PGroonga 2

Make PostgreSQL rich full text search system backend!

Kouhei Sutou

ClearCode Inc.

PGConf.ASIA 2017
2017-12-05

Targets
対象者

- Want to implement full text search with PostgreSQL
  PostgreSQLで全文検索したい
- Not good at full text search
  全文検索はよく知らない
- PGroonga 1.0.0 users
  PGroonga 1.0.0は使ったことがある

Abbreviations
略語

- PG: PostgreSQL
  ポスグレ: PostgreSQL
- FTS: Full text search
  FTS: 全文検索

FTS system: Targets
全文検索システム：対象

Many tests
大量のテキスト

- e.g.: Text data in office docs in file servers
  例：ファイルサーバー内のオフィス文書内のテキスト
- e.g.: Item descriptions, chat logs, Wiki data, ...
  例：商品説明やチャットログ、Wikiのデータなど

FTS system: Goal
全文検索システム：目的

Provide
needed info
when you need

必要な情報を必要なときに提供すること

Provide needed info
必要な情報を提供

-  Not found
  探している情報が見つからない
-  Found
  探している情報が見つかる
-  Found **unconscious needed info too!**
  意識していなかったけど実は欲しかった情報も見つかる！

When you need
必要なときに活用

- 😟 Need many times to find
  なかなか見つからない
- 😃 Find in no time
  すぐに見つかる
- 😆 Already found
  すでに見つかっていた
  - e.g.: Recommendation
    例：レコメンデーション

How to impl.: Options
実装方法：選択肢

- Use FTS server
  全文検索サーバーを使う
- Use PostgreSQL
  PostgreSQLを使う

FTS server: Pros
全文検索サーバー案：メリット

- Provides all basic features
  必要な機能が揃っている
- Provides advanced features
  ＋αの機能もある
- Fast
  速い

FTS server: Cons1
全文検索サーバー案：デメリット1

- Large implementation cost
  実装コスト大
  - Learn how to use from scratch
    使い方を1から学ぶ必要がある
  - How to implement data sync?
    マスターデータの同期はどうする？

FTS server: Cons2
全文検索サーバー案：デメリット2

- Large maintenance cost
  メンテナンスコスト大
  - Learn how to operate from scratch
    運用方法を1から学ぶ必要がある

PostgreSQL: Pros1
PostgreSQL案：メリット1

- Less implementation cost
  実装コスト小
  - Less things to be learned
    新しく覚えることが少ない
  - Can manage data at the same place
    データの一元管理

PostgreSQL: Pros2
PostgreSQL案：メリット2

- Less operation cost
  メンテナンスコスト小
  - The current operation knowledge is reusable
    既存の運用ノウハウを使える

PostgreSQL: Cons
PostgreSQL案：デメリット

- Built-in features aren't enough
  組込機能では機能不足
- SQL limits efficiency
  SQLの表現力不足
  - e.g.: SQL needs multiple queries for a process that can be done by 1 query by FTS server
    例：全文検索サーバーなら1クエリーで実現できる処理にSQLだと複数クエリー必要なことがある

The 3rd option
第3の選択肢

- Use FTS engine via PostgreSQL (SQL)
  PostgreSQL経由（SQL）で全文検索エンジンを使う

Pros
メリット

- Fast and rich features
  高速で豊富な機能
- Less implementation cost
  実装コスト小
- Less operation cost
  メンテナンスコスト小

Cons
デメリット

- Need PostgreSQL extension
  PostgreSQLに拡張機能が必要
  - Not available on DBaaS
    DBaaSで使えない

Option: No FTS knowledge
オススメの選択肢：全文検索の知識ナシ

- Need only simple features
  まだ単純な機能で十分
  - Less data: LIKE with PostgreSQL
    データ少：PostgreSQLでLIKE
- Need up-to-date FTS features
  いまどきの全文検索機能が必要
  - FTS engine via PostgreSQL
    PostgreSQL経由で全文検索エンジン

Option: With FTS knowledge
オススメの選択肢：全文検索の知識アリ

- Need tuned FTS feature
  カリカリにチューニングしたい
  - PostgreSQL + FTS server
    PostgreSQL＋全文検索サーバー
- Others
  それ以外
  - FTS engine via PostgreSQL
    PostgreSQL経由で全文検索エンジン

Described option
説明する選択肢

FTS engine via PostgreSQL

PostgreSQL経由で全文検索エンジン

FTS engine: Groonga
全文検索エンジン：Groonga（ぐるんが）

- Embeddable FTS engine
  組込可能な全文検索エンジン
  - PGroonga: Groonga in PostgreSQL
    PGroonga：PostgreSQLに組込
- Usable as FTS server
  全文検索サーバーとして単独でも使用可能
  - PostgreSQL + FTS server architecture is also available
    PostgreSQL＋全文検索サーバー構成もできる

Groonga's hobby: data update
Groongaの得意な事：データの追加・更新

- Make fresh data searchable!
  新鮮な情報をすぐ検索可能！
  - Batch update is needless
    バッチで更新しなくてもよい
  - Can use as chat backend
    チャットくらいの頻度でもOK
    e.g.: Zulip uses PGroonga
    例：ZulipはPGroongaを採用

Groonga's hobby: data update
Groongaの得意な事：データの追加・更新

- Keep search performance while updating!
  更新中も検索性能が落ちない！
  - Updatable when there are many search users
    利用ユーザーが多い時でも更新可能

PGroonga
PGroonga（ぴーじーるんが）

- PostgreSQL index
  PostgreSQLのインデックス
  - Alternative of GIN, RUM, ...
    GIN・RUMなどと同じレイヤー
- Usage
  使用方法
  - CREATE INDEX ...
    USING PGroonga ...

PostgreSQL and FTS
PostgreSQLと全文検索

- LIKE: Built-in (組込機能)
- textsearch: Built-in (組込機能)
- pg_trgm: Contrib (標準添付)
  - Bundled in the archive
    アーカイブには含まれている
  - Need to install separately
    別途インストールすれば使える

LIKE and performance
LIKEと速度

- Small data
  少ないデータ
  - Enough performance
    十分実用的
- Not small data
  少なくないデータ
  - Need to tune
    性能問題アリ

LIKE and FTS system
LIKEと全文検索システム

👍Enough performance
in most case
速度が実用的なことも多い

- Data are small in many case
  少ないデータなら

LIKE and FTS system
LIKEと全文検索システム

👎Unable to sort
それっぽい順のソート不可

- Sort is important in FTS
  全文検索ではソート順が重要
- Users check only the first N entries
  ユーザーは先頭N件しか見ない

textsearch

- 👍Fast search by index
  インデックスを作るので速い
- Need module for each lang
  言語毎にモジュールが必要
  - 👍Modules for English, French, ... are built-in
    英語やフランス語などは組込
  - 👎Modules for languages in Asia aren't maintained
    アジア圏の言語用のモジュールはメンテされていない

pg_trgm

- 👍Fast search by index
  インデックスを作るので速い
- 👎Asian languages aren't enough supported
  アジア圏の言語のサポートは十分ではない
- 👎Unable to sort
  それっぽい順のソート不可

RUM

- RUM = GIN + position
  RUMは位置情報付きのGIN
  - https://github.com/postgrespro/rum
- pg_trgm/pg_bigm are slow for much matches case
  pg_trgmとpg_bigmはマッチ数が多いと遅くなる
  - RUM will solve it
    GINの代わりにRUMを使うことで解決できるかも！

PGroonga

- 👍Fast search by index
  インデックスを作るので速い
- 👍Sortable
  それっぽい順のソート可
- 👍Support all languages
  全言語対応
- 👎Need to install separately
  別途インストールする必要アリ

FTS system with PostgreSQL
PostgreSQLで全文検索システム

- PGroonga is the best!💯
  PGroongaがベスト！
- PGroonga
  - Fast（高速）
  - Support all langs（全言語対応）
  - Sortable（それっぽい順でソート可）

FTS system: Basic features
全文検索システム：基本機能

- Fast FTS + sort
  高速全文検索＋ソート
- Show texts around keyword
  キーワード周辺テキスト表示
- Highlight keyword
  検索キーワードハイライト

FTS system: Adv. features
全文検索システム：高度な機能

- Auto complete
  オートコンプリート
- Similar search
  類似文書検索
- Synonym expansion
  同義語展開

PGroonga 1.0.0

↓ are only supported

以下の機能のみ対応

- Fast FTS + sort
  高速全文検索＋ソート
- Show texts around keyword
  キーワード周辺テキスト表示

PGroonga 2

All features are supported!

全機能対応！

PGroonga 1.0.0 → 2

-  Many new features
  たくさんの新機能
-  Improve performance
  性能改善
-  API is changed
  APIが変わった

API change
API変更

Operator is changed
演算子変更

```
@@ → &@~
%% → &@
...
```

API change
API変更

pgroonga schema is deprecated

pgroongaスキーマを非推奨に

```
pgroonga.score → pgroonga_score
pgroonga.flush → pgroonga_flush
...
```

App for PGroonga 1.0.0
PGroonga 1.0.0用アプリ

- Broken with PGroonga 2?
  PGroonga 2では動かない？
  - No! Work without any changes!
    何も変更しなくても動くよ！

Great! But why?
いいじゃん！でもなんで動くの？
↓
"Painless upgrade" technique

Painless upgrade

- PGroonga 2 provides both 1 API and 2 API
  PGroonga 2は1用のAPIも2用のAPIも両方提供
  - Can use PGroonga 2 with 1 API
    PGroonga 1のAPIでPGroonga 2を使える

Painless upgrade

- The last PGroonga 1.X provides both 1 API and partially 2 API
  PGroonga 1系の最終版は1用のAPIも2用のAPIの一部も提供
  - Can use PGroonga 1 with 2 API
    PGroonga 2のAPIでPGroonga 1を使える

Painless upgrade

- PGroonga 2 keeps 1 API
  PGroonga 2の間は1のAPIを維持
- PGroonga 3 will drop 1 API
  PGroonga 3で1のAPIを削除予定
  - Just need to upgrade API until 3
    PGroonga 3までにAPIをアップグレードすればよい

Painless upgrade

- App for PGroonga 1.0.0 doesn't work with PGroonga 2
  PGroonga 1.0.0用のアプリがPGroonga 2で動かない
  - It's a bug. Please report it!
    バグなので報告してね！

FTS system: Basic features
全文検索システム：基本機能

- Fast FTS + sort
  高速全文検索＋ソート
- Show texts around keyword
  キーワード周辺テキスト表示
- Highlight keyword
  検索キーワードハイライト

Fast FTS + sort
高速全文検索＋ソート

PHP document search

php 送信

Full text search

php .ini ディレクティブのリスト 1010

php .ini ディレクティブのリスト

Highlight keyword

以下のリストには、PHP の設定を行うための php .ini ディレクティブが含まれます。

"変更の可否" は

Texts around keyword

"1"
PHP_INI_PERDIR
PHP 4.0.0 で PHP_INI_ALL。PHP 5.4.0 で削除。
allow_url_fopen
"1"
PHP_INI_SYSTEM
PHP <= 4.3.4 で PHP_INI_ALL。
a

"0"
PHP_INI_SYSTEM
PHP 5.2.0 以降で使用可能

Table definition

```
CREATE TABLE entries (
  -- Need primary key
  -- It's needed for sort
  id integer PRIMARY KEY,
  title text,
  content text
);
```

Index definition

```
-- For FTS.
-- The default is good enough!
CREATE INDEX entries_full_text_search
  ON entries
  -- "USING PGroonga" is important!
  -- Primary key is for sort!
  USING PGroonga (id, title, content);
```

Insert data

```
-- Normal INSERT.
INSERT INTO entries
  VALUES (1,
          'Fast FTS with Groonga!',
          'Fast FTS is needed!');
```

FTS

```
SELECT title FROM entries
  WHERE
-- &@~ is for FTS
-- AND search with "search" and "fast"
    title &@~ 'search fast' OR
    content &@~ 'search fast';
```

FTS: LIKE

```
SELECT title FROM entries
  WHERE
-- Index search for LIKE is supported
-- = Improve app perf without any changes
-- NOTE: &@~ is faster than LIKE
    title LIKE '%search%' OR
    content LIKE '%search%';
```

Sort

```
SELECT
    title,
    -- pgroonga_score(TABLE_NAME) returns
    -- precision as number
    pgroonga_score(entries) AS score
  FROM entries
  WHERE -- ...
  -- Sort by precision
  ORDER BY score DESC LIMIT 10;
```

Highlight keyword
キーワードハイライト

PHP document search

Full text search

php 送信

php .ini ディレクティブのリスト 1010

php .ini ディレクティブのリスト Highlight keyword

以下のリストには、PHP の設定を行うための php .ini ディレクティブが含まれます。

"変更の可否" は Texts around keyword

"1"
PHP_INI_PERDIR
PHP 4.0.0 で PHP_INI_ALL。PHP 5.4.0 で削除。
allow_url_fopen
"1"
PHP_INI_SYSTEM
PHP <= 4.3.4 で PHP_INI_ALL。
a

"0"
PHP_INI_SYSTEM
PHP 5.2.0 以降で使用可能。

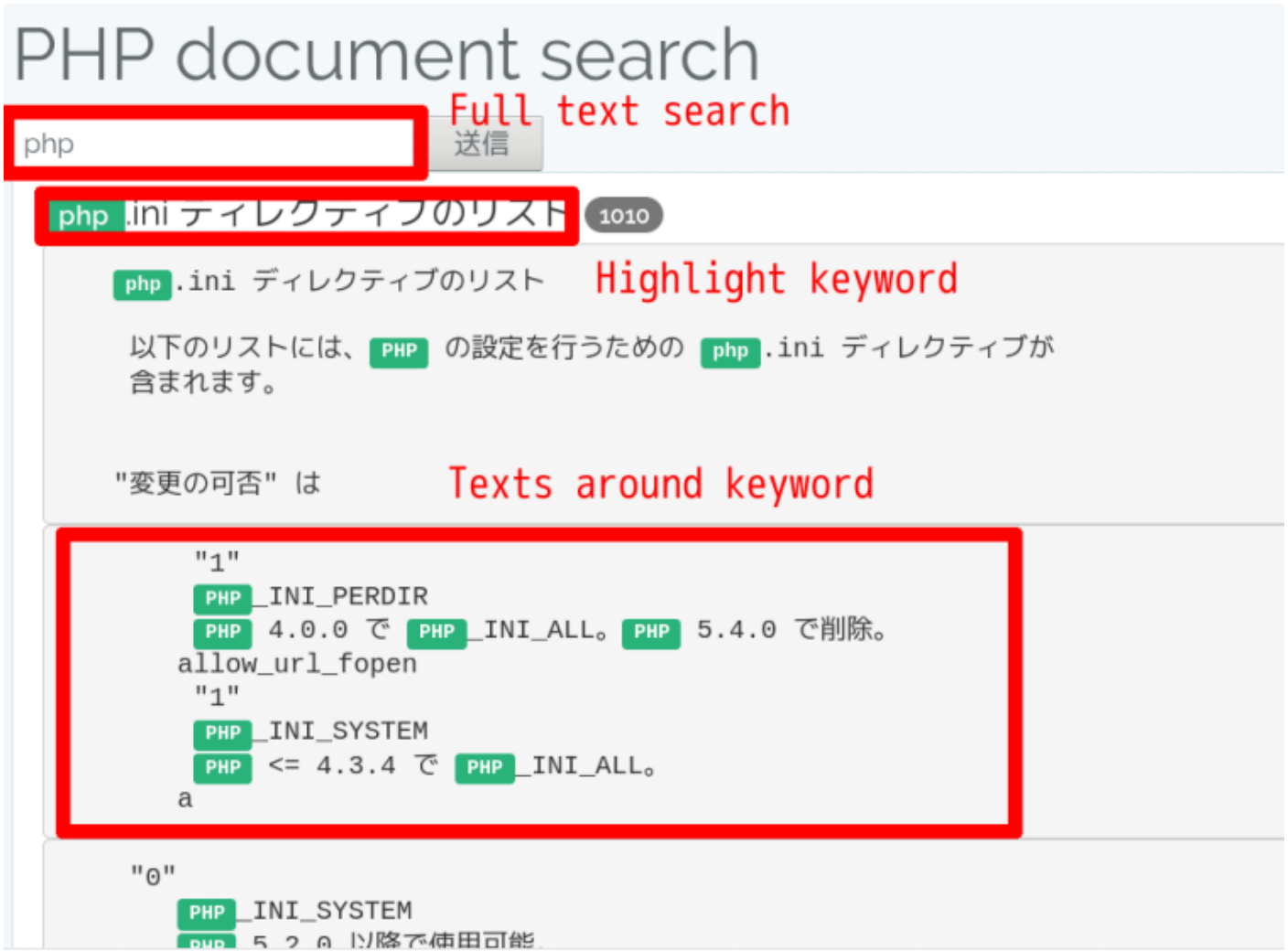
Hightlight for HTML

```
SELECT
  pgroonga_highlight_html(
    title,
    -- Extract keywords from query
    pgroonga_query_extract_keywords('search fast'))
  FROM entries
  WHERE title &@~ 'search fast' OR
        content &@~ 'search fast';
```

Highlight for HTML: Example

```
Fast search with <Groonga>!
↓
<span class="keyword">Fast</span>
↑↓ Keywords are marked up with "class"
<span class="keyword">search</span> !
with &lt;Groonga&gt;! ← Escape tag
```

Texts around keyword
キーワード周辺テキスト



Texts around keyword for HTML

```
SELECT
  pgroonga_snippet_html(
    content,
    -- Extract keywords from query
    pgroonga_query_extract_keywords('search fast'))
  FROM entries
  WHERE title &@~ 'search fast' OR
        content &@~ 'search fast';
```

Example

```
...fast search with <Groonga>!...
↓
ARRAY[
  ↓ First
  '<span class="keyword">fast</span>
  ↑↓ Keywords are marked up with "class"
  <span class="keyword">search/span> !
  with &lt;Groonga&gt;!', ← Escape tag
  '...' ← Second
]
```

FTS system: Adv. features
全文検索システム：高度な機能

- Auto complete
  オートコンプリート
- Similar search
  類似文書検索
- Synonym expansion
  同義語展開

Auto complete
オートコンプリート

PHP document search

seiki 送信

ereg正規表現
form正規化方式
getPregFlags正規表現フラグ
PCRE正規表現
PCRE正規表現処理
POSIX正規表現
POSIX正規表現拡張モジュール
POSIX正規表現関数
POSIX正規表現関数POSIX正規表現関数参考警告
realpath正規化

の php.ini ディレクティブが

PHP 4.0.0 で PHP_INI_ALL。PHP 5.4.0 で削除。
allow_url_fopen
"1"
PHP_INI_SYSTEM
PHP <= 4.3.4 で PHP_INI_ALL。
a

Auto complete: Preparation
オートコンプリート：準備

- Master table
  マスターテーブル
  - Candidate
    候補：（例：牛乳）
  - Readings in Katakana (Only for Japanese)
    ヨミ（日本語の場合。カタカナ。複数登録可。）
    - 例：ギュウニュウ・ミルク

Auto complete: Implementation
オートコンプリート：実装方法

- OR search with ...
  - Prefix search against readings (Only for Japanese)
    ヨミを前方一致検索（日本語の場合。）
  - Loose FTS against candidate
    候補をゆるく全文検索
- Sort by candidate
  候補でソート

https://pgroonga.github.io/how-to/auto-complete.html

Table definition

```
CREATE TABLE terms (
  term text, -- Candidate
  readings text[], -- Readings
);
```

Data example

```
INSERT INTO terms VALUES (
  'milk', -- Candidate
  ARRAY[
    -- Reading in Katakana
    'ギュウニュウ', -- "milk" in Japanese
    -- Multiple readings
    'ミルク'          -- "milk" in Katakana
  ]
);
```

Data management
データ管理

- Easy to maintain because it's a normal table
  普通のテーブルなので管理が楽
  - Easy to insert/delete/update
    追加・削除・更新が楽
  - Normal backup and replication
    ダンプ・リストアもレプリケーションもいつも通り

Index for prefix search
前方一致用インデックス

```
CREATE INDEX prefix_search ON terms
  USING PGroonga
  -- ...text_array_term_search...
  (readings
   pgroonga_text_array_term_search_ops_v2);
```

Index for loose FTS
緩い全文検索用インデックス

```
CREATE INDEX loose_search ON terms
  USING PGroonga (term)
  -- Tokenizer for loose full text search
  WITH (tokenizer='TokenBigramSplitSymbolAlphaDigit');
```

How to search
検索方法

```
SELECT term FROM terms
-- Prefix search against readings
  WHERE readings &^~ '${INPUT}' OR
-- Loose full text search
        term &@ '${INPUT}'
  ORDER BY term LIMIT 10; -- Sort
```

Search example: Candidate
検索例：候補

```
-- User inputs "il"
SELECT term FROM terms
-- Prefix search against readings
  WHERE readings &^~ 'il' OR
-- Loose full text search (Hit)
        term &@ 'il'
  ORDER BY term LIMIT 10; -- Sort
```

Search example: Katakana
検索例：カタカナ

```
-- User inputs "ギュウ"
SELECT term FROM terms
-- Prefix search against readings (Hit)
  WHERE readings &^~ 'ギュウ' OR
-- Loose full text search
        term &@ 'ギュウ'
  ORDER BY term LIMIT 10; -- Sort
```

Search example: Hiragana
検索例：ひらがな

```
-- User inputs "ぎゅう"
SELECT term FROM terms
-- Prefix search against readings (Hit)
  WHERE readings &^~ 'ぎゅう' OR
-- Loose full text search
        term &@ 'ぎゅう'
  ORDER BY term LIMIT 10; -- Sort
```

Search example: Romaji
検索例：ローマ字

```
-- User inputs "gyu"
SELECT term FROM terms
-- Prefix search against readings (Hit)
  WHERE readings &^~ 'gyu' OR
-- Loose full text search
        term &@ 'gyu'
  ORDER BY term LIMIT 10; -- Sort
```

Synonym expansion
同義語展開

- Synonym
  同義語
  - Same mean but different notation
    同じ意味だが表記が異なる語
  - e.g.: "PostgreSQL" and "PG"
    例：「PostgreSQL」と「ポスグレ」

Synonym expansion
同義語展開

- Users don't want to care
  ユーザーは気にしたくない
- Synonym expansion
  同義語展開
  - OR search with all synonyms
    同義語すべてでOR検索

Implementation
実装方法

- Create synonym table
  同義語管理テーブルを作成
- Expand synonyms in query
  クエリー内の同義語を展開
- Search by expanded query
  展開後のクエリーで検索

https://pgroonga.github.io/reference/functions/pgroonga-query-expand.html

Table definition

```
CREATE TABLE synonyms (
  -- Term to be expanded
  term text,
  -- Synonym list.
  -- Including the "term" itself.
  -- If you don't input the "term",
  -- the "term" is unsearchable term.
  terms text[]
);
```

Data example

```
INSERT INTO synonyms
  VALUES ('PostgreSQL', -- Expand "PostgreSQL"
          ARRAY['PostgreSQL', 'PG']),
         ('PG', -- Expand "PG"
          ARRAY['PG', 'PostgreSQL']);
```

Data management
データ管理

- Easy to maintain because it's a normal table
  普通のテーブルなので管理が楽
  - Easy to insert/delete/update
    追加・削除・更新が楽
  - Normal backup and replication
    ダンプ・リストアもレプリケーションもいつも通り

Index definition

```
CREATE INDEX synonym_search ON synonyms
  USING PGroonga
  -- ...text_term_search...
  -- For equal search
  (term pgroonga_text_term_search_ops_v2);
```

Confirm
確認方法

```
SELECT pgroonga_query_expand(
  'synonyms', -- Table name
  'term', -- Column name to be expanded
  'terms', -- Column name for synonyms
  'PostgreSQL' -- Query
);
-- '((PostgreSQL) OR (PG))'
```

Search
検索方法

```
SELECT title FROM entries
  WHERE
 -- title &@~ 'DB ((PostgreSQL) OR (PG))'
    title &@~
      pgroonga_query_expand('synonyms',
                            'term',
                            'terms',
                            'DB PostgreSQL');
```

Similar search
類似文書検索

- Query is document itself
  検索クエリーは文書そのもの
  - Not keyword
    キーワードではない
- Use case
  利用例
  - Show related entries
    関連エントリーの提示に使える

Implementation
実現方法

- Create dedicated index
  類似検索用のインデックスを作る
  - Use tokenizer for target language
    対象の言語に合わせた処理で精度向上
  - e.g.: MeCab based tokenizer for Japanese
    例：日本語ならMeCabベースのトークナイザーを活用
- Use dedicated operator
  類似検索用の演算子を使う

Index definition

```
CREATE INDEX entries_similar_search
  ON entries
  -- Target: Both title and content
  -- Reason: Title is important
  USING PGroonga (id, (title || ' ' || content))
  -- TokenMecab is good for Japanese
  WITH (tokenizer='TokenMecab');
```

Search

```
SELECT title,
    pgroonga_score(entries) AS score
  FROM entries
  WHERE
    -- &@* is operator for similar search.
    -- Search with existing document.
    (title || ' ' || content) &@*
      '...fast search with Groonga!...'
  ORDER BY score DESC LIMIT 3;
```

Result example
結果例

```
Query:
    ...search with Groonga!...

Hit example:
    ...search with PGroonga!...
```

Wrap up: Basic features
全文検索システム：基本機能

- Fast FTS + sort
  高速全文検索＋ソート
- Show texts around keyword
  キーワード周辺テキスト表示
- Highlight keyword
  検索キーワードハイライト

Wrap up: Adv. features
全文検索システム：高度な機能

- Auto complete
  オートコンプリート
- Similar search
  類似文書検索
- Synonym expansion
  同義語展開

FTS system: Next step
全文検索システム：次の一歩

- Support structured data
  構造化データ対応
  - Office document, HTML, ...
    オフィス文書・HTMLなど
- Needed features
  対応に必要な処理
  - Text/metadata extraction
    テキスト・メタデータ抽出
  - Create screenshot
    スクリーンショット作成

Extraction tool
抽出ツール

- Apache Tika
  - Apache Lucene's subproject
  - Many supported formats
    対応フォーマットが多い
- ChupaText
  - Groonga's subproject
  - Screenshot support
    スクリーンショット作成対応

ChupaText

- Supported formats (対応フォーマット)
  - Word/Excel/PowerPoint
  - ODT/ODS/ODP (OpenDocument)
  - PDF/HTML/XML/CSV/...
- Interface (インターフェイス)
  - HTTP and command line
    HTTPとコマンドライン

Install
インストール

- Use Docker or Vagrant
  DockerかVagrantを使うのが楽
  - https://github.com/ranguba/chupa-text-docker
  - https://github.com/ranguba/chupa-text-vagrant

ChupaText：Docker

```
% GITHUB=https://github.com
% git clone \
   ${GITHUB}/ranguba/chupa-text-docker.git
% cd chupa-text-docker
% docker-compose up --build
```

Usage
使い方

```
% curl \
    --form data=@XXX.pdf \
    http://localhost:20080/extraction.json
```

Result example
結果例

```
{
  "mime-type": "application/pdf", # MIME type for the original data
  "size": 147159, # Metadata
  ...,
  "texts": [ # Extracted texts
    {
      "mime-type": "text/plain", # MIME type for the extracted data
      ...,
      "creator": "Adobe Illustrator CS3", # Metadata
      "body": "This is sample PDF. ...", # Extracted text
      "screenshot": {
        "mime-type": "image/png", # MIME type for screenshot
        "data": "iVBORw...", # Base64-ed image data
        "encoding": "base64"
      }
    }
  ]
}
```

Web UI

ChupaText

Extraction

Data

参照... ファイルが選択されていません。

URI (optional)

Extract

Web UI: Extraction example
Web UI：抽出例

Extract

Metadata

mime-type	application/pdf
uri	file:/home/chupa-text/chupa-text-http-server/groonga-s and-pgroonga.pdf
path	/tmp/groonga-seminar-2017-08-mroonga-and-pgroong
size	857145

Text #1

Web UI: Extraction example
Web UI：抽出例

size

Text #1

Mroonga・PGroongaを使った
全文検索システムの実装方法
須藤功平
株式会社クリアコード
MySQL・PostgreSQL上で動かす全文検索エン
ー
2017-08-01
Mroonga・PGroongaを使った 全文検索シス
Powered by Rabbit 2.2.1
全文検索システム
対象
大量のテキスト
例：Wikiのデータ
例：オフィス文書のテキスト
例：商品説明・口コミ

ChupaText：Vagrant

```
% GITHUB=https://github.com
% git clone \
   ${GITHUB}/ranguba/chupa-text-vagrant.git
% cd chupa-text-vagrant
% vagrant up
```

Usage is the same as Docker's

使い方はDocker版と同じ

Use cases（活用例）

- Extracted text
  - Insert into PGroonga
- Extracted metadata
  - Insert into PGroonga
  - Use for condition（絞り込みに活用）
- Created screenshot
  - Show in search result（検索結果で表示）

Wrap up
まとめ

- FTS engine via PostgreSQL
  PostgreSQL経由で全文検索エンジン
  - Provide decision info
    採用の判断材料を提供

Wrap up
まとめ

- Show how to impl. FTS system
  全文検索システム実装例を紹介
  - PGroonga
- PGroonga 1.0.0 and 2
  PGroonga 1.0.0と2
  - Painless upgrade

Wrap up
まとめ

- Show how to support structured data
  構造化データの対応方法を紹介
  - ChupaText

Support service
サポートサービス紹介

- Install support (導入支援)
  設計支援・性能検証・移行支援・…
- Development support (開発支援)
  サンプルコード提供・問い合わせ対応・…
- Operation support (運用支援)
  障害対応・チューニング支援・…

Contact (問い合わせ先)

https://www.clear-code.com/contact/?type=groonga