Nepali NLP Group

Nepali Support for Full Text Search in PostgreSQL

- Ingroj Shrestha

Team Lead, NLP R&D Engineer and Tech Blogger
Nepali NLP Group

Nepali NLP Group

Agenda

- Full text search in PostgreSQL database
  - FTS types and operators
  - Parser, Dictionaries, Configurations (Testing and Debugging)
  - FTS + Timestamp
  - Controlling Text search – Highlighting, Ranking
  - JSON[b] data in PostgreSQL
  - FTS index (RUM), Prefix Search, Phrase Search, Faceted Search
  - Additional Features- Query rewriting, Document Statistics
- Nepali Support for FTS in PostgreSQL
  - Devanagari Script Support
  - **Nepali Stop word dictionary**
  - **Nepali Snowball dictionary**
  - Nepali Hunspell dictionary

What is Full Text Search(FTS) ?

- Find documents, which match a query
  - If *tsvector (document)* matches a *tsquery (query)*
- Sort them by relevance

Why FTS?

- Provides linguistic support
- Processes document only once while insertion, so there is no search overhead
- Provides index support, which allows faster searching and ranking

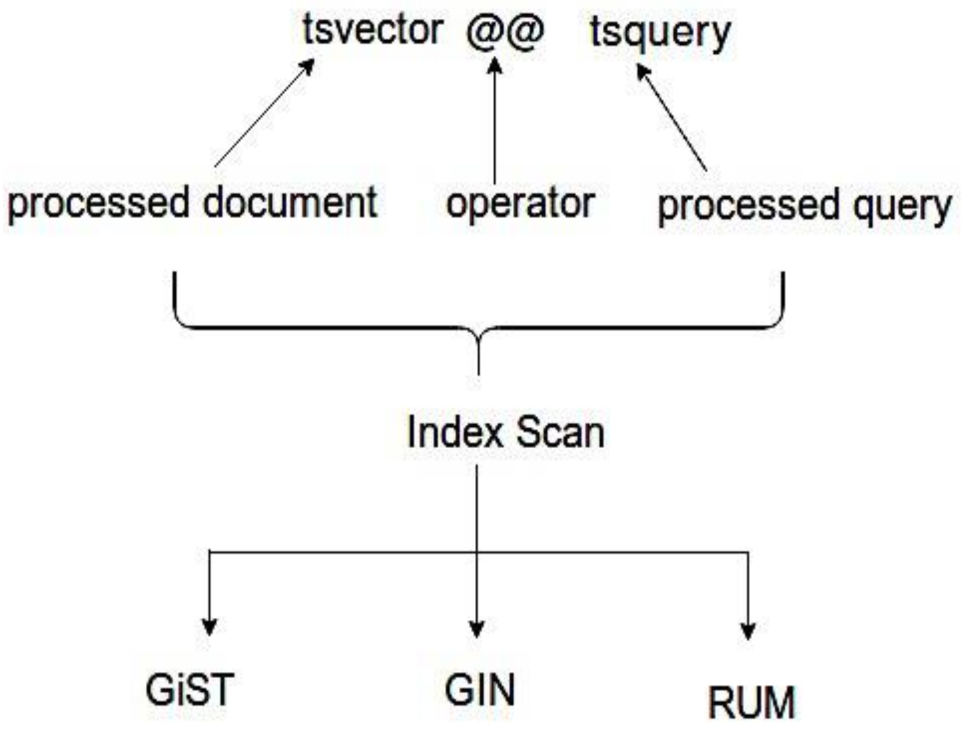
Why FTS in Database?

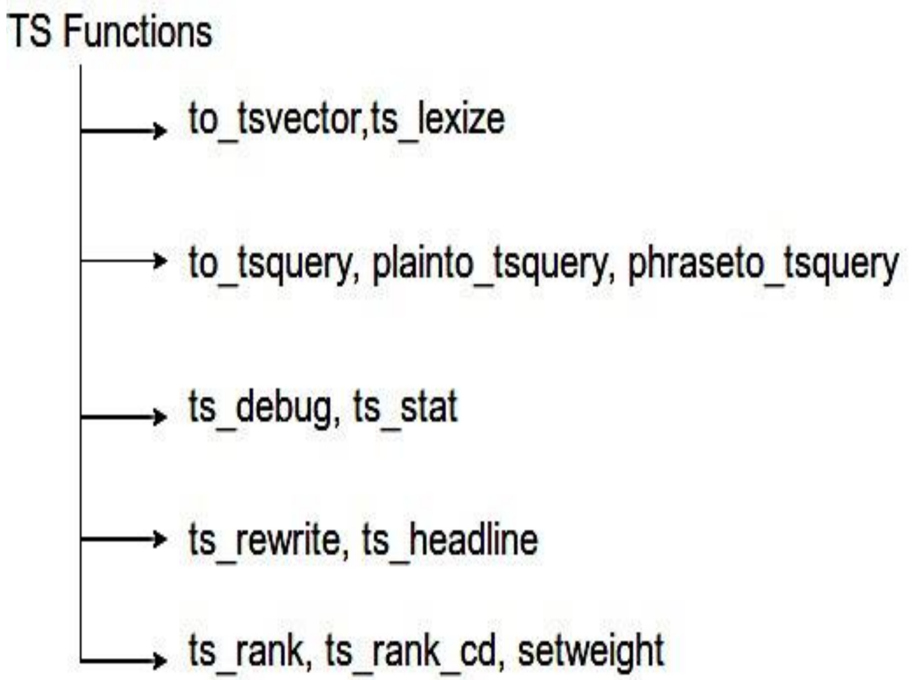
- Many external search engines exist but:
    - Can’t index all documents
    - Don’t provide access to attributes
    - Maintenance overhead
    - Additional overhead to certify
    - Don’t have instant search
    - Don’t provide consistency

FTS in Database

- FTS requires:
  - Full integration with database engine, so it should support:
    - Transactions, concurrent access, recovery, online index
  - Configurability
  - Scalability

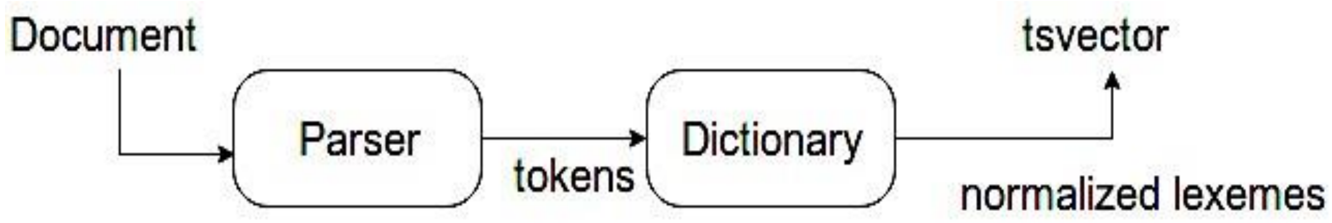
Full Text Search

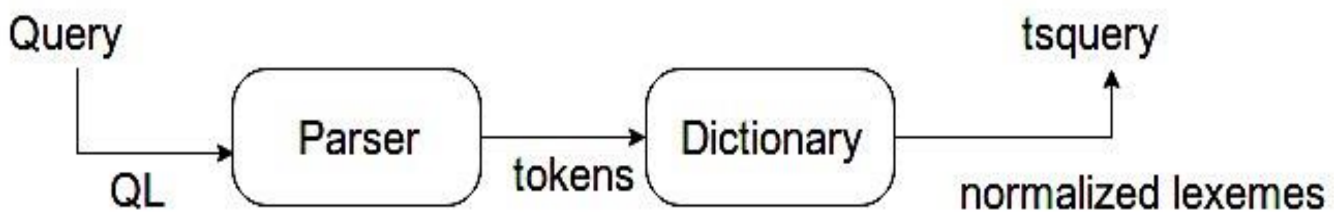



TS Functions
to_tsvector,ts_lexize
to_tsquery, plainto_tsquery, phraseto_tsquery
ts_debug, ts_stat
ts_rewrite, ts_headline
ts_rank, ts_rank_cd, setweight

Nepali NLP Group

Full Text Search





PGConf Nepal 2018
May 04- 05, 2018

Nepali NLP Group

Full Text Search

to_tsvector

Syntax: to_tsvector (*ftscfg* regconfig, *document* text) returns tsvector

Stop-word

```
postgres=# select to_tsvector('जिल्ला अध्यक्षमा दुई व्यक्ति उठ्दा एकलाई भोट त हाल्नुपर्ला नि ');

                    to_tsvector
---------------------------------------------------------------
 'अध्यक्ष':2 'उठ्':5 'एक':6 'जिल्ला':1 'भोट':7 'व्यक्ति':4 'हाल्नुपर्ला':9
(1 row)
```

Position

tsvector with labels

Syntax: setweight(*vector* tsvector, *weight* “char”) returns tsvector

```
postgres=# select setweight(to_tsvector('जिल्ला अध्यक्षमा दुई व्यक्ति '),'A')
                                 || to_tsvector('उठ्दा एकलाई भोट त हाल्नुपर्ला नि ');

                    ?column?
-----------------------------------------------------------------
 'अध्यक्ष':2A 'उठ्':5 'एक':6 'जिल्ला':1A 'भोट':7 'व्यक्ति':4A 'हाल्नुपर्ला':9
(1 row)
```

PGConf Nepal 2018
May 04- 05, 2018

Nepali NLP Group

Full Text Search

to_tsquery

Syntax: to_tsquery (*ftscfg* regconfig, *document* text) returns tsquery

```
postgres=#  select to_tsquery('जिल्ला  & अध्यक्षमा  & दुई & (एकलाई | भोट)');

           to_tsquery
----------------------------------------
 'जिल्ला' & 'अध्यक्ष' & ( 'एक' | 'भोट' )
(1 row)
```

tsquery with labels

```
postgres=# select to_tsquery('जिल्ला  & अध्यक्षमा:AB');

     to_tsquery
------------------------
 'जिल्ला' & 'अध्यक्ष':AB
(1 row)
```

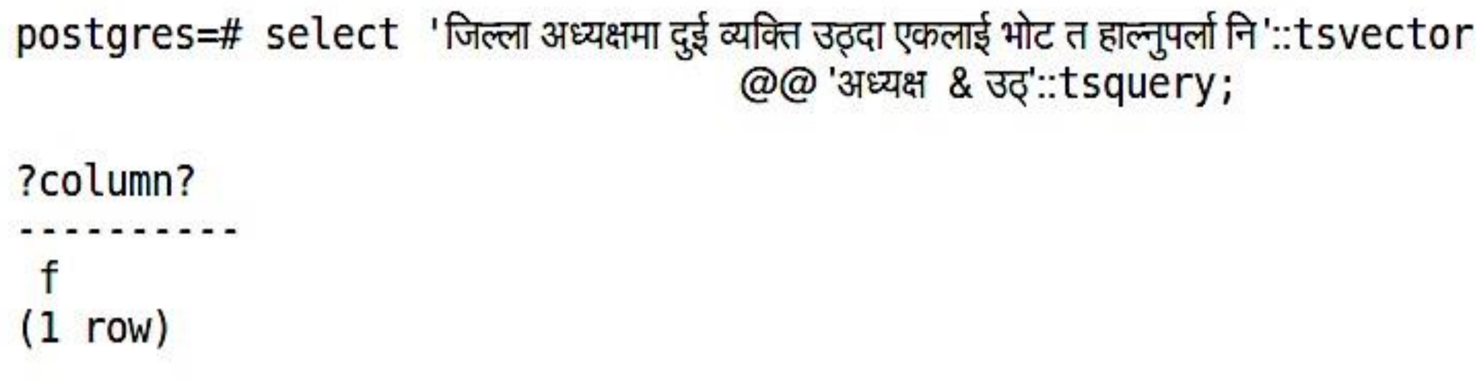
tsquery for prefix search

```
postgres=#  select to_tsquery('जिल्ला  & अध्य:*');

    to_tsquery
-------------------
 'जिल्ला' & 'अध्य':*
(1 row)
```

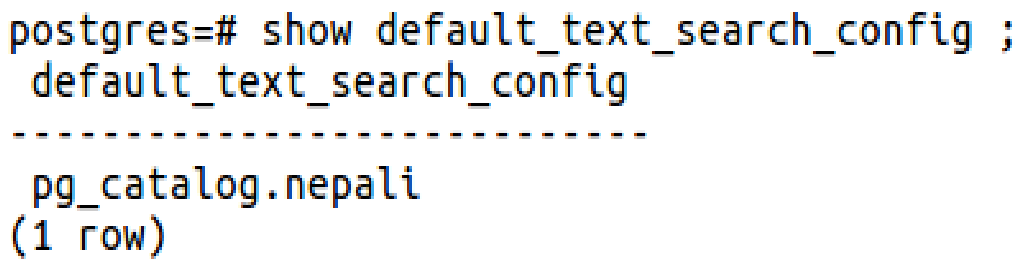
Nepali NLP Group

FTS Basic Text Matching

```
postgres=# select to_tsvector('जिल्ला अध्यक्षमा दुई व्यक्ति उठ्दा एकलाई भोट त हाल्नुपर्ला नि ')
                     @@ to_tsquery('अध्यक्ष  & उठ्');

 ?column?
----------
 t
(1 row)
```



```
postgres=# select  'जिल्ला अध्यक्षमा दुई व्यक्ति उठ्दा एकलाई भोट त हाल्नुपर्ला नि '::tsvector
                                     @@ 'अध्यक्ष  & उठ्'::tsquery;

?column?
----------
 f
(1 row)
```



```
postgres=# show default_text_search_config ;
 default_text_search_config
----------------------------
 pg_catalog.nepali
(1 row)
```

Nepali NLP Group

Why tsquery ?

- Write complex queries, for example:
  अाकाशगंगा or सूक्ष्म तारामण्डल
- Difficult to express in SQL
- tsquery (text search query) provides compact way
  '(अाकाशगंगा | (सूक्ष्म & तारामण्डल)) | (सितारा)'
- plainto_tsquery() for AND-ed query
  'सूक्ष्म तारामण्डल'

Nepali NLP Group

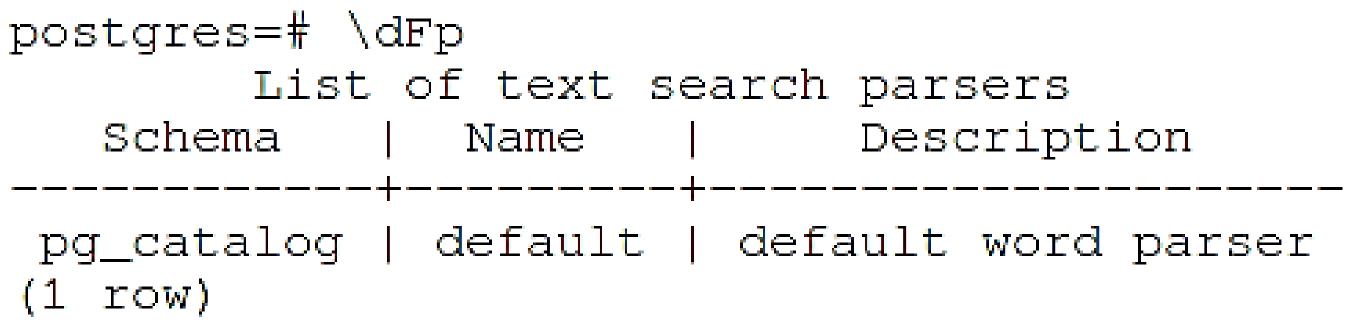
FTS Parser

- Splits document into tokens
- Determines type of each token

SQL commands

{CREATE | ALTER | DROP} TEXT SEARCH {CONFIGURATION | DICTIONARY | **PARSER**}

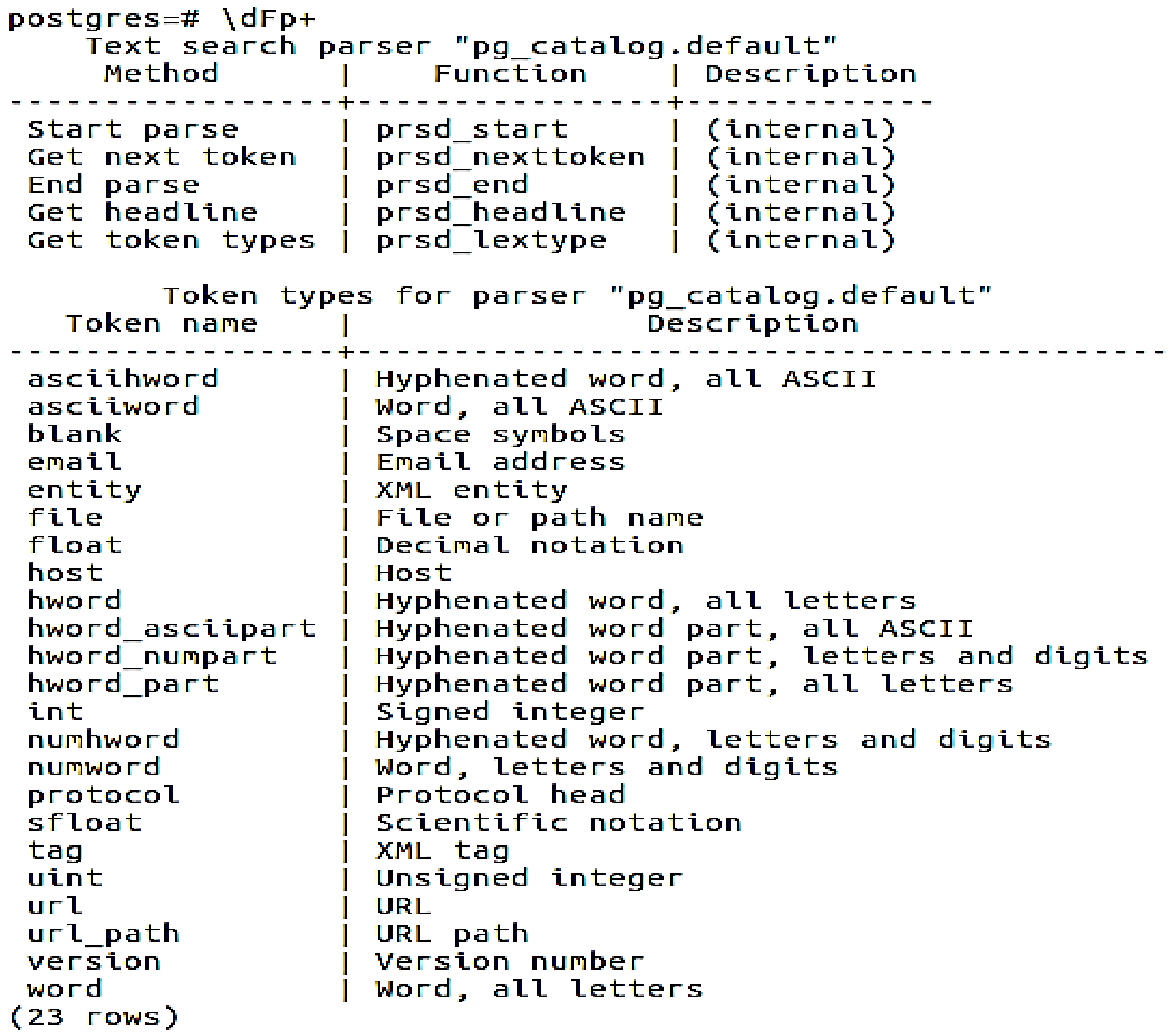
PSQL commands
\dF{,d,**p**,t}[+][PATTERN]



```
postgres=# \dFp
         List of text search parsers
   Schema    |  Name    |      Description
-------------+----------+---------------------
 pg_catalog | default | default word parser
(1 row)
```

Nepali NLP Group

FTS Parser

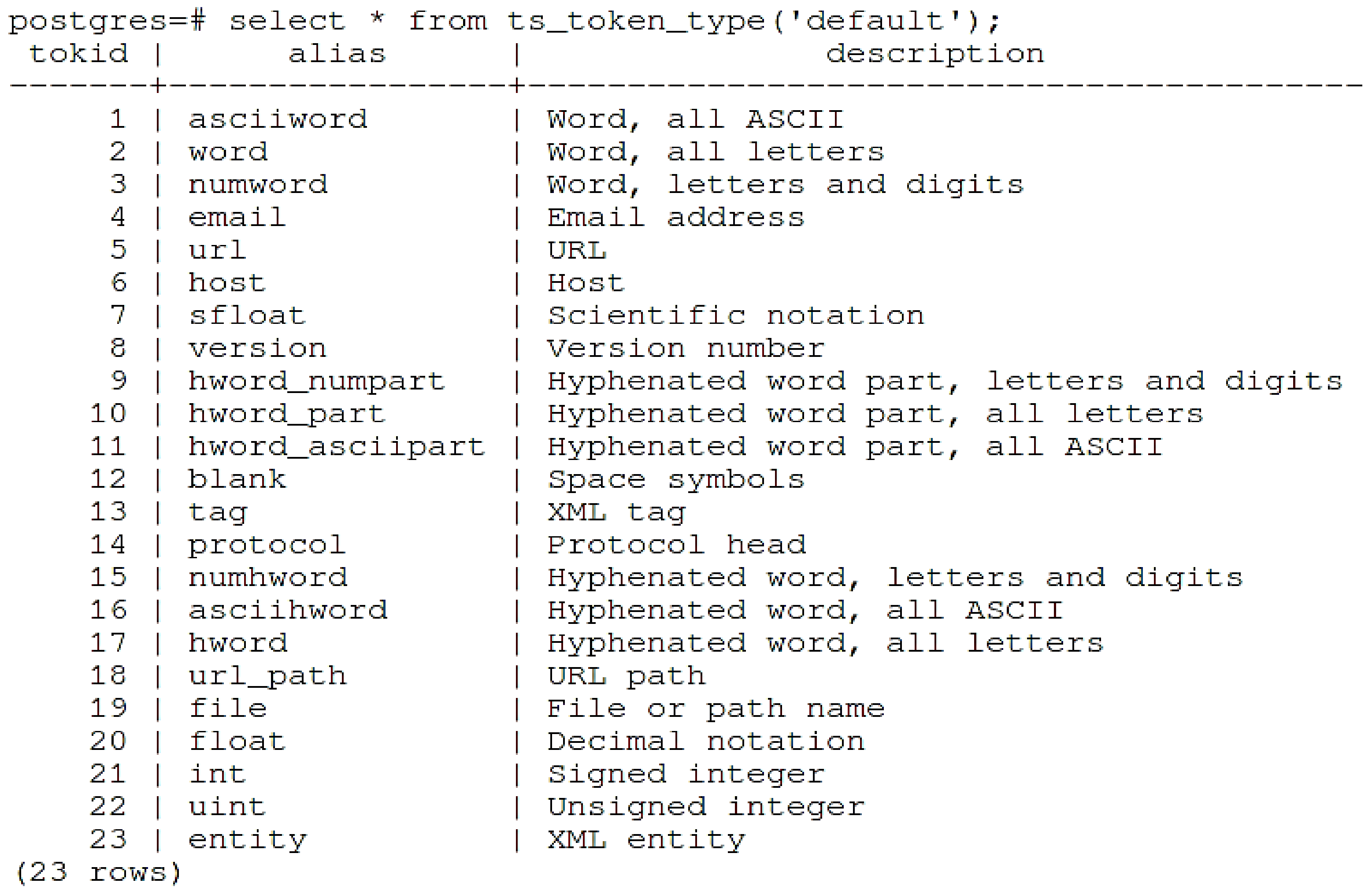


```
postgres=# \dFp+
    Text search parser "pg_catalog.default"
     Method      |    Function    | Description
-----------------+----------------+-------------
 Start parse     | prsd_start     | (internal)
 Get next token  | prsd_nexttoken | (internal)
 End parse       | prsd_end       | (internal)
 Get headline    | prsd_headline  | (internal)
 Get token types | prsd_lextype   | (internal)

        Token types for parser "pg_catalog.default"
   Token name    |               Description
-----------------+------------------------------------------
 asciihword      | Hyphenated word, all ASCII
 asciiword       | Word, all ASCII
 blank           | Space symbols
 email           | Email address
 entity          | XML entity
 file            | File or path name
 float           | Decimal notation
 host            | Host
 hword           | Hyphenated word, all letters
 hword_asciipart | Hyphenated word part, all ASCII
 hword_numpart   | Hyphenated word part, letters and digits
 hword_part      | Hyphenated word part, all letters
 int             | Signed integer
 numhword        | Hyphenated word, letters and digits
 numword         | Word, letters and digits
 protocol        | Protocol head
 sfloat          | Scientific notation
 tag             | XML tag
 uint            | Unsigned integer
 url             | URL
 url_path        | URL path
 version         | Version number
 word            | Word, all letters
(23 rows)
```

FTS Parser

ts_token_type

Syntax: ts_token_type(*parser_oid* oid) return setofrecord



```
postgres=# select * from ts_token_type('default');
 tokid |      alias      |               description
-------+-----------------+------------------------------------------
     1 | asciiword       | Word, all ASCII
     2 | word            | Word, all letters
     3 | numword         | Word, letters and digits
     4 | email           | Email address
     5 | url             | URL
     6 | host            | Host
     7 | sfloat          | Scientific notation
     8 | version         | Version number
     9 | hword_numpart   | Hyphenated word part, letters and digits
    10 | hword_part      | Hyphenated word part, all letters
    11 | hword_asciipart | Hyphenated word part, all ASCII
    12 | blank           | Space symbols
    13 | tag             | XML tag
    14 | protocol        | Protocol head
    15 | numhword        | Hyphenated word, letters and digits
    16 | asciihword      | Hyphenated word, all ASCII
    17 | hword           | Hyphenated word, all letters
    18 | url_path        | URL path
    19 | file            | File or path name
    20 | float           | Decimal notation
    21 | int             | Signed integer
    22 | uint            | Unsigned integer
    23 | entity          | XML entity
(23 rows)
```

Parser Testing

ts_parse

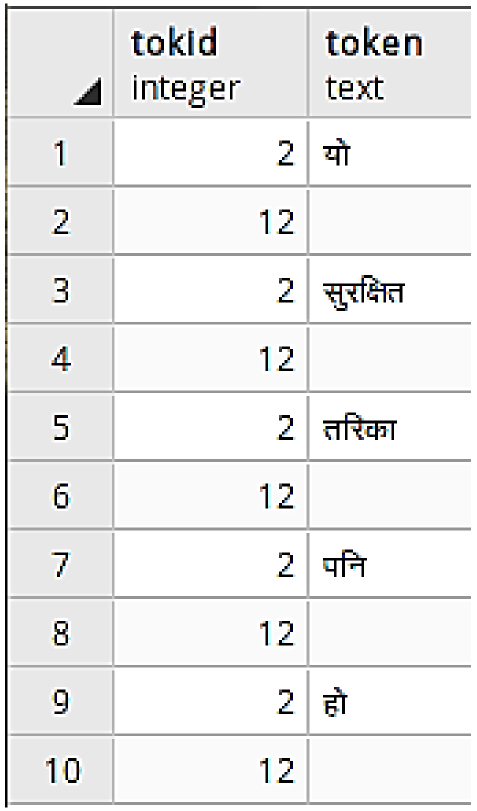
- Parses the given document and return series of records, *one for each token produced by parsing*

```
ts_parse(parser_name text, document text,
         OUT tokid integer, OUT token text) returns setof record
ts_parse(parser_oid oid, document text,
         OUT tokid integer, OUT token text) returns setof record
```

Nepali NLP Group

Parser Testing

```
postgres=# SELECT * FROM ts_parse('default','यो सुरक्षित तरिका पनि हो ');
```



	tokid integer	token text
1	2	यो
2	12	
3	2	सुरक्षित
4	12	
5	2	तरिका
6	12	
7	2	पनि
8	12	
9	2	हो
10	12	

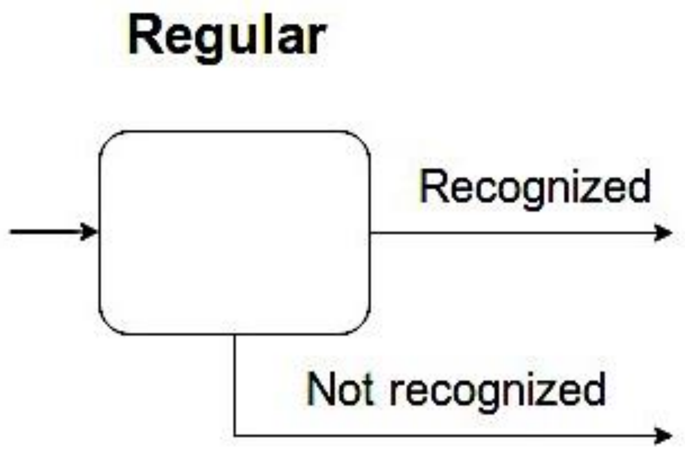
FTS Dictionary

A *program* that accepts a token and returns:

- *An array of lexemes*, if the token is recognized
- *Empty*, if token is stop word
- *NULL*, if not recognized

Regular dictionaries

- Return token, if it's recognized
- Else, pass it to subsequent dictionaries
- Example: *ispell, simple, synonym, snowball*

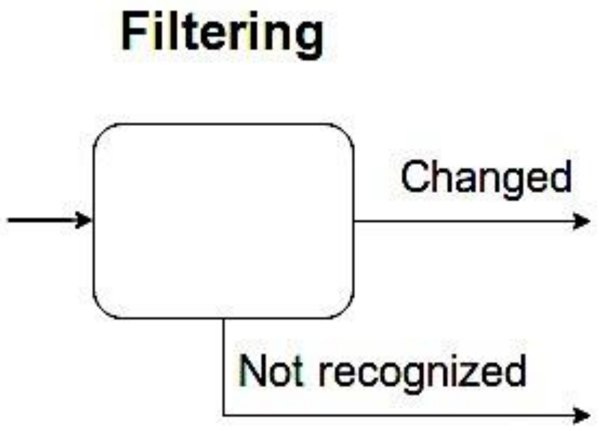


Nepali NLP Group

FTS Dictionary

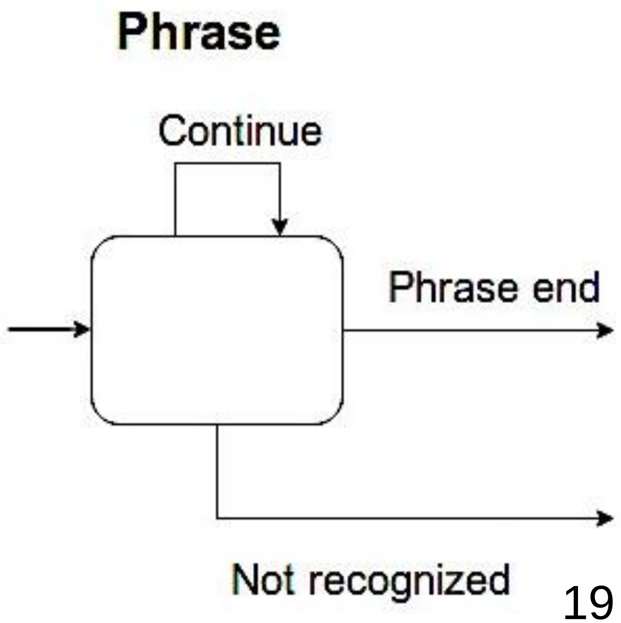
Filtering dictionaries

- Replace the recognized token
- Then, pass it to subsequent dictionaries
- Can be placed anywhere, except the end
- Example: *unaccent*



Phrase dictionaries

- Similar to Regular dictionaries in terms of functionality
- Recognize phrases (more than on e token)
- Hold control until the end of phrase processing
- Example: *thesaurus*



Nepali NLP Group

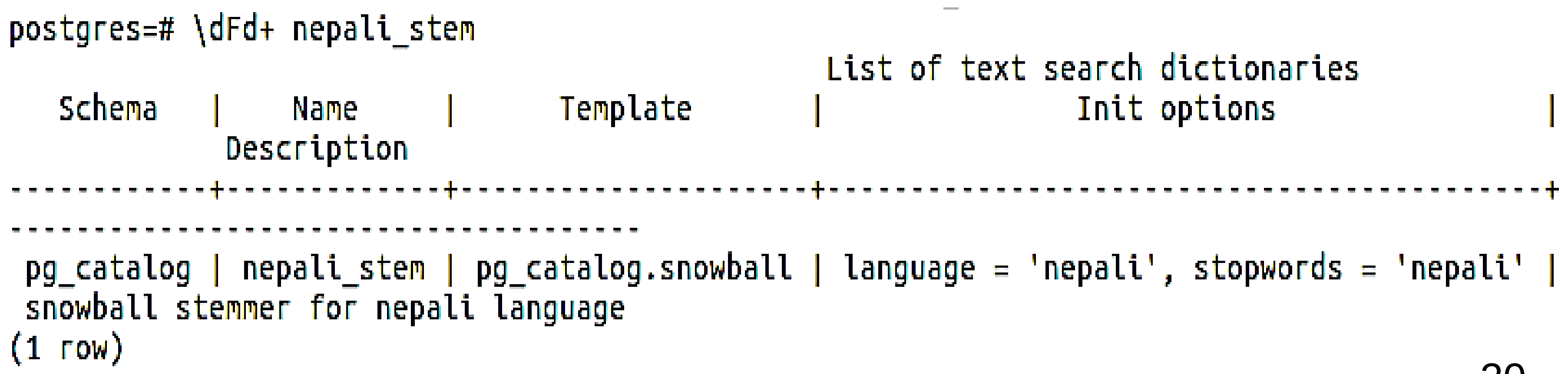
FTS Dictionary

SQL command

{CREATE | ALTER | DROP} TEXT SEARCH {CONFIGURATION | **DICTIONARY** | PARSER}

PSQL command

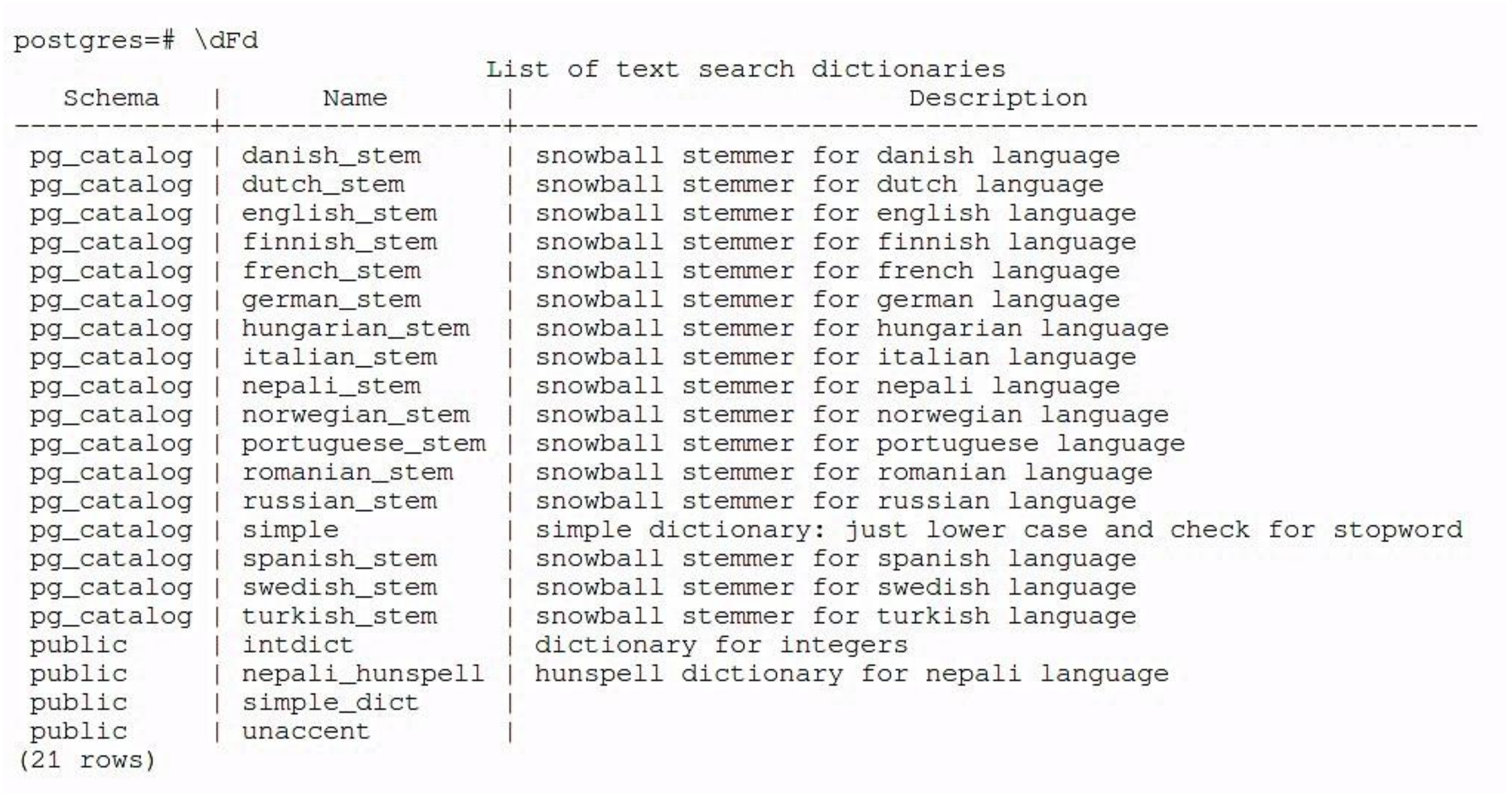
\dF{,**d**,p,t}[+][PATTERN]



```
postgres=# \dFd+ nepali_stem
                                                List of text search dictionaries
   Schema   |    Name     |      Template      |                Init options                |
           Description
------------+-------------+--------------------+--------------------------------------------+
--------------------------------------
 pg_catalog | nepali_stem | pg_catalog.snowball | language = 'nepali', stopwords = 'nepali' |
 snowball stemmer for nepali language
(1 row)
```

Nepali NLP Group

FTS Dictionary

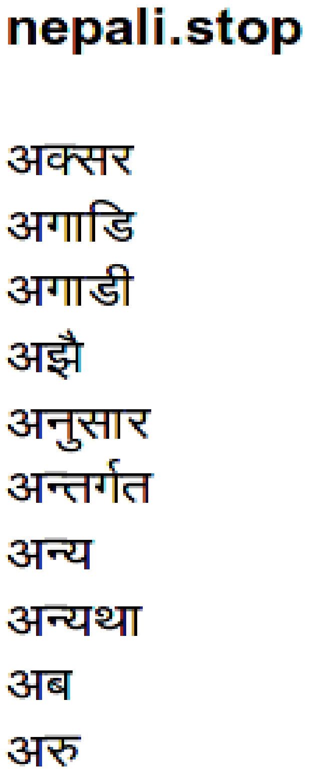


```
postgres=# \dFd
                        List of text search dictionaries
   Schema   |       Name        |                        Description
------------+-------------------+-----------------------------------------------------------
 pg_catalog | danish_stem       | snowball stemmer for danish language
 pg_catalog | dutch_stem        | snowball stemmer for dutch language
 pg_catalog | english_stem      | snowball stemmer for english language
 pg_catalog | finnish_stem      | snowball stemmer for finnish language
 pg_catalog | french_stem       | snowball stemmer for french language
 pg_catalog | german_stem       | snowball stemmer for german language
 pg_catalog | hungarian_stem    | snowball stemmer for hungarian language
 pg_catalog | italian_stem      | snowball stemmer for italian language
 pg_catalog | nepali_stem       | snowball stemmer for nepali language
 pg_catalog | norwegian_stem    | snowball stemmer for norwegian language
 pg_catalog | portuguese_stem   | snowball stemmer for portuguese language
 pg_catalog | romanian_stem     | snowball stemmer for romanian language
 pg_catalog | russian_stem      | snowball stemmer for russian language
 pg_catalog | simple            | simple dictionary: just lower case and check for stopword
 pg_catalog | spanish_stem      | snowball stemmer for spanish language
 pg_catalog | swedish_stem      | snowball stemmer for swedish language
 pg_catalog | turkish_stem      | snowball stemmer for turkish language
 public     | intdict           | dictionary for integers
 public     | nepali_hunspell   | hunspell dictionary for nepali language
 public     | simple_dict       |
 public     | unaccent          |
(21 rows)
```

Nepali NLP Group

Nepali Stop Word

- $SHAREDIR/tsearch_data/**nepali.stop**
- Contains 304 words
- Frequent words in a corpus, that:
  - Can include language-specific:
    - Determiners
    - Conjunctions
    - Postpositions
  - Also, it can include common words like:
    - Names
    - Temporal words



nepali.stop

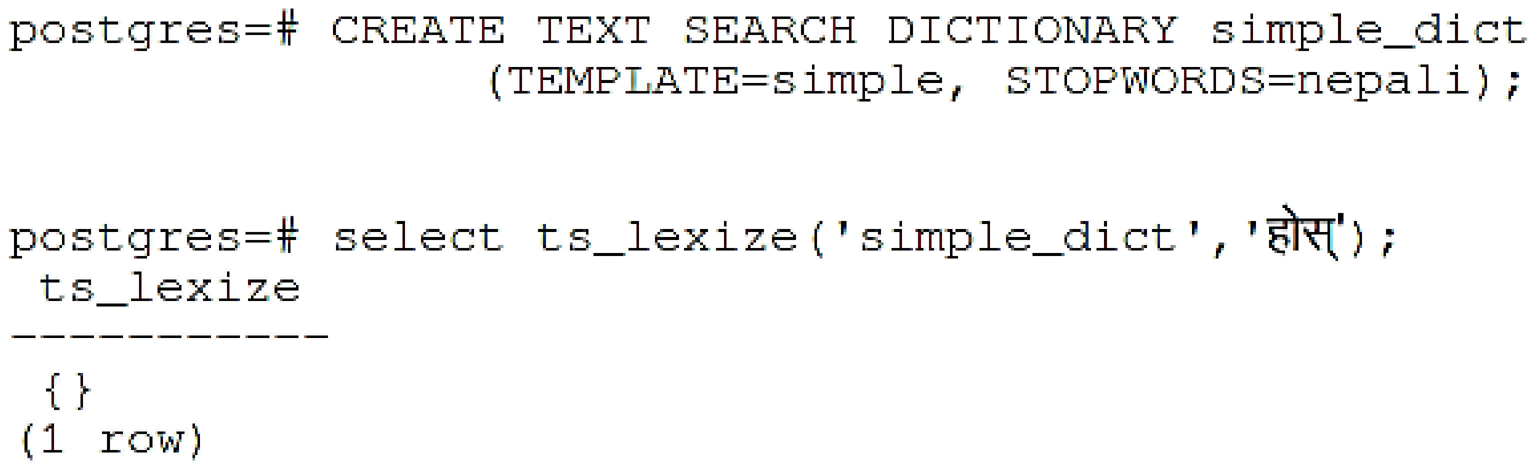
अक्सर
अगाडि
अगाडी
अझै
अनुसार
अन्तर्गत
अन्य
अन्यथा
अब
अरु

Why Remove the Stop Words?

- They normally don’t add any value to the outcome of analysis
- So, it is not necessary to store them in an index
- Improves search performance

Nepali NLP Group

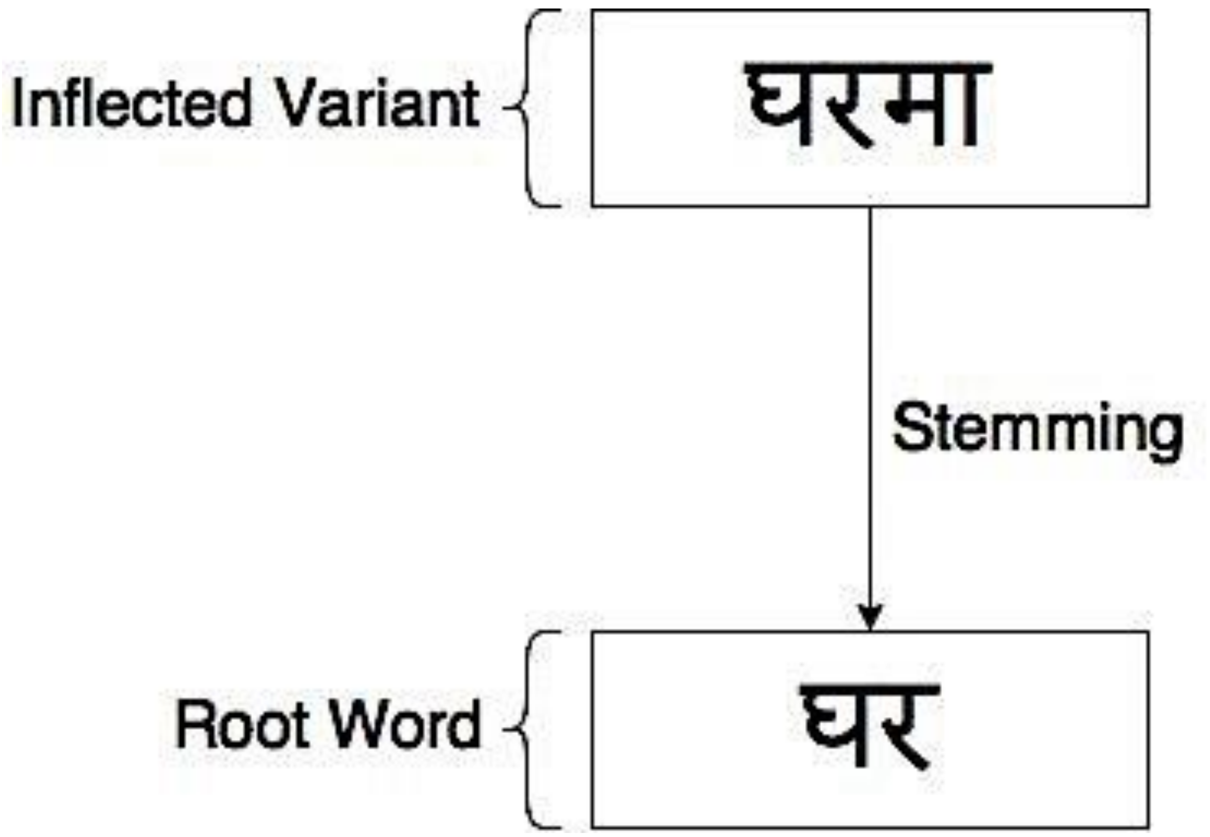
FTS Dictionary : Stop Word



```
postgres=# CREATE TEXT SEARCH DICTIONARY simple_dict
                (TEMPLATE=simple, STOPWORDS=nepali);

postgres=# select ts_lexize('simple_dict','होस');
 ts_lexize
-----------
 {}
(1 row)
```

Nepali NLP Group

Stemming Algorithm



Why is Stemming Important?

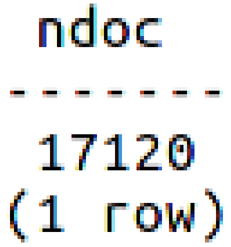
Stemming improves search performance by:

- Mapping all the inflected variants of a word to its stem
- Reducing the size of *tsvector* representation of the document

With stemming the same word was found in more number of documents!

With Stemming

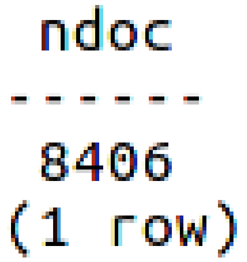
```
postgres=# SELECT  count(*) as ndoc
FROM news, to_tsquery('nepali','नेपाल') q
WHERE fts @@ q;
```



```
 ndoc
-------
 17120
(1 row)
```

Without Stemming

```
postgres=# SELECT count(*) as ndoc
FROM news
WHERE body @@
to_tsquery('nepali','नेपाल’);
```

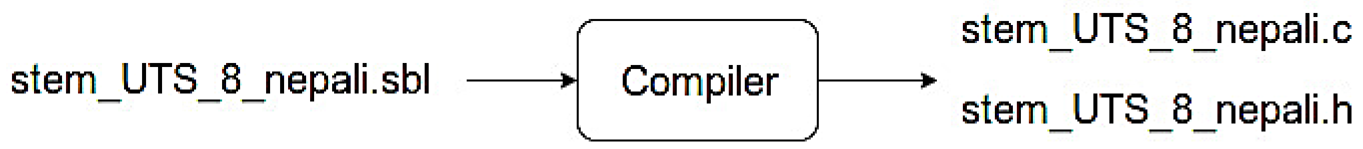


```
 ndoc
------
 8406
(1 row)
```

Nepali Snowball Stemmer

- Based on **Snowball Language** by *Martin Porter* and **A New Stemmer for Nepali Language(2016)** by *Ingroj Shrestha and Shreeya Singh Dhakal*

- **Snowball Stemmer for Nepali**



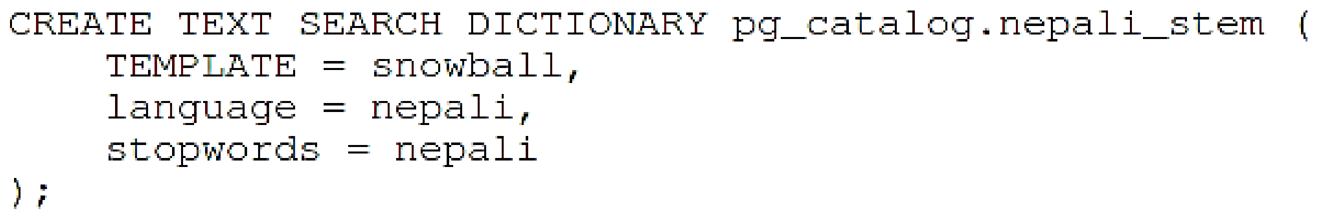
Nepali Snowball Stemmer

A New Stemmer for Nepali Language(2016)

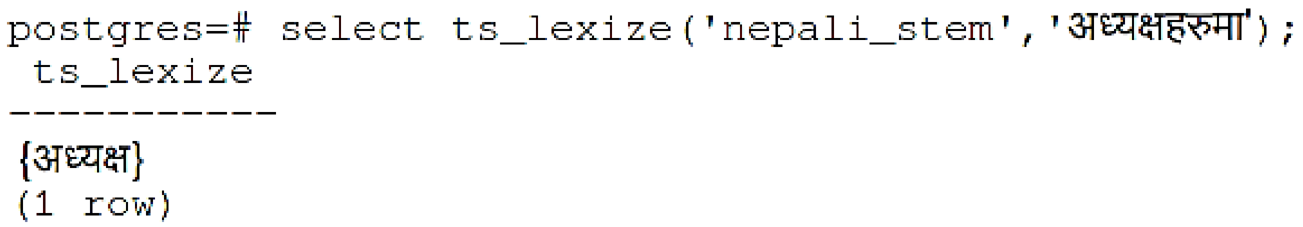
https://ieeexplore.ieee.org/document/7749008/

- Iterative rule-based stemming algorithm
- Strips inflectional suffixes from words
- Works in three parts

FTS Dictionary : Snowball



```
CREATE TEXT SEARCH DICTIONARY pg_catalog.nepali_stem (
    TEMPLATE = snowball,
    language = nepali,
    stopwords = nepali
);
```



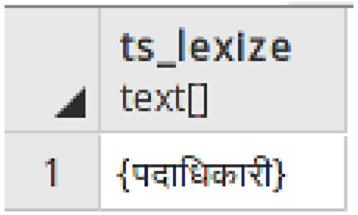
```
postgres=# select ts_lexize('nepali_stem','अध्यक्षहरुमा');
 ts_lexize
-----------
 {अध्यक्ष}
(1 row)
```

Dictionary Testing

ts_lexize

```
ts_lexize(dict regdictionary, token text) returns text[]
```

postgres =# SELECT ts_lexize('nepali_stem','पदाधिकारीहरुको');



	ts_lexize text[]
1	{पदाधिकारी}

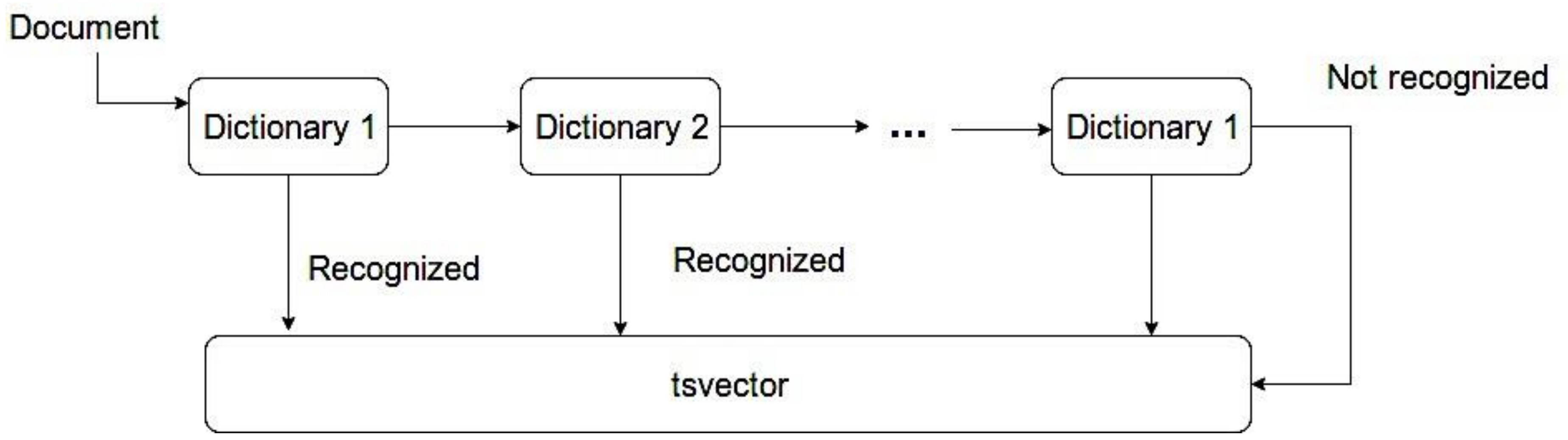
Adding Nepali snowball stemmer and stop word dictionary:

- Reduced the size of document by approximately *48%* (Over a document of *285447* unique words)
- This in turn reduced the size of *tsvector* representation of the document
- Hence, improving the search performance

Nepali NLP Group

FTS Configuration

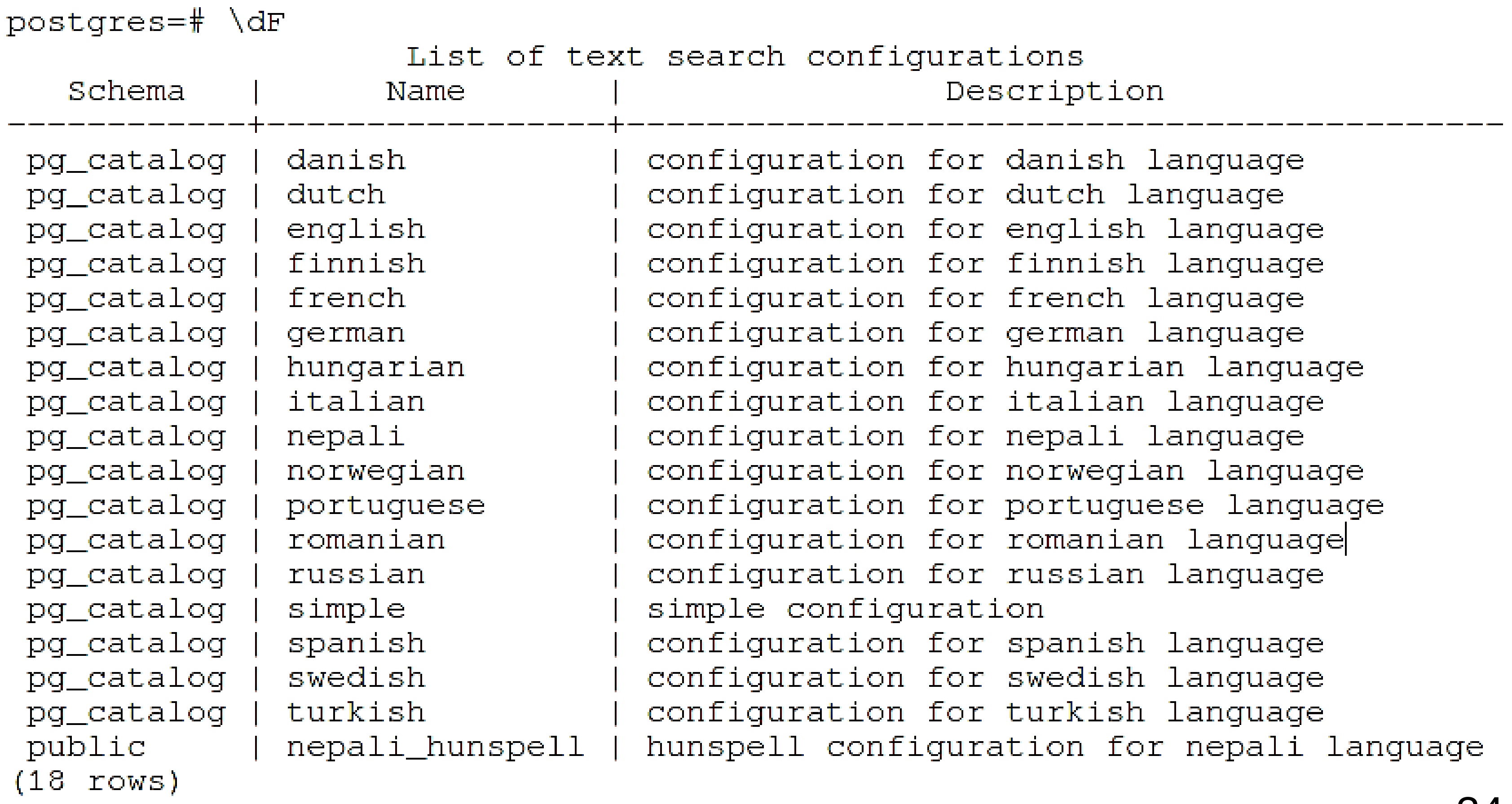
- Binds parser and dictionaries
- Specifies how the document should be processed
- Maps a dictionaries pipeline for each token type



FTS Configuration

PSQL command

\dF{,d,p,t}[+][PATTERN]



```
postgres=# \dF
                     List of text search configurations
   Schema    |      Name       |               Description
------------+-----------------+---------------------------------------------
 pg_catalog | danish          | configuration for danish language
 pg_catalog | dutch           | configuration for dutch language
 pg_catalog | english         | configuration for english language
 pg_catalog | finnish         | configuration for finnish language
 pg_catalog | french          | configuration for french language
 pg_catalog | german          | configuration for german language
 pg_catalog | hungarian       | configuration for hungarian language
 pg_catalog | italian         | configuration for italian language
 pg_catalog | nepali          | configuration for nepali language
 pg_catalog | norwegian       | configuration for norwegian language
 pg_catalog | portuguese      | configuration for portuguese language
 pg_catalog | romanian        | configuration for romanian language
 pg_catalog | russian         | configuration for russian language
 pg_catalog | simple          | simple configuration
 pg_catalog | spanish         | configuration for spanish language
 pg_catalog | swedish         | configuration for swedish language
 pg_catalog | turkish         | configuration for turkish language
 public     | nepali_hunspell | hunspell configuration for nepali language
(18 rows)
```

Nepali NLP Group

FTS Configuration

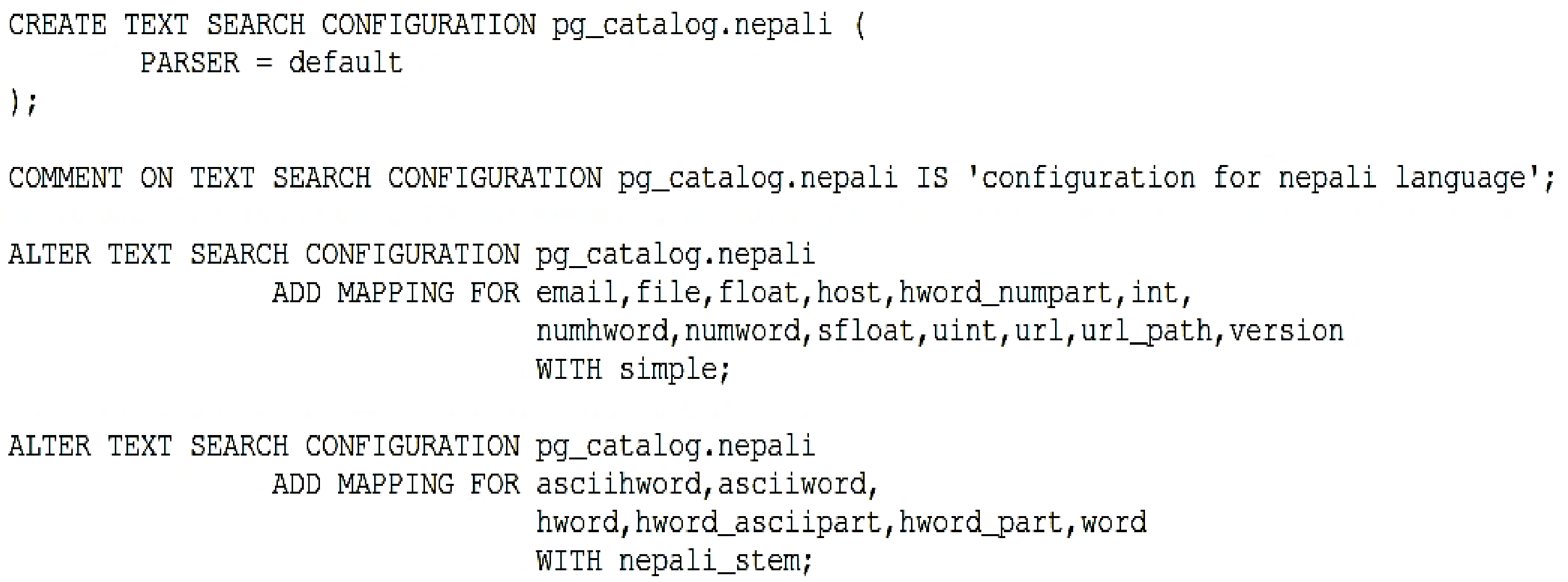
```
postgres=# \dF+ nepali
Text search configuration "pg_catalog.nepali"
Parser: "pg_catalog.default"
      Token       | Dictionaries
-----------------+---------------
 asciihword      | nepali_stem
 asciiword       | nepali_stem
 email           | simple
 file            | simple
 float           | simple
 host            | simple
 hword           | nepali_stem
 hword_asciipart | nepali_stem
 hword_numpart   | simple
 hword_part      | nepali_stem
 int             | simple
 numhword        | simple
 numword         | simple
 sfloat          | simple
 uint            | simple
 url             | simple
 url_path        | simple
 version         | simple
 word            | nepali_stem
```

FTS Configuration

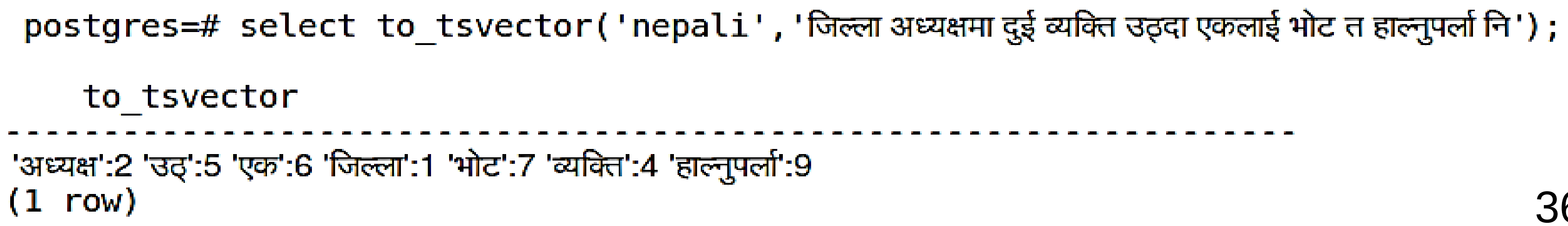
Nepali NLP Group

SQL command

{CREATE | ALTER | DROP} TEXT SEARCH {**CONFIGURATION** | DICTIONARY | PARSER}



```
CREATE TEXT SEARCH CONFIGURATION pg_catalog.nepali (
        PARSER = default
);

COMMENT ON TEXT SEARCH CONFIGURATION pg_catalog.nepali IS 'configuration for nepali language';

ALTER TEXT SEARCH CONFIGURATION pg_catalog.nepali
                ADD MAPPING FOR email,file,float,host,hword_numpart,int,
                                numhword,numword,sfloat,uint,url,url_path,version
                                WITH simple;

ALTER TEXT SEARCH CONFIGURATION pg_catalog.nepali
                ADD MAPPING FOR asciihword,asciiword,
                                hword,hword_asciipart,hword_part,word
                                WITH nepali_stem;
```



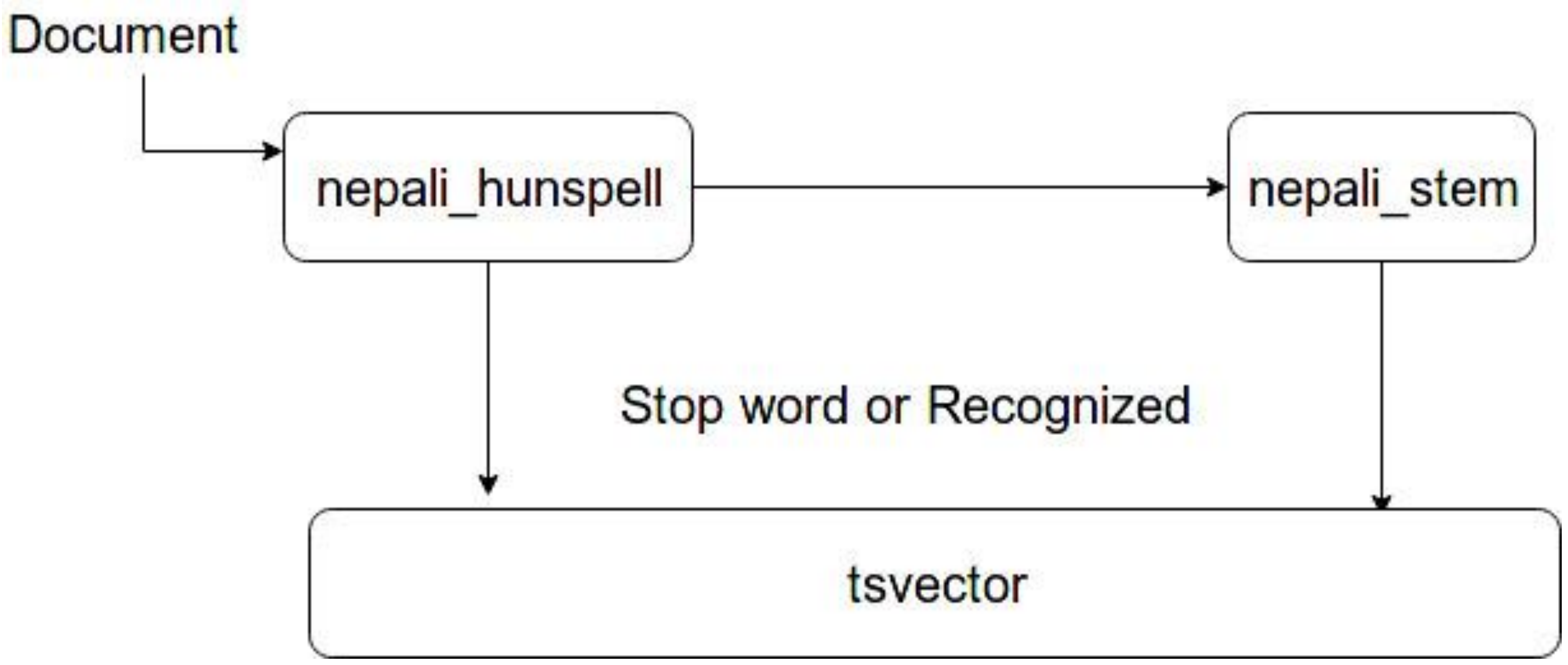
```
 postgres=# select to_tsvector('nepali','जिल्ला अध्यक्षमा दुई व्यक्ति उढ़दा एकलाई भोट त हाल्नुपर्ला नि');

    to_tsvector
-----------------------------------------------------------------
'अध्यक्ष':2 'उढ्':5 'एक':6 'जिल्ला':1 'भोट':7 'व्यक्ति':4 'हाल्नुपर्ला':9
(1 row)
```

Nepali NLP Group

FTS Configuration Example

```
CREATE TEXT SEARCH CONFIGURATION pg_catalog.nepali_config (
        PARSER = default
);

COMMENT ON TEXT SEARCH CONFIGURATION pg_catalog.nepali_config IS 'new configuration for nepali language';

ALTER TEXT SEARCH CONFIGURATION pg_catalog.nepali_config
                ADD MAPPING FOR email,file,float,host,hword_numpart,int,
                                numhword,numword,sfloat,uint,url,url_path,version
                                WITH simple;

ALTER TEXT SEARCH CONFIGURATION pg_catalog.nepali_config
                        ADD MAPPING FOR asciihword,asciiword,
                        hword,hword_asciipart,hword_part,word
                        WITH nepali_hunspell,nepali_stem;
```

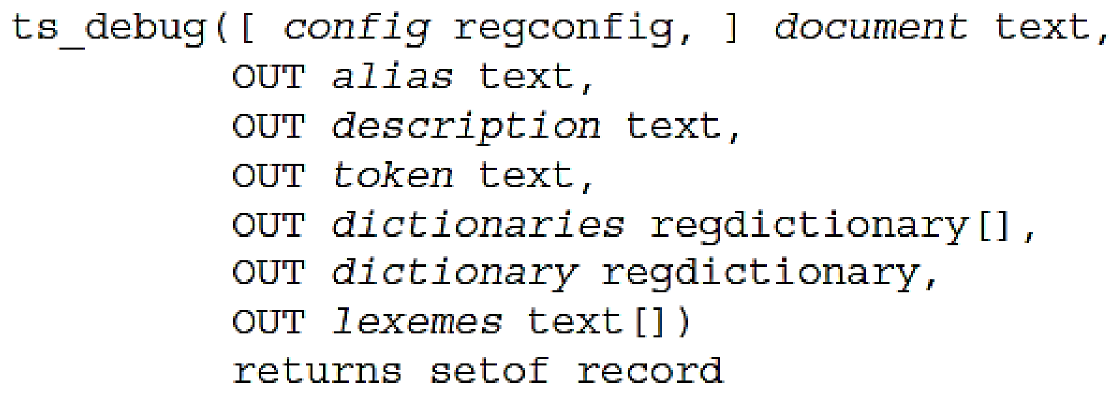
FTS Configuration Example



Configuration Testing

ts_debug

- Provides information about each token produced by parser and processed by the dictionaries



```
ts_debug([ config regconfig, ] document text,
         OUT alias text,
         OUT description text,
         OUT token text,
         OUT dictionaries regdictionary[],
         OUT dictionary regdictionary,
         OUT lexemes text[])
         returns setof record
```

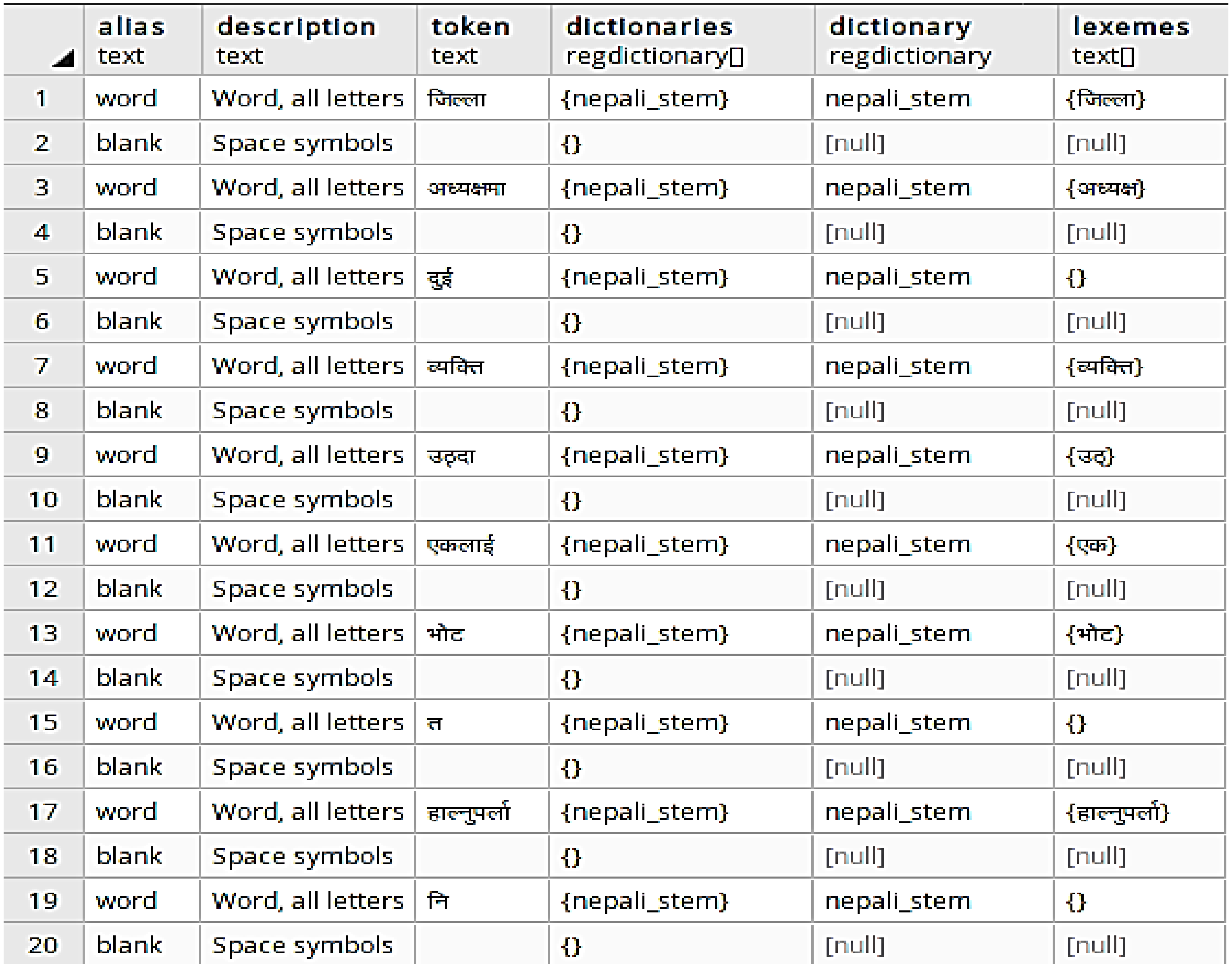
postgres =# SELECT * FROM ts_debug('nepali','जिल्ला अध्यक्षमा दुई व्यक्ति उठ्दा एकलाई भोट त हाल्नुपर्ला नि ');

Explicit use of configuration name is important!!

PGConf Nepal 2018
May 04- 05, 2018

Nepali NLP Group

Configuration Testing



	alias	description	token	dictionaries	dictionary	lexemes
	text	text	text	regdictionary[]	regdictionary	text[]
1	word	Word, all letters	जिल्ला	{nepali_stem}	nepali_stem	{जिल्ला}
2	blank	Space symbols		{}	[null]	[null]
3	word	Word, all letters	अध्यक्षमा	{nepali_stem}	nepali_stem	{अध्यक्ष}
4	blank	Space symbols		{}	[null]	[null]
5	word	Word, all letters	दुई	{nepali_stem}	nepali_stem	{}
6	blank	Space symbols		{}	[null]	[null]
7	word	Word, all letters	व्यक्ति	{nepali_stem}	nepali_stem	{व्यक्ति}
8	blank	Space symbols		{}	[null]	[null]
9	word	Word, all letters	उठ्दा	{nepali_stem}	nepali_stem	{उठ}
10	blank	Space symbols		{}	[null]	[null]
11	word	Word, all letters	एकलाई	{nepali_stem}	nepali_stem	{एक}
12	blank	Space symbols		{}	[null]	[null]
13	word	Word, all letters	भोट	{nepali_stem}	nepali_stem	{भोट}
14	blank	Space symbols		{}	[null]	[null]
15	word	Word, all letters	त	{nepali_stem}	nepali_stem	{}
16	blank	Space symbols		{}	[null]	[null]
17	word	Word, all letters	हाल्नुपर्ला	{nepali_stem}	nepali_stem	{हाल्नुपर्ला}
18	blank	Space symbols		{}	[null]	[null]
19	word	Word, all letters	नि	{nepali_stem}	nepali_stem	{}
20	blank	Space symbols		{}	[null]	[null]

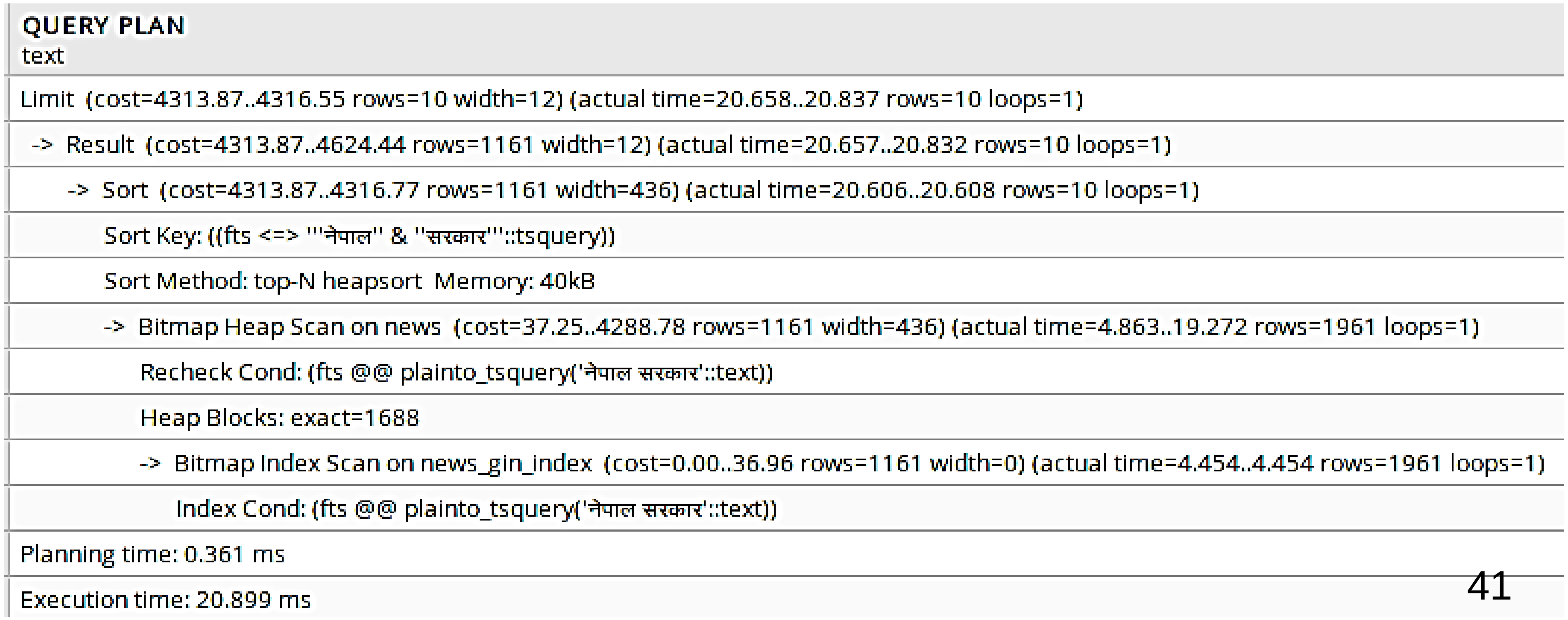
PGConf Nepal 2018
May 04- 05, 2018

Nepali NLP Group

Importance of Explicit use of Configuration Name

to_tsquery(regconfig, text) - Immutable, evaluated by optimizer and cached

postgres =# EXPLAIN ANALYSE SELECT body, ts_rank_cd(fts,plainto_tsquery('नेपाल सरकार')) AS rank FROM news WHERE fts @@ plainto_tsquery('नेपाल सरकार') ORDER BY fts <=> plainto_tsquery('nepali','नेपाल सरकार') LIMIT 10;



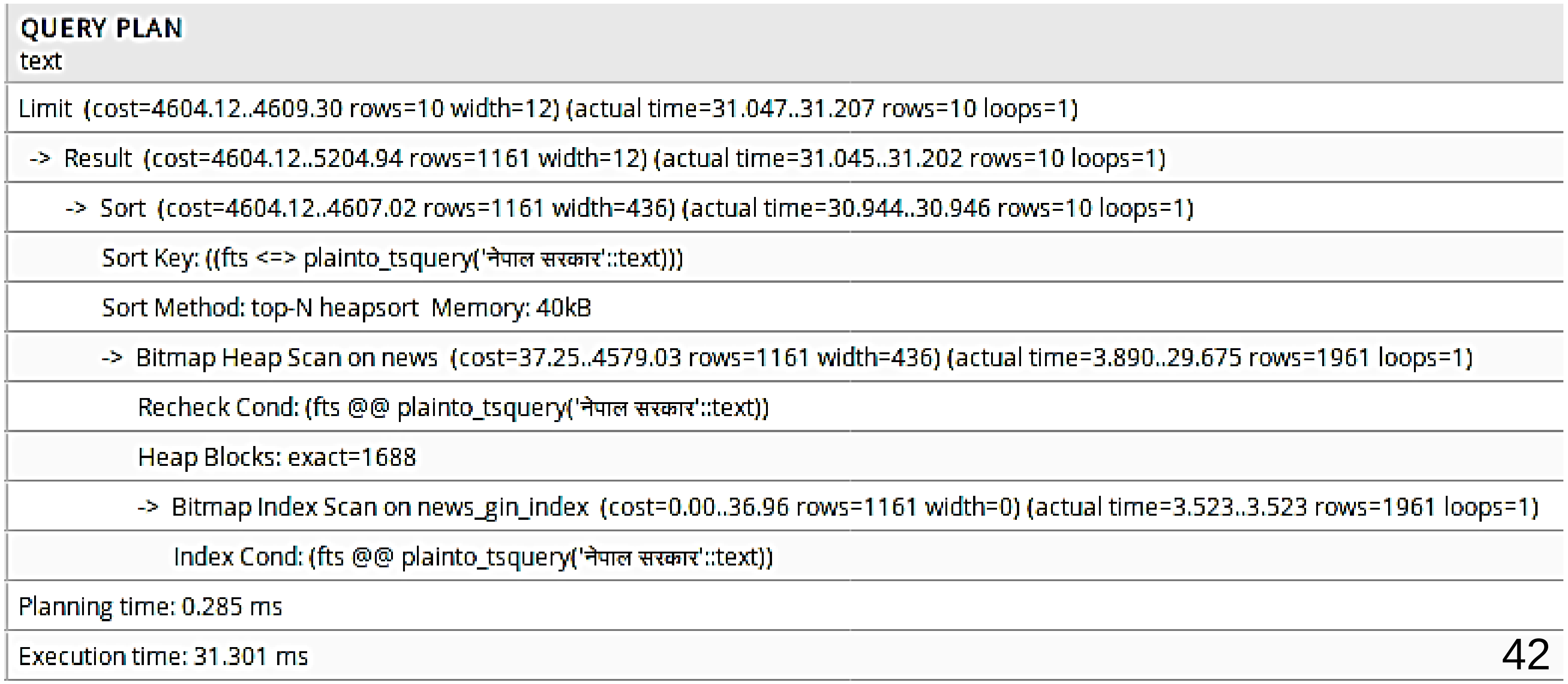
```
QUERY PLAN
text
Limit  (cost=4313.87..4316.55 rows=10 width=12) (actual time=20.658..20.837 rows=10 loops=1)
  ->  Result  (cost=4313.87..4624.44 rows=1161 width=12) (actual time=20.657..20.832 rows=10 loops=1)
        ->  Sort  (cost=4313.87..4316.77 rows=1161 width=436) (actual time=20.606..20.608 rows=10 loops=1)
              Sort Key: ((fts <=> '''नेपाल'' & ''सरकार'''::tsquery))
              Sort Method: top-N heapsort  Memory: 40kB
              ->  Bitmap Heap Scan on news  (cost=37.25..4288.78 rows=1161 width=436) (actual time=4.863..19.272 rows=1961 loops=1)
                    Recheck Cond: (fts @@ plainto_tsquery('नेपाल सरकार'::text))
                    Heap Blocks: exact=1688
                    ->  Bitmap Index Scan on news_gin_index  (cost=0.00..36.96 rows=1161 width=0) (actual time=4.454..4.454 rows=1961 loops=1)
                          Index Cond: (fts @@ plainto_tsquery('नेपाल सरकार'::text))
Planning time: 0.361 ms
Execution time: 20.899 ms
```

Nepali NLP Group

Importance of Explicit use of Configuration Name

to_tsquery(text) – stable and each tuple is evaluated

postgres =# EXPLAIN ANALYSE SELECT body, ts_rank_cd(fts,plainto_tsquery('नेपाल सरकार')) AS rank FROM news WHERE fts @@ plainto_tsquery('नेपाल सरकार') ORDER BY fts <=> plainto_tsquery('नेपाल सरकार') LIMIT 10;



```
QUERY PLAN
text
Limit  (cost=4604.12..4609.30 rows=10 width=12) (actual time=31.047..31.207 rows=10 loops=1)
  -> Result  (cost=4604.12..5204.94 rows=1161 width=12) (actual time=31.045..31.202 rows=10 loops=1)
        -> Sort  (cost=4604.12..4607.02 rows=1161 width=436) (actual time=30.944..30.946 rows=10 loops=1)
              Sort Key: ((fts <=> plainto_tsquery('नेपाल सरकार'::text)))
              Sort Method: top-N heapsort  Memory: 40kB
              -> Bitmap Heap Scan on news  (cost=37.25..4579.03 rows=1161 width=436) (actual time=3.890..29.675 rows=1961 loops=1)
                    Recheck Cond: (fts @@ plainto_tsquery('नेपाल सरकार'::text))
                    Heap Blocks: exact=1688
                    -> Bitmap Index Scan on news_gin_index  (cost=0.00..36.96 rows=1161 width=0) (actual time=3.523..3.523 rows=1961 loops=1)
                          Index Cond: (fts @@ plainto_tsquery('नेपाल सरकार'::text))
Planning time: 0.285 ms
Execution time: 31.301 ms
```

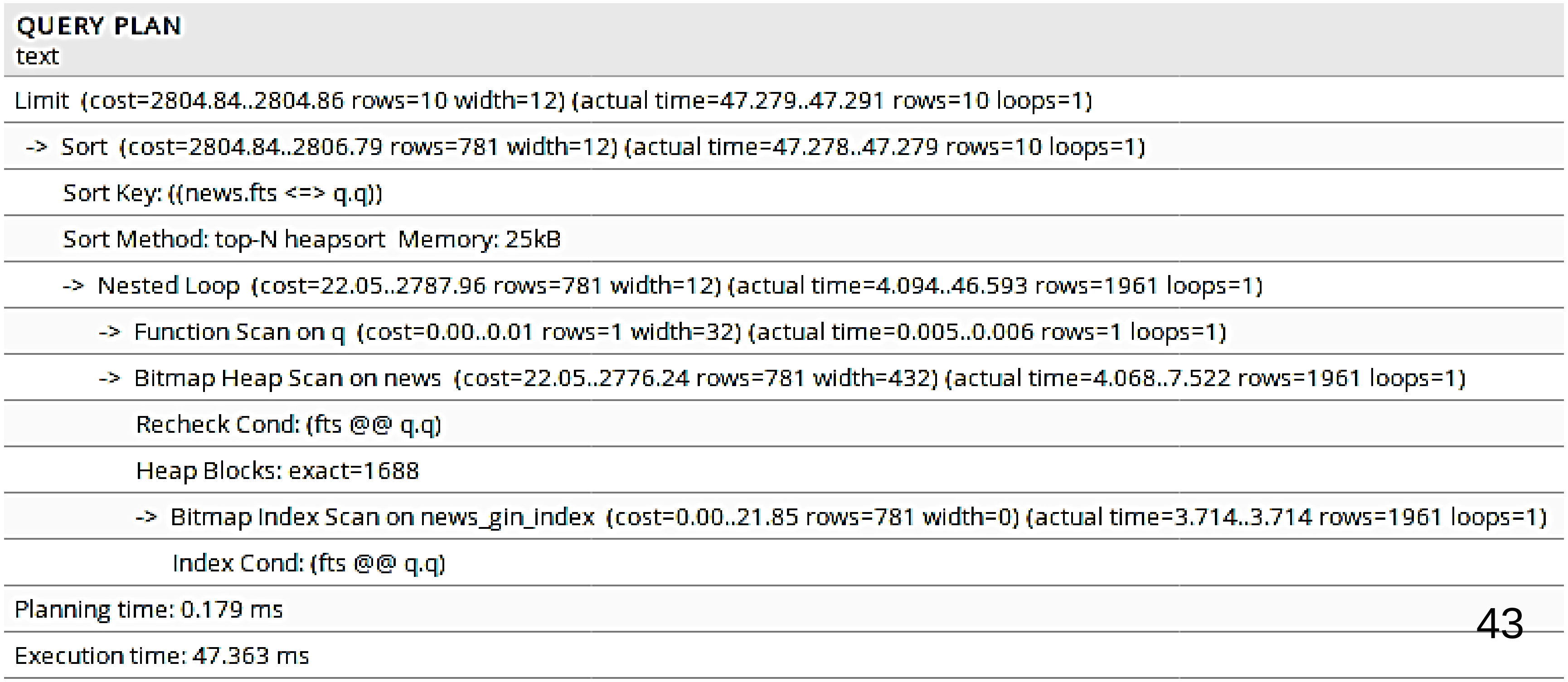
PGConf Nepal 2018
May 04- 05, 2018

Nepali NLP Group

Importance of Explicit use of Configuration Name

- Function scan
- Query almost 2x slower!!

```
postgres =# EXPLAIN ANALYSE SELECT body, ts_rank_cd(fts,q) AS rank
FROM news, plainto_tsquery('nepali','नेपाल सरकार') q
WHERE  fts @@ q ORDER BY fts <=> q LIMIT 10;
```

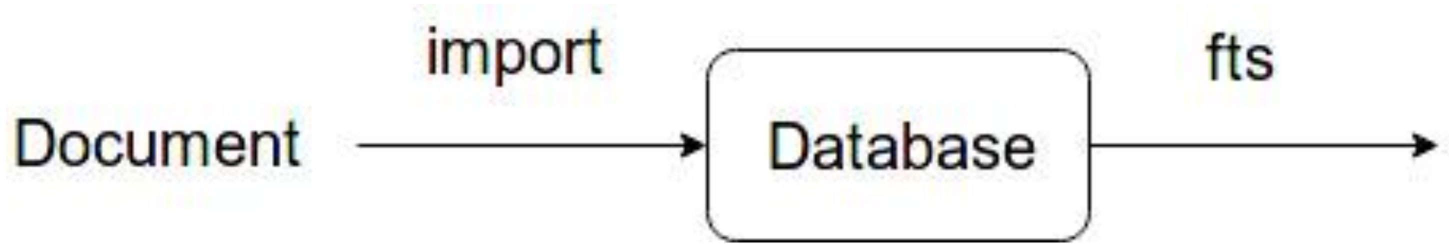


```
QUERY PLAN
text
Limit  (cost=2804.84..2804.86 rows=10 width=12) (actual time=47.279..47.291 rows=10 loops=1)
  ->  Sort  (cost=2804.84..2806.79 rows=781 width=12) (actual time=47.278..47.279 rows=10 loops=1)
        Sort Key: ((news.fts <=> q.q))
        Sort Method: top-N heapsort  Memory: 25kB
        ->  Nested Loop  (cost=22.05..2787.96 rows=781 width=12) (actual time=4.094..46.593 rows=1961 loops=1)
              ->  Function Scan on q  (cost=0.00..0.01 rows=1 width=32) (actual time=0.005..0.006 rows=1 loops=1)
              ->  Bitmap Heap Scan on news  (cost=22.05..2776.24 rows=781 width=432) (actual time=4.068..7.522 rows=1961 loops=1)
                    Recheck Cond: (fts @@ q.q)
                    Heap Blocks: exact=1688
                    ->  Bitmap Index Scan on news_gin_index  (cost=0.00..21.85 rows=781 width=0) (actual time=3.714..3.714 rows=1961 loops=1)
                          Index Cond: (fts @@ q.q)
Planning time: 0.179 ms
Execution time: 47.363 ms
```

Nepali News: FTS

Document

- Textual field within a row of database table
- Combination of fields from the same or different tables (join) – virtual document



Nepali News Example

```
postgres=# CREATE TABLE public.news
(
  id integer,
  title text COLLATE pg_catalog.”default”,
  body text COLLATE pg_catalog.”default”,
  date date
)
WITH(
  OIDS = FALSE
)
TABLESPACE pg_default;

ALTER TABLE public.news
   OWNER to postgres;
```

Nepali NLP Group

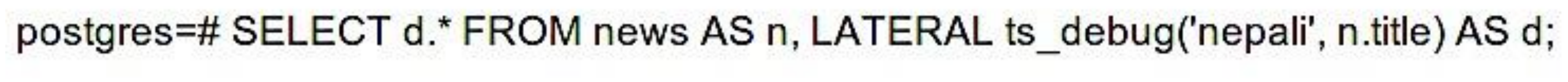
Nepali News Example : Importing to Database

```
postgres=# COPY news(id, title, body, date) FROM
‘/path/to/news.csv' DELIMITER ',' CSV HEADER;
```

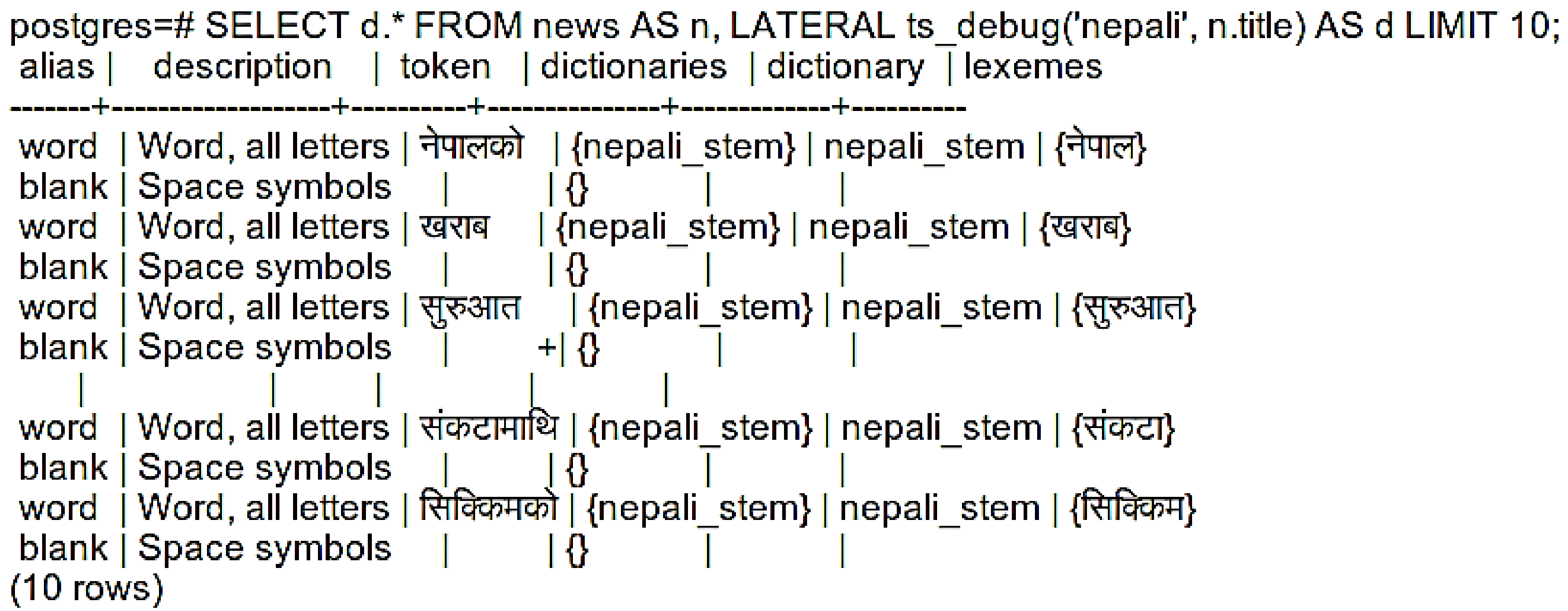
PGConf Nepal 2018
May 04- 05, 2018

Nepali NLP Group

Nepali News: FTS in Database



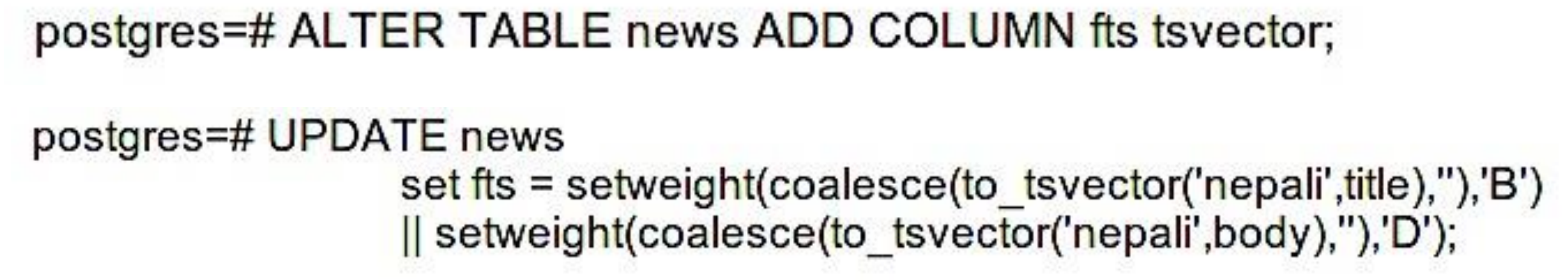
```
postgres=# SELECT d.* FROM news AS n, LATERAL ts_debug('nepali', n.title) AS d;
```



```
postgres=# SELECT d.* FROM news AS n, LATERAL ts_debug('nepali', n.title) AS d LIMIT 10;
 alias |    description    |  token   | dictionaries  | dictionary  | lexemes
-------+-------------------+----------+---------------+-------------+----------
 word  | Word, all letters | नेपालको    | {nepali_stem} | nepali_stem | {नेपाल}
 blank | Space symbols     |          | {}            |             |
 word  | Word, all letters | खराब     | {nepali_stem} | nepali_stem | {खराब}
 blank | Space symbols     |          | {}            |             |
 word  | Word, all letters | सुरुआत     | {nepali_stem} | nepali_stem | {सुरुआत}
 blank | Space symbols     |         +| {}            |             |
       |                   |          |               |             |
 word  | Word, all letters | संकटामाथि | {nepali_stem} | nepali_stem | {संकटा}
 blank | Space symbols     |          | {}            |             |
 word  | Word, all letters | सिक्किमको | {nepali_stem} | nepali_stem | {सिक्किम}
 blank | Space symbols     |          | {}            |             |
(10 rows)
```

FTS Representation of Nepali News

fts: tsvector



```
postgres=# ALTER TABLE news ADD COLUMN fts tsvector;

postgres=# UPDATE news
                        set fts = setweight(coalesce(to_tsvector('nepali',title),''),'B')
                        || setweight(coalesce(to_tsvector('nepali',body),''),'D');
```

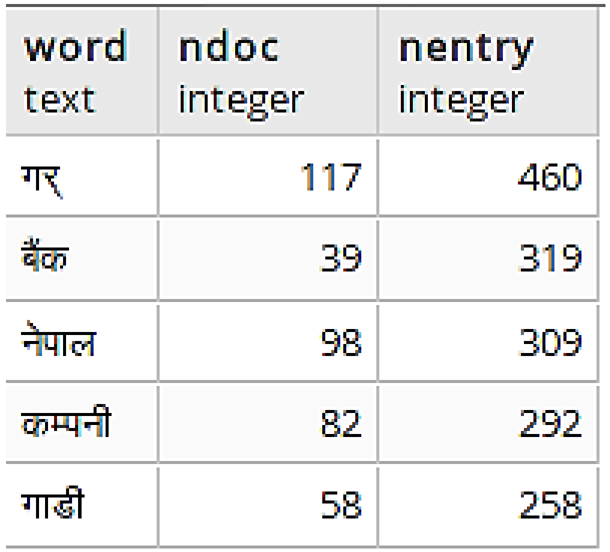
Nepali News: Document Statistics

ts_stat

Syntax: ts_stat(*sqlquery* text, [*weights* text,], OUT *word* text, OUT *ndoc* text, OUT *nentry* integer) return setof record

Find five most frequent words in document collection

```
postgres =# SELECT * FROM ts_stat('SELECT fts FROM news')
                         ORDER BY nentry DESC, ndoc DESC, word LIMIT 5;
```



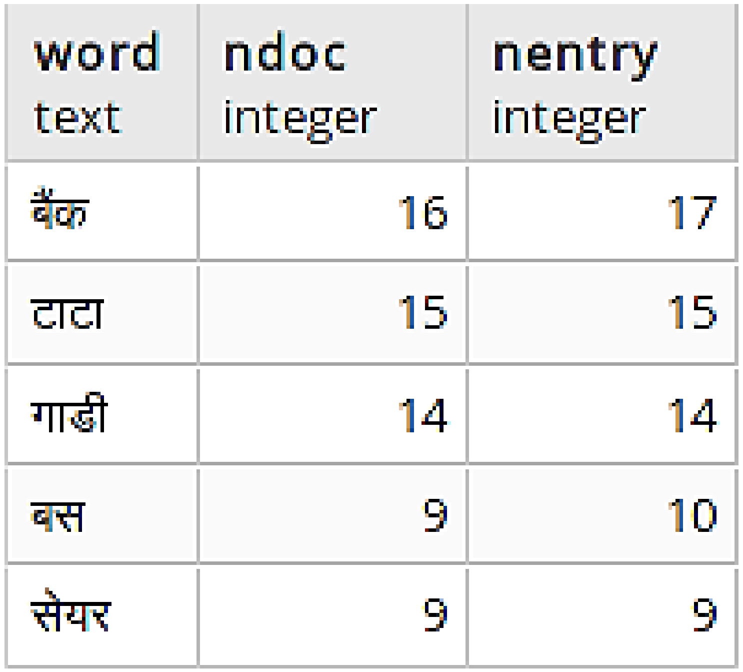
word text	ndoc integer	nentry integer
गर्	117	460
बैंक	39	319
नेपाल	98	309
कम्पनी	82	292
गाडी	58	258

Nepali NLP Group

Nepali News: Document Statistics

Find five most frequent words in document collection but counting only word occurrence with weights A and B

```
postgres=# SELECT * FROM ts_stat('SELECT fts FROM news','ab')
                                        ORDER BY nentry DESC, ndoc DESC, word LIMIT 5;
```



word	ndoc	nentry
text	integer	integer
बैंक	16	17
टाटा	15	15
गाडी	14	14
बस	9	10
सेयर	9	9

Nepali News: Ranking Search Results

ts_rank

- Ranks vector on the basis of the frequency of their matching lexemes

```
ts_rank([ weights float4[], ] vector tsvector, query tsquery [,
normalization integer ]) returns float4
```

ts_rank_cd

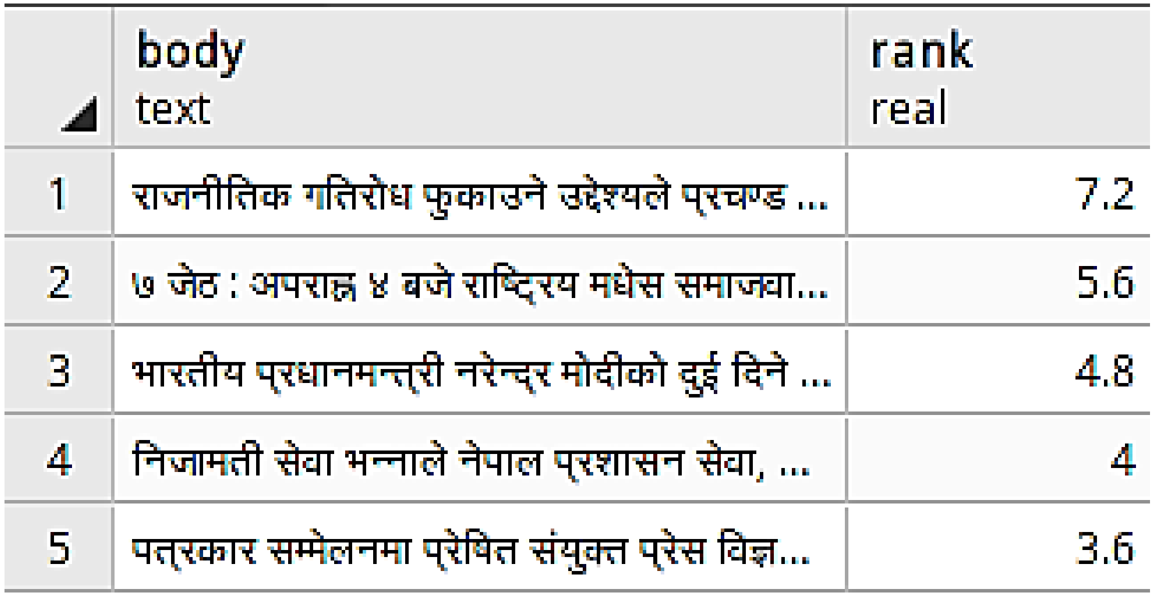
- Calculates cover density rank for the given document vector and query

```
ts_rank_cd([ weights float4[], ] vector tsvector, query tsquery [,
normalization integer ]) returns float4
```

Nepali NLP Group

Nepali News: Ranking Search Results

```
postgres =# SELECT body, ts_rank_cd(fts, to_tsquery('nepali','नेपाल')) AS rank
FROM news WHERE fts @@ to_tsquery('nepali','नेपाल')
ORDER BY rank DESC LIMIT 5;
```



	body text	rank real
1	राजनीतिक गतिरोध फुकाउने उद्देश्यले प्रचण्ड ...	7.2
2	७ जेठ : अपराह्न ४ बजे राष्ट्रिय मधेस समाजवा...	5.6
3	भारतीय प्रधानमन्त्री नरेन्द्र मोदीको दुई दिने ...	4.8
4	निजामती सेवा भन्नाले नेपाल प्रशासन सेवा, ...	4
5	पत्रकार सम्मेलनमा प्रेषित संयुक्त प्रेस विज्ञ...	3.6

- ts_rank_cd uses only local information
- 0 < rank / (rank + 1) < 1
- ts_rank_cd ('{0.1, 0.2, 0.4, 1.0}', fts, q)

Nepali News: Highlighting Results

ts_headline

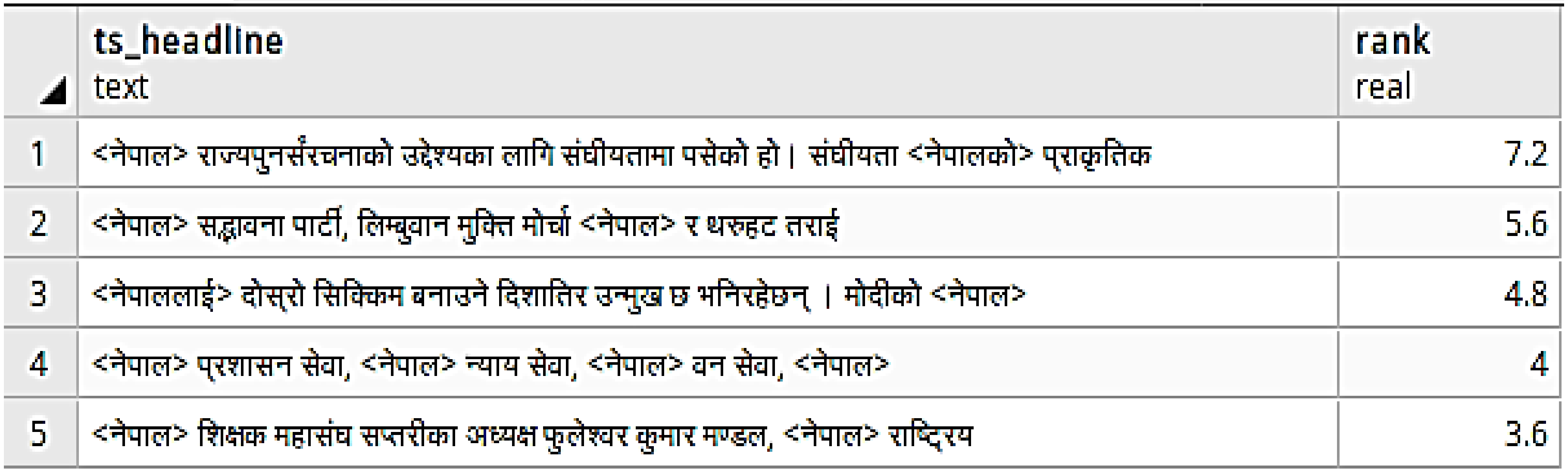
- Returns an excerpt from the document in which terms from query are highlighted

```
ts_headline([ config regconfig, ] document text, query tsquery
 [, options text ]) returns text
```

Nepali NLP Group

Nepali News: Highlighting Results

```
postgres=# SELECT ts_headline ('nepali',body, to_tsquery('nepali','नेपाल') ,'StartSel=<, StopSel=>, MaxWords= 20, MinWords=10'), ts_rank_cd(fts, to_tsquery('nepali','नेपाल') ) AS rank
FROM news WHERE fts @@ to_tsquery('nepali','नेपाल')
ORDER BY rank DESC LIMIT 5;
```



	ts_headline text	rank real
1	<नेपाल> राज्यपुनर्संरचनाको उद्देश्यका लागि संघीयतामा पसेको हो। संघीयता <नेपालको> प्राकृतिक	7.2
2	<नेपाल> सद्भावना पार्टी, लिम्बुवान मुक्ति मोर्चा <नेपाल> र थरुहट तराई	5.6
3	<नेपाललाई> दोस्रो सिक्किम बनाउने दिशातिर उन्मुख छ भनिरहेछन् । मोदीको <नेपाल>	4.8
4	<नेपाल> प्रशासन सेवा, <नेपाल> न्याय सेवा, <नेपाल> वन सेवा, <नेपाल>	4
5	<नेपाल> शिक्षक महासंघ सप्तरीका अध्यक्ष फुलेश्वर कुमार मण्डल, <नेपाल> राष्ट्रिय	3.6

Nepali News: Query Rewriting

ts_rewrite

- Search a given tsquery for occurrences of a target subquery
- Then, replace each occurrence with a substitute subquery

```
ts_rewrite (query tsquery, target tsquery, substitute tsquery) re-
turns tsquery
```

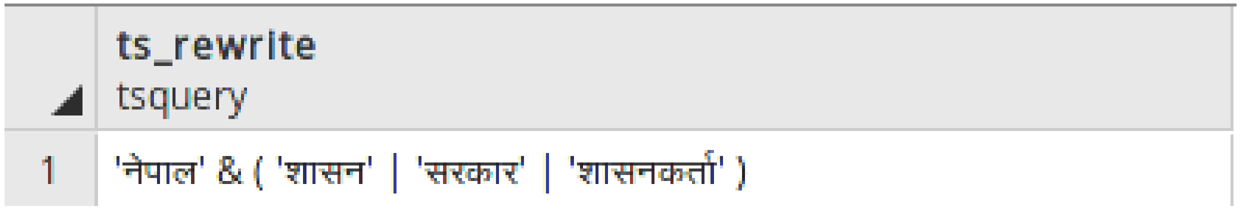
```
ts_rewrite (query tsquery, select text) returns tsquery
```

Nepali News: Query Rewriting

postgres =# CREATE TABLE aliases (t tsquery primary key, s tsquery);

postgres =# INSERT INTO aliases VALUES (to_tsquery('nepali','सरकार'), to_tsquery('nepali','सरकार | शासन | शासनकर्ता'));

postgres =# SELECT ts_rewrite (to_tsquery('nepali','नेपाल & सरकार'), 'SELECT * FROM aliases’);



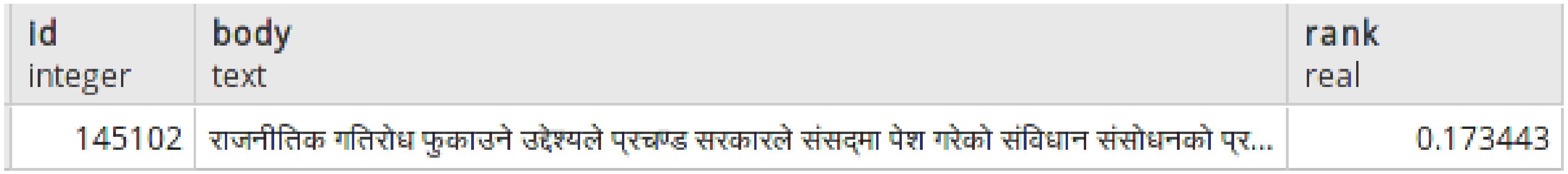
	ts_rewrite tsquery
1	'नेपाल' & ('शासन' | 'सरकार' | 'शासनकर्ता')

Nepali NLP Group

Nepali News: Query Rewriting

```
postgres=#  INSERT INTO aliases VALUES (to_tsquery('nepali','राज्य'),
to_tsquery('nepali','सरकार | शासन | शासनकर्ता | देश | राज्य | नेपाल'));
```

```
postgres=#  SELECT id, body, coalesce(ts_rank_cd(fts,
to_tsquery('nepali','राज्य'),1),2) AS rank
FROM news
WHERE fts @@ to_tsquery('nepali','राज्य') and id = 145102
ORDER BY rank  DESC ;
```

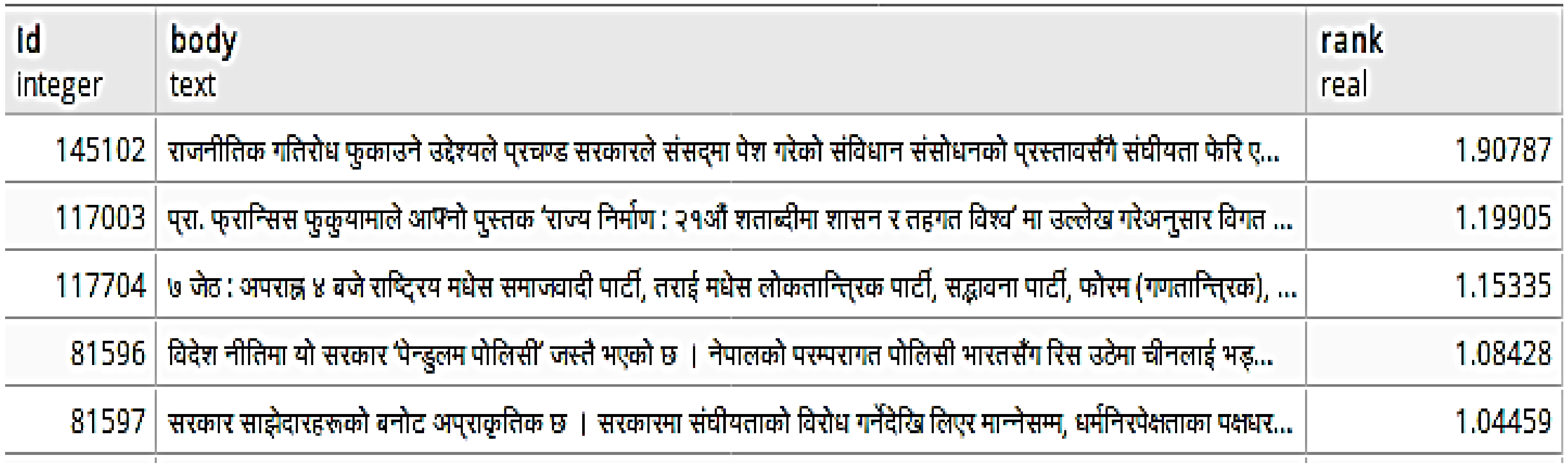


id integer	body text	rank real
145102	राजनीतिक गतिरोध फुकाउने उद्देश्यले प्रचण्ड सरकारले संसद्मा पेश गरेको संविधान संसोधनको प्र...	0.173443

Nepali NLP Group

Nepali News: Query Rewriting

```
postgres=#  SELECT body, coalesce(ts_rank_cd(fts,
ts_rewrite(to_tsquery('nepali','राज्य'), 'SELECT * FROM aliases'),1),2) AS
rank
FROM news WHERE fts @@ ts_rewrite(to_tsquery('nepali','राज्य'), 'SELECT
* FROM aliases')
ORDER BY rank  DESC LIMIT 5;
```

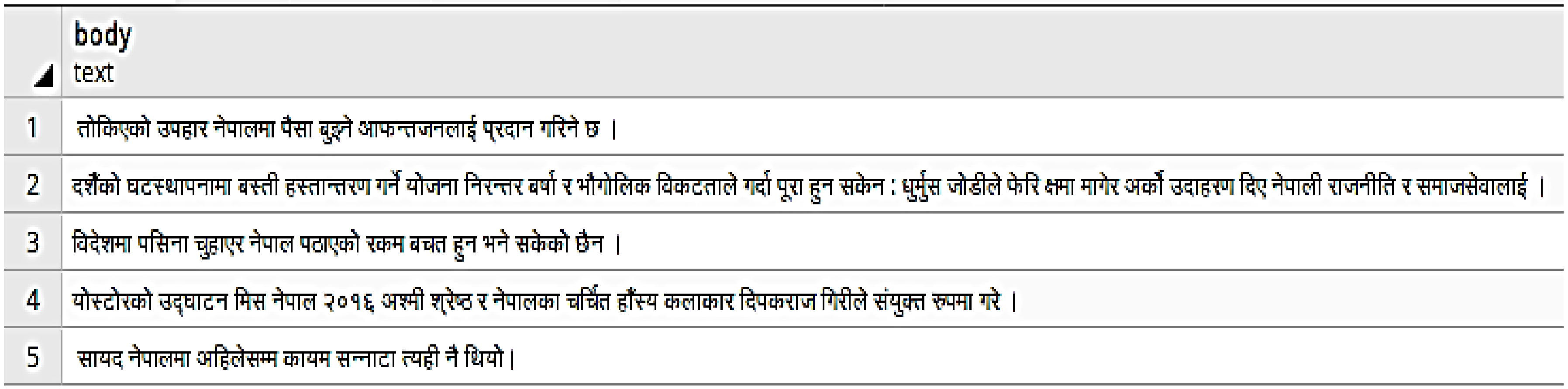


id integer	body text	rank real
145102	राजनीतिक गतिरोध फुकाउने उद्देश्यले प्रचण्ड सरकारले संसद्मा पेश गरेको संविधान संसोधनको प्रस्तावसँगै संघीयता फेरि ए...	1.90787
117003	प्रा. फरान्सिस फुकुयामाले आफ्नो पुस्तक 'राज्य निर्माण : २१औं शताब्दीमा शासन र तहगत विश्व' मा उल्लेख गरेअनुसार विगत ...	1.19905
117704	७ जेठ : अपराह्न ४ बजे राष्ट्रिय मधेस समाजवादी पार्टी, तराई मधेस लोकतान्त्रिक पार्टी, सद्भावना पार्टी, फोरम (गणतान्त्रिक), ...	1.15335
81596	विदेश नीतिमा यो सरकार 'पेन्डुलम पोलिसी' जस्तै भएको छ । नेपालको परम्परागत पोलिसी भारतसँग रिस उठेमा चीनलाई भड्...	1.08428
81597	सरकार साझेदारहरूको बनोट अप्राकृतिक छ । सरकारमा संघीयताको विरोध गर्नेदेखि लिएर मान्नेसम्म, धर्मनिरपेक्षताका पक्षधर...	1.04459

Nepali News: Prefix Search

```
postgres=# SELECT body FROM news WHERE fts @@
to_tsquery('नेपाल:*') LIMIT 5;
```

- Matches all documents, which contains words with prefix नेपाल
- ‘*’ signifies prefix match
- to_tsquery(‘नेपाल:bd*’) matches prefix on title(weight ‘b’) and body(weight ‘d’)
- Useful when stemmer is not available



	body text
1	तोकिएको उपहार नेपालमा पैसा बुझ्ने आफन्तजनलाई प्रदान गरिने छ ।
2	दशैंको घटस्थापनामा बस्ती हस्तान्तरण गर्ने योजना निरन्तर बर्षा र भौगोलिक विकटताले गर्दा पूरा हुन सकेन : धुर्मुस जोडीले फेरि क्षमा मागेर अर्को उदाहरण दिए नेपाली राजनीति र समाजसेवालाई ।
3	विदेशमा पसिना चुहाएर नेपाल पठाएको रकम बचत हुन भने सकेको छैन ।
4	योस्टोरको उद्घाटन मिस नेपाल २०१६ अश्मी श्रेष्ठ र नेपालका चर्चित हाँस्य कलाकार दिपकराज गिरीले संयुक्त रुपमा गरे ।
5	सायद नेपालमा अहिलेसम्म कायम सन्नाटा त्यही नै थियो।

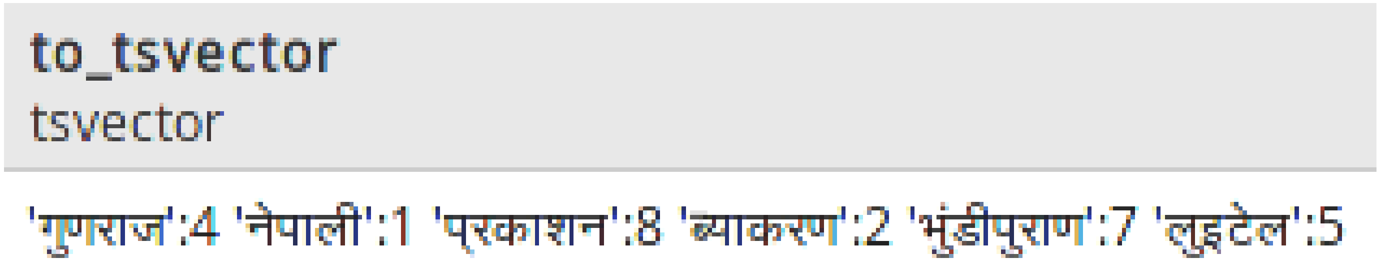
FTS of JSON[b] Data in PostgreSQL

JSONB

- Sorted key
- No duplicate keys (if duplicate last key is stored)
- Binary storage
    - does not require parsing
    - Index support
- Example: {"a":1,"b":{"c":2,"d":3},"e":[{"f":4,"g":5},{"f":6,"g":7}]}

FTS of JSON[b] Data in PostgreSQL

```
postgres=# SELECT to_tsvector('nepali',jb)
FROM (VALUES ('
{
 "title":"नेपाली ब्याकरण",
 "publisher":"भुंडीपुराण प्रकाशन",
 "writer":"गुणराज लुइटेल"
}
'::jsonb)) AS foo(jb)
```

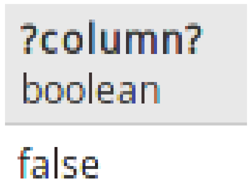


```
to_tsvector
tsvector
'गुणराज':4 'नेपाली':1 'प्रकाशन':8 'ब्याकरण':2 'भुंडीपुराण':7 'लुइटेल':5
```

Nepali NLP Group

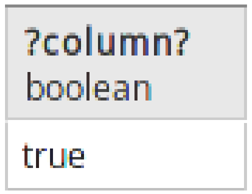
JSON[b] Data: Phrase Search

```
postgres=# phraseto_tsquery('nepali','नेपाली
प्रकाशन') @@ to_tsvector('nepali',jb) FROM
(VALUES ('
{
 "title":"नेपाली ब्याकरण",
 "publisher":"भुंडीपुराण प्रकाशन",
 "writer":"गुणराज लुइटेल"
}
'::jsonb)) AS foo(jb)
```



```
?column?
boolean
false
```

```
postgres=# SELECT
phraseto_tsquery('nepali','नेपाली ब्याकरण') @@
to_tsvector('nepali',jb) FROM (VALUES ('
{
 "title":"नेपाली ब्याकरण",
 "publisher":"भुंडीपुराण प्रकाशन",
 "writer":"गुणराज लुइटेल"
}
'::jsonb)) AS foo(jb)
```



```
?column?
boolean
true
```

Faceted Search in Single PostgreSQL Query

Faceted Search

- Results are organized according to category
- For each category
    - Obtain total number of matching documents
    - Obtain top N matching documents

Example

Implementing faceted search over *Nepali News* using windows functions and CTE

Nepali NLP Group

Faceted Search SQL Query

```
/*
* Select all matching news, calculate rank within list and total count
* within list using window functions.
*/
WITH res AS (
SELECT id, body, date,
    RANK() OVER (
        PARTITION BY date
        ORDER BY
ts_rank_cd(fts,  plainto_tsquery('nepali','नेपाल सरकार')), id
    ) rank,
    COUNT(*) OVER (PARTITION BY date) cnt
FROM news
WHERE fts @@ plainto_tsquery('nepali','नेपाल सरकार')
),
```

```
/* Aggregate news and count per list into json. */
lst AS (
SELECT
    date,
    jsonb_build_object(
        'count', cnt,
        'results', jsonb_agg(
            jsonb_build_object(
                'id', id,
                'body', body
    ))) AS data
FROM res
WHERE rank <= 2
GROUP by date, cnt
)
```

Faceted Search JSON Result

JSON document with total count of matching query on news and TOP 2 relevant news for each list

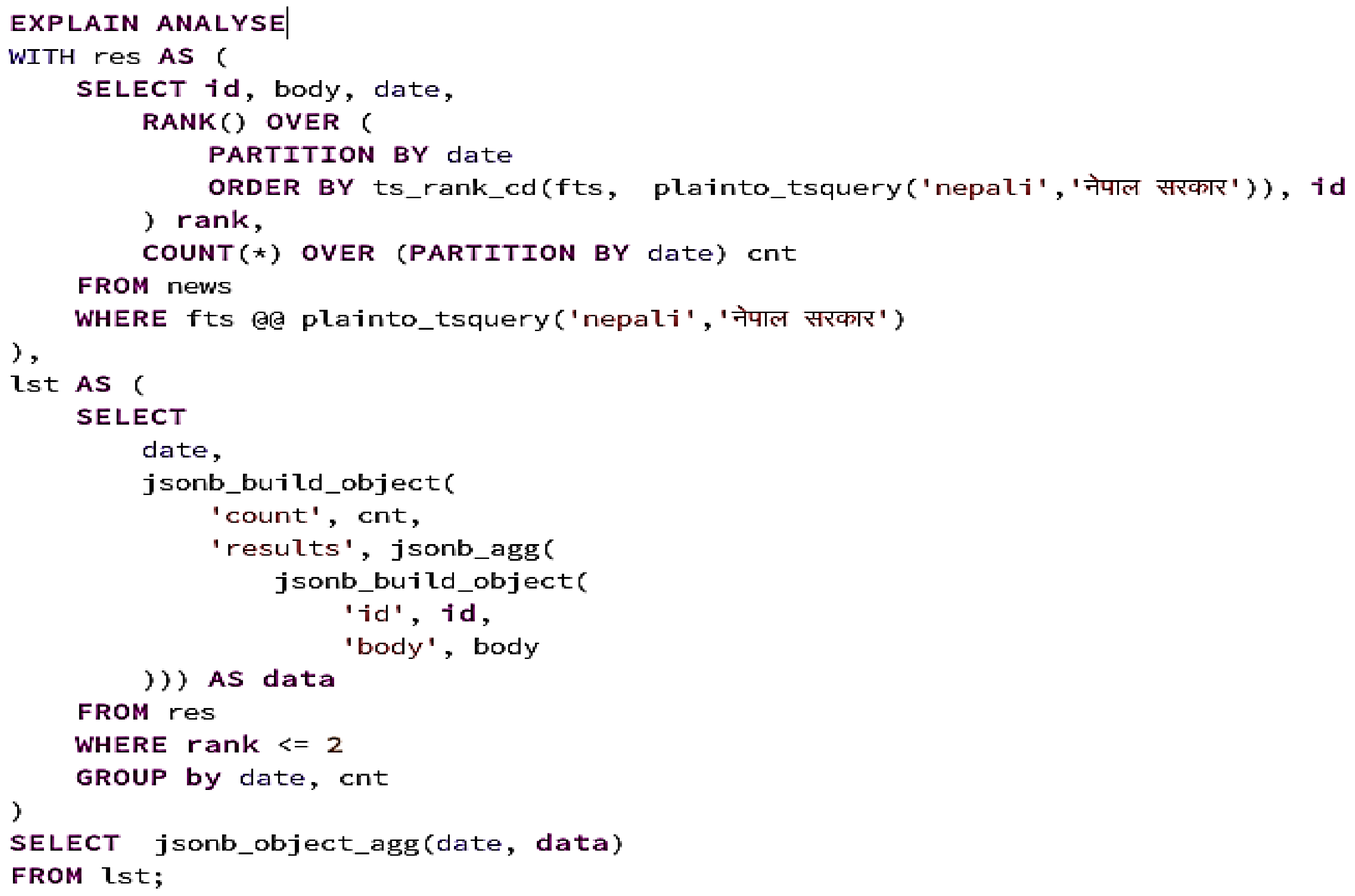
```
{
 "2018-01-10": {
"count": 841,
"results": [
  {
    "body": "आजभन्दा सात बर्ष अगाडि हामीसबै न्◌ेपाली मिलेर तानासाहि ...","id": 69814
  },
  {
    "body": "पार्टी भनेको जनताको परिवर्तन र मुक्तिको निम्ति साध्य होइन साधन बन्नुपर्छ । ...","id": 72652
  }
]
 },
 "2017-05-02": {
"count": 840,
"results": [
  {
    "body": "जनवरी–जुन २०१७, सप्तरीमानवअधिकार तथा सामाजिक न्यायका...","id": 156031
  },
  {
    "body": "यो हाम्रो दुर्भाग्य हो कि हामीले विगत इतिहासमा पाएका...","id": 128075
  }]
 }
}
```

PGConf Nepal 2018
May 04- 05, 2018

Nepali NLP Group

Faceted Search SQL Query Plan

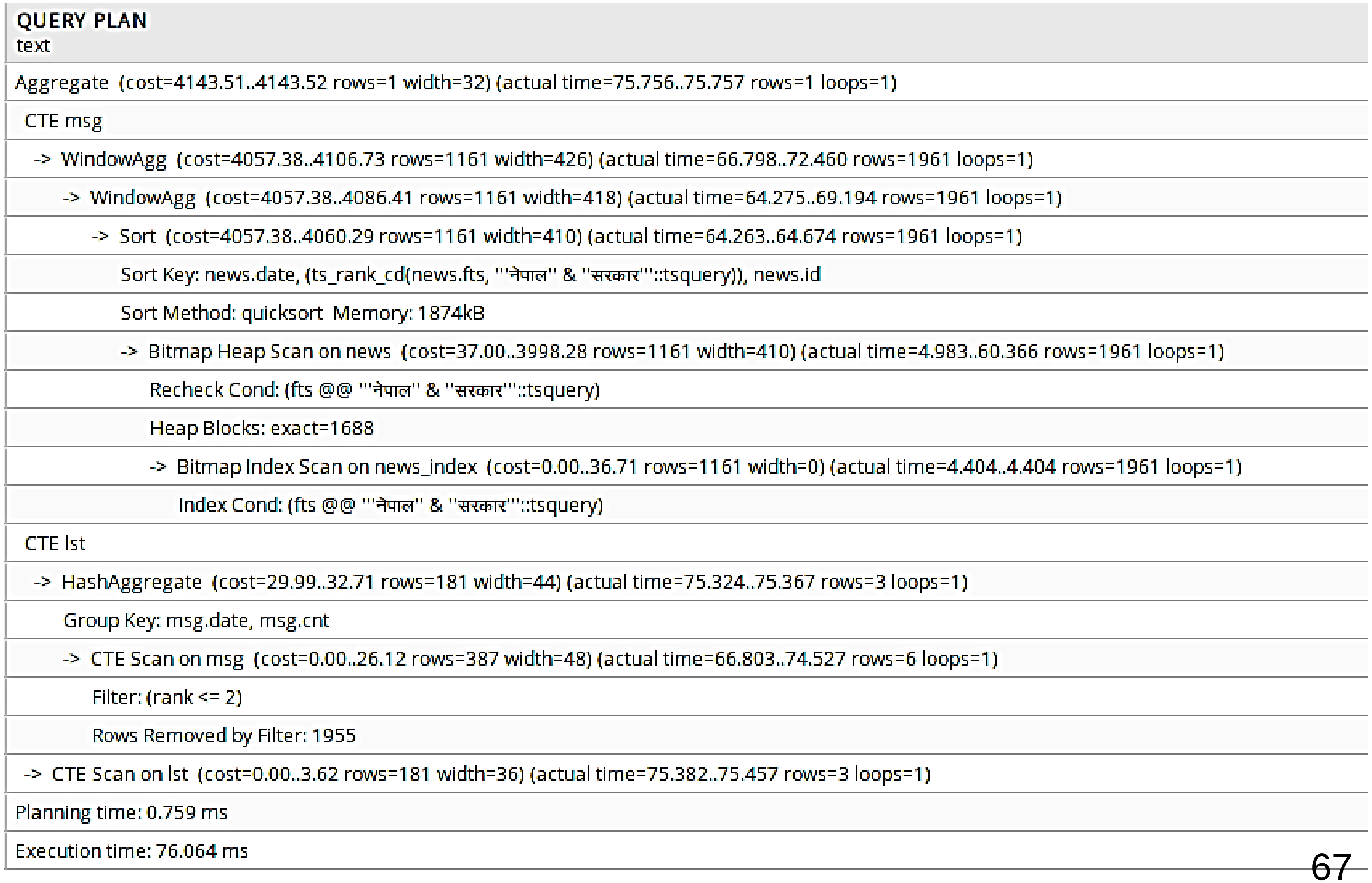
postgres=# CREATE INDEX news_index ON news USING GIN(fts)



```
EXPLAIN ANALYSE
WITH res AS (
    SELECT id, body, date,
        RANK() OVER (
            PARTITION BY date
            ORDER BY ts_rank_cd(fts,  plainto_tsquery('nepali','नेपाल सरकार')), id
        ) rank,
        COUNT(*) OVER (PARTITION BY date) cnt
    FROM news
    WHERE fts @@ plainto_tsquery('nepali','नेपाल सरकार')
),
lst AS (
    SELECT
        date,
        jsonb_build_object(
            'count', cnt,
            'results', jsonb_agg(
                jsonb_build_object(
                    'id', id,
                    'body', body
        ))) AS data
    FROM res
    WHERE rank <= 2
    GROUP by date, cnt
)
SELECT  jsonb_object_agg(date, data)
FROM lst;
```

Nepali NLP Group

Faceted Search SQL Query Plan

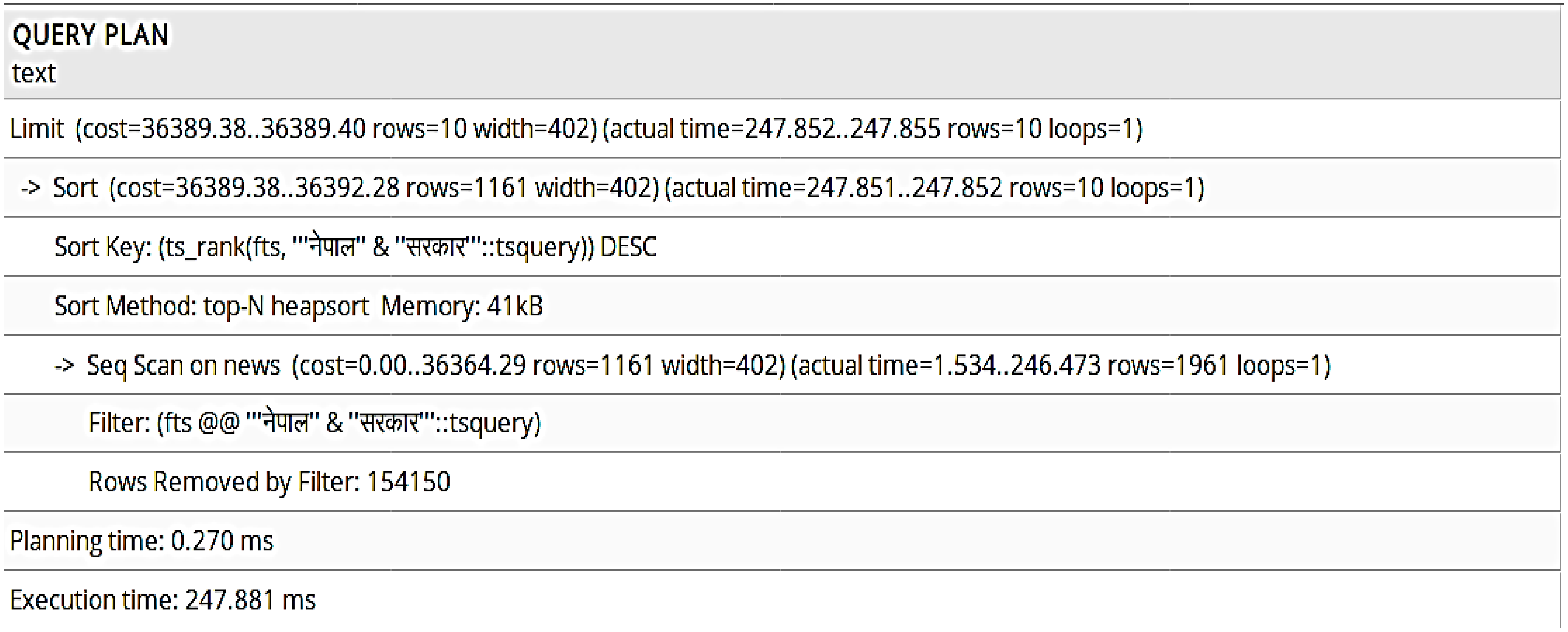


```
QUERY PLAN
text
Aggregate  (cost=4143.51..4143.52 rows=1 width=32) (actual time=75.756..75.757 rows=1 loops=1)
  CTE msg
    ->  WindowAgg  (cost=4057.38..4106.73 rows=1161 width=426) (actual time=66.798..72.460 rows=1961 loops=1)
          ->  WindowAgg  (cost=4057.38..4086.41 rows=1161 width=418) (actual time=64.275..69.194 rows=1961 loops=1)
                ->  Sort  (cost=4057.38..4060.29 rows=1161 width=410) (actual time=64.263..64.674 rows=1961 loops=1)
                      Sort Key: news.date, (ts_rank_cd(news.fts, '''नेपाल'' & ''सरकार'''::tsquery)), news.id
                      Sort Method: quicksort  Memory: 1874kB
                      ->  Bitmap Heap Scan on news  (cost=37.00..3998.28 rows=1161 width=410) (actual time=4.983..60.366 rows=1961 loops=1)
                            Recheck Cond: (fts @@ '''नेपाल'' & ''सरकार'''::tsquery)
                            Heap Blocks: exact=1688
                            ->  Bitmap Index Scan on news_index  (cost=0.00..36.71 rows=1161 width=0) (actual time=4.404..4.404 rows=1961 loops=1)
                                  Index Cond: (fts @@ '''नेपाल'' & ''सरकार'''::tsquery)
  CTE lst
    ->  HashAggregate  (cost=29.99..32.71 rows=181 width=44) (actual time=75.324..75.367 rows=3 loops=1)
          Group Key: msg.date, msg.cnt
          ->  CTE Scan on msg  (cost=0.00..26.12 rows=387 width=48) (actual time=66.803..74.527 rows=6 loops=1)
                Filter: (rank <= 2)
                Rows Removed by Filter: 1955
  ->  CTE Scan on lst  (cost=0.00..3.62 rows=181 width=36) (actual time=75.382..75.457 rows=3 loops=1)
Planning time: 0.759 ms
Execution time: 76.064 ms
```

Nepali NLP Group

FTS without Indexing

```
postgres=# EXPLAIN ANALYSE
SELECT body, ts_rank(fts, plainto_tsquery('nepali', 'नेपाल सरकार' )) AS rank
FROM news WHERE fts @@ plainto_tsquery('nepali','नेपाल सरकार')
ORDER BY rank DESC LIMIT 10;
```

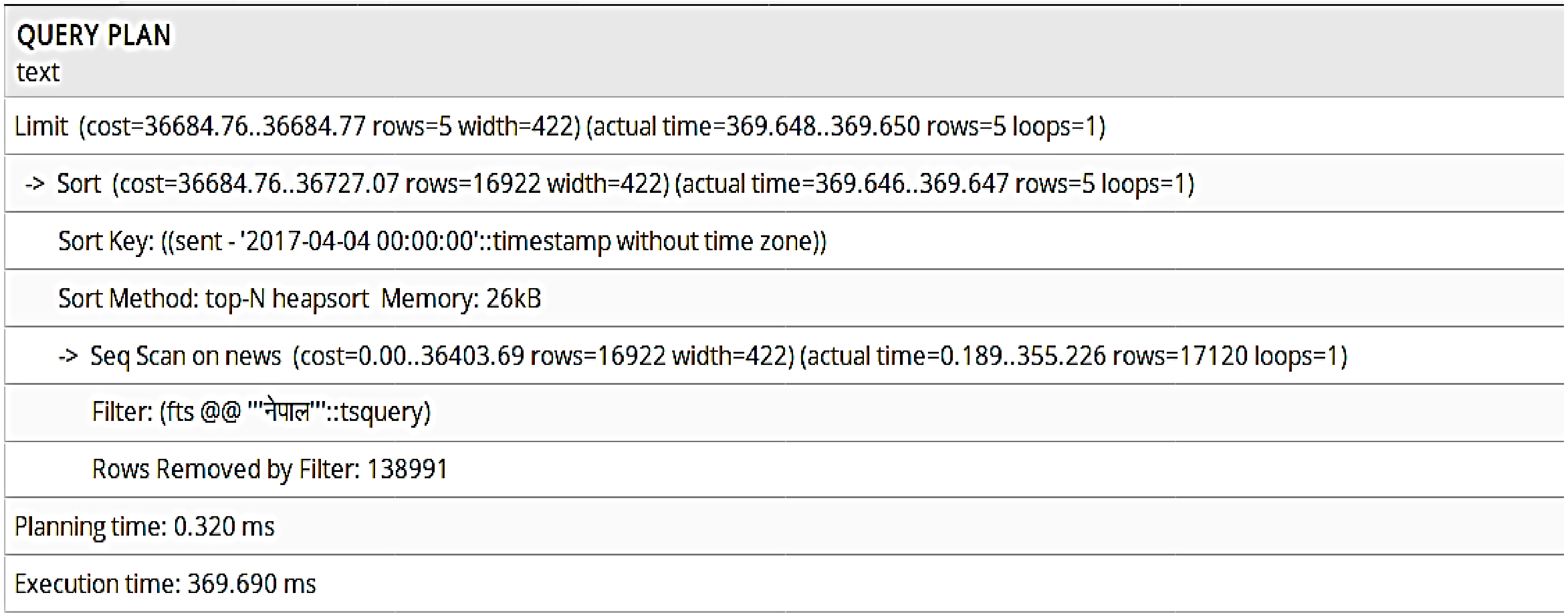


```
QUERY PLAN
text
Limit  (cost=36389.38..36389.40 rows=10 width=402) (actual time=247.852..247.855 rows=10 loops=1)
  ->  Sort  (cost=36389.38..36392.28 rows=1161 width=402) (actual time=247.851..247.852 rows=10 loops=1)
        Sort Key: (ts_rank(fts, '''नेपाल'' & ''सरकार'''::tsquery)) DESC
        Sort Method: top-N heapsort  Memory: 41kB
        ->  Seq Scan on news  (cost=0.00..36364.29 rows=1161 width=402) (actual time=1.534..246.473 rows=1961 loops=1)
              Filter: (fts @@ '''नेपाल'' & ''सरकार'''::tsquery)
              Rows Removed by Filter: 154150
Planning time: 0.270 ms
Execution time: 247.881 ms
```

Nepali NLP Group

FTS: Order by Timestamp

```
postgres=# EXPLAIN ANALYSE
SELECT sent, body FROm news
WHERE fts @@ to_tsquery('nepali', 'नेपाल')
ORDER BY (sent - '2017-04-04'::timestamp) ASC LIMIT 5;
```



```
QUERY PLAN
text
Limit  (cost=36684.76..36684.77 rows=5 width=422) (actual time=369.648..369.650 rows=5 loops=1)
  ->  Sort  (cost=36684.76..36727.07 rows=16922 width=422) (actual time=369.646..369.647 rows=5 loops=1)
        Sort Key: ((sent - '2017-04-04 00:00:00'::timestamp without time zone))
        Sort Method: top-N heapsort  Memory: 26kB
        ->  Seq Scan on news  (cost=0.00..36403.69 rows=16922 width=422) (actual time=0.189..355.226 rows=17120 loops=1)
              Filter: (fts @@ '''नेपाल'''::tsquery)
              Rows Removed by Filter: 138991
Planning time: 0.320 ms
Execution time: 369.690 ms
```

However, FTS alone is slow!!

Indexes

- Lookup table
- Speeds up data retrieval
- Produces the same results as sequence scan with filtering but *faster!*
- Indexes can be:
    - **partial** (where price > 0.0)
    - **functional** (to_tsvector(text))
    - **multicolumn** (timestamp, tsvector)
- However, don’t over-index!
    - Low selectivity
    - Maintenance overhead

Indexes

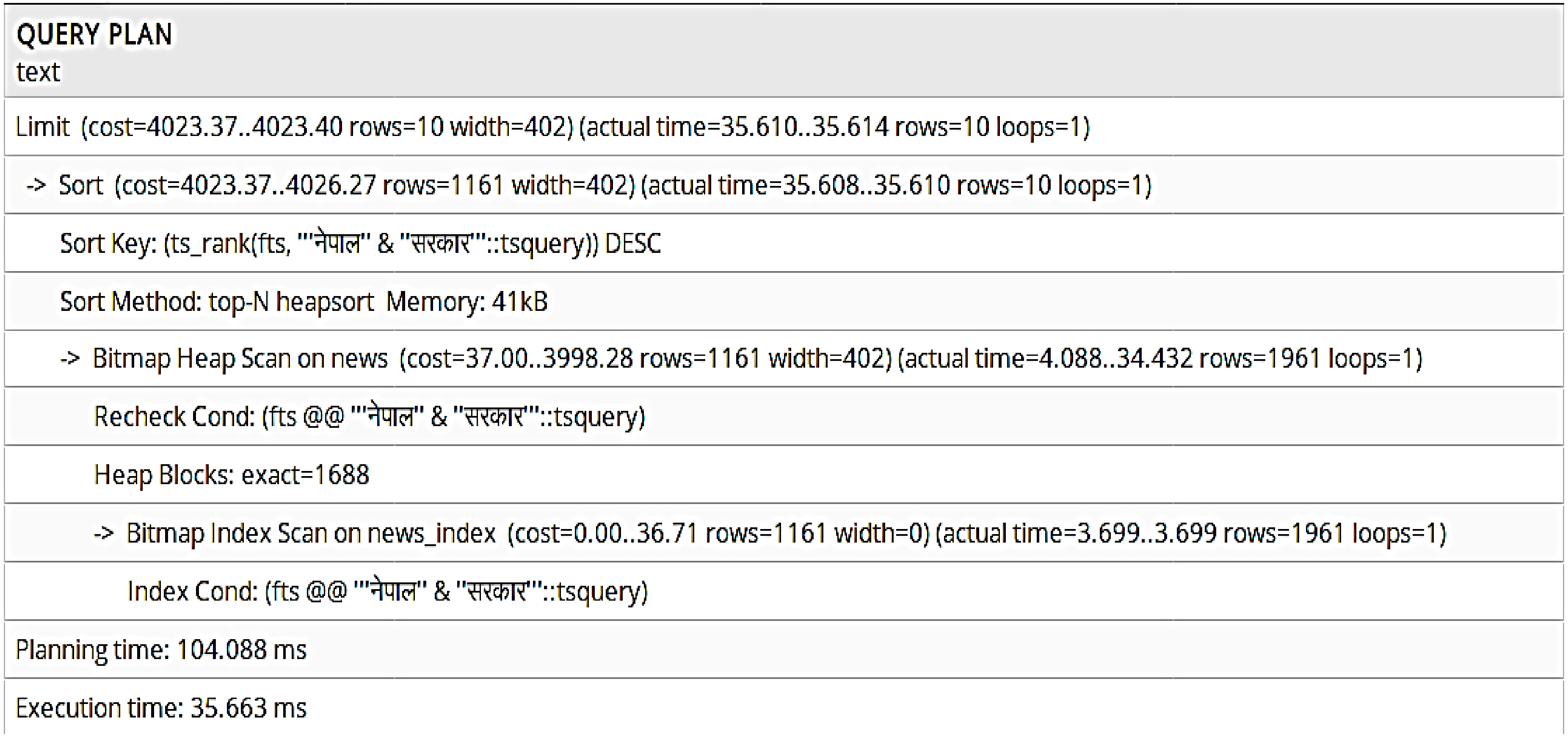
GIN	**RUM**
Perform ranking from heap	Perform ranking from index
Slower ranking	Faster ranking
Considerable overhead of phrase operator	Use position in addinfo, almost no overhead of phrase operator
Slower ordering by timestamp	Faster ordering by timestamp
Faster build and insert time	Slower build and insert time

FTS : Indexing using GIN

```
postgres=# CREATE INDEX news_index ON news USING gin(fts)

postgres=# EXPLAIN ANALYSE
SELECT body, ts_rank(fts, plainto_tsquery('nepali', 'नेपाल सरकार' ))
AS rank
FROM news WHERE fts @@ plainto_tsquery('nepali','नेपाल
सरकार')
ORDER BY rank DESC LIMIT 10;
```

Nepali NLP Group

Indexing using GIN



```
QUERY PLAN
text
Limit  (cost=4023.37..4023.40 rows=10 width=402) (actual time=35.610..35.614 rows=10 loops=1)
  ->  Sort  (cost=4023.37..4026.27 rows=1161 width=402) (actual time=35.608..35.610 rows=10 loops=1)
        Sort Key: (ts_rank(fts, '''नेपाल'' & ''सरकार'''::tsquery)) DESC
        Sort Method: top-N heapsort  Memory: 41kB
        ->  Bitmap Heap Scan on news  (cost=37.00..3998.28 rows=1161 width=402) (actual time=4.088..34.432 rows=1961 loops=1)
              Recheck Cond: (fts @@ '''नेपाल'' & ''सरकार'''::tsquery)
              Heap Blocks: exact=1688
              ->  Bitmap Index Scan on news_index  (cost=0.00..36.71 rows=1161 width=0) (actual time=3.699..3.699 rows=1961 loops=1)
                    Index Cond: (fts @@ '''नेपाल'' & ''सरकार'''::tsquery)
Planning time: 104.088 ms
Execution time: 35.663 ms
```

FTS : Indexing using RUM

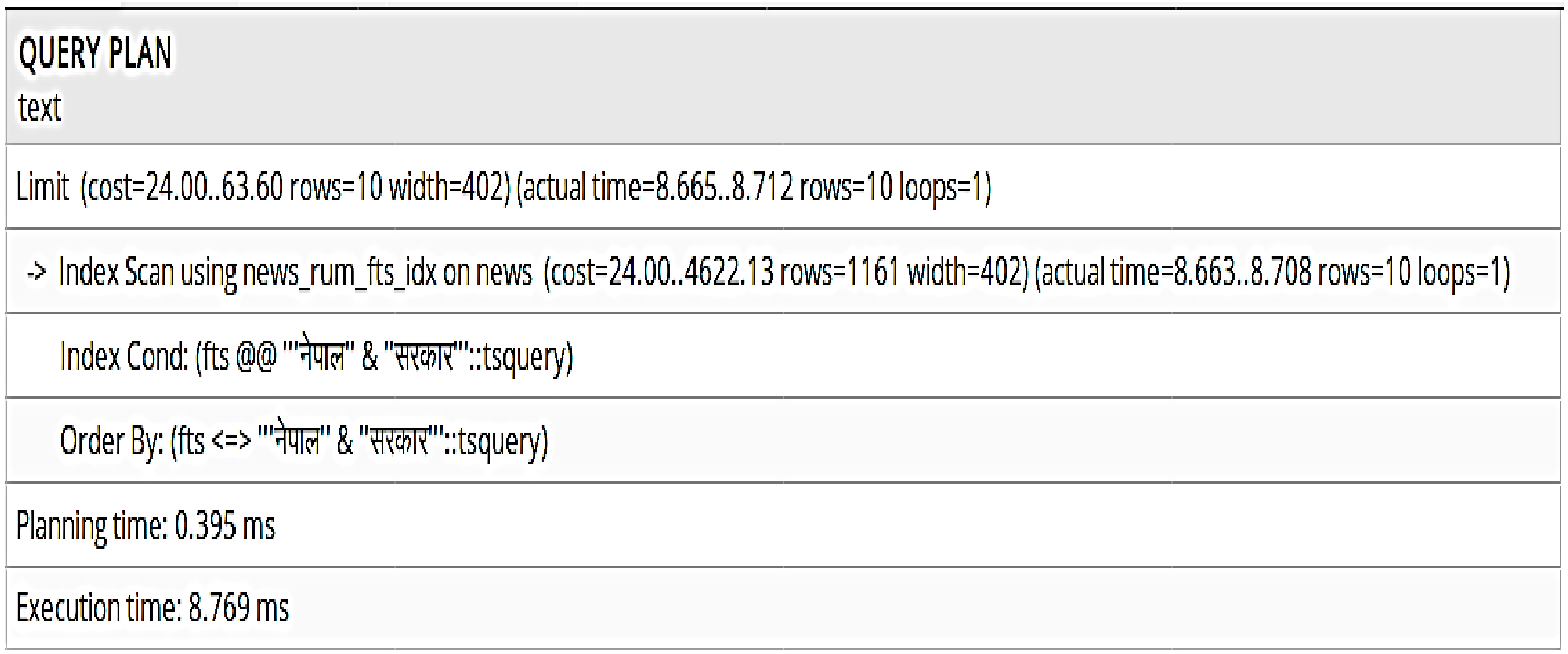
- Use positions to for ranking and ordering result.
- Introduce distance operator <=> tsquery

```
postgres=# CREATE INDEX news_rum_fts_idx ON news
USING rum(fts rum_tsvector_ops)
```

```
postgres=# EXPLAIN ANALYSE
SELECT body FROM news WHERE fts @@
plainto_tsquery('nepali','नेपाल सरकार')
ORDER BY fts <=> plainto_tsquery('nepali','नेपाल सरकार')
LIMIT 10;
```

Nepali NLP Group

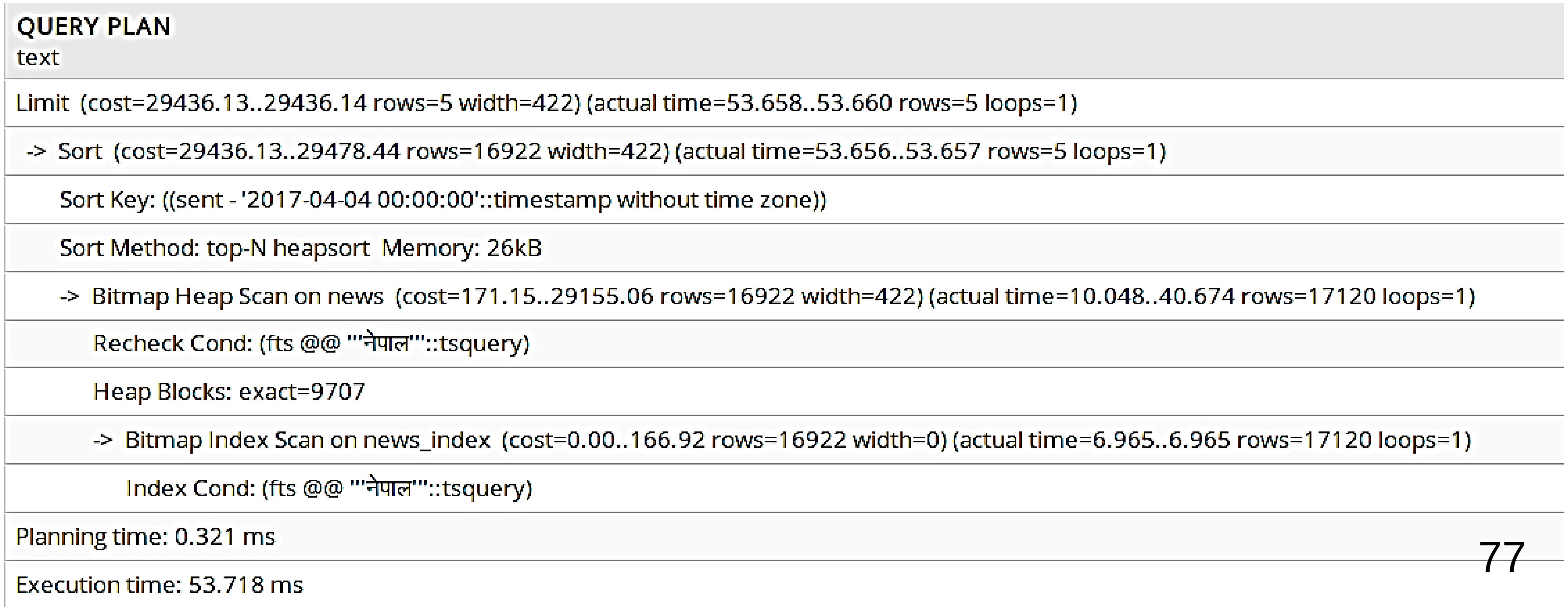
Indexing using RUM



```
QUERY PLAN
text
Limit  (cost=24.00..63.60 rows=10 width=402) (actual time=8.665..8.712 rows=10 loops=1)
  ->  Index Scan using news_rum_fts_idx on news  (cost=24.00..4622.13 rows=1161 width=402) (actual time=8.663..8.708 rows=10 loops=1)
        Index Cond: (fts @@ '''नेपाल'' & ''सरकार'''::tsquery)
        Order By: (fts <=> '''नेपाल'' & ''सरकार'''::tsquery)
Planning time: 0.395 ms
Execution time: 8.769 ms
```

FTS: Order by Timestamp using GIN

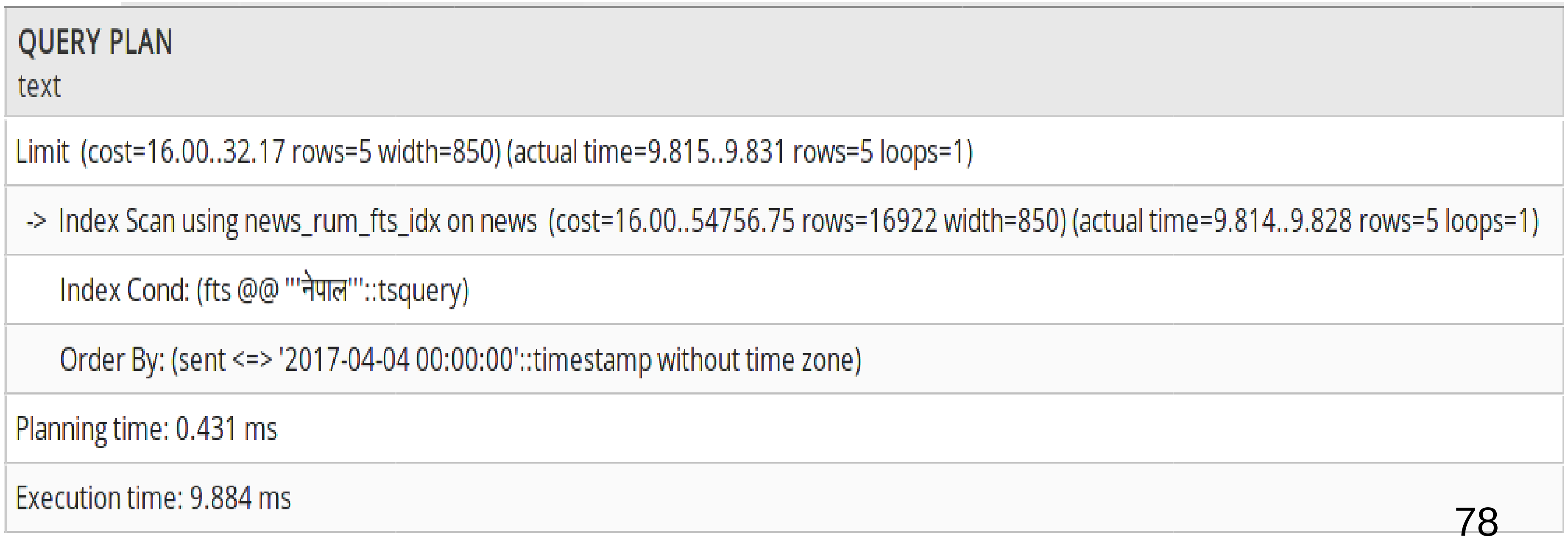
```
postgres=# CREATE INDEX news_index ON news USING gin(fts)

postgres=# EXPLAIN ANALYSE
SELECT sent, body FROm news
WHERE fts @@ to_tsquery('nepali', 'नेपाल')
ORDER BY (sent - '2017-04-04'::timestamp) ASC LIMIT 5;
```



```
QUERY PLAN
text
Limit  (cost=29436.13..29436.14 rows=5 width=422) (actual time=53.658..53.660 rows=5 loops=1)
  ->  Sort  (cost=29436.13..29478.44 rows=16922 width=422) (actual time=53.656..53.657 rows=5 loops=1)
        Sort Key: ((sent - '2017-04-04 00:00:00'::timestamp without time zone))
        Sort Method: top-N heapsort  Memory: 26kB
        ->  Bitmap Heap Scan on news  (cost=171.15..29155.06 rows=16922 width=422) (actual time=10.048..40.674 rows=17120 loops=1)
              Recheck Cond: (fts @@ '''नेपाल'''::tsquery)
              Heap Blocks: exact=9707
              ->  Bitmap Index Scan on news_index  (cost=0.00..166.92 rows=16922 width=0) (actual time=6.965..6.965 rows=17120 loops=1)
                    Index Cond: (fts @@ '''नेपाल'''::tsquery)
Planning time: 0.321 ms
Execution time: 53.718 ms
```

FTS: Order by Timestamp using RUM

```
postgres=# CREATE INDEX news_rum_fts_idx ON news USING rum(fts
rum_tsvector_addon_ops,sent)
WITH (attach='sent', to='fts')

postgres=# EXPLAIN ANALYSE select * FROM news WHERE fts @@
to_tsquery('nepali','नेपाल') order by date <=> '2017-04-04'::timestamp LIMIT 5
```



```
QUERY PLAN
text
Limit  (cost=16.00..32.17 rows=5 width=850) (actual time=9.815..9.831 rows=5 loops=1)
  ->  Index Scan using news_rum_fts_idx on news  (cost=16.00..54756.75 rows=16922 width=850) (actual time=9.814..9.828 rows=5 loops=1)
        Index Cond: (fts @@ '''नेपाल'''::tsquery)
        Order By: (sent <=> '2017-04-04 00:00:00'::timestamp without time zone)
Planning time: 0.431 ms
Execution time: 9.884 ms
```

Phrase Search

- 'A & B'::tsquery = 'B & A'::tsquery
- Phrase search preserves the order of words in a query
- Followed by (<->) operator [Postgres 9.6+]
- ‘A <-> B'::tsquery ≠ ‘B <-> A'::tsquery
- A<n>B: $0 \leq P_B - P_A \leq n$ [n= distance]

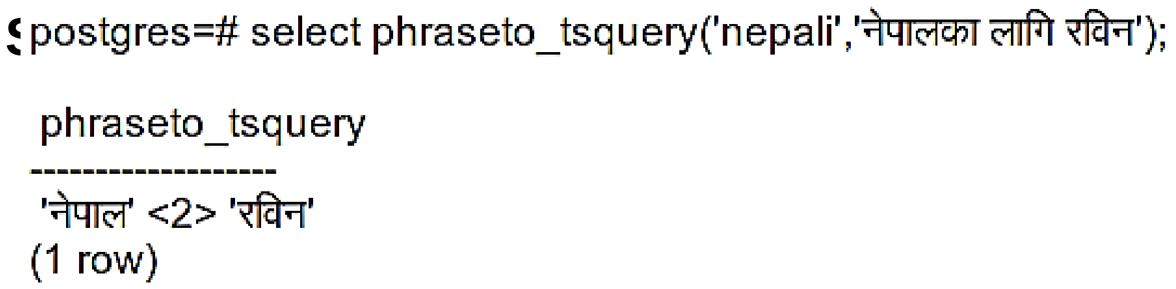
Phrase Search - Properties

- 'A <n> B <m> C' → ‘(A <n> B) <m>C' → matched phrase length ≤ max(n, m)
- ‘A <n> (B <m> C') → matched phrase length ≤ n + m
- ‘A <0> B' matches the word with two different forms (infinitives)
- **Note**
  - 'A C B' matched by ‘(A <2>B) <-> C’
  - Order is preserved for any n, m

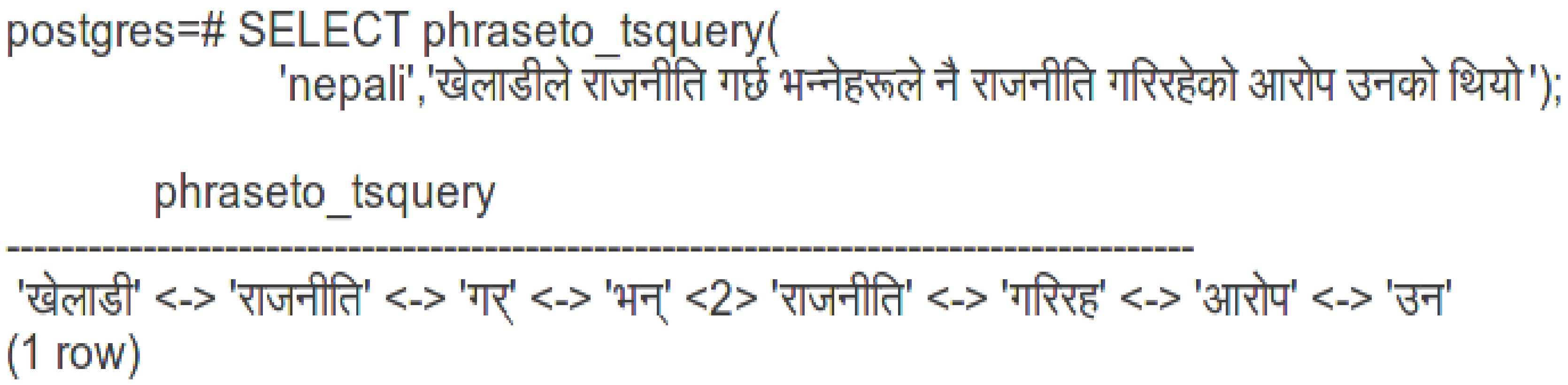
Phrase Search

phraseto_tsquery

Syntax: phraseto_tsquery ([cfg],text)



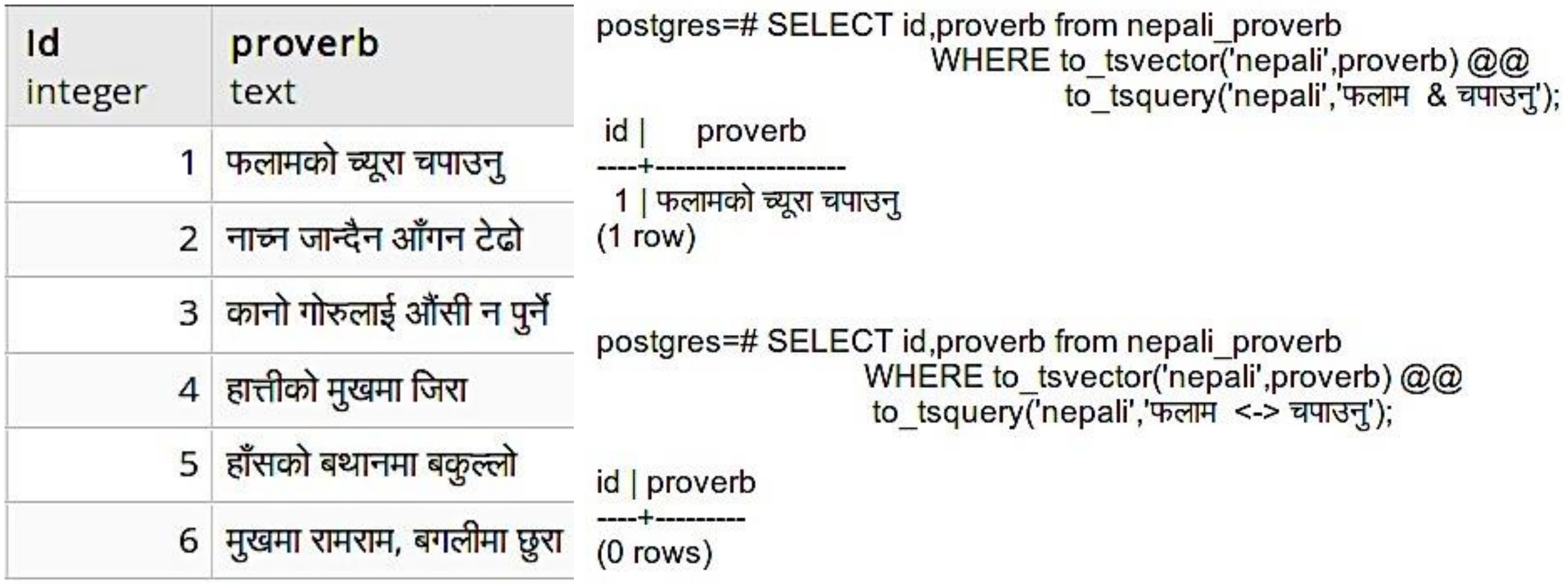
```
postgres=# select phraseto_tsquery('nepali','नेपालका लागि रविन');

 phraseto_tsquery
--------------------
 'नेपाल' <2> 'रविन'
(1 row)
```



```
postgres=# SELECT phraseto_tsquery(
                 'nepali','खेलाडीले राजनीति गर्छ भन्नेहरूले नै राजनीति गरिरहेको आरोप उनको थियो');

        phraseto_tsquery
------------------------------------------------------------------------------------
 'खेलाडी' <-> 'राजनीति' <-> 'गर्' <-> 'भन्' <2> 'राजनीति' <-> 'गरिरह' <-> 'आरोप' <-> 'उन'
(1 row)
```

Nepali NLP Group

Phrase Search Example

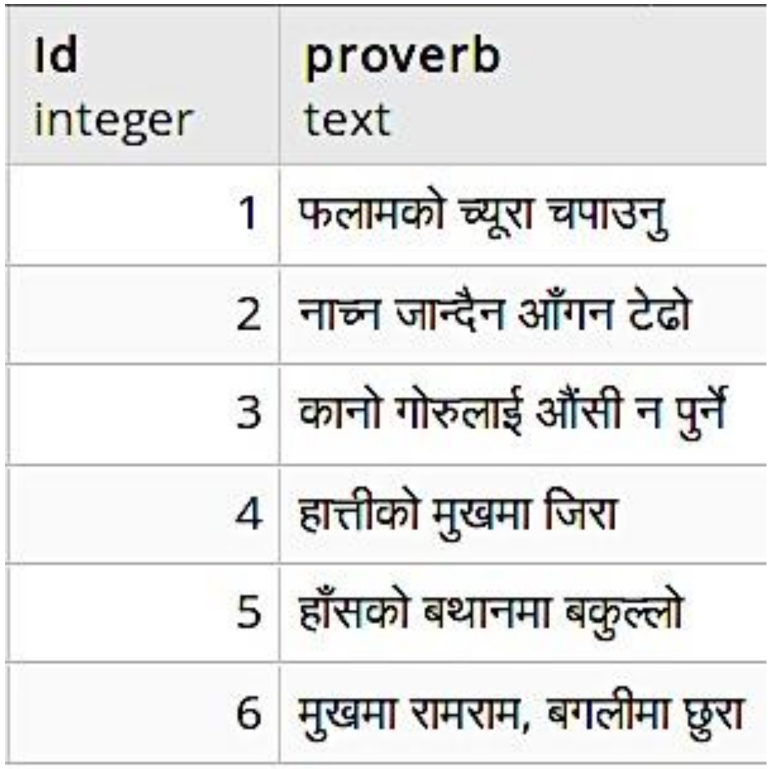


Id integer	proverb text
1	फलामको च्यूरा चपाउनु
2	नाच्न जान्दैन आँगन टेढो
3	कानो गोरुलाई औंसी न पुर्नें
4	हात्तीको मुखमा जिरा
5	हाँसको बथानमा बकुल्लो
6	मुखमा रामराम, बगलीमा छुरा

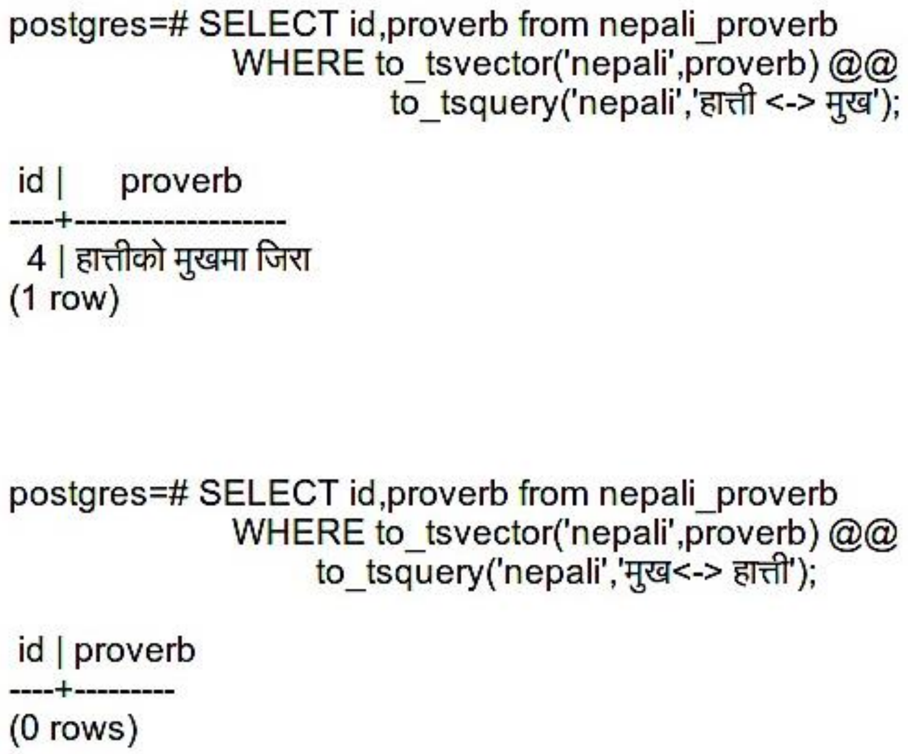
```
postgres=# SELECT id,proverb from nepali_proverb
                                WHERE to_tsvector('nepali',proverb) @@
                                      to_tsquery('nepali','फलाम  & चपाउनु');
 id |     proverb
----+------------------
  1 | फलामको च्यूरा चपाउनु
(1 row)


postgres=# SELECT id,proverb from nepali_proverb
                            WHERE to_tsvector('nepali',proverb) @@
                             to_tsquery('nepali','फलाम  <-> चपाउनु');

id | proverb
----+---------
(0 rows)
```

Nepali NLP Group

Phrase Search Example



Id	proverb
integer	text
1	फलामको च्यूरा चपाउनु
2	नाच्न जान्दैन आँगन टेढो
3	कानो गोरुलाई औँसी न पुर्ने
4	हात्तीको मुखमा जिरा
5	हाँसको बथानमा बकुल्लो
6	मुखमा रामराम, बगलीमा छुरा

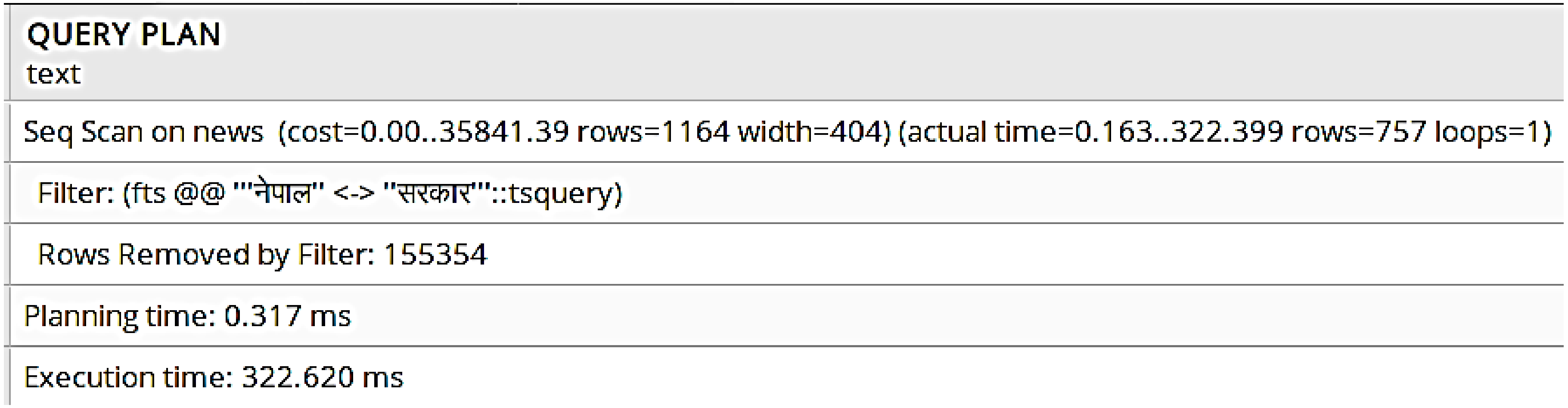


```
postgres=# SELECT id,proverb from nepali_proverb
                    WHERE to_tsvector('nepali',proverb) @@
                          to_tsquery('nepali','हात्ती <-> मुख');

 id |     proverb
----+-------------------
  4 | हात्तीको मुखमा जिरा
(1 row)
```

```
postgres=# SELECT id,proverb from nepali_proverb
                    WHERE to_tsvector('nepali',proverb) @@
                       to_tsquery('nepali','मुख<-> हात्ती');

 id | proverb
----+---------
(0 rows)
```

Phrase Search without Indexing

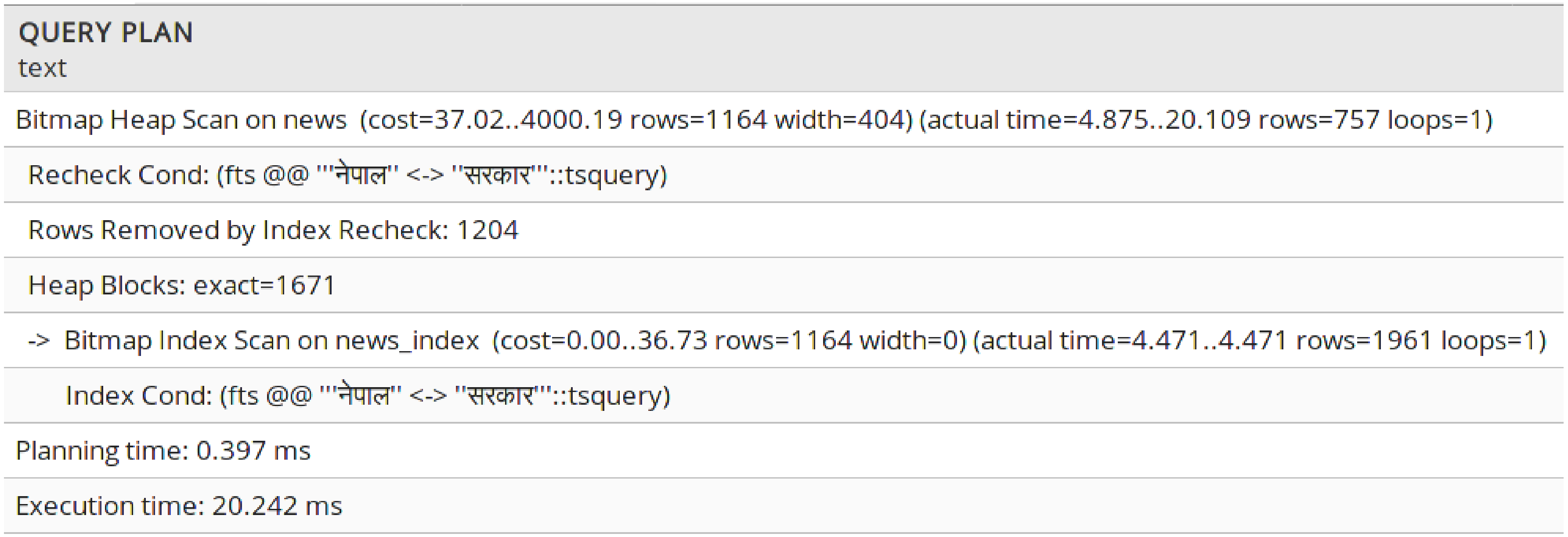
```
postgres=# EXPLAIN ANALYSE SELECT id,body FROM news WHERE fts @@
to_tsquery('nepali','नेपाल <-> सरकार');
```



```
QUERY PLAN
text
Seq Scan on news  (cost=0.00..35841.39 rows=1164 width=404) (actual time=0.163..322.399 rows=757 loops=1)
  Filter: (fts @@ '''नेपाल'' <-> ''सरकार'''::tsquery)
  Rows Removed by Filter: 155354
Planning time: 0.317 ms
Execution time: 322.620 ms
```

Phrase Search with GIN Indexing

```
postgres=# CREATE INDEX news_index ON news USING gin(fts)

postgres=# EXPLAIN ANALYSE SELECT id,body FROM news WHERE fts @@
to_tsquery('nepali','नेपाल <-> सरकार');
```



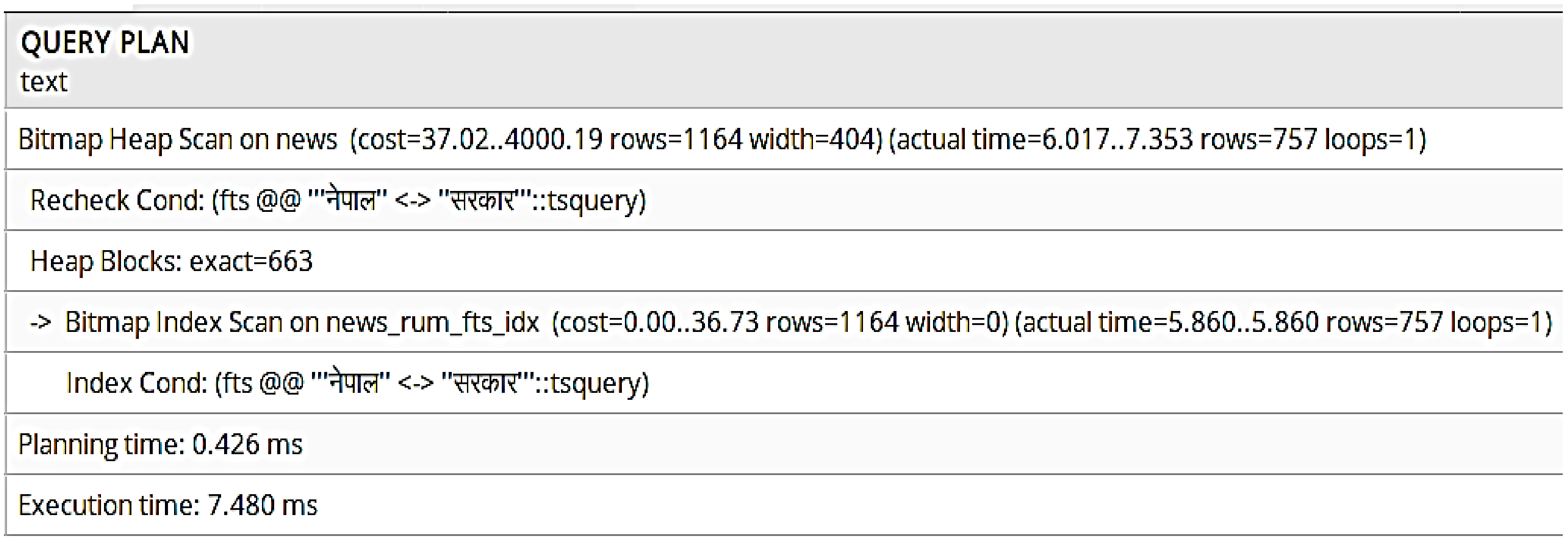
```
QUERY PLAN
text
Bitmap Heap Scan on news  (cost=37.02..4000.19 rows=1164 width=404) (actual time=4.875..20.109 rows=757 loops=1)
  Recheck Cond: (fts @@ '''नेपाल'' <-> ''सरकार'''::tsquery)
  Rows Removed by Index Recheck: 1204
  Heap Blocks: exact=1671
  ->  Bitmap Index Scan on news_index  (cost=0.00..36.73 rows=1164 width=0) (actual time=4.471..4.471 rows=1961 loops=1)
        Index Cond: (fts @@ '''नेपाल'' <-> ''सरकार'''::tsquery)
Planning time: 0.397 ms
Execution time: 20.242 ms
```

Nepali NLP Group

Phrase Search with RUM Indexing

```
postgres=# CREATE INDEX news_rum_fts_idx ON news USING rum(fts)
```

```
postgres=# EXPLAIN ANALYSE SELECT id,body from news WHERE fts @@
to_tsquery('nepali','नेपाल <-> सरकार');
```



```
QUERY PLAN
text
Bitmap Heap Scan on news  (cost=37.02..4000.19 rows=1164 width=404) (actual time=6.017..7.353 rows=757 loops=1)
  Recheck Cond: (fts @@ '''नेपाल'' <-> ''सरकार'''::tsquery)
  Heap Blocks: exact=663
  ->  Bitmap Index Scan on news_rum_fts_idx  (cost=0.00..36.73 rows=1164 width=0) (actual time=5.860..5.860 rows=757 loops=1)
        Index Cond: (fts @@ '''नेपाल'' <-> ''सरकार'''::tsquery)
Planning time: 0.426 ms
Execution time: 7.480 ms
```

To Do…

- New dictionaries for Nepali can be added for Nepali support for FTS
- The existing Nepali stemmer can be improved to handle cases of *i-ending verbs*.

Nepali NLP Group

Summary

- It is possible with minimal effort to develop Nepali search engines.
- Postgres is free, and next version will have Nepali configuration.
- Hence, any Nepali project will be powered by fts.

Acknowledgement

Oleg Bartunov, *Moscow University, Postgres Professional*
- Mentored the project and the patch

Arthur Zakirov, *Postgres Professional*
- Helped with snowball stemmer

Shreeya Singh Dhakal, *Nepali NLP Group*
- Helped with Nepali stemmer and dataset

References

- “Full-Text Search”, https://www.postgresql.org/docs/10/static/textsearch.html
- Oleg Bartunov, Teodor Sigaev, “Full-Text Search in PostgreSQL by authors”, http://www.sai.msu.su/~megera/postgres/talks/fts_postgres_by_authors_2.pdf
- Oleg Bartunov, Teodor Sigaev, “Some recent advances in full-text search”, http://www.sai.msu.su/~megera/postgres/talks/2009.pdf
- Oleg Bartunov, “Create Index Using RUM”, http://www.sai.msu.su/~megera/postgres/talks/pgopen-2016-rum.pdf
- Aleksandr Parfenov, Arthur Zakirov, “Flexible Full Text Search”, https://www.postgresql.eu/events/pgconfeu2017/sessions/session/1595/slides/43/pgconfeu-2017-fts.pdf
- Oleg Bartunov, “Full-Text Search of JSON[b] data in PostgreSQL”, https://obartunov.livejournal.com/tag/fts
- Alexander Korotkov, “Faceted Search in the Single PostgreSQL Query”, http://akorotkov.github.io/blog/2016/06/17/faceted-search/
- “Snowball”, https://snowballstem.org/
- Ingroj Shrestha, Shreeya Singh Dhakal, “A New Stemmer for Nepali Language”, https://ieeexplore.ieee.org/document/7749008/
- RUM Access Method, https://github.com/postgrespro/rum
- apod_fts, https://github.com/postgrespro/apod_fts
- Hunspell dictionaries, https://github.com/postgrespro/hunspell_dicts
- “Nepali Snowball patch”, https://www.postgresql.org/message-id/attachment/58772/nepali_snowball.patch

ANY QUERIES ?

THANK YOU