

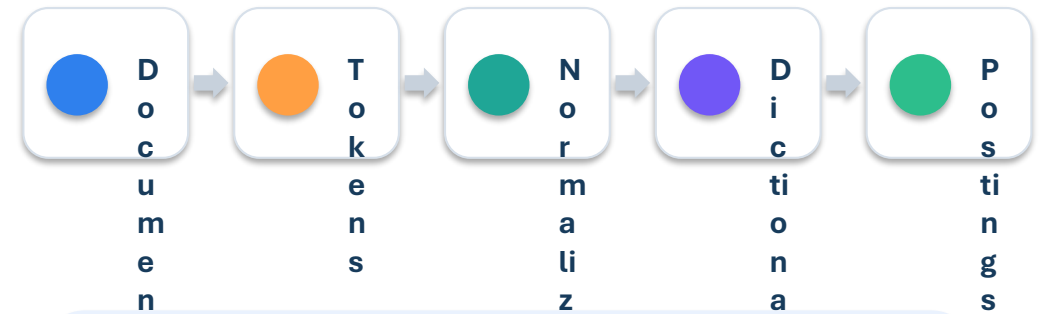
WEB INFORMATION RETRIEVAL

Pertemuan 4

Indexing

Tokenization • Normalization • Stopword • Stemming • Inverted Index •
Postings • Positional Index

FROM DOCUMENTS TO SEARCHABLE STRUCTURE



Indexer mengubah koleksi dokumen menjadi struktur data yang memungkinkan pencarian cepat tanpa memindai seluruh dokumen setiap kali query datang.

PROCESS → ORGANIZE → INDEX → RETRIEVE

Capaian Pertemuan 4

Kemampuan yang harus ditunjukkan setelah pembelajaran.

Sub-CPMK 4

Mampu menjelaskan dan mensimulasikan proses indexing: tokenisasi, normalisasi, stopword removal, stemming, serta pembangunan inverted index. (C3, A3)

Indikator

Mahasiswa mampu membentuk vocabulary, postings list, dan positional index dari koleksi dokumen sederhana.

Bukti Belajar

Simulasi manual indexing, analisis kasus search portal, latihan query pada inverted index, dan Quiz 2.

TARGET PRAKTIS

- Menjelaskan fungsi indexer dalam arsitektur Web IR.
- Menerapkan tokenization dan normalization yang tepat.
- Menjelaskan trade-off stopwords, stemming, dan lemmatization.
- Membangun inverted index dan postings list untuk query sederhana.

Posisi Pertemuan 4 dalam Semester

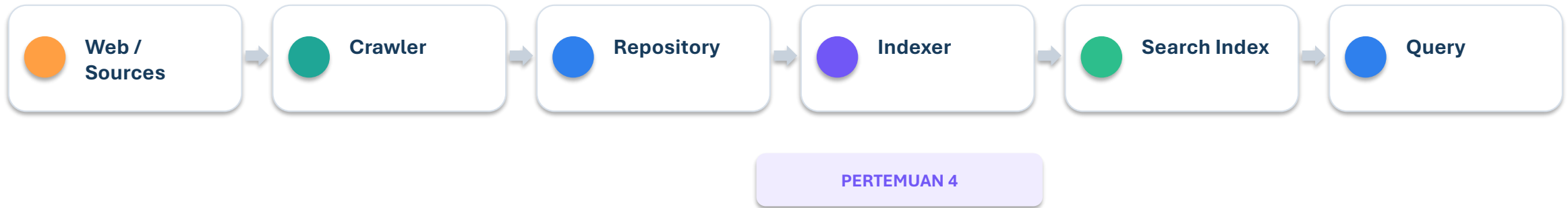
Crawler menemukan dokumen; indexer menjadikannya dapat dicari.



Fokus: bagaimana dokumen mentah direpresentasikan sebagai vocabulary + postings sehingga retrieval bisa dilakukan dalam milidetik.

Di Mana Indexer Berada?

Indexer adalah jembatan antara koleksi dokumen dan query-time retrieval.



Tanpa index, sistem harus membaca semua dokumen untuk setiap query. Dengan index, sistem langsung menuju dokumen kandidat yang mengandung term terkait.

Pertanyaan desain: apa yang menjadi token? bagaimana variasi bentuk kata ditangani? informasi apa yang disimpan di postings?

01

Fondasi Indexing

Indexing mengubah teks mentah menjadi representasi terstruktur untuk retrieval yang efisien.

Apa Itu Indexing?

Proses menganalisis dokumen dan membangun struktur akses untuk pencarian.

1 Parse

Ekstrak teks, field, metadata, dan struktur dokumen.

2 Tokenize

Pecah teks menjadi unit term yang akan diproses.

3 Normalize

Samakan variasi bentuk yang seharusnya diperlakukan serupa.

4 Index

Bangun dictionary, postings, posisi, dan statistik retrieval.

OUTPUT UTAMA: searchable representation — bukan sekadar salinan dokumen.

Mengapa Tidak Memindai Semua Dokumen?

Sequential scan bekerja, tetapi skalanya buruk ketika koleksi membesar.

Tanpa Index

Query “data mining” → buka dokumen 1, 2, 3, ... N → cek isi satu per satu.

Kompleksitas query meningkat seiring ukuran koleksi dan ukuran dokumen.

Dengan Inverted Index

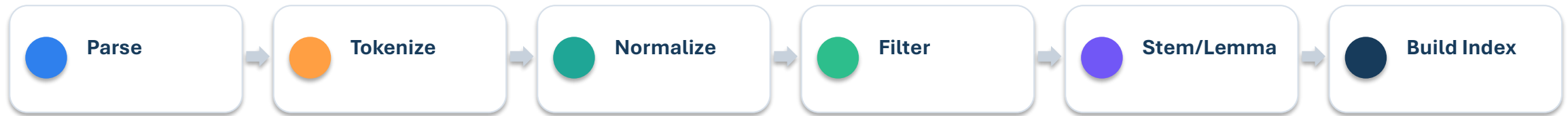
Query “data mining” → dictionary → postings(data) n postings(mining) → kandidat dokumen.

Kerja query dipindahkan dari “scan koleksi” menjadi “akses struktur index”.

INDEXING = pekerjaan di depan agar pencarian saat query jauh lebih cepat.

Alur Kerja Indexer

Setiap tahap dapat memengaruhi recall, precision, storage, dan latency.



Contoh dokumen

“Mahasiswa mempelajari pencarian informasi berbasis web pada portal kampus.”

Contoh keluaran term



Catatan: hasil akhir bergantung kebijakan tokenization, stopwords, dan stemming yang dipilih.

Tokenization

Tokenization menentukan unit dasar yang akan masuk ke vocabulary.

“Belajar Web-IR di UPI-Y.A.I., 2026!”



belajar

web-ir

di

upi-y.a.i.

2026

Pertanyaan tokenization

- Apakah “web-ir” satu token atau dua?
- Apakah singkatan dan tanda titik dipertahankan?
- Bagaimana email, URL, hashtag, angka, emoji, dan kode produk?
- Apakah aturan yang sama berlaku untuk semua bahasa?

Unicode & Multilingual Text

Web IR tidak hanya memproses teks berbahasa Inggris.

Script & Word Boundaries

Bahasa tertentu tidak memakai spasi sebagai pemisah kata secara konsisten; segmentasi memerlukan aturan/model khusus.

Unicode Variants

Karakter yang terlihat sama dapat memiliki representasi kode berbeda; normalization Unicode membantu konsistensi.

Mixed Content

Dokumen web dapat memuat bahasa campuran, URL, kode, simbol, angka, dan emoji sekaligus.

ATURAN TOKENIZATION HARUS SESUAI DOMAIN, BAHASA, DAN JENIS QUERY.

Normalization

Tujuan: menyatukan variasi permukaan tanpa menghapus perbedaan yang bermakna.

Case folding

Information → information

Unicode

varian encoding → bentuk canonical

Punctuation

“data,” → data

Accents

café ↔ cafe? tergantung bahasa/domain

Numbers

1.000 / 1000 / 1k → kebijakan domain

NORMALIZE TERLALU AGRESIF → makna berbeda bisa tercampur. TERLALU LEMAH → recall turun.

Case Folding & Accent Handling

Kebijakan sederhana pun memiliki trade-off.

Case Folding

“IR”, “Ir”, “ir” dapat disamakan untuk pencarian umum. Namun beberapa entitas/kode bersifat case-sensitive.

Accent Folding

“café” dan “cafe” dapat dipetakan bersama untuk toleransi query, tetapi bahasa tertentu membedakan karakter beraksen.

Domain Rule

Nomor part, gene symbol, legal citation, atau product SKU mungkin memerlukan kebijakan khusus.

Prinsip desain: normalization sebaiknya diuji terhadap query nyata dan relevance judgement — bukan sekadar “dibersihkan sebanyak mungkin”.

Stopword Removal

Kata sangat umum dapat diperlakukan khusus — tetapi tidak selalu harus dihapus.

Mengapa Dihapus?

Mengurangi vocabulary/postings pada sistem klasik dan mengurangi term yang kurang diskriminatif.

Risikonya

Frasa seperti “to be or not to be”, nama resmi, atau query natural-language dapat kehilangan konteks.

Praktik Modern

Sering lebih aman menyimpan stopwords dan membiarkan ranking/statistik menentukan kontribusinya, terutama pada phrase search.

STOPWORD LIST BUKAN UNIVERSAL — bahasa dan domain menentukan.

Stemming vs Lemmatization

Keduanya mencoba menyatukan variasi bentuk kata, tetapi caranya berbeda.

Stemming

Aturan heuristik memotong afiks. Cepat dan sederhana, tetapi hasil tidak selalu berupa kata valid.

contoh konseptual: connected / connecting / connection → connect

Lemmatization

Menggunakan informasi linguistik untuk memetakan bentuk infleksi ke lemma. Lebih presisi secara bahasa, tetapi lebih kompleks.

contoh: am / is / are → be

TRADE-OFF: conflation meningkatkan recall, tetapi over-stemming dapat menurunkan precision.

02

Inverted Index

Struktur inti yang menghubungkan term dengan dokumen tempat term tersebut muncul.

Dari Term–Document Matrix ke Inverted Index

Matrix membantu memahami konsep, tetapi terlalu sparse untuk disimpan langsung.

TERM	D1	D2	D3	D4
data	1	1	0	1
web	1	0	1	1
search	0	1	1	1
index	1	0	1	0



Inverted representation

data → D1, D2, D4

web → D1, D3, D4

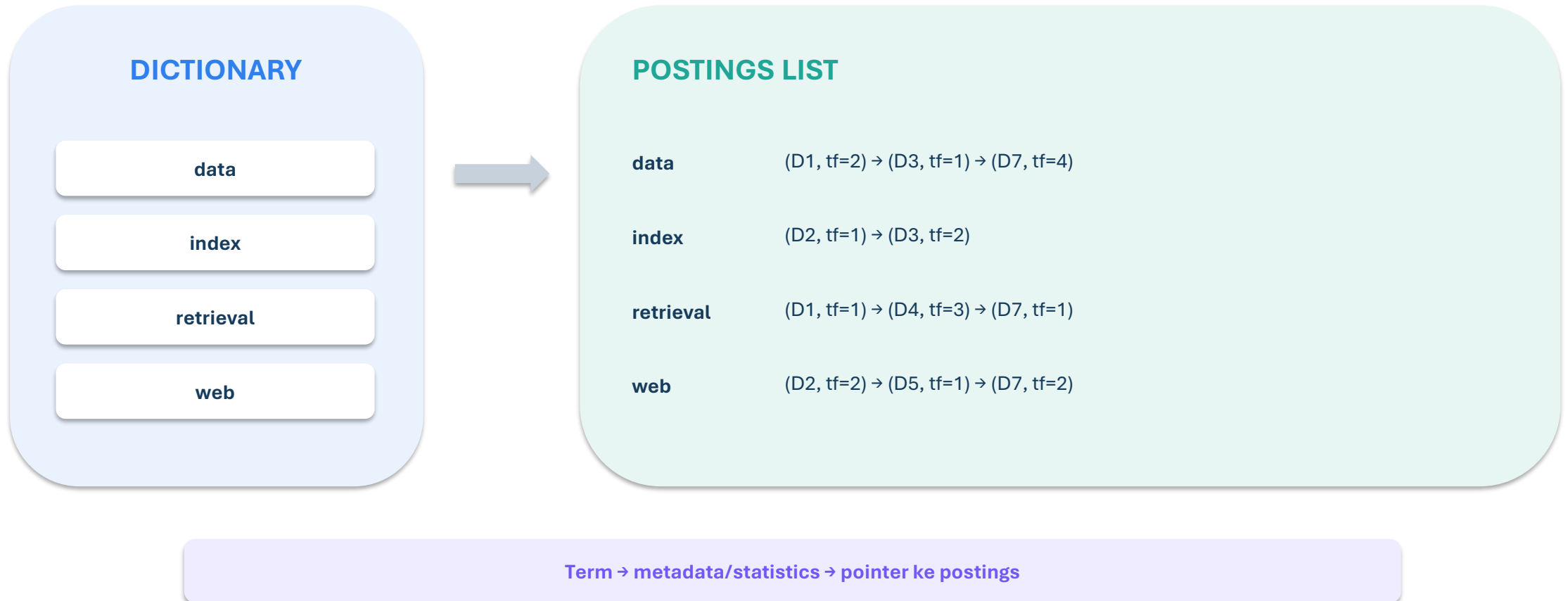
search → D2, D3, D4

index → D1, D3

Simpan hanya kemunculan — bukan semua nol.

Anatomi Inverted Index

Dictionary menunjuk ke postings list untuk setiap term.



Apa yang Disimpan di Postings?

Desain postings ditentukan oleh fitur retrieval yang ingin didukung.

DocID

Minimal: identitas dokumen yang mengandung term.

Term Frequency

tf = berapa kali term muncul pada dokumen; berguna untuk scoring.

Positions

Lokasi token dalam dokumen; diperlukan untuk phrase/proximity query.

Field / Payload

Informasi field, anchor, zone, atau payload khusus domain.

LEBIH BANYAK INFORMASI → fitur retrieval lebih kaya, tetapi index lebih besar.

TF, DF, dan Collection Frequency

Indexer juga menghitung statistik yang kelak dipakai ranking.

Term Frequency (tf)

Jumlah kemunculan term t pada dokumen d .

Contoh: $tf(data, D7)=4$

Document Frequency (df)

Jumlah dokumen yang mengandung term t .

Contoh: $df(data)=3$

Collection Frequency (cf)

Total kemunculan term t di seluruh koleksi.

Contoh: $cf(data)=7$

Mengapa disimpan? scoring seperti TF-IDF/BM25 membutuhkan statistik term dan panjang dokumen — topik ini masuk Pertemuan 5–6.

Positional Index

Postings dapat menyimpan posisi term di dalam dokumen.

D1: “web information retrieval enables web search”

Token position

web

1, 5

information

2

retrieval

3

enables

4

search

6

PHRASE QUERY dapat mengecek apakah posisi “information” diikuti “retrieval”.

Phrase & Proximity Query

Positional index memungkinkan lebih dari sekadar “dokumen mengandung term”.

Phrase Query

"information retrieval"

Cari dokumen di mana posisi retrieval = posisi information + 1.

Proximity Query

data NEAR/3 mining

Cari dokumen di mana kedua term muncul dalam jarak maksimum 3 token.

Highlight / Snippet

Posisi term dapat membantu menemukan bagian dokumen untuk snippet atau highlighting.

POSITIONAL POSTINGS lebih besar daripada postings docID-only, tetapi mendukung query yang lebih kaya.

Fielded / Zone Index

Tidak semua bagian dokumen memiliki arti yang sama.

TITLE

Term pada judul sering lebih informatif terhadap topik dokumen.

BODY

Kandungan utama; biasanya paling besar dan dominan pada postings.

ANCHOR

Anchor text dapat memberi sinyal bagaimana halaman lain mendeskripsikan dokumen.

METADATA

Kategori, author, tanggal, program studi, harga, lokasi, atau field domain lainnya.

FIELD INFORMATION memungkinkan filter, boosting, dan pencarian terstruktur.

Membangun Index untuk Koleksi Besar

Semua postings tidak selalu muat di memori dalam sekali proses.

1

Block Processing

Proses dokumen dalam blok, buat partial index, urutkan, lalu merge.

2

Memory-aware

Gunakan struktur data efisien dan flush segment ketika batas memori tercapai.

3

Merge

Gabungkan partial postings menjadi index final yang terurut menurut term/docID.

PRINSIP: external-memory indexing memindahkan sebagian pekerjaan ke disk secara terstruktur.

BSBI dan SPIMI: Dua Pola Klasik

Dua pendekatan penting untuk memahami pembangunan inverted index skala besar.

BSBI — Blocked Sort-Based Indexing

1. Kumpulkan pasangan (termID, docID) per blok
2. Sort pasangan
3. Bentuk postings per term
4. Tulis partial index
5. Merge seluruh blok

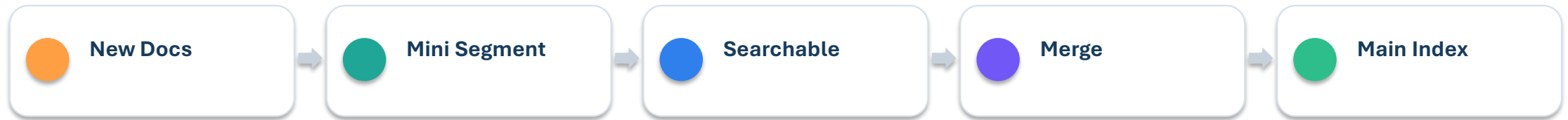
SPIMI — Single-Pass In-Memory Indexing

1. Bangun dictionary langsung di memori
2. Tambahkan docID ke postings term
3. Saat memori penuh, sort term
4. Tulis segment
5. Merge segment

SPIMI menghindari penyimpanan pasangan termID–docID eksplisit sebelum postings dibentuk.

Incremental Indexing & Segment Merge

Web berubah terus; index juga harus bisa diperbarui.



Tantangan pembaruan

- Dokumen baru harus cepat searchable.
- Dokumen lama dapat berubah atau dihapus.
- Segment terlalu banyak meningkatkan biaya query.
- Merge perlu dijadwalkan agar tidak mengganggu latency dan throughput.

INDEX = struktur dinamis, bukan artefak sekali jadi.

Compression pada Index

Index besar dapat dihemat tanpa mengorbankan kemampuan retrieval.

Dictionary Compression

Prefix sharing, blocking, atau struktur compact untuk vocabulary.

Gap Encoding

Simpan selisih docID yang terurut: 100, 107, 109 → 100, 7, 2.

Integer Coding

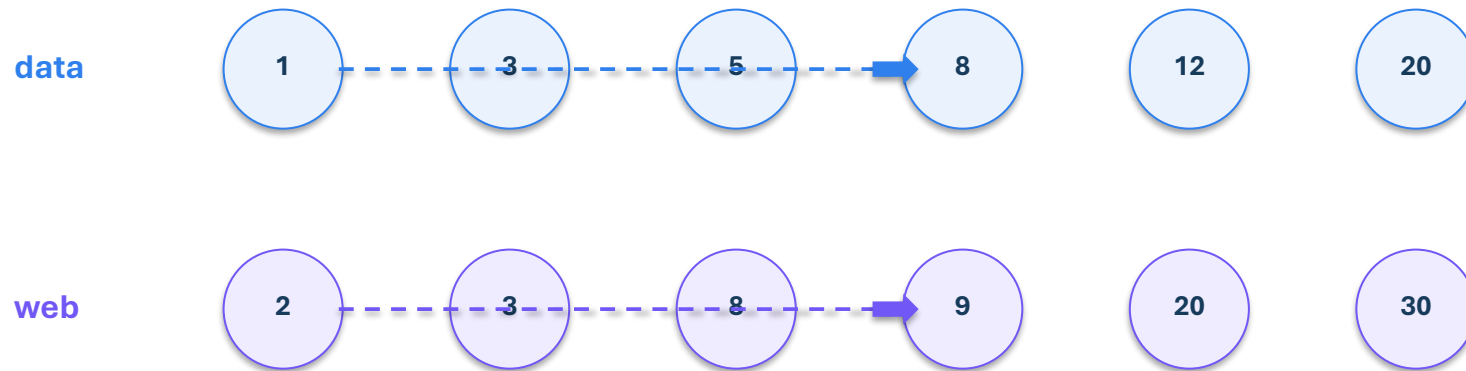
Gunakan encoding integer ringkas untuk docID gaps, tf, dan posisi.

COMPRESSION mengurangi I/O — sering kali juga mempercepat retrieval karena lebih sedikit data dibaca.

Skip Pointers & Fast Intersections

Posting lists terurut memungkinkan optimasi saat Boolean AND.

Query: data AND web



Hasil Intersection

D3, D8, D20

Skip membantu “melompati”
bagian list yang pasti tidak
cocok.

Posting list yang terurut adalah fondasi banyak optimasi query-time.

03

Studi Kasus & Praktik

Bangun index kecil secara manual agar struktur internal search benar-benar terlihat.

Kasus — Search Portal Akademik

Portal kampus harus mencari mata kuliah, dosen, dokumen, dan berita.

KOLEKSI

- Judul mata kuliah
- Deskripsi & silabus
- Nama dosen
- Dokumen PDF
- Berita kampus

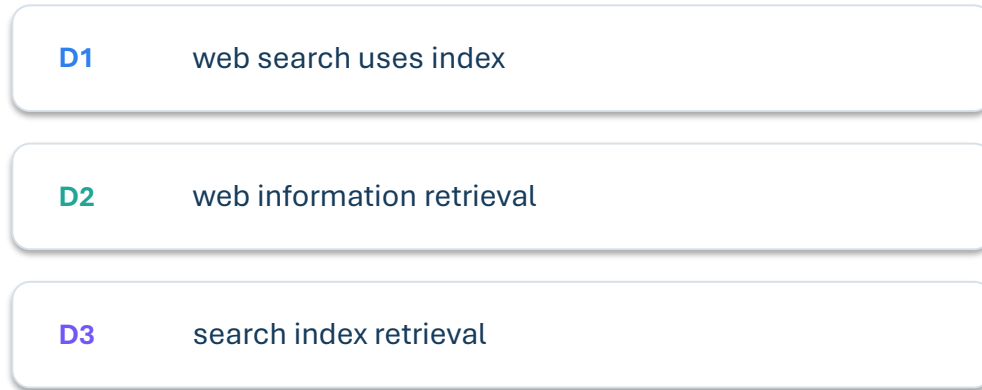
Tugas Kelompok

- Tentukan field mana yang di-index.
- Definisikan tokenization & normalization.
- Putuskan stopwords/stemming policy.
- Tentukan metadata yang disimpan pada postings.
- Jelaskan dukungan untuk phrase query dan filter.

OUTPUT: rancangan index schema + alasan desain

Simulasi: Bangun Inverted Index

Gunakan tiga dokumen pendek dan lakukan pipeline manual.



INVERTED INDEX		
index	→	D1, D3
information	→	D2
retrieval	→	D2, D3
search	→	D1, D3
uses	→	D1
web	→	D1, D2

Lakukan juga versi positional index jika waktu memungkinkan.

Menjawab Query dari Index

Retrieval Boolean adalah operasi pada postings — bukan scan dokumen.

Query 1

web AND retrieval

web $\rightarrow \{D1, D2\}$
retrieval $\rightarrow \{D2, D3\}$

intersection $\rightarrow \{D2\}$

Query 2

search OR retrieval

search $\rightarrow \{D1, D3\}$
retrieval $\rightarrow \{D2, D3\}$

union $\rightarrow \{D1, D2, D3\}$

Query 3

"search index"

Butuh posisi term.
DocID-only postings tidak cukup
untuk memastikan urutan
berdampingan.

THINK

**Apa data
minimum
yang harus
disimpan
agar Query 3
bisa
dijawab?**

Kesalahan Umum Saat Mendesain Index

Banyak masalah search muncul jauh sebelum ranking.



Tokenization salah domain

SKU / kode / URL pecah tidak tepat



Normalization terlalu agresif

entitas berbeda menjadi satu



Stopword blindly removed

phrase query kehilangan makna



No positions

phrase/proximity search tidak bisa



Field tidak dipisah

title/body/metadata kehilangan sinyal



Update tanpa strategi

segment menumpuk dan hasil basi

INDEX DESIGN HARUS DITURUNKAN DARI JENIS QUERY DAN FITUR SEARCH YANG INGIN DIDUKUNG.

Quiz Pertemuan 4

Pilih jawaban terbaik. Diskusikan alasannya.

1

Struktur utama yang memetakan term → dokumen adalah...

A. URL Frontier B. Inverted Index C. Crawler Queue D. Sitemap

2

Untuk phrase query, postings sebaiknya menyimpan...

A. posisi token B. warna font C. HTTP status D. robots policy

3

Dampak over-stemming yang paling mungkin adalah...

A. recall turun ke nol B. precision dapat turun C. index hilang D. crawler berhenti

4

df(t) berarti...

A. jumlah klik term B. jumlah dokumen yang memuat term C. panjang term D. frekuensi query

Pembahasan Jawaban

Gunakan pembahasan untuk menguatkan konsep, bukan sekadar cek benar/salah.

1

B — Inverted Index

term menjadi kunci menuju postings dokumen.

2

A — posisi token

phrase/proximity membutuhkan posisi kemunculan.

3

B — precision dapat turun

kata berbeda dapat terkonflasi menjadi stem yang sama.

4

B — document frequency

df menghitung berapa dokumen mengandung term.

REFLEKSI: fitur query apa yang tidak bisa didukung jika postings hanya berisi DocID?

Ringkasan Pertemuan 4

Empat ide yang perlu dibawa ke pertemuan berikutnya.

1. Text Processing

Tokenization, normalization, stopword, dan stemming menentukan vocabulary.

2. Inverted Index

Term → postings adalah struktur inti untuk retrieval cepat.

3. Rich Postings

tf, positions, field, dan statistik memungkinkan fitur pencarian lebih kaya.

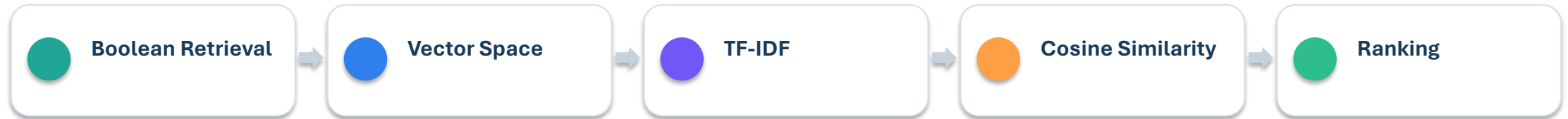
4. Scale & Updates

Segment, merge, compression, dan incremental indexing membuat index operasional.

CRAWLER MENEMUKAN DOKUMEN • INDEXER MENYUSUNNYA • RANKER MENENTUKAN URUTANNYA

Pertemuan 5 — Ranking Models

Setelah index tersedia, pertanyaan berikutnya: dokumen mana yang harus muncul paling atas?



Pertanyaan pemantik

- Mengapa dokumen yang sama-sama mengandung query tidak harus memiliki skor sama?
- Bagaimana TF dan DF mengubah pentingnya sebuah term?
- Bagaimana panjang dokumen memengaruhi kemiripan?

SIAPKAN: konsep vector, dot product, dan logaritma dasar.

Referensi Utama

Rujukan untuk pendalaman materi indexing.

- Manning, C. D., Raghavan, P., & Schütze, H. — Introduction to Information Retrieval. Cambridge University Press.
- Baeza-Yates, R. & Ribeiro-Neto, B. — Modern Information Retrieval.
- Croft, W. B., Metzler, D., & Strohman, T. — Search Engines: Information Retrieval in Practice.
- Apache Lucene documentation — analysis, tokenization, indexing, segments, postings.
- Materi RPS Web Information Retrieval — Pertemuan 4: inverted index, tokenisasi, stemming, stopword removal.

Bacaan pendalaman: pahami bagaimana analyzer memengaruhi term yang benar-benar masuk ke index.