadas por la cantidad y ordenadas por el precio. A continuación se muestra un ejemplo para ilustrar el mecanismo.

Consideremos que estamos en el mercado BTC/USDT. La sección en rojo son las órdenes de Venta y las verdes son las órdenes de Compra.

Supongamos que para la Venta (no se han sumado, se supone) se tiene 4 BTC (sumatoria de la cantidad de BTC de todas las órdenes de venta) y se tienen 5.5 BTC para la

compra (sumatoria de la cantidad de BTC a la compra). El 20 % de 4 es 0.8 y el 20 % de 5.5 es 1.1

Entonces se deberá hacer por un lado el promedio (de cantidad por precio) de las ventas hasta llegar a la cantidad de 0.8 y por otro el promedio de las compras hasta llegar a 1.1 y luego se obtiene el promedio simple entre estos dos valores.

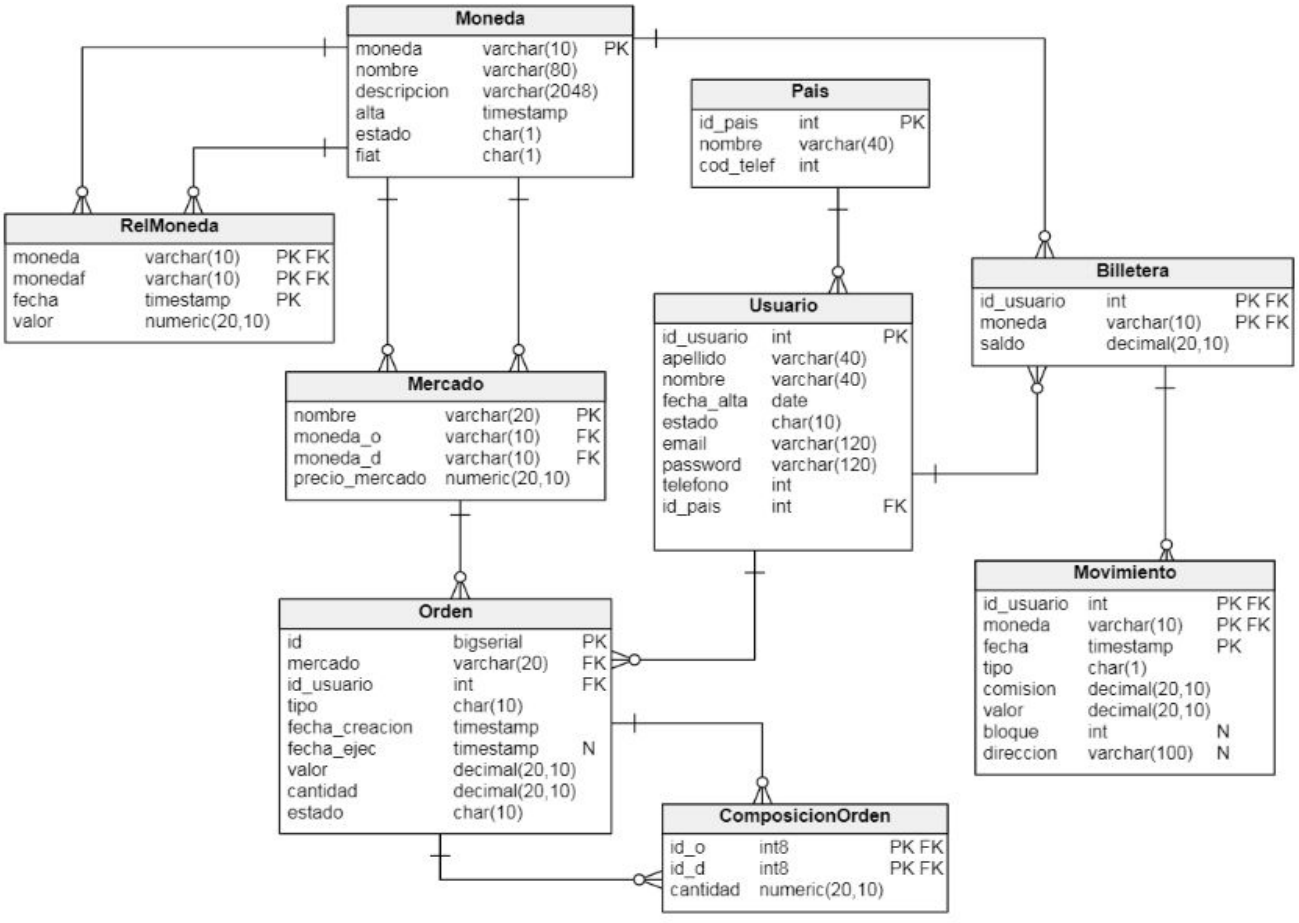
Moneda (Criptomoneda, criptomoneda estable y moneda Fiat)

El sitio maneja una cantidad de monedas, cada una de ellas puede ser FIAT (una moneda fiduciaria como dólar, Euro, Yen, es decir una moneda que está controlada por un gobierno) o no.

Si la moneda es estable (StableCoin) está atada o vinculada con una moneda fiat a una valor X. Sólo las monedas estables están vinculadas a una moneda fiat.

Por ejemplo, USDT (Theter US) es una cripto moneda que está atada al valor del dólar a una relación de 1; otro ejemplo es EUR, que es otra criptomoneda que está atada al Euro a relación de 1 (es decir, un USDT = 1 dólar estadounidense y un EUR = 1 Euro). Cabe aclarar que esta relación no es siempre exactamente 1, puede variar en el tiempo dependiendo de cómo esté respaldada dicha moneda, pero su variación en ínfima.

El modelo para el trabajo especial es el siguiente:



PAUTAS DE DESARROLLO

A partir del anterior esquema de la BD se requiere **resolver los ítems que se indican a continuación**.

A. AJUSTE DEL ESQUEMA

- Cree su esquema a partir del archivo XML de Vertabelo provisto por la cátedra.
- Ajuste dicho esquema según las pautas de nomenclatura indicada en la sección Reglas de Nombres.
- Provea un script de generación del esquema denominado ***GXX_1_Creacion.sql*** que cree todas las tablas con sus restricciones de nulidad, primary key y foreign key, para las tablas del diagrama proporcionado por la cátedra.
- Provea un script de inicialización de la base de datos. denominado ***GXX_2_Carga.sql*** con las siguientes características:
    - Debe tener por lo menos 10 usuarios.
    - El Sitio debe manejar 20 monedas y al menos tiene que tener 3 monedas Fiat y 5 estables. Entre las monedas usar:
        - BTC, ETH y las que Uds. quieran como criptomonedas.
        - USDT, DAI, USDC como monedas estables
        - Dólares, Euros y Yenes como monedas FIAT
    - Crear un mercado de cada moneda contra las estables
    - Crear un mercado de cada criptomoneda contra el BTC
    - Inicializar las órdenes con al menos 100 filas

B. ELABORACIÓN DE RESTRICCIONES Y REGLAS DEL NEGOCIO

Se deben realizar los siguientes controles en el esquema de datos:

1) Controlar que los números de bloque sean consecutivos para los movimientos de Entrada y Salida del Blockchain por moneda y fecha.
2) Controlar que no se pueda colocar una orden si no hay fondos suficientes.
3) No se pueden hacer retiros de una moneda, si esos fondos están en órdenes activas.
4) La opcionalidad del Número de Bloque en Movimiento, debe coincidir con la opcionalidad de Dirección, es decir que ambos son nulos o ambos no lo son.

Para ello:

- Escriba la restricción de la manera que considere más apropiada en *SQL estándar declarativo*, indicando su tipo y características correspondientes (Nota: NO se tendrán en cuenta las tipificaciones correctas pero con implementaciones inadecuadas).

- Implemente la restricción en PostgreSQL de la forma más adecuada, según las posibilidades que ofrece el DBMS (*incorpore la solución al script **GXX_3_Soluciones.sql** y la explicación correspondiente en el informe*)

- Provea una sentencia de modificación, borrado y/o inserción según corresponda, que promueva la activación de la restricción (*incorpórela comentada al script **GXX_3_Soluciones.sql** de modo que la ejecución del mismo no de error y explique la respuesta del DBMS en el informe)*.

C. SERVICIOS (utilizando Vistas y/o Funciones)

Implemente los siguientes aspectos con el recurso que crea más conveniente, *incorporando la solución al script **GXX_3_Soluciones.sql** y la explicación correspondiente en el informe*. Provea sentencias que realicen el test de su solución (*incorpórela comentada al script **GXX_3_Soluciones.sql** de modo que la ejecución del mismo no de error y explique la respuesta del DBMS en el informe)*.

1) Utilizando el método que considere más adecuado, implemente un servicio que:
   a) brinde el precio actual de cotización en un mercado determinado.
   b) ejecute una orden de mercado (orden del tipo Market) para compra y venta.
   c) dado un mercado X y una fecha, retorne un listado con todas las órdenes ordenadas en forma cronológica junto con su tipo y estado (si está cumplida, si es nueva, etc.).
2) Utilizando el punto 1.a y 1.b, implemente el mecanismo que considere más adecuado para que dichos servicios se mantengan actualizados permanentemente.

Para ello:

- Implemente la regla de la manera que considere más apropiada en PostgreSQL, según las posibilidades que ofrece el DBMS (*incorpore la solución al script **GXX_3_Soluciones.sql** y la explicación correspondiente en el informe*)

- Provea una sentencia según corresponda, que promueva la activación de la regla (*incorpórela comentada al script **GXX_3_Soluciones.sql** de modo que la ejecución del mismo no de error y explique la respuesta del DBMS en el informe)*.

D. DEFINICIÓN DE VISTAS

Escriba la sentencia SQL para crear las vistas detalladas a continuación (*incorpórelas en el script **GXX_3_Soluciones.sql***). En cada caso comente y ejemplifique (*en el informe*) si es actualizable o no, indicando la/s causa/s (**Importante:** siempre que sea posible, deberá generar vistas automáticamente actualizables).

- Para la/s vista/s actualizable/s, provea una sentencia (*comentada dentro del script **GXX_3_Soluciones.sql***) que provoque diferente comportamiento según la vista tenga o no especificada la opción **With Check Option**, y analice dichos comportamientos (*en el informe*).

- Para la/s vista/s no actualizable/s, implemente un trigger *instead of* para el caso de inserción, plantee una sentencia que provoque su activación (*comentada dentro del script **GXX_3_Soluciones.sql***) y explique (en el informe) la propagación de dicha actualización. También incluya (en el informe) un análisis de las actualizaciones necesarias para las otras dos operaciones (update y delete).

1) Realice una vista que dé el saldo de cada moneda para cada uno de los usuarios.

2) Realice una vista que dé el listado de la cantidad de dinero que tiene cada usuario en cada moneda junto con la cotización al momento de ejecutar la vista en BTC y USDT.
3) Realice una vista (usando la vista anterior) que dé los 10 usuarios que más dinero tienen en TOTAL en toda su billetera.

E. MODIFICACIÓN DEL ESQUEMA

Suponga que ahora el sistema tiene que considerar que al quedar una orden incompleta, se genere una nueva con el remanente. ¿Qué modificaciones realizaría al modelo, en caso de requerirse, y cómo implementaría dicha operatoria?
Incluya la modificación requerida dentro del script ***GXX_4_Modificacion.sql*** y explique en detalle cómo se llevaría a cabo la operatoria en el informe.
Importante: No debe realizar modificaciones a los servicios, vistas y restricciones antes realizados.

Pautas de entrega

Se deberán entregar los 5 (cinco) scripts siguientes, que se deben **poder ejecutar en el siguiente orden y sin producir errores**:

1. script ***GXX_1_Creacion.sql***, que deberá contener la creación de lo solicitado en el punto **A**. Nota: cambiar los nombres de los objetos para ajustarlos a los de cada grupo (GXX_, donde **XX es el número asignado al grupo**).
2. script ***GXX_2_Carga.sql***, que deberá contener las sentencias de inserción de datos según lo especificado en el punto **A.**
3. script ***GXX_3_Soluciones.sql***, que deberá incluir la implementación de las restricciones y reglas del negocio, las vistas y los servicios detallados en los puntos **B, C y D**.
   Nota: Incluir como comentario dentro del script el ítem que resuelve en cada caso.
4. script ***GXX_4_Modificacion.sql*** que, en caso de requerirse, incorpore lo solicitado en el punto **E.**
5. script ***GXX_5_Borrado.sql****,* que incluya las sentencias de borrado de todos los objetos del esquema (tanto los creados por el script **GXX_1_*Creacion.sql*** como los objetos incorporados mediante el script ***GXX_3_Soluciones.sql*** *y* ***GXX_4_Modificacion.sql****, en el orden conveniente).*

Además, deberá entregar un Informe ***GXX_Informe.pdf****,* en formato pdf, de **no más de 8 páginas**, además de la carátula e índice (si lo hay), en el que se expliquen y justifiquen las decisiones asociadas a la implementación de los ítems solicitados.

Reglas de nombres

Para el desarrollo y corrección de los esquemas se seguirán estos estándares que permiten identificar cada trabajo y objeto con nombres completos y descriptivos (ver detalles y justificaciones en: https://vertabelo.com/blog/naming-conventions-in-database-modeling/).

El prefijo de grupo <GXX> se usará en tablas, vistas, procedimientos almacenados, triggers, índices y constraints. **XX se deberá reemplazar por el número de grupo asignado** por la cátedra en cada caso.

La nomenclatura a seguir es:

1. **Tablas/Vistas**: <GXX>_<nombre_tabla o vista>
2. **Índices: IDX**_<GXX>_<nombre_indice>
3. **Constraints**

   Clave Primaria: **PK**_<GXX>_<nnn>
   donde: nnn = nombre de la tabla para la que se construye la constraint

   Clave Extranjera: **FK**_<GXX>_<ppp>_<ccc>
   donde: ppp = nombre de la tabla referenciante u original, ccc =nombre de la tabla referenciada.

   Unique: **UQ**_<GXX>_<nnn>_<ccc>
   donde: nnn = nombre de la tabla para la que la constraint se construye, ccc = descripción de el/los campos involucrados.

   Check: **CK_**<GXX>_<nnn>_<ccc>
   donde: nnn = nombre de la tabla para la que se construye la constraint, ccc = descripción del chequeo.

   Assertion: **AS_**<GXX>_<nnn>
   donde: nnn = nombre de restricción general.

4. **Procedimientos, Triggers y Funciones**

   Triggers: **TR_**<GXX>_<tbl>_<nnn>
   siendo <nnn> el nombre que considere adecuado para el trigger y <tbl> la tabla a la cual está asociada el trigger.

   Procedimientos: **PR_**<GXX>_<nnn>
   siendo <nnn> el nombre que considere adecuado para el procedimiento.

   Funciones: **FN_**<GXX>_<nnn>
   siendo <nnn> el nombre que considere adecuado para la función.

   Funciones de triggers: **TRFN_**<GXX>_<nnn>
   siendo <nnn> el nombre que considere adecuado para la función de trigger.