

Forzando pares unicos de columnas

El original en ingles : <http://people.planetpostgresql.org/greg/index.php?/archives/102-Enforcing-unique-column-pairs.html>

Alguien me pregunto como forzar la condicion que 2 columnas deban ser unicas, sin importar el orden, en otras palabras, dado 2 columnas, A y B, como asegurarse que un nuevo registro con A1 y B1 no sea igual a otro registro existente, sin importar el orden, entonces tendríamos que dar un error si existe un registro donde (A = A1 y B = B1) o si (A = B1 y B = A1).

Una forma facil para hacer esto es usar un indice unico, todo lo que hacemos es un indice funcional que ordene las dos columnas para nosotros en una forma consistente, entonces, si tenemos una tabla como esta:

```
CREATE TABLE sortme (  
  a INTEGER,  
  b INTEGER,  
  c INTEGER,  
  d TEXT,  
  e TEXT  
);
```

podríamos forzar un constraint como este:

```
CREATE UNIQUE INDEX unique_ab ON sortme(  
  (CASE WHEN a>b THEN a||'.'||b ELSE b||'.'||a END)  
);
```

Algunos puntos a notar:

- La clausa interior deberia contener padres alrededor de el (cualquier indice no simple en Postgresql tiene este requerimiento).
- Las columnas deben ser del mismo tipo y deben ser comparables con un operador <
- agregamos un separador que podria no ser encontrado en sus mismos valores de columna, en el caso, usamos un periodo. Mientras que algunas veces puede dejar de no tener un separador, es seguro y hace las cosas mucho mas claras siempre agregar uno.

Pero, digamos que quisieramos combinar 2 columnas de texto en la misma forma. Que usamos como separador? En lugar de preocuparnos sobre un separador valido, dejemos a un lado el problema poniendo los valores en un arreglo en lugar de concatenarlo junto, entonces un nuevo indice debera ser creado:

```
CREATE UNIQUE INDEX unique_de ON sortme(  
  (CASE WHEN d>e THEN ARRAY[d,e] ELSE ARRAY[e,d] END)  
);
```

Y el original con arrays se vuelve:

```
CREATE UNIQUE INDEX unique_ab ON sortme(  
  (CASE WHEN a>b THEN ARRAY[a,b] ELSE ARRAY[b,a] END)  
);
```

Verifiquemos si trabaja de acuerdo a lo esperado:

```
pp=# INSERT INTO sortme(a,b) VALUES (1,456);
pp=# INSERT INTO sortme(a,b) VALUES (14,56);
pp=# INSERT INTO sortme(a,b) VALUES (456,1);

INSERT 0 1
INSERT 0 1
ERROR:  duplicate key violates unique constraint "unique_ab"

pp=# INSERT INTO sortme(d,e) VALUES ('panama','canal');
pp=# INSERT INTO sortme(e,d) VALUES ('panama','canal');

INSERT 0 1
ERROR:  duplicate key violates unique constraint "unique_de"
```

Que tal si queremos limitar de modo que cualquier combinacion de tres columnas sea unica?. Debido a que hay mas posibles combinaciones, la parte funcional del indice funcional se vuelve un poco mas complejo, pero todavia leible con algun uso de espacios:

```
CREATE UNIQUE INDEX unique_abc ON sortme((
    CASE WHEN b BETWEEN a AND c THEN ARRAY[a,b,c]
         WHEN c BETWEEN a AND b THEN ARRAY[a,c,b]
         WHEN a BETWEEN b AND c THEN ARRAY[b,a,c]
         WHEN c BETWEEN b AND a THEN ARRAY[b,c,a]
         WHEN a BETWEEN c AND b THEN ARRAY[c,a,b]
         WHEN b BETWEEN c AND a THEN ARRAY[c,b,a]
    END
));

pp=# INSERT INTO sortme(a,b,c) VALUES (1,2,NULL);
pp=# INSERT INTO sortme(a,b,c) VALUES (1,2,NULL);
pp=# INSERT INTO sortme(a,b,c) VALUES (2,1,NULL);
pp=# INSERT INTO sortme(a,b,c) VALUES (1,2,3);
pp=# INSERT INTO sortme(a,b,c) VALUES (1,3,2);
pp=# INSERT INTO sortme(a,b,c) VALUES (2,1,3);
pp=# INSERT INTO sortme(a,b,c) VALUES (2,3,1);
pp=# INSERT INTO sortme(a,b,c) VALUES (3,1,2);
pp=# INSERT INTO sortme(a,b,c) VALUES (3,2,1);
pp=# INSERT INTO sortme(a,b,c) VALUES (1,7,2);

INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
ERROR:  duplicate key violates unique constraint "unique_abc"
ERROR:  duplicate key violates unique constraint "unique_abc"
ERROR:  duplicate key violates unique constraint "unique_abc"
ERROR:  duplicate key violates unique constraint "unique_abc"
ERROR:  duplicate key violates unique constraint "unique_abc"
INSERT 0 1
```

Note que los valores NULL previenen los primeros 3 registros de conflictos con otros, desde que NULL != NULL, los primeros 3 registros son considerados unicos uno del otro, entonces, despues que el "1,2,3" es agregado, todo el resto falla hasta el primer final, en el que introducimos un nuevo valor.

Traduccion por : Francisco Morosini