MariaDBとMroongaで作る 全言語対応 超高速全文検索システム

須藤功平

クリアコード

第一回 JPMUG DB勉強会
2018-01-30

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
- MariaDBでLIKE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

MariaDBでLIKE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
- データ少なら実用的な速度

MariaDBでLIKE案 デメリット

- 機能不足
  - 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
- SQLの表現力不足
  - nクエリーで実現すると性能に影響

実現方法
第3の選択肢

- MariaDB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
  - それっぽい順のソート可
- 実装コスト小
- メンテナンスコスト小

デメリット

- MariaDBに拡張機能が必要
  - RDS・Azure databaseで使えない
    （Azure database for MariaDBはまだリリース前）

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：MariaDB単独でLIKE
（数十万件とか）
  - データ中以上：
MariaDB経由で全文検索エンジン
- いまどきの全文検索機能が必要
  - MariaDB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
  - MariaDBと全文検索サーバーを併用
- それ以外
  - MariaDB経由で全文検索エンジン

説明する選択肢

MariaDB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
  - MariaDB・MySQLに組込→Mroonga
  - PostgreSQLに組込→PGroonga
- 全文検索サーバーとして単独でも使用可能
  - MariaDB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

GroongaとUnicode

- NFKCベースの正規化機能を組込
- Unicode 5.1ベースで古い
  - 2008年の仕様
- Unicode 10.0（最新）対応中

  正規化方式を変えるとインデックスの互換性がなくなる（=要インデックス再構築）のでデフォルトは変えない

Mroonga（むるんが）

- MariaDBのストレージエンジン
  - InnoDB・MyISAMなどと同じレイヤー
  - MariaDB 10.0.15から標準バンドル
- 使用方法
  - CREATE TABLE (...)
    ENGINE=Mroonga

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
utf8mb4_ja_0900_as_csなど

Mroongaの照合順序

- MariaDB互換のもの
  - utf8mb4_ja_0900_*互換は対応予定
- MariaDB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
  - NFKCベースのもの
  - 日本語でもいい感じ

Mroongaで照合順序

- MariaDB互換がよい！
  - 互換正規化処理を使用：デフォルト
- MariaDB互換は気にしないからいい感じに！
  - Groonga提供のものを使用

全文検索性能計測データ

- 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

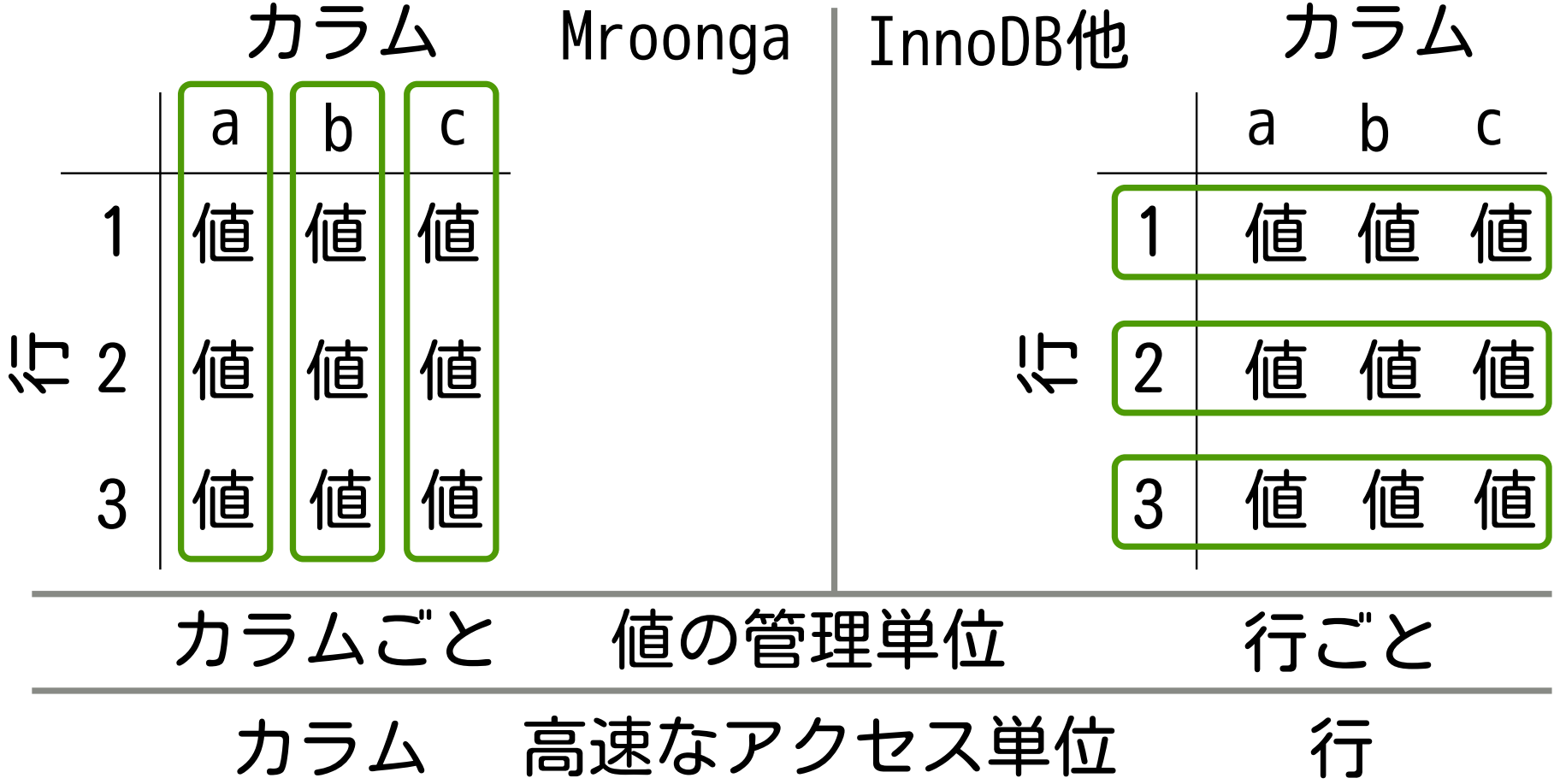
全文検索性能まとめ

- 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

普通の検索も速い

- カラムストアを活かした最適化
  - ポイント1：余計なI/Oを減らす
  - ポイント2：I/Oを局所化

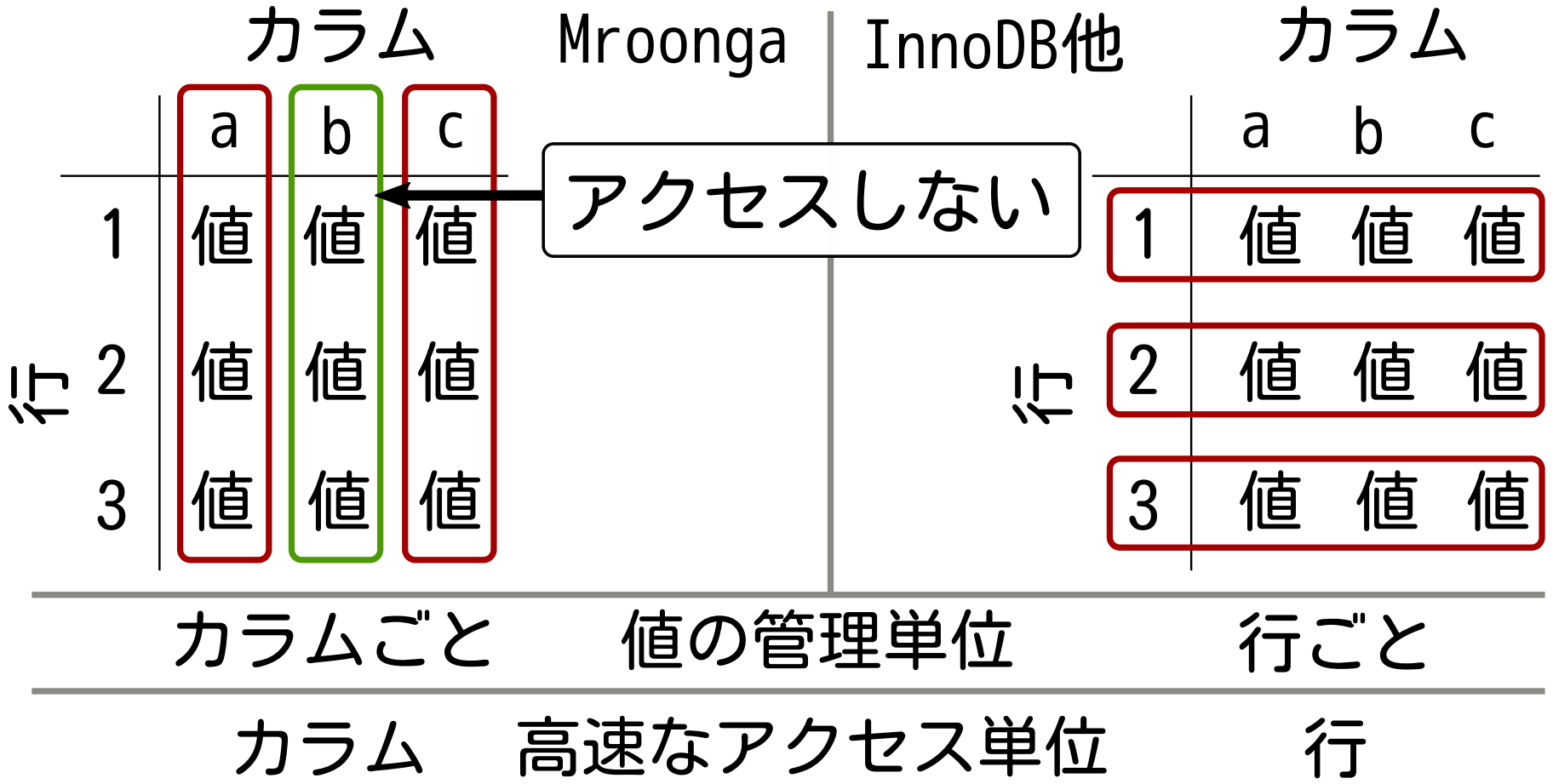
カラムストア


カラム
Mroonga
InnoDB他
カラム
a b c
1 値 値 値
2 値 値 値
3 値 値 値
1 値 値 値
2 値 値 値
3 値 値 値
カラムごと
値の管理単位
行ごと
カラム
高速なアクセス単位
行

必要なカラムのみアクセス

```
-- aのみにアクセス
SELECT a
  FROM table
-- cのみにアクセス
 WHERE c = XXX;
-- bにはアクセスしない
```

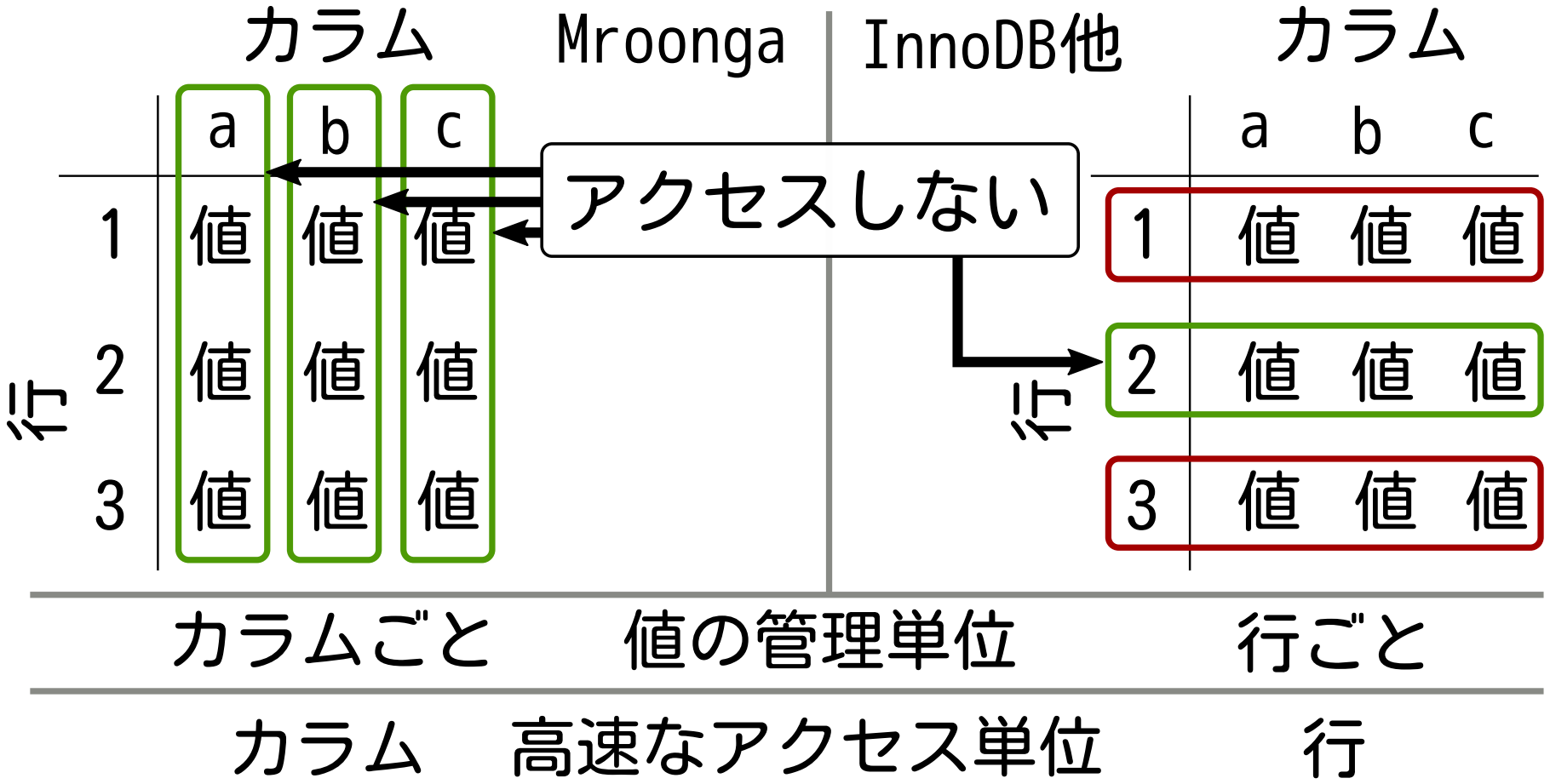
減ったI/O


カラム
Mroonga
InnoDB他
カラム
a b c
1 値 値 値
2 値 値 値
3 値 値 値
行
アクセスしない
a b c
1 値 値 値
2 値 値 値
3 値 値 値
行
カラムごと
値の管理単位
行ごと
カラム
高速なアクセス単位
行

行カウント

```
-- カラムの値は必要ない
SELECT COUNT(*)
  FROM table
-- cの全文検索インデックスにだけアクセス
 WHERE MATCH(c)
       AGAINST('+keyword' IN BOOLEAN MODE);
-- a, b, cはアクセスしない
```

減ったI/O


カラム
Mroonga
InnoDB他
カラム
a
b
c
a
b
c
1
値
値
値
1
値
値
値
2
値
値
値
2
値
値
値
3
値
値
値
3
値
値
値
行
行
アクセスしない
カラムごと
値の管理単位
行ごと
カラム
高速なアクセス単位
行

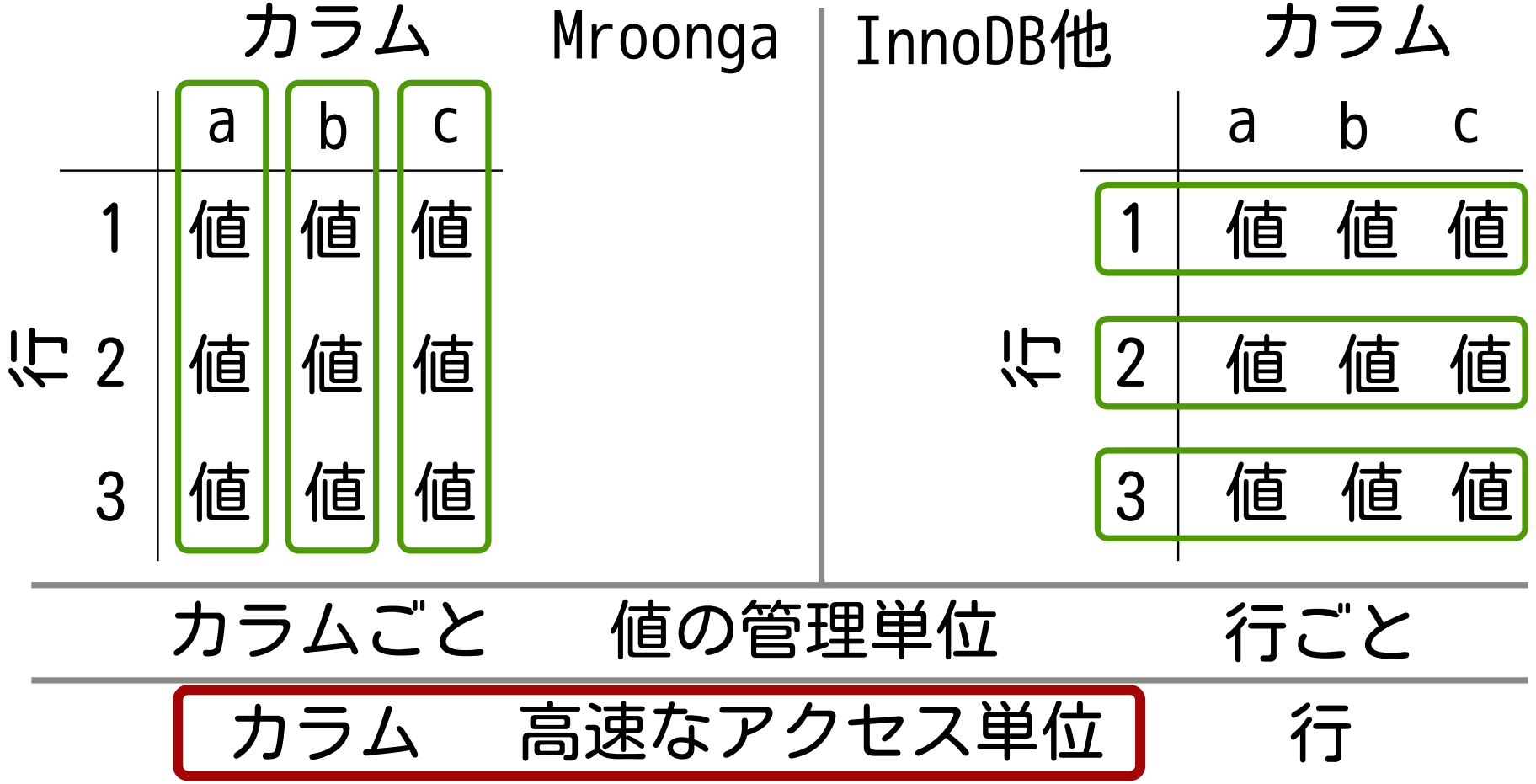
ORDER BY LIMIT

```
SELECT a
  FROM table
 WHERE MATCH(c)
       AGAINST('+keyword' IN BOOLEAN MODE)
-- MariaDBではなくMroongaがORDER BY LIMITを処理
-- →Mroongaは10レコードだけMariaDBに返す
--   マッチしたレコードすべては返さない
 ORDER BY a LIMIT 10;
```

ORDER BY LIMITの最適化

- Mroongaが検索
  - カラム毎の処理でI/Oを局所化
    （索引非使用時）
- Mroongaがソート
  - カラム毎の処理でI/Oを局所化
- MroongaがOFFSET/LIMITを処理

カラム毎の処理は速い


カラム
Mroonga
InnoDB他
カラム
a b c
1 値 値 値
2 値 値 値
3 値 値 値
1 値 値 値
2 値 値 値
3 値 値 値
カラムごと
値の管理単位
行ごと
カラム
高速なアクセス単位
行

condition push downの最適化

- 従来はMariaDBが処理していた検索条件をストレージエンジンが処理する仕組み
  - ストレージエンジンでの処理の方が高速なら全体として高速になる
- Mroonga 7.10から実験的に対応
  - デフォルトオフ

condition push downの効果

- 全文検索インデックスのみ
  - 等価条件：シーケンシャルスキャン
  - 全文検索：インデックススキャン
- データ
  - シカゴの犯罪データ（651万レコード）

詳細：https://github.com/kou/rabbit-slide-kou-jpmug-db-study-1/blob/master/memo.md

等価条件：数値1つ

26万件ヒットするケース

InnoDB	1.3s
Mroonga（デフォルト）	1.3s
Mroonga（最適化ON）	0.4s

等価条件：数値1つ+真偽値2つ

7000件ヒットするケース

InnoDB	1.6s
Mroonga（デフォルト）	2.3s
Mroonga（最適化ON）	0.4s

全文検索+等価条件

4000件ヒットするケース

InnoDB	18s
Mroonga（デフォルト）	0.4s
Mroonga（最適化ON）	0.4s

このパターンはデフォルトで最適化が効く

Mroongaの検索性能まとめ

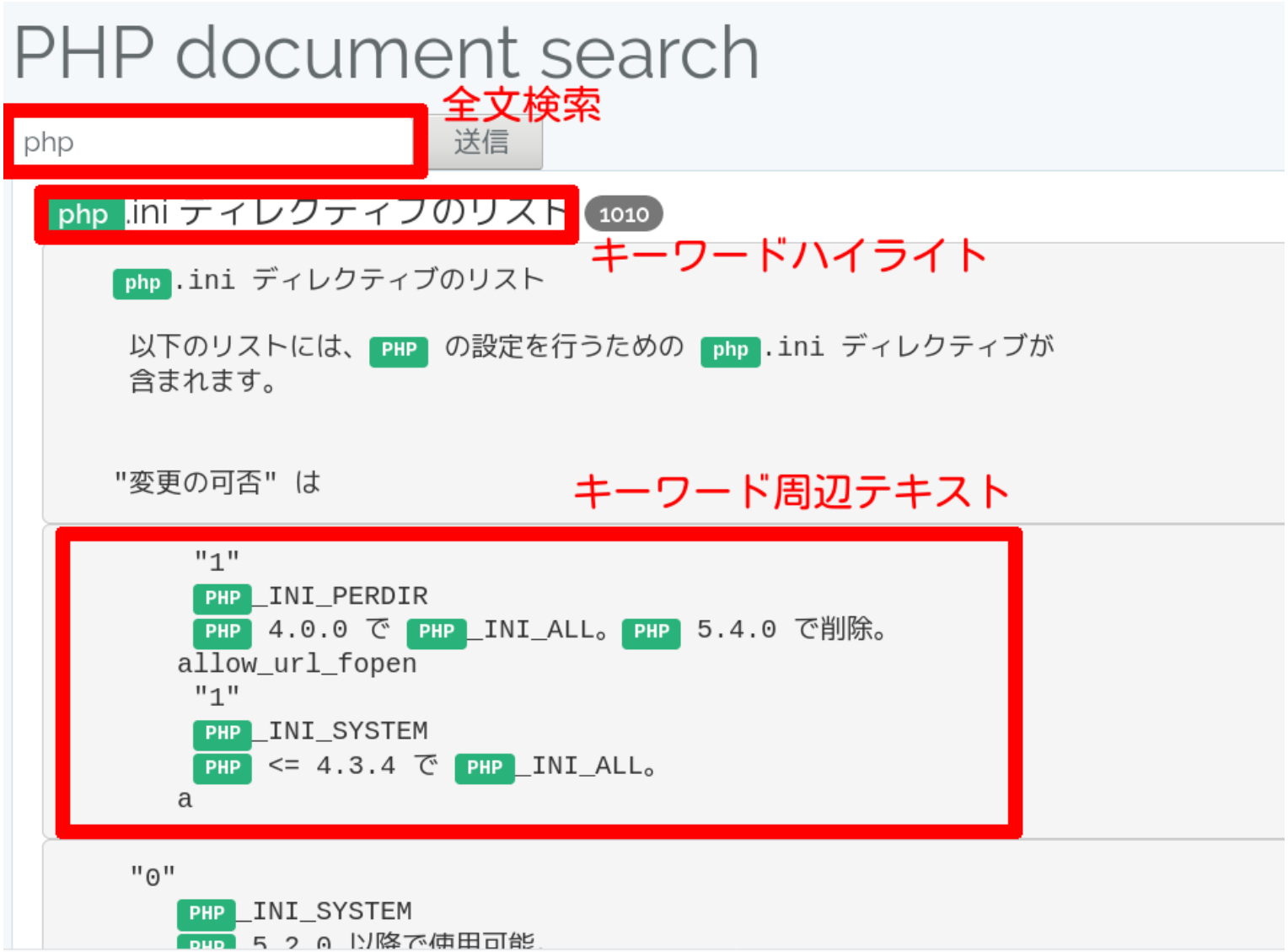
- 最適化が効くと桁違いに速い
  - 全文検索のときはデフォルトで効く
  - 7.10からさらなる最適化が！
    まだ実験的扱いなのでデフォルトオフ
- OLAP用途にも使える
  - MariaDB ColumnStoreを補完する立ち位置も可

参考：http://mroonga.org/ja/docs/reference/server_variables.html#mroonga-condition-push-down-type

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
- オートコンプリート
- 同義語展開
- 関連文書の表示

全文検索


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

テーブル定義

```
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

```
-- 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

```
SELECT title FROM entries
  WHERE -- MATCH AGAINSTで全文検索
    MATCH (title, content)
    -- デフォルトORがMariaDB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

```
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);
```

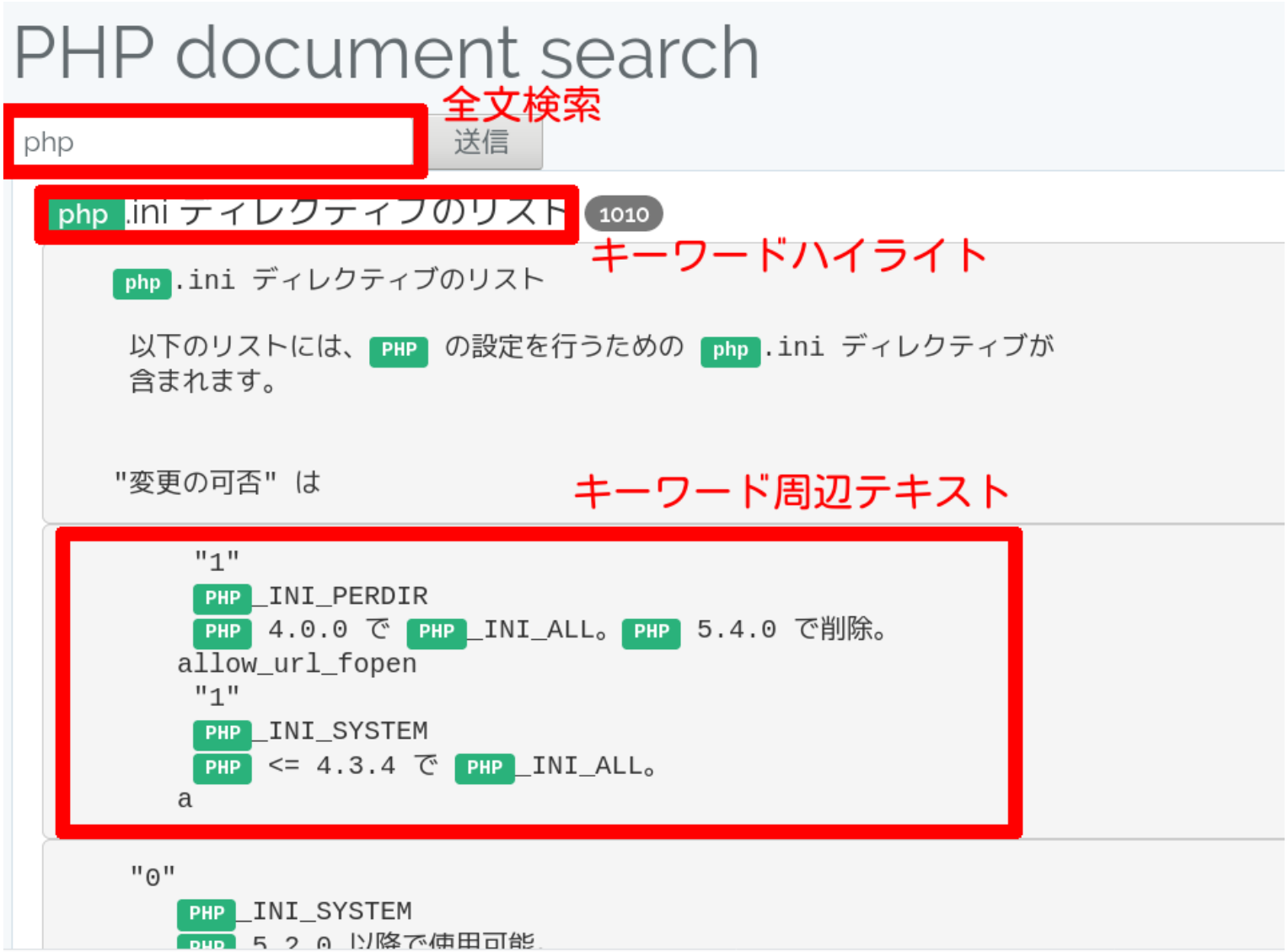
使いやすいAND全文検索

```
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);
```

それっぽい順のソート

```
SELECT
  title,
  -- ここのMATCH AGAINSTはスコアーを返す
  MATCH (title, content)
    AGAINST ('*D+ 検索 高速'
             IN BOOLEAN MODE) AS score
  FROM entries
  WHERE -- ...
  -- それっぽさでソート
  ORDER BY score DESC LIMIT 10;
```

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

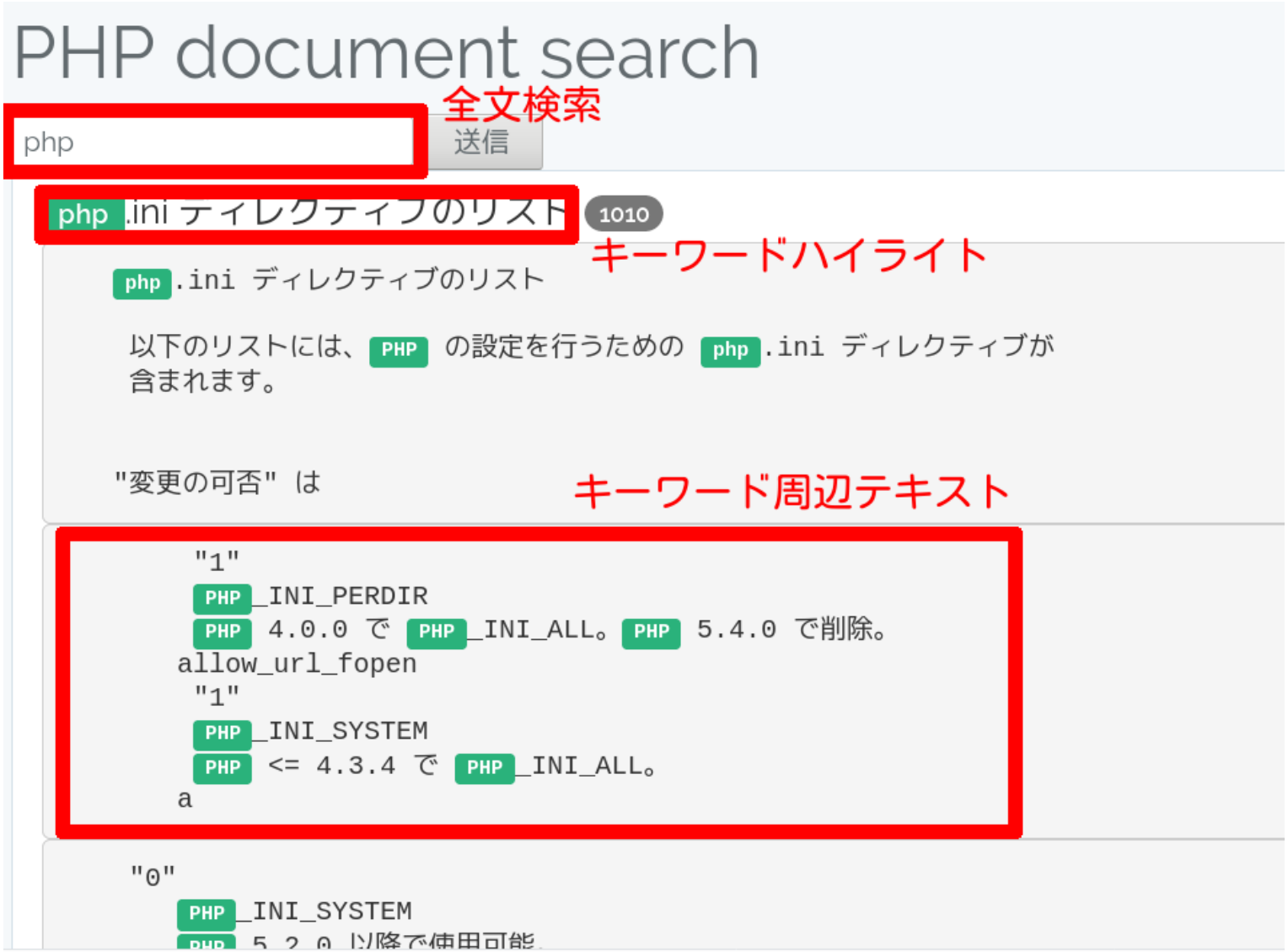
ハイライト

```
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！
↓
&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

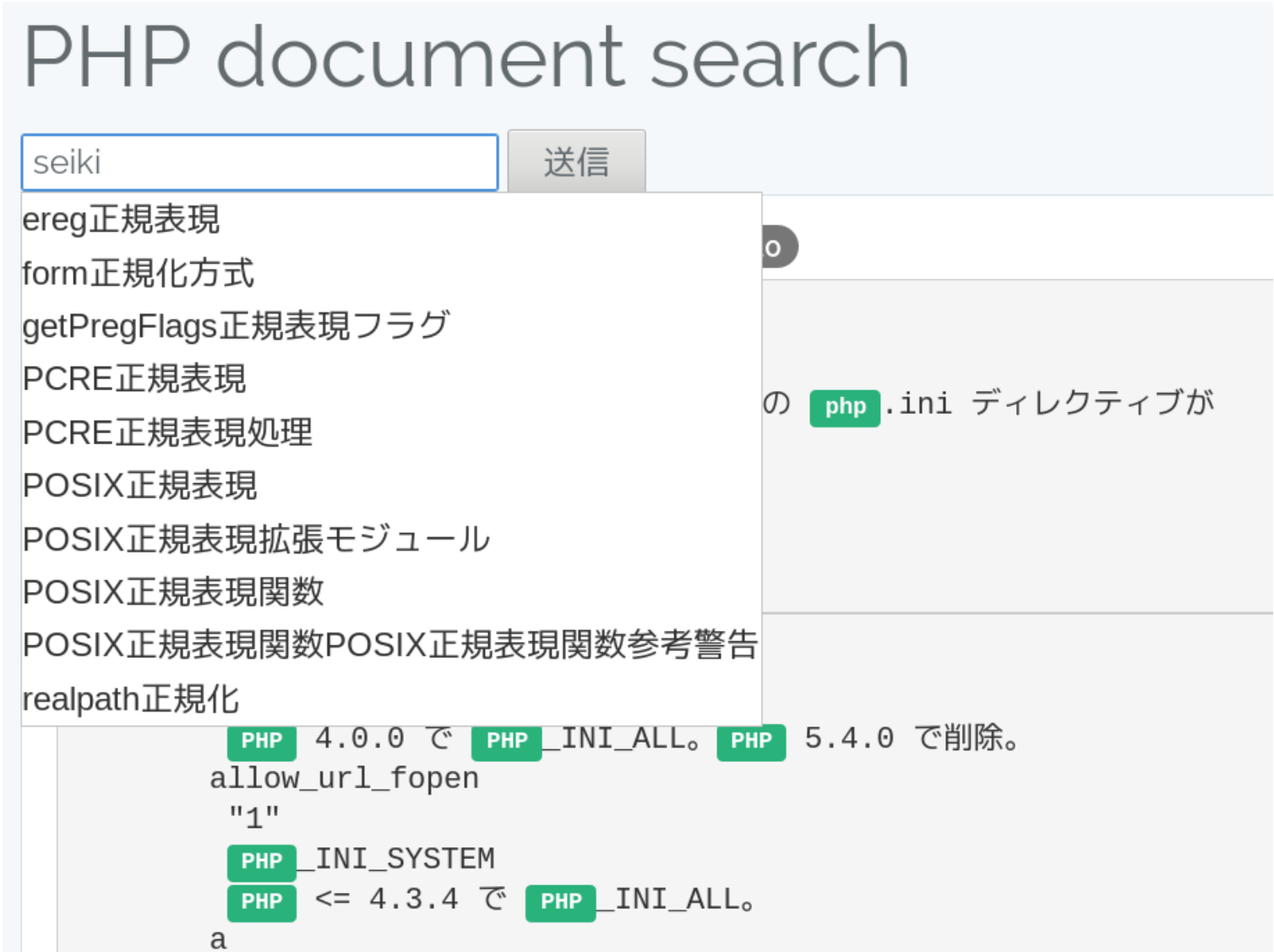
周辺テキスト

```
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&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>！
</div>
<div class="snippet">...</div> ←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

オートコンプリート：テーブル定義

```
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;
```

オートコンプリート：データ例

```
INSERT INTO terms VALUES (
  '牛乳', -- 補完候補
  'ギュウニュウ' --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
);
```

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
```

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 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
```

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 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
```

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 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('ギュウ' AS script),
                     ')') IN BOOLEAN MODE) OR
       MATCH (term) -- 候補を緩く全文検索
     AGAINST (CONCAT('*D+ ', mroonga_escape('ギュウ'))
              IN BOOLEAN MODE)
  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 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
```

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 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
```

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

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);
```

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：Mroonga 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！...

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
- オートコンプリート
- 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：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	651145

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
```

使い方はDocker版と同じ

ChupaText：活用例

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

- MariaDBの全文検索まわり
- 全文検索システム実装例を紹介
- 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