Engineer

Fulltekstsøk

søk i dokumenter etter ordene deres, sortert etter relevans

Hva det er

Fulltekstsøk finner dokumentene som inneholder ordene i en spørring, uansett hvilken bøyningsform ordene står i, og kan sette de beste treffene øverst. Mange sier også fritekstsøk, og på engelsk heter det full-text search. Et dokument er her det programmet behandler som én enhet å søke i: en e-post, en artikkel, en supportsak eller tittel og brødtekst fra én rad satt sammen.

Det som skiller fulltekstsøk fra et vanlig mønstersøk, er arbeidet som gjøres før noe blir sammenlignet. Først deles teksten opp i tokens: ord, tall, e-postadresser og andre klasser. Deretter gjøres hvert token om til et leksem, en normalisert form som får varianter av samme ord til å bli like. Store bokstaver blir små, endelser kuttes bort så ordet føres tilbake til stammen, og stoppord, ordene som er så vanlige at de ikke sier noe om innholdet, fjernes. Resultatet lagres sammen med posisjonen til hvert leksem, slik at rangeringen senere kan foretrekke dokumenter der søkeordene står nær hverandre. Spørringen går gjennom den samme behandlingen, så et søk på «kontrakter» og et dokument som skriver «kontrakt», møtes på midten.

PostgreSQL har dette innebygd. Datatypen `tsvector` holder et dokument i normalisert form, `tsquery` holder en spørring, og operatoren `@@` avgjør om den ene matcher den andre. En tekstsøkkonfigurasjon bestemmer hvordan normaliseringen skjer: hvilken parser som deler opp teksten, og hvilke ordbøker som gjør tokens om til leksemer. Standardinstallasjonen har konfigurasjoner for en lang rekke språk, blant dem `norwegian`, `danish`, `swedish` og `english`, og kommandoen `\dF` i psql viser hvilke som er installert. SQLite tilbyr den samme ideen gjennom utvidelsen FTS5, der en virtuell tabell holder indeksen og operatoren `MATCH` søker i den.

Bruk begrepet i en beslutning

Bruk «Hva er fulltekstsøk?» i en konkret beslutning. Vi kan knytte begrepet til systemet dere bygger eller kjøper. Finne spørringene og de manglende indeksene bak et system som har blitt tyngre for hvert år, og rette dem som betyr noe.

Se tjenestene våre Ta kontakt →

Undersøk antakelsene bak ordet

Bruk forklaringen og det tilknyttede arbeidet til å formulere spørsmål for prosjektet. Vi kan undersøke antakelsene sammen før et verktøy velges. 30 dagers strømdata på 180 ms — 180 ms for å aggregere en måned med måleravlesninger

30 dagers strømdata på 180 ms Ta kontakt →

Hvorfor det er viktig

PostgreSQL-dokumentasjonen forklarer hva de eldre operatorene mangler. `LIKE`, `ILIKE` og regulære uttrykk har ingen språklig forståelse, så et søk på «avtalene» går glipp av et dokument som skriver «avtaler», med mindre noen skriver ut alle bøyningsformene for hånd. De rangerer ikke, så tusen treff kommer tilbake i en rekkefølge som ikke sier noe om hvilke som er best. Og de kan ikke bruke en vanlig indeks til denne typen søk, så hver spørring må lese hvert eneste dokument. Et mønster som `%bil%` treffer dessuten både «bilag» og «stabil», og det er støy, ikke bedre dekning.

Folk skriver ord i et søkefelt og venter å finne dokumentene som handler om dem. I et kontraktarkiv, en kundestøtte eller en produktkatalog er fulltekstsøk forskjellen på et søkefelt folk stoler på, og et de gir opp etter tredje bom. Rangeringen betyr like mye som selve treffet: Den som søker etter «forsinket betaling», vil ha purrebrevet øverst, ikke de hundre fakturaene som nevner betaling én gang.

At søket bor i databasen man allerede drifter, har også en praktisk side. Søkeindeksen oppdateres i samme transaksjon som dataene, så en rad er søkbar i det øyeblikket den committes, og en slettet rad forsvinner fra resultatene i samme øyeblikk. Filtre på dato, kunde eller tilganger er vanlige `WHERE`-betingelser ved siden av tekstbetingelsen. En egen søketjeneste kan mer, men den er et system til som skal fôres og holdes i takt med det første, og hver glipp mellom dem viser seg som et treff som mangler, eller et som burde vært borte.

Fulltekstsøk har også en grense som er verdt å kjenne fra starten: Det matcher ord, ikke mening. Et søk på «bil» finner ikke «kjøretøy» med mindre en synonymordbok kobler dem sammen. Vektorsøk, som sammenligner tallmessige representasjoner av mening, dekker det tilfellet, og de to kombineres ofte.

Slik ser det ut i praksis

Ta en tabell med dokumenter som har en tittel, en brødtekst og en angivelse av hvor hvert dokument kom fra. En lagret generert kolonne holder den søkbare formen oppdatert ved hver innsetting og endring, og en GIN-indeks gjør den rask å søke i:

```
CREATE TABLE documents (
    id         bigint GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    title      text,
    body       text,
    source     text NOT NULL,
    search_vec tsvector GENERATED ALWAYS AS (
                   setweight(to_tsvector('norwegian', coalesce(title, '')), 'A')
                || setweight(to_tsvector('norwegian', coalesce(body, '')), 'D')
               ) STORED
);

CREATE INDEX documents_search_idx ON documents USING GIN (search_vec);
```

Tre detaljer bærer det hele. Konfigurasjonen er navngitt, `'norwegian'`, i stedet for å overlates til sesjonens `default_text_search_config`; det er formen av `to_tsvector` med to argumenter PostgreSQL kan lagre og indeksere. `coalesce` gjør en manglende tittel eller brødtekst om til en tom streng, for ellers ville én NULL gjort hele dokumentet til NULL. Og `setweight` merker hvert leksem med hvor det kom fra: A for tittelen, D for brødteksten. Rangeringsfunksjonene bruker vektene `{0.1, 0.2, 0.4, 1.0}` for D, C, B og A hvis man ikke oppgir andre, så et søkeord i tittelen teller ti ganger så mye som det samme ordet i brødteksten.

GIN er indekstypen dokumentasjonen anbefaler for tekstsøk: en invertert indeks med én oppføring per leksem og en komprimert liste over radene der leksemet forekommer. GiST fungerer også, men den er ikke eksakt og må sjekke selve tabellraden for å luke ut falske treff.

Spørringen tar imot ordene brukeren skrev, akkurat slik brukeren skrev dem:

```
SELECT id, title, source,
       ts_rank_cd(search_vec, query) AS rank
FROM   documents,
       websearch_to_tsquery('norwegian', 'ubetalt faktura -utkast') AS query
```

```
WHERE  search_vec @@ query
ORDER  BY rank DESC
LIMIT  10;
```

`websearch_to_tsquery` leser input slik et søkefelt på nettet gjør. Vanlige ord må alle forekomme, i hvilken som helst form den norske stammeregelen fører tilbake til samme leksem. Tekst i anførselstegn må forekomme som en sammenhengende frase, ordet «or» gir et alternativ, og et minus foran et ord utelukker det, så denne spørringen hopper over alle dokumenter som nevner et utkast. Funksjonen gir aldri syntaksfeil, og derfor er den det trygge valget for rå brukerinput. `ts_rank_cd` rangerer etter cover density, en metode fra en artikkel av Clarke, Cormack og Tudhope fra 1999 som også belønner søkeord som står tett sammen. Søsterfunksjonen `ts_rank` ser på hvor ofte ordene forekommer. Begge tar et normaliseringsflagg: 1 deler for eksempel rangen på én pluss logaritmen av dokumentets lengde, så et langt dokument ikke vinner bare fordi det er langt.

Norsk byr på en ekstra utfordring: sammensatte ord. Snowball-algoritmen bak konfigurasjonen `norwegian` fjerner bare endelser og deler aldri et ord, så et søk på «fabrikk» treffer ikke et dokument som bare skriver «sjokoladefabrikk». PostgreSQL kan dele sammensetninger med en Ispell- eller Hunspell-ordbok der affiksfilen merker hvilke ord som kan inngå i dem, med linjen `compoundwords controlled z`. Dokumentasjonen viser et norsk eksempel med en slik ordbok, kalt `norwegian_ispell`:

```
SELECT ts_lexize('norwegian_ispell', 'sjokoladefabrikk');
```

Svaret er `{sjokoladefabrikk,sjokolade,fabrikk}`: hele ordet og begge delene, så dokumentet blir funnet både på helheten og på hver del. PostgreSQL leveres uten slike ordbokfiler, så de må hentes og legges i katalogen `tsearch_data`. En Ispell-ordbok kjenner dessuten bare et begrenset utvalg ord, så i konfigurasjonen bør den følges av en bredere ordbok, for eksempel Snowball, som tar resten.

Selskapets arbeid med bedriftsdata klare for AI-agenter har det samme behovet. Siden beskriver hvordan gamle PDF-er, skanninger og e-postarkiver gjøres om til søkbar tekst med kilden lagret ved siden av, slik at de kan finnes og ikke bare oppbevares. Kolonnen `source` ovenfor har nettopp den jobben: Et treff er mer verdt når leseren kan åpne originalen.

SQLite kommer til samme resultat med mindre seremoni. En virtuell tabell opprettet med `CREATE VIRTUAL TABLE notes USING fts5(title, body)` spørres med `WHERE notes MATCH 'faktura NOT utkast'`, og `ORDER BY bm25(notes)` setter de beste treffene først: FTS5 gir bedre treff lavere verdier, så den vanlige stigende sorteringen er den riktige. Standardtokenizeren heter `unicode61`; tokenizeren `porter` legger til stemming, men ifølge dokumentasjonen bare for engelsk, så norske bøyningsformer slås ikke sammen i FTS5 uten en tokenizer man legger til selv.

Hvor det går galt

En indeks spørringen aldri rører. En uttrykksindeks bygd på `to_tsvector('norwegian', body)` hjelper bare spørringer som gjentar akkurat det uttrykket. En spørring skrevet som `to_tsvector(body)`, som lener seg på standardkonfigurasjonen, leser hele tabellen. Lagre vektoren i en generert kolonne som ovenfor, og søk i kolonnen; da kan spørring og indeks ikke gli fra hverandre.

Én NULL som skjuler et dokument. `to_tsvector(NULL)` gir NULL, og en NULL-tittel satt sammen med en brødtekst gir også NULL, så en rad uten tittel forsvinner i stillhet fra alle søk. Pakk hvert felt inn i `coalesce`.

Rå input i `to_tsquery`. `to_tsquery` forventer sin egen operatorsyntaks og tåler dårligere rotete input enn de andre konverteringsfunksjonene, så en bruker som taster en parentes uten partner eller et løst &-tegn, får en feilside. Send brukerinput gjennom `websearch_to_tsquery`, eller gjennom `plainto_tsquery` når det ikke skal være operatorer i det hele tatt.

Feil språk. Norsk tekst indeksert med den engelske konfigurasjonen blir kuttet etter engelske regler og filtrert med engelske stoppord, så halvparten av bøyningsformene aldri møtes. Velg konfigurasjonen etter språket i teksten. En samling på flere språk kan lagre navnet på konfigurasjonen i en egen kolonne og indeksere `to_tsvector(config_name, body)`, så lenge spørringene bruker den samme kolonnen.

En ordbok endret bak ryggen på sesjonene. En databasesesjon leser ordbokfilene sine første gang de brukes, og så ikke igjen. Den som legger nye ord i en Ispell-ordbok, ser dem derfor bare i nye tilkoblinger, og en tilkoblingspool kan holde på de gamle lenge. Kjør `ALTER TEXT SEARCH DICTIONARY` på ordboken, også som en tom endring der ingen parameter får ny verdi, så leser de

eksisterende sesjonene filene på nytt. En tesaurusordbok brukes når dokumentene indekseres, så etter en endring i den må de lagrede vektorene bygges på nytt.

Rangering av hvert eneste treff. Rangeringen leser `tsvector` for hver rad som matcher, og dokumentasjonen advarer om at det kan være I/O-bundet og tregt. En spørring på ett ord som matcher halve tabellen, får databasen til å rangere halve tabellen før `LIMIT` beholder ti. Snevre inn kandidatene med vanlige betingelser først, for eksempel et datointervall eller en kunde, og hold vektoren lagret så den ikke beregnes på nytt for hver rad.

En rang vist som prosent. Rangeringsfunksjonene bruker ingen kunnskap om samlingen som helhet, så en rang kan ikke gjøres om til et rettferdig «87 % relevant». Normaliseringsflagget 32 presser rangen inn mellom null og én, men dokumentasjonen kaller det en rent kosmetisk endring. Bruk rangen til å sortere resultatene og til ingenting annet.

Uthevinger med markup. `ts_headline` returnerer et utdrag der søkeordene er merket, og det er akkurat det en resultatside vil ha. PostgreSQL advarer om at utdraget ikke er garantert trygt å sette rett inn i en nettside. Et dokument med HTML kan bære et skript med seg inn på siden. Fjern markup fra input, eller send output gjennom en HTML-sanitizer. Funksjonen arbeider dessuten på originalteksten og ikke på vektoren, så kall den bare for radene på siden som vises, etter `LIMIT`, og ikke for hvert treff.

Dokumenter som er for store for én vektor. En `tsvector` må holde seg under 1 MB og lagrer høyst 256 posisjoner per leksem. En hel bok eller et års referater i én rad støter mot begge grensene. Del lange dokumenter opp i avsnitt eller sider, så hvert resultat også får et mer presist sted å peke til.

Bruddstykker og varenumre. Fulltekstsøk sammenligner hele leksemer, så «fakt» gir ingen treff på fakturaer, og et varenummer som `AB-1042-X` kan bli delt i biter av parseren. Behold en vanlig indeks på kolonnene folk slår opp på eksakt verdi, eller bruk tokenizeren `trigram` i SQLite, som gjør hver sammenhengende rekke på tre tegn til et token og dermed støtter søk på delstrenger.

«Får» som blir til «far» i SQLite. Tokenizeren `unicode61` fjerner som standard diakritiske tegn fra latinske bokstaver, og «å» regnes med, så FTS5 lagrer «får» og «far» som samme token og blander dem i treffene. «Æ» og «ø» får stå i fred.

Opprett tabellen med `tokenize = 'unicode61 remove_diacritics 0'` når «å» skal være sin egen bokstav.

En SQLite-indeks som glir fra tabellen sin. En FTS5-tabell opprettet med `content=` henter teksten fra en annen tabell, og dokumentasjonen overlater til brukeren å holde de to i samsvar. Kommer de ut av takt, gir spørringene resultater som ser ut til å motsi hverandre. Opprett triggerne for insert, update og delete som dokumentasjonen viser, i samme migrering som den virtuelle tabellen.

Les mer

- Introduction to full text search — PostgreSQLs definisjon av dokumenter, tokens, leksemer og konfigurasjoner, og hva `LIKE` og regulære uttrykk mangler.
- Tables and Indexes — søk i en tabell, uttrykksindeks mot en lagret generert kolonne, og hvorfor konfigurasjonen må navngis.
- Controlling Text Search — vekter, `websearch_to_tsquery`, `ts_rank` og `ts_rank_cd`, normaliseringsflagg og advarselen om `ts_headline`.
- Dictionaries — Snowball-, Ispell- og Hunspell-ordbøker, deling av sammensatte ord med det norske eksemplet, og når sesjoner leser ordbokfilene.
- Norwegian stemming algorithm — Snowball-prosjektets beskrivelse av den norske stammeregelen, med endelsene den fjerner og eksempler på ord og stammer.
- SQLite FTS5 Extension — virtuelle tabeller, operatoren `MATCH`, de innebygde tokenizerne med `remove_diacritics`, `bm25()` og tabeller med eksternt innhold.

Ordliste

Velg ut fra deres faktiske rammer

Ta med et representativt datasett, spørsmålene det skal besvare og dagens flaskehals. Vi kan avtale målinger av kvalitet, gjennomstrømning og gjenoppretting.

Se hva det koster Ta kontakt →

Gjør begrepene praktiske

Spør hvordan «Hva er fulltekstsøk?» gjelder i organisasjonen deres. Et konkret eksempel fra dagens system er nok til å starte samtalen.

Se tjenestene våre

welcome@engineer.company +45 50 10 61 84 København, Danmark

Ta med deg denne siden

Visittkort

Laget med kjærlighet av D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Opphavsrett © 2025 – nå · Engineer ApS · Alle rettigheter forbeholdt.

Skrevet ut fra https://engineer.company/nb/glossary/full-text-search/