PGroonga 2

PostgreSQLでの全文検索の決定版

須藤功平　　クリアコード

PostgreSQL Conference Japan 2017
2017-11-03

対象者

- PostgreSQLで全文検索したい
- 全文検索はよく知らない
- PGroongaは使ったことがない

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
- 例：チャットログ

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
- 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

PostgreSQL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

PostgreSQL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

- PostgreSQL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

- PostgreSQLに拡張機能が必要
  - DBaaSで使えない

オススメの選択肢
全文検索の知識ナシ

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

オススメの選択肢
全文検索の知識アリ

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

説明する選択肢

PostgreSQL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
  - PostgreSQLに組込→PGoonga
- 全文検索サーバーとして単独でも使用可能
  - PostgreSQL＋全文検索サーバー構成もできる

Groongaの得意なこと データの追加・更新

- 新鮮な情報がすぐ検索可能！
  - バッチで更新しなくてもよい
  - チャットくらいの頻度でもOK
    例：ZulipはPGroongaを採用
- 更新中も検索性能が落ちない！
  - 利用ユーザーが多い時でも更新可能

Groongaの得意なこと
日本語まわり

- 開発者が日本人
- 便利機能が組み込み

もちろん、日本語以外もOK！

PGroonga (ぴーじーるんが)

- 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と全文検索

- LIKE：組込機能
- textsearch：組込機能
- pg_trgm：標準添付
  - アーカイブには含まれている
  - 別途インストールすれば使える

LIKEと速度

- 少ないデータ
  - 十分実用的
  - 400文字×20万件くらいなら1秒とか
- 少なくないデータ
  - 性能問題アリ

LIKEと全文検索システム

- 👍速度が実用的なことも多い
  - 少ないデータなら
- 👎それっぽい順のソート不可
  - 全文検索ではソート順が重要
  - ユーザーは先頭n件しか見ない

textsearch

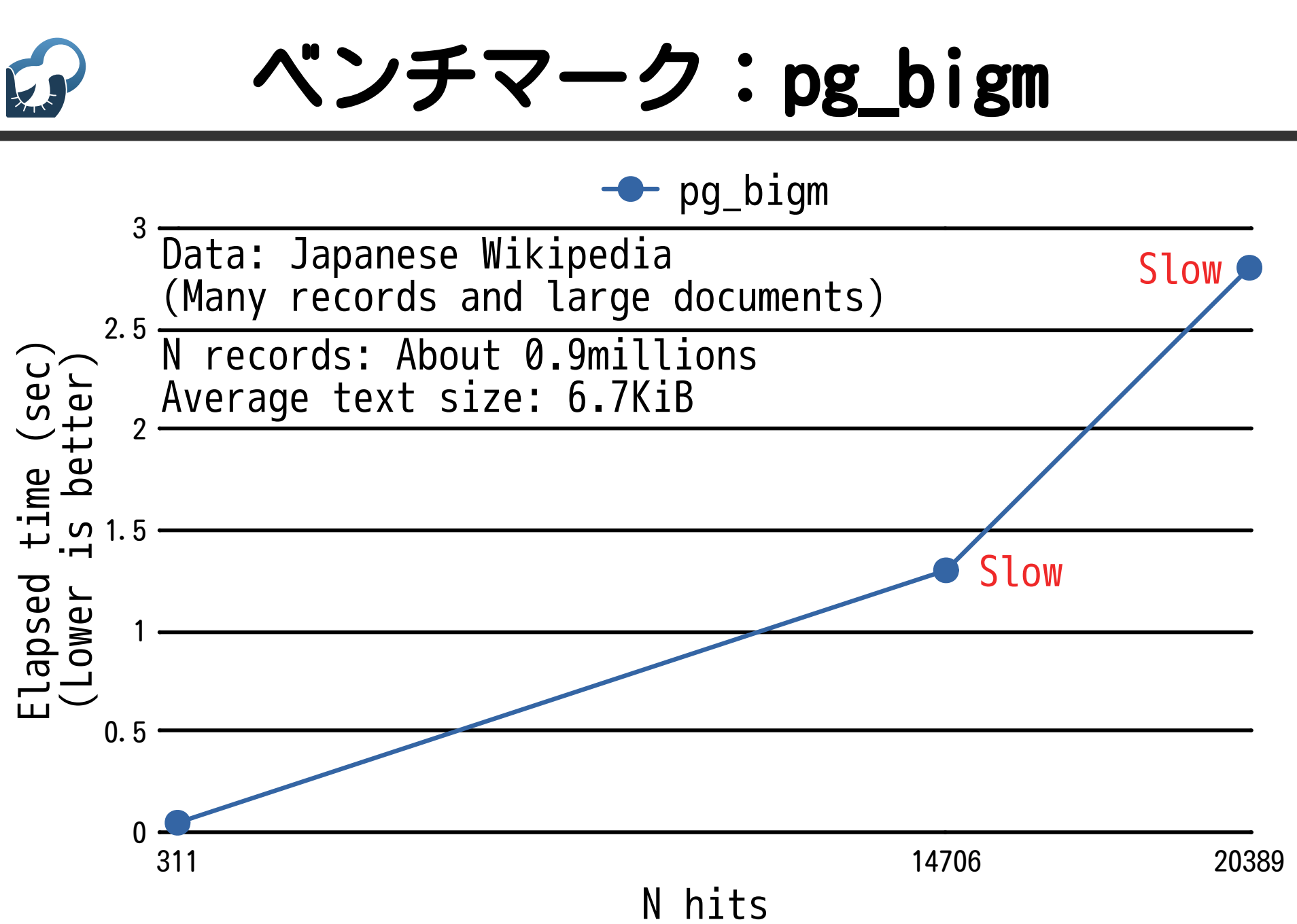
- インデックスを作るので速い
- 言語毎にモジュールが必要
  - 英語やフランス語などは組込
  - 日本語は別途必要
- 日本語用モジュールはあるが…
  - 公式にはメンテナンスされていない
    forkして動くようにしている人はいる

pg_trgm

- インデックスを作るので速い
  - 注：ヒット件数が増えると遅い
  - 注：テキスト量が多いと遅い
  - 注：1,2文字の検索は遅い（米・日本）
- 日本語を使うにはひと工夫必要
  - C.UTF-8を使う
  - ソースを変更してビルド

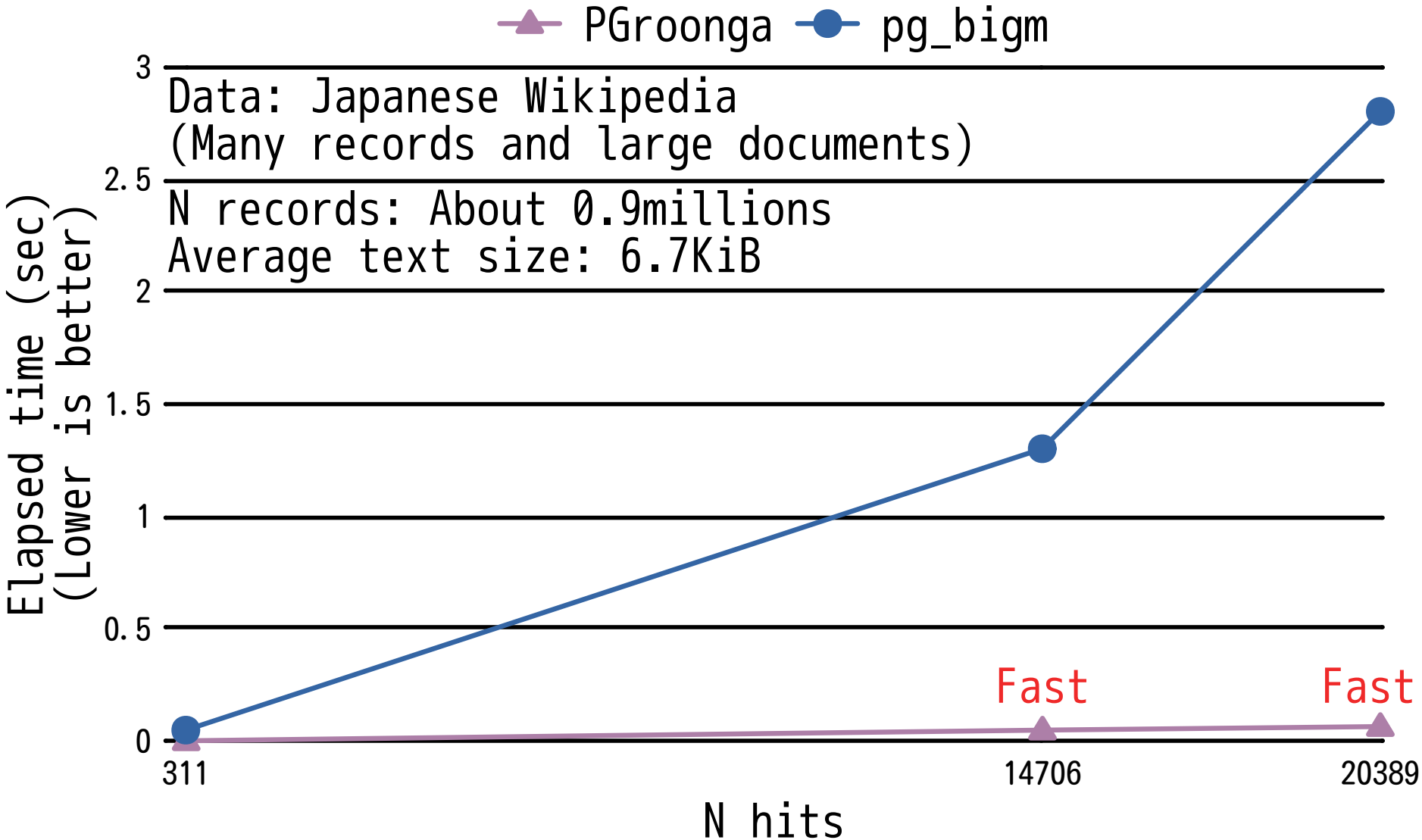
プラグイン

- pg_bigm
  - pg_trgmの日本語対応強化版
  - それっぽい順のソート不可
- PGroonga
  - 本気の全文検索エンジンを利用
  - 速いし日本語もバッチリ！
  - それっぽい順のソート可



ベンチマーク：pg_bigm

ベンチマーク：PGroonga



PostgreSQLで全文検索システム

- PostgreSQLで全文検索
  - PGroongaがベスト！💯
- PGroonga
  - 高速
  - 日本語対応
  - それっぽい順でソート可

基本機能

- 高速全文検索＋ソート
- 検索キーワードハイライト
- キーワード周辺テキスト表示

高度な機能

- オートコンプリート
  - ローマ字対応（zen→全文検索）
- 類似文書検索
- 同義語展開
  - 「牛乳」→
    「牛乳 OR ミルク」

高速全文検索＋ソート

PHP document search

全文検索

php

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

```
CREATE TABLE entries (
  -- 主キーを用意する
  -- それっぽい順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

```
-- 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」
  -- 主キーはそれっぽい順ソートのため！
  USING PGroonga (id, title, content);
```

データ挿入

```
-- 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

```
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

```
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%';
```

それっぽい順のソート

```
SELECT
    title,
    -- pgroonga_score(テーブル名)で
    -- それっぽさを数値で取得
    pgroonga_score(entries) AS score
  FROM entries
  WHERE -- ...
  -- それっぽさでソート
  ORDER BY score DESC LIMIT 10;
```

キーワードハイライト

PHP document search

全文検索

php

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

HTML用にハイライト

```
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';
```

HTML用ハイライト結果例

```
<Groonga>で高速全文検索！
↓
&lt;Groonga&gt;で ← タグをエスケープ
<span class="keyword">高速</span>
全文 ↑↓キーワードはclass付け
<span class="keyword">検索</span>！
```

周辺テキスト

PHP document search

全文検索

php

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

HTML用に周辺テキスト取得

```
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';
```

HTML用周辺テキスト結果例

```
...<Groonga>で高速全文検索！...
↓
ARRAY[
   ↓ 1つ目
   'ga&gt;で ←タグをエスケープ
    <span class="keyword">高速</span>
    全文 ↑↓キーワードはclass付け
    <span class="keyword">検索/span>！',
   '...' ← 2つ目
]
```

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

オートコンプリート：必要なもの

- マスターテーブル
  - 候補（例：牛乳）
  - 候補のヨミ（カタカナ・複数可）
    - 例1：ギュウニュウ
    - 例2：ミルク

オートコンプリート：実装方法

- 以下の検索のOR
  - ヨミでの前方一致検索
  - 候補を緩い全文検索
- 候補でソートして提示

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

オートコンプリート テーブル定義

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

オートコンプリート：データ例

```
INSERT INTO terms VALUES (
  '牛乳', -- 補完候補
  ARRAY[
    -- ヨミはカタカナで指定
    'ギュウニュウ',
    -- 「ミルク」でも補完可
    'ミルク'
  ]
);
```

オートコンプリート
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

オートコンプリート
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);
```

オートコンプリート
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');
```

オートコンプリート検索方法

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

オートコンプリート検索例：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
```

オートコンプリート検索例：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
```

オートコンプリート検索例：カタカナ

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

オートコンプリート検索例：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
```

オートコンプリート検索例：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：「牛乳」と「ミルク」
- 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：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：データ例

```
INSERT INTO synonyms
  VALUES ('牛乳', -- 「牛乳」を展開
          ARRAY['牛乳', 'ミルク']),
         ('ミルク', -- 「ミルク」を展開
          ARRAY['ミルク', '牛乳']);
```

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：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：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
  '牛乳' -- クエリー
);
-- '((牛乳) OR (ミルク))'
```

同義語展開：検索方法

```
SELECT title FROM entries
  WHERE
 -- title &@~ 'アイス ((牛乳) OR (ミルク))'になる
    title &@~
      pgroonga_query_expand('synonyms',
                            'term',
                            'terms',
                            'アイス 牛乳');
```

類似文書検索

PHP document search

エラー制御演算子　送信

エラー制御演算子　11　エラー関連のマニュアルを提示

Similar entries

1. BEGIN_SILENCE　381303
2. END_SILENCE　381303
3. 出力する PHP エラーの種類を設定する　327711

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：実現方法

- 類似検索用インデックスが必要
  - 自然言語に合わせた処理で精度向上
  - 日本語ならMeCabを活用
- 類似検索用の演算子を使う

類似文書検索：インデックス定義

```
CREATE INDEX entries_similar_search
  ON entries
  -- タイトルと内容を合わせたテキストをインデックス
  -- 理由1：タイトルも重要→対象に加えて精度向上
  -- 理由2：PostgreSQLが全文検索インデックスと
  --         区別できるように
  USING PGroonga (id, (title || ' ' || content))
  -- TokenMecabを使うと精度向上
  WITH (tokenizer='TokenMecab');
```

類似文書検索：検索方法

```
SELECT title,
    pgroonga_score(entries) AS score
  FROM entries
  WHERE
    -- &@*で類似文書検索
    -- 既存文書の内容をそのまま指定
    (title || ' ' || content) &@*
      '...Groongaで高速全文検索！...'
  ORDER BY score DESC LIMIT 3;
```

類似文書検索：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！...

おさらい：基本機能

- 高速全文検索＋ソート
- 検索キーワードハイライト
- キーワード周辺テキスト表示

おさらい：高度な機能

- オートコンプリート
  - ローマ字対応（zen→全文検索）
- 類似文書検索
- 同義語展開
  - 「牛乳」→
    「牛乳 OR ミルク」

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

size	651143

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ジン「Groonga」セミナー
2017-08-01
Mroonga・PGroongaを使った 全文検索システム
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
```

使い方はDocker版と同じ

ChupaText：活用例

- 抽出したテキスト
  - PGroongaへ挿入
- 抽出したメタデータ
  - 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

- PostgreSQL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
  - PGroonga
- 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
  - ロジカルレプリケーションは対応済み！（少しだけ設定方法を紹介）
- チューニング
- Groongaの機能を直接使う方法

ロジカルレプリケーション postgresql.conf

```
# マスター
wal_level = logical
max_wal_senders = 10
max_replication_slots = 10
```

ロジカルレプリケーション DB作成・PGroongaインストール

```
-- マスター
CREATE DATABASE test_master;
\c test_master
CREATE EXTENSION pgroonga;
```

ロジカルレプリケーション テーブル作成

```
-- マスター
CREATE TABLE memos (
  id integer PRIMARY KEY,
  content text
);
CREATE INDEX
  full_text_search_index ON memos
  USING pgroonga (id, content);
```

ロジカルレプリケーション レプリケーション用ユーザー作成

```
-- マスター
-- pg_hba.confも編集すること
CREATE ROLE replica
  WITH REPLICATION
    LOGIN PASSWORD 'pass';
GRANT SELECT
  ON ALL TABLES IN SCHEMA public
  TO replica;
```

ロジカルレプリケーション publication作成

```
-- マスター
CREATE PUBLICATION pub
  FOR TABLE memos;
```

ロジカルレプリケーション DB作成・PGroongaインストール

```
-- スレーブ
CREATE DATABASE test_slave;
\c test_slave
CREATE EXTENSION pgroonga;
```

ロジカルレプリケーション テーブル作成

```
-- スレーブ
CREATE TABLE memos (
  id integer PRIMARY KEY,
  content text
);
CREATE INDEX
  full_text_search_index ON memos
  USING pgroonga (id, content);
```

ロジカルレプリケーション subscription作成

```
-- スレーブ
CREATE SUBSCRIPTION sub
  CONNECTION
    'host=192.168.0.18 port=5432
     user=replica password=pass
     dbname=test_master'
  PUBLICATION pub;
```

ロジカルレプリケーション確認

```
-- マスター
INSERT INTO memos VALUES (1, '全文検索');
-- スレーブ
SELECT * FROM memos
  WHERE content &@~ '全文検索';
--  id | content
-- ----+----------
--   1 | 全文検索
-- (1 row)
```

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