

Apa Itu Index di Database?

Bayangkan sebuah **Buku Telepon** dengan jutaan nama.

Tanpa index, Anda harus mencari satu per satu dari halaman pertama (*Full Table Scan*).

Dengan index, Anda langsung melompat ke huruf yang dicari.

600 × 400

Cara Kerja & Mitos

Index ≠ Caching

- **Index:** Struktur data permanen di disk (Daftar Isi). Mempercepat *pencarian*.
- **Cache:** Salinan data sementara di RAM. Mempercepat *akses ulang*.

Di Dalam Postgres

Saat membuat index, Postgres membangun struktur (B-Tree) yang menyimpan:

1. Nilai kolom (misal: "Budi")
2. Pointer lokasi data (Tuple ID)

Mengapa Index Tidak Gratis?

Index adalah sebuah **Trade-off** (Pertukaran).

- ✗ **Memakan Ruang:** Setiap index butuh penyimpanan disk.
- ✗ **Memperlambat Penulisan:** Setiap `INSERT`, `UPDATE`, `DELETE`, database harus update tabel UTAMA dan semua INDEX-nya.

Analogi: Anda punya 5 daftar isi berbeda. Saat isi buku berubah, Anda harus merevisi kelima daftar isi tersebut.

600 × 400

Jenis Index Dasar



1. B-TREE (Default)

99% Kebutuhan Umum

- Exact match (`=`)
- Range (`<`, `>`, `BETWEEN`)
- Sorting (`ORDER BY`)



2. HASH Index

Spesifik & Cepat

- Hanya untuk pencarian (`=`)
- Sangat cepat untuk equality.
- Tidak bisa range / sorting.

Index Untuk Data Kompleks



GIN

Cocok untuk **JSONB**, **Array**, dan **Full Text Search**.



GiST

Ideal untuk data **Geospasial (PostGIS)** dan pencarian "tetangga".



BRIN

Untuk **Big Data** berurutan (misal: Log Timestamp). Sangat hemat space.

Teknik Optimasi Lanjutan

▼ Partial Index

```
WHERE is_active = true
```

Index hanya sebagian data. Ukuran lebih kecil, update lebih ringan.

Expression Index

```
INDEX ON users (lower(name))
```

Memungkinkan index dipakai saat query menggunakan fungsi.

Ringkasan: Kapan Butuh Index?

✓ PAKAI INDEX

- ✓ Kolom di klausa `WHERE`
- ✓ Kolom kunci `JOIN`
- ✓ Kolom `ORDER BY`

✗ JANGAN PAKAI

- ✗ Tabel sangat kecil
- ✗ Kolom jarang dicari
- ✗ Data sering berubah (High Churn)



Tip: Cheat Sheet Memilih Index

Tipe Data / Kasus	Jenis Index	Contoh Query
Standard (Text, Int, UUID)	B-Tree	<code>WHERE id = 100</code>
JSONB / Dokumen	GIN	<code>WHERE data @> '{"role": "admin"}'</code>
Pencarian Teks (Search)	GIN (tsvector)	<code>WHERE doc @@ to_tsquery('postgres')</code>
Lokasi (Lat/Long)	GiST	<code>WHERE ST_DWithin(loc, point, 1000)</code>
Log Data (Big Data)	BRIN	<code>WHERE created_at > '2023-01-01'</code>



Best Practice: Multi-Column Index

Aturan "Leftmost Prefix"

Urutan kolom dalam index **sangat penting**.

Index `(A, B)` berguna untuk:

- ✓ Query kolom `A` saja
- ✓ Query kolom `A` DAN `B`

Tapi **TIDAK** berguna untuk:

- ✗ Query kolom `B` saja

Contoh Kasus

```
-- Membuat Index Gabungan  
CREATE INDEX idx_name_age  
ON users (nama, umur);
```

```
-- ✓ Index Terpakai (Cepat)  
SELECT * FROM users WHERE nama = 'Budi';  
SELECT * FROM users WHERE nama = 'Budi' AND umur = 30;
```

```
-- ✗ Index TIDAK Terpakai (Lambat)  
SELECT * FROM users WHERE umur = 30;
```

Tip: Letakkan kolom yang paling sering difilter di urutan pertama.



Best Practice: Maintenance



Hapus Unused Index

Index yang tidak pernah dipakai oleh query hanya akan **memperlambat** proses INSERT/UPDATE. Hapus segera!



CONCURRENTLY

Gunakan `CREATE INDEX CONCURRENTLY` di production agar tidak mengunci tabel (no downtime).



REINDEX

Index bisa mengalami "bloat" (kembung). Lakukan `REINDEX` secara berkala untuk merapikan ukuran file index.

DEEP DIVE: STRUKTUR DATA

Apa yang Sebenarnya Disimpan?

- ✗ Bukan seluruh baris data.
- ✓ Hanya NILAI KOLOM + POINTER.

Index adalah file terpisah yang hanya mencatat alamat (TID - Tuple ID) menuju data aslinya di tabel heap.

Visualisasi: Index vs Tabel (Heap)

📁 File Tabel (Heap)

TID	ID	Nama	Umur
#1	1	Budi	30
#2	2	Andi	25
#3	3	Charlie	40



🚀 File Index (B-Tree)

Nama (Key)	Pointer (TID)
Andi	🔗 #2
Budi	🔗 #1
Charlie	🔗 #3

Menyimpan data lengkap, tidak terurut.

Hanya Nama & Alamat. Terurut A-Z.

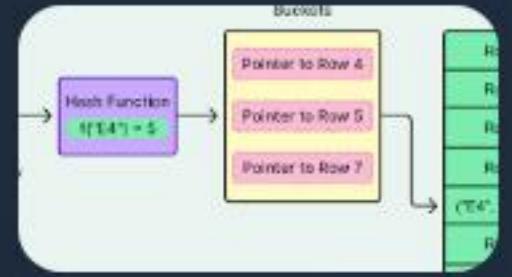
"Index tidak tahu umur Andi, dia hanya tahu dimana Andi tinggal (TID #2)."

Kesimpulan Teknis

**"Index itu milik
Kolom,
bukan milik Baris."**

- ✓ Index dibuat **per kolom** (atau kombinasi).
- ✓ Disimpan di **file terpisah**.
- ✓ Isinya: **Nilai + Pointer** ke row asli.
- ✓ Tabel tetap menyimpan row lengkap.

Image Sources



[https://substackcdn.com/image/fetch/\\$s_!L990!,f_auto,q_auto:good,f_l_progressive:steep/https%3A%2F%2Fsubstack-post-media.s3.amazonaws.com%2Fpublic%2Fimages%2F0e131acb-5f1a-4cdb-aae3-fbc1220e7549_2602x1496.png](https://substackcdn.com/image/fetch/$s_!L990!,f_auto,q_auto:good,f_l_progressive:steep/https%3A%2F%2Fsubstack-post-media.s3.amazonaws.com%2Fpublic%2Fimages%2F0e131acb-5f1a-4cdb-aae3-fbc1220e7549_2602x1496.png)

Source: blog.bytebytogo.com