SQL の基礎

データベースのデータ型

- 整数型
- 文字列型
- 日付型 `where`と比較演算子の組み合わせで、特定の日付以降以前を取得できる。

---

in演算子：複数の値に完全一致するデータの取得

あるフィールドの"hoge1", "hoge2"に一致する行が欲しいとする。 このときどうやって取得するか？

```
-- 条件をorでつなげる
select * from my_table where (field = "hoge1") or (field = "hoge2");

-- in演算子を使う（かっこ）は必ず必要
select * from my_table where field in ("hoge1", "hoge2");
```

like演算子：ある文字列を含むデータの取得

`like`演算子を使う。 `like`を使うときには、文字列にワイルドカードを用いる。ワイルドカードとは、どんな文字列にも一致することを指す記号のことである。`like`演算子では、`%`をワイルドカードとして扱う。前方・後方一致も覚えておく

```
-- field1の内、hogeに完全一致する行の抽出
select * from my_table where field1 = "hoge";

-- field1の内、hogeを含む行の抽出
select * from my_table where field1 like "%hoge%";

-- hoge前方一致 hoge以降はなんでもよい
select * from my_table where field1 like "hoge%";

-- hoge後方一致 hoge以前はなんでもよい
select * from my_table where field1 like "%hoge";
```

not演算子：条件の否定

条件を否定する。かっこでくくると`not`が何を否定しているかわかりやすい。

```
-- field1の内、hogeを含まない行の抽出
select * from my_table where not (field1 like "%hoge%");
```

NULLの取得

データベースに何も保存されていない＝`NULL`という。NULLはフィールドと等号で繋げない。つまり`field = null`や`not field = null`はできないため注意。(mysqlでは使えた。。)`= null`では、undefinedなどを含む可能性があるため、正しく判定できない。

```
-- 特定のフィールドがNULLの行を取得
select * from my_table where field is null;

-- 特定のフィールドがNULLでない行を取得
select * from my_table where not (field is null);
select * from my_table where field is not null; -- こっちが普通？
```

and/or演算子：複数の条件の指定

ひとまとまりの条件はかっこ（）でくくると分かりやすい。

```
select * from my_table where (field1 = 0) and (field2 = "hoge");
select * from my_table where (field1 = 0) or (field2 = "hoge");
```

order by：データの並べ替え

- order by [field-name] [ways-of-sort]; `order by 並べ替えたいカラム名 並べ方;`と書く。`order by`はSQL文の末尾に記述すれば良いため、`where`と併用可能。 **並べ方**
- asc 昇順 ascendの略。
- desc 降順 descendantの略

```
-- fieldの降順で表示
select * from my_table order by field desc;

-- whereとの組み合わせ
select * from my_table where field1 = "hoge" order by field2 desc;
```

limit：最大取得件数の指定

- limit [number]; `limit データの最大取得件数`と書く。 最大で何件のデータ（行数）を取得するかの指定には`limit`を用いる。 検索結果の上からしてされたデータの件数だけ取得する。 `limit`も`order by`と同様に、SQL文の末尾に記述することで、取得するデータの数（行数）を制限する。`order by`と組み合わせて大きい順にデータを必要な数だけ得ることもできる。(上位5位など)。ただし、`order by`と`limit`を併用する場合は、`limit`の方を末尾にする必要がある。

```
-- tableの上から5権を取得
select * from my_table limit 5;

-- whereとの組み合わせ
select * from my_table where field1 = "hoge" limit 5;

-- order by との組み合わせ priceが大きいもの3件を順に表示
select * from my_table order by price desc limit 3;
select * from my_table limit 3 order by price desc; -- Error
```

テーブルの作成

```
create table [table-name] (
    id int(11)
)
```

int()の()は整数値の表示幅

Lesson2

distinct：重複要素の除去

```
-- 特定のフィールドの重複要素の除去
select distinct(field1) from my_table;
```

四則演算

カラム名 [+-*/] 数字で四則演算ができる

```
-- 特定の行に四則演算
-- 商品名と税込表記を表示
select name, price * 1.08 from my_table;
-- 演算前のカラムを表示することも可能
select name, price, price * 1.08 from my_table;
```

集計関数

---

sum：カラムの合計値を求める

sum(カラム名)で合計値を計算できる。whereとの併用もできる。

```
-- sum
select price(sum) from my_table;
-- whereとの併用
-- Johnが使った金額の合計
select price(sum) from my_table where person = "John";
```

avg：カラムの平均を求める

`avg(カラム名)`でカラムの合計値を計算できる。`where`との併用も可能。

```
-- 合計と平均を求める
select sum(price), avg(price) from my_table;

-- where との併用
select sum(price), avg(price) from my_table where person = "John";
```

count：頻度を求める

`count(カラム名)`でカラムのデータ数を求められる。このとき**NULLはカウントされない**ため注意。NULLを含めたデータ数が欲しい時は`select count(*) from [table-name]`という`*`を使った書き方で全体のデータ数（行数・レコード数）を計算できる。よって両者を使えば特定のカラムにNULL値がどれくらい存在するかを調べられる。`where`との併用も可能

```
-- 特定のフィールドと、全体のデータ数（行数）
select count(field), count(*) from my_table;
```

max, min：最大・最小

`min(カラム名)`、`max(カラム名)`. whereとの併用可能。

```
-- Johnが買った中で一番高価なもの
select name, max(price) from my_table where person = "John";
```

group by：データのグループ化と集計

`group by カラム名`で、指定したカラムで、完全に同一のデータを持つレコード同士が同一のグループになる。SQL文の語尾につける。 集計関数である`sum`, `min`, `max`などと組み合わせて使える。 注意として

- group by を用いる場合、select文で使えるのは、**group byに指定しているカラム名**と**集計関数**のみ。

```

```

```
-- 良い例
-- personでグループ化して、それぞれの人の買い物の合計金額を算出
select sum(price), person from my_table group by person;
-- 集計して人ごとに集計して一番お金を使っている人上位5人を表示
select person, sum(price) from my_table group by person order by
sum(price) desc limit 5;
-- order by price とするとおかしな挙動になる

-- お金を使った回数が多い日を表示
select day_shopping, count(*) from my_table group by day_shopping;

-- ダメな例
-- person でグループ化しているのでpersonはunique状態。
-- しかしpriceは全体を表す。
-- よってこれはエラーになる。
select price, person from my_table group by person;
```

group by(where)：細かい条件でデータをグループ化

検索とグループ化の手順

whereとgroup byと集計関数は

実行の順序
検索：where ↓ グループ化：group by ↓ 関数：count, sum, avg, max, min

の順で実行されていく。 `where`でデータを検索し、グループ化するデータを定めてから`group by`と覚えると良い。

書き方

```
-- 買い物した日でグループ化して、日ごとの買い物の合計金額を算出
select sum(price), day_shopping from my_table group by day_shopping;

-- まず、where でJohnのデータのみに絞る。そのあと日ごとの買い物の合計金額を算出
select sum(price), day_shopping from my_table where person = "John" group
by day_shopping;
```

having：グループ化したデータのさらなる絞り込み

グループ化により、日毎の買い物の金額を得ることができた。 その中でも、1000円以上使った日などをさらに絞り込みたい。このときは`where`ではなく`having`というSQLを用いる。 `group by カラム名 having 条件`と書き、条件を満たすグループを取得することができる。先ほどの実行の順序で説明した通り、`where`は一番先に実行されるため、`group by`した後のデータを絞り込むことはできない。`where`はあくまでも全体からの絞り込みに使う。

- 実行の順序

検索：where ↓ グループ化：group by ↓ 関数：count, sum, avg, max, min ↓ having

- whereとhavingの検索対象の違い

where	having
テーブル全体が検索対象	グループ化されたデータが検索対象

- havingの注意点 havingはグループ化された後のテーブルから検索するため、条件文で使うカラムは必ず**グループ化されたテーブルのカラム**を使う。

```
-- グループ化されたカラムを使う
select sum(price), person from my_table group by person having sum(price)
> 1000;
-- sum(price) > 1000 のところを price > 1000とするとおかしな表示になる。必ずグループ化されたカラムを使うこと
```

lesson3

サブクエリ：2つのSQL文を使って見る

SQL文の中に他のSQL文を入れることができる。この他のSQL文をサブクエリという。

```
query (subquery)
サブクエリが実行された後外側にあるSQL文が実行される。
サブクエリにおいてはセミコロンはいらない。
```

as:カラム名の指定

```
select goals as "ウィルの得点数" from players
where name = "ウィル";
```

countriesテーブル

テーブルを紐づける

テーブルを紐づけるために、外部キーと主キーを使う。外部キーで他のテーブルにある主キーを指定することで、テーブル同士を紐づけることができる。 関連するテーブル間の整合性を保ちたい列に設定する外部キーを設定する。 外部キーを設定して参照する側を子テーブル、 設定元は親テーブルという。

紐づけるための手順

親テーブルに主キーを設定 子テーブルに親テーブルの主キーに合わせた列を作る。 それを外部キーとして設定する。

主キー(PRIMARY KEY)と外部キー(FOREIGN KEY)

テーブルに登録するレコード（データ行）の全体のうち、一つのデータに特定することをデータベースが保証する列。主キーによって特定のデータを特定するために、主キーは唯一でなければならない。同じデータは登録できない。NULLもダメ。 外部キーは、関連のあるテーブル間の整合性をデータベースに保証させるために設定する。外部キーを使うと、間違ったデータの登録や消去を防いだり、関連するデータを一括に変更できる。

補足：作成済みのテーブルにカラムを追加する方法(add)

alter table [table-name] add [column-name] [column-definition];

場所を指定(先頭) firstをつける alter table [table-name] add [column-name] [column-definition] first;

場所を指定(特定のカラムの後) `after [column-name]`をつける alter table [table-name] add [column-name] [column-definition] after height;

指定しなければ最後に挿入。

ex)

```
alter table players add country_id int(11) not null after height;
-- デフォルトの値が勝手に格納される。
```

補足2：作成済みのテーブルから指定のカラムを消去(drop)

alter table [table-name] drop column [column-name];

既存のテーブルに主キーと外部キーの追加

主キーの追加 alter table [table-name] add primary key([column-name]); 主キーの消去 alter table [table-name] drop primary key;

外部キーの追加 alter table [child-table] add foreign key([column-name]) references parent-table;

```
-- playersテーブルに外部キーの追加
alter table players add foreign key(country_id) references countries(id);
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    |                |
| country_id | int(11)     | NO   | MUL | NULL    |                |
+------------+-------------+------+-----+---------+----------------+
5 rows in set (0.01 sec)
*/
```

keyのとこが変わってる

join：テーブルの結合

複数のテーブルを一つに結合したい時に使う。 結合したテーブルは1つのテーブルとしてデータを取得することができる。 select * from tableA join tableB on [結合条件];

結合条件は、「ON テーブル名.カラム名 = テーブル名.カラム名」 ex) select * from players join countries on players.country_id = countries.id;

値が共通しているカラムを元に結合する

joinの実行順序 joinを含むSQL文では、はじめにjoinが実行される。 その次に、結合されたテーブルに対してselectが実行される。 onによる条件をつけないと、 それぞれのテーブルの行の数を掛け合わせた数の行が生成される。 (上の例だと(playersの行数)×(countriesの行数)) fromで指定したテーブルの1つの行に対して、joinで指定したテーブルの行全て結合される。

```
select * from players join countries on players.country_id = countries.id;
できた
```

fromで指定されたテーブルが基準となって表示される。 その後onで指定した条件を元にテーブルが結合される。

複数テーブルでのカラムの指定

複数のテーブルに同じカラム名が存在するときは、 「table.column」で指定する必要がある。 どのテーブルのカラムなのかを明示するのである。 ex)

```
select name, name  -- nameがかぶっていて
from players
join countries
on players.country_id = countries.id;
-- ERROR 1052 (23000): Column 'name' in field list is ambiguous
-- nameというフィールド名は曖昧ですというERRORが出る
```

続いて明示的にテーブル名をしてすると

```
select players.name, countries.name  -- nameがかぶっていて
from players
join countries
on players.country_id = countries.id;
-- ちゃんと表示される
```

SQL：全体の実行順序の確認

テーブルの指定：from 結合：on・join 取得条件：where グループ化：group by 関数：count/sum/avg/min HAVING：having（group byした後の絞り込みに使う） 検索：select/distinct 順序：order by LIMIT：limit

の順で行われる。 SQLは、取得するテーブルを形成してから検索を行うため、FROM・JOINが先に行われることを覚えてくこと。

join(2)：NULLのレコードが存在するテーブルのjoin

外部キーにNULLがぞんざいするときは、 NULLのレコードは飛ばされて実行結果に反映されない。

ちなみに、 load data local infile で、csvファイルがNULLのところには、 NULLと入っていないと認識されず、 その行は取り込まれない。 そのため、`bulk insert`を使うか、 前処理でcsv側に`NULL`とかくかを行なって無理やりSQL側にNULLを認識させる必要がある。

left join：外部キーがNULLの場合

外部キーがNULLのレコードもJOINで取得したいときは、`left join`を使う

- **leftなし select * from players join teams on players.previous_team_id = teams.id;**
- **leftあり select * from players left join teams on players.previous_team_id = teams.id;**

違いを見てみるのが早い

3つ以上のテーブルの結合

joinを二つ以上繰り返せば良い

```
select * from table1
join table2 on condition
join table3 on condition;
```

table1, 2, 3の順に(2, 3はjoinを書いた順)出力される。もちろん、leftとも組み合わせて

```
select * from table1
join table2 on condition
left join table3 on condition;
```

もできる ex)

```
select * from players
join countries on players.country_id = countries.id
left join teams on players.previous_team_id = teams.id;
```

truncate構文：テーブルの消去・再作成

テーブルの内容を消去するときに、`delete from table;`でやった。しかしこれよりも早くtableの中身を消して再作成する方法がある。`truncate table [table-name]` テーブルを消して、テーブルの設定をそのままに、もう一度作る。

```
show tables;
/*
+--------------------+
| Tables_in_scraping |
+--------------------+
| cities             |
| countries          |
| players            |
| teams              |
+--------------------+
4 rows in set (0.01 sec)
*/

truncate table players;
--Query OK, 0 rows affected (0.12 sec)

show tables;
/*
+--------------------+
| Tables_in_scraping |
+--------------------+
| cities             |
| countries          |
| players            |
| teams              |
+--------------------+
4 rows in set (0.00 sec)
*/
 describe players;
 /* テーブルの設定は消されていない
+------------------+-------------+------+-----+-------------------+-------
---------+
| 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              |
|
| country_id       | int(11)     | NO   | MUL | NULL              |
|
| previous_team_id | int(11)     | YES  | MUL | NULL              |
|
```

```
| created_at       | timestamp   | NO   |     | CURRENT_TIMESTAMP |
|
+------------------+-------------+------+-----+-------------------+-------
---------+
7 rows in set (0.00 sec)
*/
```

create user まわり

```
create user scraper@localhost identified by "password";
-- ERROR:Your password does not satisfy the current policy requirements
-- passwordっていうパスワードはポリシーを満たさなかった
-- 縛りをゆるくする必要がある

-- グローバル変数の参照
show global variables like "validate%";
/*
+--------------------------------------+-------+
| Variable_name                        | Value |
+--------------------------------------+-------+
| validate_password_check_user_name    | ON    |
| validate_password_dictionary_file    |       |
| validate_password_length             | 8     |
| validate_password_mixed_case_count   | 1     |
| validate_password_number_count       | 1     |
| validate_password_policy             | MEDIUM|
| validate_password_special_char_count | 1     |
+--------------------------------------+-------+
*/
set global validate_password_policy = LOW;
show global variables like "validate%";
/* LOWになってる
+--------------------------------------+-------+
| Variable_name                        | Value |
+--------------------------------------+-------+
| validate_password_check_user_name    | ON    |
| validate_password_dictionary_file    |       |
| validate_password_length             | 8     |
| validate_password_mixed_case_count   | 1     |
| validate_password_number_count       | 1     |
| validate_password_policy             | LOW   | ←←← 変更
| validate_password_special_char_count | 1     |
+--------------------------------------+-------+
*/
-- もう一度
create user scraper@localhost identified by "password";
-- Query OK, 0 rows affected (0.18 sec) 成功！
```

続いてユーザーscraperにデータベースscrapingを読み書き可能な権現を与える。これをしないとscraperはデータベースに対して何も操作できない。

参考

- **How To Create a New User and Grant Permissions in MySQL https://www.digitalocean.com/community/tutorials/how-to-create-a-new-user-and-grant-permissions-in-mysql**

```
-- データベースscrapingにおける全ての権現をユーザーscraperに与える
grant all on scraping.* to scraper;
-- もしすべてのデータベースを操作できる権現を与えたかったら、
grant all on *.* to scraper;
-- と書く。 一つ目の*は全てのデータベース、2つ目の*は全てのテーブルを指す。よって
scraper.* と書くと、scraperというデータベースの全てのテーブルを操作する権現を与えます
よということ。


-- ERROR 1410 (42000): You are not allowed to create a user with GRANT だめ
だった
-- GRANT付きのユーザーをCREATE USERすることは許されていない、って意味。
-- userテーブルの確認
select user, host from user;
/*
+------------------+-----------+
| user             | host      |
+------------------+-----------+
| mysql.infoschema | localhost |
| mysql.session    | localhost |
| mysql.sys        | localhost |
| root             | localhost |
| scraper          | localhost |
+------------------+-----------+
*/
```

ルートにgrantを持ったユーザーを作る権現がないらしい。 ルートのgrantテーブルに何があるか確認してみる。これでrootの持つ権現をみることができる。

```
show grants for root@localhost;  -- ユーザ権現を見てみる
show grants for root@localhost\G  -- ユーザ権現を見てみる
/*
*************************** 1. row ***************************
Grants for root@localhost: GRANT SELECT, INSERT, UPDATE, DELETE, CREATE,
DROP, RELOAD, SHUTDOWN, PROCESS, FILE, REFERENCES, INDEX, ALTER, SHOW
DATABASES, SUPER, CREATE TEMPORARY TABLES, LOCK TABLES, EXECUTE,
REPLICATION SLAVE, REPLICATION CLIENT, CREATE VIEW, SHOW VIEW, CREATE
ROUTINE, ALTER ROUTINE, CREATE USER, EVENT, TRIGGER, CREATE TABLESPACE,
CREATE ROLE, DROP ROLE ON *.* TO `root`@`localhost` WITH GRANT OPTION
*************************** 2. row ***************************
Grants for root@localhost: GRANT
```

```
BACKUP_ADMIN,BINLOG_ADMIN,CONNECTION_ADMIN,ENCRYPTION_KEY_ADMIN,GROUP_REPL
ICATION_ADMIN,PERSIST_RO_VARIABLES_ADMIN,REPLICATION_SLAVE_ADMIN,RESOURCE_
GROUP_ADMIN,RESOURCE_GROUP_USER,ROLE_ADMIN,SET_USER_ID,SYSTEM_VARIABLES_AD
MIN,XA_RECOVER_ADMIN ON *.* TO `root`@`localhost` WITH GRANT OPTION
*************************** 3. row ***************************
Grants for root@localhost: GRANT PROXY ON ''@'' TO 'root'@'localhost' WITH
GRANT OPTION
3 rows in set (0.00 sec)
*/

-- 念のため作成したユーザーscraperの権現も見ておく
show grants for scraper@localhost;
/* ほぼ何もできない状態
+---------------------------------------------+
| Grants for scraper@localhost                |
+---------------------------------------------+
| GRANT USAGE ON *.* TO `scraper`@`localhost` |
+---------------------------------------------+
1 row in set (0.00 sec)
*/
```

rootの権現がこれだと見にくいので、db:mysqlのuserテーブルからrootユーザーの情報を引っ張ってくる。
hoge_priv(privilegeの略)のところがrootの持つ権現である。 参考リンク

- **https://stackoverflow.com/questions/8484722/access-denied-for-user-rootlocalhost-while-attempting-to-grant-privileges**

```
-- root
select * from mysql.user where user = "root"\G
-- select * from information_schema.user_privileges;でも見える
/*
*************************** 1. row ***************************
                  Host: localhost
                  User: root
           Select_priv: Y
           Insert_priv: Y
           Update_priv: Y
           Delete_priv: Y
           Create_priv: Y
             Drop_priv: Y
           Reload_priv: Y
         Shutdown_priv: Y
          Process_priv: Y
             File_priv: Y
            Grant_priv: Y
       References_priv: Y
            Index_priv: Y
            Alter_priv: Y
          Show_db_priv: Y
            Super_priv: Y
 Create_tmp_table_priv: Y
```

```
       Lock_tables_priv: Y
           Execute_priv: Y
        Repl_slave_priv: Y
       Repl_client_priv: Y
       Create_view_priv: Y
         Show_view_priv: Y
    Create_routine_priv: Y
     Alter_routine_priv: Y
       Create_user_priv: Y
             Event_priv: Y
           Trigger_priv: Y
Create_tablespace_priv: Y
               ssl_type:
             ssl_cipher:
            x509_issuer:
           x509_subject:
         max_questions: 0
           max_updates: 0
       max_connections: 0
  max_user_connections: 0
                plugin: caching_sha2_password
DY_^Eyf?eB[j84FuHqMjvIEJHn8cM1ln6geM7lHjQYLPcVYJX9C.m85
      password_expired: N
 password_last_changed: 2018-06-21 11:07:04
     password_lifetime: NULL
        account_locked: N
      Create_role_priv: Y
        Drop_role_priv: Y
Password_reuse_history: NULL
   Password_reuse_time: NULL
1 row in set (0.00 sec)
*/
```

全部の権現あるやないかい。なんでgrant持ったユーザー作れないんだよ。 続いてもうわかっているかもしれないが、ユーザーscraperの権現も見ておく。 何も権現を与えていないため、すべての権現`hoge_priv`が`N`(No)になっている。

```
select * from mysql.user where user = "scraper"\G
/*
*************************** 1. row ***************************
                  Host: localhost
                  User: scraper
           Select_priv: N
           Insert_priv: N
           Update_priv: N
           Delete_priv: N
           Create_priv: N
             Drop_priv: N
           Reload_priv: N
         Shutdown_priv: N
          Process_priv: N
```

```
              File_priv: N
             Grant_priv: N
        References_priv: N
             Index_priv: N
             Alter_priv: N
           Show_db_priv: N
             Super_priv: N
  Create_tmp_table_priv: N
       Lock_tables_priv: N
           Execute_priv: N
        Repl_slave_priv: N
       Repl_client_priv: N
       Create_view_priv: N
         Show_view_priv: N
    Create_routine_priv: N
     Alter_routine_priv: N
       Create_user_priv: N
             Event_priv: N
           Trigger_priv: N
Create_tablespace_priv: N
               ssl_type:
             ssl_cipher:
            x509_issuer:
           x509_subject:
          max_questions: 0
            max_updates: 0
        max_connections: 0
   max_user_connections: 0
                 plugin: caching_sha2_password
 authentication_string: $A$005$aL:&87:<{?
h4cmGIJxOlIIFGhpFMmG7XH5HJP015REdEKhyV83CvQMIN/
       password_expired: N
 password_last_changed: 2018-09-06 00:23:30
     password_lifetime: NULL
        account_locked: N
       Create_role_priv: N
         Drop_role_priv: N
Password_reuse_history: NULL
   Password_reuse_time: NULL
1 row in set (0.00 sec)
*/
```

なぜ全ての権現を持っているはずのrootが、grantを付与したユーザーを作成することができないのか。調べて見たところ、以下のことがわかった。

> **Notice how the output of**
>
> ```
> SHOW GRANTS FOR 'root'@'localhost';
> ```

> **did not say 'ALL PRIVILEGES' but had to spell out what root@localhost has. GRANT ALL PRIVILEGES will fail, because a user can not grant what he/she does not have, and the server seem to think something is not here ...**

どうやら

```
show grants for root@localhost;
```

は全ての権現を表しているわけではないらしい。

```
grant all on scraper.* to scraper@localhost;
-- Query OK, 0 rows affected (0.05 sec)  なんかできたんだが
flush privileges;  -- 権現を更新
select * from mysql.user where user = "scraper"\G
/* 変わってないじゃねーか！！！
*************************** 1. row ***************************
                  Host: localhost
                  User: scraper
           Select_priv: N
           Insert_priv: N
           Update_priv: N
           Delete_priv: N
           Create_priv: N
             Drop_priv: N
           Reload_priv: N
         Shutdown_priv: N
          Process_priv: N
             File_priv: N
            Grant_priv: N
       References_priv: N
            Index_priv: N
            Alter_priv: N
          Show_db_priv: N
            Super_priv: N
 Create_tmp_table_priv: N
      Lock_tables_priv: N
          Execute_priv: N
       Repl_slave_priv: N
      Repl_client_priv: N
      Create_view_priv: N
        Show_view_priv: N
   Create_routine_priv: N
    Alter_routine_priv: N
      Create_user_priv: N
            Event_priv: N
          Trigger_priv: N
Create_tablespace_priv: N
              ssl_type:
            ssl_cipher:
           x509_issuer:
```

```
          x509_subject:
         max_questions: 0
           max_updates: 0
       max_connections: 0
  max_user_connections: 0
                plugin: caching_sha2_password
 authentication_string: $A$005$aL:&87:<{?
h4cmGIJxOlIIFGhpFMmG7XH5HJP015REdEKhyV83CvQMIN/
      password_expired: N
 password_last_changed: 2018-09-06 00:23:30
     password_lifetime: NULL
        account_locked: N
      Create_role_priv: N
        Drop_role_priv: N
Password_reuse_history: NULL
   Password_reuse_time: NULL
1 row in set (0.00 sec)
*/
--grantsを見てみる
show grants for scraper@localhost;
/*
+---------------------------------------------------------------+
| Grants for scraper@localhost                                  |
+---------------------------------------------------------------+
| GRANT USAGE ON *.* TO `scraper`@`localhost`                   |
| GRANT ALL PRIVILEGES ON `scraper`.* TO `scraper`@`localhost` |
+---------------------------------------------------------------+
2 rows in set (0.00 sec)
*/
```

なんかできてしまった。

```
grant all on *.* to scraper@localhost;
--Query OK, 0 rows affected (0.01 sec)
flush privileges;
-- Query OK, 0 rows affected (0.00 sec)

show grants for scraper@localhost\G
/* なんかできてしまった。あのエラーはなんだったんだ。
*************************** 1. row ***************************
Grants for scraper@localhost: GRANT SELECT, INSERT, UPDATE, DELETE,
CREATE, DROP, RELOAD, SHUTDOWN, PROCESS, FILE, REFERENCES, INDEX, ALTER,
SHOW DATABASES, SUPER, CREATE TEMPORARY TABLES, LOCK TABLES, EXECUTE,
REPLICATION SLAVE, REPLICATION CLIENT, CREATE VIEW, SHOW VIEW, CREATE
ROUTINE, ALTER ROUTINE, CREATE USER, EVENT, TRIGGER, CREATE TABLESPACE,
CREATE ROLE, DROP ROLE ON *.* TO `scraper`@`localhost`
*************************** 2. row ***************************
Grants for scraper@localhost: GRANT
BACKUP_ADMIN,BINLOG_ADMIN,CONNECTION_ADMIN,ENCRYPTION_KEY_ADMIN,GROUP_REPL
ICATION_ADMIN,PERSIST_RO_VARIABLES_ADMIN,REPLICATION_SLAVE_ADMIN,RESOURCE_
GROUP_ADMIN,RESOURCE_GROUP_USER,ROLE_ADMIN,SET_USER_ID,SYSTEM_VARIABLES_AD
MIN,XA_RECOVER_ADMIN ON *.* TO `scraper`@`localhost`
```

```
*************************** 3. row ***************************
Grants for scraper@localhost: GRANT ALL PRIVILEGES ON `scraper`.* TO
`scraper`@`localhost`
3 rows in set (0.00 sec)
*/

-- できましたー
show databases;
/*
+--------------------+
| Database           |
+--------------------+
| information_schema |
| scraping           |
+--------------------+
2 rows in set (0.00 sec)
*/
select current_user();
/*
+------------------+
| current_user()   |
+------------------+
| scraper@localhost |
+------------------+
1 row in set (0.00 sec)
*/
```

drop table if exists cities でエラーが出た話

PythonのMySQLdbから扱うとエラー（じゃなくて警告だった、その後の処理はできる）になる

```
drop table if exists cities
-- Warning: (1051, "Unknown table 'scraping.cities'")
```

ターミナルから直接起動してやるとエラーにならない

```
drop table if exists cities;
-- Query OK, 0 rows affected, 1 warning (0.00 sec)
```

Python では対応していないのだろうか？ とりあえず例外処理で逃して、それ以降の処理を行うことにした。

テーブルを作る時は、`create`文のテーブル名とカラム名をバッククオートで必ず囲むこと!!!!!!!!!!!!!!!!!!！そうじゃないとError：1064がでてくる

insert into cities () values (); ができなかった話

```
-- version の確認
select version();
/*
+-----------+
| version() |
+-----------+
| 8.0.11    |
+-----------+
1 row in set (0.00 sec)
*/

-- まずはcitiesのカラムの確認
describe cities;
-- show columns from cities; でもいける
/*
+------------+-------------+------+-----+-------------------+------------
----+
| Field      | Type        | Null | Key | Default           | Extra
|
+------------+-------------+------+-----+-------------------+------------
----+
| id         | int(5)      | NO   | PRI | NULL              |
auto_increment |
| rank       | varchar(100) | NO   |     | NULL              |
|
| city       | varchar(100) | NO   |     | NULL              |
|
| population | int(10)     | NO   |     | NULL              |
|
| visit      | int(5)      | NO   |     | 0                 |
|
| created_at | timestamp   | NO   |     | CURRENT_TIMESTAMP |
|
+------------+-------------+------+-----+-------------------+------------
----+
*/

-- table が正常に動いてない？ 空行を入れてみる
insert into cities () values();
-- >> ERROR 1364 (HY000): Field 'rank' doesn't have a default value
-- 正常には動いているみたい？

-- insertしてみる
insert into cities (rank) values("B");

-- ERROR 1064 (42000): You have an error in your SQL syntax; check the
manual that corresponds to your MySQL server version for the right syntax
to use near 'rank) values("B")' at line 1

--
insert into cities (rank, city, population, visit) values("B", "Tokyo",
10000000, 1);
```

rankが予約語だったーーーーーーーーーーーーーーーーーーーーーーーー 予約語をカラム名にすると文法エラーになる。本当に気をつけましょう。。。。（戒め）

Pythonの警告処理

例外処理ではつかまらない。下記コードのようにtry, exeptionで捕まえたかったらwarningでやる必要がある。

その他

現在のユーザーを見る

```
select user();
/*
+----------------+
| user()         |
+----------------+
| root@localhost |
+----------------+
1 row in set (0.00 sec)
*/
select current_user();
/*
+----------------+
| current_user() |
+----------------+
| root@localhost |
+----------------+
1 row in set (0.00 sec)
*/

-- 現在のMySQLのバージョン
select version();
/*
+-----------+
| version() |
+-----------+
| 8.0.11    |
+-----------+
1 row in set (0.00 sec)
*/
```

`use [database-name];`**で移動しなくても（移動した方が楽だけど）、** `select * from dbname.table-name;`**で.(ドット)メソッドみたいな感じで特定のデータベースのテーブルにアクセスできる。**