Guía de ejercicios - Manipulación de datos y transaccionalidad en las operaciones (II)

¡Hola! Te damos la bienvenida a esta nueva guía de estudio.

¿En qué consiste esta guía?

La siguiente guía de estudio tiene como objetivo practicar y ejercitar los contenidos que hemos visto en clase.

Tabla de contenidos

¡Comencemos!

Actividad guiada: Aplicando rollback en cuentas bancarias.

Para este ejercicio seguiremos trabajando en la base de datos de cuentas bancarias, recuerda que la trabajamos en la sesión anterior. La acción de volver atrás las transacciones que se realicen sobre una tabla la realizaremos a través de save point y rollback.

Resumen de códigos:

1. Creación de la tabla:

```
create table cuentas (numero_cuenta int not null unique primary key,
balance float check(balance >= 0.00));
```

2. Inserción de registros:

```
into cuentas (numero_cuenta, balance) values (1, 1000);
into cuentas (numero_cuenta, balance) values (2, 1000);
```

3. Transacción controlada donde pasamos los $1000 de la cuenta 1 a la cuenta 2.

```
begin transaction;
UPDATE cuentas set balance = balance - 1000 where numero_cuenta = 1;
UPDATE cuentas set balance = balance + 1000 where numero_cuenta = 2;
```

4. Confirmamos la transacción:

```
commit;
```

Con este resumen de pasos realizados en la sesión anterior, continuemos entonces con el procedimiento de revertir cambios y las consideraciones que debemos tener en cuenta.

- **Paso 1:** Nos conectamos a la base de datos

```
\c transacciones;
```

- **Paso 2:** En la tabla cuentas realizamos una transacción en donde transferimos $1000 de la cuenta 1 a la cuenta 2 y no obtuvimos ningún error. Sin embargo, podemos verificar que en las transacciones que terminan en error no se altera el estado de nuestros datos. ¿Y cómo lo hacemos?

Ejecuta el siguiente código en tu terminal.

```
BEGIN TRANSACTION;
UPDATE cuentas SET balance = balance + 1000 WHERE numero_cuenta = 2;
UPDATE cuentas SET balance = balance - 1000 WHERE numero_cuenta = 1;
ROLLBACK;
```

Al ejecutarlo veremos el siguiente error debido a la restricción de la columna balance que realizamos con **check** al momento de crear la tabla.

```
DETAIL:  Failing row contains (1, -1000).
```

- **Paso 3:** Verificamos el estado de la tabla.

```
select * from cuentas;
```

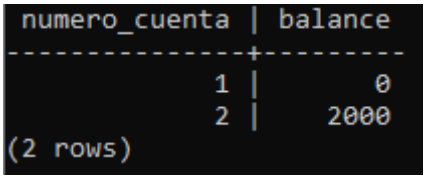

Imagen 1. Estado sin cambios
Fuente: Desafío Latam.

Como verás no hubo ningún cambio en los datos, a pesar de haber cargado $1000 a la cuenta 2 primero, esta no se realiza, puesto que la resta al balance de la cuenta 1 devolvió un error.

Si por alguna razón, en medio de una transacción se decide que ya no se quiere registrar los cambios (tal vez nos dimos cuenta de que estamos actualizando todos los registros de nuestra base y no es lo que buscábamos), se puede recurrir a la orden ROLLBACK en lugar de COMMIT y todas las actualizaciones hasta ese punto quedarán canceladas.

Save point

Recordemos que con Save point podemos tener un mayor control de las transacciones por medio de puntos de recuperación.

- **Paso 4:** Intentemos registrar una nueva cuenta de número 3 en nuestra tabla "cuentas" con un saldo de $5000 y justo luego guardemos ese punto de la transacción con un SAVEPOINT de nombre "nueva_cuenta".

```
BEGIN TRANSACTION;
INSERT INTO cuentas(numero_cuenta, balance) VALUES (3,
5000);
SAVEPOINT nueva_cuenta;
```

- **Paso 5:** Hasta este punto tenemos la transacción en curso y hemos fijado que podríamos volver a este estado en cualquier circunstancia. Ahora, intentemos transferir a esta nueva cuenta $3000 desde la cuenta 2. Para esto continua la transacción de la siguiente manera.

```
UPDATE cuentas SET balance = balance + 3000 WHERE numero_cuenta = 3;
UPDATE cuentas SET balance = balance - 3000 WHERE numero_cuenta = 2;
-- Justo acá deberás recibir un error
ROLLBACK TO nueva_cuenta;
COMMIT;
```

Error generado:

```
ERROR:  new row for relation "cuentas" violates check constraint
"cuentas_balance_check"
DETAIL:  Failing row contains (2, -1000).
```

Como puedes notar se registró con éxito la cuenta 3, no obstante su balance sigue siendo $5000 y el balance de la cuenta 2 no se redujo. Esto es porque volvimos al punto guardado, luego de haber hecho la inserción de nuestra nueva cuenta, a pesar de no haberse procesado lo que le procedía a esa instrucción.

¡Manos a la obra! - Obteniendo registros

A continuación, deberás insertar al menos 10 registros más en la base de datos de transacciones. Una vez ingresados, genera un reporte con los siguientes datos:

- Reporta aquellas cuentas cuyo balance sea mayor a 2000.
- Reporta cuántas tienen un balance inferior a 1000.
- Reporta el promedio total de las cuentas registradas según su balance.
- Reporta el promedio de cuentas cuyo balance sea mayor o igual a 10000.

Para generar este tipo de reportes puedes utilizar herramientas como:

- Google docs.
- Presentaciones de google.
- Word
- Power Point

Para las dos primeras, solo debes acceder a Drive de tu cuenta gmail y agregar los pantallazos de los resultados a partir de los códigos que hayas implementado en SQL.

Solución ejercicio propuesto

- Inserción de registros

```
INSERT INTO cuentas(numero_cuenta, balance) VALUES (4,
7000);
INSERT 0 1
postgres=# INSERT INTO cuentas(numero_cuenta, balance) VALUES (5,
500);
INSERT 0 1
postgres=# INSERT INTO cuentas(numero_cuenta, balance) VALUES (6,
12000);
INSERT 0 1
postgres=# INSERT INTO cuentas(numero_cuenta, balance) VALUES (7,
900);
INSERT 0 1
postgres=# INSERT INTO cuentas(numero_cuenta, balance) VALUES (8,
100);
INSERT 0 1
postgres=# INSERT INTO cuentas(numero_cuenta, balance) VALUES (9,
20000);
INSERT 0 1
postgres=# INSERT INTO cuentas(numero_cuenta, balance) VALUES (10,
50);
INSERT 0 1
```

- Reporta aquellas cuentas cuyo balance sea mayor a 2000.

```
 select * from cuentas where balance >= 2000;
```

- Reporta cuántas tienen un balance inferior a 1000.

```
select * from cuentas where balance <= 1000;
```

- Reporta el promedio total de las cuentas registradas según su balance.

```
select avg(balance) from cuentas;
```

- Reporta el promedio de cuentas cuyo balance sea mayor o igual a 10000.

```
select avg(balance) from cuentas where balance >= 2000;
```