Cambiar nombres de tablas y particiones

Para cambiar el nombre de una tabla existente, se usa la instrucción ALTER TABLE de la siguiente manera:

```
ALTER TABLE nombre_tabla
RENAME TO nuenvo_nombre;
```

Detalle:

1. Primero, se especifica el nombre de la tabla cuyo nombre desea cambiar después de la cláusula ALTER TABLE.
2. En segundo lugar, proporcione el nuevo nombre de la tabla después de la cláusula RENAME TO.
3. Termine con punto y coma

Si se intenta cambiar el nombre de una tabla que no existe, PostgreSQL emitirá un error. Para evitar ésto, se agrega la opción IF EXISTS de la siguiente manera:

```
ALTER TABLE IF EXISTS nombre_tabla
RENAME TO nuenvo_nombre;
```

En este caso, si la tabla 'nombre_tabla' no existe, PostgreSQL emitirá un aviso en su lugar.

Cuando cambia el nombre de una tabla, PostgreSQL actualizará automáticamente sus objetos dependientes, como restricciones de llave foránea, vistas e índices, no hay afectación a los datos almacenados en la tabla.

Sí así es, "*...actualizará automáticamente sus objetos dependientes, como restricciones de llave foránea, vistas e índices, no hay afectación a los datos almacenados en la tabla...*", genial! ¿por qué esto es importante?

Ahora suponemos que se tiene un servidor de base de datos que recibe millones de peticiones por hora, debe insertar los registros que una aplicación de IoT le envía; aplicación instalada en una gran cantidad dispositivos que han sido montados en los postes de luz de las calles de nuestra ciudad; estamos hablando de miles de dispositivos, cada uno de ellos envía información de sus sensores en intervalos de uno o dos segundos, tiene sensores de:

- Ruido
- Contaminación
- Luz

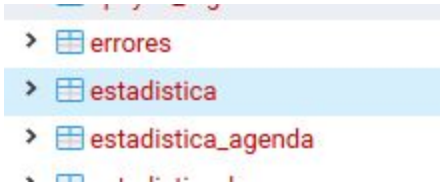
- Proximidad
- Humedad
- Radiación

La aplicación de IoT y su servicio de base de datos funciona perfecto al principio, a pesar de tener alrededor de un millón de registros por hora, no se ve reducción del rendimiento de la base de datos porque se ha configurado cuidadosamente y aplica índices sólo cuando son realmente necesarios, las inserciones funcionan relativamente rápido y todos están felices..

Sin embargo, han pasado unos 20 días y se ha solicitado empezar con el análisis de los datos almacenados hasta el momento, se tienen millones de registros que se deben procesar y entregar en reportes, los millones de registros son datos en bruto.

Primero: ¿cuántos datos hay?, para ver las estadísticas de la tabla se puede usar pgAdmin:

1. Seleccionar la tabla en el menú de la izquierda (en éste caso la tabla se llama estadística):



2. Hacer clic en el menú superior que dice “Statistics”:

3. Se observa como resultado:

Statistics	Value
Sequential scans	2069
Sequential tuples read	647034206981
Index scans	1773197
Index tuples fetched	33772838671
Tuples inserted	451042380
Tuples updated	171279
Tuples deleted	0
Tuples HOT updated	13423
Live tuples	450822959
Dead tuples	618

“Live tuples” indica que existen alrededor de 450 millones.

4. Es posible corroborar esta cifra viendo el estimado de columnas de la tabla. Sólo es necesario seleccionar la pestaña “Properties” en lugar de “Statistics”. Ir a la sección “Advanced” y en la fila “Rows (estimated)” el valor debe ser cercano.

Son entonces alrededor de 450 millones de registros, procesarlos implica que cada fila debe pasar por una serie de cálculos y luego insertar el resultado de la operación en otra tabla, al finalizar se debe borrar la fila procesada de la tabla estadística y continuar con el proceso, para empeorar se requiere que ésta tarea se ejecute “casi” en tiempo real. Que en cuestión de minutos se pueda tener un reporte.

Se hace una pequeña prueba de rendimiento, ver cuánto tarda en procesar mil registros, se tiene como resultado:

No será viable hacerlo así de simple pues según los cálculos un millón de registros por hora es aproximadamente 17.000 registros por minuto. Nunca se va a vaciar la tabla.

Se acude a las Particiones pero:

1. No podemos particionar una tabla existente.
2. Se necesitará entonces crear una tabla vacía particionada y empezar a mover los datos desde la tabla principal, dicha copia tampoco es rápida.
3. Requiere cambiar el diseño de la aplicación para usar las particiones.
4. Es necesario planear si las particiones serán por fechas, horas, minutos, segundos. Las tablas hijas de las particiones deben ser creadas a mano.

Airbnb tuvo un desafío similar y tardaron dos semanas, sí, una empresa de más de 100 desarrolladores tardó dos semanas en particionar su base de datos, ¿cuánto se tardará si lo hace solo una persona? ¿Qué hacer?

Hay una opción algo rápida de implementar.

Renombrar tablas

Crear una tabla llamada

```
estadistica_offload
```

Con exactamente la misma estructura de la tabla principal estadística.

El servidor que recibe los datos de todos los dispositivos IoT y los procesa debe tener una lógica que se ejecuta cada minuto o menos y es algo así:

...
Consultar estadística;
Si está con datos, procesar mil datos y borrar lo procesado;
Si está vacía, no hacer nada;
…

Pero ésta lógica nunca va a lograr vaciar la tabla ya que llegan más datos por minuto de los que puede procesar. Por lo tanto, se debe agregar una tarea para poder usar el renombramiento de tablas como un potencializador del rendimiento así:

...
Consultar ***estadistica_offload***
Si está con datos, procesar mil datos y borrar lo procesado
Si está vacía, ***renombra tablas****.*
…

La idea es hacer la consulta sobre la tabla **estadistica_offload** que NO está siendo “bombardeada” de datos, lo que permite que la lectura y modificación de datos sea mucho más rápida que la tabla **estadistica.**

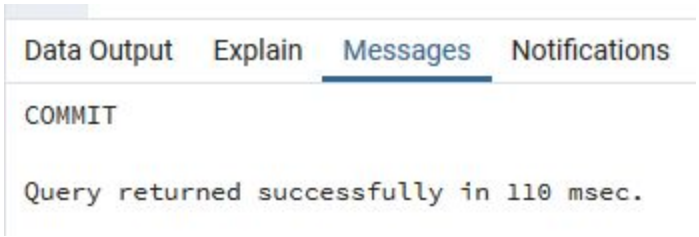
Luego el renombrado de tablas consiste en una transacción así.

```
BEGIN TRANSACTION;
```

```
ALTER TABLE estadistica RENAME TO estadistica_temp;
ALTER TABLE estadistica_offload RENAME TO estadistica;
ALTER TABLE estadistica_temp RENAME TO estadistica_offload;
COMMIT;
```

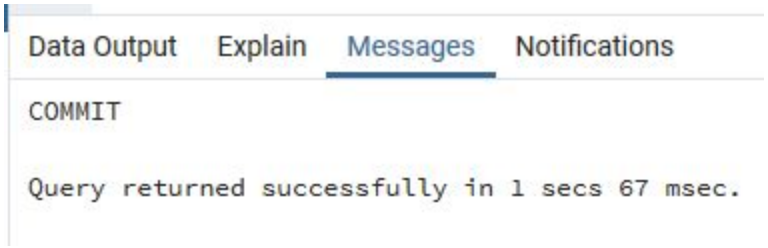
Se debe encapsular la operación en una transacción ya que ninguno de los pasos puede fallar, si falla uno, se debe reintentar todo el proceso.

Al terminar la transacción del renombrado se obtiene:



Ahora, todas las inserciones de los sensores se están haciendo sobre una tabla vacía, estadística ahora no tiene datos y **estadistica_offload** tiene todos los datos de la tabla estadística, no hubo pérdida de datos ya que debido a la transacción, la tabla estadística siempre existió y en los milisegundos que duró el renombrado todas las inserciones se pusieron en espera hasta que se terminara la transacción.

Al procesar mil registros tarda:



Y a medida que la tabla estadistica_offload baje su número de registros, el tiempo de este proceso tiende a bajar también. Cuando estadistica_offload llega a estar vacía, se vuelve a hacer el renombrado y repetimos el ciclo una y otra vez hasta el punto en el que en la tabla estadística no habrán máximo miles de registro antes que se haga el renombrado.

¿Por Qué Funciona?

Básicamente el problema es de bloqueo de tabla, cuando una consulta tipo borrado se debe ejecutar, le avisa al motor de base de datos su objetivo y este debe detener todas las consultas

y otras modificaciones a la tabla hasta que el proceso termine, pero como la cantidad de inserciones es alta, la consulta de borrado se queda esperando a que el motor le dé el turno de trabajar o peor aún,si le llega a entregar el turno, el proceso de borrado puede llegar a tardar minutos en el que ninguna inserción puede ser ejecutada, lo que causa encolamiento de peticiones hasta que se desborda la capacidad del servidor y el resto es historia.
El borrado de filas no solo implica eliminar las filas indicadas, requiere actualizar los índices y actualizar los índices es una tarea demorada en tablas grandes. El borrado de estadistica_offload también puede tardar, pero no afecta a nadie y el renombrado de tablas siempre será más rápido.

Consejo, se puede hacer TRUNK a offload cuando llegue a estar vacía, con eso se limpian los índices y el cache, dejándola cómo nueva cada vez. La solución final y escalable es usar particiones, toma más tiempo en implementar, pero es definitiva.