PRIMUS DATA PRESS

CUADERNOS

Cuaderno N.º 011

SERIE · TRUCOS DE T-SQL PARA TU DW

Partition Switching

Reemplazar o purgar millones de filas cambiando metadata: en milisegundos, sin mover una sola fila física —y la trampa que puede revivir todo el costo que veníamos a evitar.

Javier Loria

Inteligencia de Negocios

Primus Data

Inteligencia de Negocios y Analítica Avanzada

¿Cómo cargo o purgo millones de filas sin bloquear el reporting?

En un data warehouse grande, dos rutinas se vuelven un dolor cuando la fact pesa cientos de millones de filas: cargar el lote nuevo —el mes que cierra— y borrar el histórico viejo que ya nadie consulta.

A fuerza bruta, un `INSERT` masivo o un `DELETE` de millones de filas **bloquean la tabla**, inflan el log y dejan la ventana de carga sin cerrar a tiempo. La respuesta de nivel industrial es **partition switching**: meter o sacar un bloque entero de datos cambiando metadata, sin mover una sola fila.

> *Así lo hacemos en Primus Data — y así se lo enseñamos al personal técnico de nuestros clientes en nuestras mentorías.*

✦

LA SERIE

Undécima entrega de Trucos de T-SQL para tu DW, de la charla del mismo nombre en **Charlemos de SQL Server 2026**. *Es el cuaderno más avanzado de la serie, y uno de los que más tiempo ahorra: en el EDW que auditamos, resolvía en milisegundos lo que un DELETE masivo tardaba minutos. La* **Cooperativa MuuSQL** *reproduce un EDW real sin exponer al cliente; la base vive en el repo de la serie.*

ANTES DE LAS NOTAS

La bodega que cambia el rótulo, no las cajas

Tenés que reemplazar toda la mercadería de mayo en una bodega. El novato entra, saca caja por caja la vieja, la carga en un camión, y después mete caja por caja la nueva. Horas, con la bodega cerrada y nadie más pudiendo entrar.

El que sabe hace otra cosa. La mercadería nueva ya está empacada en la bodega de al lado. Va al tablero de la entrada y cambia un número: la de al lado pasa a ser «la de mayo», y la vieja pasa a «descarte». No movió una sola caja. Cambió un rótulo. Segundos, sin cerrar nada.

LO MISMO, SIN METÁFORA

Partition switching es cambiar el rótulo. La data física no se mueve: cambia a qué tabla «pertenece» cada partición. Eso convierte «reemplazá el mes de mayo en una fact de cientos de millones de filas» en una operación de metadata, instantánea.

— **J. L.**, *cambiando el rótulo de la bodega en vez de mover las cajas.*

EL PROBLEMA

El DELETE masivo cierra la bodega

Mayo se re-liquidó: un reajuste de romana subió los litros un 0.5%, y hay que reemplazar el mes entero en la fact. La forma ingenua borra y vuelve a insertar:

EL NOVATO CARGANDO CAJAS

```
DELETE FROM fact.Entregas
WHERE  FechaID BETWEEN 20260501 AND 20260531;
INSERT INTO fact.Entregas SELECT ...   -- mayo corregido
```

Funciona en una tabla chica. En una fact grande, el `DELETE` **registra en el log cada fila** que borra, escala a un lock que cierra la tabla entera, y todavía falta re-insertar otro tanto. En la demo son 186 000 filas de mayo sobre 11 millones; en el EDW real, particiones de cientos de millones. Minutos de bloqueo y un log que explota —para reemplazar data que ya tenías lista al lado.

> REGLA DE CAMPO
>
> *Un `DELETE` / `UPDATE` masivo sobre una fact grande no es «lento»: es un lock de tabla y un log que crece tanto como la data que tocás.*

EL TRUCO

El switch de dos pasos

El patrón del EDW usa tres tablas de schema idéntico, todas sobre el mismo esquema de partición: **main** (la fact viva), **switch** (la data nueva, ya cargada) y **empty** (vacía, para recibir lo viejo). Reemplazar mayo son dos `SWITCH`, los dos de metadata:

REEMPLAZAR MAYO · DOS PASOS

```
DECLARE @p INT = $PARTITION.PF_EntregasMes(20260501);

BEGIN TRAN;
-- 1) sacar el mayo viejo (deja la partición vacía):
ALTER TABLE fact.Entregas       SWITCH PARTITION @p TO fact.EntregasEmpty
PARTITION @p;
-- 2) entrar el mayo corregido:
ALTER TABLE fact.EntregasSwitch SWITCH PARTITION @p TO fact.Entregas
PARTITION @p;
COMMIT;
```

No hubo lecturas de 186 000 filas ni log de datos: solo se reescribieron punteros. El switch sí toma un lock `Sch-M`, pero como no mueve datos, entra y sale en un parpadeo. Van en una transacción a propósito: entre el paso 1 y el 2 la fact no tiene mayo, y el `BEGIN TRAN` evita que alguien lo lea a medias. El número de partición lo resuelve `$PARTITION.fn(valor)`, nunca tu cabeza.

> REGLA DE CAMPO
>
> *El número de partición lo dice `$PARTITION.fn(valor)`, nunca contando a ojo. Contar particiones a mano es cómo se reemplaza el mes que no era.*

LA OTRA VÍA

Vaciar la partición sin un DELETE

Para vaciar la partición vieja hay dos caminos, y el EDW usa los dos: switchearla a una tabla `empty` —lo del paso 1— o, desde SQL Server 2016, truncar solo esa partición:

SQL SERVER 2016+

```
TRUNCATE TABLE fact.Entregas
  WITH (PARTITIONS(@p));
```

`TRUNCATE` de una partición también es metadata —deslocaliza páginas, no registra fila por fila— y tiene un efecto que conviene notar: **vuelve opcional la tabla `empty`**. El patrón de tres tablas se reduce a dos (main + switch). La tabla `empty` es el patrón clásico de antes de 2016, cuando la única forma de vaciar sin un `DELETE` era sacar la partición con un switch.

Una cosa que el switch **no** hace: actualizar las estadísticas. El optimizador sigue creyendo la distribución vieja hasta el próximo `UPDATE STATISTICS` —si la forma de los datos cambió, agendalo después.

> REGLA DE CAMPO
>
> *En SQL Server 2016+, `TRUNCATE ... WITH (PARTITIONS)` te ahorra la tabla `empty`. Si todavía la mantenés «porque siempre se hizo así», revisá tu versión.*

LA PRIMERA TRAMPA

El switch es exigente, y avisa fuerte

Lo bueno es que no falla en silencio: si no se cumplen los requisitos, truena con un error explícito antes de tocar nada. Los que más muerden:

- **Schema idéntico** entre origen y destino: mismas columnas, tipos, nullability — incluye `IDENTITY` y columnas calculadas.
- **Todos los índices alineados** al mismo esquema de partición. Uno sin alinear y el switch es imposible (error 4907). En una fact real, eso incluye el *clustered columnstore*.
- **La partición destino debe estar vacía** (si no, error 4982). Por eso vaciás `main` antes de entrar lo nuevo; la partición *origen* sí puede tener datos.
- **Mismo filegroup**, compresión idéntica por partición y constraints compatibles.
- **No puede estar referenciada por un** `FOREIGN KEY` de otra tabla. Golpea sobre todo al switchear una dimensión; las FK que *entran* hay que soltarlas antes.

> REGLA DE CAMPO
>
> *Si el switch falla, leé el error: casi siempre es un índice sin alinear o la partición destino que no quedó vacía. No es magia, es una lista de verificación.*

LA SEGUNDA TRAMPA

La ventana deslizante mueve fronteras

Reemplazar un mes es media historia. Una *ventana deslizante* es el patrón completo: cada mes nuevo entra, el más viejo sale, la fact mantiene siempre N meses. Eso pide mover las **fronteras** de la función de partición:

CORRER LA VENTANA

```
-- la partición vieja ya se vació por switch-out:
ALTER PARTITION FUNCTION PF_EntregasMes() MERGE RANGE (20210701);  -- cierra el mes viejo
ALTER PARTITION SCHEME   PS_EntregasMes NEXT USED [PRIMARY];        -- obligatorio antes del SPLIT
ALTER PARTITION FUNCTION PF_EntregasMes() SPLIT RANGE (20260801);  -- abre el mes nuevo
```

La trampa contraintuitiva: **hacé `SPLIT` y `MERGE` solo sobre particiones vacías.** Si splitás una partición con datos, SQL Server mueve físicamente las filas y lo registra todo en el log —el costo que el switch venía a evitar, y peor, con la tabla viva. Y un apunte del cuaderno de las vigencias: con `RANGE RIGHT`, el límite `20260501` **abre** mayo; equivocar eso es un off-by-one de un mes entero.

> REGLA DE CAMPO
>
> `SPLIT` y `MERGE` *solo sobre particiones vacías. Splitear una con datos adentro mueve filas y satura el log: el anti-patrón disfrazado de mantenimiento.*

¿Y EN FABRIC?

En ninguno de los dos sabores

Fabric Warehouse no tiene particionamiento de tablas: no hay funciones ni esquemas de partición, ni `SWITCH`. **Fabric SQL Database** sí es motor Azure SQL y soporta particionar, pero el `SWITCH PARTITION` está **explícitamente prohibido** como DDL: toda Fabric SQL Database se espeja a OneLake, y el switch es incompatible con ese espejado.

Es el contraste curioso con las temporal tables del cuaderno anterior, que ahí sí corren. Así que este truco es de **SQL Server** y **Azure SQL Database** de punta a punta. Para «reemplazá esta ventana» del lado analítico de Fabric, el equivalente vive en Delta Lake/Spark: reescribir solo las particiones afectadas con `replaceWhere` o *partition overwrite*.

> REGLA DE CAMPO
>
> *El switch vive en SQL Server (2016+) y Azure SQL Database. En Fabric no —ni Warehouse ni SQL Database—; del lado analítico,* `replaceWhere` *de Delta responde lo mismo. Microsoft lo mueve seguido: reverificá.*

PARA LLEVAR

Cuatro cosas sobre partition switching

— **Switch = cambiar el rótulo, no mover las cajas.** Reemplazar una partición es metadata: instantáneo, sin locks largos, sin log de filas. El `DELETE` / `INSERT` masivo es lo contrario.

— **El patrón de tres tablas: main + empty + switch.** Vaciás la partición destino (switch a `empty` o `TRUNCATE WITH PARTITIONS`) y entrás la nueva.

— **El switch es una lista de requisitos, no magia.** Schema idéntico, índices alineados, partición destino vacía. Si falla, el error te dice cuál de las tres faltó.

— **Ventana deslizante = `SPLIT` / `MERGE RANGE`, siempre sobre particiones vacías.** Mover una frontera con datos adentro mueve las filas y satura el log.

— El script completo —la fact de 11M filas particionada por mes, con las tres tablas alineadas sobre la base de la Cooperativa MuuSQL— está en el repo de la serie. Corre en cualquier SQL Server 2016+ con un F5.

PARA SEGUIR LEYENDO

Estas notas siguen abiertas.

Si algún patrón te resonó, o te pareció equivocado, me interesa saberlo. La conversación técnica de fondo vive en el blog, y casi siempre mejora con quien la discute.

primusdata.net/blog · primusdata.net/recursos

Cuadernos Primus N.º 011, de la serie Trucos de T-SQL para tu DW. Texto compuesto en Philosopher; títulos y código en Expletus Sans y monoespaciado. Patrones auditados en un EDW real. Las opiniones son del autor; los errores, también.

Javier Loria
para Primus Data Press