

MySQLに日本語全文検索 機能を組み込む方法

MySQL Users Conference Japan 2007

住商情報システム 池田徹郎

ベンチマーク測定結果

	スループット(qps)	平均レスポンス(ms)
A	0.31	31200
B	1235.3	8.1

AとBの差・・・約4,000倍?

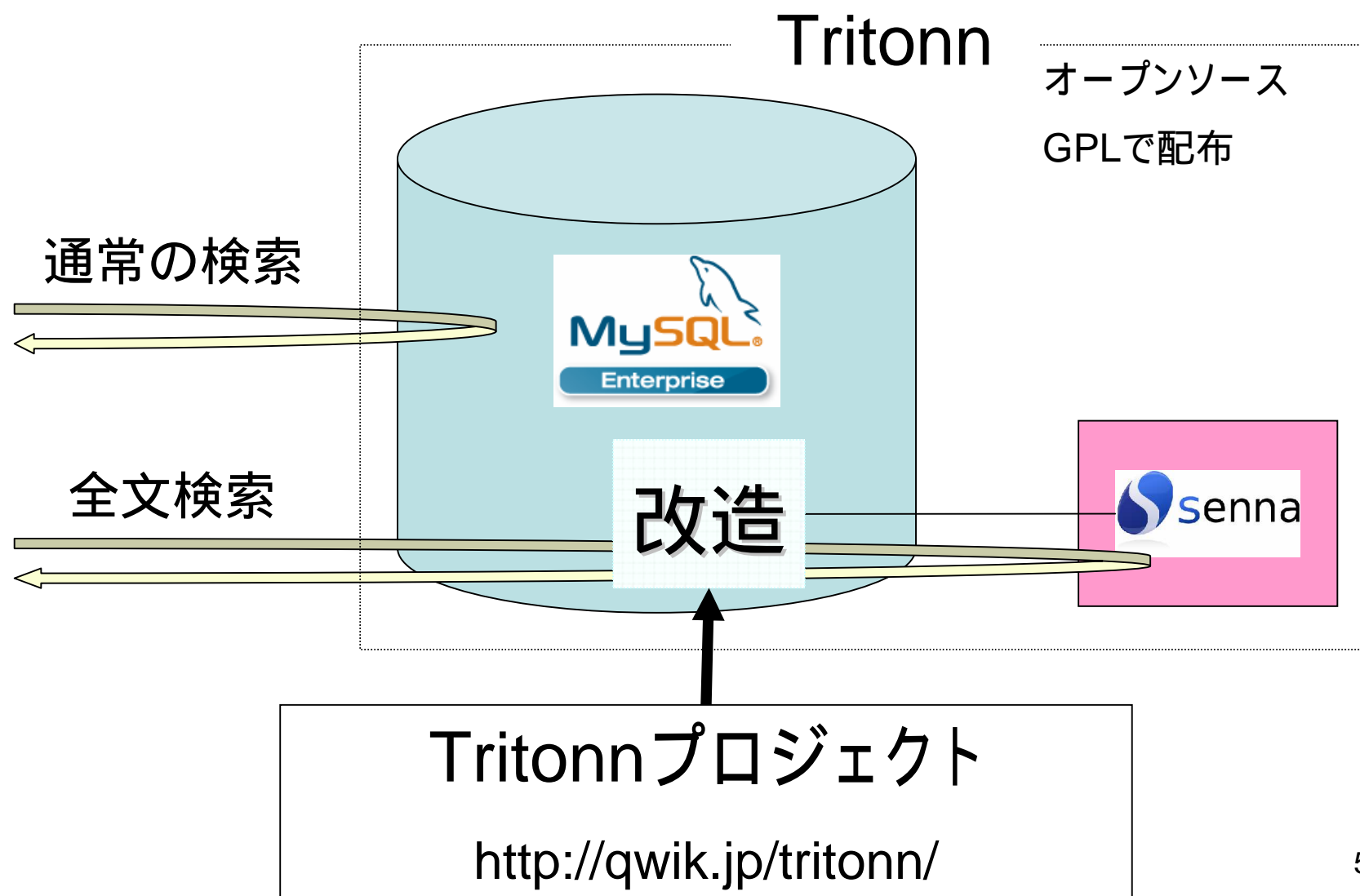
ベンチマーク測定内容

パターンA	LIKE演算子+ワイルドカードによる部分一致検索。
パターンB	Tritonn(MySQL+Senna)を使用した、MATCH...AGAINSTによる全文検索。
テスト内容	日本語版Wikipediaの記事データ 1.4GBをMySQLのテーブルに格納し、ランダムに選択した日本語の名詞を使用してキーワード検索。
マシンスペック	HP ProLiant DL385 CPU Opteron 2.6GHz (Dual Core * 2CPU), 8GB RAM BBWC 256MB Raid 0, SAS 10krpm 72GB * 4

今日の内容

- Tritonnプロジェクト
- MySQL Enterprise + Senna
- コンセプト
- 機能一覧
- どのように実装しているのか
- ビルドと品質保証についてのTips

Tritonnプロジェクト



MySQL Enterprise + Senna

- 住商情報システム (Tritonn開発者の所属企業) が有償サポートを提供
- TritonnのMySQL Enterprise Server版、MySQL, Inc.による公式認定を受けている
- お問い合わせ先: oss-sales@scs.co.jp
- 住商情報システムの展示ブース

Tritonnのコンセプト

- 高速な検索
 - 大規模システムでも高速に日本語の検索が行える
- 簡単に使える
 - MySQLを使ったことがある人であれば、MySQL+Sennaも楽に使いこなせる
 - SQLコマンドだけで使えるためアプリからみるとMySQL(+Senna)は透過的

インデックスの作成

- CREATE TABLE、CREATE INDEX、ALTER TABLEなどでSennaのインデックスを作成できる

```
CREATE TABLE article (  
  article_id INT PRIMARY KEY,  
  title VARCHAR(50),  
  body TEXT)  
DEFAULT CHARSET utf8 ENGINE = MyISAM;
```

```
CREATE FULLTEXT INDEX ft_idx  
USING SECTIONALIZE ON article (title, body);
```

作成するインデックスの細かい設定についてもSQL文で行える。
MySQLのSQL構文を拡張した。

データの追加、更新、削除

- 通常のINSERT/UPDATE/DELETEと同じ

```
INSERT INTO article VALUES (1, “起動確認”,  
“mysql_install_dbを実行してMySQLのインストールを行った  
ら、mysqldを起動します。”);
```

```
UPDATE article SET title = “起動確認(下書き)” WHERE  
article_id = 1;
```

```
DELETE FROM article WHERE id = 1;
```

データ更新が発生した時点で、インデックスも逐次更新する。

全文検索

- 通常のMATCH...AGAINST検索が可能

```
SELECT * FROM article WHERE  
MATCH(title, body) AGAINST("インストール");
```

```
SELECT * FROM article WHERE  
MATCH(title, body) AGAINST  
("*D+ 起動 インストール" IN BOOLEAN MODE);
```

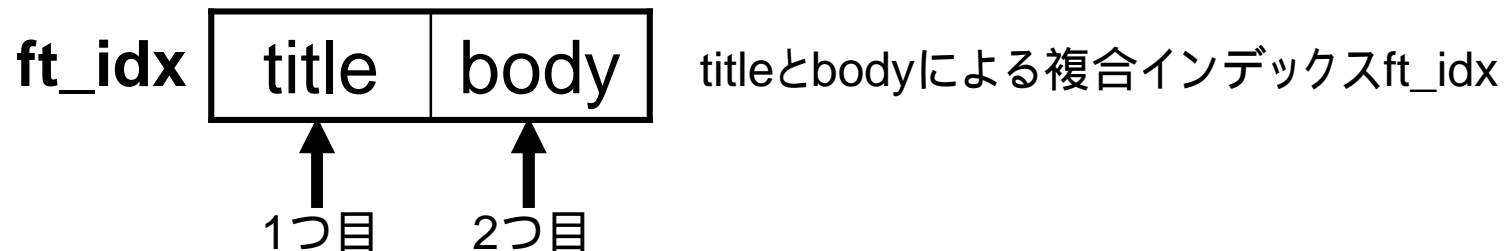
BOOLEAN MODE検索では、Sennaがサポートしている全ての演算子を利用可能。

マルチセクション対応

- 複合インデックス内の各カラムに対して、検索対象のフィルタリングとスコアの重み付けができる

```
SELECT * FROM article WHERE  
MATCH(title, body) AGAINST  
("W1:5,2:1 起動 インストール" IN BOOLEAN MODE);
```

1つ目のセクションのスコアを5倍、2つ目のスコアを1倍に設定。
W演算子を使う。



例えばbodyのみで検索させることも可能。無駄なインデックスが無くなる。

KWIC表示機能

- KWIC (Keyword in Context)表示を行うためのSQL関数を追加

```
SELECT kwic(文書, 作成文字列の最大バイト数, 最大個数, html  
エンコーディングの有無, 全体の開始タグ, 全体の終了タグ, 単語1,  
単語1の前につけられるタグ, 単語1の後につけられるタグ, 単語2,  
単語2の前につけられるタグ, 単語2の後につけられるタグ, ...);
```

- MySQLのnativeなSQL関数として実装しており、高速。
- Prepared Statementからも利用できる。

kwic関数の使用例

```
[test] > select kwic(c1,300,3,0," , " ,"テーブル"," , " ... ") as result  
from t1 where match(c1) against("テーブル")¥G
```

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

result: あずテーブル ... オプション"row_format=redundant"をつけてテーブル ... を作成します。今回のテーブル ... には主キーが設定されているのでDB_ROW_IDなる名前のシステムカラム (通称 RowID) は使用されません。主キーを持たないテーブル ... を
CREAT

対象キーワードに"テーブル"を指定して、Googleの検索結果をまねてKWIC表示させてみた例。

その他の機能

ログ出力	独自のログ出力を管理する機能。my.cnfで設定ができる。SETコマンド(SQL文)で動的にログの出力レベルの変更もできる。
インデックス情報	新しいSQLコマンド”SHOW SENNA STATUS”によりインデックスの定義情報、統計情報を取得することができる。
正規化機能	Unicode正規化(nfkc)を行える。半角と全角の英数字を同じ文字として処理したりとか。
2ind機能	複数のインデックスを同時に活用する機能。

Tritonn (コミュニティ向け) の使い方

ソースコードをダウンロード	https://sourceforge.jp/projects/tritonn/files/
準備	Sennaをインストールしておく。
ビルド	configure時に"--with-senna"を指定する。MeCabによる単語インデックスも使う場合は"--with-mecab"も指定する。
インストール、起動と停止	MySQLと同じ。mysql_install_dbで初期化。mysqld_safeで起動、mysqladminで停止。

TritonnによるMySQLの改造

- DDL構文を拡張
- MyISAMのFULLTEXT処理を置換
- 新しいISQLコマンドを追加
- 新しいシステム変数を追加
- 新しいISQL関数を追加

DDL構文を拡張

- “NGRAM”、“SECTIONALIZE”などのシンボルの追加、パーサの処理の追加、KEY構造体、HA_CREATE_INFO構造体などにメンバを追加して、CREATE INDEX時の追加情報を読み取れるように改変。
 - sql/lex.h
 - sql/sql_yacc.yy
 - sql/structs.h
 - sql/handler.h

```
opt_senna_list:
    opt_senna_item
    | opt_senna_item ',' opt_senna_list ;

opt_senna_item:
    SENNA_SYM {
#ifdef ENABLE_SENNA
        Lex->senna_flags &= ~SEN_DISABLE_SENNA;
#endif /* ENABLE_SENNA */
    }
    | NO_SYM SENNA_SYM {
#ifdef ENABLE_SENNA
        Lex->senna_flags |= SEN_DISABLE_SENNA;
#endif /* ENABLE_SENNA */
    }
    | SENNA_NORMALIZE_SYM {
#ifdef ENABLE_SENNA
        Lex->senna_flags |= SEN_INDEX_NORMALIZE;
#endif /* ENABLE_SENNA */
    }
    | NO_SYM SENNA_NORMALIZE_SYM {
#ifdef ENABLE_SENNA
```

MyISAMのFULLTEXT処理を置換

```
#ifdef ENABLE_SENNA
#define SECTIONALIZE 0x00080000
int ft_sen_index_add(MI_INFO *info, uint keynr,
    const byte *record, my_off_t pos)
{
    if (info->s->keyinfo[keynr].senna_flags & SECTIONALIZE) {
        FT_SEG_ITERATOR ftsi;
        unsigned int section;
        sen_values *values;
        _mi_ft_segiterator_init(info, keynr, record, &fts);
        while (_mi_ft_segiterator(&fts)) {
            if (fts.pos) {
                if (fts.len > 1048576) { SEN_LOG(sen_log_debug,
                    "ft_sen_index_add: fts.len=%d", fts.len); }
            }
        }
        _mi_ft_segiterator_init(info, keynr, record, &fts);
        while (_mi_ft_segiterator(&fts)) {
            if (fts.pos) {
                section = fts.num + 1;
                values = sen_values_open();
                sen_values_add(values, fts.pos, fts.len, 0);
                sen_index_update(info->s->keyinfo[keynr].senna,
                    &pos, fts.num+1, NULL, values);
                sen_values_close(values);
            }
        }
    }
    return 0;
}
```

- テーブル/インデックス作成時に Sennaのインデックスが定義されていたら、Senna APIを使用してインデックス作成。
- 検索/更新時に対象のインデックスがSennaのインデックスだったらSenna APIを使用して検索/更新を実行。
 - sql/ha_myisam.cc
 - myisam/mi_create.c
 - myisam/ft_update.c
 - myisam/ft_boolean_search.c

...他多数

新しいISQLコマンドを追加

- “show senna status”を新しいコマンドとしてパーサに登録
- “show senna status”に対するディスパッチ処理の追加
- “show senna status”を実際に処理する関数を実装
 - sql/sql_show.h
 - sql/sql_show.cc
 - sql/sql_parse.cc
 - sql/sql_yacc.yy

```
#ifdef ENABLE_SENNA
bool senna_show_status(THD *thd, LEX *lex)
{
    List<char> files;
    List<Item> field_list;
    char path[FN_LEN];
    char *file_name;
    Protocol* protocol = thd->protocol;
    char *db = lex->select_lex.db ? lex->select_lex.db : thd->db;
    const char *wild = lex->wild ? lex->wild->ptr() : "%";
    DEBUG_ENTER("senna_show_status");

    (void) my_snprintf(path, FN_LEN, "%s/%s",
        mysql_data_home, db);
    (void) unpack_dirname(path, path);

    field_list.push_back(new Item_empty_string("Table", 20));
    field_list.push_back(new
        Item_empty_string("Key_name", 20));
    field_list.push_back(new
        Item_empty_string("Column_name", 20));
    field_list.push_back(new Item_empty_string("Encoding", 20));
    field_list.push_back(new
        Item_empty_string("Index_type", 20));
}
```

新しいシステム変数を追加

```
{"secure-auth", OPT_SECURE_AUTH, "Disallow
authentication for accounts that have old (pre-4.1)
passwords.",
  (gptr*) &opt_secure_auth, (gptr*) &opt_secure_auth, 0,
  GET_BOOL, NO_ARG,
  my_bool(0), 0, 0, 0, 0, 0},
#ifdef ENABLE_SENNA
{"senna-2ind", OPT_SENNA_2IND, "Enable Senna 2ind-
patch.",
  (gptr*) &global_system_variables.senna_2ind,
  (gptr*) &global_system_variables.senna_2ind,
  0, GET_BOOL, NO_ARG, 0, 0, 0, 0, 0, 0},
{"senna-log", OPT_SENNA_LOG, "Senna log file.",
  (gptr*) &opt_senna_logname, (gptr*) &opt_senna_logname,
  0, GET_STR, OPT_ARG,
  0, 0, 0, 0, 0, 0},
{"senna-log-level", OPT_SENNA_LOG_LEVEL, "Senna log
level.",
  (gptr*) &opt_senna_log_level, (gptr*) &opt_senna_log_level,
  0, GET_STR, REQUIRED_ARG,
  0, 0, 0, 0, 0, 0},
#endif
{"secure-file-priv", OPT_SECURE_FILE_PRIV,
"Limit LOAD DATA, SELECT ... OUTFILE, and
LOAD_FILE() to files within specified directory",
  (gptr*) &opt_secure_file_priv, (gptr*) &opt_secure_file_priv,
  0,
  GET_STR_ALLOC, REQUIRED_ARG, 0, 0, 0, 0, 0, 0},
```

- セッション変数を定義する構造体に新しいシステム変数を追加
- my.cnfで扱える変数を定義する構造体に新しいシステム変数を追加
- “SHOW VARIABLES”の対象となるように配列に追加
- “SET”コマンドで変更するためのコールバック関数を実装
 - sql/set_var.h
 - sql/set_var.cc
 - sql/mysql_priv.h
 - sql/mysqld.cc

新しいISQL関数を追加

- “kwic”を新しいISQL関数名としてパーサに追加
- kwic用のクラスを定義 (Itemクラスを継承)
- kwicの実際の処理を行う関数を実装
 - sql/item_strfunc.h
 - sql/item_strfunc.cc
 - lex.h
 - sql/sql_yacc.yy

```
#ifdef ENABLE_SENNA
class Item_func_senna_kwic :public
Item_str_func
{
public:
Item_func_senna_kwic(List<Item>
&list) :Item_str_func(list) {}
Item_func_senna_kwic(Item *a,Item
*b) :Item_str_func(a,b) {}
String *val_str(String *);
void fix_length_and_dec();
const char *func_name() const { return
“kwic”; }
};
#endif /* ENABLE SENNA */
```

(triton-1.0.5で実装した)

ビルドと品質保証についてのTips(1)

- configureオプションをどうしたらいいか悩んだら、バイナリ配布版のbin/mysqlbugを参考に
にする

```
COMP_ENV_INFO="CC='ccache gcc' CFLAGS=" CXX='ccache gcc' CXXFLAGS=" LDFLAGS="
ASFLAGS=""
CONFIGURE_LINE="./configure '--prefix=/usr/local/mysql' '--localstatedir=/usr/local/mysql/data' '--libe
xecdir=/usr/local/mysql/bin' '--with-comment=MySQL Community Server (GPL)' '--with-server-suffix=' '--e
nable-thread-safe-client' '--enable-local-infile' '--enable-asm' '--with-pic' '--with-fast-mutexe
s' '--with-client-ldflags=-static' '--with-mysqld-ldflags=-static' '--with-zlib-dir=bundled' '--with-bi
g-tables' '--with-yassl' '--with-readline' '--with-archive-storage-engine' '--with-blackhole-storage-en
gine' '--with-ndbcluster' '--with-csv-storage-engine' '--with-example-storage-engine' '--with-federated
-storage-engine' '--with-innodb' '--with-extra-charsets=all' 'CC=ccache gcc' 'CXX=ccache gcc'"
```

ビルドと品質保証についてのTips(2)

- 複数のCPU(コア)があるマシン上でビルドするときはmakeに”-j”オプションをつける
 - 最大でコア数分の時間短縮に！！
- make後は”make install”ではなく”scripts/make_binary_distribution --no-strip”を実行する

```
./configure --prefix=/usr/local/mysql --with-senna  
make -j  
scripts/make_binary_distribution --no-strip
```

ビルドと品質保証についてのTips(2)

- ビルドが終わったら必ず回帰機能テストを実行する。

```
tar zxf mysql-5.0.45-linux-x86_64.tar.gz
cd mysql-5.0.45-linux-x86_64/mysql-test
./mysql-test-run.pl --force --udiff > test.log
```

- テスト終了後、ログファイルをチェックする。

チェックすべき文字列	何が起きているのか
“[fail]”	failの後に書かれている名前のテストケースが失敗している。
“Lost Connection”	高い確率で、mysqldがテストケース実行中にSIGSEGVで落ちた。

- 完 -

ご清聴ありがとうございました。