

Fundamental PostgreSQL untuk Pemadanan Data Lintas Sumber

Teknik Normalisasi, Matching, dan Deteksi Anomali

Dikti

Dukcapil

Kemnaker

Tujuan Pelatihan

- ✓ **Normalisasi Identitas:** Memahami teknik standardisasi data menggunakan Regex.
- ✓ **Teknik Pemadanan:** Menguasai hierarki matching dari exact hingga fuzzy.
- ✓ **Similarity Scoring:** Menggunakan ekstensi pg_trgm untuk mengukur kemiripan.
- ✓ **Deteksi Anomali:** Menerapkan validasi berbasis rule dan dictionary.
- ✓ **Fallback Strategy:** Menangani data sulit dengan PL/pgSQL dan Python.



A. Regex & Normalisasi Identitas

Apa itu Regex?

Regex (Regular Expression) adalah bahasa mini untuk mengenali pola teks yang kompleks. Sangat vital untuk pembersihan data sebelum proses matching.

- ✓ Membersihkan karakter tidak valid (non-alfanumerik).
- ✓ Validasi struktur (Format email, NIK).
- ✓ Ekstraksi bagian tertentu (Nama depan, Kode pos).
- ✓ Standardisasi format (Spasi ganda menjadi tunggal).

Visual REGEXP

Edit View Select mode Insert regexp

alert	\n	newline	\0	char 0	\d [[
backspace	\r	carriage	\xyz	octal code	\u [^
synomyn for \	\t	tab	\s [[\\$ [^
< same as X & 0x1F	\u0009	tab	\x	backref	\\$ [^
ESC	\w	vert tab	\w [[\w [^
form feed	\f	hexa code	\uhhh		\u [^

\t*&]*(\w+)([\t]*<([^\n]+\n)[\t*&]*>)?([\t]*(\w+)::[\t*~]*(\w+)*[\t]*([^\n])]*?\n)?

nocase all line lineanchor linestop inline

Go Select: **match** 1 2 3 4 5 6

```
emplate<class T> List::List(const List &);  
emplate<class T> List::~List();
```

matches

Konsep Dasar Regex



Character Class

Menentukan jenis karakter yang dicari.

```
[A-Za-z], [0-9], \d, \w
```



Quantifier

Menentukan jumlah pengulangan karakter.

```
+, *, ?, {n,m}
```



Grouping

Mengelompokkan pola atau pilihan (alternation).

```
(abc), (dwi|tri)
```

Fungsi PostgreSQL: regexp_replace(), regexp_matches(), regexp_split_to_array()

Regex untuk Pembersihan Identitas

Use Case Pembersihan

- Hilangkan karakter non-alfabet (simbol aneh).
- Koreksi spasi ganda menjadi spasi tunggal.
- Hapus tanda baca (titik, koma di nama).
- Satukan huruf terpisah ("A I S Y A H").

Implementasi PostgreSQL

```
-- Menghapus non-huruf & spasi  
regexp_replace(nama, '[^A-Za-z ]', '',  
'g')  
  
-- Collapse spasi ganda  
regexp_replace(nama, '\s+', ' ', 'g')
```

Pipeline Normalisasi Nama



Trim

Hapus spasi di awal/akhir menggunakan `btrim()`.



Case

Ubah ke format seragam dengan `upper()` atau `lower()`.



Space

Hapus spasi ganda berlebih dengan regex.



Noise

Hapus gelar dan simbol (misal: "H.", "S.Kom").



Pattern

Perbaiki huruf terpisah "A B U".

Normalisasi Kode Identitas (NIK/NIP)

Permasalahan Umum

- ✓ **Leading Zero Hilang:** "102..." padahal seharusnya "0102...".
- ✓ **Karakter Asing:** Masuknya spasi, strip (-), atau titik.
- ✓ **Panjang Tidak Valid:** Digit kurang atau lebih dari standar.

Teknik Perbaikan

```
-- Ambil angka saja  
regexp_replace(nik, '[^0-9]', '', 'g')  
  
-- Padding nol di kiri (misal 16 digit)  
lpad(nik_bersih, 16, '0')  
  
-- Validasi panjang  
WHERE length(nik) = 16
```

B. Matching Data Lintas Sumber

Hierarki teknik dari yang paling ketat hingga paling longgar.



1. Exact

Ketat. Harus sama persis.



2. Similarity

Menggunakan skor kemiripan (Trigram).



3. Token

Memecah kata dan mengabaikan urutan.



4. Fuzzy

Toleransi typo & fonetik (Levenshtein).

Exact Match Strategy

Kapan Digunakan?

Metode ini paling cepat dan akurat, namun membutuhkan data yang sangat bersih.

- ✓ NIK/NIP sudah tervalidasi dan standar.
- ✓ Nama sudah melalui proses normalisasi ketat.
- ✓ Kombinasi kunci unik (Nama + Tempat Lahir + Tanggal Lahir) tersedia lengkap.

Implementasi SQL

```
-- Perbandingan langsung  
SELECT * FROM table_a a  
JOIN table_b b ON a.nik = b.nik;  
  
-- Hashing untuk privasi  
ON md5(a.nama) = md5(b.nama)
```

Similarity Match (pg_trgm)

Konsep Trigram

Ekstensi `pg_trgm` memecah teks menjadi 3 karakter berurutan untuk menghitung skor kemiripan.

- ✓ Cocok untuk menangani salah ketik (typo).
- ✓ Efektif untuk kemiripan diatas 80-90%.
- ✓ Fungsi: `similarity(a,b)`, `word_similarity(a,b)`.



Token-Based Match

Masalah Urutan Kata

Seringkali nama ditulis dengan urutan berbeda:

"MUHAMMAD RIZKI AKBAR"
"RIZKI MUHAMMAD AKBAR"

Solusi: Tokenisasi

Pecah string menjadi array kata (token), urutkan, lalu bandingkan himpunannya.

Logika Matching

```
-- 1. Split ke array  
regexp_split_to_array(nama, '\s+')  
  
-- 2. Bandingkan Array  
-- Apakah array A beririsan dengan B?  
WHERE token_a && token_b  
  
-- Apakah A mengandung semua elemen B?  
WHERE token_a @> token_b
```

Nama Panjang vs Singkatan



Remove Stopwords

Hapus kata umum yang tidak unik via regex.

BIN, BINTI, ALM



Dictionary Mapping

Ganti singkatan umum dengan bentuk lengkap.

M. → MUHAMMAD



Word Similarity

Cek apakah token pendek bagian dari token panjang.

word_similarity()

Contoh: "MUHAMMAD RIZKI" match dengan "M RIZKI"

Fuzzy Match (Levenshtein & Fonetik)

Levenshtein Distance

Menghitung jumlah "edit" (insert, delete, replace) untuk mengubah string A ke B.

Phonetic (Soundex/Metaphone)

Mencocokkan berdasarkan cara pengucapan (bunyi), bukan tulisan.

Contoh: "AQBAR" match dengan "AKBAR".

Extension: fuzzystrmatch

```
-- Jarak edit (makin kecil makin mirip)
levenshtein('GALIH', 'GALUH') -- Hasil: 1

-- Double Metaphone (Fonetik)
dmetaphone('MACHMUD') = dmetaphone('MAKHMUD')
```

C. Deteksi Anomali Data

Mengapa Penting?

Sebelum dipadankan, data harus logis. Anomali dapat menyebabkan 'false match' atau mengotori database tujuan.

- ✓ Mencegah match yang salah secara logika.
- ✓ Membersihkan data sampah.
- ✓ Memastikan integritas referensi lintas sumber.



Rule-Based Anomaly Detection

Logika Domain

Menerapkan aturan bisnis dunia nyata ke dalam query database.

- Masa studi kedokteran tidak mungkin < 6 tahun.
- Umur kelulusan sarjana tidak mungkin < 15 tahun.
- Tanggal lahir harus lebih kecil dari tanggal pendaftaran.

Implementasi SQL

```
SELECT *,  
CASE  
WHEN tgl_lulus < tgl_masuk THEN 'INVALID_DATE'  
WHEN umur < 15 THEN 'UNDERAGE'  
ELSE 'OK'  
END as data_status  
FROM mahasiswa;
```

Kategori Rule Validasi

Temporal Rules

Validasi urutan waktu, durasi, dan rentang tanggal.

Contoh: Tgl Masuk vs Tgl Keluar.

Cross-Field Rules

Kombinasi logika antar kolom berbeda.

Contoh: Gelar "dr." tapi Jurusan "Teknik".

Boundary Rules

Batasan nilai angka atau panjang string.

Contoh: IPK > 4.00 atau NIK != 16 digit.

Structural Rules

Kesesuaian format kode standar.

Contoh: Format NPWP atau Kode Prodi.

Output: Flagging & Scoring

Mekanisme Filtering

Jangan langsung hapus data. Berikan flag/tanda untuk review.

- ✓ **flag_invalid:** Boolean (TRUE/FALSE).
- ✓ **severity_score:** Skala 1-5 (5 = Critical).
- ✓ **reason_code:** Kode error untuk diagnosa (misal: ERR_DATE_LOGIC).

Contoh Output Table

ID	Status	Score	Reason
001	INVALID	5	AGE_TOO_LOW
002	VALID	0	-

Dictionary-Based Detection

Menggunakan "Kamus Kata Kunci" untuk standardisasi dan mendeteksi anomali pada free-text.



Pekerjaan

"Buruh", "Staff", "Direktur"

Deteksi variasi nama
jabatan



Pendidikan

"TI", "Informatika", "SI"

Grouping jurusan sejenis



Industri

"Retail", "Manufaktur", "Jasa"

Klasifikasi sektor usaha

Teknik Dictionary di PostgreSQL

Lookup & Mapping

Buat tabel referensi (keyword, category).

- ✓ Gunakan position() untuk cek keberadaan kata.
- ✓ Gunakan regexp_matches() untuk pola fleksibel.
- ✓ Gunakan similarity() untuk match kata kunci yang typo.

Logic Mapping

```
SELECT a.raw_text, d.kategori_baku
FROM data_mentah a
JOIN kamus_jabatan d
ON position(lower(d.keyword) in
lower(a.raw_text)) > 0
```

Real World Use Case

“

*Input: "Staff Proyek", "Mandor Lapangan",
"Tenaga Harian"*

Normalized: "PEKERJA KONTRAK"

Memungkinkan agregasi statistik ketenagakerjaan yang akurat dari ribuan variasi input manual.

D. Fallback Strategy

Ketika teknik standar gagal, kita membutuhkan solusi custom, hybrid, atau external processing.

Fallback 1 & 2: Custom SQL & Python

PL/pgSQL Custom Function

Membuat fungsi logika sendiri di dalam database.

- Kombinasi Trigram + Phonetic + Rule.
- Normalisasi token yang sangat spesifik.

```
CREATE FUNCTION match_custom() ...
```

PL/Python (In-Database)

Menjalankan kode Python langsung di dalam query SQL.

- Akses library fuzzywuzzy atau nltk.
- Algoritma kompleks yang sulit ditulis di SQL murni.

```
CREATE EXTENSION plpython3u;
```

Fallback 3: External Engine

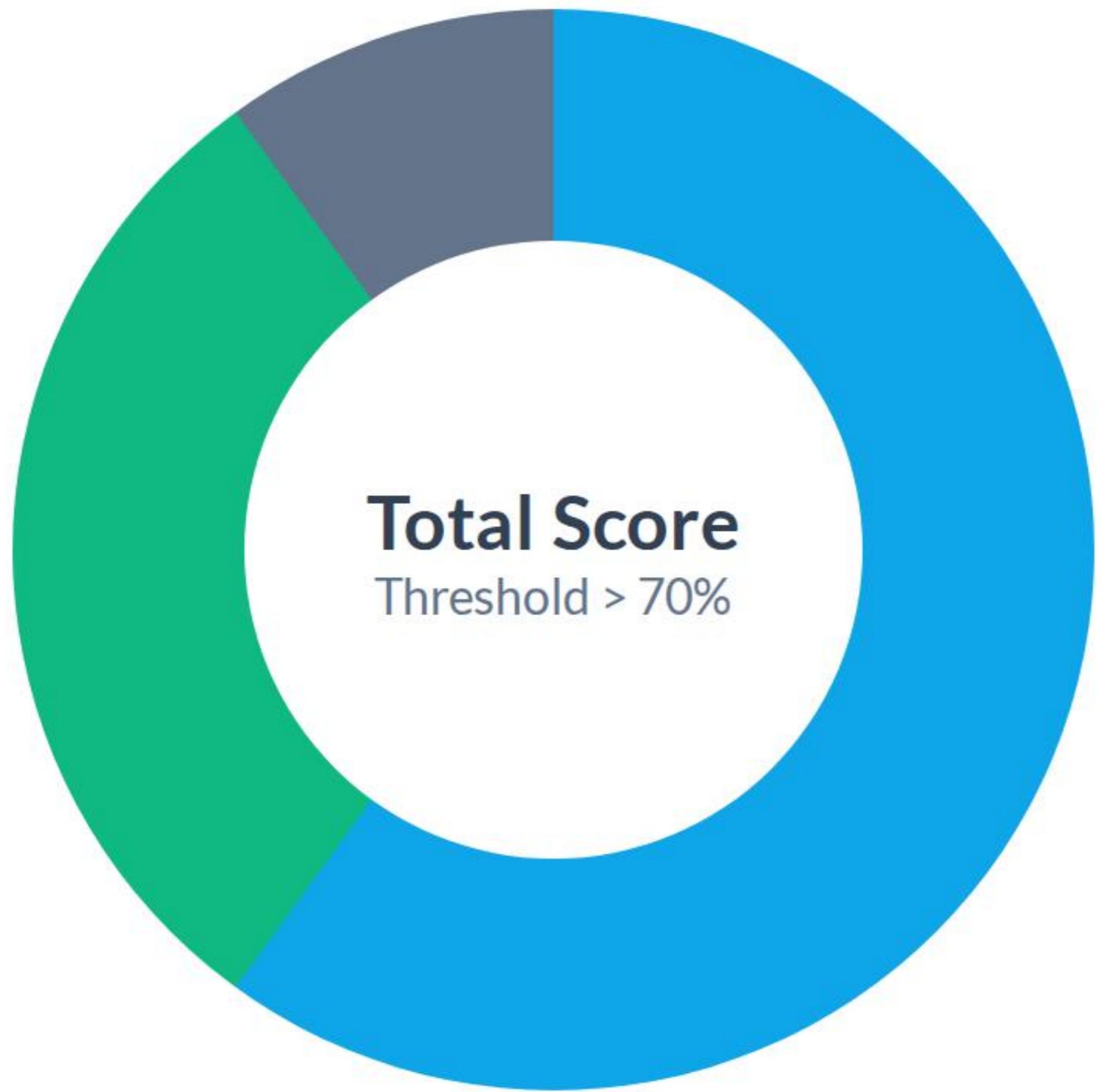
Python + Psycopg2 + Pandas

Jika PostgreSQL terlalu lambat untuk fuzzy match jutaan baris yang kompleks.

- ✓ **Pipeline:** Load Data -> Processing (Python) -> Commit Result.
- ✓ **Libraries:** rapidfuzz (Sangat cepat), pandas, jellyfish.
- ✓ **Kelebihan:** Multiprocessing (Parallel CPU) dan manajemen memori yang lebih baik untuk "Big Data".



Fallback 4: Hybrid Model Scoring



- Nama (60%)
- TTL (30%)
- Wilayah (10%)

E. Rangkuman Materi

- ✓ **Normalisasi:** Langkah krusial pertama (Regex, Trim, Upper).
- ✓ **Matching Bertingkat:** Mulai dari Exact, Token, hingga Fuzzy (Trigram/Levenshtein).
- ✓ **Validasi:** Deteksi anomali dengan Rule dan Dictionary untuk menjaga kualitas data.
- ✓ **Fallback:** Gunakan kekuatan Python (via PL/Python atau External) ketika SQL murni mencapai batasnya.

Overall Data Matching Flow



Q & A

Terima Kasih

 PostgreSQL Data Matching Training

Image Sources



<http://laurent.riesterer.free.fr/regexp/screenshot1.png>

Source: laurent.riesterer.free.fr



<https://tech5.ai/wp-content/uploads/2025/09/insite-banner-1200x630-1.jpg>

Source: tech5.ai



https://static.vecteezy.com/system/resources/previews/055/855/274/non_2x/data-cleansing-process-to-arrange-examining-or-prepare-data-to-be-analyzed-preprocessing-normalization-or-validate-missing-data-concept-businessman-cleaning-database-metaphor-of-cleansing-process-vector.jpg

Source: www.vecteezy.com



https://static.vecteezy.com/system/resources/previews/012/697/295/non_2x/3d-python-programming-language-logo-free-png.png

Source: www.vecteezy.com



<https://www.volumez.com/wp-content/uploads/2024/10/Untitled-500-x-500-px-Website.png>

Source: www.volumez.com