

Kuyruk tablosunda index'i canlı kümeye göre kurmak

Postgres 17 kuyruk tablosunda partial index: 41 kat küçük, generic planda uçurum, 15 dakikalık churn altında bloat ve hiç gelmeyen autovacuum.

01 POSTGRESQL · ÖLÇÜM

Kuyruk tablosunda index'i canlı kümeye göre kurmak

Tek makine, Postgres 17, sentetik tezgâh: 30 saniyelik ölçüm, planner ve 15 dakikalık churn.

Muhammet Şafak · 3 Ekim 2026



41x

02 10 MİLYON ÖLÜ SATIRDA

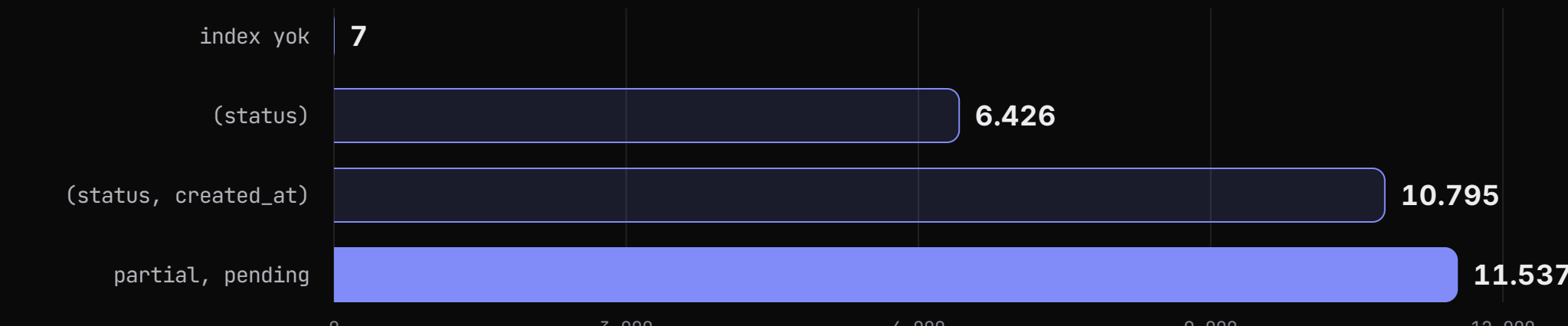
daha küçük index

Partial 7,6 MB, composite 310,4 MB. Throughput farkı yalnız %6,9: seçme sebebi hız değil, index'in tabloyla birlikte büyümemesi.

03 10M ÖLÜ SATIR, 5.000 CANLI SATIR

Index'siz kuyruk tablosu yok oluyor

tps



partial: (created_at) WHERE status = 'pending' · 8 istemci, 30 saniye, üç tekrarın medyanı · bench/run.sh

04 AYNI TABLO, AYNI İNDEK, AYNI SORGU

Partial index generic planda

11.752 tps, custom plan Index Scan, 0,68 ms	7 tps, generic plan Sort → Seq Scan; index hiç taranmadı	1.113 ms, generic gecikme Yaklaşık 1.635 kat; force_generic_plan ile zorlandı	11.416 Composite, generic Kanıtlanacak predicate'i yok, etkilenmedi
--	---	--	--

05 VARSAYILAN AUTO MOD, TEK OTURUM

Kırk çalıştırma, kırk custom plan

- 00

İlk beş çalıştırma: üçü de custom plan

tamam
- 01

Altıncı: composite generic'e geçer

tamam
- 02

Partial: 40'ın 40'ı custom

tamam
- 03

(status): 40'ın 40'ı custom

tamam

06 DÜZELTME

Bu koşullarda geçmedi

- ⚠ DÜZELTME

Önceki kaydımda 'Isındıkça bozulur' demiştim. Ölçüldüğünde yanlış çıktı; kırk çalıştırmanın kırkı custom plan kaldı.
- ❓ NEDEN

En olası açıklama: partial index'i kullanamayan generic planın tahmini maliyeti yüksek çıkıyor. Plan maliyetleri kaydedilmedi.
- ❓ BEDEL

Partial her çalıştırmada yeniden planlanır. Bu ölçekte fark ölçülemedi: 0,20 ms'ye 0,21 ms.

07 15 DK · 2.000 İŞ/SN · PG_RELATION_SIZE, 15 SN'DE BİR

Canlı küme sabit, index'ler büyüyor

SÜRE	PARTIAL	COMPOSITE
0 sn	0,125 MB	301 MB
286 sn	12,4 MB	344 MB
587 sn	25,3 MB	388 MB
888 sn	38,2 MB	427 MB
Artış	~305 kat	%42, +126 MB

08 HEDEF 2.000 İŞ/SN · 900. SANİYE

Otuz saniyede neredeyse eşit olan iki index, on beş dakikada ayrıldı

	composite	partial
BAŞLANGIÇ	2.000 tps · 0,52 ms	✓ 2.024 tps · 1,7 ms
900. SANİYE	1.204 tps · 61.533 ms	✓ 2.016 tps · 4.906 ms
BEKLEYEN İŞ	5.000 → 126.024	✓ 5.003 → 14.364
SONUÇ	Hedefi tutamadı; 427 MB belleğe sığmıyor	✓ Hedefi tuttu

09 EŞİK TABLOyla BÜYÜR

Autovacuum hiç gelmedi

1,75 Milyon ölü satır 15 dakikada 1.753.949	0 Autovacuum Bir kez bile koşmadı	≈2 Eşik, milyon satır 50 + 0,2 × 10.005.000 ≈ 2.001.050	17 Dakika, tahmini aralık Bu iş yüküne özgü; vacuum maliyeti ölçülmedi
--	--	--	---

10 NE YAPMALI

Kuyruk tablosunu canlı kümeye göre kurun

- ☒ Index'i kuyruğa kurun: partial (created_at) WHERE status = 'pending'
- ☒ Predicate'li index'e parametreyle ulaşmayın
- ☒ Autovacuum eşiğini tablodan koparın; etkisi ölçülmedi, 50.000 bir başlangıç
- ☒ Önce pending/done oranına, index boyutuna, n_dead_tup'a ve eşiğe bakın
- ☐ Kalıbı bırakın: tamamlananı hemen siliyor ya da archive taşıyorsanız, ORM status'u parametre bağlıyor ve generic plan zorlanıyorsa, tabloyu zamana göre partition'lıyorsanız



Tablo boyutu bir gürültüdür

github.com/muhammetsafak/pg-queue-bench

Index'i ve vacuum'u, işi yapan kümeye göre kurun. Sayılar tek makineden, Postgres 17'den ve tek erişim deseninden; gerçek bir kuyruğun aylarını temsil etmez.

