Los posibles problemas de transaccionalidad y concurrencia pueden ocurrir en nuestro caso cuando haya alguna tabla la cual tenga que un lock exclusivo. En nuestro caso solo ocurre en al insertar, exceptuando en dos casos. Las funciones cargar\_saldo() y calcular\_importe().

La función cargar\_saldo ()dada una cuenta, aumenta el saldo de la misma haciendo un update, reescribe el valor de saldo, por lo cual necesita un lock exclusivo de la tupla cuyo saldo actualiza. De esta forma, si no es confirmada la transacción se mantendrá el lock, haciéndolo imposible acceder a la tupla.

La función calcular\_importe permite la obtención del código de tarifa, y número de cuenta correspondiente a un auto que pasa por un peaje, además del cálculo de importe.

Tomando el caso donde existe una cuenta asociada al auto que esta pasando, se necesita actualizar el saldo de la misma, esto se hace a través de una update sobre la tupla correspondiente a la cuenta y sobrescribiendo el saldo, necesitando un lock exclusivo (write lock) de la tupla.

Lo mismo causa que si intenta acceder a la tupla con otra transacción la misma quede en espera. En el caso que otro auto asociado a la cuenta pase por un peaje, o alguna acción que necesite acceder a la tupla (como cargar saldo) ocurra, al intentar acceder al importe para actualizar o incluso leer en caso de hacer una consulta de valores, las transacciones quedarían en espera hasta que se confirme la primera. Esto no sucedería al intentar borrar pasadas, siendo que esto no actualiza el saldo ni accede a la tabla de cuentas.

Para solucionar esto, se puede abortar la transacción, esto se puede hacer de la siguiente forma:

**-select pg\_cancel\_backend()**

Esta función intenta cancelar la consulta que está siendo ejecutada por el backend identificado por pid (identificador de proceso).

Envía una solicitud de cancelación al backend. Si el backend está ejecutando una consulta en ese momento, intentará cancelarla.

Es una operación no bloqueante. Retorna inmediatamente después de enviar la solicitud de cancelación.

**-select pg\_terminate\_backend()**

Esta función termina de manera forzada el proceso del backend identificado por pid. Termina abruptamente el backend y finaliza su sesión actual

Estas funciones precisan del pid. Para conseguir pids de transacciones, y los pids de las transacciones que lo están bloquando con la siguiente transacción.

SELECT

blocked.pid AS blocked\_pid,

blocked.usename AS blocked\_user,

blocked.query AS blocked\_query,

blocking.pid AS blocking\_pid,

blocking.usename AS blocking\_user,

blocking.query AS blocking\_query

FROM

pg\_catalog.pg\_locks AS blocked\_locks

JOIN

pg\_catalog.pg\_stat\_activity AS blocked

ON blocked\_locks.pid = blocked.pid

JOIN

pg\_catalog.pg\_locks AS blocking\_locks

ON blocking\_locks.locktype = blocked\_locks.locktype

AND blocking\_locks.database IS NOT DISTINCT FROM blocked\_locks.database

AND blocking\_locks.relation IS NOT DISTINCT FROM blocked\_locks.relation

AND blocking\_locks.page IS NOT DISTINCT FROM blocked\_locks.page

AND blocking\_locks.tuple IS NOT DISTINCT FROM blocked\_locks.tuple

AND blocking\_locks.virtualxid IS NOT DISTINCT FROM blocked\_locks.virtualxid

AND blocking\_locks.transactionid IS NOT DISTINCT FROM blocked\_locks.transactionid

AND blocking\_locks.classid IS NOT DISTINCT FROM blocked\_locks.classid

AND blocking\_locks.objid IS NOT DISTINCT FROM blocked\_locks.objid

AND blocking\_locks.objsubid IS NOT DISTINCT FROM blocked\_locks.objsubid

AND blocking\_locks.pid != blocked\_locks.pid

JOIN

pg\_catalog.pg\_stat\_activity AS blocking

ON blocking\_locks.pid = blocking.pid

WHERE

blocked\_locks.granted = false;