SQLの基礎

データベースを扱う言語。

目次

概要

MySQLを使って、SQLの基礎的なコマンドの挙動を一通り確認する。 ほぼ以下の記事に準拠している。大変参考になった。

- https://qiita.com/okamuuu/items/c4efb7dc606d9efe4282

SQLのコマンド

参照

- SELECT テーブル内の一つまたは複数のフィールドの全データを読み込むことができる。同じ数値が繰り返されるとき、それを`DISTINCT`コマンドで絞ることができる。例えばカテゴリー（店舗のある土地など）が絞りたいときは以下のようにする。カテゴリーのuniqueみたいなもの。
- FROM
- DISTINCT
- WHERE
- IN
- INSERT INTO

テーブルの作成

- create
- drop

テーブルはデータベース内の仕組み。テーブルにデータを保存し、テーブルを処理することによりデータを取り出す。

MySQL入門

基本的にテーブルをデータのひとまとまりとして扱う。文末にセミコロンつけるのを忘れずに。

起動・終了

- mysql.server start
- mysql.server stop

MySQLにの対話環境に入る

- mysql -u root -p rootユーザーとして、password(`-p`オプション)を入力してMySQLとの対話環境に入る。

MySQLとの対話

- show databases; データベース一覧
- use mysql; 基本的なデータが格納されているデータベース. `Databases changed`と表示されたら、扱えるデータベースが切り替わっている
- use [database-name]; `show databases`の中にあるデータベースから選択. `Databases changed`と表示されたら、扱えるデータベースが切り替わっている
- show tables; 現在のデータベース(`use [database-name]`で指定したもの)に存在するテーブル一覧が表示される

データベースの扱いの基本

- select [field-name] from [table-name]; テーブルから特定のフィールド名と合致する**列**の抽出. `table-name`は`show tables`で表示されたものの中から選択。
- select * from [table-name];
- select * from [table-name]\G テーブルの中身を全て表示.`\G`とすると少し表示が見やすくなる。
- exit; 対話モードを終了.
- `--`(行末まで) or `/**/`(複数行) コメントは

MySQLへの命令をコマンドラインで実行する

- mysql -u root -p [database-name] -e ['command']

```
$ mysql -u root -p mysql -e 'show tables;'
# クオーテーションはシングルでもダブルでも良い
```

ルートユーザーとして(`-u root`)、パスワードを入力して(`-p`)、特定のデータベースに対して(`database-name`)、コマンドを実行(`-e`オプション)。

- mysql -u root -p [database-name] < command.sql 特定のデータベースに対して、テキストファイルの書かれたコマンドを1行ずつ実行する。テキストファイルであれば拡張子は特に関係ないが、エディタを使用するときに拡張子を`.sql`にするとコマンドの補完が効く。この場合は`-e`オプションは必要ない。
- mysql -u root -p [database-name] 特定のデータベースとの対話モードの開始。

練習用のデータベースを作る

まず対話環境に入る。

- create database [database-name]; 任意の名前でデータベースを作成できる。

テーブルの作成・消去

データベースの中にテーブルを作る。 まず、`use [db-name];`でテーブルを作りたいデータベースに切り替える。

- create table `new-table-name`; 任意の名前でテーブルを作成。 バッククオーテートを使っていることに注意。
- drop table `table-name`; 存在するテーブルの消去。指定したテーブルが存在しないと・エラーになる。あるテーブルを新しく作るときにはエラーを吐き出さないように、もし指定したテーブル名で既存のテーブルが存在したら消去するようなコードを書く。以下を参照

```
/* if exists [table-name] もしテーブルが存在したら*/
drop table if exists [table-name];
create table `table-name`;
/*
下記で説明するがこのコードでは実はtableは作れない。テーブルの作成の仕方は複雑なため下記記事に譲る。
テーブルを作成するときには、
・特定のフィールド（列）のフィールド名・デフォルト値を設定する必要がある。
・デフォルト値は設定しなくても良いが、設定しない場合行の追加を行うときに必ず値を指定する必要がある。指定しないと
Field 'hoge' doesn't have a default value
というエラーになる
*/
```

行を追加・参照・更新・消去

行の追加

- insert into [table-name] (field-name1, field-name2, ...) values(field1-value, field2-value...); フィールド名が一つでも、必ずフィールド名とvaluesはかっこ()でくくる。

```
-- title, statusのcolumnに、それぞれ「料理」、「0」を挿入
insert into todos (title, status) values ("料理", 0);

-- 単独の挿入をした場合はどうなるか
insert into todos (title) values ("ストレッチ");  -- 1つでもかっこ()は必要
/*
デフォルト値を設定してる場合、title以外の列はデフォルト値が入る。
デフォルト値を設定していない場合はエラーになる。
*/
```

行の参照

```
-- カラム（フィールド）の情報の表示
show columns from [table-name];
```

```
-- テーブルの中身を全て表示
select * from [table-name];

-- 特定のフィールド名（列名）を指定して、その列を表示
select [field-name] from [table-name];

-- フィールド名を複数指定して表示
select [field-name1], [field-name2], ... from [table-name];

-- フィールド名のunique表示
-- 一意な要素のみを抽出
select distinct [field-name] from [table-name];
```

条件に一致する行の参照（絞り込み）

```
-- フィールド（列）の要素のうち、hogeに一致する行の抽出
select * from [table-name] where [field-name] = "hoge";

-- フィールド（列）の要素のうち、0に一致する行の抽出
select * from [table-name] where [field-name] = 0;

-- フィールド（列）の要素が3未満の行の抽出
select * from [table-name] where [field-name] < 3;

-- フィールド（列）の要素が3以下の行の抽出
select * from [table-name] where [field-name] <= 3;

-- フィールド（列）の要素が2以上3以下の行の抽出
select * from [table-name] where 2 <= [field-name] and[field-name] <= 3;

-- フィールド（列）の要素が1以上2以下、または4以上の行の抽出
-- 論理演算はandとorを使う
-- 複数の条件をまとめるときは()でくくる
select * from [table-name] where (1 <= [field-name] and [field-name] <= 2)
or 4 <= [field-name];

-- フィールド（列）の要素が1以上2以下、かつ4以上の行の抽出
-- この条件を満たすものはない
select * from [table-name] where (1 <= [field-name] and [field-name] <= 2)
and 4 <= [field-name];

-- INを使った行の参照
-- 特定のフィールド(field-name)が"hoge1", "hoge2"の行を抽出
select * from [table-name] where [field-name] in ("hoge1", "hoge2");
-- e.g)
SELECT * FROM Store_Information WHERE Store_Name IN ('Los Angeles', 'San
Diego');
```

行の更新

- update [table-name] set [field-name1] = value1 where [field-name2] = value2; whereをつけないと指定したフィールドすべてが変更されてしまうため注意。

```
-- 特定の行の特定の列の値を更新
-- 条件を満たしたフィールド（特定の行）の、特定のフィールド（列）の値を更新
-- e.g.)
-- idが3の行のフィールドstatusの値を1に変更
update todos set status = 1 where id= 3;
```

`where id = 3` を忘れてしまうと全部の行が更新されるので注意。最初に select 文で where区を確認してからそのクエリの where 区に構文を足すようにすると間違いがない。

行の消去

```
-- フィールド "id" が2の行を消去
delete from [table-name] where id = 2;  -- 行の消去
```

whereを設定しないとがtableがemptyになってしまうため必ずwhereで条件を設定すること。

バックアップとリストア

dump.sqlにこれまでの作業を保存する。

```
$ mysqldump -u root -p practice_db > dump.sql
$ cat dump.sql
```

これでバックアップが取れた。 対話環境に入り、間違えてテーブルの要素を消去してみる。`where`をつけずに`delete`を行う。

```
delete from todos;  -- 間違えてテーブルの要素を消去してみる。whereをつけずにdelete
select title from todos;  -- empty になってしまった
```

もう一度ターミナルに戻る。先ほどのバックアップからテーブルを復元する。

```
$ mysql -u root -p practice_db < dump.sql  # 標準入力
```

テーブルが復元されているかどうか確認（対話環境に戻っても良い）

```

```

```
$ mysql -u root -p practice_db -e "select * from todos;"
```

ちゃんと復元できていることが確認できる。

pagerの設定

`select`した時の表示の形式を変更する。最初は気にしなくて良いが頭の片隅に置いておくと良い。

```
pager less;
select * from mysql.user;  -- 十字キーでテーブルを移動できる別の画面に移動する
```

`less -S`をやめる時

```
nopager
select * from mysql.user; -- pagerが治ってる
```

表示形式をCSVにする

MySQLからデータをCSVとして抽出。以下の記事を参照。

- http://qiita.com/tasmas256/items/ec7e23278ee2b40aad79 またはMySQL Workbench(GUIツール)を使うと良い。

終わりに

SQLに関して簡単にまとめてみました。他のデータベース(MongoDB, PostgreSQL etc..)に関しての知識は皆無ですが、一つのデータベースを扱えるようになったら他のも入りやすいと思うので（プログラミング言語と一緒ですかね？）当面はMySQLを扱い、一通り操作を習熟することに努めます。
　また、クローリング・スクレイピングって様々な知識がないといけないんだなというのを改めて実感しています。データ分析の一連の流れ（収集・前処理・統計処理・可視化・アウトプット）をできるってのは本当に大変。WEB・統計・DB・サーバーなどなど総合力が要求される。世の中のデータ分析界隈の方々は本当にすごいな。これからもぼちぼちやっていくので温かい目で見守ってください。　記事中の間違い、またはご指摘等ございましたらコメントしていただけると幸いです。最後まで読んでくださりありがとうございました。

参考記事

参考にさせていただいた記事

- https://qiita.com/okamuuu/items/c4efb7dc606d9efe4282
- https://qiita.com/kenju/items/ad1550882565bad188d2

InnoDB：データベースエンジン（ストレージエンジン）

データの読み書きを行う基盤のことを**ストレージエンジン**という。 MySQL5.5以降でのデフォルトのストレージエンジンはInnoDB。 主なものに、InnoDBとMyISAMがある。

トランザクション ・・・複数のSQLクエリを1つの作業としてまとめたもの。 一連の手順が正常に終われば、作業は完了。しかし、 途中でトラブルが発生し、処理が中断することがあるかもしれない。 いずれかのステップで失敗した場合はロールバックしたい。 そこで登場するのがトランザクション。一連の処理をトランザクションとしてまとめておくと、いずれかのステップで失敗した時にロールバックを行ってくれる。 InnoDBにはトランザクション機能がある。 MyISAMにはない。

csvファイルの読み込み

手順としては、

1. テーブルの作成 csvのカラムを使ってテーブルの作成
2. csvの読み込み(load data local infile [file-path])

```
-- make players
drop table if exists players;
create table `players`(
    `id` int(11) not null auto_increment,
    `name` varchar(50) not null,
    `goals` int(11) not null,
    `height` int(11) not null,
    primary key (`id`)
) engine=InnoDB default charset=utf8mb4;

load data local infile
"/Users/okuwaki/Projects/Database/article/script/players.csv"
into table players
fields terminated by ','
optionally enclosed by '"';  --これしないとうまくいかなった

-- MySQLサーバー上で
load data local infile
"/Users/okuwaki/Projects/Database/article/script/players.csv" into table
players fields terminated by ',' optionally enclosed by '"';
--ERROR 1148 (42000) at line 11: The used command is not allowed with this
MySQL version
```

が返ってきた。

```
コマンド行クライアントの場合、--local-infile[=1] オプションを指定することによって
LOAD DATA LOCAL を有効にするか、--local-infile=0 オプションを指定することによってこ
れを無効にします。

(中略)

サーバーまたはクライアントのいずれかで LOAD DATA LOCAL が無効な場合、そのようなステー
トメントを発行しようとしたクライアントは次のエラーメッセージを受け取ります。
```

```
ERROR 1148: The used command is not allowed with this MySQL version
```

どうやらセキュリティがらみの問題でデフォルトではできなくなっているよう。 local infileを利用するには、設定で許可されている必要があるよう。 これが許可されていないと、以下のエラーが返ってくる。 ってことで指示通り

```
$ mysql -u scraper -p --local_infile=1;

-- Mysql localを消してみた
load data infile
"/Users/okuwaki/Projects/Database/article/script/players.csv"
into table players
fields terminated by ','
optionally enclosed by '"';
--ERROR 1290 (HY000): The MySQL server is running with the --secure-file-
priv option so it cannot execute this statement
```

このページを見ながらエラーの解決をして見た。 https://dev.mysql.com/doc/refman/8.0/en/load-data-local.html

```
$ cd /usr/local/Cellar/mysql/8.0.11/.bottle/etc
$ cp my.cnf my.cnf.origin  # 怖いのでコピー
$ emacs my.cnf
```

以下のような情報を追加した。

```
# Default Homebrew MySQL server config
[mysqld]
# Only allow connections from localhost
bind-address = 127.0.0.1
これ以下を新しく追加
# Enable load data local
[client]
loose-local-infile=1
```

これでクライアント側のファイルを読み込めるようになるはず。

```
デフォルトオプションの読み込みはどこから行っているか？
Default options are read from the following files in the given order:
/etc/my.cnf /etc/mysql/my.cnf /usr/local/etc/my.cnf ~/.my.cnf
```

環境変数を見ると

```
show global variables like "%infile%";
/* 設定できてないやんけ、なぜ？
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| local_infile  | OFF   |
+---------------+-------+
1 row in set (0.00 sec)
*/

-- もう一度ルートで入って環境変数を書き換えて見る
mysql --local-infile=1 -u root -p  scraping;
show global variables like "%infile%";
/*
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| local_infile  | OFF   |
+---------------+-------+
いやまじでオプションが全く働いていないんだが・・・
*/
set global local_infile = ON;
Query OK, 0 rows affected (0.00 sec)

show global variables like "%infile%";
/*
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| local_infile  | ON    |
+---------------+-------+
1 row in set (0.00 sec)
*/
load data local infile
"/Users/okuwaki/Projects/Database/article/script/players.csv" into table
players fields terminated by ',' optionally enclosed by '"';
/* できたーーー
Query OK, 29 rows affected (0.05 sec)
Records: 29  Deleted: 0  Skipped: 0  Warnings: 0
*/
select * from players;
/*
+----+--------------------+-------+--------+
| id | name               | goals | height |
+----+--------------------+-------+--------+
|  1 | ウィル             |    14 |    178 |
|  2 | タクヤ             |     8 |    181 |
|  3 | マルコ             |    11 |    172 |
|  4 | ケン               |    12 |    169 |
|  5 | マイケル           |    14 |    180 |
|  6 | ショウ             |     9 |    175 |
|  7 | フミヤ             |     3 |    178 |
|  8 | ケヴィン           |    20 |    188 |
```

```
|  9 | テツヤ              |    10 |    165 |
| 10 | ツバサ              |    17 |    176 |
| 11 | カリム              |     4 |    179 |
| 12 | ロベルト            |     2 |    170 |
| 13 | アユム              |     7 |    176 |
| 14 | ルイジ              |    12 |    190 |
| 15 | ペレ                |    19 |    181 |
| 16 | ルーク              |    14 |    176 |
| 17 | トーマス            |    12 |    181 |
| 18 | ベン                |     2 |    188 |
| 19 | ヒカル              |     6 |    170 |
| 20 | ユウヘイ            |     7 |    185 |
| 21 | ブルース            |     7 |    168 |
| 22 | ドメニコ            |    16 |    179 |
| 23 | ダイキ              |     9 |    180 |
| 24 | フェルナンド        |    25 |    186 |
| 25 | フィリップ          |     3 |    172 |
| 26 | エリック            |     1 |    188 |
| 28 | ラファエル          |    17 |    175 |
| 29 | ピーター            |    20 |    179 |
| 30 | カルロス            |    11 |    176 |
+----+--------------------+-------+--------+
29 rows in set (0.00 sec)
*/

describe players;
/*
+--------+-------------+------+-----+---------+----------------+
| Field  | Type        | Null | Key | Default | Extra          |
+--------+-------------+------+-----+---------+----------------+
| id     | int(11)     | NO   | PRI | NULL    | auto_increment |
| name   | varchar(50) | NO   |     | NULL    |                |
| goals  | int(11)     | NO   |     | NULL    |                |
| height | int(11)     | NO   |     | NULL    |                |
+--------+-------------+------+-----+---------+----------------+
4 rows in set (0.01 sec)
*/
```

できた。なぜかコマンドラインオプションによる設定が反映されず、自分で環境変数（グローバル変数？）を書き換えてやっと`load data local infile`できた。これでクライアント側（localhost）側のパスを指定してcsvファイルを読み込むことができた。まだ書き込み時の設定に関しては使いこなせていないので少しずつ使い方を身につけていく。

load data local infile：NULL値の保持

追記：オプションが働かない理由がわかりました

```
select @@local_infile;
show global variables like "%infile%";
で見れるのはサーバー側の設定。
```

そこに、クライアント側が--local-infile=1で入れば、クライアント側でも可能であるということになり、`load data local infile`ができるようになる。