

# MySQL+Sennaによる 日本語全文検索

住商情報システム(株) 池田徹郎  
2007年3月16日 @ OSC2007 Spring/Tokyo

# 自己紹介

- 日本MySQLユーザ会 所属
- 住商情報システム(株) 所属
- MySQL, Incにて開発者インターン(昨年度)
- Sennaプロジェクトへ参加(半年くらい前)
  - MySQLバインディングパッチの保守・開発

# 本日の内容

1. MySQL+Sennaの背景
2. MySQL+Sennaの特徴
3. MySQL+Senna性能検証
4. MySQLバイインディング

# 1. MySQL+Sennaの背景

# RDBMSにおける全文検索

- 特定のキーワードを含む文書を検索すること
  - 長い文書を格納しているテーブルで使う
  - VARCHARやTEXTカラムに対する検索
- Btree/Hashインデックス検索とは異なる
  - これらは数値型/時間型、短い文字列型向け
- 全文検索 ≠ 部分一致検索
  - `SELECT ... WHERE MATCH(col) AGAINST('hoge')`
  - `SELECT ... WHERE col LIKE '%hoge%'`
  - 部分一致検索はテーブルスキャンが発生するため遅い

# MySQLにおける全文検索の現状

- 日本語の場合、入力データを事前に分ち書きしておかない限り、検索結果が0件になる。

```
C:\WINDOWS\system32\cmd.exe - mysql -uroot -p
mysql> CREATE TABLE t2 (c1 TEXT, FULLTEXT (c1)) ENGINE = MyISAM;
Query OK, 0 rows affected (0.06 sec)

mysql> INSERT INTO t2 VALUES ('I have a pen.'),
-> ('Please tell me why you went there.'),
-> ('Have a nice day, see you.'),
-> ('May I help you?'),
-> ('I bought a pen yesterday.');
```

c1
I have a pen.
I bought a pen yesterday.

```
Query OK, 5 rows affected (0.02 sec)
Records: 5 Duplicates: 0 Warnings: 0

mysql> SELECT c1 FROM t2 WHERE MATCH(c1) AGAINST ('pen');
+-----+
| c1    |
+-----+
| I have a pen. |
| I bought a pen yesterday. |
+-----+
2 rows in set (0.00 sec)

mysql>
```

```
C:\WINDOWS\system32\cmd.exe - mysql -uroot -p
mysql> CREATE TABLE t1 (c1 TEXT CHARSET cp932, FULLTEXT (c1)) ENGINE = MyISAM;
Query OK, 0 rows affected (0.06 sec)

mysql> INSERT INTO t1 VALUES ('今日は良い天気ですね。'),
-> ('明日も晴れるといいですね。'),
-> ('去年の夏は暑かったです。'),
-> ('台風が接近しているらしいですよ。'),
-> ('今年の夏も暑いらしいですよ。');
```

```
Query OK, 5 rows affected (0.00 sec)
Records: 5 Duplicates: 0 Warnings: 0

mysql> SELECT c1 FROM t1 WHERE MATCH(c1) AGAINST ('夏');
Empty set (0.02 sec)

mysql>
```

# そこでSennaを使う

- 組み込み用全文検索エンジン Senna
- 手軽、高精度、高速
- MySQLやPostgreSQL、Ruby等に対応
- (有) 未来検索ブラジルさんが開発し、LGPLで公開
- <http://qwik.jp/senna/>
- 最新バージョンはSenna 1.0.1
- 対応文字コード
  - utf8 / sjis / eucjp / latin1 / koi8r

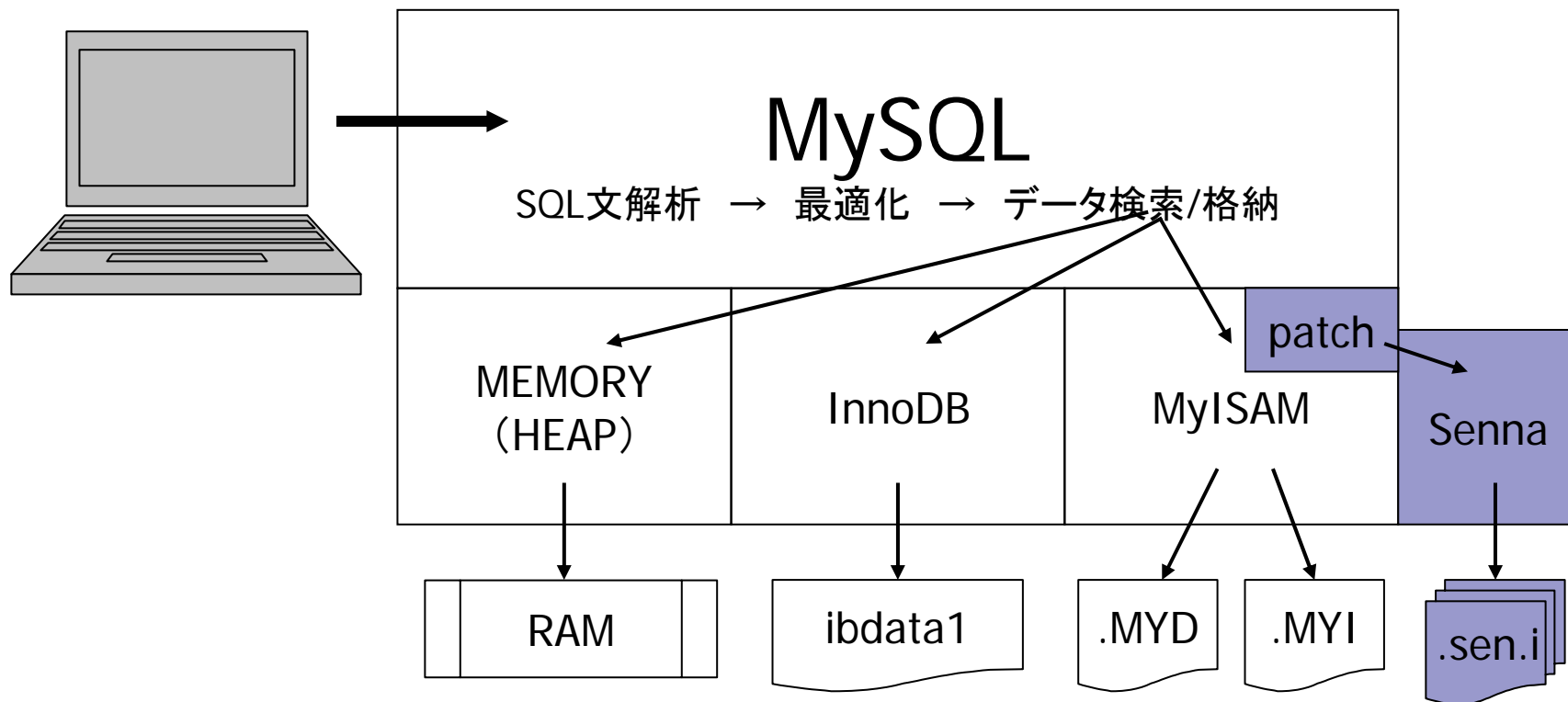
## 2. MySQL+Sennaの特徴

# 特徴

- MySQLがアプリケーションからSennaを完全に隠蔽するため、扱いが容易
- さまざまな検索方法を提供（多機能）
- 高速（特に日本語環境）

# MySQL+Sennaアーキテクチャ概要

- FULLTEXTインデックスを作るとMyISAMの代わりにSennaのインデックスを使うように変更



# MySQL+Sennaによる検索機能

## ■ 自然言語検索

- SELECT ... WHERE MATCH(col) AGAINST('hoge')

## ■ boolean modeによる検索

- SELECT ... WHERE MATCH(col)  
AGAINST('+hoge -fuga' IN BOOLEAN MODE)

- サポートしている演算子

- +, -, OR, ~, <, >, \*, “, ()

- その他、近傍検索、関連文書検索、等々

# インデックスのキーの切り出し方法

- mecab: 形態素解析エンジンMeCabを利用
  - 明日も晴れの予想です  
→明日 / も / 晴れ / の / 予想 / です
- ngram: N=2(2文字単位)のN-gram方式
  - 明日も晴れの予想です  
→明日/日も/も晴/晴れ/れの/の予/予想/想で/です
- delimited: 半角スペース区切り
  - あらかじめ分かち書きされているデータを処理

# MySQL+Senna拡張DDL構文

```
CREATE TABLE tbl_name (  
  col_name TEXT,  
  FULLTEXT INDEX idx USING  
    [ ngram | delimited ]           /* 省略した場合にmecab */  
    [, normalize | no normalize ]   /* デフォルトは normalize */  
    [, ${initial_n_segments} ] (col_name) /* デフォルトは 512 */  
) ENGINE = MyISAM;
```

例: CREATE TABLE t1 (c1 TEXT, FULLTEXT INDEX idx USING  
ngram, 256) ENGINE = MyISAM DEFAULT CHARSET utf8;

# 完全転置インデックスを採用

## ■ 完全転置インデックスって何？

- キーワードに対して、それを含むレコードがどれかだけでなく、レコード中のキーワードの位置情報も記録するインデックス実装方法

## ■ それ何が良いの？

- 複数形態素(フレーズ)による検索が速い・正確
- 日本語は複数形態素による検索が多い
  - “明日の天気” → 明日/の/天気 (mecabの場合)
  - “SCS” → SC/CS (ngramの場合)

## 2ind機能（インデックスマージ補完）

- MySQLでは、FULLTEXTインデックスを用いた場合、他のインデックスが利用できないため、以下のような問題が生じます。
  - LIMITが遅い
  - COUNT(\*)が遅い
  - 他条件でさらに絞り込むと遅い
  - 他条件でORDER BYすると遅い
- インデックスを内部で複数扱って高速化

# 3. MySQL+Senna性能検証

# 性能検証: テスト環境

- 機種: HP ProLiant 385
- CPU: AMD Dual Core Opteron 2.6GHz x2 (1MB L2 cache x2)
- RAM: DDR-SDRAM ECC PC3200 8GB
- DISK: SAS 2.5inch 10000rpm 72GB x4 (4本でRAID 0を構成)
- RAID: BBWC 256MB
- OS: Miracle Linux 4.0 (Asianux 2.0) / Linux kernel 2.6.9-11AXCustom
- アーキテクチャ: x86\_64 (EM64T/AMD64)
- MySQL: ver5.0.34
  - チューニング: myisam\_sort\_buffer\_size=1GB、key\_buffer\_size=1GB
- Senna: ver1.0.1
  - チューニング initial\_n\_segments=2048 (更新バッファサイズ=512MB)
- テストデータ: 日本語版Wikipediaのダンプデータ
  - テーブルサイズ 1.4GB、約57万行、1行あたり平均2,300バイト
- 負荷ツール: MySQL C APIを使った独自Cプログラムにて別マシンから

# 性能検証：インデックスの構築

- テーブルを作成してデータを格納した後、以下のSQL文を実行。  
create fulltext index ft using ngram, normalize, 2048 on t1 (c2);
- 結果（データ量は57万行、1.4GB）

index_type	要した時間	スループット	SENファイルの サイズ合計	SENファイル内の 語彙数
mecab	18分22秒	513.7行/秒	1.47GB	417万語
ngram	14分32秒	649.2行/秒	2.13GB	523万語

- 補足事項
  - myisam\_sort\_buffer\_sizeとinitial\_n\_segmentsを大きくすることで高速化が可能。
  - myisam\_repair\_threadsは意味無し。

# 性能検証：検索速度 SELECT

- mecab、ngram、インデックス無しのテーブルに対して、それぞれ10接続同時にラッシュ検索を2分間実行。キーワードは事前にIPA辞書から抽出した約6万個の名詞の中からランダムで選んでおいたものを使用。
  - SELECT COUNT(\*) FROM t1 WHERE MATCH(c2) AGAINST(“キーワード”);
  - SELECT COUNT(\*) FROM t1 WHERE c2 LIKE ‘%キーワード%’;
  - mecab/ngramはmatch検索、インデックス無しではLIKE検索を実施
- 結果（複数回計測した値の平均値）

index_type	スループット	平均レスポンス	相対値
mecab	3190.7 クエリ/秒	3.1 ミリ秒	10293 倍
ngram	1235.3 クエリ/秒	8.09 ミリ秒	3985 倍
LIKE演算子	0.31 クエリ/秒	31.2 秒	1

- 補足
  - 同時実行数を増やすと、CPUコア数まではスケールする。
  - 同時実行数を増やすと、その分だけ、平均レスポンスは低下。

# 性能検証：更新性能 INSERT

- mecab、ngram、インデックス無しのテーブルに対して、それぞれ10接続同時にラッシュ更新を2分間実行。入力データは別テーブルに用意しておいた日本語版Wikipediaの記事データからランダムで選んでおいたものを使用。
  - INSERT INTO t1 (c2) SELECT c2 FROM t1\_copy WHERE c1 = キー;

- 結果（複数回計測した値の平均値）

index_type	スループット	平均レスポンス	相対値
mecab	246.7 行/秒	43.7 ミリ秒	0.38 倍
ngram	254.3 行/秒	40.5 ミリ秒	0.40 倍
インデックス無し	641.2 行/秒	15.6 ミリ秒	1

- 補足

- 平均行長が約2300バイトなので、そもそも重たいINSERT。
- MyISAMでは更新時に暗黙的にテーブルロックを行うため、同時に処理されるのは1接続のみ。従って、1 CPUしか使用されない。

# 性能に関する過去メモ

IA-32 vs EM64T/AMD64	総じて64bitの方が速い。
Linux vs Windows	以前はWindows上ではLinuxと比べると非常に遅かったが、改良した。計測待ち。
データ量が増えると、、、	データがN倍に増えると、性能は1/N以下に。
ハードウェアの選定	30万円→60万円→120万円クラスのそれぞれのサーバでは、掛けたお金の倍率以上のスケールアップが！
同時実行数を増やすと、、、	検索はCPUコア数までスケール、更新はスケールしない。同時実行数を増やしまくとスループットが増えないのでレスポンスが悪化。

# 4. MySQLバインディング

# Tritonnプロジェクト

- Sennaのサブプロジェクト
- MySQLバインディングパッチを保守・開発
- 今日、誕生しました
- MySQLユーザ向けの機能/性能を強化していくことが目的
- ライセンスはLGPL
- <http://qwik.jp/tritonn/>

# 文字コード対応の強化

- 従来は/var/senna/senna.confに文字コードを書いていた。
  - MySQLとは別に設定が必要だった。
  - サーバ単位でしか文字コードを設定できなかった。
- tritonnでは、CREATE TABLE時の文字コード定義情報を使用してSennaのインデックスを構築する。
  - 設定ファイル不要。通常のMySQLと同じ感覚。
  - テーブル/カラム単位の文字コードに対応可能。
  - CREATE TABLE t1 (c1 INT PRIMARY KEY,  
c2 TEXT, FULLTEXT INDEX idx(c2))  
**DEFAULT CHARSET utf8 ENGINE = MyISAM;**

# ログ機能の強化

- 従来は/var/senna/logディレクトリの有無でログのOn/Offを制御していた。出力レベルも固定だった。
- tritonnでは、以下の2つのMySQLサーバ変数を追加。my.cnfにてログのOn/Offの制御が可能、出力レベルはSETコマンドで動的に変更可能。
  - --senna-log [= filename]
  - --senna-log-level=[ DEBUG | INFO | ...etc ]

# 新しい管理用SQLコマンド

- 従来は”SHOW TABLE STATUS” にSennaの統計情報が追記される形になっていた。
  - mysql-testテストスイートの失敗の原因
  - 複数インデックスを持つテーブルでデータが混在
- tritonnでは以下の新しいSQLコマンドを追加し、show table statusでの情報取得を廃止。
  - SHOW SENNA STATUS  
[ FROM 'db\_name' ] [ LIKE 'tbl\_pattern' ]

# SHOW SENNA STATUS実行例

```
[lab] create fulltext index ft using ngram, normalize, 2048 on t1 (c2);  
Query OK, 566109 rows affected (14 min 31.99 sec)  
Records: 566109 Duplicates: 0 Warnings: 0
```

```
[lab] show senna status¥G
```

```
***** 1. row *****
```

```
      Table: t1  
      Key_name: ft  
      Column_name: c2  
      Encoding: utf8  
      Index_type: NGRAM  
      Normalize: ON  
      Split_alpha: OFF  
      Split_digit: OFF  
      Split_symbol: OFF  
      Initial_n_segments: 2048  
      Senna_keys_size: 565887  
      Senna_keys_file_size: 21045248  
      Senna_lexicon_size: 5230838  
      Senna_lexicon_file_size: 159457280  
      Senna_inv_seg_size: 603983872  
      Senna_inv_chunk_size: 1443106816
```

# 2indパッチの統合

- 従来は2ind機能は2indパッチとしてSennaパッチとは別にリリースされていた。patch実行時に使う/使わないを選択していた。
- tritonでは2indパッチはSennaパッチに統合され、以下の変数をmy.cnfあるいはSETコマンドで変更することで制御可能となっている。
  - `--senna-2ind [= ON | OFF]`

# バグの修正

- 従来のMySQLバインディングに含まれていたバグを修正
  - ALTER TABLE等のテーブルの再作成を行う場合にフラグ情報が消失していた問題
  - 不要な関数呼び出しが原因の性能劣化問題

# MySQL+Senna (Tritonn)使い方

- 事前にSennaやMeCabを入れておく
- MySQLのソース配布版を入手
- patchコマンドでパッチを当てる
- MySQLをビルドする
- 後は通常のMySQLの使い方と同じ

# ビルド例

```
#!/bin/sh  
PREFIX=/usr/local/mysql
```

```
libtoolize -c -f  
aclocal-1.9  
autoheader  
automake-1.9 -c -a -i  
autoconf  
touch sql/sql_yacc.yy
```

```
./configure ¥  
--prefix=$PREFIX --localstatedir=$PREFIX/data ¥  
--libexecdir=$PREFIX/bin --enable-thread-safe-client ¥  
--enable-local-infile --enable-asmblr ¥  
--with-pic --with-fast-mutexes --disable-shared ¥  
--with-zlib-dir=bundled --with-big-tables ¥  
--with-yassl --with-readline --with-archive-storage-engine ¥  
--with-blackhole-storage-engine --with-example-storage-engine ¥  
--with-federated-storage-engine --with-innodb ¥  
--with-extra-charsets=complex --with-senna --with-mecab
```

```
make  
sudo make install
```

# 品質管理について

- MySQL本体に付属しているmysql-testテストスイートをカスタマイズしたものを使用して回帰テストを実施中。
- これによりMySQLオリジナルとの相違が分かるので、それを適宜プロジェクトホームページに情報を記載する予定。

# 現状の制約と課題

- mecabを使う場合はmecabビルド時の文字コードに限定される
- IPA辞書 (mecab-ipadic)はライセンスが不透明なため、商用利用に懸念。
- 2ind機能がまだβ版
- 対応エンジンがMyISAMのみ
- 対応MySQLバージョンが5.0系のみ

# Tritonnの今後の展開

- MySQL 5.0以外 (ver4.0/4.1/5.1) への対応
- 2ind機能の強化と安定化
- その他、不要なI/O処理を減らせるようにする改良など

# 参考URL、連絡先

## ■ Tritonnプロジェクト

- <http://qwik.jp/tritonn/>

- メーリングリストはSennaに間借り中

## ■ Sennaプロジェクト

- <http://qwik.jp/senna/>

- [senna-dev@sourceforge.jp](mailto:senna-dev@sourceforge.jp)