

PostgreSQL İndeksleme & Sorgu Optimizasyonu

İndeks Tipleri, EXPLAIN Analizi ve Planner Ayarları için Production Rehberi

⚡ PostgreSQL 15 / 16 / 17 uyumlu

Güncelleme: Ekim 2026 · Production-grade



P6

01 İndeks Tipleri

TEMEL



TİP	OPERATÖRLER	NE ZAMAN?
B-tree	= < > BETWEEN IN, LIKE 'abc%'	Varsayılan. Eşitlik, aralık, sıralama (ORDER BY).
Hash	=	Sadece eşitlik. PG10+ WAL-logged; nadiren B-tree'den iyi.
GIN	@> ? && @@	JSONB, array, full-text, pg_trgm (LIKE '%x%').
GiST	&& @> <->	Geometri (PostGIS), range, KNN, exclusion constraint.
SP-GiST	<< >> ~=	Dengesiz veri: IP (inet), quad-tree, prefix arama.
BRIN	= < >	Fiziksel sıralı dev tablolar (zaman serisi). Çok küçük.
Bloom	=	Extension. Çok kolonlu, rastgele eşitlik kombinasyonları.

</> TİP SEÇİMİ ÖRNEKLERİ

```
-- B-tree (varsayılan)
CREATE INDEX idx_orders_created ON orders (created_at);
```

```
-- JSONB içerik araması
CREATE INDEX idx_evt_payload ON events
USING gin (payload jsonb_path_ops);
```

```
-- LIKE '%kelime%' için trigram
CREATE EXTENSION IF NOT EXISTS pg_trgm;
CREATE INDEX idx_cust_name_trgm ON customers
USING gin (name gin_trgm_ops);
```

```
-- 1 milyar satırlık log tablosu: ~MB boyutunda
CREATE INDEX idx_logs_ts_brin ON logs
USING brin (ts) WITH (pages_per_range = 64);
```

```
-- Range çakışma (rezervasyon)
CREATE INDEX idx_booking_period ON bookings
USING gist (period);
```

✓ İNDEKS BOYUTLARI

```
SELECT indexrelid::regclass AS idx,
       pg_size_pretty(pg_relation_size(indexrelid)) AS size
FROM   pg_index WHERE indrelid = 'orders'::regclass;
```

📌 **BRIN kuralı:** Sadece kolon değeri ile fiziksel sıra arasında yüksek korelasyon varsa işe yarar. Kontrol: `pg_stats.correlation ≈ ±1`.

02 EXPLAIN Okuma Rehberi

KRİTİK



> DOĞRU KULLANIM

```
EXPLAIN (ANALYZE, BUFFERS, SETTINGS, VERBOSE)
SELECT * FROM orders WHERE customer_id = 42;

-- DML'i gerçekten çalıştırmadan analiz et
BEGIN; EXPLAIN ANALYZE DELETE FROM t WHERE ...; ROLLBACK;
```

PLAN DÜĞÜMÜ	ANLAMI / NE ZAMAN SORUN?
Seq Scan	Tüm tablo okunur. Küçük tabloda normal; büyük tabloda seçici filtre varsa indeks eksik.
Index Scan	İndeks + heap erişimi. Az satır döndüren sorgular için ideal.
Index Only Scan	Heap'e gitmez. Heap Fetches yüksekse → VACUUM.
Bitmap Heap Scan	Orta seçicilik; birden fazla indeks BitmapAnd/Or ile birleşir.
Nested Loop	Küçük dış küme + iç tarafta indeks. Dış küme büyükse felaket.
Hash Join	Büyük eşitlik join'leri. Batches > 1 → work_mem yetersiz.
Merge Join	İki taraf da sıralı (İndeks) olduğunda verimli.
Sort / HashAgg	external merge Disk: → diske taşıyor, work_mem artır.

👁 ÇIKTIDA NEYE BAKMALI?

```
Index Scan using idx_orders_cust on orders
(cost=0.43..8.45 rows=5 width=64)
(actual time=0.021..0.030 rows=48210 loops=1)
Index Cond: (customer_id = 42)
Rows Removed by Filter: 0
Buffers: shared hit=1204 read=3310
Planning Time: 0.112 ms
Execution Time: 41.733 ms
```

SİNYAL	TEŞHİS
rows tahmini ≠ actual	10x fark → istatistik eski: ANALYZE
Rows Removed by Filter	Yüksekse filtre kolonu indekste değil
shared read >> hit	Veri cache'te değil, disk I/O baskın
loops=N	Süre × loops = gerçek toplam maliyet

✓ **İyi Pratik:** Planı **FORMAT JSON** ile al, explain.dalibo.com veya explain.depsz.com ile görselleştir. PG17: **SERIALIZE** ve **MEMORY** seçenekleri eklendi.

03 İndeks Oluşturma Stratejileri



TEKNİK	KULLANIM
Composite	Eşitlik kolonları önce, aralık kolonu en sonda.
Partial	WHERE ile alt küme — küçük, hızlı indeks.
Expression	lower(email) gibi fonksiyonlu filtreler.
Covering	INCLUDE (PG11+) → Index Only Scan.
CONCURRENTLY	Yazmaları kilitmeden oluştur (production).

🔗 COMPOSITE — KOLON SIRASI

```
-- WHERE status = 'paid' AND created_at > now()-'7d'
CREATE INDEX idx_o_status_created
ON orders (status, created_at); -- ✓ eşitlik → aralık (tersi ✗)
```

▼ PARTIAL & EXPRESSION

```
-- Sadece aktif kayıtlar (%3 satır)
CREATE INDEX idx_o_pending ON orders (created_at)
WHERE status = 'pending';

-- Soft-delete tablolar için unique
CREATE UNIQUE INDEX uq_users_email ON users (lower(email))
WHERE deleted_at IS NULL;
-- Sorgu ifadeyi BİREBİR kullanmalı:
SELECT * FROM users WHERE lower(email) = 'a@b.com'
AND deleted_at IS NULL;
```

🌱 COVERING INDEX (INCLUDE)

```
CREATE INDEX idx_o_cust_cov ON orders (customer_id)
INCLUDE (total, status);
-- SELECT total, status ... WHERE customer_id = $1
-- → Index Only Scan (heap'e gitmez)
```

🔧 PRODUCTION'DA GÜVENLİ OLUŞTURMA

```
SET maintenance_work_mem = '2GB';
CREATE INDEX CONCURRENTLY idx_o_cust ON orders (customer_id);
-- İlerleme takibi (PG12+)
SELECT phase, blocks_done, blocks_total
FROM   pg_stat_progress_create_index;
```

⚠ **Dikkat:** **CONCURRENTLY** transaction bloğu içinde çalışmaz. Başarısız olursa geride **INVALID** indeks kalır — yazma yükü getirir: **DROP INDEX CONCURRENTLY** ile temizle.

04

Planner & Maliyet Parametreleri

PARAMETRE	VARSAYILAN	ÖNERİLEN
random_page_cost Rastgele sayfa okuma maliyeti. SSD'de düşür → indeks tercih edilir.	4.0	1.1 (SSD)
effective_cache_size OS + shared_buffers cache tahmini. Bellek ayırmaz.	4GB	RAM × 0.75
work_mem Sort/Hash işlemi başına bellek. Bağlantı × düğüm ile çarpılır!	4MB	16-64MB
hash_mem_multiplier Hash işlemleri için work_mem çarpanı.	2.0	2.0-4.0
effective_io_concurrency Bitmap Heap Scan prefetch derinliği.	1	200 (SSD)
default_statistics_target Histogram / MCV örnek sayısı.	100	100-500
jit OLTP'de kısa sorgulara derleme gecikmesi ekler.	on	off (OLTP)
max_parallel_workers_per_gather Sorgu başına paralel worker.	2	2-4
plan_cache_mode Prepared statement: generic / custom plan.	auto	auto

⚙️

POSTGRESQL.CONF (64 GB RAM · SSD · OLTP)

shared_buffers

=

16GB

effective_cache_size

=

48GB

work_mem

=

32MB

maintenance_work_mem

=

2GB

random_page_cost

=

1.1

effective_io_concurrency

=

200

default_statistics_target

=

200

jit

=

off

👤

OTURUM BAZLI TEST

-- Planner neden indeksi seçmiyor? Zorla & karşılaştıır

SET enable_seqscan = off;

EXPLAIN ANALYZE SELECT ...;

RESET enable_seqscan;

-- Tek bir ağır rapor için

SET LOCAL work_mem = '512MB';

-- sadece bu TX

⚠️ ⚠️ Dikkat:

enable_* parametreleri sadece teşhis içindir — global olarak kapatma.

work_mem global yükseltmesi OOM'a yol açabilir; rol bazlı ayarla: ALTER ROLE report SET work_mem='256MB'.

05

İstatistikler & ANALYZE

🔍

KOLON İSTATİSTİĞİNİ İNCELE

SELECT attname, null_frac, n_distinct, correlation,

most_common_vals, most_common_freqs

FROM pg_stats

WHERE tablename = 'orders' AND attname = 'status';

⚙️

KOLON BAZLI HASSASİYET

-- Çarpık dağılımlı kolon için daha fazla örnek

ALTER TABLE orders ALTER COLUMN customer_id

SET STATISTICS 1000;

ANALYZE orders;

🔗

EXTENDED STATISTICS — KORELASYONLU KOLONLAR

-- city ve zip bağımlı → planner satır sayısını

-- çarparak hatalı (çok düşük) tahmin eder

CREATE STATISTICS st_addr (dependencies, ndistinct, mcv)

ON city, zip_code FROM addresses;

ANALYZE addresses;

SELECT statistics_name, dependencies

FROM pg_stats_ext WHERE tablename = 'addresses';

🔄

SON ANALYZE & DEĞİŞEN SATIRLAR

SELECT relname, n_mod_since_analyze,

last_analyze, last_autoanalyze

FROM pg_stat_user_tables

ORDER BY n_mod_since_analyze DESC LIMIT 10;

-- Büyük tablo: daha sık autoanalyze

ALTER TABLE orders SET (

autovacuum_analyze_scale_factor = 0.02,

autovacuum_analyze_threshold = 1000);

⚠️ ⚠️ Dikkat:

pg_upgrade ve büyük toplu yüklemeler sonrası istatistikler eksik/eski olur. Hemen vacuumdb --all --analyze-in-stages çalıştır; aksi halde planlar çöker.

💡

Temp tablolar autovacuum tarafından analiz edilmez — doldurduktan sonra elle ANALYZE çalıştır.

06

Sorgu Anti-Pattern'leri

❗ SIK HATA

× KÖTÜ

✓ İYİ

WHERE date(created_at) = '2026-10-13'

WHERE created_at >= '2026-10-13'

AND created_at < '2026-10-14'

WHERE lower(email) = \$1

(indeks email üzerinde)

Expression index:

(lower(email))

WHERE phone = 5551234

(phone text → implicit cast)

WHERE phone = '5551234'

WHERE name LIKE '%ali%'

pg_trgm GIN indeksi

WHERE a = 1 OR b = 2

Ayrı indeksler (BitmapOr)

veya UNION ALL

WHERE id NOT IN (SELECT ...)

WHERE NOT EXISTS (...)

SELECT * ...

Sadece gereken kolonlar

COUNT(*) > 0 kontrolü

EXISTS (SELECT 1 ...)

▶️

OFFSET YERİNE KEYSSET PAGINATION

-- ✗ Sayfa 5000: 100.000 satır okunup atılır

SELECT * FROM orders ORDER BY id LIMIT 20 OFFSET 100000;

-- ✓ Sabit maliyet, indeks kullanır

SELECT id, total FROM orders

WHERE (created_at, id) < (:last_ts, :last_id)

ORDER BY created_at DESC, id DESC

LIMIT 20;

🔗

N+1 YERİNE TOPLU SORGU

-- ✗ Döngüde 1000 kez: WHERE id = \$1

-- ✓ Tek seferde:

SELECT * FROM products WHERE id = ANY(\$1::bigint[]);

🏗️

CTE & MATERIALIZATION (PG12+)

-- PG12+: CTE varsayılan olarak inline edilir

WITH recent AS NOT MATERIALIZED (

SELECT * FROM orders WHERE created_at > now() - '1 day'::interval

)

SELECT * FROM recent WHERE customer_id = 42;

⚠️ ⚠️ Dikkat:

NOT IN alt sorgusu tek bir NULL döndürürse sonuç boş olur — hem yanlış hem yavaş. Her zaman NOT EXISTS tercih et.

07

İndeks Bakımı & Teşhis

KULLANILMAYAN İNDEKSLER

```
SELECT s.schemaname, s.relname AS table, s.indexrelname AS idx,
       pg_size_pretty(pg_relation_size(s.indexrelid)) AS size
FROM   pg_stat_user_indexes s
JOIN   pg_index i USING (indexrelid)
WHERE  s.idx_scan = 0
       AND NOT i.indisunique AND NOT i.indisprimary
ORDER BY pg_relation_size(s.indexrelid) DESC;
```

TEKRARLAYAN (DUPLICATE) İNDEKSLER

```
SELECT indrelid::regclass AS table,
       array_agg(indexrelid::regclass) AS duplicates
FROM   pg_index
GROUP BY indrelid, indkey::text, indclass::text,
       coalesce(indexprs::text, ''), coalesce(indpred::text, '')
HAVING count(*) > 1;
```

INVALID İNDEKSLER

```
SELECT indexrelid::regclass FROM pg_index WHERE NOT indisvalid;
```

BLOAT GİDERME — REINDEX

```
-- PG12+: kilitlemeden yeniden oluşturun
REINDEX INDEX CONCURRENTLY idx_orders_created;
REINDEX TABLE CONCURRENTLY orders;

-- Bloat tahmini (pgstattuple extension)
CREATE EXTENSION IF NOT EXISTS pgstattuple;
SELECT avg_leaf_density, leaf_fragmentation
FROM   pgstattuple('idx_orders_created');
-- avg_leaf_density < 50 → REINDEX adayı
```

EN PAHALI SORGULAR (PG_STAT_STATEMENTS)

```
SELECT round(total_exec_time::numeric, 0) AS total_ms,
       calls, round(mean_exec_time::numeric, 2) AS mean_ms,
       rows, shared_blks_read, left(query, 80) AS query
FROM   pg_stat_statements
ORDER BY total_exec_time DESC LIMIT 10;
```

⚠ ⚠ Dikkat: idx_scan = 0 sayacı son istatistik sıfırlamasından beri geçerlidir. Silmeden önce replica'ları da kontrol et ve ay sonu raporlarını hesaba kat.

08

Optimizasyon Kontrol Listesi

YAVAŞ SORGU AKIŞI

KONTROL MADDESİ	ÖNCELİK
✓ pg_stat_statements kurulu, top 10 sorgu biliniyor ORDER BY total_exec_time DESC	KRİTİK
✓ EXPLAIN (ANALYZE, BUFFERS) ile plan incelendi tahmini rows ≈ actual rows ?	KRİTİK
✓ Foreign key kolonlarında indeks var JOIN ve ON DELETE CASCADE performansı	KRİTİK
✓ İstatistikler güncel pg_stat_user_tables.last_autoanalyze	KRİTİK
✓ random_page_cost SSD için ayarlı SHOW random_page_cost; -- 1.1	ÖNEMLİ
✓ Kullanılmayan / duplicate indeksler temizlendi pg_stat_user_indexes.idx_scan = 0	ÖNEMLİ
✓ INVALID indeks yok pg_index WHERE NOT indisvalid	ÖNEMLİ
✓ Disk'e taşan sort/hash yok log_temp_files = 0 · temp_blks_written	ÖNEMLİ
✓ Yavaş sorgu logu ve auto_explain aktif log_min_duration_statement = 500	ÖNERİLEN
✓ Index bloat düzenli izleniyor pgstattuple() · REINDEX CONCURRENTLY	ÖNERİLEN
KRİTİK hemen düzelt ÖNEMLİ bu sprint ÖNERİLEN planla	

İNDEKSİZ FOREIGN KEY'LERİ BUL

```
-- dbaguide.com · missing FK index finder
SELECT c.conrelid::regclass AS table,
       c.conname AS fk,
       pg_get_constraintdef(c.oid) AS definition
FROM   pg_constraint c
WHERE  c.contype = 'f'
       AND NOT EXISTS (
         SELECT 1 FROM pg_index i
         WHERE  i.indrelid = c.conrelid
               AND (i.indkey::int2[])[0]:cardinality(c.conkey)-1
                   OPERATOR(pg_catalog.@>) c.conkey);
```

AUTO_EXPLAIN — YAVAŞ PLANLARI OTOMATİK LOGLA

```
shared_preload_libraries = 'pg_stat_statements,auto_explain'
auto_explain.log_min_duration = '1s'
auto_explain.log_analyze     = on
auto_explain.log_buffers     = on
auto_explain.log_timing      = off # overhead ↓
auto_explain.sample_rate    = 0.1
log_temp_files              = 0
```

✓ Akış: pg_stat_statements → en pahalı sorgu → EXPLAIN (ANALYZE, BUFFERS) → tahmin hatası mı? (ANALYZE) · Seq Scan mı? (indeks) · Disk sort mu? (work_mem) → tekrar ölç.