Disparadores (triggers) en PostgreSQL

Una de las funcionalidades disponibles en PostgreSQL son los denominados disparadores (triggers). En este artículo vamos a introducirnos en el mundo de los disparadores, como funcionan y como podemos empezar a utilizarlos.

Un disparador no es otra cosa que una acción definida en una tabla de nuestra base de datos y ejecutada automáticamente por una función programada por nosotros. Esta acción se activará, segun la definamos, cuando realicemos un INSERT, un UPDATE ó un DELETE en la susodicha tabla.

Un disparador se puede definir de las siguientes maneras:

- Para que ocurra ANTES de cualquier INSERT,UPDATE ó DELETE
- Para que ocurra DESPUES de cualquier INSERT,UPDATE ó DELETE
- Para que se ejecute una sola vez por comando SQL (statement-level trigger)
- Para que se ejecute por cada linea afectada por un comando SQL (row-level trigger)

Esta es la definición del comando SQL que se puede utilizar para definir un disparador en una tabla.

```
CREATE TRIGGER nombre { BEFORE | AFTER } { INSERT | UPDATE | DELETE [ OR ... ] }
    ON tabla [ FOR [ EACH ] { ROW | STATEMENT } ]
    EXECUTE PROCEDURE nombre de funcion ( argumentos )
```

Antes de definir el disparador tendremos que definir el procedimiento almacenado que se ejecutará cuando nuestro disparador se active.

El procedimiento almacenado usado por nuestro disparador se puede programar en cualquiera de los lenguajes de procedimientos disponibles, entre ellos, el proporcionado por defecto cuando se instala PostgreSQL, PL/pgSQL. Este lenguaje es el que utilizaremos en todos los ejemplos de este árticulo. Podeis encontrar mas información sobre procedimientos almacenados en el artículo "Procedimientos almacenados y PL/pgSQL"

Características y reglas a seguir

A continuación teneis algunas de las características y reglas más importantes a tener en cuenta, cuando definamos un disparador y/ó programemos un procedimiento almacenado que se vaya a utilizar por un disparador:

1. El procedimiento almacenado que se vaya a utilizar por el disparador debe de definirse e instalarse antes de definir el propio disparador.
2. Un procedimiento que se vaya a utilizar por un disparador no puede tener argumentos y tiene que devolver el tipo "trigger".
3. Un mismo procedimiento almacenado se puede utilizar por múltiples disparadores en diferentes tablas.
4. Procedimientos almacenados utilizados por disparadores que se ejecutan una sola vez per comando SQL (statement-level) tienen que devolver siempre NULL.
5. Procedimientos almacenados utilizados por disparadores que se ejecutan una vez per linea afectada por el comando SQL (row-level) pueden devolver una fila de tabla.

6. Procedimientos almacenados utilizados por disparadores que se ejecutan una vez per fila afectada por el comando SQL (row-level) y ANTES de ejecutar el comando SQL que lo lanzó, pueden:

    - Retornar NULL para saltarse la operación en la fila afectada.
    - Ó devolver una fila de tabla (RECORD)

7. Procedimientos almacenados utilizados por disparadores que se ejecutan DESPUES de ejecutar el comando SQL que lo lanzó, ignoran el valor de retorno, asi que pueden retornar NULL sin problemas.
8. En resumen, independendientemente de como se defina un disparador, el procedimiento almacenado utilizado por dicho disparador tiene que devolver ó bien NULL, ó bien un valor RECORD con la misma estructura que la tabla que lanzó dicho disparador.
9. Si una tabla tiene más de un disparador definido para un mismo evento (INSERT,UPDATE,DELETE), estos se ejecutarán en orden alfabético por el nombre del disparador. En el caso de disparadores del tipo ANTES / row-level, la file retornada por cada disparador, se convierte en la entrada del siguiente. Si alguno de ellos retorna NULL, la operación será anulada para la fila afectada.
10. Procedimientos almacenados utilizados por disparadores pueden ejecutar sentencias SQL que a su vez pueden activar otros disparadores. Esto se conoce como disparadores en cascada. No existe límite para el número de disparadores que se pueden llamar pero es responsabilidad del programador el evitar una recursión infinita de llamadas en la que un disparador se llame asi mismo de manera recursiva.

Otra cosa que tenemos que tener en cuenta es que, por cada disparador que definamos en una tabla, nuestra base de datos tendrá que ejecutar la función asociada a dicho disparador. El uso de disparadores de manera incorrecta ó inefectiva puede afectar significativamente al rendimiento de nuestra base de datos. Los principiantes deberian de usar un tiempo para entender como funcionan y asi poder hacer un uso correcto de los mismos antes de usarlos en sistemas en producción.

Variables especiales en PL/pgSQL

Cuando una función escrita en PL/pgSQL es llamada por un disparador tenemos ciertas variable especiales disponibles en dicha función. Estas variables son las siguientes:

NEW

Tipo de dato RECORD; Variable que contiene la nueva fila de la tabla para las operaciones INSERT/UPDATE en disparadores del tipo row-level. Esta variable es NULL en disparadores del tipo statement-level.

OLD

Tipo de dato RECORD; Variable que contiene la antigua fila de la tabla para las operaciones UPDATE/DELETE en disparadores del tipo row-level. Esta variable es NULL en disparadores del tipo statement-level.

TG_NAME

Tipo de dato name; variable que contiene el nombre del disparador que está usando la función actualmente.

TG_WHEN

Tipo de dato text; una cadena de texto con el valor BEFORE o AFTER dependiendo de como el disparador que está usando la función actualmente ha sido definido

TG_LEVEL

Tipo de dato text; una cadena de texto con el valor ROW o STATEMENT dependiendo de como el disparador que está usando la función actualmente ha sido definido

TG_OP

Tipo de dato text; una cadena de texto con el valor INSERT, UPDATE o DELETE dependiendo de la operación que ha activado el disparador que está usando la función actualmente.

TG_RELID

Tipo de dato oid; el identificador de objeto de la tabla que ha activado el disparador que está usando la función actualmente.

TG_RELNAME

Tipo de dato name; el nombre de la tabla que ha activado el disparador que está usando la función actualmente. Esta variable es obsoleta y puede desaparacer en el futuro. Usar TG_TABLE_NAME.

TG_TABLE_NAME

Tipo de dato name; el nombre de la tabla que ha activado el disparador que está usando la función actualmente.

TG_TABLE_SCHEMA

Tipo de dato name; el nombre de la schema de la tabla que ha activado el disparador que está usando la función actualmente.

TG_NARGS

Tipo de dato integer; el número de argumentos dados al procedimiento en la sentencia CREATE TRIGGER.

TG_ARGV[]

Tipo de dato text array; los argumentos de la sentencia CREATE TRIGGER. El índice empieza a contar desde 0. Indices inválidos (menores que 0 ó mayores/iguales que tg_nargs) resultan en valores nulos.

Obtenido de: http://www.postgresql.org.es/node/301
Consultado: 27 de enero de 2017