

MySQL・PostgreSQLだけで作る 高速でリッチな全文検索システム

須藤功平

株式会社クリアコード

db tech showcase Tokyo 2017
2017-09-07





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



全文検索システム 目的

- 必要な情報を
- 必要なときに
- 活用



必要な情報を活用

- ×
 - 探している情報が見つからない
- ○
 - 探している情報が見つかる
- ◎
 - 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！



必要なときに活用

- ×
 - なかなか見つからない
- ○
 - すぐに見つかる
- ◎
 - すでに見つかっていた
 - 例：レコメンデーション



実装方法 選択肢

- 全文検索サーバーを使う
- RDBMSを使う
 - MySQL・MariaDB・PostgreSQLを使う



全文検索サーバー案 メリット

- 必要な機能が揃っている
- + α の機能もある
- 速い



全文検索サーバー案 デメリット

- 実装コスト大
 - それぞれ独自の使い方だから
 - マスターデータの同期はどうする？
- メンテナンスコスト大
 - それぞれ独自の仕組みだから



RDBMS案 メリット

- 実装コスト小
 - 新しく覚えることが少ない
 - データの一元管理
- メンテナンスコスト小
 - 既存の運用ノウハウを使える



RDBMS案 デメリット

- 組込機能では機能不足
- SQLの表現力不足
 - 1クエリーで実現できない機能アリ
 - ↑は性能を出しにくい



実現方法 第3の選択肢

- RDBMS経由（SQL）で
全文検索エンジンを使う



メリット

- 高速で豊富な機能
- 実装コスト小
- メンテナンスコスト小



デメリット

- RDBMSに拡張機能が必要
 - DBaaSで使えない



オススの選肢 全文検索の知識ナシ

- まだ単純な機能で十分
 - データ少：RDBMS単独でLIKE
(数十万件とか)
 - データ中：RDBMS組込全文検索機能
- いまどきの全文検索機能が必要
 - RDBMS経由で全文検索エンジン



オススのメの選択肢 全文検索の知識アリ

- カリカリにチューニングしたい
 - RDBMSと全文検索サーバーを併用
- それ以外
 - RDBMS経由で全文検索エンジン



説明する選択肢

RDBMS経由で
全文検索
エンジン



全文検索エンジン Groonga (ぐるんが)

- 組込可能な全文検索エンジン
 - MySQL・MariaDBに組込→Mroonga
 - PostgreSQLに組込→PGoonga
- 全文検索サーバーとして単独でも使用可能
 - RDBMSと全文検索サーバーを併用もできる



Groongaの得意なこと

- データの追加・更新
 - 新鮮な情報をすぐに検索可能に！
 - 更新中も検索性能を落とさない！
- 日本語
 - 開発者が日本人
 - 便利機能が組み込み



Mroonga (むるんが)

- MySQLのストレージエンジン
 - InnoDB・MyISAMなどと同じレイヤー
- 使用方法
 - `CREATE TABLE (...)`
`ENGINE=Mroonga`

MySQL組込の全文検索機能

- MySQL : 5.7から使える
 - InnoDB + 日本語対応パーサー
- MariaDB : 10.0.15から使える
 - Mroongaをバンドル



全文検索機能：基本

	InnoDB	Mroonga
AND/OR/ NOT対応	○	○
ハイライ ト	×	○
周辺テキ スト表示	×	○



ハイライト 周辺テキスト表示

PHP document search

php

全文検索

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

キーワードハイライト

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

キーワード周辺テキスト

```
"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
```

全文検索機能：高度な機能

	InnoDB	Mroonga
入力補完	×	○
類似文書 検索	○	○
クエリー 展開	○	○



全文検索性能の違い 計測データ

- 対象：Wikipedia日本語版
- レコード数：約185万件
- データサイズ：約7GB
- メモリー4GB・SSD250GB (ConoHa)



検索性能1

キーワード：テレビアニメ

(ヒット数：約2万3千件)

InnoDB ngram	3m2s
InnoDB MeCab	6m20s
Mroonga: 1	0.11s



検索性能2

キーワード：データベース

(ヒット数：約1万7千件)

InnoDB ngram	36s
InnoDB MeCab: 1	0.03s
Mroonga: 2	0.09s



検索性能3

キーワード：PostgreSQL OR MySQL
(ヒット数：約400件)

InnoDB ngram	N/A(Error)
InnoDB MeCab: 1	0.005s
Mroonga: 2	0.028s



検索性能4

キーワード：日本

(ヒット数：約63万件)

InnoDB ngram	1.3s
InnoDB MeCab	1.3s
Mroonga: 1	0.21s



検索性能まとめ

- Mroonga : 安定して速い
 - SQLで使えて機能豊富で速い!
- InnoDB FTS MeCab
 - ハマれば速い
- InnoDB FTS ngram
 - 安定して遅い



PGroonga (ピージーるんが)

- PostgreSQLのインデックス
 - B-tree・GINなどと同じレイヤー
- 使用方法
 - CREATE INDEX ...
USING PGroonga ...

PostgreSQLの全文検索機能

- textsearch (組込)
 - 言語依存
 - 日本語対応はメンテされていない
- pg_trgm (同梱)
 - 言語非依存：が、ほぼ日本語非対応
- pg_bigm (サードパーティー)
 - 言語非依存：日本語対応



全文検索機能：基本

	pg_bigm	PGroonga
AND/OR/ NOT対応	△※1	○
ハイライ ト	△※2	○
周辺テキ スト表示	△※2	○

※1 SQLでAND/OR/NOTを組み合わせると実現可能

※2 PostgreSQLが提供する関数で実現可能。ただし、結果をHTMLで出力する用途では使えない。

全文検索機能：高度な機能

	pg_bigm	PGroonga
入力補完	×	○
類似文書 検索	△※	○
クエリー 展開	×	○

※ 類似文書検索と言うよりはあいまい検索。



全文検索性能の違い 計測データ

- 対象：Wikipedia日本語版
- レコード数：約90万件
- データサイズ：約6GB
- メモリー32GB・SSD500GB



検索性能1

キーワード：テレビアニメ

(ヒット数：約2万件)

pg_bigm	2.800s
PGroonga:1	0.065s
Groonga(参考)	0.038s



検索性能2

キーワード：データベース
(ヒット数：約1万5千件)

pg_bigm	1.300s
PGroonga:1	0.049s
Groonga(参考)	0.031s



検索性能3

キーワード：PostgreSQL OR MySQL
(ヒット数：約300件)

pg_bigm	0.049s
PGroonga:1	0.002s
Groonga(参考)	0.001s



検索性能4

キーワード：日本

(ヒット数：約53万件)

pg_bigm:1	0.479s
PGroonga	0.563s
Groonga(参考)	0.059s



検索性能まとめ

- PGroonga : 安定して速い
 - SQLで使えて機能豊富で速い!
- pg_bigm
 - ヒット数が少なければ速い
 - キーワードが2文字以下なら速い

全文検索システムの実装

- 全文検索
- キーワードハイライト
- 周辺テキスト表示
- 入力補完
- 同義語展開
- 関連文書の表示



全文検索

PHP document search

php

全文検索

送信

php .ini ディレクティブのリスト

1010

キーワードハイライト

php .ini ディレクティブのリスト

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

キーワード周辺テキスト

```
"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
```



テーブル定義：MrOonga

```
CREATE TABLE entries (  
  title text,  
  content text,  
  -- 全文検索用インデックス  
  -- よくわからないならデフォルトのまま使うこと!  
  FULLTEXT INDEX (title, content)  
) ENGINE=Mroonga  
  DEFAULT CHARSET=utf8mb4;
```



データ挿入：Mroonga

```
-- 普通に挿入するだけでよい
INSERT INTO entries
VALUES ('タイトル',
       '高速に全文検索したいですね!');
```



全文検索：MrOonga

```
SELECT title FROM entries
WHERE -- MATCH AGAINSTで全文検索
      MATCH (title, content)
      -- デフォルトORがMySQLの仕様
      -- 「検索」または「高速」を含むとマッチ
      AGAINST ('検索 高速'
               IN BOOLEAN MODE);
```



AND全文検索：MrOonga

```
MATCH (title, content)
-- 各キーワードの前に「+」をつけるとAND
-- 「検索」かつ「高速」を含むとマッチ
AGAINST ('+検索 +高速'
          IN BOOLEAN MODE);
```



使いやすいAND全文検索 Mroonga

```
MATCH (title, content)
-- 最初に「*D+」をつけるとデフォルトAND
-- Mroonga独自機能
-- 「検索」かつ「高速」を含むとマッチ
AGAINST ('*D+ 検索 高速'
          IN BOOLEAN MODE);
```



スコアー : Mroonga

```
SELECT
  title,
  -- このMATCH AGAINSTはスコアを返す
  MATCH (title, content)
    AGAINST ('*D+ 検索 高速'
             IN BOOLEAN MODE) AS score
FROM entries
WHERE -- ...
-- スコアが高い順にソート
ORDER BY score DESC LIMIT 10;
```



テーブル定義：PGroonga

```
CREATE TABLE entries (  
  -- プライマリーキーを用意する  
  -- スコアでソートするために必要  
  id integer PRIMARY KEY,  
  title text,  
  content text  
);
```



インデックス定義 PGroonga

```
-- 全文検索用インデックス  
-- よくわからないなら  
-- デフォルトのまま使うこと！  
CREATE INDEX entries_full_text_search  
ON entries  
-- 「USING PGroonga」 = 「PGroongaを使う」  
-- スコアを使うために主キーを入れること！  
USING PGroonga (id, title, content);
```



データ挿入：PGroonga

```
-- 普通に挿入するだけでよい
INSERT INTO entries
VALUES (1,
       'Groongaで高速全文検索！',
       '高速に全文検索したいですね！');
```



全文検索：PGroonga

```
SELECT title FROM entries  
WHERE
```

```
-- &@~で全文検索
```

```
-- 「検索」と「高速」をAND検索
```

```
title &@~ '検索 高速' OR  
content &@~ '検索 高速';
```



全文検索：LIKE PGroonga

```
SELECT title FROM entries  
WHERE
```

```
-- LIKEでもインデックスが効く  
-- = アプリを書き換えずに高速化可能  
-- ただし~より性能が落ちる  
title LIKE '%検索%' OR  
content LIKE '%検索%';
```



スコアー : PGroonga

```
SELECT
  title,
  -- pgroonga_score(テーブル名)で
  -- スコアーを取得
  pgroonga_score(entries) AS score
FROM entries
WHERE -- ...
  -- スコアーが高い順にソート
ORDER BY score DESC LIMIT 10;
```



ハイライト

PHP document search

php

全文検索

送信

php .ini ディレクティブのリスト

1010

キーワードハイライト

php .ini ディレクティブのリスト

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

キーワード周辺テキスト

```
"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
```



ハイライト : Mroonga

```
SELECT mroonga_highlight_html(  
    title, '*D+ 検索 高速' AS query)  
-- クエリーからハイライト対象のキーワードを抽出  
FROM entries  
WHERE  
    MATCH (title, content)  
    AGAINST ('*D+ 検索 高速' IN BOOLEAN MODE);
```



ハイライト結果例

<Groonga>で高速全文検索！

↓

<Groonga>で ← タグをエスケープ

高速

全文 ↑ ↓ キーワードはclass付け

検索！



ハイライト : PGroonga

```
SELECT
  pgroonga_highlight_html(
    title,
    -- クエリーから対象キーワードを抽出
    pgroonga_query_extract_keywords('検索 高速'))
FROM entries
WHERE title &@~ '検索 高速' OR
       content &@~ '検索 高速';
```



ハイライト結果例

<Groonga>で高速全文検索！



<Groonga>で ← タグをエスケープ

高速

全文 ↑ ↓ キーワードはclass付け

検索！



周辺テキスト

PHP document search

php

全文検索

送信

php .ini ディレクティブのリスト

1010

キーワードハイライト

php .ini ディレクティブのリスト

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

キーワード周辺テキスト

```
"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
```



周辺テキスト : MrOonga

```
SELECT mroonga_snippet_html(  
    content, '*D+ 検索 高速' AS query)  
    -- クエリーから対象のキーワードを抽出  
FROM entries  
WHERE  
    MATCH (title, content)  
    AGAINST ('*D+ 検索 高速' IN BOOLEAN MODE);
```



周辺テキスト結果例

...<Groonga>で高速全文検索！...

↓

<div class="snippet"> ←1つ目

ga>;で ←タグをエスケープ

高速

全文 ↑ ↓キーワードはclass付け

検索 !

</div>

<div class="snippet">...</div> ←2つ目



周辺テキスト : PGroonga

```
SELECT
  pgroonga_snippet_html(
    content,
    -- クエリーから対象キーワードを抽出
    pgroonga_query_extract_keywords('検索 高速'))
FROM entries
WHERE title &@~ '検索 高速' OR
       content &@~ '検索 高速';
```



周辺テキスト結果例

...<Groonga>で高速全文検索！...

↓

ARRAY[

↓ 1つ目

'ga>で ←タグをエスケープ

高速

全文 ↑ ↓キーワードはclass付け

検索/span> ! ',

'...' ← 2つ目

]



入力補完

PHP document search

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`



入力補完 実装方法

- 以下の検索のOR
 - ヨミガナでの前方一致検索
 - 緩い全文検索
- 表示文字列でソートして提示

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



入力補完：Mroonga テーブル定義：補完候補

```
CREATE TABLE terms (  
  term varchar(256),      -- 補完候補  
  reading varchar(256),  -- ヨミガナ  
  PRIMARY KEY (term, reading),  
  FULLTEXT INDEX (term)  -- 候補全文検索用  
  -- 緩い全文検索用トークナイザー  
  COMMENT 'tokenizer "TokenBigramSplitSymbolAlpha"',  
  FULLTEXT INDEX (reading) -- ヨミガナ前方一致用  
  COMMENT 'normalizer "NormalizerAuto",  
           tokenizer "off"' -- トークナイザー不要  
) ENGINE=Mroonga DEFAULT CHARSET=utf8mb4;
```



入力補完：Mroonga データ例

```
INSERT INTO terms VALUES (  
  '牛乳', -- 補完候補  
  'ギユウニユウ' --ヨミガナはカタカナで指定  
);  
INSERT INTO terms VALUES (  
  '牛乳',  
  'ミルク' -- 「ミルク」でも補完できるように  
);
```



入力補完：Mroonga データ管理のポイント

- 普通のテーブルなので管理が楽
 - 追加・削除・更新が楽
 - ダンプ・リストアもいつも通り
 - レプリケーションもいつも通り



入力補完：Mroonga 検索方法

```
SELECT DISTINCT(term) FROM terms
WHERE MATCH (reading) -- ヨミガナ前方一致検索
  AGAINST (CONCAT('*SS prefix_rk_search(reading, ',
               mroonga_escape(${入力} AS script),
               ')') IN BOOLEAN MODE) OR
  MATCH (term) -- 候補を緩く全文検索
  AGAINST (CONCAT('*D+ ', mroonga_escape(${入力})))
  IN BOOLEAN MODE)
ORDER BY term LIMIT 10; -- ソート
```



入力補完 : Mroonga 検索例 : 漢字1

```
-- ユーザーが「牛」を入力した場合
SELECT DISTINCT(term) FROM terms
WHERE MATCH (reading) -- ヨミガナ前方一致検索
  AGAINST (CONCAT('*SS prefix_rk_search(reading, ',
              mroonga_escape('牛' AS script),
              ')') IN BOOLEAN MODE) OR
  MATCH (term) -- 候補を緩く全文検索 (ヒット)
  AGAINST (CONCAT('*D+ ', mroonga_escape('牛'))
            IN BOOLEAN MODE)
ORDER BY term LIMIT 10; -- ソート
```



入力補完 : Mroonga

検索例 : 漢字2

```
-- ユーザーが「乳」を入力した場合
SELECT DISTINCT(term) FROM terms
WHERE MATCH (reading) -- ヨミガナ前方一致検索
  AGAINST (CONCAT('*SS prefix_rk_search(reading, ',
              mroonga_escape('乳' AS script),
              ')') IN BOOLEAN MODE) OR
  MATCH (term) -- 候補を緩く全文検索 (ヒット)
  AGAINST (CONCAT('*D+ ', mroonga_escape('乳'))
            IN BOOLEAN MODE)
ORDER BY term LIMIT 10; -- ソート
```



入力補完 検索例：カタカナ

```
-- ユーザーが「ギユウ」を入力した場合
SELECT DISTINCT(term) FROM terms
WHERE MATCH (reading) -- ヨミガナ前方一致検索 (ヒット)
  AGAINST (CONCAT('*SS prefix_rk_search(reading, ',
                mroonga_escape('ギユウ' AS script),
                ')') IN BOOLEAN MODE) OR
  MATCH (term) -- 候補を緩く全文検索
  AGAINST (CONCAT('*D+ ', mroonga_escape('ギユウ'))
    IN BOOLEAN MODE)
ORDER BY term LIMIT 10; -- ソート
```



入力補完 検索例：ひらがな

```
-- ユーザーが「ぎゅう」を入力した場合
SELECT DISTINCT(term) FROM terms
WHERE MATCH (reading) -- ヨミガナ前方一致検索 (ヒット)
  AGAINST (CONCAT('*SS prefix_rk_search(reading, ',
                mroonga_escape('ぎゅう' AS script),
                ')') IN BOOLEAN MODE) OR
  MATCH (term) -- 候補を緩く全文検索
  AGAINST (CONCAT('*D+ ', mroonga_escape('ぎゅう'))
    IN BOOLEAN MODE)
ORDER BY term LIMIT 10; -- ソート
```



入力補完：Mroonga 検索例：ローマ字

```
-- ユーザーが「gyu」を入力した場合
SELECT DISTINCT(term) FROM terms
WHERE MATCH (reading) -- ヨミガナ前方一致検索 (ヒット)
  AGAINST (CONCAT('*SS prefix_rk_search(reading, ',
               mroonga_escape('gyu' AS script),
               ')') IN BOOLEAN MODE) OR
  MATCH (term) -- 候補を緩く全文検索
  AGAINST (CONCAT('*D+ ', mroonga_escape('gyu'))
           IN BOOLEAN MODE)
ORDER BY term LIMIT 10; -- ソート
```



入力補完：PGroonga テーブル定義

```
CREATE TABLE terms (  
  -- 補完候補  
  term text,  
  -- この候補のヨミガナ (N個可)  
  readings text[]  
);
```



入力補完：PGroonga データ例

```
INSERT INTO terms VALUES (  
  '牛乳', -- 補完候補  
  ARRAY[  
    -- ヨミガナはカタカナで指定する  
    'ギユウニユウ',  
    -- 「ミルク」でも補完できるようになる  
    'ミルク'  
  ]  
);
```



入力補完：PGroonga データ管理のポイント

- 普通のテーブルなので管理が楽
 - 追加・削除・更新が楽
 - ダンプ・リストアもいつも通り
 - レプリケーションもいつも通り



入力補完：PGroonga 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);
```



入力補完：PGroonga 緩い全文検索用

```
CREATE INDEX loose_search ON terms
  USING PGroonga (term)
  -- 緩い全文検索用トークナイザー
  WITH (tokenizer='TokenBigramSplitSymbolAlphaDigit');
```



入力補完：PGroonga 検索方法

```
SELECT term FROM terms
  -- ヨミガナで前方一致検索
WHERE readings &^~ '${入力}' OR
  -- 緩い全文検索
  term &@ '${入力}'
ORDER BY term LIMIT 10; -- ソート
```



入力補完 : PGroonga 検索例 : 漢字1

```
-- ユーザーが「牛」を入力した場合  
SELECT term FROM terms  
  -- ヨミガナで前方一致検索  
WHERE readings &^~ '牛' OR  
  -- 緩い全文検索 (ヒット)  
  term &@ '牛'  
ORDER BY term LIMIT 10; -- ソート
```



入力補完：PGroonga 検索例：漢字2

```
-- ユーザーが「乳」を入力した場合  
SELECT term FROM terms  
  -- ヨミガナで前方一致検索  
WHERE readings &^~ '乳' OR  
  -- 緩い全文検索（ヒット）  
  term &@ '乳'  
ORDER BY term LIMIT 10; -- ソート
```



入力補完：PGroonga 検索例：カタカナ

```
-- ユーザーが「ギユウ」を入力した場合  
SELECT term FROM terms  
-- ヨミガナで前方一致検索（ヒット）  
WHERE readings &^~ 'ギユウ' OR  
-- 緩い全文検索  
term &@ 'ギユウ'  
ORDER BY term LIMIT 10; -- ソート
```



入力補完：PGroonga 検索例：ひらがな

```
-- ユーザーが「ぎゅう」を入力した場合  
SELECT term FROM terms  
-- ヨミガナで前方一致検索（ヒット）  
WHERE readings &^~ 'ぎゅう' OR  
-- 緩い全文検索  
term &@ 'ぎゅう'  
ORDER BY term LIMIT 10; -- ソート
```



入力補完：PGroonga 検索例：ローマ字

```
-- ユーザーが「gyu」を入力した場合  
SELECT term FROM terms  
-- ヨミガナで前方一致検索（ヒット）  
WHERE readings &^~ 'gyu' OR  
-- 緩い全文検索  
term &@ 'gyu'  
ORDER BY term LIMIT 10; -- ソート
```



同義語展開

- 同義語
 - 同じ意味だが表記が異なる語
 - 例：「刺身」と「お造り」
- どの表記でもヒットして欲しい
 - 同義語展開→同義語すべてでOR検索



同義語展開 実装方法

- 同義語管理テーブルを作成
- クエリー内の同義語を展開
- 展開後のクエリーで検索

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



同義語展開：Mroonga テーブル定義

```
CREATE TABLE synonyms (  
  term varchar(255),      -- 展開対象の語  
  synonym varchar(255),  -- 同義語  
  INDEX (term)           -- 高速化と精度向上  
  COMMENT 'normalizer "NormalizerAuto"  
) ENGINE=Mroonga DEFAULT CHARSET=utf8mb4;
```



同義語展開 データ例

```
INSERT INTO synonyms
```

```
-- 「刺身」を「刺身 OR お造り」に展開
```

```
VALUES ('刺身', '刺身'),
```

```
      ('刺身', 'お造り'),
```

```
-- 「お造り」を「お造り OR 刺身」に展開
```

```
      ('お造り', 'お造り'),
```

```
      ('お造り', '刺身');
```



同義語展開 データ管理のポイント

- 普通のテーブルなので管理が楽
 - 追加・削除・更新が楽
 - ダンプ・リストアもいつも通り
 - レプリケーションもいつも通り



同義語展開：Mroonga 確認方法

```
SELECT mroonga_query_expand(  
  'synonyms',    -- テーブル名  
  'term',        -- 展開対象のカラム名  
  'synonym',     -- 対応する同義語のカラム名  
  '居酒屋 刺身' -- クエリー  
);  
-- '居酒屋 ((刺身) OR (お造り))'
```



同義語展開：Mroonga 検索方法

```
SELECT title FROM entries
WHERE
  MATCH (title)
  -- '*D+ 居酒屋 OR ((刺身) OR (お造り))'になる
  AGAINST (mroonga_query_expand('synonyms',
                                'term',
                                'synonym',
                                '*D+ 居酒屋 刺身')
           IN BOOLEAN MODE);
```



同義語展開：PGroonga テーブル定義

```
CREATE TABLE synonyms (  
  -- 展開対象の語  
  term text,  
  -- 同義語のリスト  
  -- term自身も含める  
  -- 含めない場合はtermが検索禁止語になる  
  terms text[]  
);
```



同義語展開：PGroonga データ例

```
INSERT INTO synonyms
VALUES ('刺身', -- 「刺身」を展開
       ARRAY['刺身', 'お造り']),
('お造り', -- 「お造り」を展開
       ARRAY['お造り', '刺身']);
```



同義語展開：PGroonga データ管理のポイント

- 普通のテーブルなので管理が楽
 - 追加・削除・更新が楽
 - ダンプ・リストアもいつも通り
 - レプリケーションもいつも通り



同義語展開：PGroonga インデックス定義

```
CREATE INDEX synonym_search ON synonyms
  USING PGroonga
  -- ...text_term_search...
  -- termで完全一致検索をするため
  (term pgroonga_text_term_search_ops_v2);
```



同義語展開：PGroonga 確認方法

```
SELECT pgroonga_query_expand(  
  'synonyms', -- テーブル名  
  'term', -- 展開対象のカラム名  
  'terms', -- 対応する同義語配列のカラム名  
  '刺身' -- クエリー  
);  
-- '((刺身) OR (お造り))'
```



同義語展開：PGroonga 検索方法

```
SELECT title FROM entries
WHERE
-- title &@~ '居酒屋 ((刺身) OR (お造り))'になる
title &@~
    pgroonga_query_expand('synonyms',
                          'term',
                          'terms',
                          '居酒屋 刺身');
```



類似文書検索

- 検索クエリーは文書そのもの
 - キーワードではない
- 関連エントリーの提示に使える
 - メタデータがあるなら組み合わせる
→精度向上
 - メタデータ：タグ・行動履歴など



類似文書検索：MrOonga インデックス定義

```
CREATE TABLE entries (  
  -- ...  
  FULLTEXT INDEX (content)  
  -- TokenMecabを使わないと精度がでない  
  -- 必要なときだけカスタマイズ!  
  COMMENT 'tokenizer "TokenMecab"  
) -- ...
```



類似文書検索：Mrroonga 検索方法

```
SELECT title
FROM entries
WHERE
  MATCH (content)
  -- ↓ 既存文書の内容をそのまま指定
  AGAINST ('...Groongaで高速全文検索！...')
  IN NATURAL LANGUAGE MODE);
```



類似文書検索：Mroonga 結果例

クエリー：

...Groongaで高速全文検索！...

ヒット例：

...Mroongaで高速全文検索！...



照合順序：COLLATION

- 文字の並び順の規則
 - 文字が同一かどうかの判定にも利用
- 適切な日本語規則なし
 - いわゆる 🍷 = 🍺 問題

MySQL 8では適切な日本語規則が追加される



MrOongaの照合順序

- MySQL互換のもの
- MySQL互換を微調整したもの
 - 日本語でもいい感じ
- Groonga提供のもの
 - 日本語でもいい感じ



微調整した照合順序

```
FULLTEXT INDEX (content)
```

```
COMMENT 'normalizer "${ノーマライザー名}"'
```

ノーマライザー名 :

```
NormalizerMySQLUnicode520CI
```

```
ExceptKanaCI
```

```
KanaWithVoicedSoundMark
```



類似文書検索：PGroonga インデックス定義

```
CREATE INDEX entries_similar_search
ON entries
USING PGroonga (id, content)
-- TokenMecabを使うと精度向上
WITH (tokenizer='TokenMecab');
```



類似文書検索：PGroonga 検索方法

```
SELECT title
FROM entries
WHERE
  -- &@*で類似文書検索
  -- 既存文書の内容をそのまま指定
  content &@*
  '...Groongaで高速全文検索！...';
```



類似文書検索：PGroonga 結果例

クエリー：

...Groongaで高速全文検索！...

ヒット例：

...PGroongaで高速全文検索！...

全文検索システムの実装 まとめ

- 全文検索
- キーワードハイライト
- 周辺テキスト表示
- 入力補完・同義語展開
- 関連文書の表示

全文検索システムの実装 次の一歩

- 構造化データ対応
 - オフィス文書・HTMLなど
- 対応に必要な処理
 - テキスト抽出
 - メタデータ抽出 (例：タイトル・更新日時)
 - スクリーンショット作成 (なおよい)



抽出ツール

- Apache Tika
 - Apache Luceneのサブプロジェクト
 - 対応フォーマット数が多い
- ChupaText
 - Groongaのサブプロジェクト
 - スクリーンショット作成対応



ChupaText

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

ChupaText : インストール

- 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
```



ChupaText : 使い方

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



ChupaText : 結果例

```
{
  "mime-type": "application/pdf", # 元データのMIMEタイプ
  "size": 147159, # メタデータ
  ...,
  "texts": [ # 抽出されたテキスト (N個)
    {
      "mime-type": "text/plain", # 抽出後のMIMEタイプ
      ...,
      "creator": "Adobe Illustrator CS3", # メタデータ
      "body": "This is sample PDF. ...", # 抽出したテキスト
      "screenshot": {
        "mime-type": "image/png", # スクリーンショットのMIMEタイプ
        "data": "iVBORw...", # Base64にした画像データ
        "encoding": "base64" # Base64であることを明記
      }
    }
  ]
}
```



ChupaText : Web UI

ChupaText

Extraction

Data

参照...

ファイルが選択されていません。

URI (optional)

Extract

ChupaText : 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

ChupaText : Web UI抽出例

Text #1



Mrroonga・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

Mrroonga・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：...



ChupaText : Vagrant

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

使い方はDocker版と同じ



ChupaText : 活用例

- 抽出したテキスト
 - Mroonga・PGroongaへ挿入
- 抽出したメタデータ
 - Mroonga・PGroongaへ挿入
 - 絞り込みに活用
- 作成したスクリーンショット
 - 検索結果表示時に掲載



まとめ

- RDBMS経由で全文検索エンジン
 - 採用の判断材料を提供
- 全文検索システム実装例を紹介
 - Mroonga・PGroonga両方
- 構造化データの対応方法を紹介
 - ChupaText



扱わなかった話題

- 運用について
 - 障害対策・レプリケーション
- チューニング
- Groongaの機能を直接使う方法



サポートサービス紹介

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

問い合わせ先：

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