



JURUSAN TEKNOLOGI INFORMASI

Mata Kuliah Basis Data Lanjut

## 08. Full-Text Search dan JSONB

# Topik



1. Konsep Full-Text Search (FTS)
2. Implementasi Full-Text Search
3. Mengenal JSON dan JSONB
4. Analisis dan Manipulasi Data JSONB
5. Latihan Full-Text Search
6. Latihan Manipulasi dan Query JSONB

# *Topik-1: Konsep Full-Text Search (FTS)*

---

---

# 1. Konsep Full-Text Search (FTS)

- Full-Text Search adalah mekanisme pencarian teks “selevel mesin pencari” yang memahami **kata**, **bentuk kata**, dan **relevansi**, bukan sekadar mencocokkan substring.
- Di PostgreSQL, FTS dibangun di atas dua tipe utama:
  - **tsvector** → representasi indeks dari dokumen (teks)
  - **tsquery** → representasi kueri pencarian
- FTS mendukung:
  - bahasa (mis. 'indonesian'),
  - stemming (mengurai “pembelajaran”, “belajar”, “dipelajari” ke akar yang sama),
  - stopwords (abaikan kata umum),
  - operator pencarian (AND/OR/NOT, frasa/urutan),
  - ranking (mengurutkan hasil paling relevan).

# 1. Konsep Full-Text Search (FTS)

## Mengapa tidak cukup dengan LIKE?

- **FTS  $\neq$  LIKE:** FTS paham bentuk kata, punya ranking, dan cepat dengan indeks.
- LIKE '%kata%' hanya cocok untuk:
  - **Substring literal** (tidak paham bentuk kata “belajar/pembelajaran”).
  - **Tanpa ranking** (semua cocok dianggap sama).
  - **Lambat** di teks panjang kecuali sangat terspesifikasi, dan **tidak memanfaatkan indeks** secara efektif untuk pola %...%.
- FTS unggul karena:
  - **Memahami bahasa** (stemming, stopwords),
  - **Lebih cepat** dengan indeks GIN/GiST pada tsvector,
  - **Bisa menghitung skor relevansi** (mis. kemunculan di judul > deskripsi),
  - **Mendukung operator kaya** (frasa, jarak, prefix, dsb.).

## 1. Konsep Full-Text Search (FTS)

### Pipeline FTS: Dari teks sampai ranking (1/2)

- **C. Matching + Ranking**
  - **Pencocokan:** tsvector @@ tsquery (true/false).
  - **Ranking:** ts\_rank / ts\_rank\_cd memberi skor relevansi (frekuensi, kepadatan, posisi).
  - **Highlighting** (opsional, untuk UI): ts\_headline memberi penekanan (mis. <b>kata</b>).

## 1. Konsep Full-Text Search (FTS)

### Pipeline FTS: Dari teks sampai ranking (2/2)

- Bayangkan dua pipa proses:
  - **dokumen** (saat menyimpan) dan **kueri** (saat mencari).
- **A. Pipeline dokumen → tsvector**
  - Teks mentah (mis. judul\_proposal || ' ' || deskripsi) melewati langkah:
  - **Normalisasi**: huruf kecil, buang tanda baca tertentu.
  - **Tokenisasi**: pecah jadi token/kata.
  - **Stopwords removal**: hapus kata umum (“dan”, “yang”, dll.).
  - **Stemming**: turunkan ke akar kata (“pembelajaran” → “ajar”).
  - **Indexing**: simpan sebagai tsvector (kumpulan **lexeme** + posisi kemunculan).
- **B. Pipeline kueri → tsquery**
  - Kata/frasa yang diketik pengguna diproses serupa lalu dibentuk tsquery:
    - **Operator**: & (AND), | (OR), ! (NOT), <-> (jarak 1 kata), <N> (jarak N), :\* (prefix).
    - Contoh: 'tugas & akhir', 'jaringan <-> intrusi', 'rekomendasi:\*'.

# 1. Konsep Full-Text Search (FTS)

## Komponen Inti

- **Text Search Configuration:**
  - Aturan bahasa/dictionary, mis. 'indonesian'.
  - Mendukung berbagai macam bahasa internasional lainnya.
- **Matrikulasi bobot:**
  - `setweight()` — mis. bobot tinggi untuk judul (A), lebih rendah untuk deskripsi (D).
- **Indeks:**
  - GIN paling umum untuk FTS.
    - Model **kolom materialisasi**: simpan `search_vector` dan buat indeks GIN.
    - Atau **expression index** langsung dari `to_tsvector(...)`.



# *Topik-2: Implementasi Full-Text Search*

---

---

## 2. Implementasi Full-Text Search

- Seperti sudah dijelaskan sebelumnya, dalam FTS, terdapat dua fungsi PostgreSQL utama yang menjadi kunci yaitu: **to\_tsvector()** dan **to\_tsquery()**.
- **Fungsi Utama: to\_tsvector() dan to\_tsquery()**

Fungsi	Deskripsi	Contoh Output
to_tsvector()	Mengubah teks biasa menjadi representasi indeks ( <i>vector of lexemes</i> )	'keamanan':2 'jaringan':1 'siber':3
to_tsquery()	Mengubah teks kueri menjadi bentuk logika pencarian	'jaringan' & 'keamanan'

- Singkatnya:
  - **to\_tsvector()** dipakai untuk teks sumber (kolom data), sedangkan;
  - **to\_tsquery()** dipakai untuk kata pencarian.

## 2. Implementasi Full-Text Search

### Contoh Dasar



- Membentuk tsvector:

```
SELECT to_tsvector('indonesian', 'Sistem deteksi intrusi jaringan dan keamanan siber');
```

- Hasil:

```
'deteksi':2 'intrusi':3 'jaringan':4 'keamanan':6 'siber':7 'sistem':1
```

- Membentuk tsquery:

```
SELECT to_tsquery('indonesian', 'keamanan & jaringan');
```

- Artinya, setiap kata (*lexeme*) disimpan dalam bentuk akar kata & posisi kemunculannya.

- Hasil:

```
'keamanan' & 'jaringan'
```

- Operator & artinya **AND** (dua kata harus muncul bersama).

## 2. Implementasi Full-Text Search

# Menghubungkan Dokumen dan Query

- Pencocokan dilakukan dengan operator @@:

```
SELECT to_tsvector('indonesian', 'sistem keamanan jaringan') @@ to_tsquery('indonesian', 'keamanan & jaringan');
```

- Hasil:

t

- (t = true → cocok)

## 2. Implementasi Full-Text Search

### Contoh Kasus: Pencarian Proposal

- Kita gunakan tabel **proposal** yang berisi judul\_proposal dan deskripsi.
- a) Tambahkan kolom search\_vector

```
ALTER TABLE proposal ADD COLUMN search_vector tsvector;
```

- b) Isi data awal dengan FTS

```
UPDATE proposal  
SET search_vector =  
    setweight(to_tsvector('indonesian', coalesce(judul_proposal, '')), 'A') ||  
    setweight(to_tsvector('indonesian', coalesce(deskripsi, '')), 'D');
```

- Fungsi **setweight()** memberi bobot berbeda untuk kolom penting, misalnya:
  - 'A' = paling penting (judul)
  - 'D' = kurang penting (deskripsi)

## 2. Implementasi Full-Text Search

### Contoh Kasus: Pencarian Proposal

- c) Membuat indeks GIN

```
CREATE INDEX idx_proposal_search  
ON proposal  
USING gin(search_vector);
```

- GIN (Generalized Inverted Index) adalah indeks khusus yang sangat efisien untuk pencarian teks penuh.
- d) pencarian dengan to\_tsquery()

```
SELECT  
    proposal_id,  
    judul_proposal,  
    ts_rank(search_vector, to_tsquery('indonesian', 'jaringan & keamanan')) AS rank  
FROM proposal  
WHERE search_vector @@ to_tsquery('indonesian', 'jaringan & keamanan')  
ORDER BY rank DESC;
```

- Hasil akan menampilkan proposal dengan kata “*jaringan*” dan “*keamanan*” di judul/deskripsi, diurutkan berdasarkan skor relevansi (rank).

## 2. Implementasi Full-Text Search Operator Logika TS



Operator	Arti	Contoh	Keterangan
&	AND	'jaringan & keamanan'	Harus mengandung dua kata
`	`	OR	'ai
!	NOT	'!keamanan'	Tidak mengandung kata
<->	Phrase	'jaringan <-> intrusi'	Berurutan (jarak 1 kata)
.*	Prefix	'rekomendasi:.*'	Kata berawalan "rekomendasi"

- Contoh Query:

```
SELECT
  proposal_id,
  judul_proposal,
  ts_rank_cd(search_vector, to_tsquery('indonesian', 'sistem & rekomendasi')) AS rank
FROM proposal
WHERE search_vector @@ to_tsquery('indonesian', 'sistem & rekomendasi')
ORDER BY rank DESC;
```

- ts\_rank\_cd()** adalah versi ranking *coverage density* yang mempertimbangkan seberapa padat kata kueri muncul di dokumen.

# *Topik-3: Mengenal JSON dan JSONB*

---

---



### 3. Mengenal JSON dan JSONB

- **JSON (JavaScript Object Notation)** adalah format data ringan berbentuk pasangan key: value.
- PostgreSQL mendukung dua tipe penyimpanan:
  - JSON (disimpan sebagai teks mentah)
  - JSONB (*Binary JSON*) — versi yang sudah diurai dan dioptimasi

- Contoh data JSON:

```
{  
  "merek": "MikroTik",  
  "model": "CCR1009",  
  "fitur": ["VPN", "Firewall", "QoS"],  
  "port": {"ethernet": 8, "sfp": 1},  
  "status": "aktif"  
}
```

- PostgreSQL memungkinkan kolom seperti ini disimpan dalam satu field JSONB, tanpa perlu membuat banyak kolom relasional.

### 3. Mengenal JSON dan JSONB

## Perbedaan JSON vs JSONB



Aspek	JSON	JSONB	Aspek
Penyimpanan	Disimpan sebagai teks mentah (string)	Disimpan dalam format biner terstruktur	Penyimpanan
Kecepatan Akses	Lambat (harus diurai ulang tiap query)	Cepat (langsung bisa diakses)	Kecepatan Akses
Duplikasi Key	Diperbolehkan (tetap disimpan)	Dihapus, hanya satu key yang disimpan	Duplikasi Key
Urutan Key	Dipertahankan seperti aslinya	Diabaikan (tidak berurutan)	Urutan Key
Indeks GIN	✗ Tidak bisa	✓ Bisa	Indeks GIN
Ukuran Storage	Sedikit lebih besar	Sedikit lebih efisien	Ukuran Storage
Fungsi Manipulasi	Terbatas	Lengkap dan cepat	Fungsi Manipulasi

- **Best Practice:**

- Gunakan JSONB untuk semua kasus nyata (analisis, filter, query cepat).
- Gunakan JSON hanya jika kita perlu menyimpan teks JSON *mentah* (tanpa parsing).

### 3. Mengenal JSON dan JSONB

## Keunggulan PostgreSQL JSONB dibanding MySQL JSON

- Berikut ini ringkasan perbandingan penanganan JSON di PostgreSQL vs MySQL.

Fitur	PostgreSQL JSONB	MySQL JSON
Operator Akses	Lengkap: ->, ->>, #>, @>, ?, `?`	, ?&`
Indeks JSON	GIN / GiST native	Harus menggunakan <i>generated column</i>
Manipulasi JSON	Fungsi insert, replace, delete langsung	Lebih terbatas
Query Bersarang (CTE + JSON)	Sangat fleksibel	Kurang mendukung
Kinerja Query	Cepat karena binary storage	Lebih lambat (text parsing)
Kombinasi SQL + JSON	Sangat kuat (fitur FTS, CTE, JOIN)	Terbatas di beberapa versi MySQL

- Secara umum PostgreSQL JSONB memungkinkan developer membuat sistem **semi-terstruktur** (data fleksibel tapi tetap bisa diquery SQL) dengan lebih baik.

### 3. Mengetahui JSON dan JSONB

## Operator JSONB yang Paling Sering Digunakan



Operator	Kegunaan	Contoh	Hasil
->	Mengambil <b>nilai JSON</b> (hasilnya tetap tipe JSON)	spesifikasi -> 'merek'	"MikroTik"
->>	Mengambil <b>nilai teks</b> (hasilnya tipe TEXT)	spesifikasi ->> 'merek'	MikroTik
#>	Mengambil <b>nested object</b> (jalur bertingkat)	spesifikasi #> '{port, ethernet}'	8
@>	Mengecek apakah JSON <b>mengandung</b> substruktur tertentu	spesifikasi @> '{"fitur": ["VPN"]}'	true
?	Mengecek apakah ada key tertentu	spesifikasi ? 'gpu'	true/false
`?	`	Mengecek apakah key-nya salah satu dari daftar	`spesifikasi ?
?&	Mengecek apakah <b>semua key</b> ada	spesifikasi ?& array['merek','model']	true
jsonb_array_elements()	Memecah array menjadi baris	jsonb_array_elements(spesifikasi->'fitur')	baris per fitur
jsonb_set()	Memperbarui nilai di JSONB	jsonb_set(spesifikasi, '{status}', '"tidak aktif"')	JSONB baru

### 3. Mengenal JSON dan JSONB

## Tips Performa



- **Gunakan indeks GIN pada kolom JSONB**
  - Untuk mempercepat operator @>, ?, `?
- **Gunakan operator spesifik, bukan fungsi umum**
  - Operator ->> lebih cepat daripada CAST()
- **Gunakan jsonb\_path\_ops untuk indeks ringan**
  - Cocok untuk pencarian subset sederhana
- **Gunakan jsonb\_set() daripada UPDATE manual**
  - Lebih efisien dan aman
- **Gunakan EXPLAIN ANALYZE untuk evaluasi**
  - Cek apakah indeks JSONB benar-benar digunakan

# *Topik-4: Analisis dan Manipulasi Data JSONB*

---

---

## 4. Analisis dan Manipulasi Data JSONB

- Dalam dunia nyata, tidak semua data punya struktur kolom yang tetap. Misalnya:
  - Produk dengan atribut berbeda-beda per kategori,
  - Spesifikasi alat laboratorium yang variatif,
  - Metadata penelitian yang beragam antar proposal.
- JSONB memungkinkan kita **menyimpan semua itu dalam satu kolom**, dan PostgreSQL memberi kita **alat untuk mengekstrak, memfilter, dan mengolahnya secara efisien**.
- PostgreSQL dengan JSONB memberi kemampuan data exploration yang sangat fleksibel.
- Kita bisa meng-query, memfilter, menganalisis, bahkan memperbarui isi JSON langsung di SQL — tanpa memerlukan NoSQL eksternal.
- Fitur ini menjadikan PostgreSQL sangat *powerful* untuk aplikasi modern yang datanya campuran antara terstruktur dan semi-terstruktur.

## 4. Analisis dan Manipulasi Data JSONB

### Konsep Dasar Manipulasi JSONB

- Sebelum menganalisis, kita harus pahami dua konsep inti operasi JSONB di PostgreSQL:
- **◆ Ekstraksi Nilai (Extraction)**
  - `->` : ambil **objek/array** (hasil tetap JSON)
  - `->>` : ambil **nilai teks**
  - `#>` : navigasi **nested path** (mis. `'{a,b,c}'`)
- **◆ Transformasi dan Filter**
  - `jsonb_array_elements()` → memecah array JSON jadi baris tabel
  - `jsonb_each()` → memecah objek JSON jadi pasangan *key-value*
  - `jsonb_set()` → mengganti atau menambah elemen
  - `@>` → filter data JSONB yang **mengandung** pola tertentu



## 4. Analisis dan Manipulasi Data JSONB

### Contoh Kasus: Inventaris Lab

- Perhatikan data di dalam tabel **inventaris\_lab**:

58 `SELECT * FROM inventaris_lab;`

Data Output Messages Notifications

Showing rows: 1 to 10 Page No: 1 of 1

	inventaris_id [PK] integer	lab_kode character varying (50)	nama_barang text	spesifikasi jsonb
1	1	NCS	Router MikroTik CCR10...	{ "port": { "sfp": 1, "ethernet": 8 }, "tags": [ "network", "router" ], "fitur": [ "BGP", "VPN", "Firewall" ], "merek": "MikroTik", "model": "
2	2	NCS	Firewall Appliance	{ "tags": [ "security", "firewall" ], "fitur": [ "Stateful Firewall", "IDS/IPS", "OpenVPN" ], "merek": "Netgate", "model": "pfSense-Plu
3	3	IVSS	Kamera AI Vision	{ "fitur": [ "object-detection", "object-tracking" ], "merek": "Sony", "sensor": "CMOS", "dataset": [ "COCO", "Custom" ], "aksesori"
4	4	IVSS	GPU Workstation	{ "os": "Ubuntu 22.04", "cpu": "Intel i9-13900K", "gpu": "RTX 4090", "merek": "NVIDIA", "tools": [ "CUDA", "cuDNN", "Docker" ], "
5	5	SE	Server CI/CD GitLab	{ "cpu": "AMD EPYC", "backup": { "jadwal": "harian", "retensi_hari": 14 }, "ram_gb": 256, "layanan": [ "GitLab", "Runner", "Registr
6	6	DT	Cluster Mini Big Data	{ "nodes": 3, "network": "10GbE", "software": [ "Hadoop", "Spark" ], "management": { "monitoring": "Prometheus+Grafana", "or

- Pada data **spesifikasi**, isinya JSON. Salah satu contohnya:

```
{
  "merek": "MikroTik",
  "model": "CCR1009",
  "fitur": [ "BGP", "VPN", "Firewall" ],
  "port": { "ethernet": 8, "sfp": 1 },
  "status": "aktif"
}
```







## 4. Analisis dan Manipulasi Data JSONB

### Contoh Kasus: Inventaris Lab - Ekstraksi











- Contoh-1: Ambil kontak supplier (nested object)

```
SELECT
  nama_barang,
  spesifikasi #> '{supplier, kontak, email}' AS email_supplier
FROM inventaris_lab
WHERE spesifikasi ? 'supplier';
```

- Hasil:

Data Output Messages Notifications		
<div><div>≡+</div><div></div><div>▼</div><div></div><div>▼</div><div></div><div></div><div></div><div></div><div>SQL</div></div>		
	nama_barang text	email_supplier jsonb
1	Router MikroTik CCR1009	"sales@netindo.id"
2	Kit IoT ESP32	"support@makerhub.id"

- ```
SELECT
    nama_barang,
    spesifikasi #> '{port, ethernet}' AS jumlah_ethernet
FROM inventaris_lab
WHERE lab_kode = 'NCS';
```

- | Data Output                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | Messages                                                                                                                                                                                                                                                                                                                                        | Notifications                   |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------|
|       |     |                                 |
|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      | <b>nama_barang</b><br>text                                                                                                                                                                                                                                                                                                                      | <b>jumlah_ethernet</b><br>jsonb |
| 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | Router MikroTik CCR1009                                                                                                                                                                                                                                                                                                                         | 8                               |
| 2                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | Firewall Appliance                                                                                                                                                                                                                                                                                                                              | [null]                          |

## 4. Analisis dan Manipulasi Data JSONB

### Contoh Kasus: Inventaris Lab - Transform

- CTE memudahkan membaca atau memecah struktur JSONB kompleks sebelum diolah lebih lanjut.
  - Kita bisa “mengeluarkan” elemen array atau nested object ke bentuk tabel sementara.
- **Contoh-3: Ekstrak setiap fitur alat ke dalam baris terpisah**

```
WITH fitur AS (  
  SELECT  
    lab_kode,  
    nama_barang,  
    jsonb_array_elements_text(spesifikasi->'fitur') AS fitur  
  FROM inventaris_lab  
  WHERE spesifikasi ? 'fitur'  
)  
SELECT * FROM fitur WHERE fitur ILIKE '%firewall%';
```

- **Hasil:**

| Data Output Messages Notifications |                                    |                         |                   |
|------------------------------------|------------------------------------|-------------------------|-------------------|
|                                    | lab_kode<br>character varying (50) | nama_barang<br>text     | fitur<br>text     |
| 1                                  | NCS                                | Router MikroTik CCR1009 | Firewall          |
| 2                                  | NCS                                | Firewall Appliance      | Stateful Firewall |

## 4. Analisis dan Manipulasi Data JSONB

### *Best Practices*

- Gunakan **CTE** untuk data bersarang
  - Lebih mudah dibaca dan di-debug
- Gunakan **GIN index** pada kolom JSONB
  - Percepat filter @> dan ?
- Gunakan **operator** dibanding fungsi
  - Operator ->> lebih cepat dari jsonb\_extract\_path\_text()
- Gunakan **jsonb\_set()** dan **jsonb\_insert()**
  - Untuk update/penambahan aman
- Gabungkan **JSONB + SQL relasional**
  - Kombinasi paling kuat untuk semi-structured data
- **Hindari JSONB berlebihan** di tabel utama
  - Simpan JSONB hanya untuk atribut fleksibel atau opsional

# *Topik-5: Latihan Full-Text Search*

---

---

## 5. Latihan Full-Text Search

- Penjelasan topik ini ada di **Petunjuk Praktikum**.

# *Topik-6: Latihan Manipulasi dan Query JSONB*

---

---



## 6. Latihan Manipulasi dan Query JSONB

- Penjelasan topik ini ada di **Petunjuk Praktikum**.

# Pertanyaan?



*Terima Kasih*

# Tugas



- Selesaikan langkah-langkah pada petunjuk praktikum.

# Referensi



[1] -