PRIMUS DATA PRESS

CUADERNOS

Cuaderno N.º 002

SERIE · TRUCOS DE T-SQL PARA TU DW

El truco del INSERT idempotente

Sembrar catálogos sin miedo al re-run — y las trampas del NOT IN, el config drift y Fabric que nadie cuenta.

Javier Loria

Inteligencia de Negocios

Primus Data

Inteligencia de Negocios y Analítica Avanzada

¿Cómo hago un script de carga que pueda correr muchas veces sin duplicar datos?

La pregunta aparece apenas tenés que sembrar los **catálogos de un data warehouse** —tipos de movimiento, estados, conceptos— en dev, en QA, en producción y en el ambiente nuevo de la semana que viene. El reflejo —un `INSERT` que asume tabla vacía— vuelve cada despliegue una ruleta: o truena a la mitad, o inserta filas duplicadas.

Lo que querés es un **script idempotente**: mismo resultado lo corras una o diez veces. Y se escribe con una sola técnica de conjuntos, `(VALUES) EXCEPT SELECT`.

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

✦

LA SERIE

Segunda entrega de la serie Trucos de T-SQL para tu DW, nacida de la charla del mismo nombre en **Charlemos de SQL Server 2026**. *Los patrones salen de auditar un EDW real de retail y agroindustria con más de diez años en producción; la* **Cooperativa MuuSQL** *es un escenario sintético que reproduce esos casos sin exponer al cliente. La base de demo y el script viven en el repo de la serie.*

ANTES DE LAS NOTAS

Reponer la góndola, no vaciar el estante

Pensá en reponer una góndola del supermercado. Nadie vacía el estante para volver a llenarlo desde bodega: el reponedor mira qué falta y trae solo eso. Si la góndola está completa, no trae nada. Y puede pasar por el mismo pasillo diez veces al día sin que el estante se desborde.

Tu script de seed no conoce ese oficio. Trae la tarima completa de bodega cada vez, y cuando encuentra la góndola llena, truena.

> EL CASO DE ESTAS NOTAS
>
> Los ejemplos siembran `cfg.Bonificaciones`, un catálogo de la **Cooperativa MuuSQL** —escenario sintético que reproduce casos reales de un EDW de retail y agroindustria sin exponer al cliente.
>
> Léelas como conversación en voz alta, no como postura cerrada.

— **J. L.**, *entre una sesión de estudio y un cliente.*

EL PROBLEMA

El script que solo se puede correr una vez

Todo data warehouse tiene catálogos chicos que alguien tiene que sembrar. La versión ingenua es un `INSERT … VALUES` con todas las filas. Funciona perfecto… la primera vez. La segunda —el deploy a QA, el hotfix, el ambiente nuevo— choca contra la primary key.

ERROR

```
Msg 2627: Violation of PRIMARY KEY constraint 'PK_Bonificaciones'…
```

En el EDW que auditamos venían en tres sabores, y los tres muerden.

— **El INSERT pelado.** Corre una vez. A la segunda, error 2627 y el deploy se cae a la mitad.

— **El `IF NOT EXISTS` por fila.** Funciona, pero con 50 conceptos son 200 líneas de boilerplate.

— **El `DELETE` + `INSERT`.** Truena con foreign keys, pisa columnas de auditoría, y si el INSERT falla a media carga la tabla queda vacía.

> REGLA DE CAMPO
>
> *Un script de seed que solo se puede correr una vez no es un script: es una trampa con fecha de activación.*

EL TRUCO

Declará el conjunto, restá lo que ya existe

T-SQL permite usar `(VALUES …)` como tabla inline en el `FROM` — el *table value constructor*. Combinado con `EXCEPT`, el seed se escribe una sola vez y el motor resta lo que ya está en la tabla.

T-SQL

```
INSERT INTO cfg.Bonificaciones (Codigo, Concepto, MontoPorLitro)
SELECT Codigo, Concepto, MontoPorLitro
FROM (VALUES ('ORG',  'Certificación orgánica',   12.50),
             ('FRIO', 'Tanque de frío',             4.00),
             ('VOL',  'Volumen >200 L/día',         6.00),
             ('CAL',  'Calidad bacteriológica A',   8.00)
             -- … más conceptos, una línea cada uno
     ) AS v (Codigo, Concepto, MontoPorLitro)
EXCEPT
SELECT Codigo, Concepto, MontoPorLitro
FROM cfg.Bonificaciones;
```

Leelo como teoría de conjuntos, porque eso es: *el conjunto que quiero, menos el que tengo, es lo que falta*. Primera corrida: inserta las filas que faltaban. Segunda: 0. Décima: 0. F5 sin miedo.

EL TRUCO · LETRA CHICA

Dos detalles antes de confiarle el deploy

`EXCEPT` empareja por **posición**, no por nombre[1]. Los dos `SELECT` —y la lista del `INSERT`— tienen que enumerar las mismas columnas en el mismo orden.

`EXCEPT` compara según la **collation** de la columna[2]. En una base case-sensitive, `'ORG'` y `'org'` son filas distintas: si el dato guardado difiere del literal aunque sea en el casing, la fila "falta" y se re-inserta.

1 · Una columna de más, o fuera de orden, y ninguna fila matchea: el seed re-inserta todo hasta toparse con la PK.

2 · El mismo enemigo de la Trampa 4 del Cuaderno N.º 001: la collation y el byte no opinan igual.

> REGLA DE CAMPO
>
> *En `(VALUES) EXCEPT SELECT`, ambos lados listan las mismas columnas en el mismo orden. `EXCEPT` empareja por posición; un desajuste re-inserta en cada corrida.*

LAS TRAMPAS

Trampa 1 · el NOT IN que parecía equivalente

La alternativa que más se ve en producción es `NOT IN`. Parece equivalente; no lo es.

T-SQL

```
SELECT v.Codigo
FROM (VALUES ('X'), ('Y')) AS v (Codigo)
WHERE v.Codigo NOT IN (SELECT 'X' UNION ALL SELECT NULL);
-- 0 filas. ¿Y la 'Y'?
```

Cero filas. La `'Y'` no está en la lista, pero no sale, porque la lista trae un NULL: `'Y' <> NULL` da UNKNOWN, el filtro entero da UNKNOWN, y la fila desaparece. Sin error, sin warning. El seed reporta 0 insertadas y todos asumen que ya estaban.

> REGLA DE CAMPO
>
> *`NOT IN` contra una subconsulta que pueda traer NULL no filtra: envenena. `EXCEPT` o `NOT EXISTS`, nunca `NOT IN`.*

LAS TRAMPAS

Trampa 2 · alguien cambió la config a mano

`EXCEPT` compara **todas las columnas que proyecta el seed**, no solo la llave. Supongamos que un viernes alguien ajustó un monto directo en producción.

T-SQL

```
UPDATE cfg.Bonificaciones SET MontoPorLitro = 4.50
WHERE Codigo = 'FRIO';
```

La próxima corrida ve que `('FRIO', …, 4.00)` del script "no existe" (existe con 4.50), intenta insertarla, y la PK truena con 2627. No lo calles: ese error te avisa que **el script y la base ya no cuentan la misma historia**. Config drift, detectado en el deploy y no tres meses después. La misma filosofía del índice UNIQUE filtrado del Cuaderno N.º 001.

> REGLA DE CAMPO
>
> *Este patrón inserta lo que falta, no actualiza lo que cambió. Si el seed truena con PK violation, investigá quién tocó la config — no lo silencies con `IF NOT EXISTS`.*

PARA CERRAR

Bonus · el DISTINCT que viene gratis

`EXCEPT` opera sobre conjuntos, y los conjuntos no tienen duplicados. Si un copy-paste deja la misma fila dos veces en el `VALUES`, el resultado la trae una sola: el insert no duplica. Un seguro más que el `INSERT … VALUES` pelado no da.

¿Y en Fabric?

`(VALUES)` y `EXCEPT` funcionan en Fabric Warehouse y el patrón corre tal cual. La letra chica está en la Trampa 2: las primary keys de Fabric son `NOT ENFORCED`, así que el drift no truena — la fila vieja entra como duplicado, calladita. Ahí conviene un check explícito después del seed.

FABRIC

```
SELECT Codigo, COUNT(*)
FROM cfg.Bonificaciones
GROUP BY Codigo
HAVING COUNT(*) > 1;   -- llaves duplicadas tras el seed
```

PARA LLEVAR

Cuatro cosas para correr sin miedo

— `(VALUES) EXCEPT SELECT` **= el conjunto que querés, menos el que tenés.** Declarado una vez, idempotente, el mismo archivo para todos los ambientes.

— **Nunca `NOT IN` contra una subconsulta con posibles NULL.** Un solo NULL envenena el filtro completo, sin error. `EXCEPT` es NULL-safe.

— **El 2627 por drift es un feature.** El script y la base dejaron de contar la misma historia; mejor enterarse en el deploy que en el cierre de mes.

— **Inserta faltantes, no actualiza cambiados.** Para un upsert real, la herramienta es `MERGE`, con sus propias trampas.

EL SCRIPT COMPLETO

La demo —con la base sintética de la Cooperativa MuuSQL incluida — está en el repo público de la serie. Corre en cualquier SQL Server 2008+ con un F5. O con veinte.

— *sigue en la serie* **Trucos de T-SQL para tu DW**.

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 002, 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